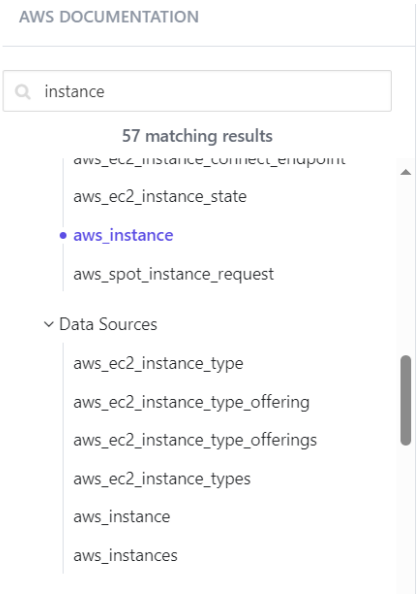
AKSHIT MITTAL
10/12/23

CLOUD & DEVOPS

EC2 Instance creation on AWS via Terraform

We have connected the Terraform to our AWS machine. Now we will create the Infrastructure, through Terraform on our CLI.
Here we will create EC2 instances virtual servers on our AWS using AMIs images
Links :-
[aws_instance | Resources | hashicorp/aws | Terraform | Terraform Registry]

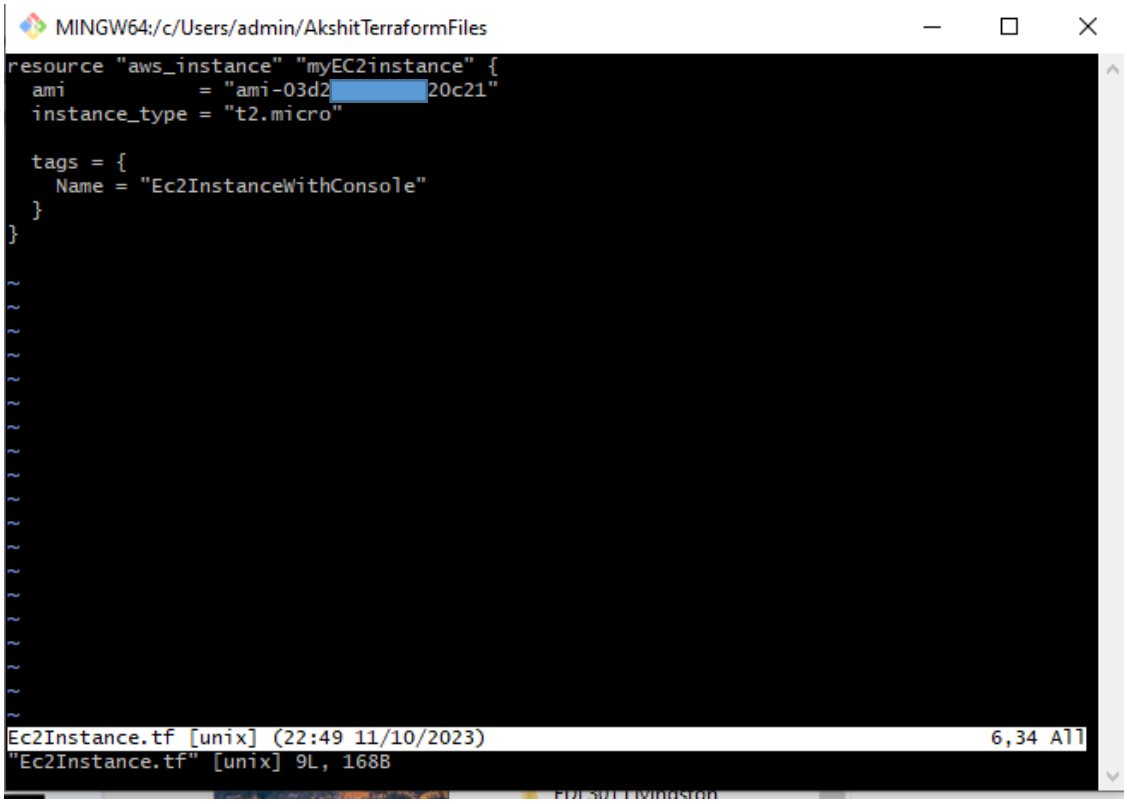
1. In the Terraform Resources section of HashiCorp documentation you will find the syntax of code used to provide basic configuration of the new EC2 instance you want to create i.e AMI id, Instance type, Name of the virtual server.
   Open the Terraform root folder which was previously created → Write this code into the separate file or the same file as AWSProvider_Configuration, filling all the details



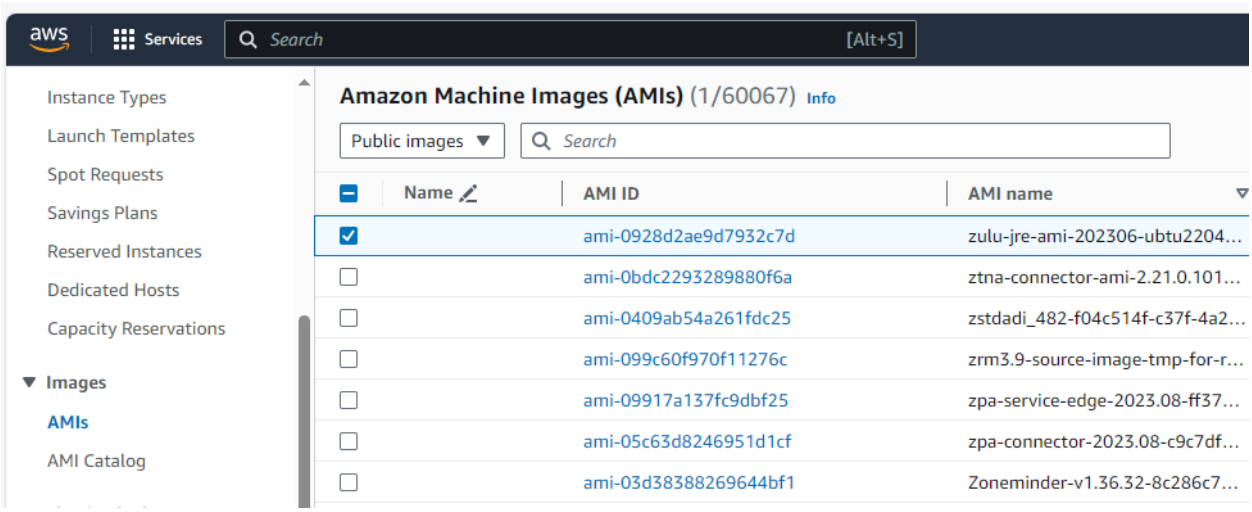

```
    name   = "virtualization-type"
    values = ["hvm"]
  }

  owners = ["099720109477"] # Canonical
}

resource "aws_instance" "web" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t3.micro"

  tags = {
    Name = "HelloWorld"
  }
}
```

Spot instance example

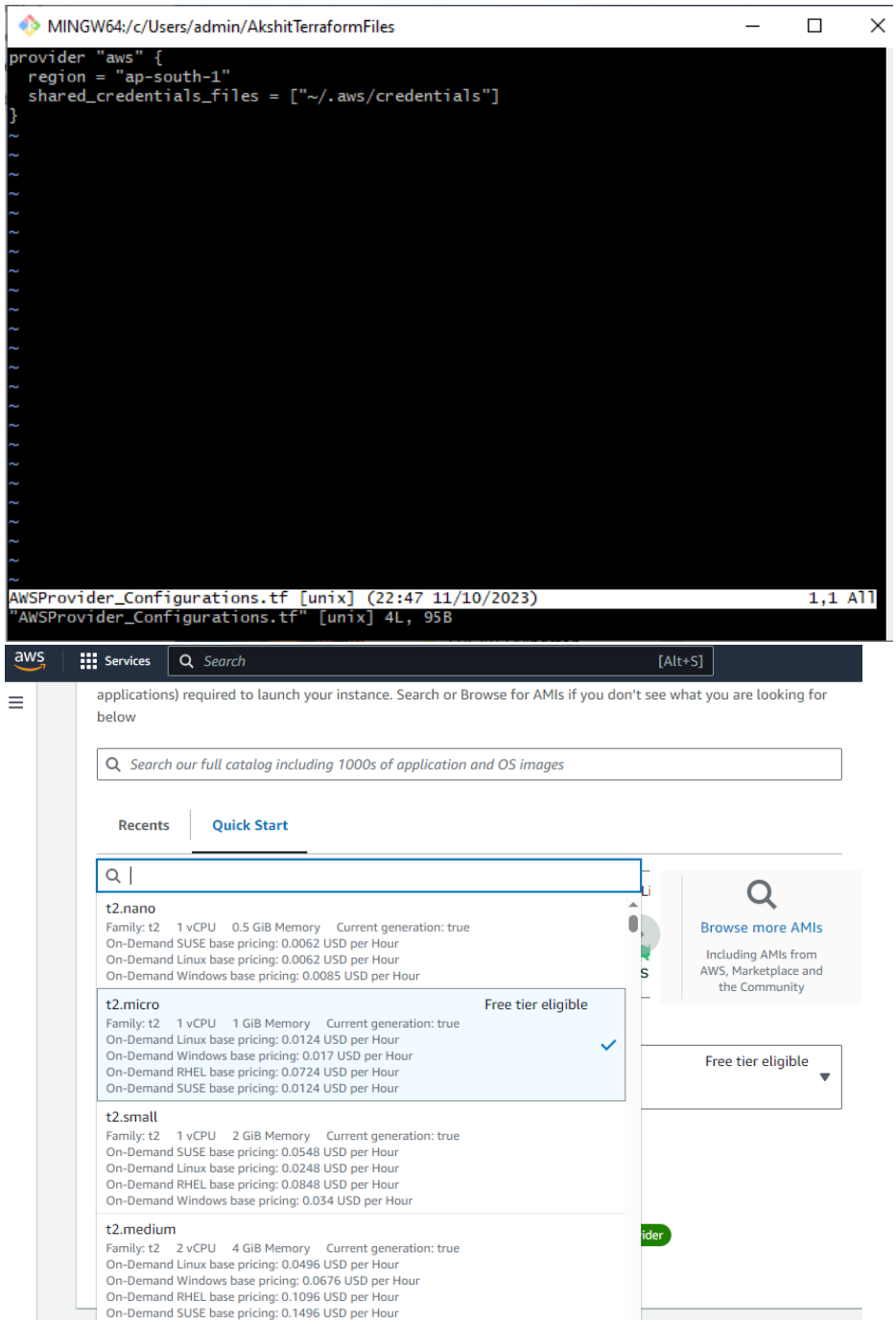
```
data "aws_ami" "this" {
  most_recent = true
```



2. You will find the AMI ID from AWS AMI Tab. You can find it in the public images with the configurations you need for your virtual machine.



3. Please make sure the AWS region for the AMI id is same as specified in the AWSProvider_Configuration file OR You may specify the region of AMI id separately, in the configuration file as shown in the screenshot → The instance_type should be specified according to what is needed (t2.micro is what we have used)→ Name section in tags specifies the name of your virtual machine you want to provide.

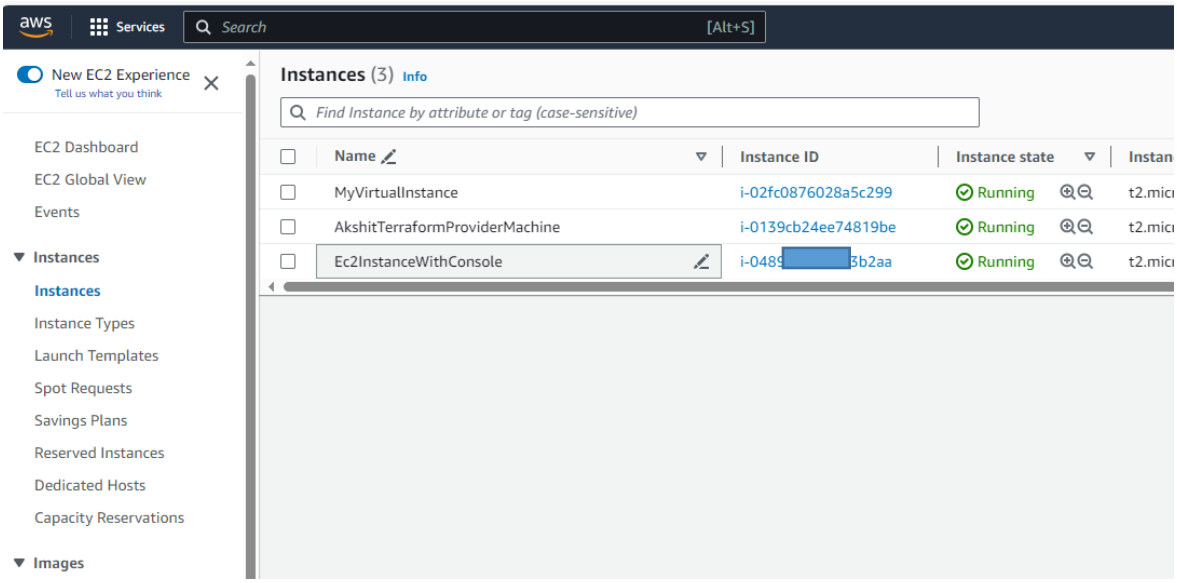


4. Terraform Plan command generates execution plan that shows the changes that Terraform will make to the Infrastructure
   Terrafrom Apply command to apply the changes defined in your terraform configurations. It will create or modify infrastructure.

MINGW64:/c/Users/admin/AkshitTerraformFiles

```
$ vim Ec2Instance.tf

admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ terraform apply
aws_instance.myEC2instance: Refreshing state... [id=i-0489ac1cb0c33b2aa]

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_instance.myEC2instance will be updated in-place
  ~ resource "aws_instance" "myEC2instance" {
        id                                   = "i-0489ac1cb0c33b2aa"
      ~ tags                                 = {
          ~ "Name" = "AkshitEC2machine1" -> "Ec2InstanceWithConsole"
        }
      ~ tags_all                             = {
          ~ "Name" = "AkshitEC2machine1" -> "Ec2InstanceWithConsole"
        }
        # (30 unchanged attributes hidden)

        # (8 unchanged blocks hidden)
    }

Plan: 0 to add, 1 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_instance.myEC2instance: Modifying... [id=i-0489ac1cb0c33b2aa]
aws_instance.myEC2instance: Modifications complete after 1s [id=i-0489ac1cb0c33b2aa]

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

5. Virtual Machine on EC2 instance named – Ec2InstanceWithConsole, is successfully created on the EC2 instances that will be visible and in the running status.



6. Terraform Destroy Command- Destroy or will Terminate the Running instances which was created through terraform.
   The Instance which was previously created, will turn to the Terminated state when seen in AWS EC2 console.

MINGW64:/c/Users/admin/AkshitTerraformFiles

```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ terraform destroy
aws_instance.myEC2instance: Refreshing state... [id=i-0489ac1cb0c33b2aa]

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_instance.myEC2instance will be destroyed
  - resource "aws_instance" "myEC2instance" {
      - ami                                  = "ami-03d294e37a4820c21" -> null
      - arn                                  = "arn:aws:ec2:ap-south-1:346357902589:in
nce/i-0489ac1cb0c33b2aa" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1a" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-0489ac1cb0c33b2aa" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - monitoring                           = false -> null
      - placement_partition_number           = 0 -> null
      - primary_network_interface_id         = "eni-0382fd916ca591dd1" -> null
      - private_dns                          = "ip-172-31-39-197.ap-south-1.compute.in
nal" -> null
      - private_ip                           = "172.31.39.197" -> null
      - public_dns                           = "ec2-3-111-219-36.ap-south-1.compute.am
naws.com" -> null
      - public_ip                            = "3.111.219.36" -> null
      - secondary_private_ips                = [] -> null
```

aws | Services | Search [Alt+S]

New EC2 Experience
Tell us what you think

EC2 Dashboard
EC2 Global View
Events

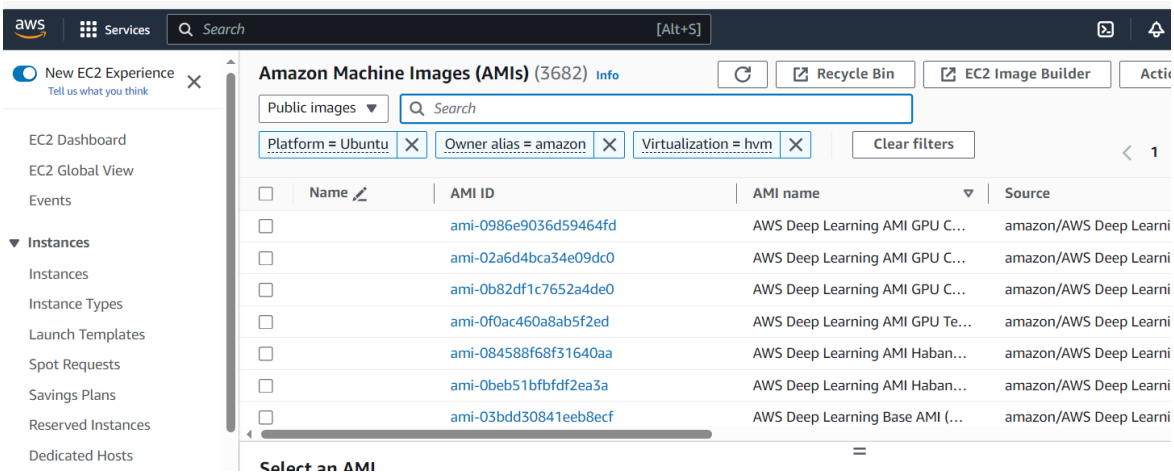
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts

Instances (3) Info

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state
MyVirtualInstance	i-02fc0876028a5c299	Running
AkshitTerraformProviderMachine	i-0139cb24ee74819be	Running
Ec2InstanceWithConsole	i-0489ac1cb0c33b2aa	Terminated

7. We can Perform this task by giving AMI ID attributes instead of directly providing an AMI ID. As we can see there are various filters we can set up in AWS AMIs section, filtering out the AMI IDs based on those. Similar filter we can set up in the terraform configuration file instead of giving an ID directly, and the terraform will automatically pick up the AMI id based on those attributes/filters.



8. The syntax of the attributes setup for AMI Ids can be found from the HashiCorp Terrafrom AWS documentation itself. Here,
most_recent attribute signifies that terraform will pick up the latest AMI based on the attributes
In the name filter → in values, we have provided * after a name, which signifies that the AMI who's name are based upon that prefix should be picked up.
Virtualization-type filter is also provided. You may also provide other filters based on the requirements.
The resource tab will be same as previous, only the AMI section will change in which we will provide the reference to the AMI id instead of the AMI id itself.
→ Save the file

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
- App Mesh
- App Runner
- AppConfig
- AppFlow
- AppIntegrations

Example Usage

Basic example using AMI lookup

```
data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"]
  }

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

  owners = ["099720109477"] # Canonical
}

resource "aws_instance" "web" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t3.micro"
```

MINGW64:/c/Users/admin/AkshitTerraformFiles

```
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 = "Ec2InstanceWithConsole"
  }
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
Ec2Instance.tf [unix] (09:03 13/10/2023)
"Ec2Instance.tf" [unix] 22L, 361B
```

9. Terraform apply command will create the infrastructure for virtual EC2 instance successfully. With same configurations provided in the File.
   Please Note, status after terraform apply result command is showing as 1 deletion of instance because I created same instance previously with same name for testing Purpose. Hence it shows, that we can't have 2 instances with same name or the previous instance will be automatically terminated.

```
          + "Name" = "Ec2InstanceWithConsole"
        }
      + tenancy                              = (known after apply)
      + user_data                            = (known after apply)
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = false
      + vpc_security_group_ids               = (known after apply)
    }

Plan: 1 to add, 0 to change, 1 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_instance.myec2: Destroying... [id=i-07dfcba22ce866d51]
aws_instance.myec2Instance: Creating...
aws_instance.myec2: Still destroying... [id=i-07dfcba22ce866d51, 10s elapsed]
aws_instance.myec2Instance: Still creating... [10s elapsed]
aws_instance.myec2Instance: Still creating... [20s elapsed]
aws_instance.myec2: Still destroying... [id=i-07dfcba22ce866d51, 20s elapsed]
aws_instance.myec2: Still destroying... [id=i-07dfcba22ce866d51, 30s elapsed]
aws_instance.myec2Instance: Still creating... [30s elapsed]
aws_instance.myec2: Destruction complete after 31s
aws_instance.myec2Instance: Creation complete after 32s [id=i-0f0bf1e3dc8ce4f20]

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

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