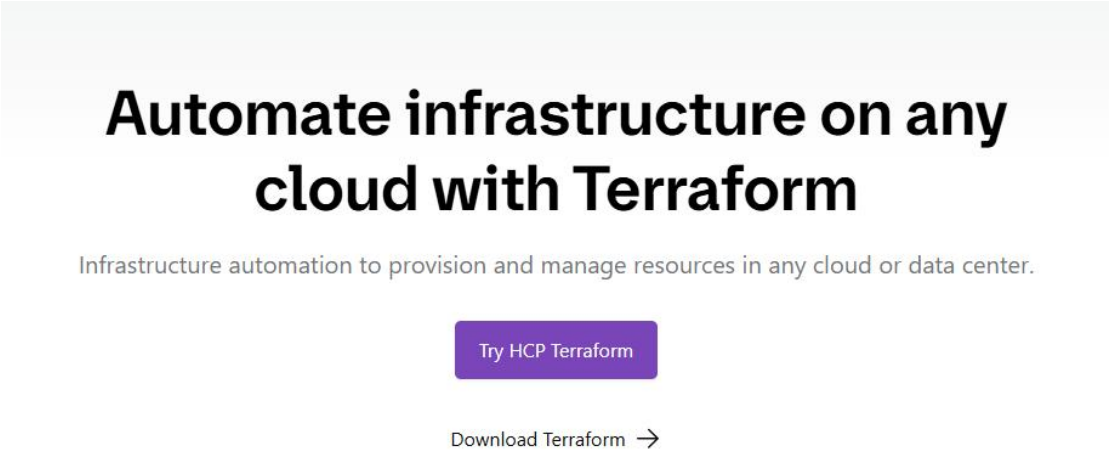
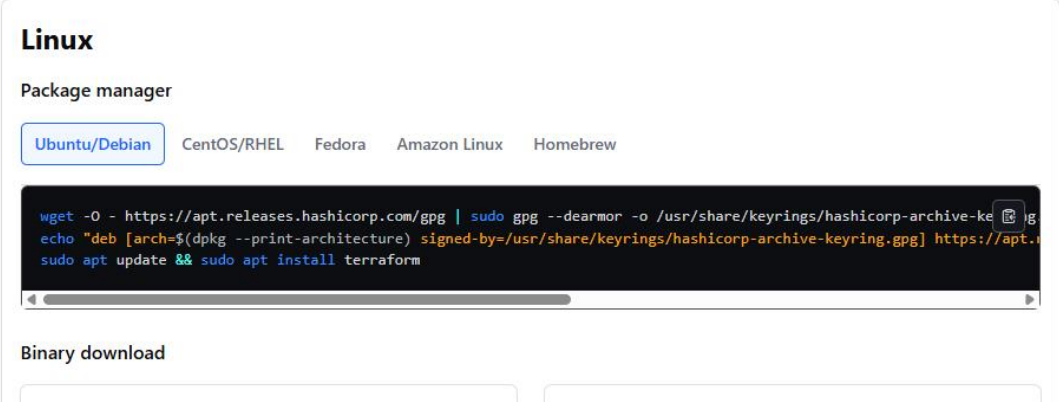
Terraform

https://www.terraform.io/



On Linux run this command, Terraform will be installed



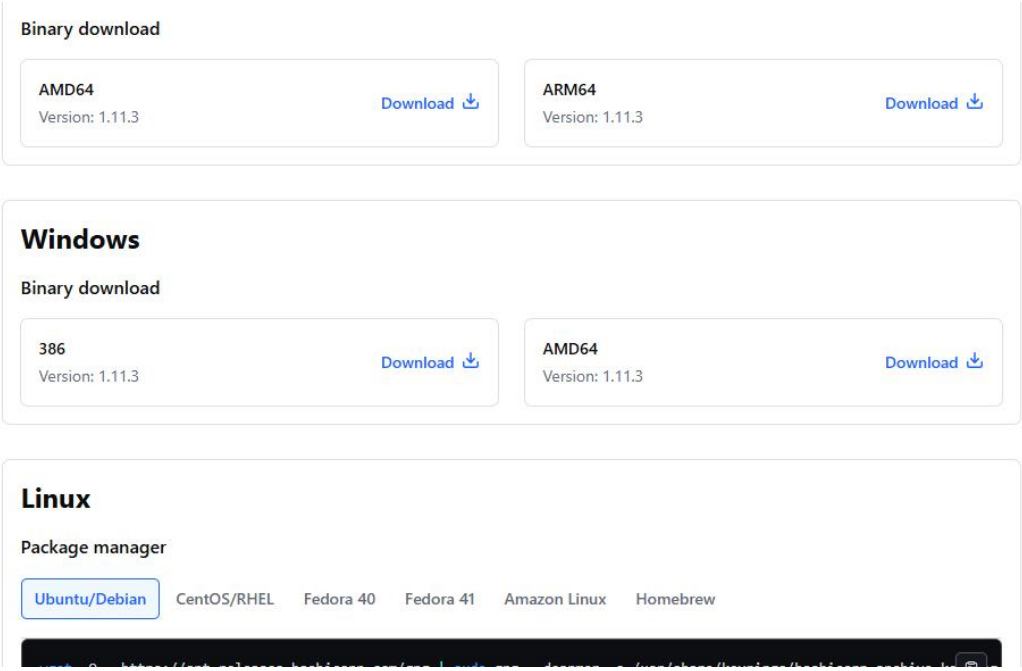
wget -O - https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
sudo apt update && sudo apt install terraform

It is open-source and free to use

Terraform is an open-source and free software developed by HashiCorp

It is developed to create/provision infrastructure in cloud platform ---> it supports almost all the cloud platforms. (Infrastructure as a Code)
Terraform will use HCL ---> Hashicorp configuration language

Terraform installation:



Windows Binary download

Extract Terraform files and copy terraform.exe to a folder

Open System variables and set path

Environment Variables

User variables for saito

Variable	Value
OneDrive	C:\Users\saito\OneDrive
OneDriveConsumer	C:\Users\saito\OneDrive
Path	C:\Program Files\MySQL\MySQL Shell 8.0\bin\;C:\Users\saito\...
TEMP	C:\Users\saito\AppData\Local\Temp
TMP	C:\Users\saito\AppData\Local\Temp

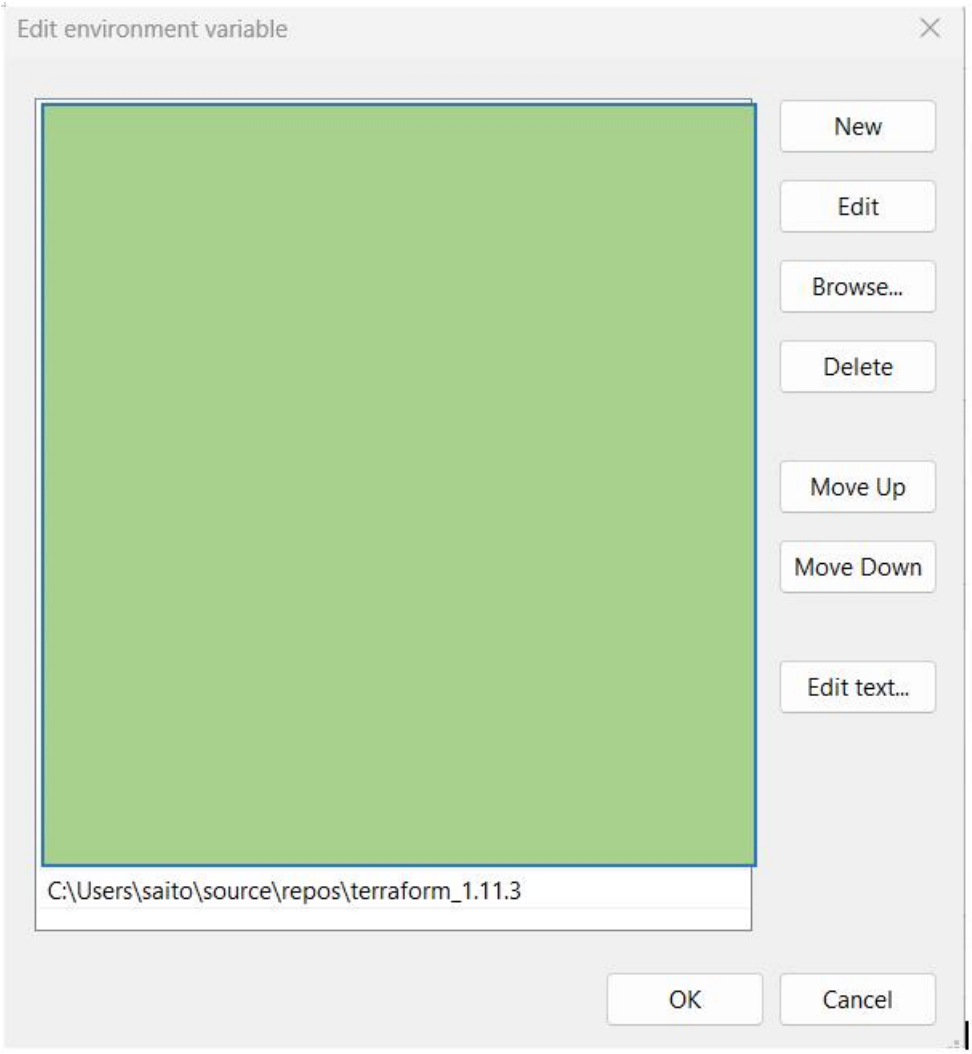
New... Edit... Delete

System variables

Variable	Value
JAVA_HOME	C:\Users\saito\Downloads\openjdk-22.0.2_windows-x64_bin\j...
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files (x86)\Common Files\Oracle\Java\java8path;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTU...	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 154 Stepping 4, GenuineIntel

New... Edit... Delete

OK Cancel



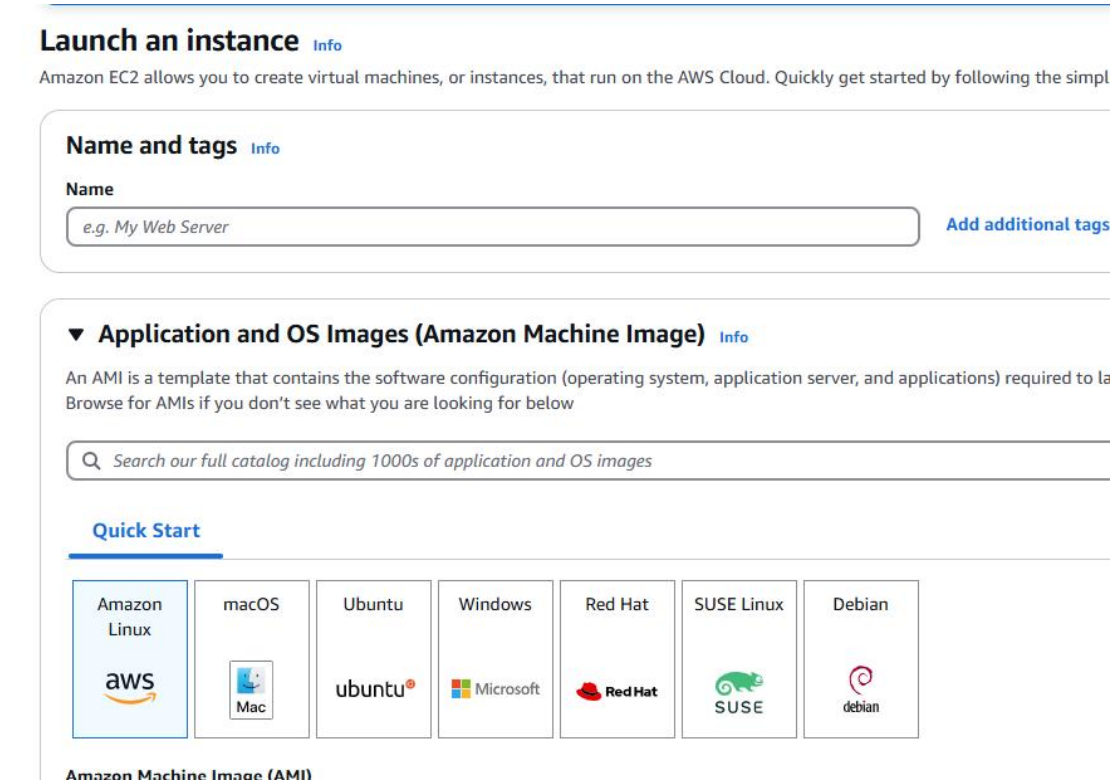
Add terraform path to the list of variables --> Ok

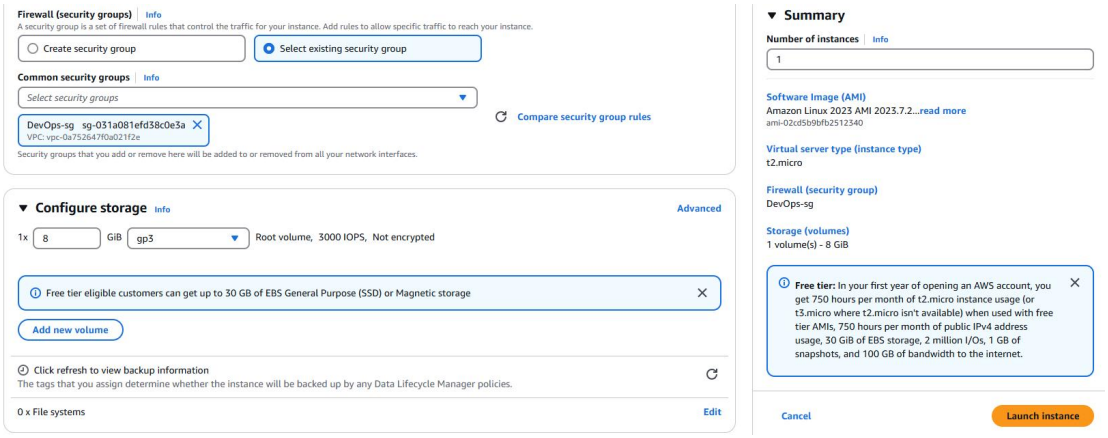
Open Windows Powershell or Command prompt

```
PS C:\Users\saito> terraform version
Terraform v1.11.3
on windows_386
PS C:\Users\saito>
```

```
on windows_386
PS C:\Users\saito> terraform -help
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure
```

Create a new EC2

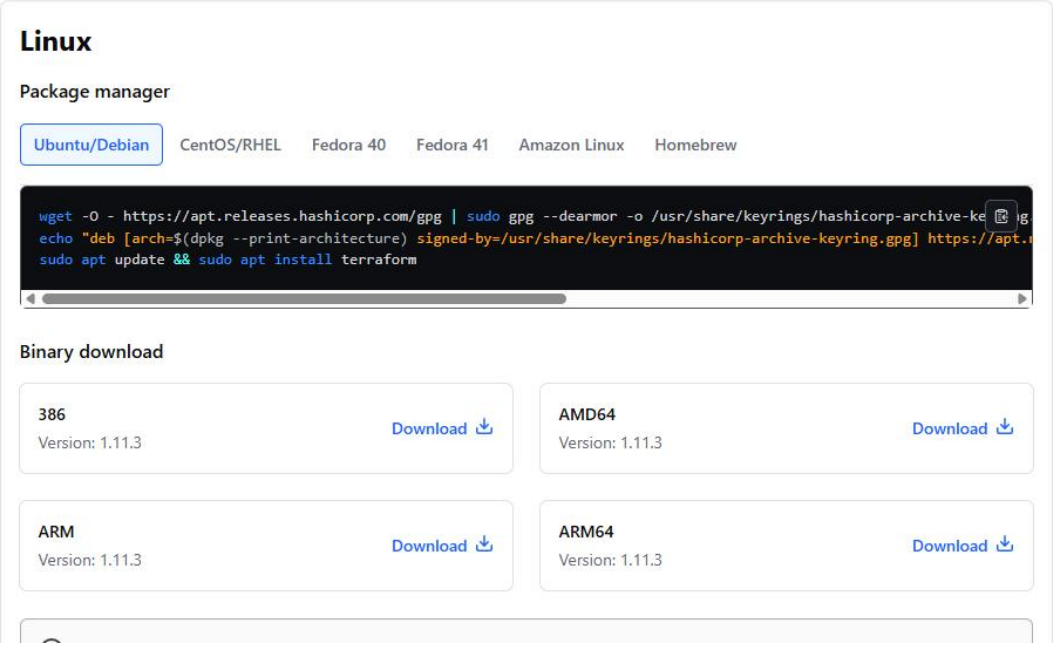




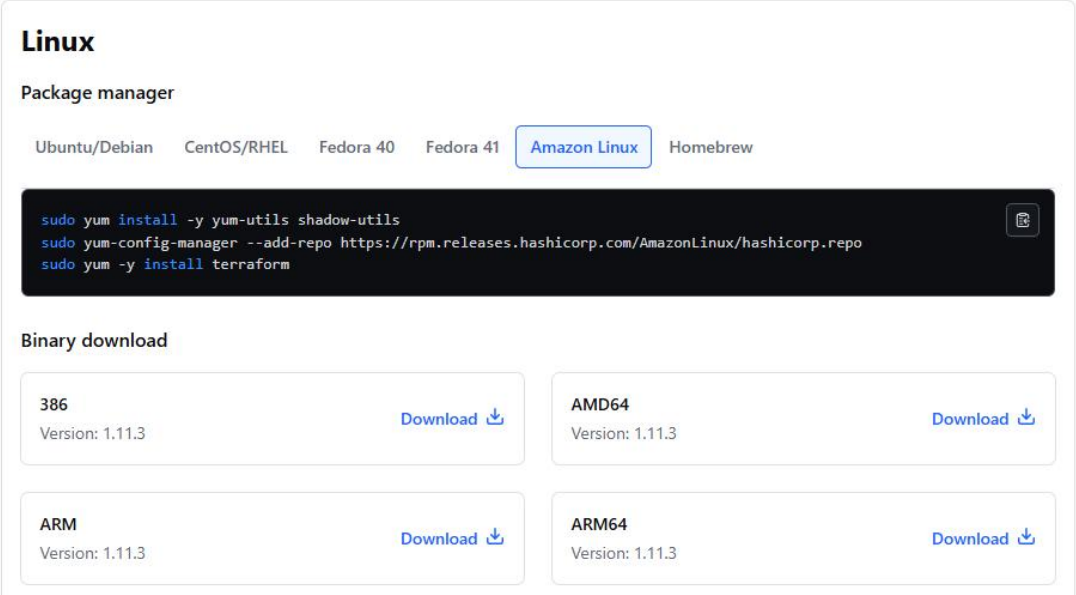
Copy public IP and open in MobaXTerm

```
  ~~        \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~        V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-12-203 ~]$
[ec2-user@ip-172-31-12-203 ~]$
[ec2-user@ip-172-31-12-203 ~]$
```

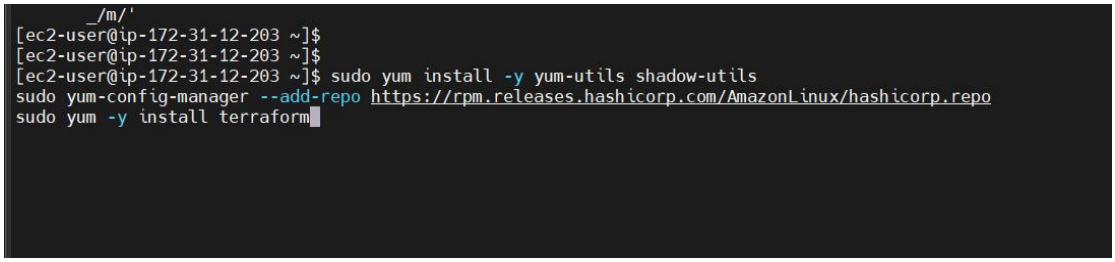
Go to Terraform website then download
https://developer.hashicorp.com/terraform/install?product_intent=terraform

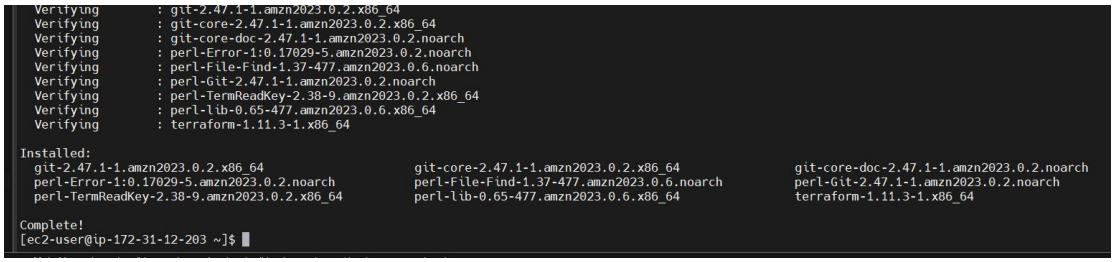


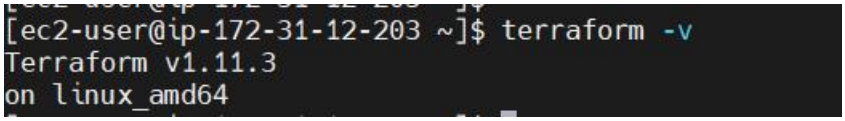
Because our EC2 has Amazon Linux, click that



Copy and paste into EC2

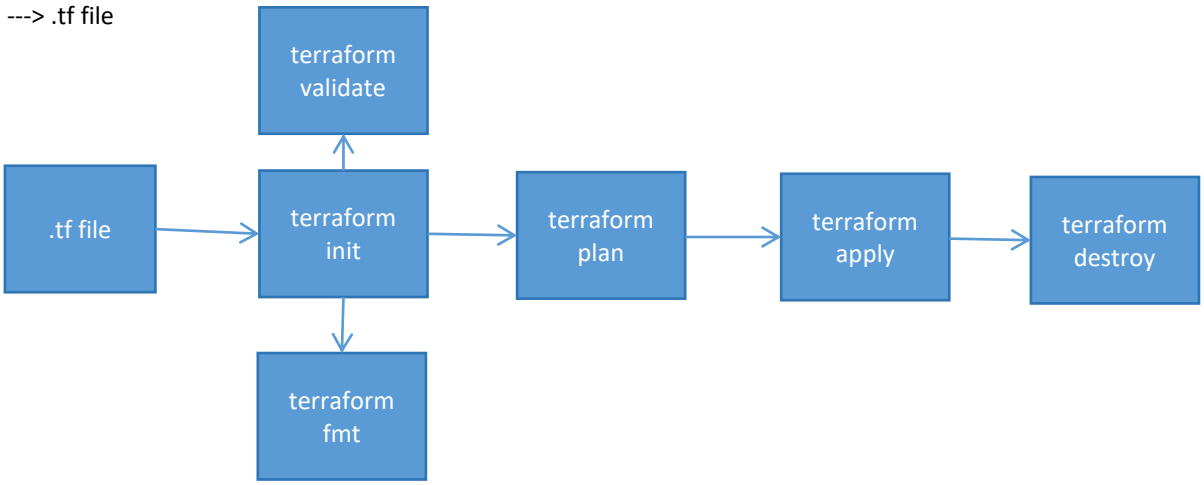






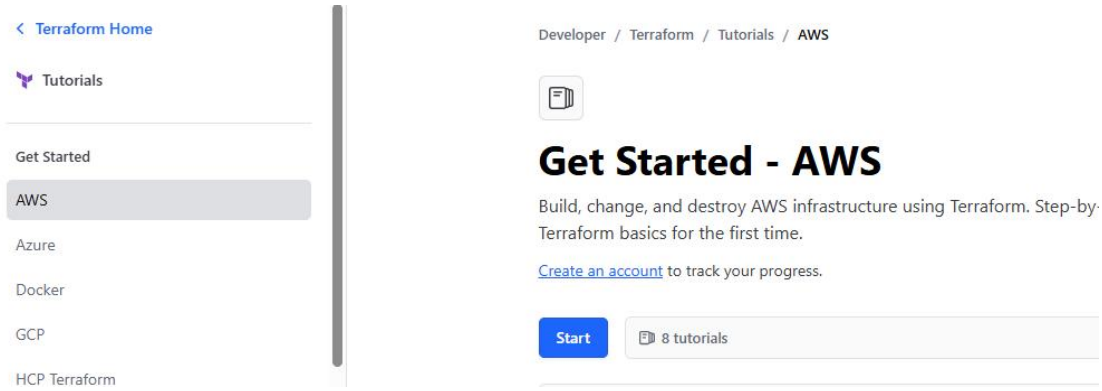
1. Created Linux VM in AWS cloud (AMI: Amazon Linux)
2. Conenct with Linux VM using MobaXTerm or Gitbash
3. Execute the below commands to setup Terraform in Linux VM
sudo yum install -y yum-utils shadow-utils
sudo yum-config-manager --add-repo
https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
sudo yum -y install terraform
4. Verify Terraform installation
    a) terraform -v
    b) terraform version

Terraform Architecture
---> Terraform uses HCL to write scripts
---> .tf file



terraform destroy --> used to destroy only the resources created using the script
terraform validate --> validate the terraform commands you are written
terraform fmt --> it will format terraform commands with proper indent spacing
terraform plan --> create execution plan for the script
terraform apply --> it will actually create the resources

Go to the tutorials page: https://developer.hashicorp.com/terraform/tutorials



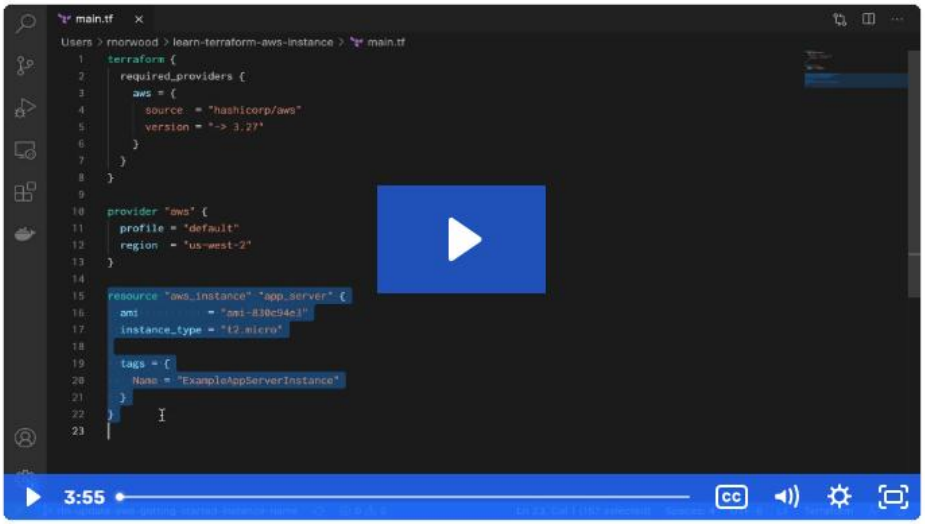
Build infrastructure

Developer / Terraform / Tutorials / AWS / **Build**

Build infrastructure

11min | Terraform ▷ Video

Reference this often? Create an account to bookmark tutorials.



Example

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.16"
    }
  }

  required_version = ">= 1.2.0"
}

provider "aws" {
  region  = "us-west-2"
}

resource "aws_instance" "app_server" {
  ami           = "ami-830c94e3"
  instance_type = "t2.micro"

  tags = {
    Name = "ExampleAppServerInstance"
  }
}
```

https://developer.hashicorp.com/terraform/tutorials/aws-get-started/aws-build

---> Terraform script we use .tf extension

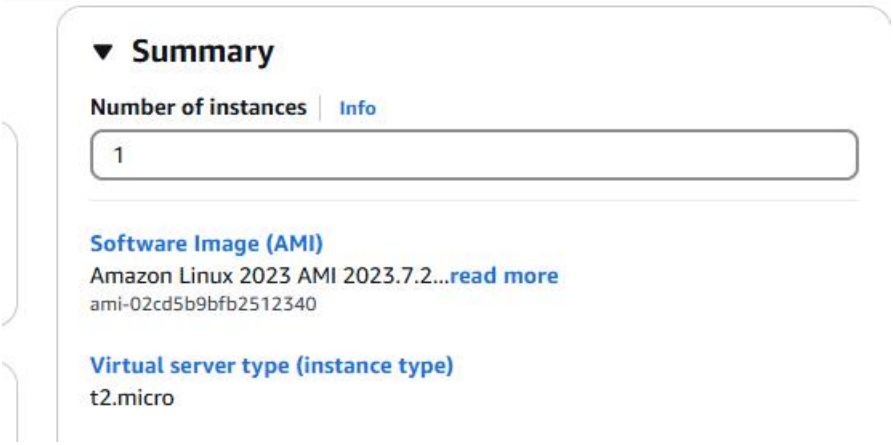
.tf --> init --> fmt --> validate --> plan --> apply --> destroy

terraform init --> initialize terraform script (.tf file)
terraform fmt --> format terraform script indent spacings (optional)
terraform validate --> verify terraform script syntax valid or not
terraform plan --> create execution plan for terraform script
terraform apply --> create actual resource in cloud based on given plan
terraform destroy --> it is used to delete resources created wkth our Terraform script

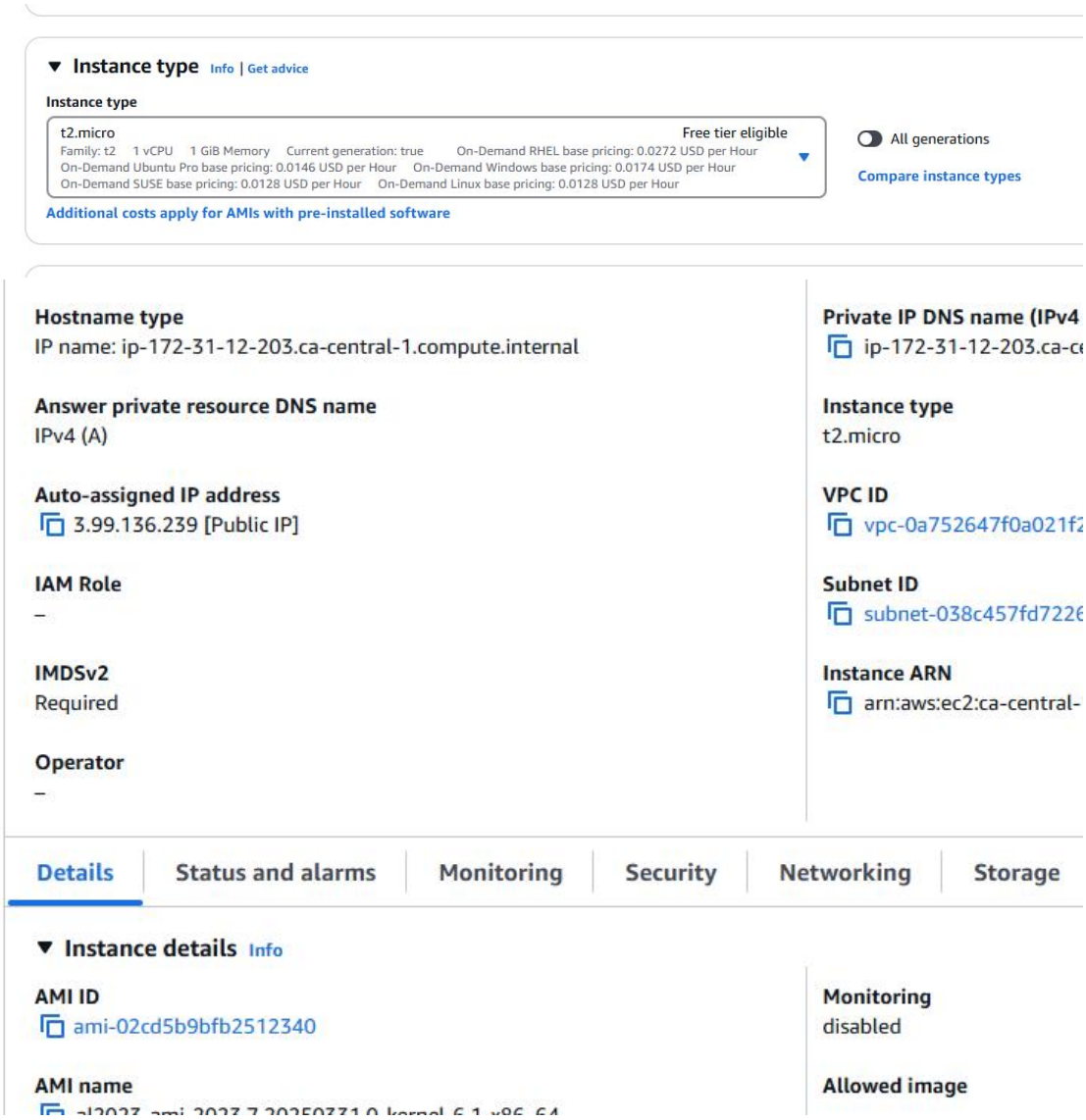
Terraform script to create EC2 instance

```
provider “aws” {
        region=”ca-central-1”
        access_key=xxxxxxxxxxxxx
        secret_key=xxxxxxxxxxxxxxxxxx
}

resource “aws_instance” “linux-vm”{
        ami=”ami-02cd5b9bfb2512340”
        instance_type=”t2.micro”
        key_name=”terraform”
        security_groups=[“default”]
        tags={
                Name=”Terraform-test-VM”
            }

}
```

Create access keys
Go to EC2 launch instance and find AMI ID
AMI ID



t2.micro



Which cloud provider you want to use? Which resource within provider you want to use?

```
[ec2-user@ip-172-31-12-203 ~]$ mkdir 01-tf-script
[ec2-user@ip-172-31-12-203 ~]$
[ec2-user@ip-172-31-12-203 ~]$ ls -l
total 0
drwxr-xr-x. 2 ec2-user ec2-user 6 Apr  5 18:08 01-tf-script
[ec2-user@ip-172-31-12-203 ~]$

[ec2-user@ip-172-31-12-203 01-tf-script]$ vi main.tf
[ec2-user@ip-172-31-12-203 01-tf-script]$
```

Paste the entire script in Vim

```
provider “aws” {
        region=”ca-central-1”
        access_key=xxxxxxxxxxxx
        secret_key=xxxxxxxxxxxxxxxxxx
}

resource “aws_instance” “linux-vm”{
        ami=”ami-02cd5b9bfb2512340”
        instance_type=”t2.micro”
        key_name=”terraform”
        security_groups=[“default”]
        tags={
                Name=”Terraform-test-VM”
                 }

}

~
~
```


```
provider “aws” {
    region=”ca-central-1”
    access_key=xxxxxxxxxxxx
    secret_key=xxxxxxxxxxxxxxxxxx
}

resource “aws_instance” “linux-vm”{
    ami=”ami-02cd5b9bfb2512340”
    instance_type=”t2.micro”
    key_name=”terraform”
    security_groups=[“default”]
    tags={
        Name=”Terraform-test-VM”
         }

}
```


```
[ec2-user@ip-172-31-12-203 01-tf-script]$
[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.94.1...
```

```
[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.94.1...
- Installed hashicorp/aws v5.94.1 (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.
[ec2-user@ip-172-31-12-203 01-tf-script]$
```

[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform validate
Success! The configuration is valid.

```
[ec2-user@ip-172-31-12-203 01-tf-script]$
[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.94.1

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.
[ec2-user@ip-172-31-12-203 01-tf-script]$
[ec2-user@ip-172-31-12-203 01-tf-script]$
[ec2-user@ip-172-31-12-203 01-tf-script]$
[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform validate
Success! The configuration is valid.
```

Shape before formatting

```
resource "aws_instance" "linux-vm"{
        ami="ami-02cd5b9bfb2512340"
        instance_type="t2.micro"
        key_name="terraform"
        security_groups=["default"]
        tags={
                Name="Terraform-test-VM"
          }

}
```

[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform fmt
main.tf

After formatting

```
resource "aws_instance" "linux-vm" {
  ami             = "ami-02cd5b9bfb2512340"
  instance_type   = "t2.micro"
  key_name        = "terraform"
  security_groups = ["default"]
  tags = {
    Name = "Terraform-test-VM"
  }

}
```

terraform apply

```
[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform apply

Terraform used the selected providers to generate the following executi
  + create

Terraform will perform the following actions:

  # aws_instance.linux-vm will be created
  + resource "aws_instance" "linux-vm" {
      + ami                                  = "ami-02cd5b9bfb2512340"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "terraform"
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
```

```
      + ephemeral_block_device (known after apply)

      + instance_market_options (known after apply)

      + maintenance_options (known after apply)

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (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:
```

We verify what resources we want to create then we say yes

TerraformEC2	i-0f3b562c215e434b7	Running	t2.micro	2/2 checks passec	View alarms +
Terraform-test...	i-0c9b2aa4366ed3ac0	Running	t2.micro	Initializing	View alarms +

See now the new EC2 is being created ---> Terraform-test-VM

```
    }

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_instance.linux-vm: Creating...
aws_instance.linux-vm: Still creating... [11s elapsed]
aws_instance.linux-vm: Creation complete after 13s [id=i-0c9b2aa4366ed3ac0]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-12-203 01-tf-script]$
```

[ec2-user@ip-172-31-12-203 01-tf-script]$ ls -l
total 16
-rw-r--r--. 1 ec2-user ec2-user 374 Apr 5 18:30 main.tf
-rw-r--r--. 1 ec2-user ec2-user 4935 Apr 5 18:31 terraform.tfstate
-rw-r--r--. 1 ec2-user ec2-user 181 Apr 5 18:31 terraform.tfstate.backup

terraform.tfstate --> state file has all the information about terraform created resources

If I re-run terraform apply, no changes

```
[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform apply
aws_instance.linux-vm: Refreshing state... [id=i-0c9b2aa4366ed3ac0]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-12-203 01-tf-script]$
```

For destroying also, terraform will look for state file only

When I say destroy we can see it is shutting down

☐	TerraformEC2	i-0f3b562c215e434b7	Running	t2.micro
☐	Terraform-test-VM	i-0c9b2aa4366ed3ac0	Shutting-d...	t2.micro

```
aws_instance.linux-vm: Destroying... [id=i-0c9b2aa4366ed3ac0]
aws_instance.linux-vm: Still destroying... [id=i-0c9b2aa4366ed3ac0, 10s elapsed]
aws_instance.linux-vm: Still destroying... [id=i-0c9b2aa4366ed3ac0, 20s elapsed]
aws_instance.linux-vm: Still destroying... [id=i-0c9b2aa4366ed3ac0, 30s elapsed]
aws_instance.linux-vm: Still destroying... [id=i-0c9b2aa4366ed3ac0, 40s elapsed]
aws_instance.linux-vm: Still destroying... [id=i-0c9b2aa4366ed3ac0, 50s elapsed]
aws_instance.linux-vm: Still destroying... [id=i-0c9b2aa4366ed3ac0, 1m0s elapsed]
aws_instance.linux-vm: Destruction complete after 1m1s

Destroy complete! Resources: 1 destroyed.
[ec2-user@ip-172-31-12-203 01-tf-script]$
```

```
[ec2-user@ip-172-31-12-203 01-tf-script]$ ls -l
total 16
-rw-r--r--. 1 ec2-user ec2-user  374 Apr  5 18:30 main.tf
-rw-r--r--. 1 ec2-user ec2-user  181 Apr  5 18:45 terraform.tfstate
-rw-r--r--. 1 ec2-user ec2-user 4935 Apr  5 18:45 terraform.tfstate.backup
[ec2-user@ip-172-31-12-203 01-tf-script]$ rm terraform.tfstate
[ec2-user@ip-172-31-12-203 01-tf-script]$ rm terraform.tfstate.backup
[ec2-user@ip-172-31-12-203 01-tf-script]$ ls -l
total 4
-rw-r--r--. 1 ec2-user ec2-user 374 Apr  5 18:30 main.tf
[ec2-user@ip-172-31-12-203 01-tf-script]$
```

Dealing with Access key and Secret key
--> Instead of configuring access key and secret key in Terraform script file, we could configure them in environment variables. But even if you disconnect and re-connect to VM with the tools, then also environment variables will be gone. Again you have to add these values into environment variables

I have removed access key and secret key from main.tf

```
2. 3.99.136.239 (ec2-user)
provider "aws" {
  region     = "ca-central-1"
}

resource "aws_instance" "linux-vm" {
  ami             = "ami-02cd5b9bfb2512340"
  instance_type   = "t2.micro"
  key_name        = "DevOpsMar30"
  security_groups = ["default"]
  tags = {
    Name = "Terraform-test-VM"
  }

}
```

Currently we don't have the keys in environment variables

```
[ec2-user@ip-172-31-12-203 01-tf-script]$
[ec2-user@ip-172-31-12-203 01-tf-script]$
[ec2-user@ip-172-31-12-203 01-tf-script]$ echo $AWS_ACCESS_KEY_ID

[ec2-user@ip-172-31-12-203 01-tf-script]$
[ec2-user@ip-172-31-12-203 01-tf-script]$
[ec2-user@ip-172-31-12-203 01-tf-script]$ echo $AWS_SECRET_KEY

[ec2-user@ip-172-31-12-203 01-tf-script]$
```

Add environment variables using export command
[ec2-user@ip-172-31-12-203 01-tf-script]$ export AWS_SECRET_KEY=""

You can print and see as well
[ec2-user@ip-172-31-12-203 01-tf-script]$ echo $AWS_ACCESS_KEY_ID
[ec2-user@ip-172-31-12-203 01-tf-script]$ echo $AWS_SECRET_KEY

[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.94.1

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.
[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform fmt
main.tf
[ec2-user@ip-172-31-12-203 01-tf-script]$ ls -l
total 4

```
-rw-r--r--. 1 ec2-user ec2-user 274 Apr  5 19:14 main.tf
[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-31-12-203 01-tf-script]$ cat main.tf
provider "aws" {
  region = "ca-central-1"
}

resource "aws_instance" "linux-vm" {
  ami           = "ami-02cd5b9bfb2512340"
  instance_type   = "t2.micro"
  key_name       = "DevOpsMar30"
  security_groups = ["default"]
  tags = {
    Name = "Terraform-test-VM"
  }

}
```

```
[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.94.1

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.
[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform fmt
main.tf
[ec2-user@ip-172-31-12-203 01-tf-script]$ ls -l
total 4
-rw-r--r--. 1 ec2-user ec2-user 274 Apr  5 19:14 main.tf
[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-31-12-203 01-tf-script]$ cat main.tf
provider "aws" {
  region = "ca-central-1"
}

resource "aws_instance" "linux-vm" {
  ami             = "ami-02cd5b9bfb2512340"
  instance_type   = "t2.micro"
  key_name        = "DevOpsMar30"
  security_groups = ["default"]
  tags = {
    Name = "Terraform-test-VM"
  }

}
```

Because we have deleted tfstate files, now Terraform init and apply will work again

We have added secret key and access key in the environment variables

```
[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform plan

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

Terraform will perform the following actions:

  # aws_instance.linux-vm will be created
  + resource "aws_instance" "linux-vm" {
      + ami                                  = "ami-02cd5b9bfb2512340"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
```

terraform apply

```
[ec2-user@ip-172-31-12-203 01-tf-script]$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource act
  + create

Terraform will perform the following actions:

  # aws_instance.linux-vm will be created
  + resource "aws_instance" "linux-vm" {
      + ami                                  = "ami-02cd5b9bfb2512340"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
```

```
      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (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_instance.linux-vm: Creating...
aws_instance.linux-vm: Still creating... [10s elapsed]
aws_instance.linux-vm: Creation complete after 12s [id=i-018109c075363d356]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-12-203 01-tf-script]$
```

TerraformEC2	i-0f3b562c215e434b7	Running	t2.micro	2/2 checks passed
Terraform-test...	i-018109c075363d356	Running	t2.micro	2/2 checks passed

Again it's up and running

terraform init, fmt, validate, plan, apply then yes
[ec2-user@ip-172-31-12-203 01-tf-script]$ cat terraform.tfstate

```
[ec2-user@ip-172-31-12-203 01-tf-script]$
[ec2-user@ip-172-31-12-203 01-tf-script]$ cat terraform.tfstate
{
  "version": 4,
  "terraform_version": "1.11.3",
  "serial": 1,
  "lineage": "36f8d576-62e4-bc44-1fc0-f4b62e4300e4",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "aws_instance",
      "name": "linux-vm",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
      "instances": [
        {
          "schema_version": 1,
          "attributes": {
            "ami": "ami-02cd5b9bfb2512340",
            "arn": "arn:aws:ec2:ca-central-1:577638386543:instance/i-018109c075363d356",
            "associate_public_ip_address": true,
            "availability_zone": "ca-central-1b",
            "capacity_reservation_specification": [
              {
                "capacity_reservation_preference": "open",
                "capacity_reservation_target": []
              }
            ],
            "cpu_core_count": 1
```

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

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.linux-vm: Destroying... [id=i-018109c075363d356]
aws_instance.linux-vm: Still destroying... [id=i-018109c075363d356, 10s elapsed]
aws_instance.linux-vm: Still destroying... [id=i-018109c075363d356, 20s elapsed]
aws_instance.linux-vm: Still destroying... [id=i-018109c075363d356, 30s elapsed]
aws_instance.linux-vm: Still destroying... [id=i-018109c075363d356, 40s elapsed]
aws_instance.linux-vm: Still destroying... [id=i-018109c075363d356, 50s elapsed]
aws_instance.linux-vm: Still destroying... [id=i-018109c075363d356, 1m0s elapsed]
aws_instance.linux-vm: Still destroying... [id=i-018109c075363d356, 1m10s elapsed]
aws_instance.linux-vm: Still destroying... [id=i-018109c075363d356, 1m20s elapsed]
aws_instance.linux-vm: Destruction complete after 1m20s

Destroy complete! Resources: 1 destroyed.
[ec2-user@ip-172-31-12-203 01-tf-script]$
```

Creating EC2 VM with User data:

[ec2-user@ip-172-31-12-203 ~]$ cd 01-tf-script-userdata/
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$ ls

For that we got to write a script and reference in the Terraform file
Create Script sh file
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$ vi installHttpd.sh
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1> Welcome to Terraform Webserver </h1></html>" > index.html
service httpd start

[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$ cat installHttpd.sh
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1> Welcome to Terraform Webserver </h1></html>" > index.html
service httpd start

Give permission
ec2-user@ip-172-31-12-203 01-tf-script-userdata]$ chmod u+x installHttpd.sh

```
provider "aws" {
  region = "ca-central-1"
}

resource "aws_instance" "linux-vm" {
  ami             = "ami-02cd5b9bfb2512340"
  instance_type   = "t2.micro"
  key_name        = "DevOpsMar30"
  security_groups = ["default"]
  user_data=file("installHttpd.sh)
  tags = {
    Name = "Terraform-test-VM"
  }

}
```

```
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.94.1...
- Installed hashicorp/aws v5.94.1 (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.
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$
```

```
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.94.1...
- Installed hashicorp/aws v5.94.1 (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.
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$ terraform fmt
main.tf
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$ tarraform validate
-bash: tarraform: command not found
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$ terraform validate
Success! The configuration is valid.
```

terraform apply --auto-approve also works

```
      + ebs_block_device (known after apply)

      + enclave_options (known after apply)

      + ephemeral_block_device (known after apply)

      + instance_market_options (known after apply)

      + maintenance_options (known after apply)

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (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_instance.linux-vm: Creating...
aws_instance.linux-vm: Still creating... [10s elapsed]
aws_instance.linux-vm: Creation complete after 12s [id=i-07922b45ed2c5e74a]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-12-203 01-tf-script-userdata]$
```

It created a VM

Terraform-test-VM	i-018109c075363d356	Terminated	t2.micro	–
Terraform-test-VM	i-096c47ba2c61c7831	Running	t2.micro	Initializing
Terraform-test-VM	i-07922b45ed2c5e74a	Terminated	t2.micro	–

Open copy Public IP

Instance summary for i-096c47ba2c61c7831 (Terraform-test-VM) Info

Updated 3 minutes ago

Instance ID
i-096c47ba2c61c7831

IPv6 address
–

Hostname type
IP name: ip-172-31-9-8.ca-central-1.compute.internal

Answer private resource DNS name
–

Auto-assigned IP address
99.79.73.245 [Public IP]

Public IPv4 address
99.79.73.245 | open address

Instance state
Running

Private IP DNS name (IPv4 only)
ip-172-31-9-8.ca-central-1.compute.internal

Instance type
t2.micro

VPC ID
vpc-0a752647f0a021f2e (default_VPC)

Open Public IP in browser

Not secure 99.79.73.245

New Tab | Where should finge... | Move or copy cells...

Welcome to Terraform Webserver

Add Access Key and Secret Key in environment variables
Create a new directory: mkdir 01-tf-script-userdata
Create a script file: vi installHttpd.sh

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1> Welcome to Terraform Webserver </h1></html>" > index.html
service httpd start
```

Provide the execute permission for Script file
chmod u+x installHttpd.sh

Create a script file: vi main.tf

```
provider "aws" {
  region = "ca-central-1"
}

resource "aws_instance" "linux-vm" {
  ami           = "ami-02cd5b9bfb2512340"
  instance_type   = "t2.micro"
  key_name       = "DevOpsMar30"
  security_groups = ["default"]
  user_data = file("installHttpd.sh")
```

```
  tags = {
    Name = "Terraform-test-VM"
  }

}
```

terraform init
terraform fmt
terraform validate
terraform apply --auto-approve
terraform destroy

https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket

```
resource "aws_s3_bucket" "example" {
  bucket = "my-tf-test-bucket"

  tags = {
    Name        = "My bucket"
    Environment = "Dev"
  }
}
```

Variables in Terraform:
Variables are used to store data in key-value format
name="DevOps"

Types of variables in Terraform ecosystem
1. Input variables ---> Supply values to Terraform script
We can remove hardcoded values from our resources script using Input variables concept
2. Output variables ---> Get the values from Terraform script after execution
Examples: After EC2 VM is created, print EC2-VM Public IP
After IAM user got created, print IAM user information
After S3 bucket got created, print bucket info

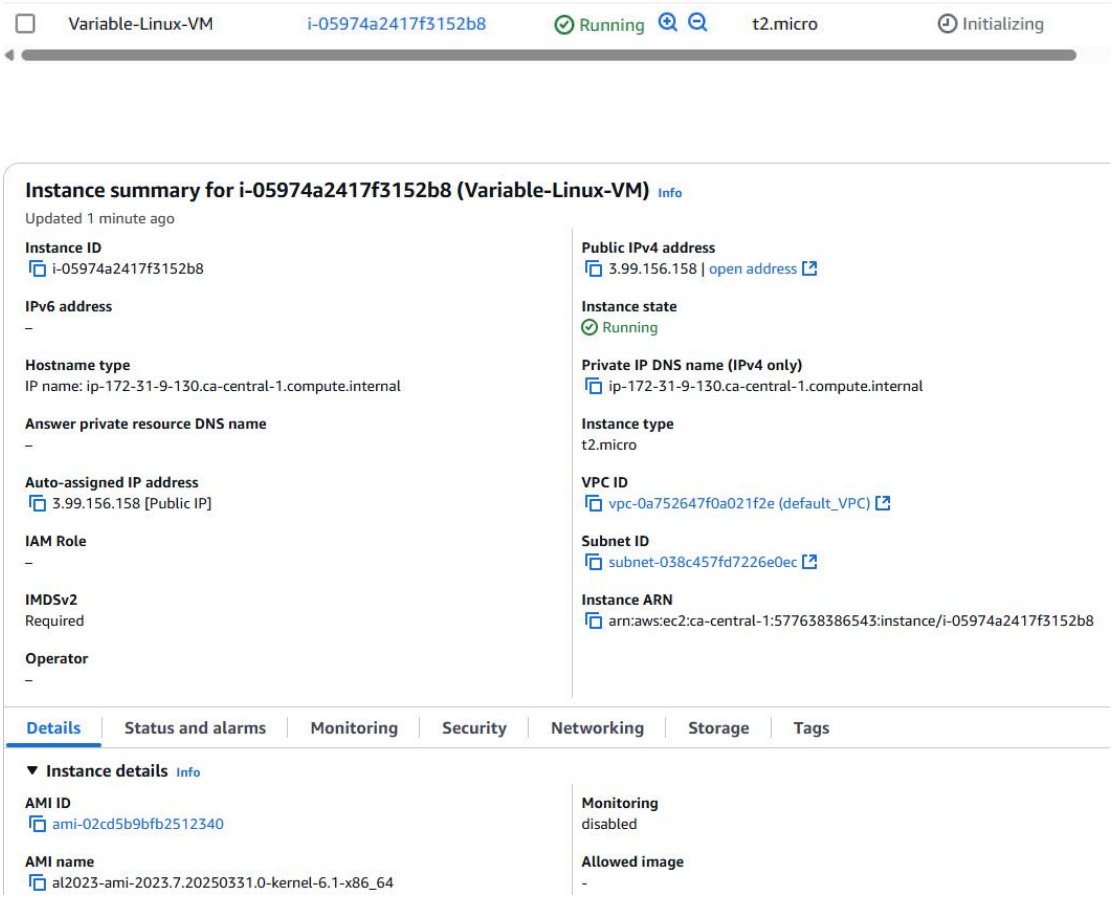
===> Variables
These variables we can maintain in separate Terraform tf file

```
[ec2-user@ip-172-31-12-203 03-tf-script-vars]$ cat vars.tf
variable "ami" {
      description = "Amazon vm image value"
      default = "ami-02cd5b9bfb2512340"
      }
variable "instance_type" {
      description = "Represents the type of instance"
      default = "t2.micro"
}
```

Created a new directory mkdir 03-tf-script-vars
cd 03-tf-script-vars
Created separate tf file for vars

```
vi vars.tf
variable "ami" {
  description = "Amazon vm image value"
  default     = "ami-02cd5b9bfb2512340"
}
variable "instance_type" {
  description = "Represents the type of instance"
  default     = "t2.micro"
}


$ vi main.tf
[ec2-user@ip-172-31-12-203 03-tf-script-vars]$ cat main.tf
resource "aws_instance" "linux-vm" {
  ami           = "${var.ami}"
  instance_type   = "${var.instance_type}"
  key_name       = "DevOpsMar30"
  security_groups = ["default"]
  tags = {
    name = "Variable-Linux-VM"
  }

}


[ec2-user@ip-172-31-12-203 03-tf-script-vars]$ ls -l
total 12
-rw-r--r--. 1 ec2-user ec2-user 230 Apr  5 23:04 main.tf
-rw-r--r--. 1 ec2-user ec2-user  44 Apr  5 23:05 provider.tf
-rw-r--r--. 1 ec2-user ec2-user 197 Apr  5 22:55 vars.tf

[ec2-user@ip-172-31-12-203 03-tf-script-vars]$ cat provider.tf
provider "aws" {
      region = "ca-central-1"
}


terraform init
terraform fmt
terraform validate
terraform plan
terraform apply
```

```
      + metadata_options (known after apply)

      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.
aws_instance.linux-vm: Creating...
aws_instance.linux-vm: Still creating... [10s elapsed]
aws_instance.linux-vm: Creation complete after 13s [id=i-05974a2417f3152b8]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-12-203 03-tf-script-vars]$
```



Instance_type: t2.micro and AMI ID: ami-02cd5b9bfb2512340 match the vars.tf file

```
[ec2-user@ip-172-31-12-203 04-tf-script-var]$ ls -l
total 16
-rw-r--r--. 1 ec2-user ec2-user 207 Apr  5 23:43 input-vars.tf
-rw-r--r--. 1 ec2-user ec2-user 220 Apr  5 23:45 main.tf
-rw-r--r--. 1 ec2-user ec2-user  71 Apr  5 23:52 output.tf
-rw-r--r--. 1 ec2-user ec2-user  48 Apr  5 23:42 provider.tf
```

```
[ec2-user@ip-172-31-12-203 04-tf-script-var]$ cat input-vars.tf
variable "ami" {
  description = "Amazon vm image value"
  default     = "ami-02cd5b9bfb2512340"
}
variable "instance_type" {
  description = "Represents the type of instance"
  default     = "t2.micro"
}

[ec2-user@ip-172-31-12-203 04-tf-script-var]$ cat main.tf
resource "aws_instance" "linux_vm" {
  ami           = vars.ami
  instance_type   = vars.instance_type
  key_name        = "DevOpsMar30"
  security_groups = ["default"]
  tags = {
```

```
  Name = "Variable-Linux-VM"
 }

}
```

```
[ec2-user@ip-172-31-12-203 04-tf-script-var]$ cat output.tf
output "ec2_vm_public_ip" {
      value=aws_instance.linux_vm.public_ip

}
```

```
[ec2-user@ip-172-31-12-203 04-tf-script-var]$ cat provider.tf
provider "aws" {
      region = "ca-central-1"
}
```

Change output.tf

```
aws_instance.linux_vm: Creation complete after 13s [id=i-0d512bd8

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

Outputs:

ec2_vm_info = "35.182.231.68"
[ec2-user@ip-172-31-12-203 04-tf-script-var]$
```

```
[ec2-user@ip-172-31-12-203 04-tf-script-var]$ cat output.tf
output "ec2_vm_info" {
 value = aws_instance.linux_vm
}
```

Now in the output we see the entire virtual machine

Terraform Script with Input & Output variables
Create new directory
Cd into new dir
Create provider.tf
vi provider.tf
Create input-vars.tf
vi input-vars.tf
Create output.tf
vi output.tf

```
resource "aws_s3_bucket" "example" {
 bucket = "my-tf-test-bucket"

 tags = {
  Name       = "My bucket"
  Environment = "Dev"
 }
}
```

Creating S3

```
[ec2-user@ip-172-31-12-203 05-tf-script-var]$ ls -l
total 20
-rw-r--r--. 1 ec2-user ec2-user  200 Apr  6 01:50 main.tf
-rw-r--r--. 1 ec2-user ec2-user  231 Apr  6 01:45 output.tf
-rw-r--r--. 1 ec2-user ec2-user  254 Apr  6 01:51 provider.tf
-rw-r--r--. 1 ec2-user ec2-user 3181 Apr  6 01:52 terraform.tfstate
-rw-r--r--. 1 ec2-user ec2-user  181 Apr  6 01:52 terraform.tfstate.backup
[ec2-user@ip-172-31-12-203 05-tf-script-var]$ cat main.tf
resource "aws_s3_bucket" "my_bucket" {
  bucket = "my-unique-bucket-name-12345-sai-dev-20250330" # make sure it's unique


  tags = {
    Name        = "My S3 Bucket"
    Environment = "Dev"
  }
}


[ec2-user@ip-172-31-12-203 05-tf-script-var]$ cat provider.tf
provider "aws" {
  region = "ca-central-1" # or your preferred AWS region like us-east-1, etc.

  # Optional if you have AWS credentials configured via CLI or environment variables
  # access_key = "YOUR_ACCESS_KEY"
  # secret_key = "YOUR_SECRET_KEY"
}


[ec2-user@ip-172-31-12-203 05-tf-script-var]$ cat output.tf
output "s3_bucket_name" {
  description = "The name of the S3 bucket"
  value       = aws_s3_bucket.my_bucket.id
}

output "s3_bucket_arn" {
  description = "The ARN of the S3 bucket"
  value       = aws_s3_bucket.my_bucket.arn
}
```

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

Changes to Outputs:
  + s3_bucket_arn  = (known after apply)
  + s3_bucket_name = (known after apply)
aws_s3_bucket.my_bucket: Creating...
aws_s3_bucket.my_bucket: Creation complete after 1s [id=my-unique-bucket-name-12345-sai-dev-20250330]

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

Outputs:

s3_bucket_arn = "arn:aws:s3:::my-unique-bucket-name-12345-sai-dev-20250330"
s3_bucket_name = "my-unique-bucket-name-12345-sai-dev-20250330"
[ec2-user@ip-172-31-12-203 05-tf-script-var]$
```

Terraform has created the S3 bucket

Name	AWS Region
elasticbeanstalk-ca-central-1-577638386543	Canada (Central) ca-central-1
elasticbeanstalk-us-east-1-577638386543	US East (N. Virginia) us-east-1
my-unique-bucket-name-12345-sai-dev-20250330	Canada (Central) ca-central-1

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

Outputs:

ec2_vm_info = {
  "ami" = "ami-02cd5b9bfb2512340"
  "arn" = "arn:aws:ec2:ca-central-1:577638386543:instance/i-0f054ee91e6bb3cf0"
  "associate_public_ip_address" = true
  "availability_zone" = "ca-central-1b"
  "capacity_reservation_specification" = tolist([
    {
      "capacity_reservation_preference" = "open"
      "capacity_reservation_target" = tolist([])
    },
  ])
  "cpu_core_count" = 1
  "cpu_options" = tolist([
    {
      "amd_sev_snp" = ""
      "core_count" = 1
      "threads_per_core" = 1
    },
  ])
  "cpu_threads_per_core" = 1
  "credit_specification" = tolist([
    {
      "cpu_credits" = "standard"
    },
  ])
  "disable_api_stop" = false
```