

# 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.

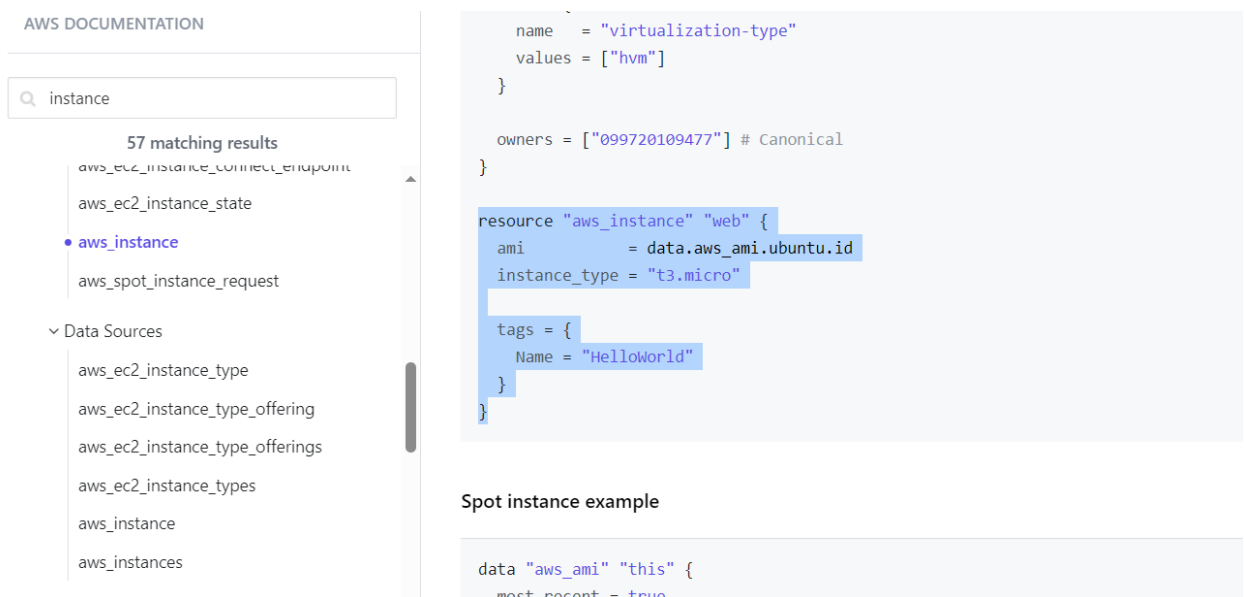
Here we will create EC2 instances virtual servers on our AWS using AMIs images

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



The image shows a screenshot of the AWS Documentation search results for the term "instance". The search results list 57 matching results, with "aws\_instance" highlighted as the selected resource. Below the search results, there is a section for "Data Sources" listing various AWS resources. To the right of the search results, there is a code block showing the Terraform configuration for creating an EC2 instance. The code includes the "virtualization-type" block, the "owners" list, and the "aws\_instance" resource definition with attributes like "ami", "instance\_type", and "tags".

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

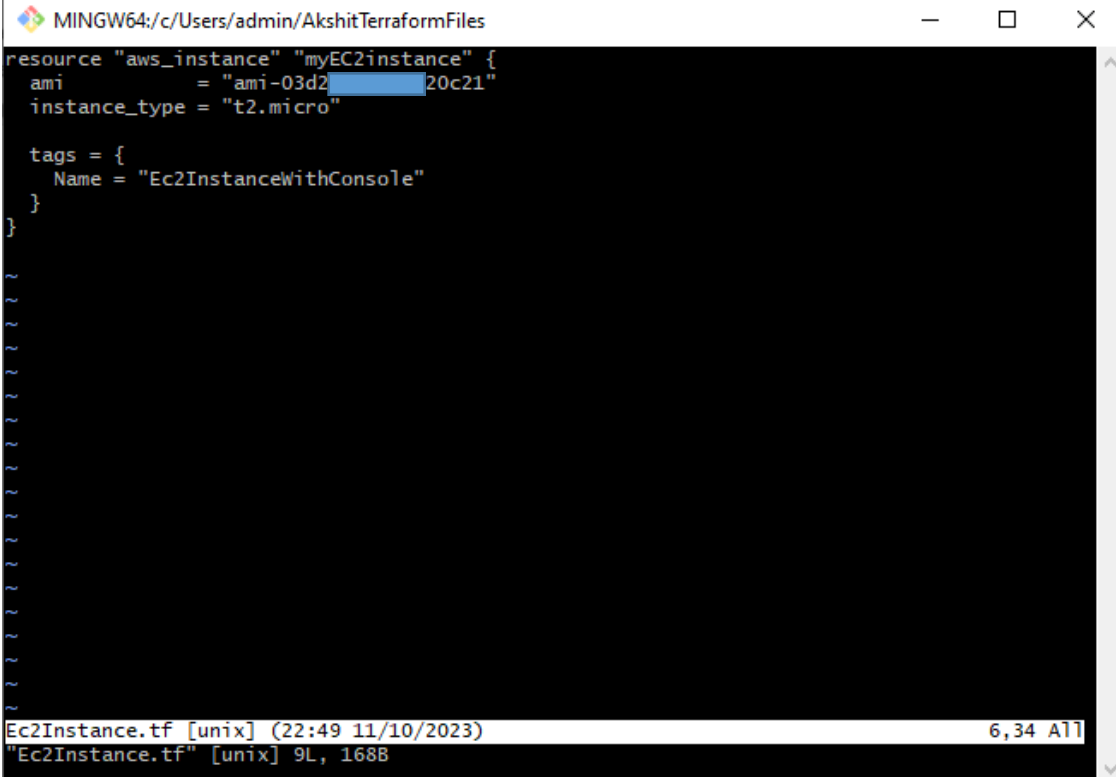
owners = ["099720109477"] # Canonical
}

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
```

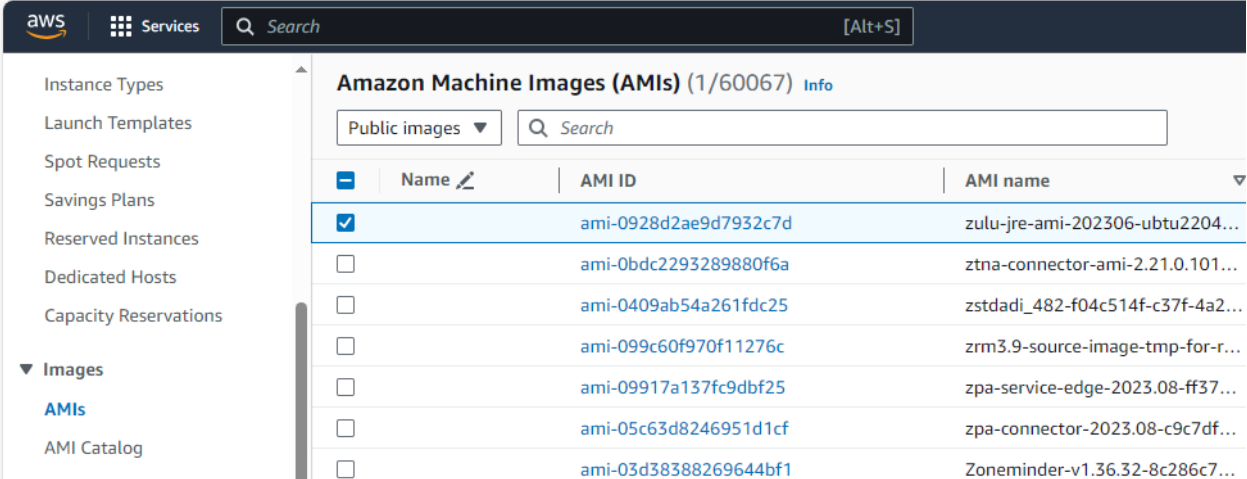


```
MINGW64/c/Users/admin/AkshitTerraformFiles
resource "aws_instance" "myEC2instance" {
  ami           = "ami-03d2[REDACTED]20c21"
  instance_type = "t2.micro"

  tags = {
    Name = "Ec2InstanceWithConsole"
  }
}
```

Ec2Instance.tf [unix] (22:49 11/10/2023) 6,34 Alt  
"Ec2Instance.tf" [unix] 9L, 168B

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.



aws Services Search [Alt+S]

Instance Types  
Launch Templates  
Spot Requests  
Savings Plans  
Reserved Instances  
Dedicated Hosts  
Capacity Reservations  
▼ Images  
AMI  
AMI Catalog

### Amazon Machine Images (AMIs) (1/60067) Info

Public images Search

	Name	AMI ID	AMI name
<input checked="" type="checkbox"/>		ami-0928d2ae9d7932c7d	zulu-jre-ami-202306-ubuntu2204...
<input type="checkbox"/>		ami-0bdc2293289880f6a	ztna-connector-ami-2.21.0.101...
<input type="checkbox"/>		ami-0409ab54a261fdc25	zstdadi_482-f04c514f-c37f-4a2...
<input type="checkbox"/>		ami-099c60f970f11276c	zrm3.9-source-image-tmp-for-r...
<input type="checkbox"/>		ami-09917a137fc9dbf25	zpa-service-edge-2023.08-ff37...
<input type="checkbox"/>		ami-05c63d8246951d1cf	zpa-connector-2023.08-c9c7df...
<input type="checkbox"/>		ami-03d38388269644bf1	Zoneminder-v1.36.32-8c286c7...

CLOUD & DEVOPS

- 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.

The screenshot shows a terminal window at the top with the following Terraform configuration for the AWS provider:

```
provider "aws" {  
  region = "ap-south-1"  
  shared_credentials_files = ["~/.aws/credentials"]  
}
```

Below the terminal is the AWS IAM console. The search bar contains the text "t2.micro". The results list several instance types:

- t2.nano**  
Family: t2 1 vCPU 0.5 GiB Memory Current generation: true  
On-Demand SUSE base pricing: 0.0062 USD per Hour  
On-Demand Linux base pricing: 0.0062 USD per Hour  
On-Demand Windows base pricing: 0.0085 USD per Hour
- t2.micro** (highlighted)  
Family: t2 1 vCPU 1 GiB Memory Current generation: true  
On-Demand Linux base pricing: 0.0124 USD per Hour  
On-Demand Windows base pricing: 0.017 USD per Hour  
On-Demand RHEL base pricing: 0.0724 USD per Hour  
On-Demand SUSE base pricing: 0.0124 USD per Hour  
Free tier eligible
- t2.small**  
Family: t2 1 vCPU 2 GiB Memory Current generation: true  
On-Demand SUSE base pricing: 0.0548 USD per Hour  
On-Demand Linux base pricing: 0.0248 USD per Hour  
On-Demand RHEL base pricing: 0.0848 USD per Hour  
On-Demand Windows base pricing: 0.034 USD per Hour
- t2.medium**  
Family: t2 2 vCPU 4 GiB Memory Current generation: true  
On-Demand Linux base pricing: 0.0496 USD per Hour  
On-Demand Windows base pricing: 0.0676 USD per Hour  
On-Demand RHEL base pricing: 0.1096 USD per Hour  
On-Demand SUSE base pricing: 0.1496 USD per Hour

On the right side of the console, there is a "Browse more AMIs" button and a "Free tier eligible" dropdown menu.

4. Terraform Plan command generates execution plan that shows the changes that Terraform will make to the Infrastructure  
Terraform 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.

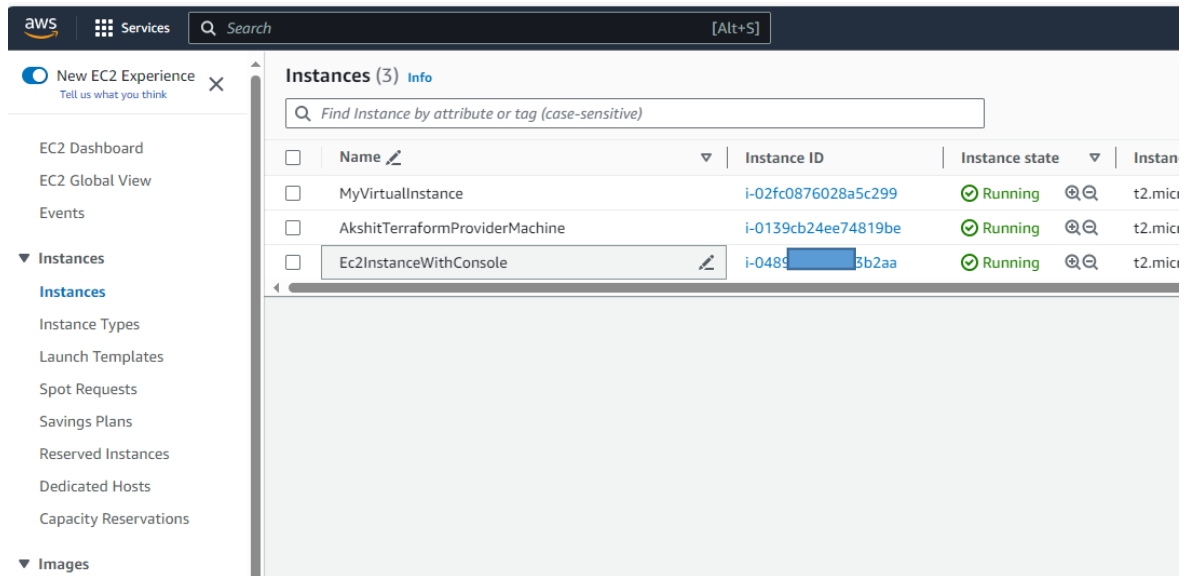
Enter a value: yes

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.
```

CLOUD & DEVOPS

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.

CLOUD & DEVOPS

```
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

EC2 Dashboard

EC2 Global View

Events

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

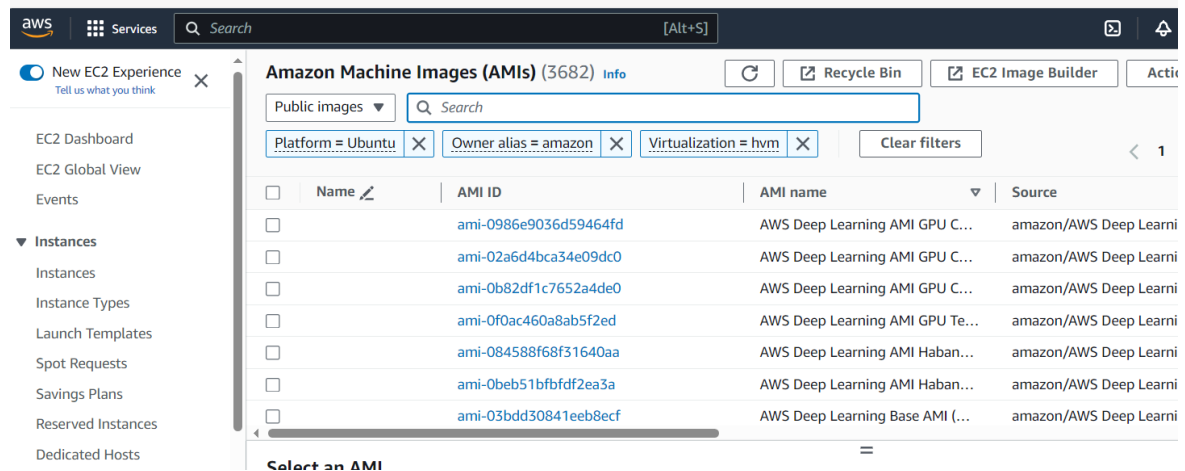
Dedicated Hosts

Instances (3) Info

Find Instance by attribute or tag (case-sensitive)

	Name	Instance ID	Instance state
<input type="checkbox"/>	MyVirtualInstance	i-02fc0876028a5c299	Running
<input type="checkbox"/>	AkshitTerraformProviderMachine	i-0139cb24ee74819be	Running
<input type="checkbox"/>	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 Terraform 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 whose 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

```
MINGW64/c/Users/admin/AkshiTerraformFiles
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"
    }
}
```



## CLOUD &amp; DEVOPS

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.myc2: Destroying... [id=i-07dfcba22ce866d51]  
aws\_instance.myc2Instance: Creating...  
aws\_instance.myc2: Still destroying... [id=i-07dfcba22ce866d51, 10s elapsed]  
aws\_instance.myc2Instance: Still creating... [10s elapsed]  
aws\_instance.myc2Instance: Still creating... [20s elapsed]  
aws\_instance.myc2: Still destroying... [id=i-07dfcba22ce866d51, 20s elapsed]  
aws\_instance.myc2: Still destroying... [id=i-07dfcba22ce866d51, 30s elapsed]  
aws\_instance.myc2Instance: Still creating... [30s elapsed]  
aws\_instance.myc2: Destruction complete after 31s  
aws\_instance.myc2Instance: Creation complete after 32s [id=i-0f0bf1e3dc8ce4f20]

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

admin@DESKTOP-9UJRCUE MINGW64 ~/AkshiTerraformFiles

Name	Instance ID	Instance state	Instance type	Status
MyVirtualInstance	i-02fc0876028a5c299	Running	t2.micro	2
AkshiTerraformProviderMachine	i-0139cb24ee74819be	Running	t2.micro	2
Ec2InstanceWithConsole	i-07dfcba22ce866d51	Terminated	t2.micro	-
Ec2InstanceWithConsole	i-0f0bf1e3dc8ce4f20	Running	t2.micro	2

Instance: i-0f0bf1e3dc8ce4f20 (Ec2InstanceWithConsole)		
Platform	AMI ID	Monitoring
Amazon Linux (Inferred)	ami-03d294e37a4820c21	disabled
Platform details	AMI name	Termination protection
Linux/UNIX	amzn2-ami-kernel-5.10-hvm-2.0.20230926.0-x86_64-gp2	Disabled
Stop protection	Launch time	AMI location
Disabled	Fri Oct 13 2023 09:04:16 GMT+0530 (India Standard Time) (13 minutes)	amazon/amzn2-ami-kx86_64-gp2