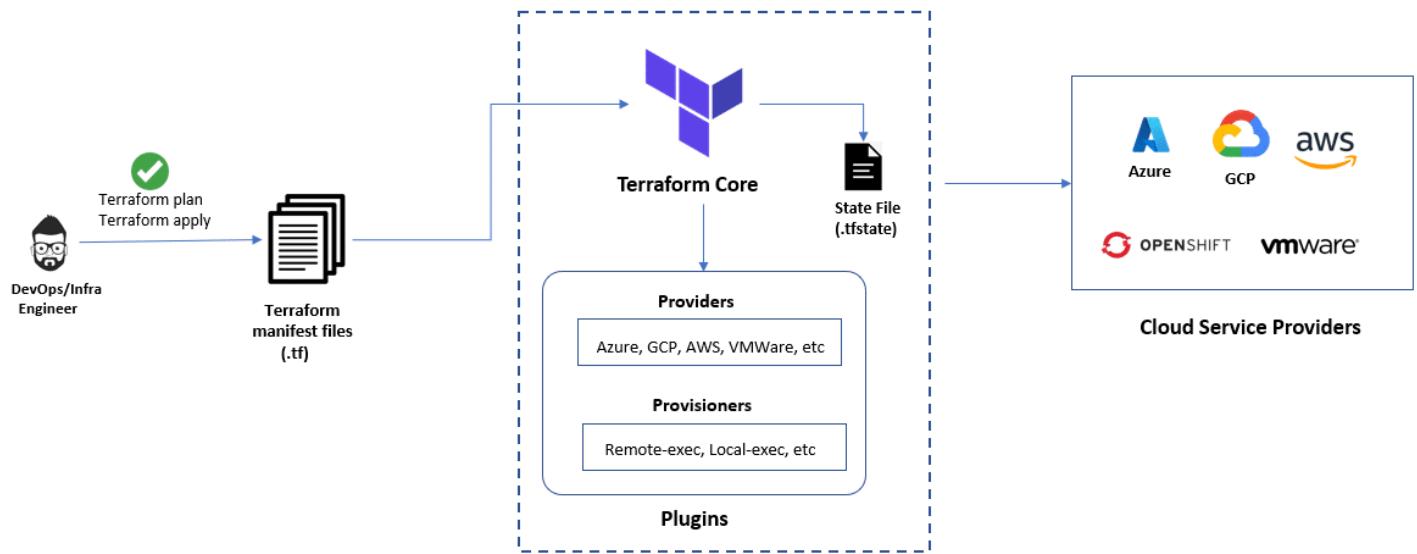


1. Create an Ec2 instance with 8080 port number using terraform.

Terraform: Terraform is an open-source infrastructure as code (IaC) tool developed by HashiCorp. It enables users to define and provision infrastructure resources in a declarative configuration language, allowing them to describe the desired state of their infrastructure and then automatically create and manage those resources.

Terraform is widely used in DevOps and cloud computing environments to automate the provisioning and management of infrastructure, making it easier to scale and maintain complex systems.

Terraform Architecture



- Install yum-config-manager to manage your repositories.
sudo yum install -y yum-utils
- Use yum-config-manager to add the official HashiCorp Linux repository.
sudo yum-config-manager --add-repo
<https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo>
- Install Terraform from the new repository.
sudo yum -y install terraform
- Verify the installation
Terraform --version

Instances | EC2 | ap-south-1 | EC2 Instance Connect | ap-south-1 | Install Terraform | Terraform | HashiCorp

developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

Inbox (1,251) - ashok... Essay on Distribution... 3 Gmail YouTube Maps All Bookmarks

Terraform Install Tutorials Documentation Registry Try Cloud

Search

On this page:

- Install Terraform
- Install Terraform
- Verify the installation
- Quick start tutorial
- Next Steps

Install Terraform

Manual installation Homebrew on OS X Chocolatey on Windows **Linux**

HashiCorp officially maintains and signs packages for the following Linux distributions.

Ubuntu/Debian CentOS/RHEL Fedora **Amazon Linux**

Install `yum-config-manager` to manage your repositories.

```
$ sudo yum install -y yum-utils
```

Use `yum-config-manager` to add the official HashiCorp Linux repository.

```
$ sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/A
```

Install Terraform from the new repository.

```
$ sudo yum -y install terraform
```

We use cookies & other similar technology to collect data to improve your experience on our site, as described in our [Privacy Policy](#) and [Cookie Policy](#). Manage Preferences **ACCEPT**

. IAM user

Create access key | IAM | Global | EC2 Instance Connect | ap-south-1 | Install Terraform | Terraform | HashiCorp

Inbox (1,251) - ashok... Essay on Distribution... 3 Gmail YouTube Maps All Bookmarks

IAM Services Search [Alt+S]

aws Global Ashok

Access key created
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

IAM > Users > ashok > Create access key

Step 1
[Access key best practices & alternatives](#)

Step 2 - optional
[Set description tag](#)

Step 3
[Retrieve access keys](#)

Retrieve access keys [Info](#)

Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIA2KHQRDVOZ2CTTMV	***** Show

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

Download .csv file Done

CloudShell Feedback

© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Type here to search

- `terraform --version`

```
[root@ip-172-31-46-33 ~]# terraform --version
Terraform v1.6.4
on linux_amd64
[root@ip-172-31-46-33 ~]# mkdir ec2
[root@ip-172-31-46-33 ~]# cd ec2/
[root@ip-172-31-46-33 ec2]# vi ec2.tf
```

i-0dd3cf711ec900450 (terraform)
PublicIPs: 15.206.172.228 PrivateIPs: 172.31.46.33

. vi ec2.tf

```
provider "aws" {
  access_key = "AKIA2KHQRYDVL7JFY5PD"
  secret_key = "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"
  region     = "ap-south-1"
}

variable "privatekey" {
  default = "my-keypair.pem"
}

resource "aws_security_group" "allow-ssh-and-8080" {
  name            = "allow-ssh-and-8080"
  description     = "Allow SSH and port 8080 inbound traffic"

  ingress {
    from_port    = 22
    to_port      = 22
    protocol     = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port    = 8080
    to_port      = 8080
    protocol     = "tcp"
  }
}

"main.tf" 74L, 2444B
```

i-0b3ad43d356073807 (ansible-master)
PublicIPs: 13.233.198.163 PrivateIPs: 172.31.37.116

The screenshot shows a browser window with several tabs open. The tabs include "Install Terraform | Terraform | H", "ChatGPT", "Instances | EC2 | ap-south-1", and "EC2 Instance Connect | ap-south-1". The main content area displays a Terraform configuration file:

```

aws {
  egress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

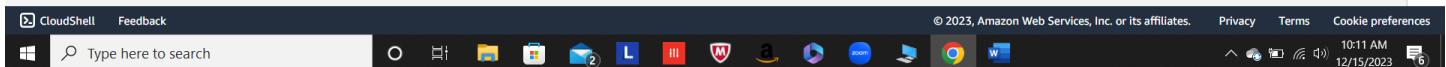
resource "aws_instance" "web" {
  ami           = "ami-0d92749d46e71c34c"
  instance_type = "t2.micro"
  key_name      = "my-keypair"
  security_groups = ["allow-ssh-and-8080"]

  user_data = <<-EOF
    #!/bin/bash
    echo "Hello, World!" > index.html
    nohup python -m SimpleHTTPServer 8080 &
  EOF

  tags = {
    Name = "project-3"
  }
}

```

Below the code, it shows the instance ID: i-0b3ad43d356073807 (ansible-master) and its public and private IP addresses: PublicIPs: 13.233.198.163 PrivateIPs: 172.31.37.116.



- **Terraform init:** Initializes a Terraform working directory. This command is used to set up the backend, download necessary plugins, and prepare the directory for Terraform configuration.
- **Terraform validate:** Validates the syntax and format of the Terraform configuration files. It checks for errors without actually executing the configuration.
- **Terraform plan:** Generates an execution plan, showing what actions Terraform will take to create or modify infrastructure. This command helps you understand the changes before applying them.
- **Terraform apply:** Applies the changes specified in the Terraform configuration. This command creates or modifies infrastructure according to the execution plan generated by **terraform plan**.
- **Terraform destroy:** Destroys the Terraform-managed infrastructure. It deletes all resources created by Terraform based on the configuration.

- Terraform init

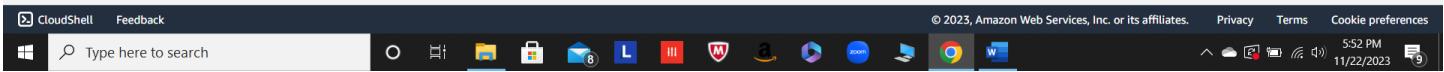
```
[root@ip-172-31-46-33 ec2]# terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.26.0

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.
[root@ip-172-31-46-33 ec2]#
```

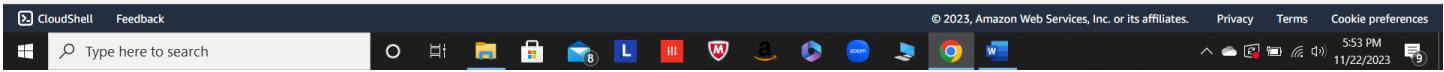
i-0dd3cf711ec900450 (terraform)
PublicIPs: 13.233.97.252 PrivateIPs: 172.31.46.33



- Terraform validate

```
[root@ip-172-31-46-33 ec2]# terraform validate
Success! The configuration is valid.
[root@ip-172-31-46-33 ec2]#
```

i-0dd3cf711ec900450 (terraform)
PublicIPs: 13.233.97.252 PrivateIPs: 172.31.46.33



- Terraform plan

```
[root@ip-172-31-46-33 ec2]# terraform plan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.project-3 will be created
+ resource "aws_instance" "project-3" {
  + ami                               = "ami-0a6ed68999bf32a5"
  + 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)"
  + 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)"

i-0dd3cf711ec900450 (terraform)
PublicIPs: 13.233.97.252 PrivateIPs: 172.31.46.33
```

```
+ "0.0.0.0/0",
]
+ description      = ""
+ from_port       = 8080
+ ipv6_cidr_blocks = []
+ prefix_list_ids = []
+ protocol        = "tcp"
+ security_groups = []
+ self             = false
+ to_port          = 8080
},
]
+ name              = "allow-ssh-and-8080"
+ name_prefix       = "(known after apply)"
+ owner_id          = "(known after apply)"
+ revoke_rules_on_delete = false
+ tags_all          = "(known after apply)"
+ vpc_id            = "(known after apply"
}

Plan: 2 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
```

- Terraform apply --auto-approve

```
[root@ip-172-31-46-33 ec2]# terraform apply --auto-approve
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.project-3 will be created
+ resource "aws_instance" "project-3" {
  + ami                               = "ami-0a6ed68999bf32a5"
  + 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)"
  + 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)"

i-0dd3cf711ec900450 (terraform)
PublicIPs: 13.233.97.252 PrivateIPs: 172.31.46.33
```

```
+ from_port           = 8080
+ ipv6.cidr_blocks   = []
+ prefix_list_ids    = []
+ protocol           = "tcp"
+ security_groups    = []
+ self                = false
+ to_port              = 8080
},
],
+ name                 = "allow-ssh-and-8080"
+ name_prefix          = "(known after apply)"
+ owner_id              = "(known after apply)"
+ revoke_rules_on_delete = false
+ tags_all              = "(known after apply)"
+ vpc_id                = "(known after apply"
)

Plan: 2 to add, 0 to change, 0 to destroy.
aws_instance.project-3: Creating...
aws_security_group.allow-ssh-and-8080: Creating...
aws_security_group.allow-ssh-and-8080: Creation complete after 2s [id=sg-0dcb166b9dff3dad4]
aws_instance.project-3: Still creating... [10s elapsed]
aws_instance.project-3: Still creating... [20s elapsed]
aws_instance.project-3: Creation complete after 22s [id=i-05da409bf32b7677a]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

i-0dd3cf711ec900450 (terraform)
PublicIPs: 13.233.97.252 PrivateIPs: 172.31.46.33
```

- I have created a single terraform file for the whole infrastructure. The file will have '.tf' as extension. After describing the provider which we are going to use mentioning region, I have created the security group which allow the ports 8080 and 22 to perform the specific work. Port 8080 -Hello World, Port 22 - SSH.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with links like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, and Elastic Block Store (Volumes, Snapshots, Lifecycle Manager). The main content area displays a table of instances. One instance, named 'project' with ID i-0638c16ce31e817ee, is selected and highlighted in blue. The table columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. Below the table, a modal window titled 'Instance: i-0638c16ce31e817ee (project)' shows detailed information such as Instance ID (i-0638c16ce31e817ee), Public IPv4 address (43.205.231.158), Private IPv4 addresses (172.31.43.56), Instance state (Running), Public IPv4 DNS (ec2-43-205-231-1.compute.amazonaws.com), and Hostname type (ip-172-31-17-56). At the bottom of the page, there's a CloudShell feedback bar and a Windows taskbar.

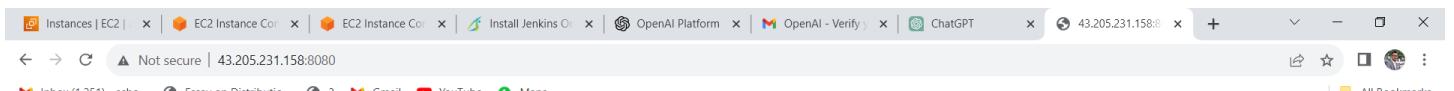
This screenshot is identical to the one above, but it shows the security group rules for the selected instance. The modal window now displays the 'INBOUND rules' section, which contains two entries: one for port 8080 (TCP) and another for port 22 (TCP), both allowing traffic from 0.0.0.0/0. There's also an 'OUTBOUND rules' section with a 'Filter rules' search bar. The rest of the interface is the same, including the sidebar and the CloudShell feedback bar at the bottom.

CloudShell Feedback Type here to search © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 5:46 PM 11/22/2023

```
# Amazon Linux 2
# AL2 End of Life is 2025-06-30.
# A newer version of Amazon Linux is available!
# Amazon Linux 2023, GA and supported until 2028-03-15.
# https://aws.amazon.com/linux/amazon-linux-2023/
7 package(s) needed for security, out of 14 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-43-56 ~]$
```

i-0638c16ce31e817ee (project)
Public IPs: 43.205.231.158 Private IPs: 172.31.43.56

CloudShell Feedback Type here to search © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 6:30 PM 11/22/2023



Hello, World!



2. Run an Ansible Playbook with Terraform.

The screenshot shows a browser window with the AWS CloudShell interface. The terminal session is running on an EC2 instance (i-0b3ad43d356073807). The user has run the command `ls`, which lists several files: `ansible.cfg`, `host`, `main.tf`, `my-keypair.pem`, `playbook.yml`, `terraform.tfstate`, and `terraform.tfstate.backup`. The terminal prompt ends with `[root@ip-172-31-37-116 Terraform_Ansible_cicd]#`.

Below the terminal, the AWS CloudShell interface displays the instance details: `i-0b3ad43d356073807 (ansible-master)`. It also shows the public and private IP addresses: `PublicIPs: 43.205.118.79 PrivateIPs: 172.31.37.116`.

The system tray at the bottom of the screen shows various icons for file explorer, task manager, and network status, along with the date and time: `3:54 PM 12/14/2023`.

- vi host (empty file)

The screenshot shows a browser window with the AWS CloudShell interface. The terminal session is running on an EC2 instance (i-0b3ad43d356073807). The user has run the command `vi host`, which creates an empty file named `host`. The terminal prompt ends with `[root@ip-172-31-37-116 Terraform_Ansible_cicd]#`.

Below the terminal, the AWS CloudShell interface displays the instance details: `i-0b3ad43d356073807 (ansible-master)`. It also shows the public and private IP addresses: `PublicIPs: 43.205.118.79 PrivateIPs: 172.31.37.116`.

The system tray at the bottom of the screen shows various icons for file explorer, task manager, and network status, along with the date and time: `3:57 PM 12/14/2023`.

- vi main.tf

```

provider "aws" {
  access_key = "AKIA2KHQRYDVL7JFY5PD"
  secret_key = "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"
  region     = "ap-south-1"
}

variable "privatekey" {
  default = "my-keypair.pem"
}

resource "aws_security_group" "allow-ssh-and-8080" {
  name            = "allow-ssh-and-8080"
  description     = "Allow SSH and port 8080 inbound traffic"

  ingress {
    from_port    = 22
    to_port      = 22
    protocol     = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port    = 8080
    to_port      = 8080
    protocol     = "tcp"
  }
}

i-0b3ad43d356073807 (ansible-master)
PublicIPs: 43.205.118.79 PrivateIPs: 172.31.37.116

```

```

egress {
  from_port    = 0
  to_port      = 0
  protocol     = "-1"
  cidr_blocks = ["0.0.0.0/0"]
}

resource "aws_instance" "web" {
  ami           = "ami-0d92749d46e71c34c"
  instance_type = "t2.micro"
  key_name      = "my-keypair"
  security_groups = ["allow-ssh-and-8080"]

  user_data = <<-EOF
    #!/bin/bash
    echo "Hello, World!" > index.html
    nohup python -m SimpleHTTPServer 8080 &
  EOF

  tags = {
    Name = "project-3"
  }
}

i-0b3ad43d356073807 (ansible-master)
PublicIPs: 43.205.118.79 PrivateIPs: 172.31.37.116

```

```

tags = {
  Name = "project-3"
}

provisioner "local-exec" {
  command = "echo ${aws_instance.web.public_ip} >> /root/Terraform_Ansible_cicd/host"
}

provisioner "remote-exec" {
  inline = [
    "echo 'build ssh connection' "
  ]
}

connection {
  host = self.public_ip
  type = "ssh"
  user = "ec2-user"
  private_key = file("/root/Terraform_Ansible_cicd/my-keypair.pem")
}

provisioner "local-exec" {
  command = "ansible-playbook -i '${aws_instance.web.public_ip}', --private-key ${var.privatekey} playbook.yml"
}

```

i-0b3ad43d356073807 (ansible-master)
Public IPs: 43.205.118.79 Private IPs: 172.31.37.116

- **Provider:** In Terraform, a provider is a plugin that enables Terraform to interact with a specific cloud or infrastructure platform. Providers are responsible for translating the desired state described in your Terraform configuration into API calls to create, update, or delete resources on the target platform. Providers handle the communication between Terraform and the underlying infrastructure.
- **Variable:** You declare variables using the **variable** block in your Terraform configuration. Variables can have a name, a type, an optional default value, and other constraints.
- **Security group:** In the context of cloud computing, a security group is a fundamental component used to control inbound and outbound traffic to resources such as virtual machines (VMs) or instances in a network. Security groups act as virtual firewalls that define rules governing the traffic flow to and from these resources. They are commonly used in cloud platforms like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). Security groups play a crucial role in ensuring the security and isolation of resources within cloud environments, and they are an essential part of infrastructure as code (IaC) practices using tools like Terraform.
- **Aws instance:** In Amazon Web Services (AWS), an instance refers to a virtual server in the cloud. Instances are the fundamental building blocks of computing resources on AWS, providing on-demand and scalable compute capacity. AWS instances are fundamental to the AWS cloud computing model, providing the compute capacity needed to run applications and services. They are a central component in building scalable, flexible, and resilient cloud architectures.

- **Provisioner:** In Terraform, provisioners are a mechanism for defining and executing scripts or commands on a resource as part of the resource creation or update process. Provisioners are typically used to perform tasks such as initializing software, configuring resources, or running custom scripts on instances after they have been created. There are two main types of provisioners in Terraform: local-exec and remote-exec.

Local-Exec Provisioner: The **local-exec** provisioner allows you to run commands on the machine running Terraform, rather than on the resource being managed. This can be useful for tasks that don't require execution on the target resource, such as local script execution or interacting with local tools.

Remote-Exec Provisioner: The **remote-exec** provisioner connects to the created resource via SSH and executes commands or scripts on that resource. This is commonly used for configuring software on remote instances.

Note:

- While provisioners can be helpful, it's often preferable to use configuration management tools (e.g., Ansible, Chef, Puppet) for complex software configuration tasks. Terraform provisioners are more suitable for simple tasks or situations where external tools are not applicable.
- Using provisioners should be done cautiously, as they can introduce challenges in terms of idempotence, reliability, and maintainability. It's recommended to consider infrastructure as code best practices and explore alternative solutions when possible.
- Terraform also supports other provisioners, such as file, template, and null provisioners, which offer additional capabilities for managing resources.
- **Ansible.cfg:** The **ansible.cfg** file is a configuration file used by Ansible to define settings and preferences for running Ansible commands and playbooks. It is not directly related to Terraform but is used when you are managing infrastructure with Ansible, which is a complementary tool to Terraform for configuration management and automation.

In the context of using Ansible with Terraform, you might have an Ansible configuration file (**ansible.cfg**) to specify settings such as the location of inventory files, where to find roles, default user, and other options.

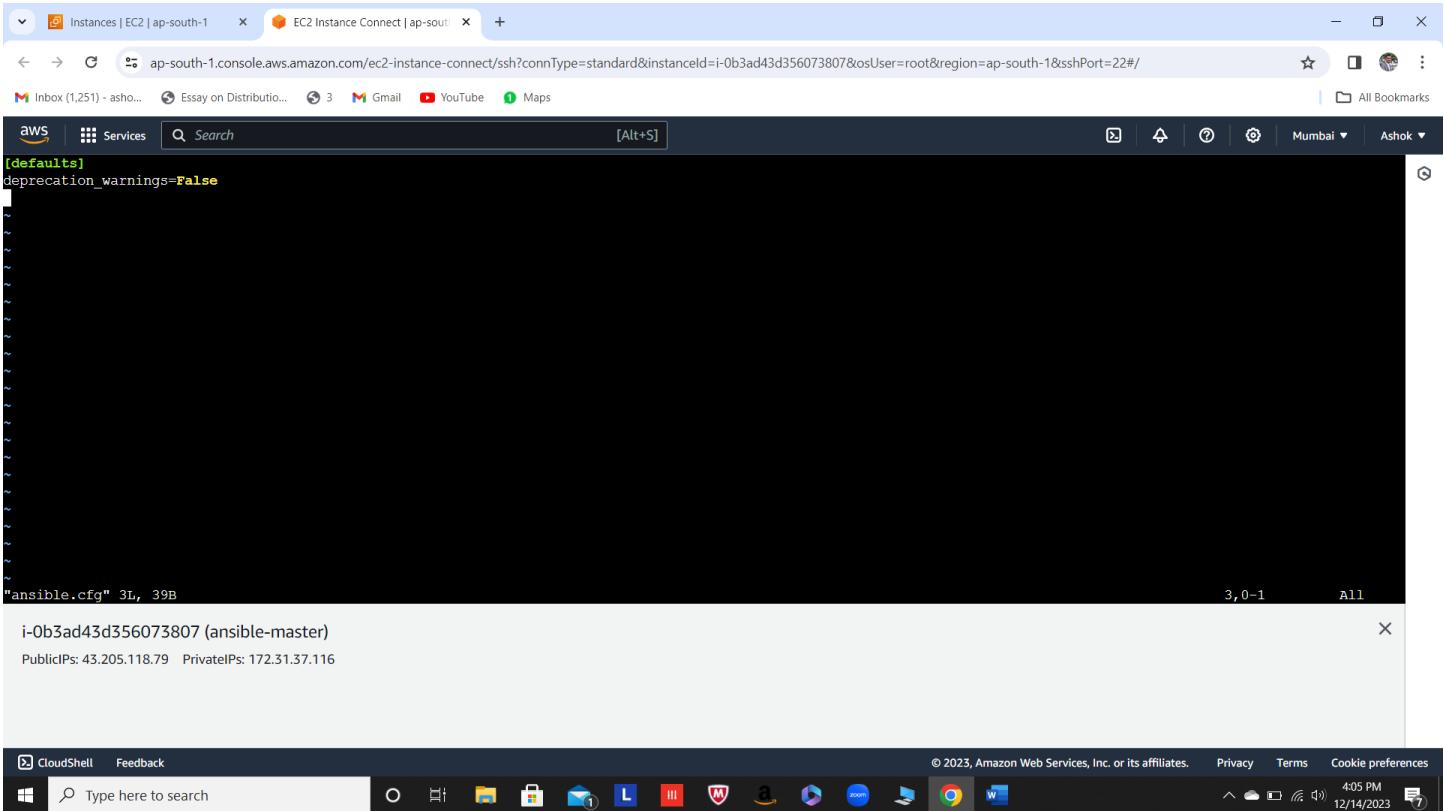
- **Host:** In Ansible, the inventory file (often referred to as the "hosts file") is a crucial component that defines the target hosts or nodes on which Ansible should operate. The inventory file is used to specify the hosts and groups of hosts that Ansible will manage. The default name for the inventory file is **hosts**.

Usage with Terraform:

When integrating Ansible with Terraform, Terraform can generate an inventory file dynamically based on the infrastructure it provisions. The generated inventory file can then be used by Ansible to configure and manage the provisioned resources.

Terraform can output the necessary information (IP addresses, SSH keys, etc.) to create a dynamic inventory script or file that Ansible can use.

- vi ansible.cfg



```
[defaults]
deprecation_warnings=False
```

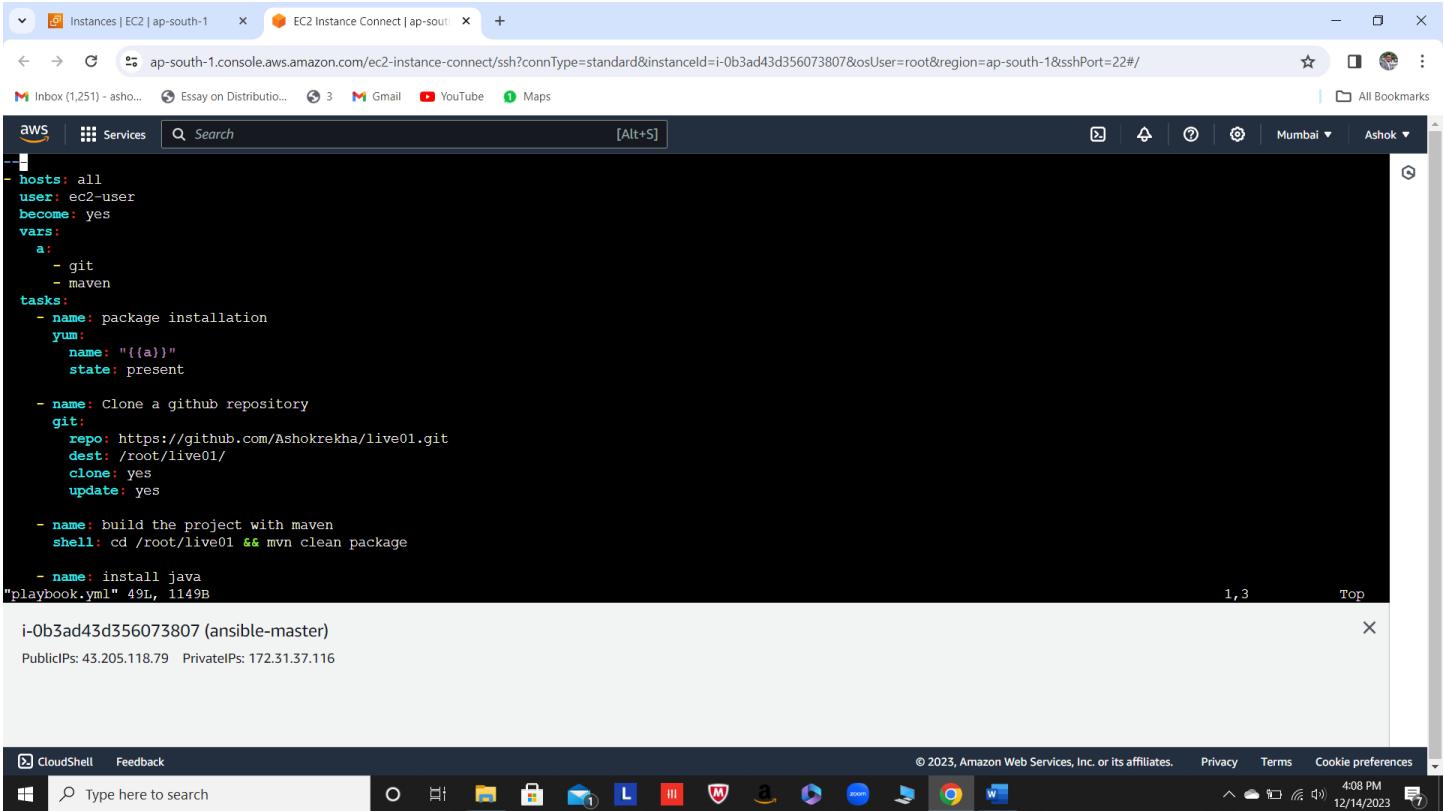
"ansible.cfg" 3L, 39B 3, 0-1 All

i-0b3ad43d356073807 (ansible-master)

PublicIPs: 43.205.118.79 PrivateIPs: 172.31.37.116

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences Type here to search 405 PM 12/14/2023

- vi playbook.yml (Ansible playbook)



```
hosts: all
user: ec2-user
become: yes
vars:
  a:
    - git
    - maven
tasks:
  - name: package installation
    yum:
      name: "{{a}}"
      state: present

  - name: Clone a github repository
    git:
      repo: https://github.com/Ashokrekha/live01.git
      dest: /root/live01/
      clone: yes
      update: yes

  - name: build the project with maven
    shell: cd /root/live01 && mvn clean package

  - name: install java
playbook.yml" 49L, 1149B 1, 3 Top
```

i-0b3ad43d356073807 (ansible-master)

PublicIPs: 43.205.118.79 PrivateIPs: 172.31.37.116

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences Type here to search 408 PM 12/14/2023

```

aws | Services | Search [Alt+S]
- name: install java
  yum:
    name: java
    state: present

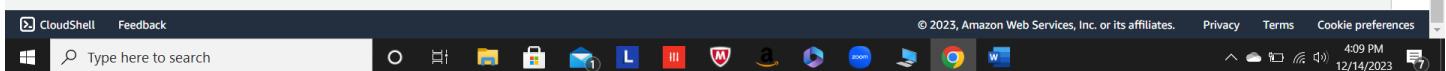
- name: directory creation
  file:
    path: /root/tomcat
    state: directory

- name: download & unarchive tomcat9
  unarchive:
    src: https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.83/bin/apache-tomcat-9.0.83.tar.gz
    dest: /root/tomcat
    remote_src: yes

- name: Run Tomcat
  shell: nohup ./startup.sh
  args:
    chdir: /root/tomcat/apache-tomcat-9.0.83/bin
- name: copy the WAR file to tomcat webapps directory
  copy:
    src: /root/live01/target/live.war
    dest: /root/tomcat/apache-tomcat-9.0.83/webapps/
    remote_src: yes

```

i-0b3ad43d356073807 (ansible-master)
Public IPs: 43.205.118.79 Private IPs: 172.31.37.116



- terraform apply --auto-approve

```

[root@ip-172-31-37-116 Terraform_Ansible_cicd]# terraform apply --auto-approve
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
Terraform will perform the following actions:

# aws_instance.web will be created
+ resource "aws_instance" "web" {
  + ami                               = "ami-0d92749d46e71c34c"
  + 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)
  + 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)
}

i-0b3ad43d356073807 (ansible-master)  
Public IPs: 43.205.118.79 Private IPs: 172.31.37.116

```



```
# aws_security_group.allow-ssh-and-8080 will be created
+ resource "aws_security_group" "allow-ssh-and-8080" {
+   arn           = (Known after apply)
+   description    = "Allow SSH and port 8080 inbound traffic"
+   egress         =
+   {
+     + cidr_blocks  = [
+       + "0.0.0.0/0",
+     ]
+     + description   = ""
+     + from_port     = 0
+     + ipv6_cidr_blocks = []
+     + prefix_list_ids = []
+     + protocol      = "-1"
+     + security_groups = []
+     + self          = false
+     + to_port        = 0
+   },
+   id           = (known after apply)
+   ingress       =
+   {
+     + cidr_blocks  = [
+       + "0.0.0.0/0",
+     ]
+   }
}

i-0b3ad43d356073807 (ansible-master)
Public IPs: 43.205.118.79 Private IPs: 172.31.37.116
```

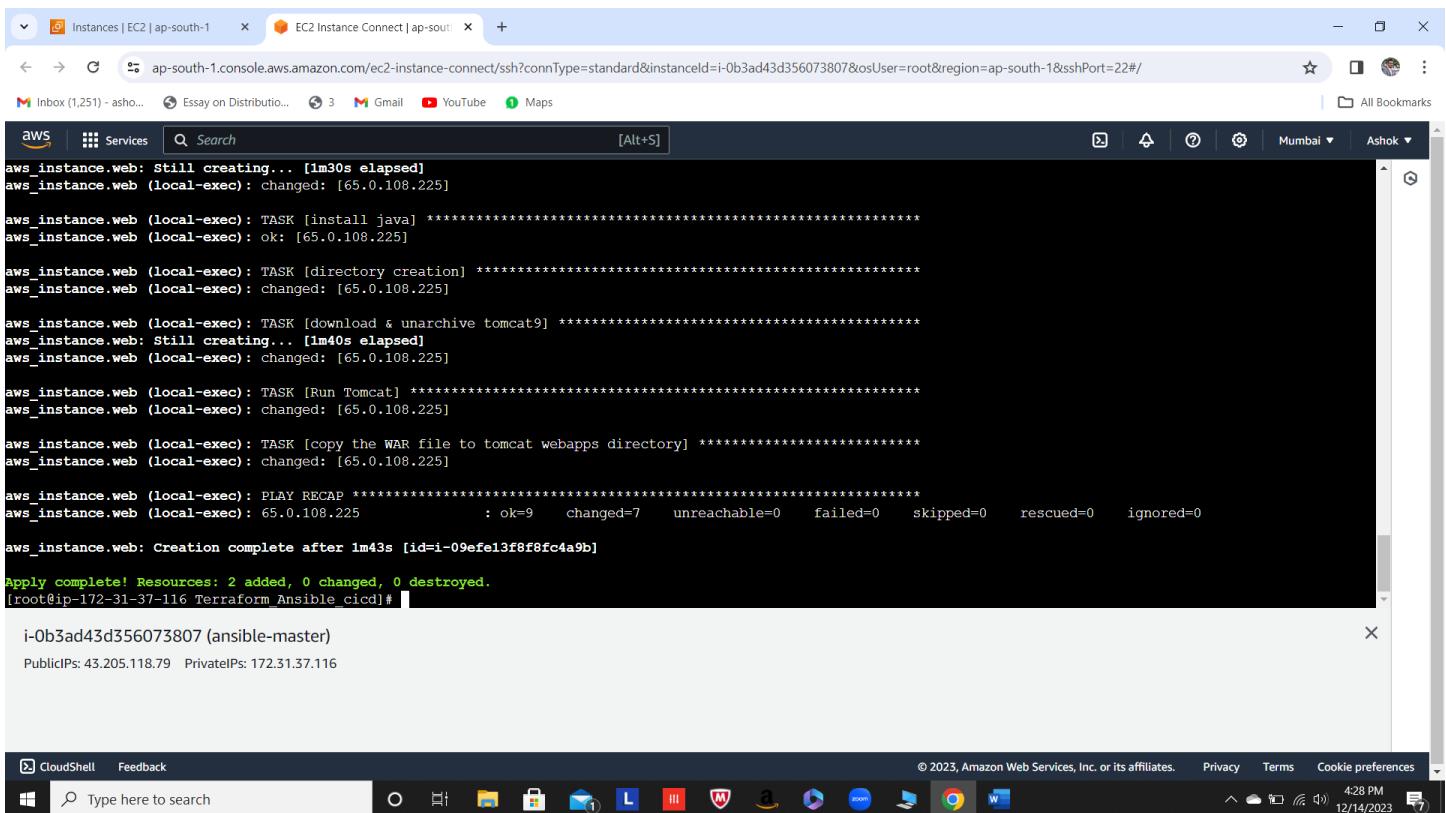
```
aws_instance.web (local-exec): PLAY [all] *****
aws_instance.web (local-exec): TASK [Gathering Facts] *****
The authenticity of host '65.0.108.225' (65.0.108.225) can't be established.
ED25519 key fingerprint is SHA256:kvXRerAerwW9DStqZUOhCpdZSLA5KDBTBa8txbI.
ED25519 key fingerprint is MD5:e0:78:38:6e:cd:32:bd:46:de:e1:1b:48:6e:10:21:75.
Are you sure you want to continue connecting (yes/no)? yes
aws_instance.web: Still creating... [40s elapsed]
aws_instance.web (local-exec): [WARNING]: Platform linux on host 65.0.108.225 is using the discovered Python
aws_instance.web (local-exec): interpreter at /usr/bin/python, but future installation of another Python
aws_instance.web (local-exec): interpreter could change the meaning of that path. See https://docs.ansible.com
aws_instance.web (local-exec): /ansible-core/2.11/reference_appendices/interpreter_discovery.html for more
aws_instance.web (local-exec): information.
aws_instance.web (local-exec): ok: [65.0.108.225]

aws_instance.web (local-exec): TASK [package installation] *****
aws_instance.web: Still creating... [50s elapsed]
aws_instance.web: Still creating... [1m0s elapsed]
aws_instance.web: Still creating... [1m10s elapsed]
aws_instance.web (local-exec): changed: [65.0.108.225]

aws_instance.web (local-exec): TASK [Clone a github repository] *****
aws_instance.web (local-exec): changed: [65.0.108.225]

aws_instance.web (local-exec): TASK [build the project with maven] *****
aws_instance.web: Still creating... [1m20s elapsed]

i-0b3ad43d356073807 (ansible-master)
Public IPs: 43.205.118.79 Private IPs: 172.31.37.116
```



```
aws_instance.web: Still creating... [1m30s elapsed]
aws_instance.web (local-exec): changed: [65.0.108.225]

aws_instance.web (local-exec): TASK [install java] ****
aws_instance.web (local-exec): ok: [65.0.108.225]

aws_instance.web (local-exec): TASK [directory creation] ****
aws_instance.web (local-exec): changed: [65.0.108.225]

aws_instance.web (local-exec): TASK [download & unarchive tomcat9] ****
aws_instance.web: Still creating... [1m40s elapsed]
aws_instance.web (local-exec): changed: [65.0.108.225]

aws_instance.web (local-exec): TASK [Run Tomcat] ****
aws_instance.web (local-exec): changed: [65.0.108.225]

aws_instance.web (local-exec): TASK [copy the WAR file to tomcat webapps directory] ****
aws_instance.web (local-exec): changed: [65.0.108.225]

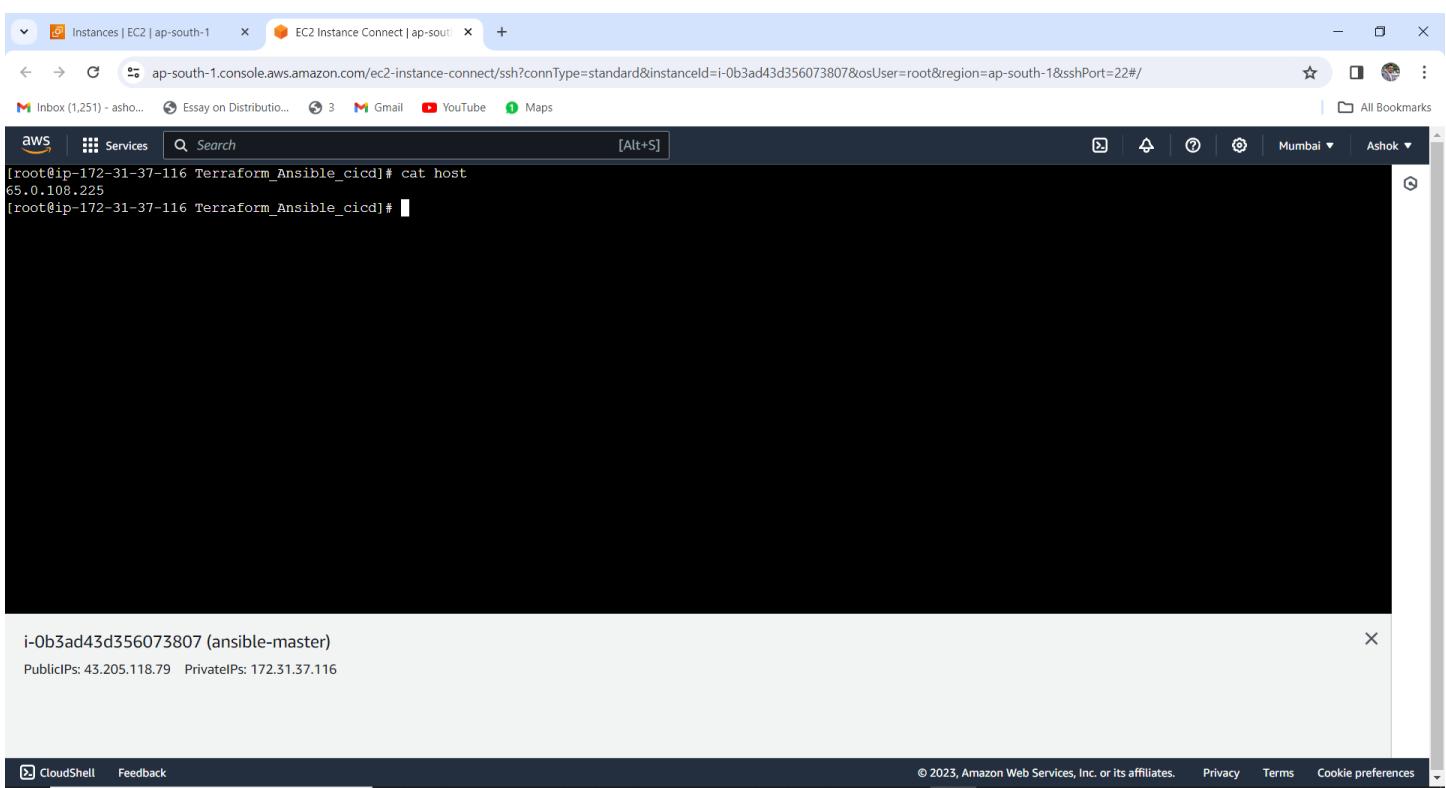
aws_instance.web (local-exec): PLAY RECAP ****
aws_instance.web (local-exec):   65.0.108.225 : ok=9    changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

aws_instance.web: Creation complete after 1m43s [id=i-09efef13f8f8fc4a9b]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[root@ip-172-31-37-116 Terraform_Ansible_ci_cd]#
```

i-0b3ad43d356073807 (ansible-master)
Public IPs: 43.205.118.79 Private IPs: 172.31.37.116

. cat host (Auto generated public ip)



```
[root@ip-172-31-37-116 Terraform_Ansible_ci_cd]# cat host
65.0.108.225
[root@ip-172-31-37-116 Terraform_Ansible_ci_cd]#
```

i-0b3ad43d356073807 (ansible-master)
Public IPs: 43.205.118.79 Private IPs: 172.31.37.116

- Now check created instance “project-3” and playbook.yml output

Instances (1/5) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
project-3	i-021ae5813a09ae13f	Terminated	t2.micro	-	No alarms	ap-south-1b	-
project-3	i-09efe13f8f8fc4a9b	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1b	ec2-65-0-1
project-3	i-035f0e66d28c46a26	Terminated	t2.micro	-	No alarms	ap-south-1b	-
ansible-master	i-0b3ad43d356073807	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1a	ec2-43-20
ansible-node	i-0ade29123cc026f33	Stopped	t2.micro	-	No alarms	ap-south-1a	-

Instance: i-09efe13f8f8fc4a9b (project-3)

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

Instance summary

Instance ID i-09efe13f8f8fc4a9b (project-3)	Public IPv4 address 65.0.108.225 [open address]	Private IPv4 addresses 172.31.13.146
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-65-0-108-225.ap-south-1.compute.amazonaws.com [open address]
Hostname type	Private IP DNS name (IPv4 only)	

Instances (1/5) Info

Name	Security group rule ID	Port range	Protocol	Source	Security groups
-	sgr-07dc06e399cf7c244	22	TCP	0.0.0.0/0	allow-ssh-and-8l
-	sgr-09911663ae791549e	8080	TCP	0.0.0.0/0	allow-ssh-and-8l

Inbound rules

Outbound rules

The screenshot shows the AWS EC2 Instance Connect connection dialog. At the top, there are tabs for 'EC2 Instance Connect' (selected), 'Session Manager', 'SSH client', and 'EC2 serial console'. Below these, the 'Instance ID' is listed as 'i-09efe13f8f8fc4a9b (project-3)'. The 'Connection Type' section contains two options: 'Connect using EC2 Instance Connect' (selected, with a note about using a browser-based client with a public IPv4 address) and 'Connect using EC2 Instance Connect Endpoint' (with a note about using a browser-based client with a private IPv4 address and a VPC endpoint). Under 'Public IP address', the value '65.0.108.225' is shown. The 'User name' field contains 'root'. A note at the bottom left says: 'Note: In most cases, the default user name, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.' At the bottom right are 'Cancel' and 'Connect' buttons.

The screenshot shows a browser window titled "EC2 Instance Connect | ap-sout" with three tabs open. The active tab displays a terminal session on an Amazon Linux 2 instance. The terminal output includes:

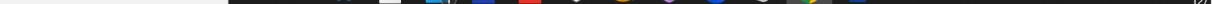
```
[root@ip-172-31-13-146 ~]# ls
live01 tomcat

[root@ip-172-31-13-146 ~]# git -v
git version 2.40.1

[root@ip-172-31-13-146 ~]# mvn -v
Apache Maven 3.0.5 (Red Hat 3.0.5-17)
Maven home: /usr/share/maven
Java version: 17.0.9, vendor: Amazon.com Inc.
Java home: /usr/lib/jvm/java-17-amazon-corretto.x86_64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.10.199-190.747.amzn2.x86_64", arch: "amd64", family: "unix"

[root@ip-172-31-13-146 ~]#
```

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



4:33 PM 12/14/2023

