Terraform Hands-On Setup on AWS EC2:

Step 1: Launch an EC2 Instance

1. Go to AWS Console and sign in.
2. Search for EC2 and open the EC2 Dashboard.
3. Click "Launch Instance".
4. Name the instance

`terraform-server`.

5. Keep default settings, but in Network settings, enable all traffic.
6. Select Amazon Linux 2023 AMI.
7. Click "Launch" (use an existing key pair or create a new one).

Step 2: Connect to EC2 Terminal

1. Go back to your EC2 Dashboard.
2. Select your

`terraform-server` instance.

3. Click "Connect" → "EC2 Instance Connect" → "Connect".
4. You will enter the Linux terminal of your EC2 instance.

Step 3: Install Terraform on Amazon Linux 2023

Run the following commands one by one in the EC2 terminal:

- `sudo su`
- `dnf install -y yum-utils`
- `yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo`
- `dnf install -y terraform`

After installation, verify with:

- `terraform version` (should show `Terraform v1.xx.x`)
- `terraform plan --help` (should give help info)

Step 4: Create an IAM User with Access Keys

1. Go to AWS Console → Search IAM → Open.
2. Click "Users" → "Add User".
3. Name the user

`sit-user`.

4. Select "Access key - Programmatic access".
5. Click "Next" → "Attach permission" → Choose "AdministratorAccess".
6. Keep clicking "Next" until "Create User".
7. After creation, click your user → Go to "Security credentials" tab.
8. Click "Create access key".
9. Choose "Other".
10. Click through until "Create access key".
11. Copy both Access key ID and Secret access key and save them safely.

Step 5: Configure AWS CLI on EC2

Go back to your EC2 terminal and run:

- `aws configure`
  - Enter your AWS Access Key ID
  - Enter your AWS Secret Access Key
  - Default region name: `ap-south-1`
  - Default output format: json

To verify connection:

- `aws s3 ls` (no output/error means it's working)

Step 6: Create Your Terraform Script (main.tf)

In the EC2 terminal:

1. `vi main.tf`
2. Press

`i` to enter insert mode.

3. Paste the following HCL code:

Terraform
```
provider "aws" {
  region = "ap-south-1" # Change to your chosen region [cite: 84, 85]
}

resource "aws_instance" "example" {
  ami           = "ami-0f5ee92e2d63afc18" # Amazon Linux 2023 AMI for
ap-south-1 [cite: 87, 88, 89]
  instance_type = "t2.micro" [cite: 90]
  key_name      = "devops-batch" # Use your actual key pair name
[cite: 91]
  tags = {
    Name = "Terraform EC2" [cite: 92, 93]
  }
}

output "instance_ip" {
  value = aws_instance.example.public_ip [cite: 96, 97]
}
```

4. Replace

`ami` with the correct AMI ID for your region.

5. Replace

`key_name` with the key pair name you used when launching EC2.

Step 7: Deploy Using Terraform

Still in the terminal:

1. Initialize Terraform:

```
terraform init
```

2. Show execution plan:

```
terraform plan
```

3. Apply the configuration:

```
terraform apply
```

4. When prompted "Do you want to perform these actions?", type

`yes`.

    a. The public IP of the created EC2 instance will be shown.

Final Tips

- To destroy created resources:

`terraform destroy` (confirm with `yes`).

- To edit the

`main.tf` file again: `vi main.tf`.

Terraform Notes - Lecture Recap (with Clean Explanations)

Quick Tip (vi editor)

- gg: go to the top of the file
- V: start line-wise visual selection
- G: go to the bottom (select entire file)
- d: delete the selected content
- ggVGd: Deletes the entire file content in `vi` (useful for pasting fresh code).

Why Use Variables in Terraform?

- In production, you might want to reuse resource definitions but change values (instance type, number of instances, tags).
- Hardcoding values is not recommended.
- Terraform variables allow defining dynamic values for reuse and easy management.

Example 1: String Variable (instance_type)

- Used to dynamically control the instance type.
- **Code:**

Terraform
```
provider "aws" {
  region = "us-east-1" [cite: 140, 141]
}

resource "aws_instance" "ec2_example" {
  ami           = "ami-05ffe3c48a9991133" [cite: 143, 144, 145]
  instance_type = var.instance_type [cite: 146]
  tags = {
    Name = "Terraform EC2" [cite: 147, 148]
  }
}

variable "instance_type" {
  description = "Instance type t2.micro" [cite: 151, 152]
  type        = string [cite: 153, 154]
  default     = "t2.micro" [cite: 155, 156]
}
```

- **Explanation:** `var.instance_type` refers to the variable's value. The instance type can be changed by updating the variable, not the full code.
- **Terminal commands:** `terraform init`, `terraform plan`, `terraform apply` (type yes when prompted).

Example 2: Integer Variable (count)

- Used to create multiple instances at once.
- **Code:**

Terraform
```
provider "aws" {
  region = "us-east-1" [cite: 173, 174]
}
```

```
resource "aws_instance" "ec2_example" {
  ami           = "ami-05ffe3c48a9991133" [cite: 176, 177, 178]
  instance_type = "t2.micro" [cite: 179]
  count         = var.instance_count [cite: 180, 181]
  tags = {
    Name = "Terraform EC2" [cite: 182, 183]
  }
}

variable "instance_count" {
  description = "EC2 instance count" [cite: 186, 187]
  type        = number [cite: 188, 189]
  default     = 2 [cite: 190, 191]
}
```

- **Explanation:** `count` allows Terraform to create multiple copies of a resource. The value comes from the

`instance_count` variable.

- **Terminal commands:** `terraform plan`, `terraform apply` (type yes). Terraform will spin up the specified number of EC2 instances.

Example 3: Map Variable (Tags)

- Used to manage multiple key-value pairs like tags.
- **Code:**

Terraform
```
provider "aws" {
  region = "us-east-1" [cite: 207, 208]
}

resource "aws_instance" "ec2_example" {
  ami           = "ami-05ffe3c48a9991133" [cite: 210, 211, 212]
  instance_type = "t2.micro" [cite: 213]
  tags          = var.project_environment [cite: 214, 215]
}
```

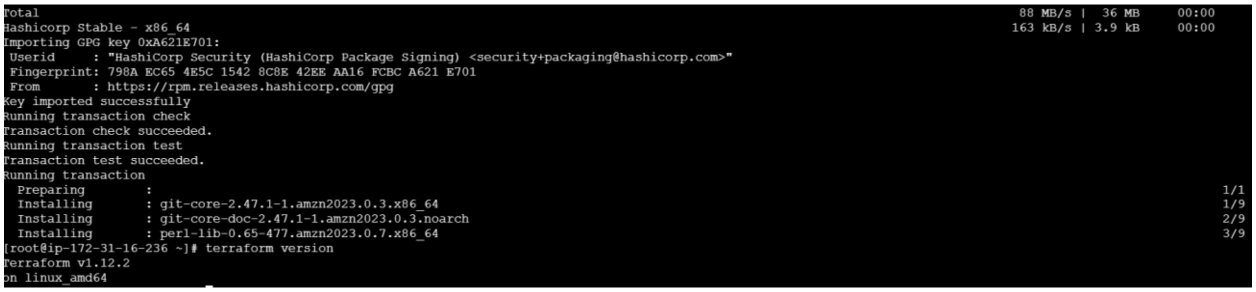
```
variable "project_environment" {
  description = "Project name and environment" [cite: 217, 218]
  type        = map(string) [cite: 219, 220]
  default     = {
    project     = "project-alpha" [cite: 221, 225]
    environment = "dev" [cite: 226]
  }
}
```

- **Explanation:** `map(string)` defines a collection of key-value pairs, useful for assigning multiple tags or grouped values.
- **Terminal commands:** `terraform plan`, `terraform apply` (say yes). The EC2 instance will have `project = project-alpha` and `environment = dev` tags.

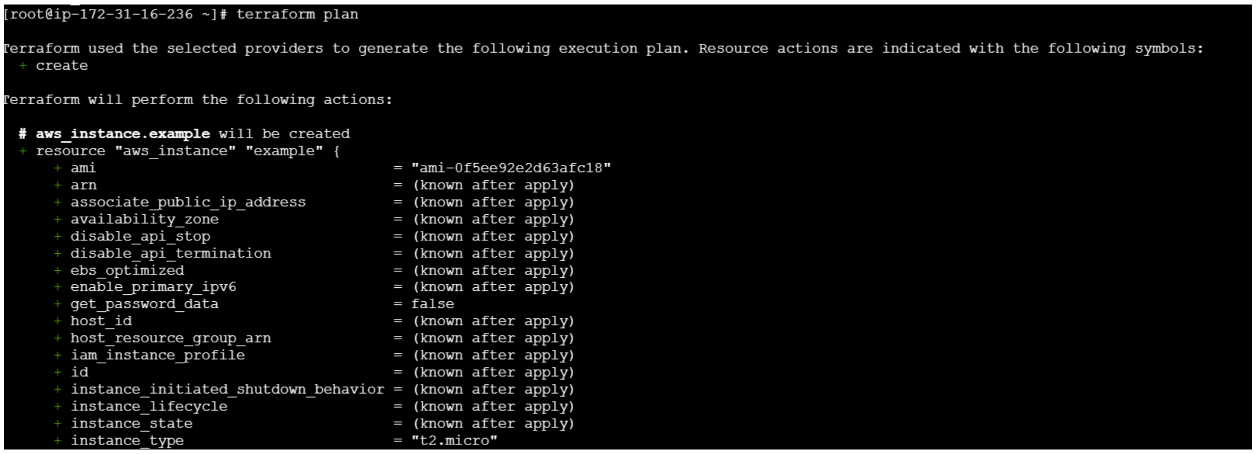
Destroy Resources After Testing

- To remove all created resources:

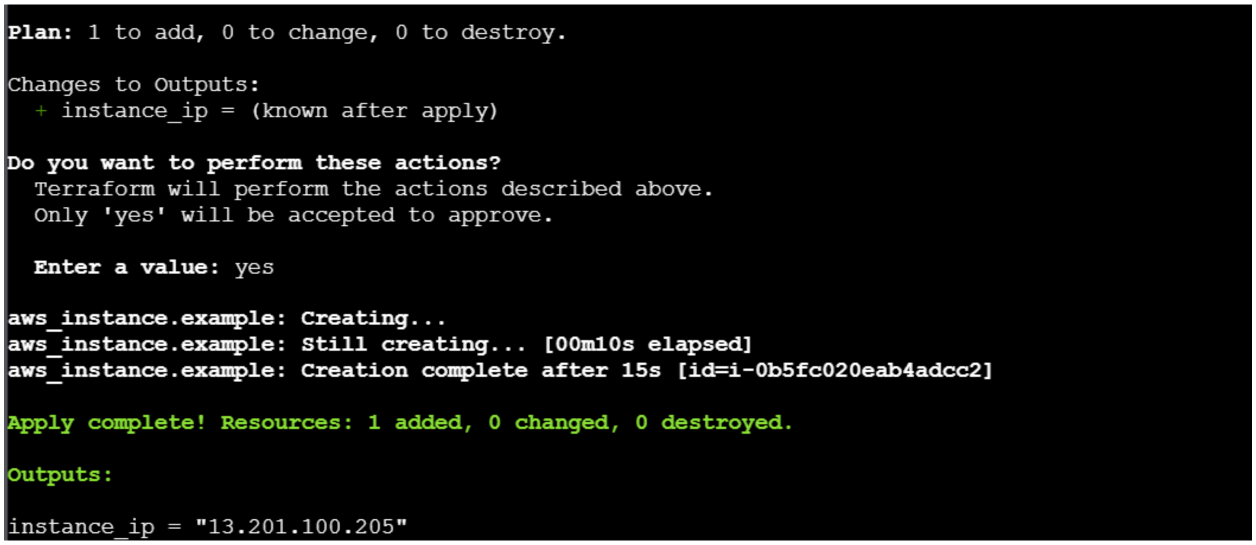
  `terraform destroy` (confirm with yes)



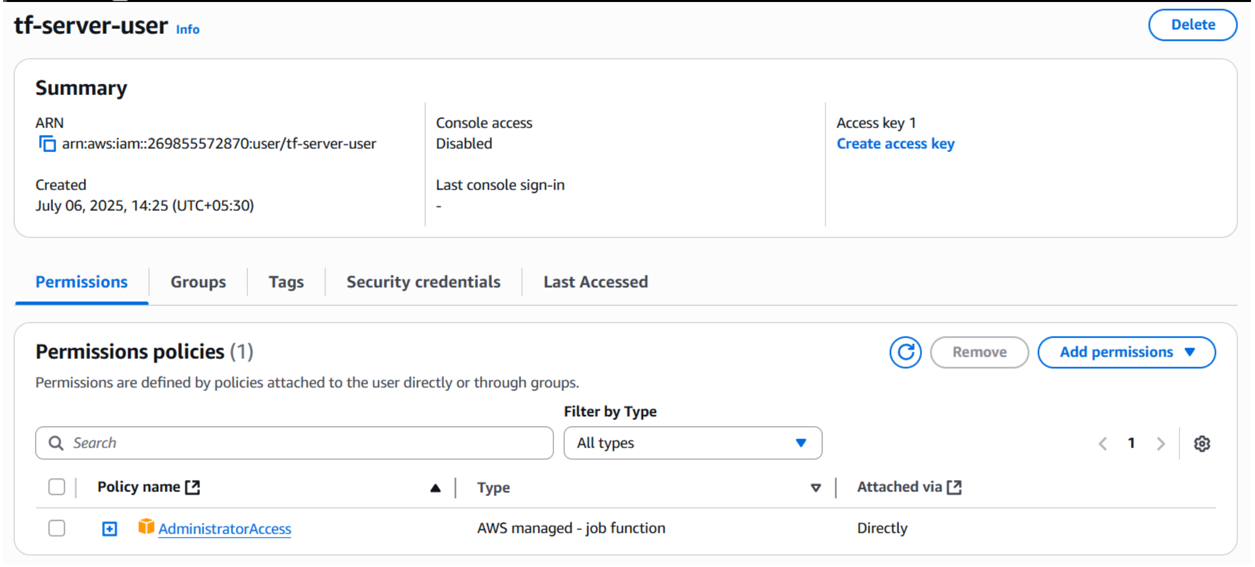
```
Total                                                                                                         88 MB/s |  36 MB     00:00
Hashicorp Stable - x86_64                                                                                    163 kB/s | 3.9 kB     00:00
Importing GPG key 0xA621E701:
 Userid     : "HashiCorp Security (HashiCorp Package Signing) <security+packaging@hashicorp.com>"
 Fingerprint: 798A EC65 4E5C 1542 8C8E 42EE AA16 FCBC A621 E701
 From       : https://rpm.releases.hashicorp.com/gpg
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                            1/1
  Installing       : git-core-2.47.1-1.amzn2023.0.3.x86_64                                                                      1/9
  Installing       : git-core-doc-2.47.1-1.amzn2023.0.3.noarch                                                                  2/9
  Installing       : perl-lib-0.65-477.amzn2023.0.7.x86_64                                                                      3/9
[root@ip-172-31-16-236 ~]# terraform version
Terraform v1.12.2
on linux_amd64
```



```
[root@ip-172-31-16-236 ~]# 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.example will be created
  + resource "aws_instance" "example" {
      + ami                                  = "ami-0f5ee92e2d63afc18"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (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"
```



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

Changes to Outputs:
  + instance_ip = (known after apply)

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.example: Creating...
aws_instance.example: Still creating... [00m10s elapsed]
aws_instance.example: Creation complete after 15s [id=i-0b5fc020eab4adcc2]

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

Outputs:

instance_ip = "13.201.100.205"
```



tf-server-user Info

Summary

ARN
arn:aws:iam::269855572870:user/tf-server-user

Created
July 06, 2025, 14:25 (UTC+05:30)

Console access
Disabled

Last console sign-in
-

Access key 1
Create access key

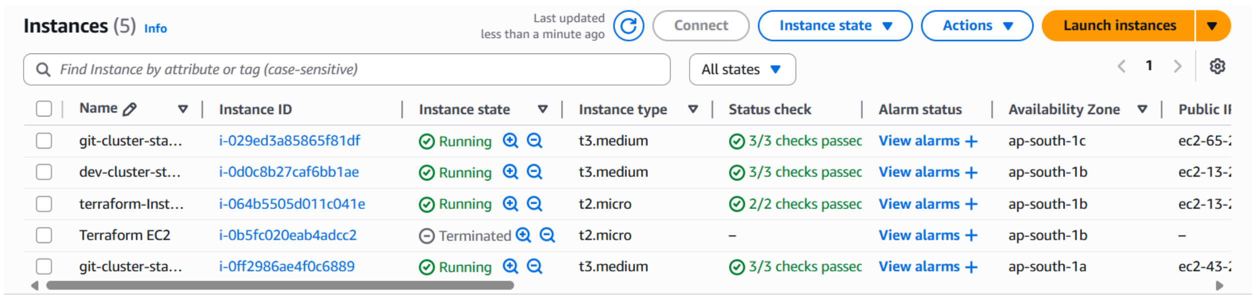
Permissions | Groups | Tags | Security credentials | Last Accessed

Permissions policies (1)

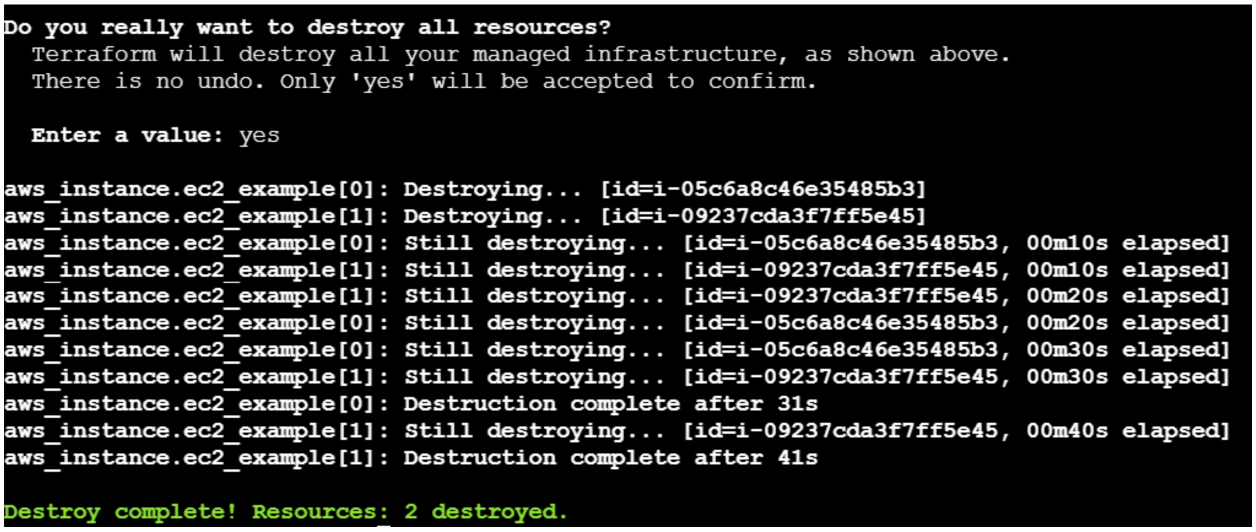
Permissions are defined by policies attached to the user directly or through groups.

Filter by Type: All types

Policy name	Type	Attached via
AdministratorAccess	AWS managed - job function	Directly


Instances (5) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
1

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
	git-cluster-sta…	i-029ed3a85865f81df	Running	t3.medium	3/3 checks passec	View alarms +	ap-south-1c	ec2-65-
	dev-cluster-st…	i-0d0c8b27caf6bb1ae	Running	t3.medium	3/3 checks passec	View alarms +	ap-south-1b	ec2-13-
	terraform-Inst…	i-064b5505d011c041e	Running	t2.micro	2/2 checks passec	View alarms +	ap-south-1b	ec2-13-
	Terraform EC2	i-0b5fc020eab4adcc2	Terminated	t2.micro	–	View alarms +	ap-south-1b	–
	git-cluster-sta…	i-0ff2986ae4f0c6889	Running	t3.medium	3/3 checks passec	View alarms +	ap-south-1a	ec2-43-



```
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.ec2_example[0]: Destroying... [id=i-05c6a8c46e35485b3]
aws_instance.ec2_example[1]: Destroying... [id=i-09237cda3f7ff5e45]
aws_instance.ec2_example[0]: Still destroying... [id=i-05c6a8c46e35485b3, 00m10s elapsed]
aws_instance.ec2_example[1]: Still destroying... [id=i-09237cda3f7ff5e45, 00m10s elapsed]
aws_instance.ec2_example[1]: Still destroying... [id=i-09237cda3f7ff5e45, 00m20s elapsed]
aws_instance.ec2_example[0]: Still destroying... [id=i-05c6a8c46e35485b3, 00m20s elapsed]
aws_instance.ec2_example[0]: Still destroying... [id=i-05c6a8c46e35485b3, 00m30s elapsed]
aws_instance.ec2_example[1]: Still destroying... [id=i-09237cda3f7ff5e45, 00m30s elapsed]
aws_instance.ec2_example[0]: Destruction complete after 31s
aws_instance.ec2_example[1]: Still destroying... [id=i-09237cda3f7ff5e45, 00m40s elapsed]
aws_instance.ec2_example[1]: Destruction complete after 41s

Destroy complete! Resources: 2 destroyed.
```