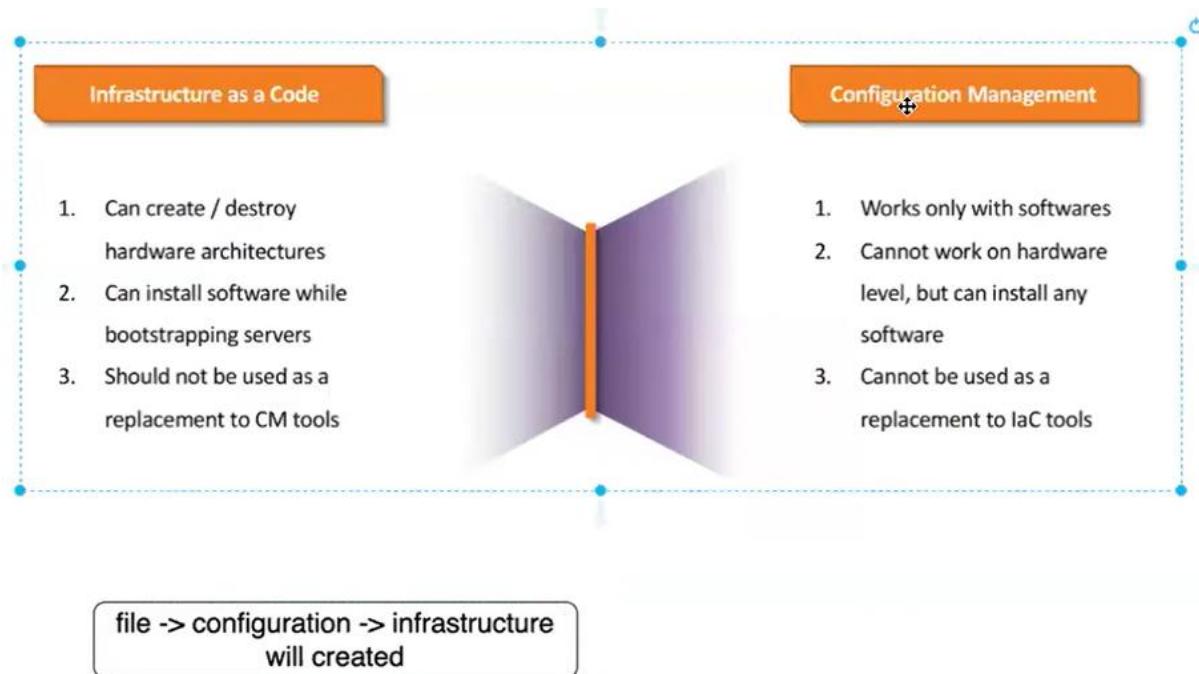


## Terraform DAY-1

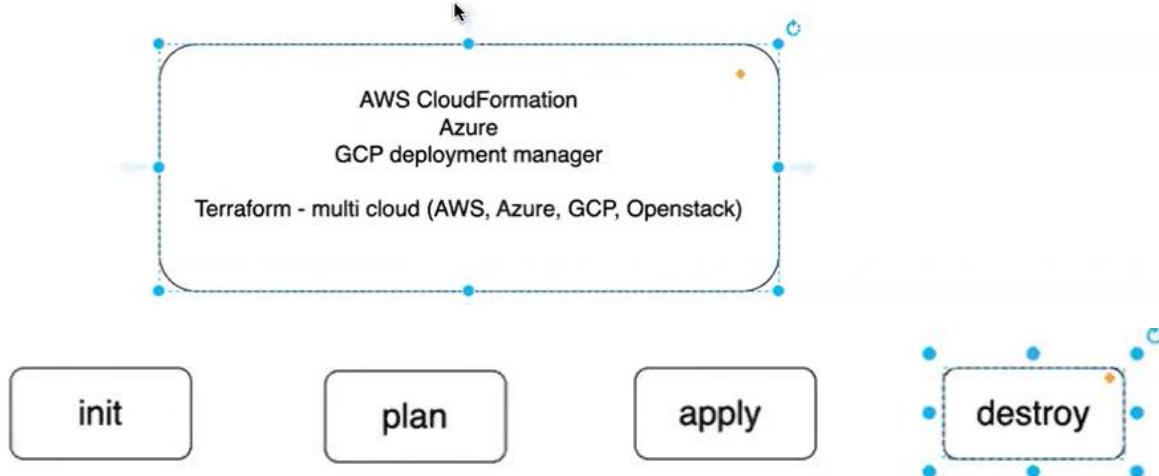
### Agenda: Terraform

1. What is infrastructure as a code?
2. Infrastructure as a code vs Configuration Management
3. Intro to Terraform
4. Installing Terraform
5. Terraform Operations
6. Terraform code
7. Terraform Demo

immutability (0EWQw84XngI)



Terraform is an open-source infrastructure as code software tool created by HashiCorp. It enables users to define and provision a datacenter infrastructure using a high-level configuration language known as Hashicorp Configuration Language, or optionally JSON.



### 1. Launching Instances for Terraform task

The screenshot shows the AWS CloudWatch Metrics console with a single metric named "Terraform-Demo" having a value of 1. The metric is displayed in a chart with a blue line and a corresponding bar chart.

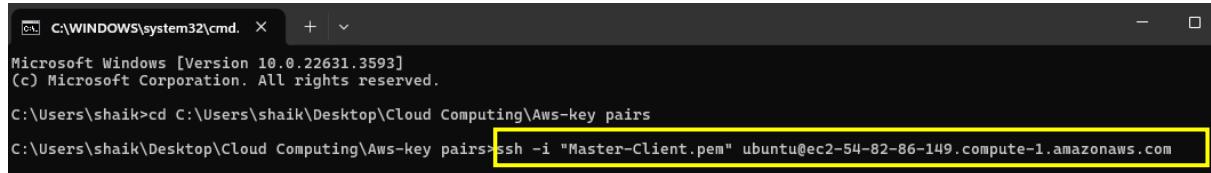
### 2. SSH-Client

The screenshot shows the AWS EC2 Instance Connect interface. The "SSH client" tab is selected. The instance ID "i-0d511b6404a87dd06 (Terraform-Demo)" is listed. Below it, a numbered list provides instructions for connecting via SSH:

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is `Master-Client.pem`
3. Run this command, if necessary, to ensure your key is not publicly viewable.  
    `chmod 400 "Master-Client.pem"`
4. Connect to your instance using its Public DNS:  
    `ec2-54-82-86-149.compute-1.amazonaws.com`

Example:  
`ssh -i "Master-Client.pem" ubuntu@ec2-54-82-86-149.compute-1.amazonaws.com`

3. Paste it on key pair path and connect the instances

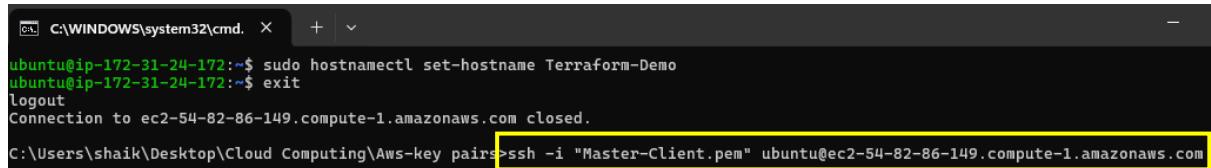


```
C:\WINDOWS\system32\cmd. X + V

Microsoft Windows [Version 10.0.22631.3593]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shaik>cd C:\Users\shaik\Desktop\Cloud Computing\Aws-key pairs
C:\Users\shaik\Desktop\Cloud Computing\Aws-key pairs>ssh -i "Master-Client.pem" ubuntu@ec2-54-82-86-149.compute-1.amazonaws.com
```

4. change the hostname name (Optional)



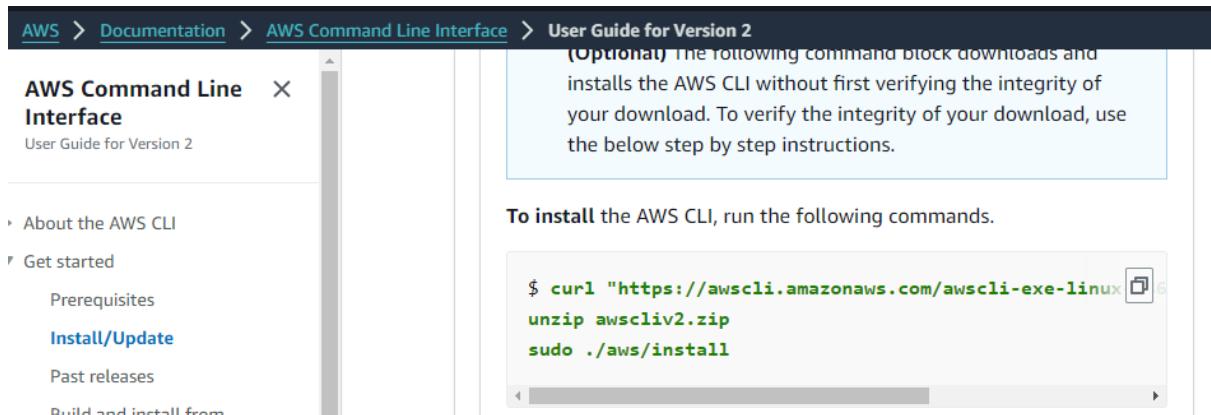
```
C:\WINDOWS\system32\cmd. X + V

ubuntu@ip-172-31-24-172:~$ sudo hostnamectl set-hostname Terraform-Demo
ubuntu@ip-172-31-24-172:~$ exit
logout
Connection to ec2-54-82-86-149.compute-1.amazonaws.com closed.

C:\Users\shaik\Desktop\Cloud Computing\Aws-key pairs>ssh -i "Master-Client.pem" ubuntu@ec2-54-82-86-149.compute-1.amazonaws.com
```

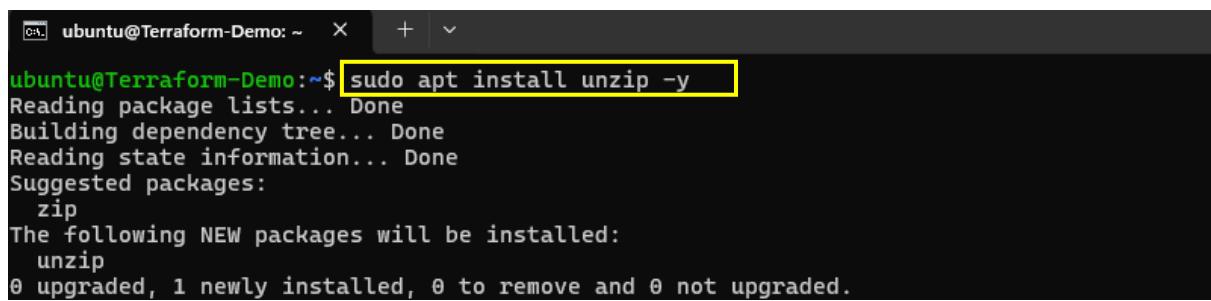
5. Installing the AWS CLI(Command Line Interface) and go through below the link

<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>



The screenshot shows the AWS Command Line Interface User Guide for Version 2. The left sidebar contains links for About the AWS CLI, Get started (Prerequisites, Install/Update, Past releases, Build and install from), and a search bar. The main content area has a header "User Guide for Version 2" and a note about optional command block downloads. It includes instructions to run commands like curl, unzip, and sudo. A code block shows the command: \$ curl "https://awscli.amazonaws.com/awscli-exe-linux.zip" | sudo ./aws/install.

6. sudo apt install unzip -y

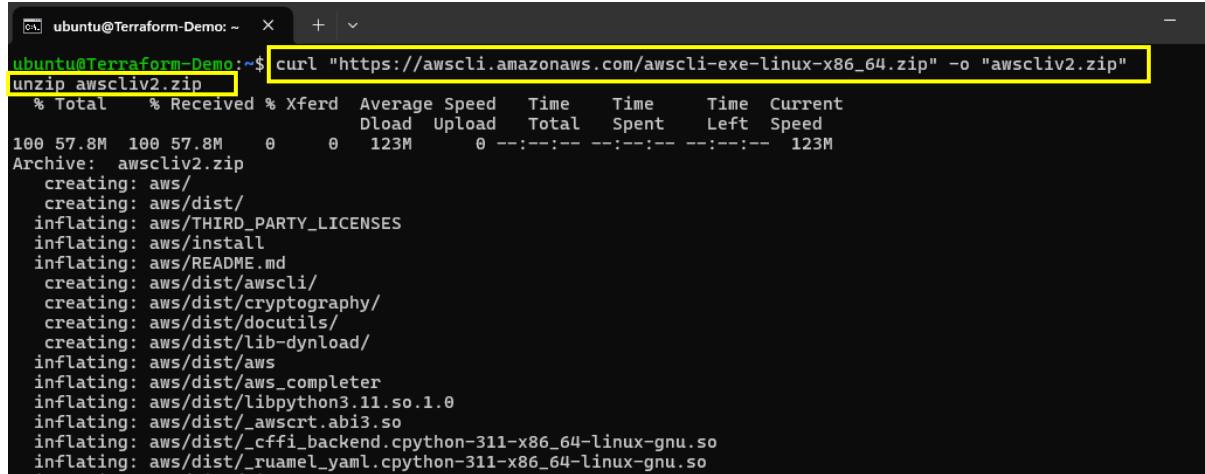


```
ubuntu@Terraform-Demo:~ X + V

ubuntu@Terraform-Demo:~$ sudo apt install unzip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  zip
The following NEW packages will be installed:
  unzip
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
```

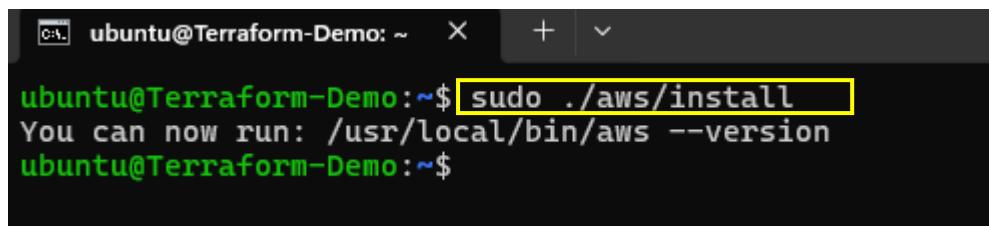
## 7. and then run below the commands to install AWS CLI

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"  
unzip awscliv2.zip
```



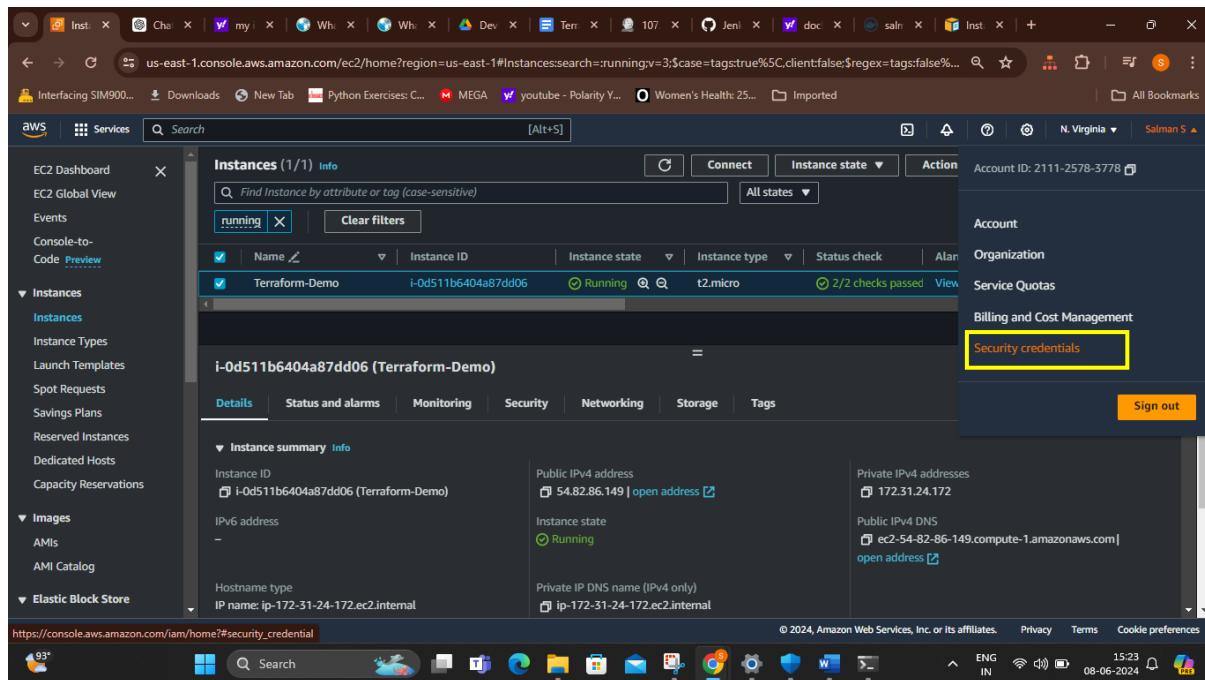
```
ubuntu@Terraform-Demo:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"  
unzip awscliv2.zip  
% Total    % Received  % Xferd  Average Speed   Time     Time     Time  Current  
          Dload   Upload  Total   Spent    Left  Speed  
100 57.8M  100 57.8M    0     0  123M      0 --:--:-- --:--:-- 123M  
Archive: awscliv2.zip  
  creating: aws/  
  creating: aws/dist/  
  inflating: aws/THIRD_PARTY_LICENSES  
  inflating: aws/install  
  inflating: aws/README.md  
  creating: aws/dist/awscli/  
  creating: aws/dist/cryptography/  
  creating: aws/dist/docutils/  
  creating: aws/dist/lib-dynload/  
  inflating: aws/dist/aws  
  inflating: aws/dist/aws_completer  
  inflating: aws/dist/libpython3.11.so.1.0  
  inflating: aws/dist/_awscrt.abi3.so  
  inflating: aws/dist/_cffi_backend.cpython-311-x86_64-linux-gnu.so  
  inflating: aws/dist/_ruamel_yaml.cpython-311-x86_64-linux-gnu.so
```

## 8. sudo ./aws/install



```
ubuntu@Terraform-Demo:~$ sudo ./aws/install  
You can now run: /usr/local/bin/aws --version  
ubuntu@Terraform-Demo:~$
```

## 9. After Installing AWS CLI, Creating Security Credentials



The screenshot shows the AWS Management Console interface. The user is navigating through the Services menu and has selected the EC2 Dashboard. Under the Instances section, there is a single instance named "Terraform-Demo" (Instance ID: i-0d511b6404a87dd06). The instance status is "Running". On the right side of the instance details, there is a "Security credentials" button, which is highlighted with a yellow box. This button is used to generate temporary AWS credentials for the instance.

## 10. It will trigger to IAM Click on create access key

The screenshot shows the AWS Identity and Access Management (IAM) service. On the left, there's a sidebar with options like Dashboard, Access management, User groups, Users, Roles, Policies, Identity providers, and Account settings. The main area is titled 'Access keys (0)' and contains a message: 'No MFA devices. Assign an MFA device to improve the security of your AWS environment'. Below this is a 'Create access key' button, which is highlighted with a yellow box. A note below says, 'As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials.' Another 'Create access key' button is located at the bottom of this note.

## 11. Create access key

This screenshot shows the first step of the 'Create access key' wizard. The title is 'Alternatives to root user access keys'. It includes a warning box about root user access keys being not recommended. It suggests using IAM roles or users instead. There's also a note about creating an IAM user with least privilege permissions. Below this is a 'Continue to create access key?' dialog with a checked checkbox for accepting the risk. At the bottom right are 'Cancel' and 'Create access key' buttons, with the latter being highlighted by a yellow box.

## 12. Copy the Credentials on local machine notepad.

This screenshot shows the second step of the 'Retrieve access key' wizard. It displays the 'Access key' section with two fields: 'Access key' containing 'AKIATCKATZTRAMZ5N6OY' and 'Secret access key' containing a masked value. Below this is a 'Show' link. The next section is 'Access key best practices' with a bulleted list: '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.', and 'Rotate access keys regularly.' A note at the bottom says, 'For more details about managing access keys, see the [best practices for managing AWS access keys](#)'.

13. After Installing AWS CLI, Now we can configure on aws and paste the credentials and it will connected to your AWS Account.

```
ubuntu@Terraform-Demo:~$ aws configure
AWS Access Key ID [None]: AKIATCKATZTRAMZ5N6OY
AWS Secret Access Key [None]: vQn3HD5XLKNDKKWURX6cfLORF7CBOyjP6ZfT3Sg
Default region name [None]:
Default output format [None]:
ubuntu@Terraform-Demo:~$ aws sts get-caller-identity
{
    "UserId": "211125783778",
    "Account": "211125783778",
    "Arn": "arn:aws:iam::211125783778:root"
}
ubuntu@Terraform-Demo:~$
```

13. Now installing Terraform

```
ubuntu@Terraform-Demo:~$ nano terraform.sh
ubuntu@Terraform-Demo:~$
```

14. sudo apt-get update && sudo apt-get install -y gnupg software-properties-common

```
wget -O- https://apt.releases.hashicorp.com/gpg | \
gpg --dearmor | \
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg

gpg --no-default-keyring \
--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
--fingerprint

echo "deb [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 -y

sudo apt-get install terraform -y
```

```
GNU nano 7.2                                     terraform.sh *
sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
wget -O- https://apt.releases.hashicorp.com/gpg | \
gpg --dearmor | \
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg
gpg --no-default-keyring \
--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
--fingerprint
echo "deb [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 -y
sudo apt-get install terraform -y
```

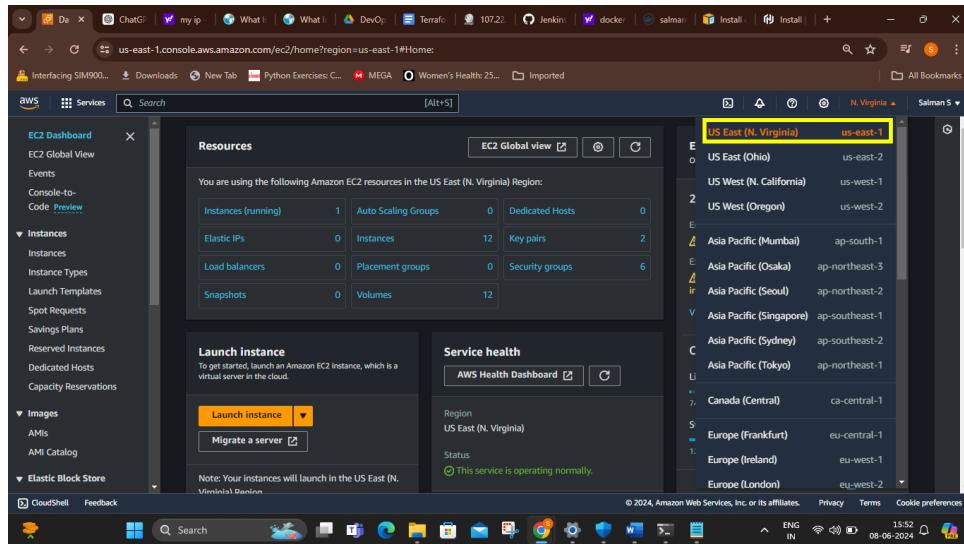
15. and Execute it.

```
ubuntu@Terraform-Demo:~$ nano terraform.sh
ubuntu@Terraform-Demo:~$ chmod +x terraform.sh
ubuntu@Terraform-Demo:~$ bash terraform.sh
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 Packages [1401 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main Translation-en [513 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [129 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [34.6 kB]
```

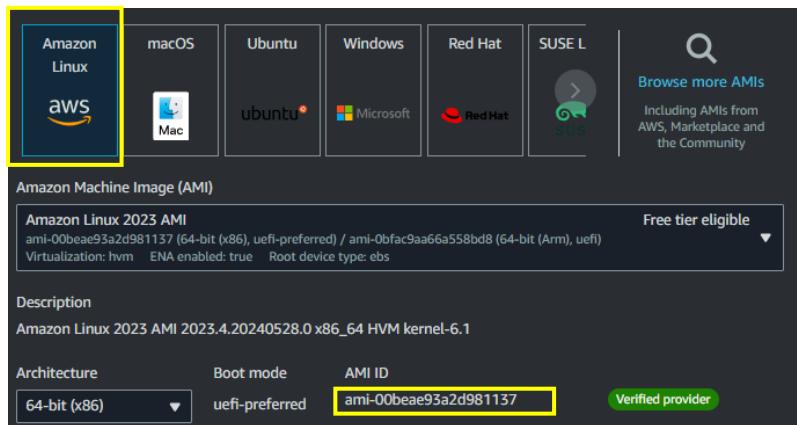
16. Checking that terraform is installed or not, and created demo folder and inside that creating main.tf terraform script file

```
ubuntu@Terraform-Demo:~/ X + v
ubuntu@Terraform-Demo:~$ terraform --version
Terraform v1.8.5
on linux_amd64
ubuntu@Terraform-Demo:~$ mkdir demo
ubuntu@Terraform-Demo:~$ cd demo
ubuntu@Terraform-Demo:~/demo$ nano main.tf
```

## 17. Now I am launching instances at North virginia(us-east-1)



## 18. Copy AMI, I am here Launching Amazon Linux



## 19. Now this the code to Launching instances on AWS using terraform

Terraform we can Install infrastructure at different-different cloud providers

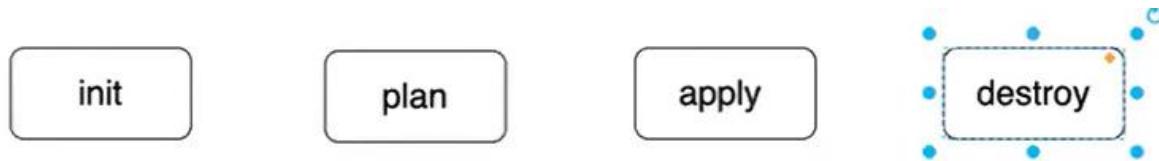
Provider : aws, region: us-east-1(N.V) resource "aws\_instances" Name of Instances: terraform-instance, ami and instance type

A screenshot of a terminal window titled 'ubuntu@Terraform-Demo: ~/'. The window contains a file named 'main.tf' with the following content:

```
provider "aws" {
    region="us-east-1"
}

resource "aws_instance" "instance1" {
    ami = "ami-00beae93a2d981137"
    instance_type = "t2.micro"
    tags = {
        Name = "terraform-instance"
    }
}
```

The first two lines of the provider block and the entire resource block are highlighted with yellow boxes.



- **Init:**

- Initializes the Terraform workspace by downloading and installing required plugins mentioned in your Terraform configuration files (`.tf` files).

- **Plan:**

- Reads your Terraform configuration and compares it to the current state of your infrastructure (if any exists).
- Generates an execution plan that outlines the actions Terraform would take to create or modify resources to match your desired state as defined in the configuration.

- **Apply:**

- Executes the plan generated in the previous step.
- Creates, updates, or destroys resources as necessary to match your desired infrastructure state.

- **Destroy:**

- Similar to `plan`, it creates a plan to destroy your infrastructure based on your Terraform configuration.
- Once confirmed, it removes the resources managed by Terraform, essentially tearing down your infrastructure.

20. Now `terraform init`(Downloading and Installing required plugins)

```
ubuntu@Terraform-Demo: ~/x + - 
ubuntu@Terraform-Demo:~/demo$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.53.0...
- Installed hashicorp/aws v5.53.0 (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.
ubuntu@Terraform-Demo:~/demo$
```

## 21. We can see the its downloaded the things

```
ubuntu@Terraform-Demo:~/demo$ cd .terraform/
ubuntu@Terraform-Demo:~/demo/.terraform$ ls
providers
ubuntu@Terraform-Demo:~/demo/.terraform$ cd providers
ubuntu@Terraform-Demo:~/demo/.terraform/providers$ ls
registry.terraform.io
ubuntu@Terraform-Demo:~/demo/.terraform/providers$ cd registry.terraform.io
ubuntu@Terraform-Demo:~/demo/.terraform/providers/registry.terraform.io$ ls
hashicorp
ubuntu@Terraform-Demo:~/demo/.terraform/providers/registry.terraform.io$ cd hashicorp
ubuntu@Terraform-Demo:~/demo/.terraform/providers/registry.terraform.io/hashicorp$ ls
aws
ubuntu@Terraform-Demo:~/demo/.terraform/providers/registry.terraform.io/hashicorp$ cd aws
ubuntu@Terraform-Demo:~/demo/.terraform/providers/registry.terraform.io/hashicorp/aws$ ;s
-bash: syntax error near unexpected token `;'
ubuntu@Terraform-Demo:~/demo/.terraform/providers/registry.terraform.io/hashicorp/aws$ ls
5.53.0
ubuntu@Terraform-Demo:~/demo/.terraform/providers/registry.terraform.io/hashicorp/aws$ cd 5.53.0
ubuntu@Terraform-Demo:~/demo/.terraform/providers/registry.terraform.io/hashicorp/aws/5.53.0$ ls
linux_amd64
ubuntu@Terraform-Demo:~/demo/.terraform/providers/registry.terraform.io/hashicorp/aws/5.53.0$ cd linux_amd64
ubuntu@Terraform-Demo:~/demo/.terraform/providers/registry.terraform.io/hashicorp/aws/5.53.0/linux_amd64$ ls
LICENSE.txt  terraform-provider-aws_v5.53.0_x5
ubuntu@Terraform-Demo:~/demo/.terraform/providers/registry.terraform.io/hashicorp/aws/5.53.0/linux_amd64$
```

## 22. Now terraform plan(Reads our Configurations, generate and execute the plans)

```
ubuntu@Terraform-Demo:~/demo$ 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.instance1 will be created
+ resource "aws_instance" "instance1" {
    + ami                               = "ami-00beae93a2d981137"
    + 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)
    + instance_type                   = "t2.micro"
    + ipv6_address_count              = (known after apply)
    + ipv6_addresses                  = (known after apply)
    + key_name                        = (known after apply)
    + 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)
    + primary_network_interface_id    = (known after apply)
    + private_dns                     = (known after apply)
    + private_ip                      = (known after apply)
}
```

## 23. 1 to add there is no changes and there is no destroy and create. Because this is our first execution

```
+ subnet_id                         = (known after apply)
+ tags                                = {}
  + "Name" = "terraform-instance"
}
+ tags_all                           = {}
  + "Name" = "terraform-instance"
}
+ 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, 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.

ubuntu@Terraform-Demo:~/demo\$

## 24. terraform apply now, it will going to install our infrastructure

```
ubuntu@Terraform-Demo:~/demo$ terraform apply
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.instance1 will be created
+ resource "aws_instance" "instance1" {
  + ami
  + arn
  + associate_public_ip_address
  + availability_zone
  + cpu_core_count
  + cpu_threads_per_core
  + disable_api_stop
  + disable_api_termination
  + ebs_optimized
  + get_password_data
  + host_id
  + host_resource_group_arn
  + iam_instance_profile
  + id
  + instance_initiated_shutdown_behavior
  + instance_lifecycle
  + instance_state
  + instance_type
  + ipv6_address_count
  + ipv6_addresses
  + key_name
  + monitoring
  + outpost_arn
  + password_data
  + placement_group
  + placement_partition_number
  + primary_network_interface_id
  + private_dns
  + private_ip
}
```

## 25. click on yes and it will start creating the infrastructure

```
+ private_ip = (known after apply)
+ public_dns = (known after apply)
+ public_ip = (known after apply)
+ secondary_private_ips = (known after apply)
+ security_groups = (known after apply)
+ source_dest_check = true
+ spot_instance_request_id = (known after apply)
+ subnet_id = (known after apply)
+ tags = {
  + "Name" = "terraform-instance"
}
+ tags_all = {
  + "Name" = "terraform-instance"
}
+ 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, 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.instance1: Creating...
aws_instance.instance1: Still creating... [10s elapsed]
aws_instance.instance1: Still creating... [20s elapsed]
aws_instance.instance1: Creation complete after 21s [id=i-060e99d10f3c27600]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
ubuntu@Terraform-Demo:~/demo$
```

26. We can see the terraform-instances has been created, we given tag name terraform-instance

| Instances (1/2) <a href="#">Info</a>                                    |                    |                               |                      |                |                                |                                  |                   |                      |
|---|--------------------|-------------------------------|----------------------|----------------|--------------------------------|----------------------------------|-------------------|----------------------|
|   |                    | <a href="#">Connect</a>       |                      | Instance state | Actions                        | <a href="#">Launch instances</a> |                   |                      |
| <input type="text"/> Find Instance by attribute or tag (case-sensitive) |                    | <a href="#">Clear filters</a> |                      | All states     |                                |                                  | < 1 >             | <a href="#">Edit</a> |
|   | Name               | Instance ID                   | Instance state       | Instance type  | Status check                   | Alarm status                     | Availability Zone |                      |
|   | Terraform-Demo     | i-0d511b6404a87dd06           | <span>Running</span> | t2.micro       | <span>2/2 checks passed</span> | <a href="#">View alarms</a> +    | us-east-1a        |                      |
| <input checked="" type="checkbox"/>                                     | terraform-instance | i-060e99d10f3c27600           | <span>Running</span> | t2.micro       | <span>Initializing</span>      | <a href="#">View alarms</a> +    | us-east-1a        |                      |

27. terraform destroy, means it will destroy the infrastructure

```
ubuntu@Terraform-Demo:~/demo$ terraform destroy
aws_instance.instance1: Refreshing state... [id=i-060e99d10f3c27600]

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.instance1 will be destroyed
- resource "aws_instance" "instance1" {
    - ami
    - arn
    - associate_public_ip_address
    - availability_zone
    - cpu_core_count
    - cpu_threads_per_core
    - disable_api_stop
    - disable_api_termination
    - ebs_optimized
    - get_password_data
    - hibernation
    - id
    - instance_initiated_shutdown_behavior
    - instance_state
    - instance_type
    - ipv6_address_count
    - ipv6_addresses
    - monitoring
    - placement_partition_number
    - primary_network_interface_id
    - private_dns
    - private_ip
    - public_dns
    - public_ip
    - secondary_private_ips
    - security_groups
        - "default",
}
```

28. Previously 1 added now its 1 destroy.

```
- root_block_device {
    - delete_on_termination = true -> null
    - device_name           = "/dev/xvda" -> null
    - encrypted              = false -> null
    - iops                   = 3000 -> null
    - tags                   = {} -> null
    - tags_all               = {} -> null
    - throughput              = 125 -> null
    - volume_id               = "vol-0bcfc06b1bc81ab94" -> null
    - volume_size              = 8 -> null
    - volume_type              = "gp3" -> null
    # (1 unchanged attribute hidden)
}
}

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.instance1: Destroying... [id=i-060e99d10f3c27600]
aws_instance.instance1: Still destroying... [id=i-060e99d10f3c27600, 10s elapsed]
aws_instance.instance1: Still destroying... [id=i-060e99d10f3c27600, 20s elapsed]
aws_instance.instance1: Still destroying... [id=i-060e99d10f3c27600, 30s elapsed]
aws_instance.instance1: Still destroying... [id=i-060e99d10f3c27600, 40s elapsed]
aws_instance.instance1: Destruction complete after 40s

Destroy complete! Resources: 1 destroyed.
ubuntu@Terraform-Demo:~/demo$
```

29. Now the terraform-instance is terminated.

| Instances (1/13) <a href="#">Info</a> |                    |                     |  |               |                                   |   |                   |   |
|---------------------------------------|--------------------|---------------------|--|---------------|-----------------------------------|---|-------------------|---|
|                                       | Name               | Instance ID         | Instance state   | Instance type | Status check                      | Alarm status                                  | Availability Zone | P |
|                                       |                    |                     | <a href="#">All states</a>                                     |               |                                   |   |                   |   |
| <input type="checkbox"/>              | Ansible-Slave2     | i-0cdcf4ed7ee38a75  | <a href="#">Stopped</a> <a href="#">Q</a> <a href="#">Q</a>    | t2.micro      | -                                 | <a href="#">View alarms</a> <a href="#">+</a> | us-east-1a        | - |
| <input type="checkbox"/>              | Terraform-Demo     | i-0d511b6404a87dd06 | <a href="#">Running</a> <a href="#">Q</a> <a href="#">Q</a>    | t2.micro      | <a href="#">2/2 checks passed</a> | <a href="#">View alarms</a> <a href="#">+</a> | us-east-1a        | - |
| <input checked="" type="checkbox"/>   | terraform-instance | i-060e99d10f3c27600 | <a href="#">Terminated</a> <a href="#">Q</a> <a href="#">Q</a> | t2.micro      | -                                 | <a href="#">View alarms</a> <a href="#">+</a> | us-east-1a        | - |
| <input type="checkbox"/>              | Kubernetes-Master  | i-0182e44fa7376634f | <a href="#">Stopped</a> <a href="#">Q</a> <a href="#">Q</a>    | t2.medium     | -                                 | <a href="#">View alarms</a> <a href="#">+</a> | us-east-1e        | - |
| <input type="checkbox"/>              | Kubernetes-Slave   | i-0e0d3dc5efcf9445  | <a href="#">Stopped</a> <a href="#">Q</a> <a href="#">Q</a>    | t2.medium     | -                                 | <a href="#">View alarms</a> <a href="#">+</a> | us-east-1e        | - |

30. Again **terraform plan** and no need to **terraform init**, because we already initialize

```
ubuntu@Terraform-Demo: ~/ ... + 
ubuntu@Terraform-Demo:~/demo$ terraform plan
```

31. again **terraform apply**

```
ubuntu@Terraform-Demo: ~/ ... + 
ubuntu@Terraform-Demo:~/demo$ terraform apply
```

32. Again it will going to creating one instances

```
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.instance1: Creating...
aws_instance.instance1: Still creating... [10s elapsed]
aws_instance.instance1: Still creating... [20s elapsed]
aws_instance.instance1: Still creating... [30s elapsed]
aws_instance.instance1: Creation complete after 32s [id=i-096b588d095b89dfa]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
ubuntu@Terraform-Demo:~/demo$
```

33. See Now its again running status

| Instances (1/13) <a href="#">Info</a> |                    |                     |  |               |  |  |            |  |
|---------------------------------------|--------------------|---------------------|--|---------------|--|--|------------|--|
|                                       | Name               | Instance ID         | Instance state   | Instance type |  |  | All states |  |
|                                       |                    |                     |  |               |  |  |            |  |
| <input type="checkbox"/>              | Ansible-Slave2     | i-0cdcf4ed7ee38a75  | <a href="#">Stopped</a> <a href="#">Q</a> <a href="#">Q</a>    | t2.micro      |  |  |            |  |
| <input type="checkbox"/>              | Terraform-Demo     | i-0d511b6404a87dd06 | <a href="#">Running</a> <a href="#">Q</a> <a href="#">Q</a>    | t2.micro      |  |  |            |  |
| <input type="checkbox"/>              | terraform-instance | i-060e99d10f3c27600 | <a href="#">Terminated</a> <a href="#">Q</a> <a href="#">Q</a> | t2.micro      |  |  |            |  |
| <input checked="" type="checkbox"/>   | terraform-instance | i-096b588d095b89dfa | <a href="#">Running</a> <a href="#">Q</a> <a href="#">Q</a>    | t2.micro      |  |  |            |  |
| <input type="checkbox"/>              | Kubernetes-Master  | i-0182e44fa7376634f | <a href="#">Stopped</a> <a href="#">Q</a> <a href="#">Q</a>    | t2.medium     |  |  |            |  |

### 34. In Terraform:

- **terraform.tfstate**: Stores the **current state** of your infrastructure managed by **Terraform** (resources created, configurations applied). (This is the main file)
- **terraform.tfstate.backup**: Acts as a **backup** of the `terraform.tfstate` file, created automatically before significant changes are made. (This is a safety net)

```
ubuntu@Terraform-Demo:~/demo$ ls
main.tf  terraform.tfstate  terraform.tfstate.backup
ubuntu@Terraform-Demo:~/demo$ cat terraform.tfstate
{
  "version": 4,
  "terraform_version": "1.8.5",
  "serial": 5,
  "lineage": "a536d83a-5fce-9012-4cb0-68a097938a0c",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "aws_instance",
      "name": "instance1",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
      "instances": [
        {
          "schema_version": 1,
          "attributes": {
            "ami": "ami-00beae93a2d981137",
            "arn": "arn:aws:ec2:us-east-1:211125783778:instance/i-096b588d095b89dfa",
            "associate_public_ip_address": true,
            "availability_zone": "us-east-1a",
            "capacity_reservation_specification": [
              {
                "capacity_reservation_preference": "open",
                "capacity_reservation_target": []
              }
            ],
            "cpu_core_count": 1,
            "cpu_options": [
              {
                "amd_sev_snp": "",
                "core_count": 1,
                "threads_per_core": 1
              }
            ],
            "cpu_threads_per_core": 1,
            "credit_specification": [

```

### 35. `terraform destroy` it will destroy your infrastructure again

```
ubuntu@Terraform-Demo:~/demo$ terraform destroy
Terraform will perform the following actions:
  main.tf  terraform.tfstate  terraform.tfstate.backup
# aws_instance.instance1 will be destroyed
  - resource "aws_instance" "instance1" { . [id=i-096b588d095b89dfa]
    - ami           = "ami-00beae93a2d981137" -> null
    - arn          = "arn:aws:ec2:us-east-1:211125783778:instance/i-096b588d095b89dfa" -> null
    - associate_public_ip_address = true -> null
    - availability_zone       = "us-east-1a" -> null
    - cpu_core_count         = 1 -> null
    - cpu_threads_per_core   = 1 -> null

```

### 36. Now its destroying

```
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.instance1: Destroying... [id=i-096b588d095b89dfa]
aws_instance.instance1: Still destroying... [id=i-096b588d095b89dfa, 10s elapsed]
aws_instance.instance1: Still destroying... [id=i-096b588d095b89dfa, 20s elapsed]
aws_instance.instance1: Still destroying... [id=i-096b588d095b89dfa, 30s elapsed]
aws_instance.instance1: Still destroying... [id=i-096b588d095b89dfa, 40s elapsed]
aws_instance.instance1: Destruction complete after 40s

Destroy complete! Resources: 1 destroyed.
ubuntu@Terraform-Demo:~/demo$
```

37. its terminated.

| Instances (1/14) <a href="#">Info</a>                                   |                            | <a href="#">Connect</a>                                     |                                 | Instance state <a href="#">▼</a> |                                | Actions <a href="#">▼</a>           |  | Launch instances <a href="#">▼</a> |  |
|---|----------------------------|---|---------------------------------|----------------------------------|--------------------------------|-------------------------------------|--|------------------------------------|--|
| <input type="text"/> Find Instance by attribute or tag (case-sensitive) |                            |   |                                 | All states <a href="#">▼</a>     |                                |                                     |  |                                    |  |
| Name <a href="#">▼</a>  | Instance ID                | Instance state <a href="#">▼</a>                            | Instance type <a href="#">▼</a> | Status check <a href="#">▼</a>   | Alarm status <a href="#">▼</a> | Availability Zone <a href="#">▼</a> |  |                                    |  |
| terraform-Demo  | i-005110d6404ab7dd06       | <span>Running</span> <a href="#">Q</a> <a href="#">Q</a>    | t2.micro                        | <span>2/2 checks passed</span>   | <a href="#">View alarms +</a>  | us-east-1a                          |  |                                    |  |
| terraform-instance  | i-060e99d10f3c27600        | <span>Terminated</span> <a href="#">Q</a> <a href="#">Q</a> | t2.micro                        | -                                | <a href="#">View alarms +</a>  | us-east-1a                          |  |                                    |  |
| <b>terraform-instance</b>   | <b>i-096b588d095b89dfa</b> | <span>Terminated</span> <a href="#">Q</a> <a href="#">Q</a> | t2.micro                        | -                                | <a href="#">View alarms +</a>  | us-east-1a                          |  |                                    |  |
| Kubernetes-Master   | i-0182e44fa7376634f        | <span>Stopped</span> <a href="#">Q</a> <a href="#">Q</a>    | t2.medium                       | -                                | <a href="#">View alarms +</a>  | us-east-1e                          |  |                                    |  |
| Kubernetes-Slave  | i-0e0d3dc5efcf9445         | <span>Stopped</span> <a href="#">Q</a> <a href="#">Q</a>    | t2.medium                       | -                                | <a href="#">View alarms +</a>  | us-east-1e                          |  |                                    |  |

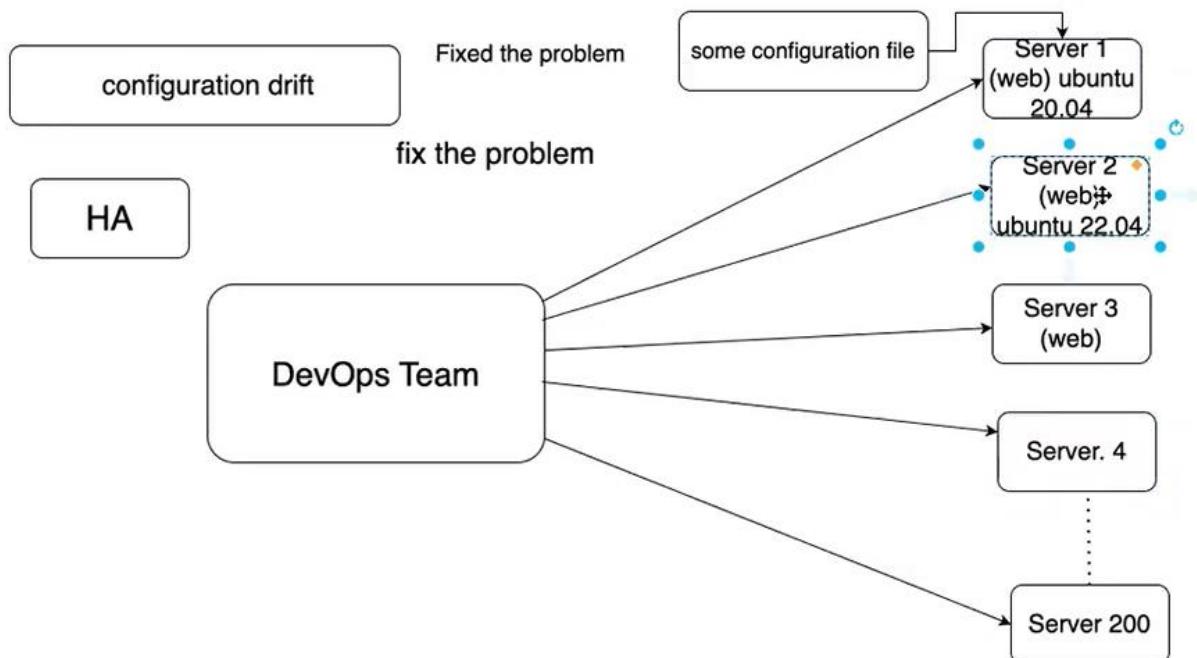
**i-096b588d095b89dfa (terraform-instance)**

(8 minutes)

|                                   |                                  |   |
|-----------------------------------|----------------------------------|---|
| Instance auto-recovery<br>Default | Lifecycle<br>normal              | Stop-hibernate behavior<br>Disabled   |
| AMI Launch index<br>0             | Key pair assigned at launch<br>- | State transition reason<br>User initiated (2024-06-08 10:51:02 GMT)               |
| Credit specification<br>standard  | Kernel ID<br>-                   | State transition message<br>Client.UserInitiatedShutdown: User initiated shutdown |
| Usage operation                   | RAM disk ID                      | Owner   |

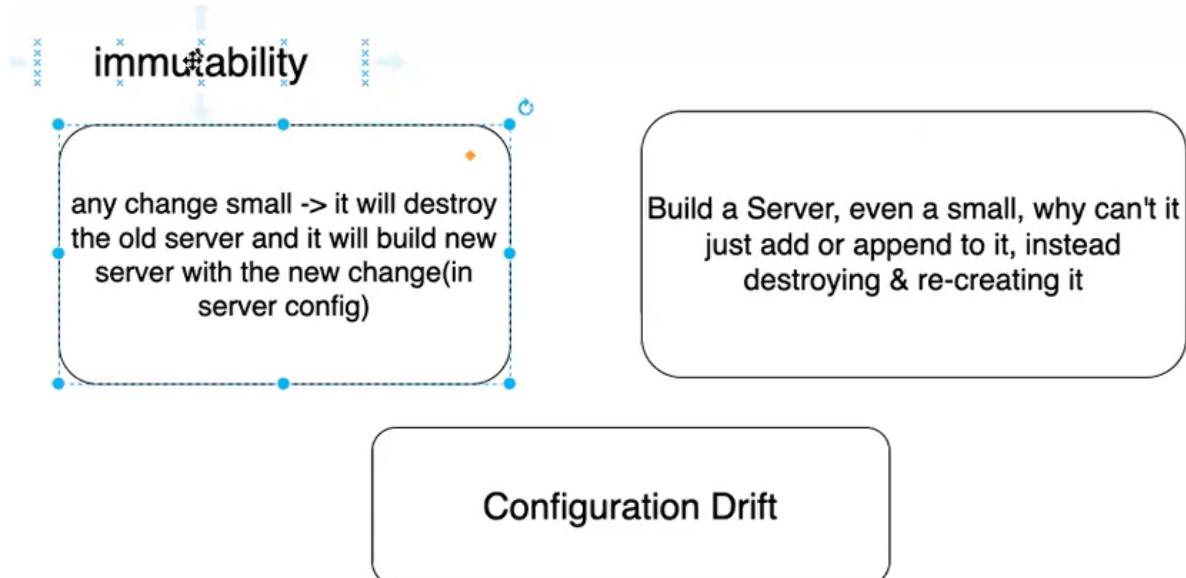
### CONFIGURATION DRIFT:

**Configuration drift** refers to the gradual, unintended divergence between the actual state of deployed infrastructure and its intended state as defined in code. It typically occurs due to manual changes or updates made directly to resources outside of the defined configuration management system, like Terraform.



## IMMUTABILITY:

**Immutability**, in the context of Terraform, refers to the principle of never modifying infrastructure in-place. Instead, it involves creating new resources or replacing existing ones when changes are needed, ensuring consistency and predictability. This approach minimizes configuration drift and promotes reproducibility in infrastructure deployment.



## 38. Again Terraform plan

```
ubuntu@Terraform-Demo: ~/ X + | ~
ubuntu@Terraform-Demo:~/demo$ terraform plan
```

## 39. Again Terraform apply

```
ubuntu@Terraform-Demo: ~/ X + | ~
ubuntu@Terraform-Demo:~/demo$ terraform apply
```

## 40. Now Again its creating the infrastructure

```
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.instance1: Creating...
aws_instance.instance1: Still creating... [10s elapsed]
aws_instance.instance1: Still creating... [20s elapsed]
aws_instance.instance1: Still creating... [30s elapsed]
aws_instance.instance1: Creation complete after 32s [id=i-0dfa3fad4a48d56c5]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
ubuntu@Terraform-Demo:~/demo$
```

41. We can see, but there is no key-pair attached here

|   | terraform-instance | i-0dfa3fad4a48d56c5   | Running | t2.micro | Initializing | View alarms + |
|---|--------------------|-----------------------|---------|----------|--------------|---------------|
| □ | DockerDemo         | i-09ea869fc35027a1    | Stopped | t2.micro | -            | View alarms + |
| □ | Ansible-Master     | i-0810c56f8a6550852   | Stopped | t2.micro | -            | View alarms + |
| □ | Ansible-Slave1     | i-051a1d40514cf61c8   | Stopped | t2.micro | -            | View alarms + |
| □ | Ansible-Slave2     | i-0rrirfe4er17ee38a75 | Stopped | t2.micro | -            | View alarms - |

**i-0dfa3fad4a48d56c5 (terraform-instance)**

|                                   |                                  |                                     |
|-----------------------------------|----------------------------------|-------------------------------------|
| Instance auto-recovery<br>Default | Lifecycle<br>normal              | Stop-hibernate behavior<br>Disabled |
| AMI Launch index<br>0             | Key pair assigned at launch<br>- | State transition reason<br>-        |

42. Now Again going to modifying the main.tf file

```
ubuntu@Terraform-Demo: ~/ X + v
ubuntu@Terraform-Demo:~/demo$ nano main.tf
```

43. `resource key-pair` named as `sshkeydemo` and `public-key` leaving empty as of now and follow the upcoming steps

```
ubuntu@Terraform-Demo: ~/ X + v
GNU nano 7.2
provider "aws" {
  region="us-east-1"
}

resource "aws_key_pair" "sshkey" {
  key_name = "sshkeydemo"
  public_key = file("")
}

resource "aws_instance" "instance1" {
  ami = "ami-00beae93a2d981137"
  instance_type = "t2.micro"
  tags = {
    Name = "terraform-instance"
  }
}
```

44. generating ssh-keygen and after that going inside the .ssh/

```
ubuntu@Terraform-Demo:~/demo$ nano main.tf
ubuntu@Terraform-Demo:~/demo$ cd ~
ubuntu@Terraform-Demo:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_ed25519
Your public key has been saved in /home/ubuntu/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:IkHjRtTySno424LZzvc1iNZyt5fRaGMZR23fJOJRnyk ubuntu@Terraform-Demo
The key's randomart image is:
+--[ED25519 256]--+
|   .   o. |
| +... + +.+|
| +o   o E *o|
| .... . o . o|
| +... S *|
| + o+ o B .|
| + =+ + +o +|
|o.+oo o oo|
| .oo . . .|
+---[SHA256]---+
ubuntu@Terraform-Demo:~$ cd .ssh
ubuntu@Terraform-Demo:~/ssh$ ls
authorized_keys  id_ed25519  id_ed25519.pub
ubuntu@Terraform-Demo:~/ssh$
```

45. and copy the path of present working directory

```
ubuntu@Terraform-Demo:~/demo$ cd ~
ubuntu@Terraform-Demo:~$ cd .ssh
ubuntu@Terraform-Demo:~/ssh$ ls
authorized_keys  id_ed25519  id_ed25519.pub
ubuntu@Terraform-Demo:~/ssh$ pwd
/home/ubuntu/.ssh
ubuntu@Terraform-Demo:~/ssh$
```

46. and come demo folder and go to main.tf file

```
ubuntu@Terraform-Demo:~/demo$ ls
main.tf  terraform.tfstate  terraform.tfstate.backup
ubuntu@Terraform-Demo:~/demo$ nano main.tf
```

47. paste the path and public key file

```
GNU nano 7.2 main.tf *
provider "aws" {
  region="us-east-1"
}

resource "aws_key_pair" "sshkey" {
  key_name = "sshkeydemo"
  public_key = file("/home/ubuntu/.ssh/id_ed25519.pub")
}

resource "aws_instance" "instance1" {
  ami = "ami-00beae93a2d981137"
  instance_type = "t2.micro"
  tags = {
    Name = "terraform-instance"
  }
}
```

48. And paste key\_name in resource part, and It will trigger to public\_key, save and exit

```
GNU nano 7.2 main.tf *
provider "aws" {
  region="us-east-1"
}

resource "aws_key_pair" "sshkey" {
  key_name = "sshkeydemo"
  public_key = file("/home/ubuntu/.ssh/id_ed25519.pub")
}

resource "aws_instance" "instance1" {
  ami = "ami-00beae93a2d981137"
  key_name = aws_key_pair.sshkey.key_name
  instance_type = "t2.micro"
  tags = {
    Name = "terraform-instance"
  }
}
```

49. And Now terraform plan(Reads Our Configurations, generate and execute the plans)

```
ubuntu@Terraform-Demo:~/demo$ terraform plan
```

## 50. Now I Again its Executing the plan

```
ubuntu@Terraform-Demo:~/demo$ 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.instance1 will be created
+ resource "aws_instance" "instance1" {
    + ami                                = "ami-00beae93a2d981137"
    + 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)
    + instance_type                     = "t2.micro"
    + ipv6_address_count                = (known after apply)
    + ipv6_addresses                    = (known after apply)
    + key_name                           = (known after apply)
    + 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)
    + primary_network_interface_id     = (known after apply)
    + private_dns                        = (known after apply)
    + private_ip                         = (known after apply)
```

## 51. Here are replaced

```
# aws_instance.instance1 must be replaced
+ resource "aws_instance" "instance1" {
    ~ arn                                = "arn:aws:ec2:us-east-1:211125783778:instance/i-0dfa3f
    ~ associate_public_ip_address        = true -> (known after apply)
    ~ availability_zone                  = "us-east-1a" -> (known after apply)
    ~ cpu_core_count                     = 1 -> (known after apply)
    ~ cpu_threads_per_core              = 1 -> (known after apply)
    ~ disable_api_stop                  = false -> (known after apply)
    ~ disable_api_termination           = false -> (known after apply)
    ~ ebs_optimized                      = false -> (known after apply)
    - hibernation                        = false -> null
    + host_id                            = (known after apply)
    + host_resource_group_arn           = (known after apply)
    + iam_instance_profile              = (known after apply)
    ~ id                                 = "i-0dfa3fad4a48d56c5" -> (known after apply)
    ~ instance_initiated_shutdown_behavior = "stop" -> (known after apply)
    + instance.lifecycle                = (known after apply)
    ~ instance.state                    = "running" -> (known after apply)
    ~ ipv6_address_count                = 0 -> (known after apply)
    ~ ipv6_addresses                    = [] -> (known after apply)
    + key_name                           = "sshkeydemo" # forces replacement
```

## 52. Terraform Apply

```
ubuntu@Terraform-Demo:~/demo$ terraform apply
```

53. We can see it will add 2 and 1 to destroy, that means its destroying the instances and adding the instances and key -pair

```
Plan: 2 to add, 0 to change, 1 to destroy.

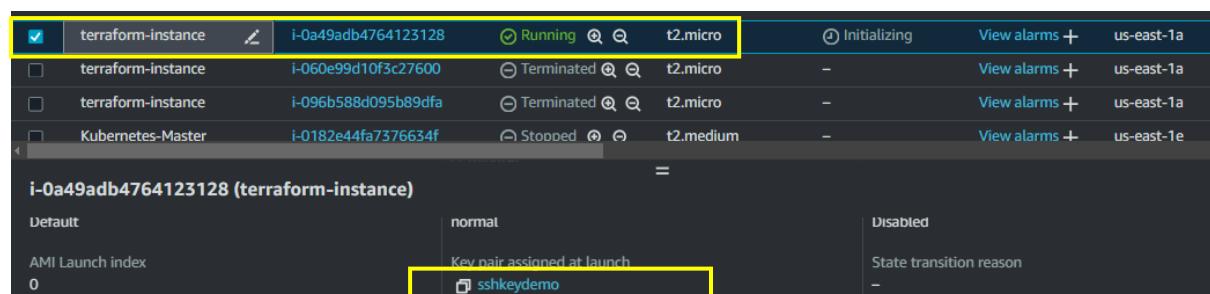
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

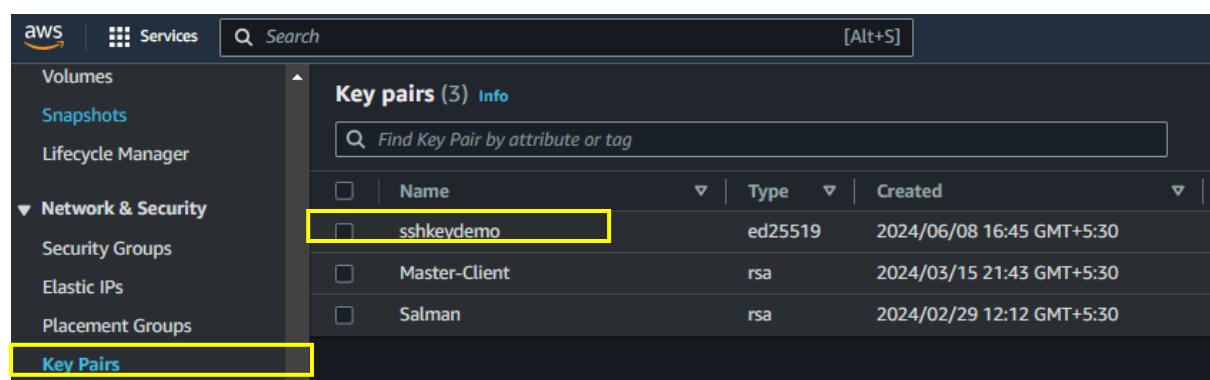
aws_instance.instance1: Destroying... [id=i-0dfa3fad4a48d56c5]
aws_instance.instance1: Still destroying... [id=i-0dfa3fad4a48d56c5, 10s elapsed]
aws_instance.instance1: Still destroying... [id=i-0dfa3fad4a48d56c5, 20s elapsed]
aws_instance.instance1: Still destroying... [id=i-0dfa3fad4a48d56c5, 30s elapsed]
aws_instance.instance1: Still destroying... [id=i-0dfa3fad4a48d56c5, 40s elapsed]
aws_instance.instance1: Destruction complete after 40s
aws_key_pair.sshkey: Creating...
aws_key_pair.sshkey: Creation complete after 0s [id=sshkeydemo]
aws_instance.instance1: Creating...
aws_instance.instance1: Still creating... [10s elapsed]
aws_instance.instance1: Still creating... [20s elapsed]
aws_instance.instance1: Still creating... [30s elapsed]
aws_instance.instance1: Creation complete after 32s [id=i-0a49adb4764123128]

Apply complete! Resources: 2 added, 0 changed, 1 destroyed.
ubuntu@Terraform-Demo:~/demo$
```

54. We can see the key-pair attached



55. We Can see the Public Key-Pair



56. Again go to main.tf file

```
ubuntu@Terraform-Demo:~/ X + | ~
ubuntu@Terraform-Demo:~$ ls
aws awscliv2.zip demo terraform.sh
ubuntu@Terraform-Demo:~$ cd demo
ubuntu@Terraform-Demo:~/demo$ ls
main.tf terraform.tfstate terraform.tfstate.backup
ubuntu@Terraform-Demo:~/demo$ nano main.tf
```

57. This time I am attaching private key pair and save it

```
ubuntu@Terraform-Demo:~/ X + | ~
GNU nano 7.2
provider "aws" {
  region="us-east-1"
}

resource "aws_instance" "instance1" {
  ami = "ami-00beae93a2d981137"
  key_name = "Master-Client"
  instance_type = "t2.micro"
  tags = {
    Name = "terraform-instance"
  }
}
```

58. Now, I am last one destroying

```
ubuntu@Terraform-Demo:~/ X + | ~
ubuntu@Terraform-Demo:~/demo$ terraform destroy
```

59. we can see the 2 destroys

```
Plan: 0 to add, 0 to change, 2 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.instance1: Destroying... [id=i-0a49adb4764123128]
aws_instance.instance1: Still destroying... [id=i-0a49adb4764123128, 10s elapsed]
aws_instance.instance1: Still destroying... [id=i-0a49adb4764123128, 20s elapsed]
aws_instance.instance1: Still destroying... [id=i-0a49adb4764123128, 30s elapsed]
aws_instance.instance1: Still destroying... [id=i-0a49adb4764123128, 40s elapsed]
aws_instance.instance1: Destruction complete after 40s
aws_key_pair.sshkey: Destroying... [id=sshkeydemo]
aws_key_pair.sshkey: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
ubuntu@Terraform-Demo:~/demo$
```

## 60. its terminated

| Instances (1/14) <a href="#">Info</a>                              |                             | <a href="#">C</a>  | <a href="#">Connect</a>         | <a href="#">Instance state ▾</a> | <a href="#">Actions ▾</a>     | <a href="#">Launch instances</a> ▾  |
|--|-----------------------------|--|---------------------------------|----------------------------------|-------------------------------|-------------------------------------|
| <a href="#">Find Instance by attribute or tag (case-sensitive)</a> |                             |  |                                 | <a href="#">All states ▾</a>     |                               |                                     |
| <a href="#">Name ▾</a>   | <a href="#">Instance ID</a> | <a href="#">Instance state ▾</a>                               | <a href="#">Instance type ▾</a> | <a href="#">Status check</a>     | <a href="#">Alarm status</a>  | <a href="#">Availability Zone ▾</a> |
| <input checked="" type="checkbox"/> terraform-instance             | i-0a49adb4764123128         | <a href="#">Terminated</a> <a href="#">Q</a> <a href="#">Q</a> | t2.micro                        | -                                | <a href="#">View alarms</a> + | us-east-1a                          |
| <input type="checkbox"/> Target-Server                             | i-0b33a2c181e7d1b9e         | <a href="#">Stopped</a> <a href="#">Q</a> <a href="#">Q</a>    | t2.micro                        | -                                | <a href="#">View alarms</a> + | us-east-1a                          |
| <input type="checkbox"/> DockerDemo2                               | i-07443e651f82bbe47         | <a href="#">Stopped</a> <a href="#">Q</a> <a href="#">Q</a>    | t2.micro                        | -                                | <a href="#">View alarms</a> + | us-east-1a                          |
| <input type="checkbox"/> terraform-instance                        | i-0dfa3far4a48rl56rs        | <a href="#">Terminated</a> <a href="#">Q</a> <a href="#">Q</a> | t2.micro                        | -                                | <a href="#">View alarms</a> + | us-east-1a                          |

**i-0a49adb4764123128 (terraform-instance)**

|                  |                             |  |
|------------------|-----------------------------|--|
| Default          | normal                      | Disabled                                 |
| AMI Launch index | Key pair assigned at launch | State transition reason                  |
| 0                | <a href="#">sshkeydemo</a>  | User initiated (2024-06-08 12:11:48 GMT) |

## 61. Now I am adding SG(Security Group)

| <a href="#">Find resources by attribute or tag</a> |                      |                                      |                                     |
|--|----------------------|--------------------------------------|-------------------------------------|
| <input type="checkbox"/>                           | <a href="#">Name</a> | <a href="#">Security group ID</a>    | <a href="#">Security group name</a> |
| <input type="checkbox"/>                           | -                    | <a href="#">sg-0c10ed06ddfd3367b</a> | nfs                                 |
| <input type="checkbox"/>                           | Master-SG            | <a href="#">sg-0a9bb37fb12663cd1</a> | launch-wizard-3                     |
| <input type="checkbox"/>                           | -                    | <a href="#">sg-0b2ff2bf490e20073</a> | launch-wizard-1                     |
| <input type="checkbox"/>                           | -                    | <a href="#">sg-0f3245b0ce1ce713f</a> | default                             |

62. Here we can see the key\_name and instance\_type and vpc\_security\_group\_ids and provisioner remote-exec connecting through type ssh and user ubuntu and private key and host = self.public\_ip

Inline in sided it will be print what we written in echo and it will install nano and git and save it.

```
ubuntu@Terraform-Demo: ~/ ~ + v
GNU nano 7.2
provider "aws" {
  region="us-east-1"
}

resource "aws_instance" "instance1" {
  ami = "ami-00beae93a2d981137"
  key_name = "Master-Client"
  instance_type = "t2.micro"
  vpc_security_group_ids = ["sg-0a9bb37fb12663cd1"]
  tags = {
    Name = "terraform-instance"
  }

  provisioner "remote-exec" {
    connection {
      type = "ssh"
      user = "ubuntu"
      private_key = file("")
      host = self.public_ip
    }

    inline = [
      "echo This command is excuted remotely on ec2 instance",
      "sudo apt-get install -y nano",
      "sudo apt-get install -y git",
    ]
  }
}
```

63. Come to home creating private key to connect remote server(exec)

```
ubuntu@Terraform-Demo:~/demo$ nano main.tf
ubuntu@Terraform-Demo:~/demo$ nano main.tf
ubuntu@Terraform-Demo:~/demo$ cd ~
ubuntu@Terraform-Demo:~$ nano Master-Client.pem
```

64. Go to local machine key pair location and copy that key pair and paste here

```
GNU nano 7.2                                         Master-Client.pem *
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAiRbt+42IpYpwVuKMzLKHyr73m8YBa+f2mXwzw5/Vh4+pu4
bnDxccK+xj0/e8608ZiglgNWTEB2sLHGc3B3exKpv/VLP0Dfj4jpjN4R0WoapYHx
RXez5EVmfenJl5WBeUAYr7y8Jf9vRbBc2SM0k4R51uRN71gseEmaq7fSF/uumxf+
NH2HQsFuyVcLqQqGvDRoIW9Reoo0JipX8/M+RCqVQwlm2XU2vhRUMNoY0S7bSM3D
3aMZow5RpAP5UmxN+7tsiZad910sI/7bjiaEP/Bla/d7bIns1fMB47D3uVFXAev/
5PqnpgZG3uQjWzp0t7QljQp989T7KR978sLwIDAQABoIBABgCRJff0n2WXbPu
ZPXigQcSQ7nLUdJmjYEZtpgVXBdgzsqAVDe2xt6h93BFJ2b3W+1sxLzfK8Yb3Hm
XuMjZNoat77WoRpwK3hBm96/mtFaS1aH+oPBViQYf844H8Y80pRZSUd8wFvLdnR4
/BpqT0ozXv/jtz7f69IZ2kyPgu4ONjjS74tx7X7nKFHzkEgrlIQyjqF4A0CdUVAL
+cWtrbyjLn1WBULIPcvI2d9frWmTleZGssCLfLPKCxqBqW0Es+sK0psnfVn47fM
sM/byb9HHu8/T0qnD8YceM5w+vimmUdkA5Ibi26A07gGo4lFryAtk/z0AMYnch0Q
k0wc/QECgYEA04HLfKR/szwtg0BkgNuSzmjx8Qdp2NsD9N1Tc0Nw00Edilfo2CEI
LM/y5A7YKt8vAJVPzPHNSnClui0BTLMBl2jF6VsQz4i4686/gDDkdh0AHu02FZ/R
/fpBSyFI50xTjGS3ko/F0Npwn4YWW0Th+HdnsUvbWC8I2M5y30Usq8CgYEapeii
HRuLWv+sC5/gRaEGDkfTin1+kzcHwk5RtlUKY7MHstb1oZY2qUIqbub0LPHLOK2
p5Ev064hKXvEEaMFOL+DMgQECEi7wUEZm+SltLxD1a96n2fRdxdz6vqn6eho30yG
0JoppUC9+zXQ8A3t6gzCPmaHazrlmJZA+Akbl0EcgYEAkDihGIzSxtTx892kqnk0
0dCdqUz0z/oH6KBClwYVmPd9vSFUjgt4F5Z0eS3mGNW2px8pGwaGhYvqddjaWx4e
YMWArmodTwzbke+YhUC/zWpY+r2lc3lxJ2fNvf0kR3UeQ00p2MvQH8RTp3M3KcZo
j5RFg13RyTN9FmRCtIhLReEcgYBF80hGtRHutIspF17fQj8irMXMwhLL1GeNF4m
K3uQ6ouKf9hbe4+BvQkF5N0vAlr5BSpTQcqjkiLQxihF9x+AaNv5/c9mKJHySmmt
Whqycpt2Pd/Hy5B1oMHxUXHwHdX3LI846YSc8colW06YBIdDn+BJPhfgkJex18XT
x+0+AQKBgEB0Lpd4T6Dieomql+szj18xMqlQeSGcuLxIG61wPPnkBVUlmjZdHGz5
PdVIrbBq6B/783P7irTwgRk0Tc/bP1T0LJ7bTDejDiVwzo2nIjsbJNzvCFVpev74
HNIQpcd08Nwb1HPe7GrvoEuXUr0kXcgTnAxHjLJgW+x+5G74wkFB
-----END RSA PRIVATE KEY-----
```

65. Provide permission to key-pair to read the remote server

```
ubuntu@Terraform-Demo:~/demo$ nano main.tf
ubuntu@Terraform-Demo:~/demo$ nano main.tf
ubuntu@Terraform-Demo:~/demo$ cd ~
ubuntu@Terraform-Demo:~$ nano Master-Client.pem
ubuntu@Terraform-Demo:~$ sudo chmod 600 Master-Client.pem
ubuntu@Terraform-Demo:~$ ls
Master-Client.pem  aws  awscliv2.zip  demo  terraform.sh
ubuntu@Terraform-Demo:~$ pwd
/home/ubuntu
ubuntu@Terraform-Demo:~$ cd demo
ubuntu@Terraform-Demo:~/demo$ ls
main.tf  terraform.tfstate  terraform.tfstate.backup
ubuntu@Terraform-Demo:~/demo$ nano main.tf
```

66. Now Came here paste the key-pair path in provisioner “remote-exec”, here changing AMI previously I am using Amazon AMI

```
ubuntu@Terraform-Demo: ~/ + v
GNU nano 7.2                                     main.tf

provider "aws" {
  region="us-east-1"
}

resource "aws_instance" "instance1" {
  ami = "ami-04b70fa74e45c3917"
  key_name = "Master-Client"
  instance_type = "t2.micro"
  vpc_security_group_ids = ["sg-0a9bb37fb12663cd1"]
  tags = {
    Name = "terraform-instance"
  }

  provisioner "remote-exec" {
    connection {
      type = "ssh"
      user = "ubuntu"
      private_key = file("/home/ubuntu/Master-Client.pem")
      host = self.public_ip
    }

    inline = [
      "echo This command is excuted remotely on ec2 instance",
      "sudo apt-get install -y nano",
      "sudo apt-get install -y git",
    ]
  }
}
```

67. terraform plan for me its getting error so I use below the command

```
ubuntu@Terraform-Demo: ~/ + v
ubuntu@Terraform-Demo:~/demo$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
```

68. terraform plan -lock=false

```
ubuntu@Terraform-Demo: ~/ + v
ubuntu@Terraform-Demo:~/demo$ terraform plan -lock=false

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.instance1 will be created
+ resource "aws_instance" "instance1" {
  + ami                               = "ami-04b70fa74e45c3917"
  + 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
```

## 69. Now its plan to 1 to add

```
[  
}  
  
Plan: 1 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.
```

## 70. yes, and it will start create

```
ubuntu@Terraform-Demo: ~ / +   
  
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.instance1: Creating...  
aws_instance.instance1: Still creating... [10s elapsed]  
aws_instance.instance1: Still creating... [20s elapsed]  
aws_instance.instance1: Still creating... [30s elapsed]  
aws_instance.instance1: Provisioning with 'remote-exec'...  
aws_instance.instance1 (remote-exec): Connecting to remote host via SSH...  
aws_instance.instance1 (remote-exec): Host: 54.159.45.151  
aws_instance.instance1 (remote-exec): User: ubuntu  
aws_instance.instance1 (remote-exec): Password: false  
aws_instance.instance1 (remote-exec): Private key: true  
aws_instance.instance1 (remote-exec): Certificate: false  
aws_instance.instance1 (remote-exec): SSH Agent: false  
aws_instance.instance1 (remote-exec): Checking Host Key: false  
aws_instance.instance1 (remote-exec): Target Platform: unix  
aws_instance.instance1 (remote-exec): Connecting to remote host via SSH...  
aws_instance.instance1 (remote-exec): Host: 54.159.45.151  
aws_instance.instance1 (remote-exec): User: ubuntu  
aws_instance.instance1 (remote-exec): Password: false  
aws_instance.instance1 (remote-exec): Private key: true  
aws_instance.instance1 (remote-exec): Certificate: false  
aws_instance.instance1 (remote-exec): SSH Agent: false  
aws_instance.instance1 (remote-exec): Checking Host Key: false  
aws_instance.instance1 (remote-exec): Target Platform: unix  
aws_instance.instance1: Still creating... [40s elapsed]  
aws_instance.instance1 (remote-exec): Connecting to remote host via SSH...  
aws_instance.instance1 (remote-exec): Host: 54.159.45.151
```

## 71. Now its connecting to remote host

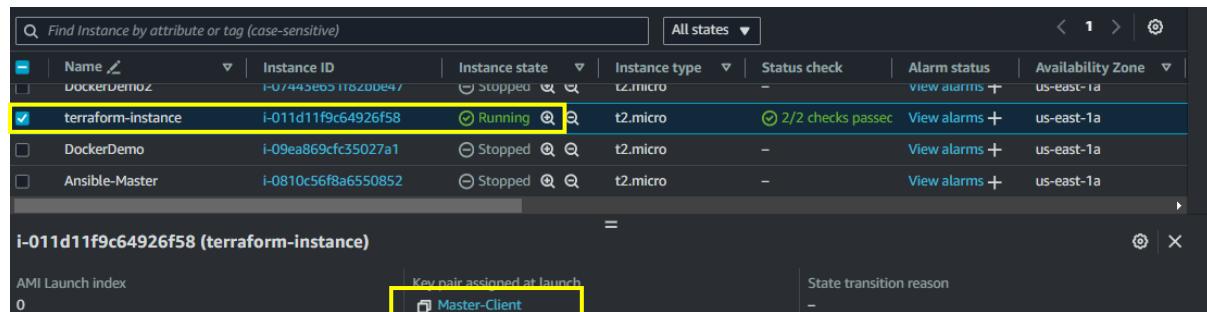
```
ubuntu@Terraform-Demo: ~ / +   
  
aws_instance.instance1 (remote-exec): Connecting to remote host via SSH...  
aws_instance.instance1 (remote-exec): Host: 54.159.45.151  
aws_instance.instance1 (remote-exec): User: ubuntu  
aws_instance.instance1 (remote-exec): Password: false  
aws_instance.instance1 (remote-exec): Private key: true  
aws_instance.instance1 (remote-exec): Certificate: false  
aws_instance.instance1 (remote-exec): SSH Agent: false  
aws_instance.instance1 (remote-exec): Checking Host Key: false  
aws_instance.instance1 (remote-exec): Target Platform: unix  
aws_instance.instance1 (remote-exec): Connecting to remote host via SSH...  
aws_instance.instance1 (remote-exec): Host: 54.159.45.151  
aws_instance.instance1 (remote-exec): User: ubuntu  
aws_instance.instance1 (remote-exec): Password: false  
aws_instance.instance1 (remote-exec): Private key: true  
aws_instance.instance1 (remote-exec): Certificate: false  
aws_instance.instance1 (remote-exec): SSH Agent: false  
aws_instance.instance1 (remote-exec): Checking Host Key: false  
aws_instance.instance1 (remote-exec): Target Platform: unix  
aws_instance.instance1 (remote-exec): Connected!  
aws_instance.instance1 (remote-exec): This command is excuted remotely on ec2 instance  
aws_instance.instance1 (remote-exec): Reading package lists... 0%  
aws_instance.instance1 (remote-exec): Reading package lists... 0%  
aws_instance.instance1 (remote-exec): Reading package lists... 0%  
aws_instance.instance1 (remote-exec): Still creating... [50s elapsed]  
aws_instance.instance1 (remote-exec): Reading package lists... 60%  
aws_instance.instance1 (remote-exec): Reading package lists... 60%  
aws_instance.instance1 (remote-exec): Reading package lists... 86%  
aws_instance.instance1 (remote-exec): Reading package lists... 86%  
aws_instance.instance1 (remote-exec): Reading package lists... 92%  
aws_instance.instance1 (remote-exec): Reading package lists... 92%  
aws_instance.instance1 (remote-exec): Reading package lists... 94%  
aws_instance.instance1 (remote-exec): Reading package lists... 94%  
aws_instance.instance1 (remote-exec): Reading package lists... Done  
aws_instance.instance1 (remote-exec): Building dependency tree... 0%  
aws_instance.instance1 (remote-exec): Building dependency tree... 0%
```

## 72. Creation completed

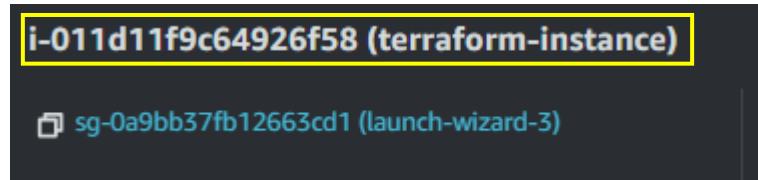
```
aws_instance.instance1 (remote-exec): Reading package lists... Done
aws_instance.instance1 (remote-exec): Building dependency tree... 0%
aws_instance.instance1 (remote-exec): Building dependency tree... 0%
aws_instance.instance1 (remote-exec): Building dependency tree... 50%
aws_instance.instance1 (remote-exec): Building dependency tree... 50%
aws_instance.instance1 (remote-exec): Building dependency tree... Done
aws_instance.instance1 (remote-exec): Reading state information... 0%
aws_instance.instance1 (remote-exec): Reading state information... 0%
aws_instance.instance1 (remote-exec): Reading state information... Done
aws_instance.instance1 (remote-exec): git is already the newest version (1:2.43.0-1ubuntu7).
aws_instance.instance1 (remote-exec): git set to manually installed.
aws_instance.instance1 (remote-exec): 0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
aws_instance.instance1: Creation complete after 51s [id=i-011d11f9c64926f58]
```

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
ubuntu@Terraform-Demo:~/demo$
```

## 73. We can see the instance with Master-client key-pair which I gave that



## 74. We can see the security group also attached to terraform instance



```
ubuntu@Terraform-Demo:~/demo$ cd demo
ubuntu@Terraform-Demo:~/demo$ ls -la
total 32
drwxrwxr-x  3 ubuntu ubuntu 4096 Jun  8 13:06 .
drwxr-x--- 10 ubuntu ubuntu 4096 Jun  8 12:30 ..
drwxr-xr-x  3 ubuntu ubuntu 4096 Jun  8 10:29 .terraform
-rw-r--r--  1 ubuntu ubuntu 1377 Jun  8 10:29 .terraform.lock.hcl
-rw-rw-r--  1 ubuntu ubuntu   611 Jun  8 13:06 main.tf
-rw-rw-r--  1 ubuntu ubuntu 4877 Jun  8 13:12 terraform.tfstate
-rw-rw-r--  1 ubuntu ubuntu  181 Jun  8 13:12 terraform.tfstate.backup
ubuntu@Terraform-Demo:~/demo$
```

## 75. Now destroy the infrastructure

```
ubuntu@Terraform-Demo:~/demo$ terraform destroy
```

76. its destroyed

```
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.instance1: Destroying... [id=i-011d11f9c64926f58]
aws_instance.instance1: Still destroying... [id=i-011d11f9c64926f58, 10s elapsed]
aws_instance.instance1: Still destroying... [id=i-011d11f9c64926f58, 20s elapsed]
aws_instance.instance1: Still destroying... [id=i-011d11f9c64926f58, 30s elapsed]
aws_instance.instance1: Still destroying... [id=i-011d11f9c64926f58, 40s elapsed]
aws_instance.instance1: Destruction complete after 40s

Destroy complete! Resources: 1 destroyed.
```

77. Now I again come to modification

```
ubuntu@Terraform-Demo:~/demo$ nano main.tf
```

78. Now Creating 3 instances and here key\_name = Master-Client which is available on us-east-1 it will took from there and in the tags goint to run \${count.index + 1}

```
ubuntu@Terraform-Demo:~/demo$ nano main.tf
main.tf *
GNU nano 7.2
provider "aws" {
  region="us-east-1"
}

resource "aws_instance" "instance1" {
  count = 3
  ami = "ami-04b70fa74e45c3917"
  key_name = "Master-Client"
  instance_type = "t2.micro"
  vpc_security_group_ids = ["sg-0a9bb37fb12663cd1"]
  tags = {
    Name = "terraform-instance-${count.index + 1}"
  }
}
```

79. terraform plan

```
ubuntu@Terraform-Demo:~/demo$ terraform plan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
+ create
```

80. it will add 3

```
ubuntu@Terraform-Demo:~/demo$ terraform plan
Plan: 3 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.
```

81. Now terraform apply

```
ubuntu@Terraform-Demo: ~/demo$ terraform apply
```

82. yes, and then it will create all three instances

```
Plan: 3 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

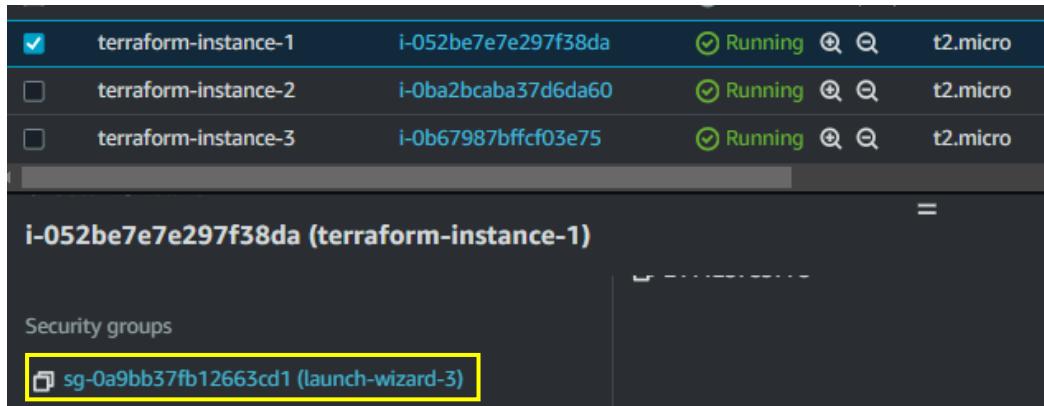
aws_instance.instance1[0]: Creating...
aws_instance.instance1[1]: Creating...
aws_instance.instance1[2]: Creating...
aws_instance.instance1[1]: Still creating... [10s elapsed]
aws_instance.instance1[0]: Still creating... [10s elapsed]
aws_instance.instance1[2]: Still creating... [10s elapsed]
aws_instance.instance1[0]: Still creating... [20s elapsed]
aws_instance.instance1[1]: Still creating... [20s elapsed]
aws_instance.instance1[2]: Still creating... [20s elapsed]
aws_instance.instance1[1]: Still creating... [30s elapsed]
aws_instance.instance1[0]: Still creating... [30s elapsed]
aws_instance.instance1[2]: Still creating... [30s elapsed]
aws_instance.instance1[2]: Creation complete after 32s [id=i-0b67987bfff03e75]
aws_instance.instance1[0]: Creation complete after 32s [id=i-052be7e7e297f38da]
aws_instance.instance1[1]: Creation complete after 32s [id=i-0ba2bcaba37d6da60]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
ubuntu@Terraform-Demo:~/demo$
```

83. running three instances

|  |                      |                             |                      |                  |                         |                           |                               |            |
|--|----------------------|-----------------------------|----------------------|------------------|-------------------------|---------------------------|-------------------------------|------------|
| <input checked="" type="checkbox"/>                        | terraform-instance-1 | i-052be7e7e297f38da         | <span>Running</span> | <span>Q Q</span> | t2.micro                | <span>Initializing</span> | <a href="#">View alarms</a> + | us-east-1a |
| <input type="checkbox"/>                                   | terraform-instance-2 | i-0ba2bcaba37d6da60         | <span>Running</span> | <span>Q Q</span> | t2.micro                | <span>Initializing</span> | <a href="#">View alarms</a> + | us-east-1a |
| <input type="checkbox"/>                                   | terraform-instance-3 | i-0b67987bfff03e75          | <span>Running</span> | <span>Q Q</span> | t2.micro                | <span>Initializing</span> | <a href="#">View alarms</a> + | us-east-1a |
| =  |                      |                             |                      |                  |                         |                           |                               |            |
| <a href="#">i-052be7e7e297f38da (terraform-instance-1)</a> |                      |                             |                      |                  |                         |                           |                               |            |
| Default  |                      | normal                      |                      |                  | Disabled                |                           |                               |            |
| AMI Launch index   | 0                    | Key pair assigned at launch |                      |                  | State transition reason |                           |                               |            |
|  |                      | <span>Master-Client</span>  |                      |                  | -                       |                           |                               |            |

84. If we want to delete Particular instances



85. follow the command

```
ubuntu@Terraform-Demo:~/demo$ terraform destroy -target=aws_instance.instance1[1] -lock=false
aws_instance.instance1[1]: Refreshing state... [id=i-0ba2bcaba37d6da60]

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.instance1[1] will be destroyed
- resource "aws_instance" "instance1" {
  - ami                               = "ami-04b70fa74e45c3917" -> null
  - arn                               = "arn:aws:ec2:us-east-1:211125783778:instance/i-0ba2bcaba37d6da60" -> null
  - associate_public_ip_address       = true -> null
  - availability_zone                 = "us-east-1a" -> null
}
```

86. it will destroying the instances

```
aws_instance.instance1[1]: Destroying... [id=i-0ba2bcaba37d6da60]
aws_instance.instance1[1]: Still destroying... [id=i-0ba2bcaba37d6da60, 10s elapsed]
aws_instance.instance1[1]: Still destroying... [id=i-0ba2bcaba37d6da60, 20s elapsed]
aws_instance.instance1[1]: Still destroying... [id=i-0ba2bcaba37d6da60, 30s elapsed]
aws_instance.instance1[1]: Destruction complete after 40s

Warning: Applied changes may be incomplete

The plan was created with the -target option in effect, so some changes requested in the configuration may have been ignored and the output values may not be fully updated. Run the following command to verify that no other changes are pending:
  terraform plan

Note that the -target option is not suitable for routine use, and is provided only for exceptional situations such as recovering from errors or mistakes, or when Terraform specifically suggests to use it as part of an error message.

Destroy complete! Resources: 1 destroyed.
ubuntu@Terraform-Demo:~/demo$
```

87. its destroyed. That particular instances and destroy all because day 1 completed

|                                     |                      |                     |                         |                  |          |                                |                            |            |
|-------------------------------------|----------------------|---------------------|-------------------------|------------------|----------|--------------------------------|----------------------------|------------|
| <input checked="" type="checkbox"/> | terraform-instance-1 | i-052be7e7e297f38da | <span>Running</span>    | <span>Q Q</span> | t2.micro | <span>2/2 checks passed</span> | <span>View alarms +</span> | us-east-1a |
| <input type="checkbox"/>            | terraform-instance-2 | i-0ba2bcaba37d6da60 | <span>Terminated</span> | <span>Q Q</span> | t2.micro | -                              | <span>View alarms +</span> | us-east-1a |
| <input type="checkbox"/>            | terraform-instance-3 | i-0b67987bfff03e75  | <span>Running</span>    | <span>Q Q</span> | t2.micro | <span>2/2 checks passed</span> | <span>View alarms +</span> | us-east-1a |

## DAY-2

### Agenda: Terraform

1. What is infrastructure as a code?
2. Infrastructure as a code vs Configuration Management
3. Intro to Terraform
4. Installing Terraform
5. Terraform Operations
6. Terraform code
7. Terraform Demo

#### 1. Launch Instances



The screenshot shows the AWS CloudWatch Metrics console. A metric named "immutability" is displayed with a value of 0. The metric has a timestamp of "0EWQw84XngI".

**Instances (1/1) Info**

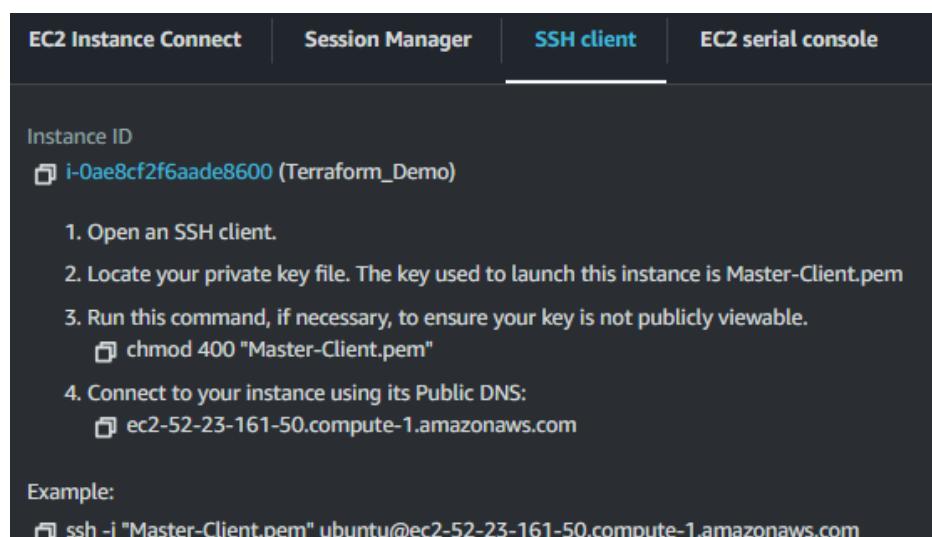
Instances table:

| Name           | Instance ID         | Instance state | Instance type | Status check | Alarm status  | Availability Zone |
|----------------|---------------------|----------------|---------------|--------------|---------------|-------------------|
| Terraform_Demo | i-0ae8cf2f6aade8600 | Running        | t2.micro      | Initializing | View alarms + | us-east-1a        |

Instance details for i-0ae8cf2f6aade8600 (Terraform\_Demo):

- Details tab selected.
- Public IPv4 address: 52.23.161.50 (with "open address" link)
- Private IPv4 addresses: 172.31.22.14

#### 2. Copying the SSH-client



**EC2 Instance Connect**    **Session Manager**    **SSH client**    **EC2 serial console**

Instance ID: i-0ae8cf2f6aade8600 (Terraform\_Demo)

- Open an SSH client.
- Locate your private key file. The key used to launch this instance is Master-Client.pem
- Run this command, if necessary, to ensure your key is not publicly viewable.  
chmod 400 "Master-Client.pem"
- Connect to your instance using its Public DNS:  
ec2-52-23-161-50.compute-1.amazonaws.com

Example:

```
ssh -i "Master-Client.pem" ubuntu@ec2-52-23-161-50.compute-1.amazonaws.com
```

### 3. Paste and connect the instances

```
C:\WINDOWS\system32\cmd. X + ▾  
Microsoft Windows [Version 10.0.22631.3593]  
(c) Microsoft Corporation. All rights reserved.  
C:\Users\shaik>cd C:\Users\shaik\Desktop\Cloud Computing\Aws-key pairs  
C:\Users\shaik\Desktop\Cloud Computing\Aws-key pairs>ssh -i "Master-Client.pem" ubuntu@ec2-52-23-161-50.compute-1.amazonaws.com
```

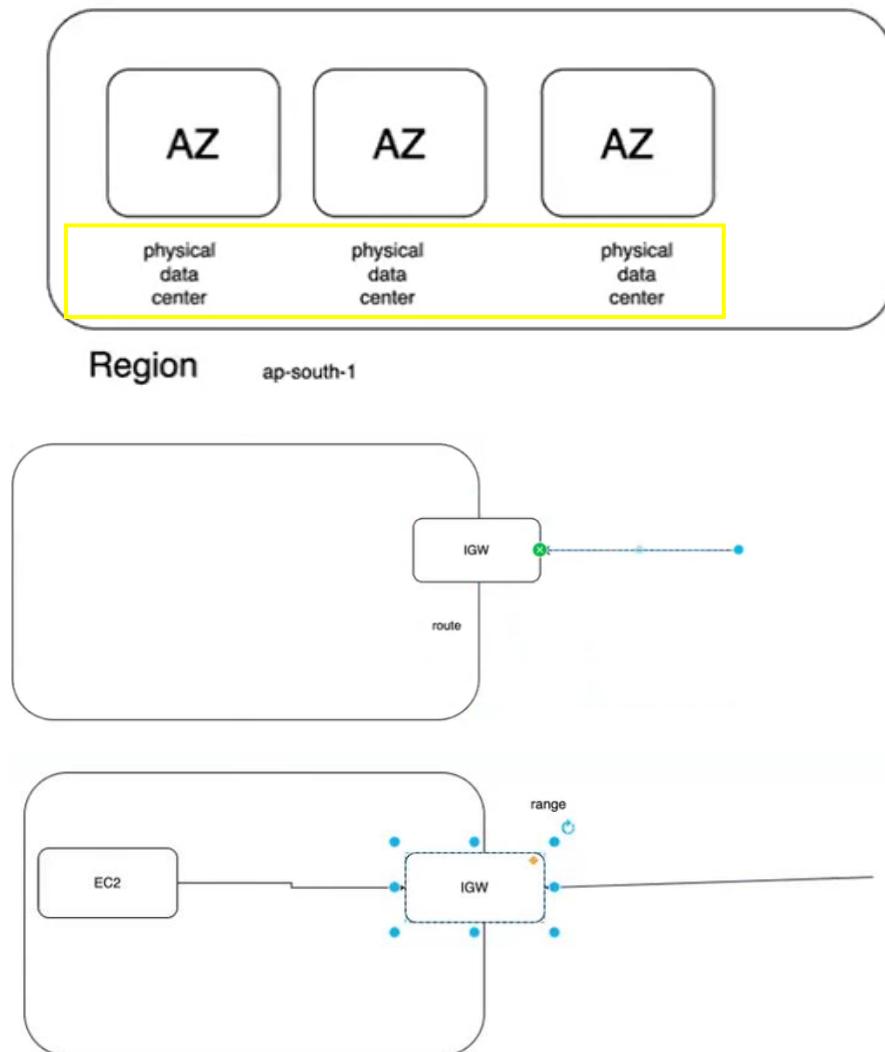
### 4. Now Creating Another folder Named as terraform

```
ubuntu@Terraform-Demo:~/ X + ▾  
ubuntu@Terraform-Demo:~$ ls  
aws awscliv2.zip demo terraform.sh  
ubuntu@Terraform-Demo:~$ mkdir terraform  
ubuntu@Terraform-Demo:~$ ls  
aws awscliv2.zip demo terraform terraform.sh  
ubuntu@Terraform-Demo:~$ cd terraform  
ubuntu@Terraform-Demo:~/terraform$ nano main.tf
```

### 5. Now Creating Vpc and giving cidr\_block range 10.0.0.0/16 and adding subnet resources

```
GNU nano 7.2 main.tf  
provider "aws" {  
    region = "us-east-1"  
}  
  
resource "aws_vpc" "my_vpc" {  
    cidr_block = "10.0.0.0/16"  
    enable_dns_support = true  
    enable_dns_hostnames =true  
}  
  
resource "aws_subnet" "my_subnet" {  
    vpc_id = aws_vpc.my_vpc.id  
    cidr_block = "10.0.1.0/24"  
    availability_zone = "us-east-1a"  
}  
  
resource "aws_internet_gateway" "my_igw" {  
    vpc_id = aws_vpc.my_vpc.id  
}
```

## Creating resources in Availability Zone



6. Continue attaching Route Table and cidr to everyone, and adding subnet resources

```
resource "aws_route_table" "my_route_table" {
  vpc_id = aws_vpc.my_vpc.id

  route {
    cidr_block = "0.0.0.0/0"
  }
}

resource "aws_route_table_association" "my_subnet_association" {
  subnet_id = aws_subnet.my_subnet.id
  route_table_id = aws_route_table.my_route_table.id
}
```

7. Now Allowing security groups are ssh and tcp port and particularly 22 and 80

```
resource "aws_security_group" "my_security_group" {
  vpc_id = aws_vpc.my_vpc.id

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

  ingress {
    from_port = 80
    to_port = 80
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

8. and Adding instances Resources

```
resource "aws_instance" "my_instance" {
  ami = "ami-04b70fa74e45c3917"
  instance_type = "t2.micro"
  subnet_id = aws_subnet.my_subnet.id
  key_name = "Master-Client"
  security_groups = [aws_security_group.my_security_group.id]

  user_data = file("")
  tags = {
    Name = "MyEC2Instance"
  }
}
```

9. Now writing nginx\_install.sh for user data

```
ubuntu@Terraform-Demo:~/terraform$ nano nginx_install.sh
```

10. Write commands and save it.

```
ubuntu@Terraform-Demo:~/terraform$ nano nginx_install.sh *
GNU nano 7.2
#!/bin/bash
sudo apt update -y
sudo apt install nginx -y
sudo systemctl enable nginx -y
```

11. Now copy the path

```
ubuntu@Terraform-Demo:~/terraform$ nano nginx_install.sh
ubuntu@Terraform-Demo:~/terraform$ ls
main.tf  nginx_install.sh
ubuntu@Terraform-Demo:~/terraform$ pwd
/home/ubuntu/terraform
ubuntu@Terraform-Demo:~/terraform$ nano main.tf
```

12. Come into main.tf file and paste the path

```
resource "aws_instance" "my_instance" {
  ami = "ami-04b70fa74e45c3917"
  instance_type = "t2.micro"
  subnet_id = aws_subnet.my_subnet.id
  key_name = "Master-Client"
  security_groups = [aws_security_group.my_security_group.id]

  user_data = file("/home/ubuntu/terraform/nginx_install.sh")
  tags = {
    Name = "MyEC2Instance"
  }
}
```

Follow the code:

Note: This is edited code in below I got an errors, but this is code is rectified one.

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_vpc" "my_vpc" {
  cidr_block = "10.0.0.0/16"
  enable_dns_support = true
  enable_dns_hostnames = true
}

resource "aws_subnet" "my_subnet" {
```

```
vpc_id = aws_vpc.my_vpc.id  
cidr_block = "10.0.6.0/24"  
availability_zone = "us-east-1a"  
}
```

```
resource "aws_internet_gateway" "my_igw" {  
vpc_id = aws_vpc.my_vpc.id  
}
```

```
resource "aws_route_table" "my_route_table" {  
vpc_id = aws_vpc.my_vpc.id
```

```
route {  
cidr_block = "0.0.0.0/0"  
gateway_id = aws_internet_gateway.my_igw.id  
}  
}
```

```
resource "aws_route" "my_route" {  
route_table_id = aws_route_table.my_route_table.id  
gateway_id = aws_internet_gateway.my_igw.id  
destination_cidr_block = "0.0.0.0/0"  
}
```

```
resource "aws_route_table_association" "my_subnet_association" {  
subnet_id = aws_subnet.my_subnet.id  
route_table_id = aws_route_table.my_route_table.id  
}
```

```
resource "aws_security_group" "my_security_group" {  
vpc_id = aws_vpc.my_vpc.id
```

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

```
ingress {  
    from_port = 80  
    to_port = 80  
    protocol = "tcp"  
    cidr_blocks = ["0.0.0.0/0"]  
}  
}
```

```
resource "aws_instance" "my_instance" {  
    ami = "ami-04b70fa74e45c3917"  
    instance_type = "t2.micro"  
    subnet_id = aws_subnet.my_subnet.id  
    key_name = "Master-Client"  
    security_groups = [aws_security_group.my_security_group.id]  
  
    user_data = file("/home/ubuntu/terraform/nginx_install.sh")  
    tags = {  
        Name = "MyEC2Instance"  
    }  
}
```

### 13. terraform init

```
ubuntu@Terraform-Demo:~/terraform$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.53.0...
- Installed hashicorp/aws v5.53.0 (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.
```

### 14. terraform plan and we got an error

```
ubuntu@Terraform-Demo:~/terraform$ ls -a
. ... .terraform .terraform.lock.hcl main.tf nginx_install.sh
ubuntu@Terraform-Demo:~/terraform$ terraform plan
Error: Invalid combination of arguments

  with aws_route.my_route,
  on main.tf line 29, in resource "aws_route" "my_route":
  29: resource "aws_route" "my_route" {

"destination_ipv6_cidr_block": one of `destination_cidr_block,destination_ipv6_cidr_block,destination_prefix_list_id` must be specified

Error: Invalid combination of arguments

  with aws_route.my_route,
  on main.tf line 29, in resource "aws_route" "my_route":
  29: resource "aws_route" "my_route" {

"destination_prefix_list_id": one of `destination_cidr_block,destination_ipv6_cidr_block,destination_prefix_list_id` must be specified

Error: Invalid combination of arguments

  with aws_route.my_route,
  on main.tf line 29, in resource "aws_route" "my_route":
  29: resource "aws_route" "my_route" {

"destination_cidr_block": one of `destination_cidr_block,destination_ipv6_cidr_block,destination_prefix_list_id` must be specified
```

### 15. Use above the code

```
resource "aws_route" "my_route" {
  route_table_id = aws_route_table.my_route_table.id
  gateway_id = aws_internet_gateway.my_igw.id
  destination_cidr_block = "0.0.0.0/0"
}

resource "aws_route_table_association" "my_subnet_association" {
  subnet_id = aws_subnet.my_subnet.id
  route_table_id = aws_route_table.my_route_table.id
}
```

## 16. Now terraform init and plan

```
ubuntu@Terraform-Demo:~/terraform$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.53.0

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.
ubuntu@Terraform-Demo:~/terraform$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
+ create
```

## 17. Now its Generating

```
Terraform will perform the following actions:

# aws_instance.my_instance will be created
+ resource "aws_instance" "my_instance" {
    + ami                                = "ami-04b70fa74e45c3917"
    + 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)"
    + instance.type                     = "t2.micro"
    + ipv6_address_count               = "(known after apply)"
    + ipv6_addresses                   = "(known after apply)"
    + key_name                          = "Master-Client"
    + 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)"
    + primary_network_interface_id     = "(known after apply)"
    + private_dns                       = "(known after apply)"
    + private_ip                        = "(known after apply)"
    + public_dns                        = "(known after apply)"
```

```
+ private_ip = (known after apply)
+ public_dns = (known after apply)
+ public_ip = (known after apply)
+ secondary_private_ips = (known after apply)
+ security_groups = (known after apply)
+ source_dest_check = true
+ spot_instance_request_id = (known after apply)
+ subnet_id = (known after apply)
+ tags =
  + "Name" = "MyEC2Instance"
}
+ tags_all = {
  + "Name" = "MyEC2Instance"
}
+ tenancy = (known after apply)
+ user_data = "e2f83e187ce0ce308f940d40cf54e4e7c59d12b5"
+ user_data_base64 = (known after apply)
+ user_data_replace_on_change = false
+ vpc_security_group_ids = (known after apply)
}

# aws_internet_gateway.my_igw will be created
+ resource "aws_internet_gateway" "my_igw" {
  + arn = (known after apply)
  + id = (known after apply)
  + owner_id = (known after apply)
  + tags_all = (known after apply)
  + vpc_id = (known after apply)
}

# aws_route.my_route will be created
+ resource "aws_route" "my_route" {
  + destination_cidr_block = "0.0.0.0/0"
  + gateway_id = (known after apply)
  + id = (known after apply)
```

```
+ id                      = (known after apply)
+ instance_id              = (known after apply)
+ instance_owner_id        = (known after apply)
+ network_interface_id    = (known after apply)
+ origin                  = (known after apply)
+ route_table_id           = (known after apply)
+ state                   = (known after apply)
}

# aws_route_table.my_route_table will be created
+ resource "aws_route_table" "my_route_table" {
  + arn                    = (known after apply)
  + id                     = (known after apply)
  + owner_id                = (known after apply)
  + propagating_vgwss      = (known after apply)
  + route                  = [
    +
    + {
      + cidr_block            = "0.0.0.0/0"
      # (12 unchanged attributes hidden)
    },
  ]
  + tags_all                = (known after apply)
  + vpc_id                  = (known after apply)
}

# aws_route_table_association.my_subnet_association will be created
+ resource "aws_route_table_association" "my_subnet_association" {
  + id                      = (known after apply)
  + route_table_id          = (known after apply)
  + subnet_id                = (known after apply)
}

# aws_security_group.my_security_group will be created
+ resource "aws_security_group" "my_security_group" {
  + arn                    = (known after apply)
```

```
+ id                      = (known after apply)
+ ingress                 = [
+ {
+   + cidr_blocks      = [
+     + "0.0.0.0/0",
+   ]
+   + from_port        = 22
+   + ipv6_cidr_blocks = []
+   + prefix_list_ids = []
+   + protocol          = "tcp"
+   + security_groups  = []
+   + self              = false
+   + to_port           = 22
+     # (1 unchanged attribute hidden)
},
{
+   + cidr_blocks      = [
+     + "0.0.0.0/0",
+   ]
+   + from_port        = 80
+   + ipv6_cidr_blocks = []
+   + prefix_list_ids = []
+   + protocol          = "tcp"
+   + security_groups  = []
+   + self              = false
+   + to_port           = 80
+     # (1 unchanged attribute hidden)
},
]
+
+ name                    = (known after apply)
+ 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)
```

```

+ vpc_id           = (known after apply)
}

# aws_subnet.my_subnet will be created
+ resource "aws_subnet" "my_subnet" {
    + arn           = (known after apply)
    + assign_ipv6_address_on_creation = false
    + availability_zone      = "us-east-1a"
    + availability_zone_id   = (known after apply)
    + cidr_block          = "10.0.1.0/24"
    + enable_dns64        = false
    + enable_resource_name_dns_a_record_on_launch = false
    + enable_resource_name_dns_aaaa_record_on_launch = false
    + id                = (known after apply)
    + ipv6_cidr_block_association_id = (known after apply)
    + ipv6_native        = false
    + map_public_ip_on_launch = false
    + owner_id          = (known after apply)
    + private_dns_hostname_type_on_launch = (known after apply)
    + tags_all          = (known after apply)
    + vpc_id            = (known after apply)
}

# aws_vpc.my_vpc will be created
+ resource "aws_vpc" "my_vpc" {
    + arn           = (known after apply)
    + cidr_block          = "10.0.0.0/16"
    + default_network_acl_id = (known after apply)
    + default_route_table_id = (known after apply)
    + default_security_group_id = (known after apply)
    + dhcp_options_id     = (known after apply)
    + enable_dns_hostnames = true
    + enable_dns_support  = true
    + enable_network_address_usage_metrics = (known after apply)
    + id                = (known after apply)
    + instance_tenancy   = "default"
    + ipv6_association_id = (known after apply)
    + ipv6_cidr_block    = (known after apply)
    + ipv6_cidr_block_network_border_group = (known after apply)
    + main_route_table_id = (known after apply)
    + owner_id          = (known after apply)
    + tags_all          = (known after apply)
}

```

Plan: 8 to add, 0 to change, 0 to destroy.

17. Again While doing terraform apply got an error so below the edited one. And above code is edited one

```

resource "aws_route_table" "my_route_table" {
  vpc_id = aws_vpc.my_vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.my_igw.id
  }
}

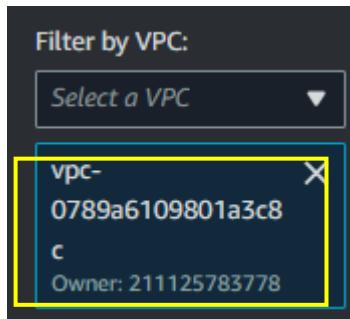
resource "aws_route" "my_route" {
  route_table_id = aws_route_table.my_route_table.id
  gateway_id = aws_internet_gateway.my_igw.id
  destination_cidr_block = "0.0.0.0/0"
}

```

## 18. Now Its Creating all the 7 resources

```
aws_vpc.my_vpc: Creating...
aws_vpc.my_vpc: Still creating... [10s elapsed]
aws_vpc.my_vpc: Creation complete after 12s [id=vpc-0789a6109801a3c8c]
aws_internet_gateway.my_igw: Creating...
aws_route_table.my_route_table: Creating...
aws_security_group.my_security_group: Creating...
aws_subnet.my_subnet: Creating...
aws_internet_gateway.my_igw: Creation complete after 0s [id=igw-0f2a2f84e64cc3487]
aws_subnet.my_subnet: Creation complete after 1s [id=subnet-059193af925bce55a]
aws_security_group.my_security_group: Creation complete after 2s [id=sg-01067e31fbe6ab135]
aws_instance.my_instance: Creating...
aws_instance.my_instance: Still creating... [10s elapsed]
aws_instance.my_instance: Still creating... [20s elapsed]
aws_instance.my_instance: Still creating... [30s elapsed]
```

## 19. VPC ID



The screenshot shows the AWS VPC dashboard. On the left, there's a sidebar with navigation links like 'EC2 Global View', 'Virtual private cloud', and 'Your VPCs'. The main area is titled 'Resources by Region' and shows various VPC components across regions. A specific VPC, 'vpc-0789a6109801a3c8', is selected and highlighted with a yellow border in the sidebar.

| Category                      | Region    | Count |
|-------------------------------|-----------|-------|
| VPCs                          | US East 2 | 2     |
| NAT Gateways                  | US East 0 | 0     |
| Subnets                       | US East 7 | 7     |
| VPC Peering Connections       | US East 0 | 0     |
| Route Tables                  | US East 3 | 3     |
| Network ACLs                  | US East 2 | 2     |
| Internet Gateways             | US East 2 | 2     |
| Security Groups               | US East 8 | 8     |
| Egress-only Internet Gateways | US East 0 | 0     |
| Customer Gateways             | US East 0 | 0     |
| DHCP option sets              | US East 1 | 1     |
| Virtual Private Gateways      | US East 0 | 0     |

## 19. This are the generated ids

```
ubuntu@Terraform-Demo:~/terraform$ terraform apply
aws_vpc.my_vpc: Refreshing state... [id=vpc-0789a6109801a3c8c]
aws_subnet.my_subnet: Refreshing state... [id=subnet-00d4f1ab6b3fad25f]
aws_internet_gateway.my_igw: Refreshing state... [id=igw-0f2a2f84e64cc3487]
aws_security_group.my_security_group: Refreshing state... [id=sg-01067e31fbe6ab135]
aws_route_table.my_route_table: Refreshing state... [id=rtb-0d1fec63bb68b48fb]
aws_instance.my_instance: Refreshing state... [id=i-096b7d0174367166a]
aws_route_table_association.my_subnet_association: Refreshing state... [id=rtbassoc-0c8f6041d1a61fb15]
```

20. this is subnet id

| Name | Subnet ID                | State     | VPC                   | IPv4 CIDR   |
|------|--------------------------|-----------|-----------------------|-------------|
| -    | subnet-00d4f1ab6b3fad25f | Available | vpc-0789a6109801a3c8c | 10.0.6.0/24 |

21. And this is IGW-ID

| VPC ID : vpc-0789a6109801a3c8c |                       | Clear filters | < 1 > @               |             |
|--------------------------------|-----------------------|---------------|-----------------------|-------------|
| Name                           | Internet gateway ID   | State         | VPC ID                | Owner       |
| -                              | igw-0f2a2f84e64cc3487 | Attached      | vpc-0789a6109801a3c8c | 21112578377 |

22. And this is Route Table ID and we can see the CIDR block and IGW attached

| VPC : vpc-0789a6109801a3c8c |                       | Clear filters             | < 1 > @           |      |                       |
|-----------------------------|-----------------------|---------------------------|-------------------|------|-----------------------|
| Name                        | Route table ID        | Explicit subnet associ... | Edge associations | Main | VPC                   |
| -                           | rtb-0d1fec63bb69b48fb | subnet-00d4f1ab6b3fad25f  | -                 | No   | vpc-0789a6109801a3c8c |
| -                           | rtb-0a669fcf83aa9ac16 | -                         | -                 | Yes  | vpc-0789a6109801a3c8c |

Details | **Routes** | Subnet associations | Edge associations | Route propagation | Tags

Both | Edit routes | < 1 > @

**Routes (2)**

| Destination | Target                | Status | Propagated |
|-------------|-----------------------|--------|------------|
| 0.0.0.0/0   | igw-0f2a2f84e64cc3487 | Active | No         |
| 10.0.0.0/16 | local                 | Active | No         |

23. See the what we given cidr for vpc and cidr for subnet association, **Follow Above the code**

```
resource "aws_vpc" "my_vpc" {
  cidr_block = "10.0.0.0/16"
  enable_dns_support = true
  enable_dns_hostnames = true
}

resource "aws_subnet" "my_subnet" {
  vpc_id = aws_vpc.my_vpc.id
  cidr_block = "10.0.6.0/24"
  availability_zone = "us-east-1a"
}
```

24. Here the subnet association cidr

The screenshot shows the AWS VPC console with a search bar 'VPC : vpc-0789a6109801a3c8c' and a 'Clear filters' button. A table lists route tables. One row is selected, highlighted with a yellow box around its Route table ID 'rtb-0d1fec63bb68b48fb'. This row also has a yellow box around its 'Explicit subnet associations' column, which contains the value 'subnet-00d4f1ab6b3fad25f'. Below this, another row is listed with the Route table ID 'rtb-0a669fcf83aa9ac16'. A modal window titled 'Explicit subnet associations (1)' is open, showing a table with one entry. This entry has a yellow box around its Subnet ID 'subnet-00d4f1ab6b3fad25f'. The table also includes columns for Name, IPv4 CIDR (10.0.6.0/24), and IPv6 CIDR (empty).

25. we can see the given port and cidr source was everyone

The screenshot shows the AWS EC2 instance details for 'MyEC2Instance' (i-032a365516cec3eb7). It displays basic information like state (Running), instance type (t2.micro), and availability zone (us-east-1). A table lists security group rules. Two rules are highlighted with a yellow box: the first rule allows TCP port 80 from 0.0.0.0/0, and the second rule allows TCP port 22 from 0.0.0.0/0.

26. user data

The screenshot shows the 'Edit user data' page for the instance 'i-032a365516cec3eb7'. The 'Instance ID' section shows the instance ID 'i-032a365516cec3eb7 (MyEC2Instance)'. The 'Current user data' section contains the following script:

```
#!/bin/bash
sudo apt update -y
sudo apt install nginx -y
sudo systemctl enable nginx -y
```

A yellow box highlights this entire code block. Below it is a 'Copy user data' button. A note at the bottom states: 'To edit your instance's user data you first need to stop your instance.' At the bottom right are 'Cancel' and 'Save' buttons.

## 27. Now Destroying the infrastructure

```
ubuntu@Terraform-Demo:~/terraform$ terraform destroy
aws_vpc.my_vpc: Refreshing state... [id=vpc-0789a6109801a3c8c]
aws_security_group.my_security_group: Refreshing state... [id=sg-01067e31fbe6ab135]
aws_internet_gateway.my_igw: Refreshing state... [id=igw-0f2a2f84e64cc3487]
aws_subnet.my_subnet: Refreshing state... [id=subnet-00d4f1ab6b3fad25f]
aws_route_table.my_route_table: Refreshing state... [id=rtb-0d1fec63bb68b48fb]
aws_route_table_association.my_subnet_association: Refreshing state... [id=rtbassoc-0c8f6041d1a61fb15]
aws_instance.my_instance: Refreshing state... [id=i-032a365516cec3eb7]
```

## 28. All Seven going to be destroying

```
Plan: 0 to add, 0 to change, 7 to destroy.

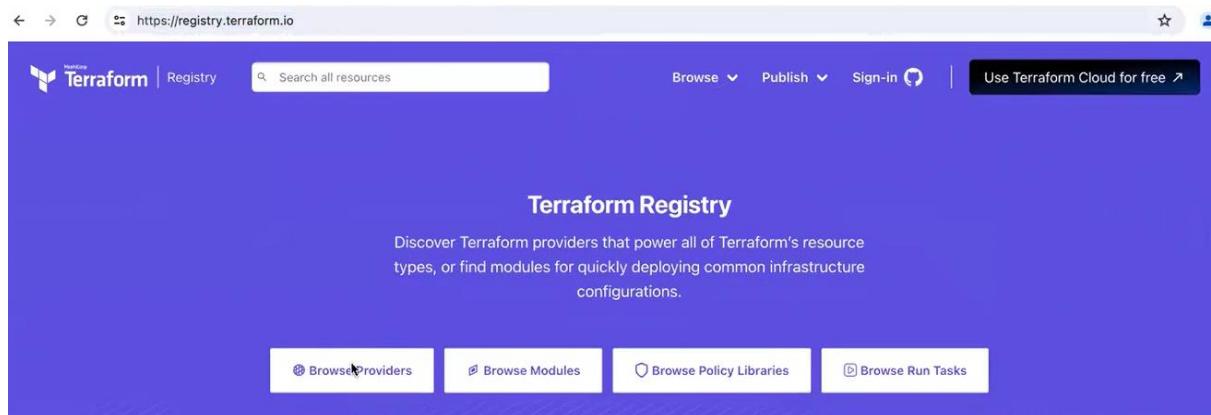
Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_route_table_association.my_subnet_association: Destroying... [id=rtbassoc-0c8f6041d1a61fb15]
aws_instance.my_instance: Destroying... [id=i-032a365516cec3eb7]
aws_route_table_association.my_subnet_association: Destruction complete after 1s
aws_route_table.my_route_table: Destroying... [id=rtb-0d1fec63bb68b48fb]
aws_route_table.my_route_table: Destruction complete after 0s
aws_internet_gateway.my_igw: Destroying... [id=igw-0f2a2f84e64cc3487]
aws_internet_gateway.my_igw: Destruction complete after 0s
aws_instance.my_instance: Still destroying... [id=i-032a365516cec3eb7, 10s elapsed]
aws_instance.my_instance: Still destroying... [id=i-032a365516cec3eb7, 20s elapsed]
aws_instance.my_instance: Still destroying... [id=i-032a365516cec3eb7, 30s elapsed]
aws_instance.my_instance: Still destroying... [id=i-032a365516cec3eb7, 40s elapsed]
aws_instance.my_instance: Still destroying... [id=i-032a365516cec3eb7, 50s elapsed]
aws_instance.my_instance: Destruction complete after 51s
aws_security_group.my_security_group: Destroying... [id=sg-01067e31fbe6ab135]
aws_subnet.my_subnet: Destroying... [id=subnet-00d4f1ab6b3fad25f]
aws_subnet.my_subnet: Destruction complete after 0s
aws_security_group.my_security_group: Destruction complete after 0s
aws_vpc.my_vpc: Destroying... [id=vpc-0789a6109801a3c8c]
aws_vpc.my_vpc: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
ubuntu@Terraform-Demo:~/terraform$
```

Follow the below the links we can go to explore it



https://registry.terraform.io/browse/providers

Partner  
 Community

**Category**

- HashiCorp Platform
- Infrastructure Management
- Public Cloud
- Asset Management
- Cloud Automation
- Communication & Messaging
- Container Orchestration
- Continuous Integration/Deployment (CI/CD)
- Data Management
- Database
- Infrastructure (IaaS)
- Logging & Monitoring
- Networking
- Platform (PaaS)
- Security & Authentication

### AWS

### Azure

### Google Cloud Platform

### Kubernetes

### Alibaba Cloud

### Oracle Cloud Infrastructure

Active Directory  
by: hashicorp

Archive  
by: hashicorp

AWS Cloud Control  
by: hashicorp

We use cookies and other similar technology to collect data to improve your experience on our site, as described in our Privacy Policy and Cookie Policy.

[Manage Preferences](#) [Dismiss](#)

Active Directory  
by: hashicorp

Archive  
by: hashicorp

AWS Cloud Control  
by: hashicorp

Azure Active Directory  
by: hashicorp

Azure Stack  
by: hashicorp

Boundary  
by: hashicorp

Cloudinit  
by: hashicorp

Consul  
by: hashicorp

DNS  
by: hashicorp

External  
by: hashicorp

Google Beta  
by: hashicorp

Google Workspace  
by: hashicorp

HashiCorp Cloud Platform  
by: hashicorp

HashiCorp Consul Service  
by: hashicorp

Helm  
by: hashicorp

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

Terraform Registry

Providers / hashicorp / aws / Version 5.52.0 / Latest Version

**aws**

**AWS DOCUMENTATION**

26 matching results

- Elemental MediaLive
- Resources
  - aws\_mediavive\_input\_security\_group
- STS (Security Token)
- Data Sources
  - aws\_caller\_identity
- Security Hub
- Resources
  - aws\_securityhub\_account

**AWS Provider**

Use the Amazon Web Services (AWS) provider to interact with the many resources supported by AWS. You must configure the provider with the proper credentials before you can use it.

Use the navigation to the left to read about the available resources. There are currently 1373 resources and 559 data sources available in the provider.

To learn the basics of Terraform using this provider, follow the hands-on [get started tutorials](#). Interact with AWS services, including Lambda, RDS, and IAM by following the AWS services [tutorials](#).

**Example Usage**

**ON THIS PAGE**

[Multi-language provider docs](#)  
[Terraform](#)

The Registry now supports multi-language docs powered by CDK for Terraform. [Learn more](#)

[Example Usage](#)  
[Authentication and Configuration](#)  
[AWS Configuration Reference](#)  
[Custom User-Agent Information](#)  
[Argument Reference](#)  
[Getting the Account ID](#)

[Report an issue](#)

Links:

<https://registry.terraform.io/>

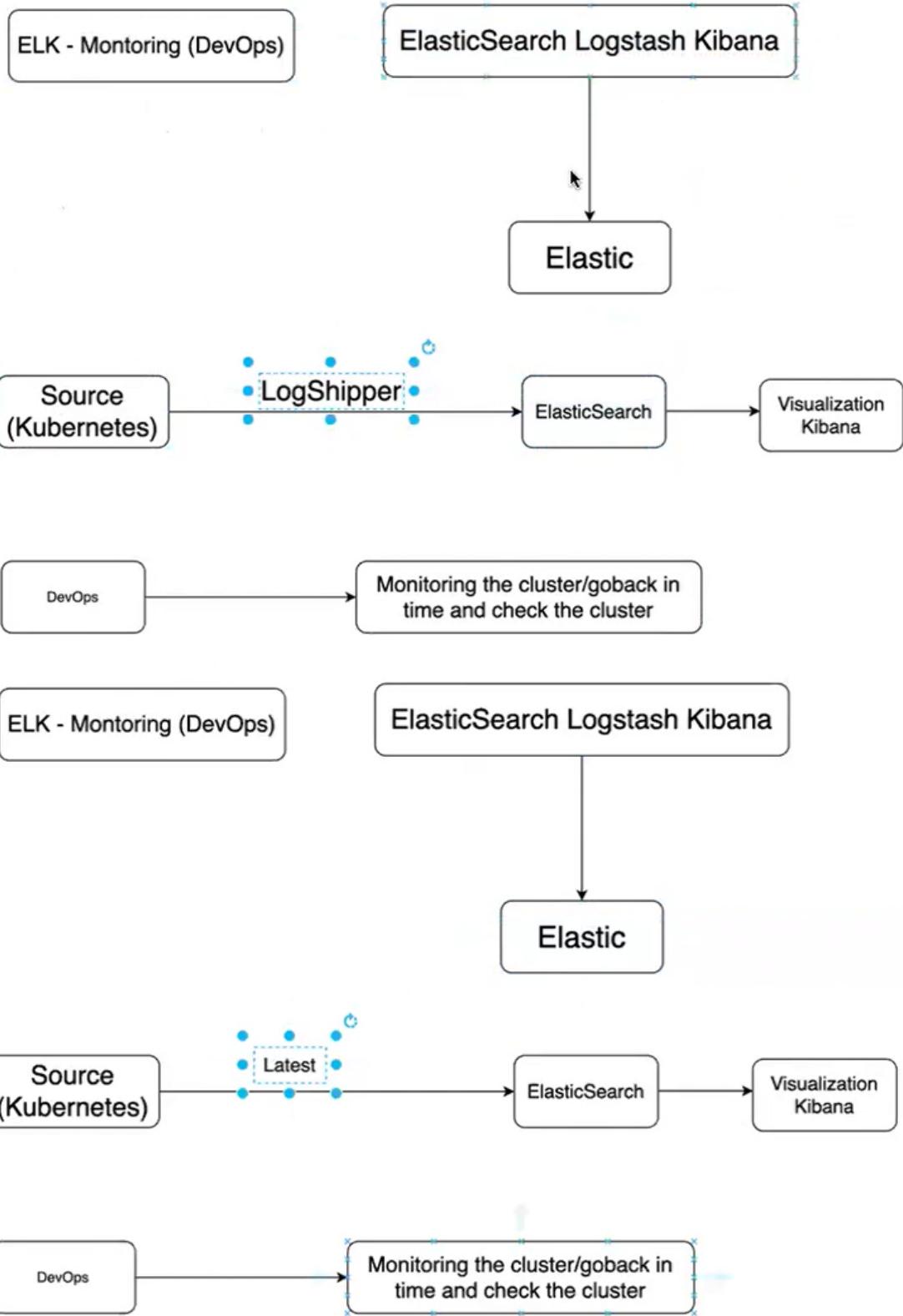
<https://registry.terraform.io/browse/providers>

Aws eg:

[https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/service\\_discovery\\_instance](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/service_discovery_instance)

**Go and Explore it.**

## ELK(ElasticSearch Logstash Kibana)



ELK in DevOps is like having a search engine for your system's logs. "ELK" stands for Elasticsearch, Logstash, and Kibana.

- **Elasticsearch** is like the big database where all your logs are stored.
- **Logstash** helps you collect, parse, and send your logs to Elasticsearch.
- **Kibana** is a visualization tool that lets you search, analyze, and visualize your logs in a user-friendly way.

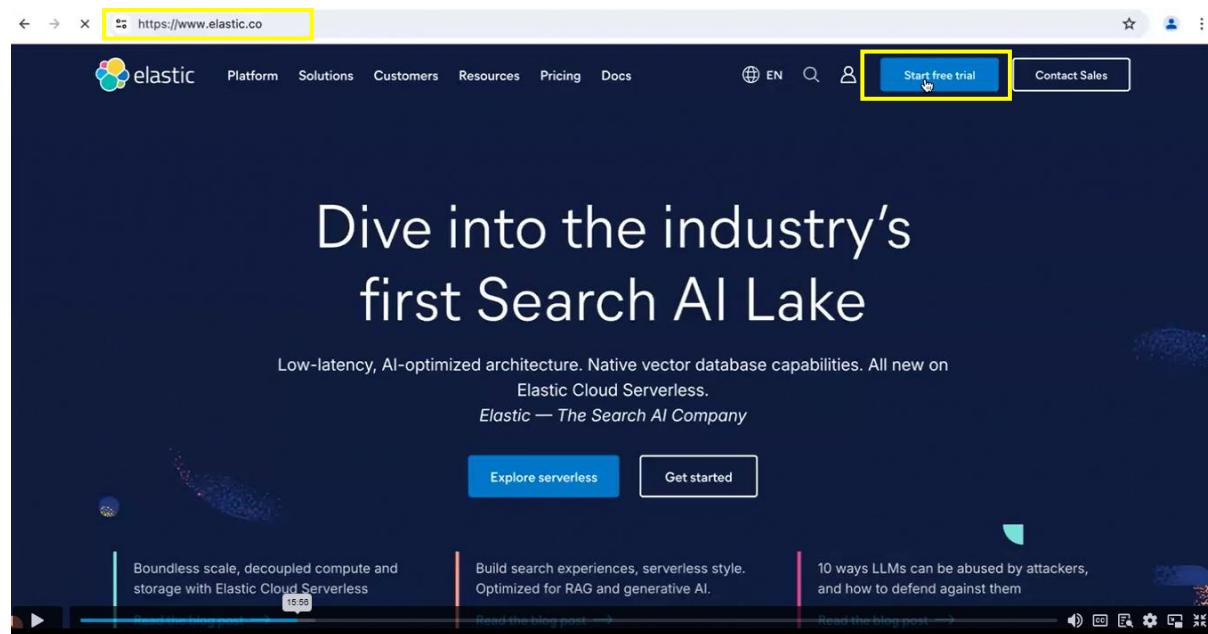
Together, they help DevOps teams manage and troubleshoot their systems by providing a centralized place to store, search, and analyze logs, helping them detect issues, monitor performance, and improve their systems' reliability.

Links:

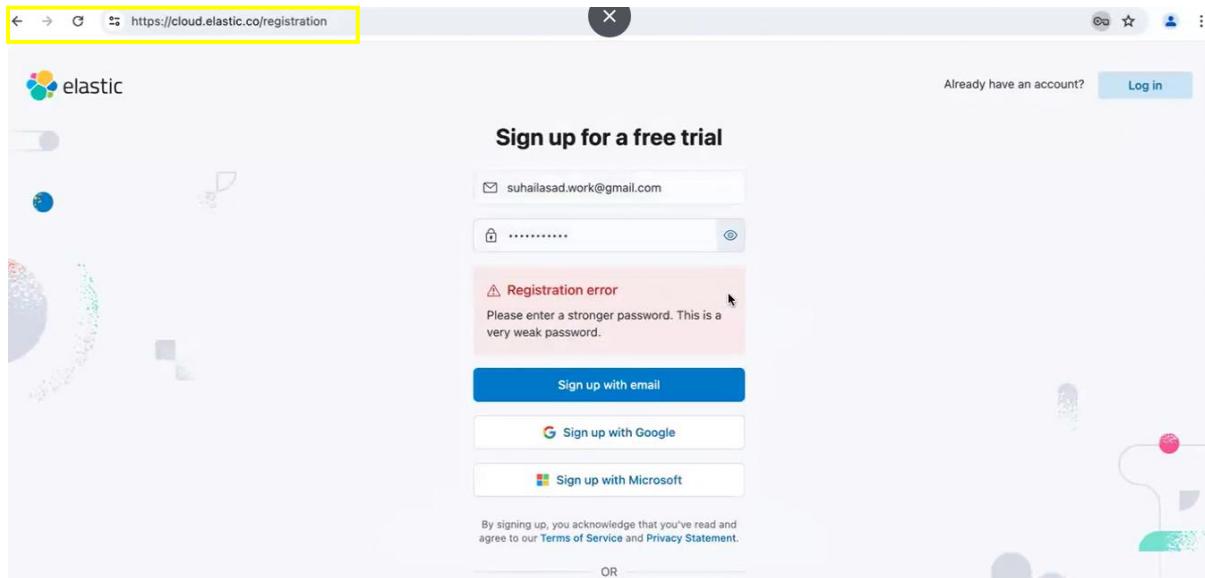
<https://www.elastic.co/>

is indeed the official website of Elastic, the company behind Elasticsearch, as well as the other components of the ELK stack (Logstash and Kibana). On this website, you can find information about their products, solutions, documentation, and resources related to search, logging, and analytics.

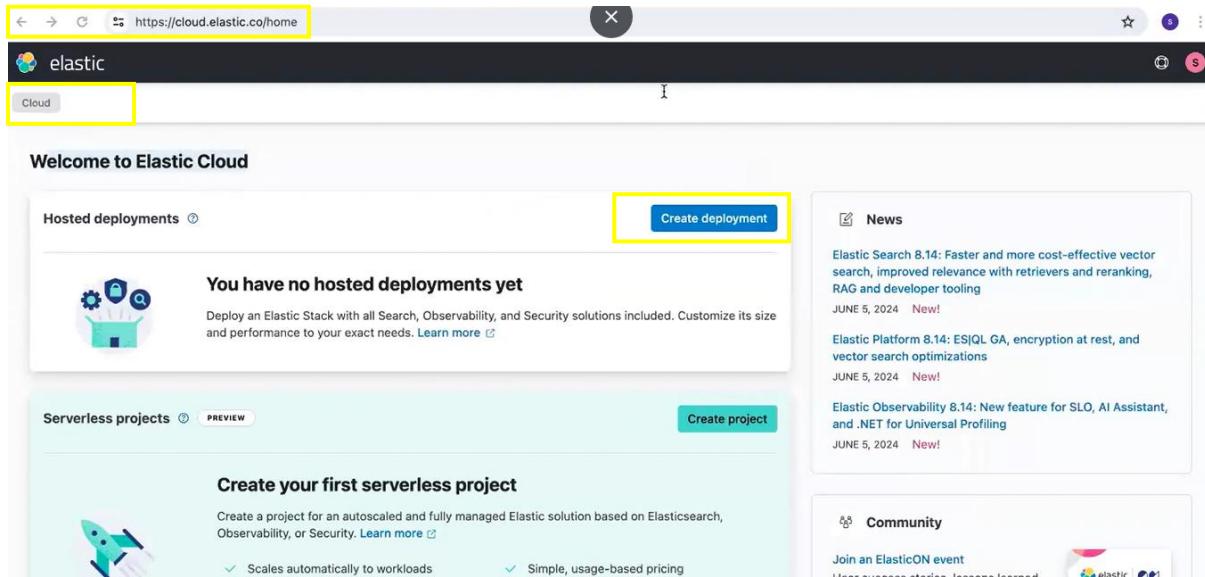
Click start free trial, if u want to practice.



After that you will trigger here <https://cloud.elastic.co/registration>



## After Login home page in elastic search



# Create a deployment

The screenshot shows the Elastic Cloud interface with a dark header bar containing the 'elastic' logo. Below the header, a breadcrumb navigation bar shows 'Cloud > Deployments > Create'. The main content area has a title 'Create a deployment' and a subtitle explaining that a deployment includes Elasticsearch, Kibana, and other Elastic Stack features. A 'Name' field is present with the value 'My deployment' highlighted by a yellow box. Below the name field is a large, blurred section representing 'Advanced settings'. At the bottom of the form is a 'Create deployment' button with the text '22.27' next to it.

U can give any name and u can create it

## Create a deployment

A deployment includes Elasticsearch, Kibana, and other Elastic Stack features, allowing you to store, search, and analyze your data.

Name

My deployment

## Settings

Restore snapshot data ⓘ

Cloud provider

Google Cloud



Region

Iowa (us-central1)



Hardware profile ⓘ

Storage optimized



Hourly

Monthly

\$0.4793

For 1 zone or 2 zone select it, click on create deployment

The screenshot shows the configuration for creating a new deployment. At the top, there's a dropdown for 'Size per zone' set to '1 GB RAM | Up to 8 vCPU'. Below it, under 'Availability zones', the '1 zone' option is selected and highlighted with a yellow box. The total configuration is summarized as 'Total (size x zone) 1 GB RAM | Up to 8 vCPU'. This section is repeated below for a second deployment instance.

**Enterprise Search**

**Enterprise Search instances** [info](#) [X](#)

Add modern search to your application or connect and unify content across your workplace.

Size per zone 2 GB RAM | Up to 8 vCPU

Availability zones  1 zone  2 zones  3 zones

Total (size x zone) 2 GB RAM | Up to 8 vCPU

Hourly Monthly \$0.2192 [▼](#)

[Create deployment](#) Equivalent API request

This is Username and password save it.

A progress bar indicates the deployment is currently 'Creating your deployment (takes about five minutes)'. A 'Continue' button is available to the right.

**Save the deployment credentials**

These root credentials are shown only once.  
They provide super user access to your deployment. Keep them safe.

Username  
**elastic**

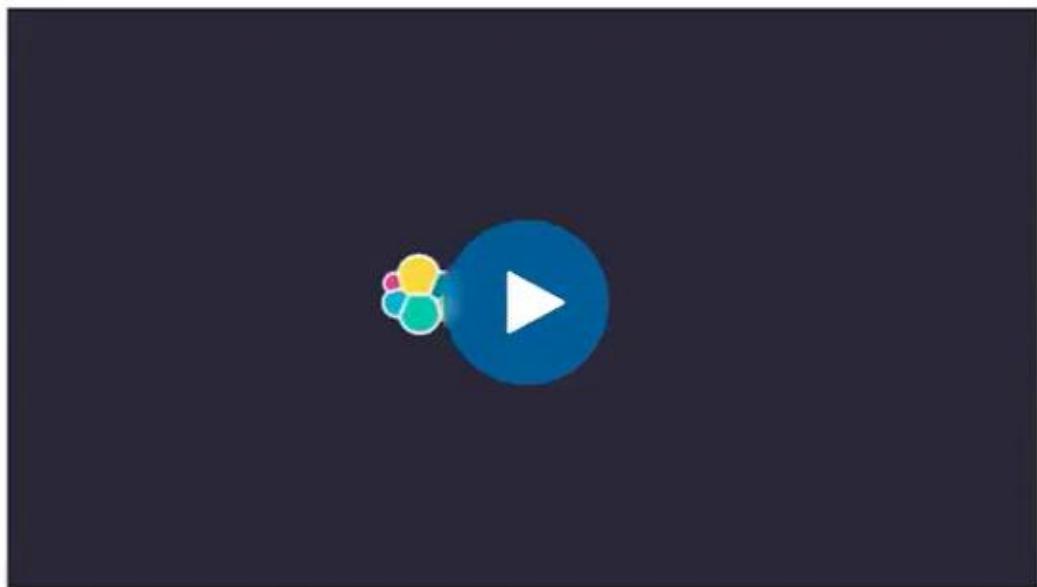
Password  
**PYu68zVwZygyLzjap4TZ4eaR**

[Download](#)

**Continue**



## Discover what you can do with Elastic



This is Dashboard of Elastic search

The screenshot shows the Elastic search dashboard with a navigation bar at the top. The 'Home' link is highlighted with a yellow box. Below the navigation, a main heading reads 'What would you like to do first?'. There are six cards arranged in two rows:

- Collect and analyze my logs** (with a log icon)
- Monitor my application performance (APM / tracing)** (with a chart icon)
- Monitor my host metrics** (with a metric icon)

- Monitor Kubernetes clusters** (with a cluster icon)
- Create a Synthetic Monitor** (with a monitor icon)
- Optimize my workloads with Universal Profiling** (with a profile icon)

This is Home Page and click on 3 bar

The screenshot shows the Elastic Home Page. At the top left is the Elastic logo. A search bar is at the top center with placeholder text "Find apps, content, and more.". On the right side of the header are "Setup guides" and other user icons. Below the header is a navigation bar with three bars icon, a "Home" button, and a yellow-highlighted "Discover" button. The main content area has a title "Welcome home". Below it are four cards:

- Search**: Create search experiences with a refined set of APIs and tools.
- Observability**: Consolidate your logs, metrics, application traces, and system availability with purpose-built UIs.
- Security**: Prevent, collect, detect, and respond to threats for unified protection across your infrastructure.
- Analytics**: Explore, visualize, and analyze your data using a powerful suite of analytical tools and applications.

Below these cards is a section titled "Get started by adding integrations" with a brief description and a "Get started" button. To the right of this section is a decorative graphic of various charts and data visualizations.

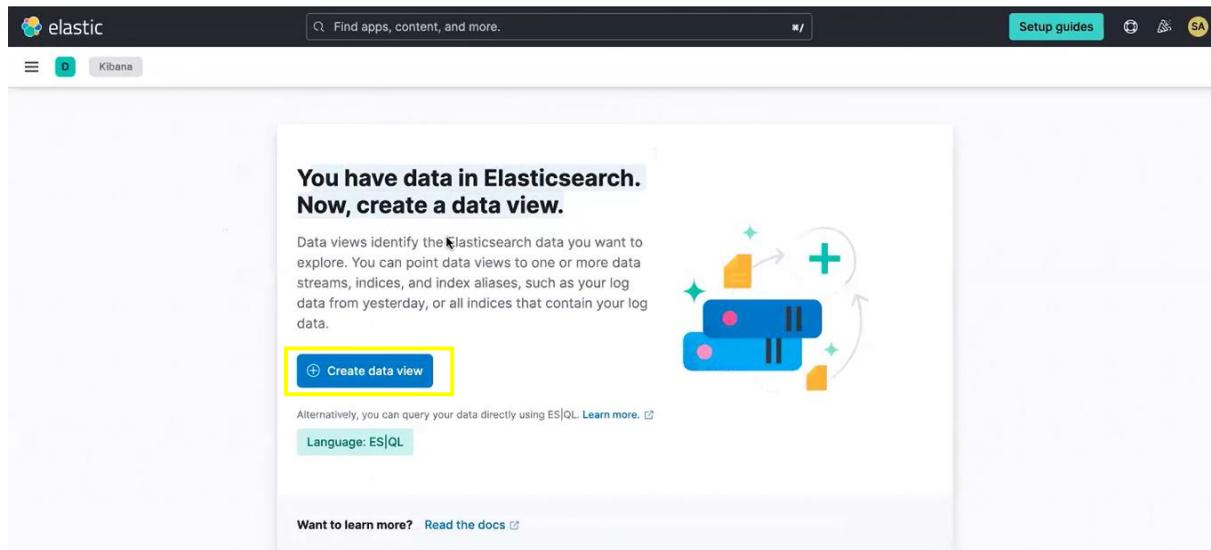
Click on discover

The screenshot shows the Elastic Home Page with the sidebar open. The sidebar includes a "Manage this deployment" section, a "Home" link, and a navigation menu with the following items:

- Analytics** (highlighted with a yellow box):
  - Discover** (highlighted with a yellow box)
  - Dashboards
  - Canvas
  - Maps
  - Machine Learning
  - Graph
  - Visualize Library
- Search**:
  - Overview
  - Content
  - Elasticsearch
- Add integrations**

The main content area is identical to the one in the first screenshot, featuring the "Welcome home" title and the four main service cards: Search, Observability, Security, and Analytics.

We have data in elastic search and click to create a data view



You have data in Elasticsearch.  
Now, create a data view.

Data views identify the Elasticsearch data you want to explore. You can point data views to one or more data streams, indices, and index aliases, such as your log data from yesterday, or all indices that contain your log data.

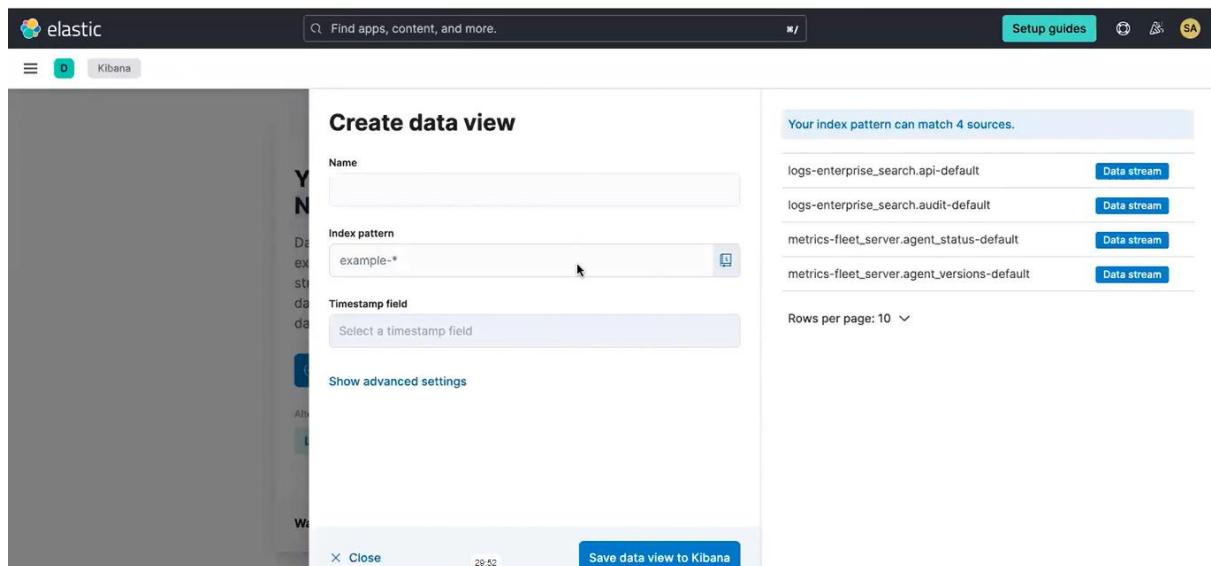
[Create data view](#)

Alternatively, you can query your data directly using ES|QL. [Learn more.](#)

Language: ES|QL

Want to learn more? [Read the docs](#)

Remember this page and go back u need credentials to connect



Create data view

Name

Index pattern

example-\*

Timestamp field

Select a timestamp field

Show advanced settings

Your index pattern can match 4 sources.

|   |             |
|---|-------------|
| logs-enterprise_search.api-default          | Data stream |
| logs-enterprise_search.audit-default        | Data stream |
| metrics-fleet_server.agent_status-default   | Data stream |
| metrics-fleet_server.agent_versions-default | Data stream |

Rows per page: 10

[Close](#) [Save data view to Kibana](#)

The screenshot shows the Elastic Stack interface with the Kibana tab selected. A central modal window titled "You have data in Elasticsearch. Now, create a data view." is displayed. It contains instructions about creating data views, a "Create data view" button, and an alternative query option using ES|QL. The background sidebar shows various management and monitoring options like Home, Cases, Timelines, Intelligence, Explore, Manage, Dev Tools, Integrations, Fleet, Oquery, Stack Monitoring, and Stack Management.

The link <https://www.elastic.co/beats/filebeat> directs you to a specific page on the official Elastic website dedicated to Filebeat, which is a **lightweight shipper for forwarding and centralizing log data**. Elastic provides various "Beats" modules, each tailored for different purposes, and **Filebeat is specifically designed for collecting and forwarding log files**. On this page, you can find information about Filebeat, its features, use cases, documentation, and how to get started with it.

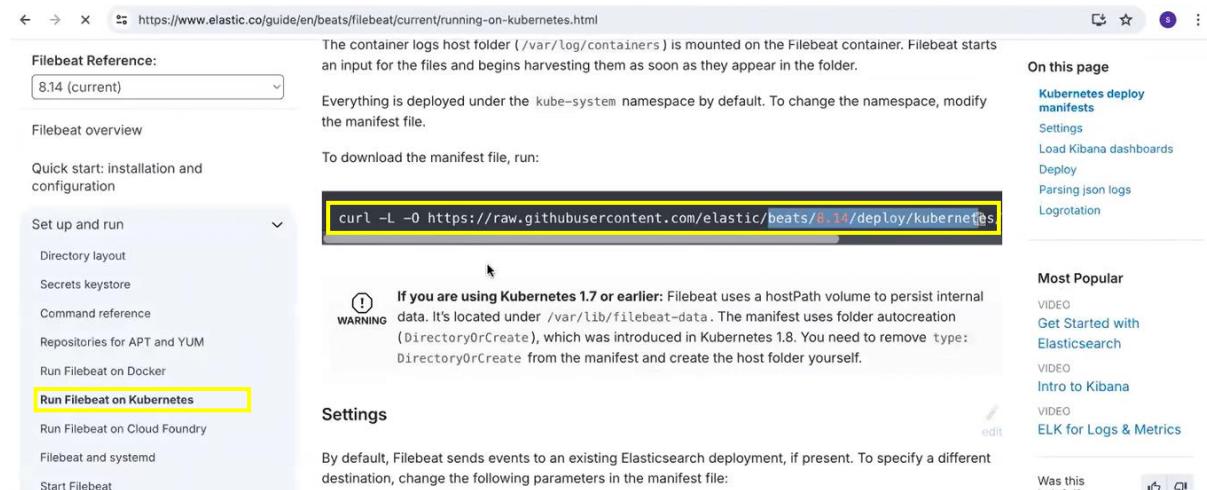
The screenshot shows the official Elastic Filebeat landing page. The URL in the browser bar is https://www.elastic.co/beats/filebeat. The page features a large pink hexagonal graphic on the left and a white background with a teal header. The title "Filebeat" is at the top right. Below the title, the heading "Lightweight shipper for logs" is prominently displayed. A subtext explains that Filebeat helps keep simple things simple by offering a lightweight way to forward and centralize logs and files. Two buttons are present: "Download" and "Filebeat documentation →". To the right, there's a large teal and yellow 3D cube graphic.

The screenshot shows the official Elastic Filebeat Reference guide. The URL in the browser bar is https://www.elastic.co/guide/en/beats/filebeat/current/index.html. The page has a dark header with the title "Filebeat Reference". Below the header, there's a dropdown menu for "Filebeat Reference:" set to "8.14 (current)". A sidebar on the left lists navigation links: "Filebeat overview", "Quick start: installation and configuration", "Set up and run" (which is currently selected), "Upgrade", and "How Filebeat works". A "Most Popular" sidebar on the right lists video links: "Get Started with Elasticsearch", "Intro to Kibana", and "ELK for Logs & Metrics". At the bottom, there's a "Was this helpful?" section with thumbs up and down icons.

<https://www.elastic.co/guide/en/beats/filebeat/current/running-on-kubernetes.html>- visit page

curl -L -O <https://raw.githubusercontent.com/elastic/beats/8.14/deploy/kubernetes/filebeat-kubernetes.yaml>- for download

## Installing filebeats on Kubernetes cluster



The container logs host folder (`/var/log/containers`) is mounted on the Filebeat container. Filebeat starts an input for the files and begins harvesting them as soon as they appear in the folder.

Everything is deployed under the `kube-system` namespace by default. To change the namespace, modify the manifest file.

To download the manifest file, run:

```
curl -L -O https://raw.githubusercontent.com/elastic/beats/8.14/deploy/kubernetes/filebeat-kubernetes.yaml
```

**WARNING** If you are using Kubernetes 1.7 or earlier: Filebeat uses a `hostPath` volume to persist internal data. It's located under `/var/lib/filebeat-data`. The manifest uses `folder` auto-creation (`DirectoryOrCreate`), which was introduced in Kubernetes 1.8. You need to remove type: `DirectoryOrCreate` from the manifest and create the host folder yourself.

**Settings**

By default, Filebeat sends events to an existing Elasticsearch deployment, if present. To specify a different destination, change the following parameters in the manifest file:

## Installing filebeats on Kubernetes and Is we can the the filebeats

```
ubuntu@kubernetesmaster:~$ curl -L -O https://raw.githubusercontent.com/elastic/beats/8.14/deploy/kubernetes/filebeat-kubernetes.yaml
% Total    % Received % Xferd  Average Speed   Time     Time     Time  Current
          Dload  Upload   Total   Spent    Left  Speed
0      0    0      0      0       0       0      0 --:--:-- --:--:-- --:--:--
100  5981  100  5981      0       0  14223      0 --:--:-- --:--:-- --:--:-- 1
4240
ubuntu@kubernetesmaster:~$ ls
clusterip.yml  filebeat-kubernetes.yaml  nodeport.yml  replicaset.yml
deployment.yml  ingressrule.yml        pod.yml
ubuntu@kubernetesmaster:~$
```

Elastic and Elastic Cloud are both services provided by Elastic, but they serve different purposes:

### 1. Elastic (<https://www.elastic.co/>):

This is the main website for Elastic, the company. Here, you can find information about all the products and solutions offered by Elastic, including Elasticsearch, Kibana, Logstash, Beats, APM, and more. This website provides resources for developers, users, and businesses interested in using Elastic's products for search, logging, analytics, and observability.

## 2. Elastic Cloud (<https://cloud.elastic.co/>):

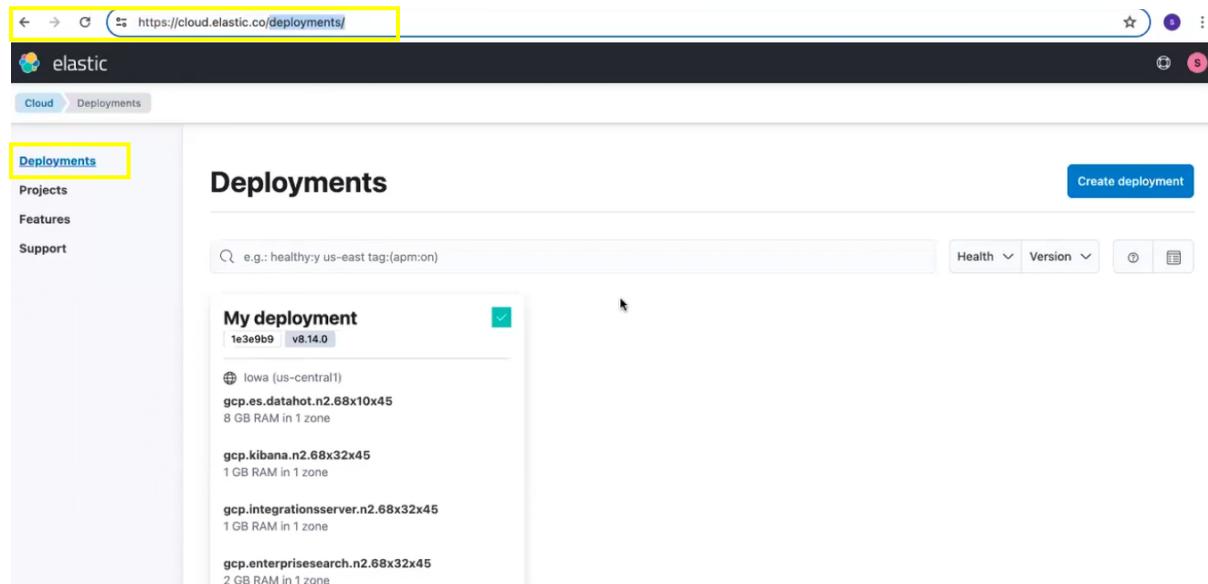
Elastic Cloud is a cloud-based service offered by Elastic. It provides managed Elasticsearch and Kibana deployments, allowing users to easily deploy, manage, and scale Elasticsearch clusters without the need to set up or maintain infrastructure. The Elastic Cloud website is specifically focused on the Elastic Cloud service, providing information and tools for managing deployments in the cloud.

While both websites are operated by Elastic, they cater to different audiences and offer distinct sets of services and information.

## Now go to cloud elastic website and click your deployment

The link <https://cloud.elastic.co/deployments>

directs you to the Elastic Cloud website, specifically to the "Deployments" page. Elastic Cloud is a cloud-based service provided by Elastic that offers managed Elasticsearch, Kibana, and other Elastic Stack products. On this page, users can manage their deployments, including creating, configuring, and monitoring Elasticsearch clusters and associated services through a user-friendly interface.



The screenshot shows a web browser window with the URL <https://cloud.elastic.co/deployments> in the address bar. The page has a dark header with the Elastic logo and navigation links for Cloud, Deployments, Projects, Features, and Support. The main content area is titled "Deployments" and features a search bar with placeholder text "e.g.: healthy:y us-east tag:(apm:on)". Below the search bar is a table titled "My deployment" showing four entries:

|                                    | 1e3e9b9            | v8.14.0            |
|------------------------------------|--------------------|--------------------|
| gcp.es.datashot.n2.68x10x45        | Iowa (us-central1) | 8 GB RAM in 1 zone |
| gcp.kibana.n2.68x32x45             |                    | 1 GB RAM in 1 zone |
| gcp.integrationsserver.n2.68x32x45 |                    | 1 GB RAM in 1 zone |
| gcp.enterprisearch.n2.68x32x45     |                    | 2 GB RAM in 1 zone |

On the right side of the table are "Health" and "Version" dropdown menus, and icons for "Create deployment", "Edit", and "Delete".

## Copy the Elastic search endpoint

The screenshot shows the Elasticsearch Cloud interface. On the left, a sidebar lists various deployment-related options like My deployment, Edit, Monitoring, Health, Logs and metrics, Performance, Elasticsearch, Snapshots, API console, Kibana, Integrations Server, Enterprise Search, Activity, Security, Projects, Features, and Support. The main area is titled 'My deployment' and shows a status bar indicating 'HEALTHY, WITH WARNINGS'. Below this, the 'Deployment name' is 'My deployment' and the 'Custom endpoint alias' is 'my-deployment-1e3e9b'. The 'Deployment version' is 'v8.14.0'. The 'Applications' section contains links for Elasticsearch, Kibana, APM, Fleet, and Enterprise Search, each with 'Copy endpoint' and 'Copy cluster ID' buttons. A 'Copied!' message is visible above the Elasticsearch row. To the right, there's a 'Cloud ID' field containing a long hex string: 'My\_deployment:dXNtY2VudHJhbDEuZ2NwLmNsB3VkLnVzLalvOjQ0MyRjMzU4Ny0NhJjZTM0HTAwYWFLODQ3Njg2ZWFjNjFhzSQNmI4MmQ1MnUzZn10MzQ000U2MjhMjQzZDQxNzIwNg=='. At the bottom, there are filters for 'Health', 'Instance configuration', 'Data tier', and a grid icon.

Now again come to Kubernetes server and open editor

```
ubuntu@kubernetesmaster:~$ vi filebeat-kubernetes.yaml
```

In the Env part this is without changing, remember

```
env:  
  - name: ELASTICSEARCH_HOST  
    value: elasticsearch  
  - name: ELASTICSEARCH_PORT  
    value: "9200"  
  - name: ELASTICSEARCH_USERNAME  
    value: elastic  
  - name: ELASTICSEARCH_PASSWORD  
    value: changeme  
  - name: ELASTIC_CLOUD_ID  
    value:  
  - name: ELASTIC_CLOUD_AUTH  
    value:  
  - name: NODE_NAME  
    valueFrom:  
      fieldRef:  
        fieldPath: spec.nodeName
```

Now copy the cloud id on right side

The screenshot shows the Cloud.Elastic deployment interface for a deployment named 'My deployment'. The deployment is healthy with some warnings. On the left, there's a sidebar with various monitoring and management options like 'Edit', 'Monitoring', 'Health', 'Logs and metrics', 'Performance', 'Elasticsearch', 'Snapshots', 'API console', 'Kibana', 'Integrations Server', 'Enterprise Search', 'Activity', and 'Security'. The main area is titled 'My deployment' and shows the deployment details: Deployment name 'My deployment', Custom endpoint alias 'my-deployment-1e3e9b', Deployment version 'v8.14.0', and a 'Cloud ID' field which has been highlighted with a yellow box and contains the copied value: 'My\_deployment:JhbDEuZ2NwLsNb3VkJ...'. There are also sections for 'Applications' (Elasticsearch, Kibana, APM, Fleet, Enterprise Search) and 'Hardware profile' (Storage optimized).

# On the value paste copied endpoint, cloud id, paste deployment credentials on Elastic cloud Auth

```
env:
  - name: ELASTICSEARCH_HOST
    value: https://my-deployment-1e3e9b.es.us-central1.gcp.cloud.es.io
  - name: ELASTICSEARCH_PORT
    value: "9243"
  - name: ELASTICSEARCH_USERNAME
    value: elastic
  - name: ELASTICSEARCH_PASSWORD
    value: changeme
  - name: ELASTIC_CLOUD_ID
    value: "My_deployment:dXMtY2VudHJhbDEuZ2NwLmNs3VkLmVzLmlvOjQ0MyRjMzU4MWY0NWJjZTM0MTAwYWFlODQ3MjQ2ZWfjMjFhZSQ0NmI4MmQ1MmUzZmI0MzQ00WU2MjhMjQzZDQxNzIwNg=="
  - name: ELASTIC_CLOUD_AUTH
    value: "elastic:PYu68zVwZgyyLzjap4TZ4eaR"
  - name: NODE_NAME
    valueFrom:
      fieldRef:
        fieldPath: spec.nodeName
```

## Save it. And Apply it

```
ubuntu@kubernetesmaster:~$ vi filebeat-kubernetes.yaml
ubuntu@kubernetesmaster:~$ kubectl apply -f filebeat-kubernetes.yaml
serviceaccount/filebeat created
clusterrole.rbac.authorization.k8s.io/filebeat created
role.rbac.authorization.k8s.io/filebeat created
role.rbac.authorization.k8s.io/filebeat-kubeadm-config created
clusterrolebinding.rbac.authorization.k8s.io/filebeat created
rolebinding.rbac.authorization.k8s.io/filebeat created
rolebinding.rbac.authorization.k8s.io/filebeat-kubeadm-config created
configmap/filebeat-config created
daemonset.apps/filebeat created
ubuntu@kubernetesmaster:~$ clea
```

## Click on Security

The screenshot shows the 'My deployment' page in the Elastic Cloud UI. On the left, there's a sidebar with various navigation options like Deployments, Monitoring, Elasticsearch, Kibana, Integrations Server, Enterprise Search, Activity, and Security. The 'Security' option is highlighted with a yellow box. The main content area has a title 'My deployment' and a status bar indicating 'HEALTHY, WITH WARNINGS'. It shows deployment details such as 'Deployment name: My deployment', 'Custom endpoint alias: my-deployment-1e3e9b', 'Deployment version: v8.14.0', and 'Cloud ID' which is a long string of characters. The 'Cloud ID' field is also highlighted with a yellow box.

# Here u can reset the password, if u want

The screenshot shows the Elastic Cloud interface. The top navigation bar has items: Cloud, Deployments, My deployment, and Security. The 'My deployment' item is highlighted with a yellow box. The left sidebar lists various deployment-related sections like Monitoring, Elasticsearch, and Kibana. The main content area is titled 'Security'. It contains a 'Settings' section with a 'Reset password' button and a note about generating a new password for the 'elastic' user. Below it is a 'Traffic filters' section with an 'Apply filter' button. A modal dialog box is open in the center, titled 'Reset your password?' with 'Cancel' and 'Reset' buttons.

# Just viewing

This screenshot is similar to the one above, showing the Elastic Cloud interface with the 'Security' page. The 'Reset password' button in the 'Settings' section is highlighted with a yellow box. A modal dialog box is centered over the page, titled 'Reset your password?' with 'Cancel' and 'Reset' buttons. The background content is partially visible, including the 'Traffic filters' section and the 'Elasticsearch keystore' section at the bottom.

Now come to elastic search website click on 3 bar

The screenshot shows the Elastic homepage at <https://my-deployment-1e3e9b.kb.us-central1.gcp.cloud.es.io:9243/app/home#/>. The top navigation bar includes the Elastic logo, a search bar, and a 'Setup guides' button. A yellow box highlights the 'Home' button in the top left of the navigation bar. Below the header, there's a 'Welcome home' message and four main service cards:

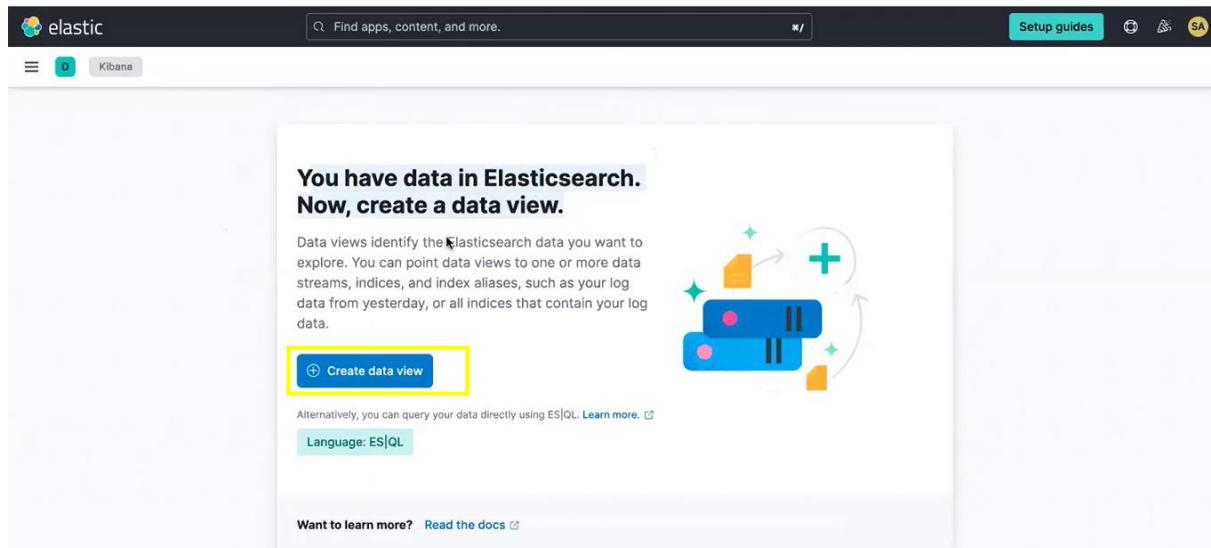
- Search**: Create search experiences with a refined set of APIs and tools.
- Observability**: Consolidate your logs, metrics, application traces, and system availability with purpose-built UIs.
- Security**: Prevent, collect, detect, and respond to threats for unified protection across your infrastructure.
- Analytics**: Explore, visualize, and analyze your data using a powerful suite of analytical tools and applications.

Below the cards, a section titled 'Get started by adding integrations' provides instructions for starting work with data.

## Click Discover or Analytics

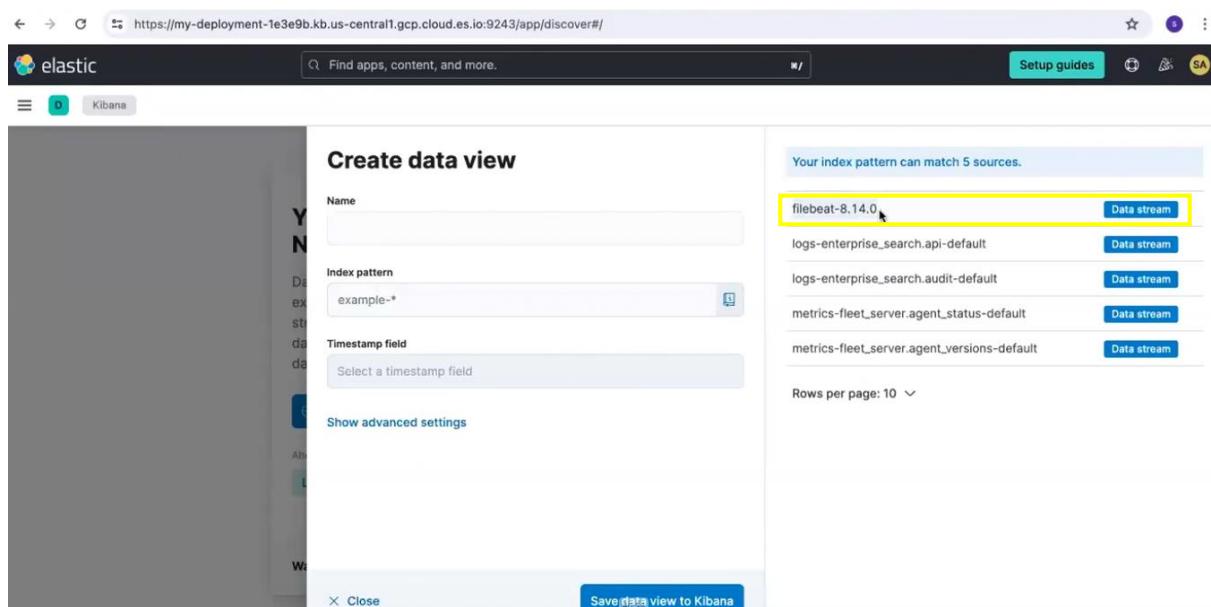
The screenshot shows the same Elastic homepage as above, but with a different view of the sidebar. The 'Discover' link under the 'Analytics' section is highlighted with a yellow box. The rest of the sidebar and the main content area are identical to the first screenshot.

**Now Click create data view**



The screenshot shows the Elasticsearch Kibana interface. At the top, there's a search bar with the placeholder "Find apps, content, and more." and a "Setup guides" button. Below the search bar, there are navigation icons for "Discover", "Dashboard", "Visualize", and "Logs". The main content area has a title "You have data in Elasticsearch. Now, create a data view." with a sub-instruction: "Data views identify the Elasticsearch data you want to explore. You can point data views to one or more data streams, indices, and index aliases, such as your log data from yesterday, or all indices that contain your log data." A large yellow button labeled "Create data view" is prominently displayed. To the right of the text, there's a graphic of three blue rectangular boxes with arrows indicating data flow. Below the main text, there's a note: "Alternatively, you can query your data directly using ES|QL. [Learn more.](#)" and a "Language: ES|QL" button. At the bottom, there's a link "Want to learn more? [Read the docs](#)".

**Now u will get the filebeat option on right side previous not so we gone through cloud elastic and my deployment copying endpoint and configuring on Kubernetes server and applying and now came to elastic search to visualizing logs through kibana it will asking credentials that we have created**



The screenshot shows the "Create data view" dialog box. On the left, there are fields for "Name" (empty), "Index pattern" (set to "example-\*"), and "Timestamp field" (empty). Below these is a "Show advanced settings" link. On the right, a sidebar lists index patterns: "Your index pattern can match 5 sources." followed by a list of five items, each with a "Data stream" button: "filebeat-8.14.0" (highlighted with a yellow box), "logs-enterprise\_search.api-default", "logs-enterprise\_search.audit-default", "metrics-fleet\_server.agent\_status-default", and "metrics-fleet\_server.agent\_versions-default". At the bottom of the sidebar, there's a "Rows per page: 10" dropdown. At the very bottom of the dialog box, there are "Close" and "Save data view to Kibana" buttons.

Type Name as ur wish and click data stream and follow it.

Create data view

Name: KubernetesLogsView

Index pattern: filebeat-\*

Timestamp field: @timestamp

All sources Matching sources

filebeat-8.14.0 Data stream

Rows per page: 10

Now this is our Kubernetes logs dashboard in kibana

KubernetesLogsView

Discover

Filter your data using KQL syntax

Auto interval No breakdown

07:37 07:38 07:39 07:40 07:41 07:42 07:43 07:44 07:45 07:46 07:47 07:48 07:49 07:50 07:51 07:52

Jun 7, 2024 @ 07:37:33.458 - Jun 7, 2024 @ 07:52:33.458 (interval: Auto - 30 seconds)

07:49:30 Count of records 103

Documents (240) Field statistics

Get the best look at your search results

Take the tour Dismiss

@timestamp Document

Jun 7, 2024 @ 07:52:12.497 @timestamp Jun 7, 2024 @ 07:52:12.497 agent.ephemeral\_id b5e148b2-a5b7-4d99-a07c-f7d5ae3780e2 agent.hostname kubernetes worker agent.id 56d17631-7496-432e-8632-9ecdd7a6ae4d agent.name kubernetesworker agent.type filebeat agent.version 8.14.8 cloud.account.id 654088781472 cloud.availability\_zone ap-south-1a cloud.image.id ami-0f58b397bc5cf2e...

Jun 7, 2024 @ 07:52:04.035 @timestamp Jun 7, 2024 @ 07:52:04.035 agent.ephemeral\_id b5e148b2-a5b7-4d99-a07c-f7d5ae3780e2 agent.hostname kubernetes worker agent.id 56d17631-7496-432e-8632-9ecdd7a6ae4d agent.name kubernetesworker agent.type filebeat agent.version 8.

# We can see the logs in table and json format

The screenshot shows the Elasticsearch Discover interface. On the left, there's a sidebar with 'Available fields' (72 items listed) and a histogram for '@timestamp' on June 7, 2024. The main area has a search bar 'Filter your data using KQL syntax'. On the right, a 'Document' view is shown with 16 of 236 results. It includes 'Actions' for 'View single document' and 'View surrounding documents'. Below this, there are two tabs: 'Table' (selected) and 'JSON'. The 'Table' tab displays a table of log entries with columns for 'Actions', 'Field', and 'Value'. The 'JSON' tab shows the raw log documents.

| Actions                                    | Field              | Value                                 |
|--|--------------------|---------------------------------------|
| <a href="#">View single document</a>       | _id                | 4Ep_8I8BGJkePnhHx7RG                  |
| <a href="#">View surrounding documents</a> | _index             | .ds-filebeat-8.14.0-2024.06.07-000001 |
|  | _score             | -                                     |
|  | @timestamp         | Jun 7, 2024 @ 07:58:52.393            |
|  | agent.ephemeral_id | b5e148b2-a5b7-4d99-a87c-f7d5ae3780e2  |
|  | agent.hostname     | kubernetesworker                      |
|  | agent.id           | 5d617631-7496-432e-8632-9ecdd7a6aed4  |
|  | agent.name         | kubernetesworker                      |
|  | agent.type         | filebeat                              |
|  | agent.version      | 8.14.0                                |

This is table format

The screenshot shows the Elasticsearch Discover interface. The layout is identical to the previous one, with a sidebar for 'Available fields', a histogram for '@timestamp' on June 7, 2024, and a search bar. The 'Document' view on the right shows 16 of 236 results with 'Actions' for 'View single document' and 'View surrounding documents'. The 'Table' tab is selected, displaying a table of log entries with columns for 'Actions', 'Field', and 'Value'. The 'JSON' tab shows the raw log documents.

| Actions                                    | Field                | Value   |
|--|----------------------|---|
| <a href="#">View single document</a>       | kubernetes.node.uid  | 44a82840-51b2-447b-a774-9c3c385015f7  |
| <a href="#">View surrounding documents</a> | kubernetes.pod.ip    | 172.31.41.70  |
|  | kubernetes.pod.name  | filebeat-k89zj  |
|  | kubernetes.pod.uid   | 2f14b625-1978-428c-a479-19663737128f  |
|  | log.file.device_id   | 51713   |
|  | log.file.fingerprint | 76bd2d1b06ddec2d9a3cea4da349c1ebcfef2be3db07b8d759de0511cc9aab677   |
|  | log.file.inode       | 274559  |
|  | log.file.path        | /var/log/containers/filebeat-k89zj_kube-system_filebeat-465cbf99cd4543222b5df0e0b96e0d7e1325c78b081334b7614146bcf6abc2a80.log   |
|  | # log.offset         | 46,598  |
|  | message              | {"log.level": "info", "@timestamp": "2024-06-07T02:28:52.392Z", "log.logger": "monitoring", "log.origin": {"function": "github.com/elasticsearch/beats/filebeat/test/testpb.go:146", "file": "testpb.go", "line": 146, "module": "github.com/elasticsearch/beats/filebeat/test"}, "log.type": "filebeat"} |

# And we can filter according to our time frame

The screenshot shows the Elasticsearch Discover interface. On the left, there's a sidebar with 'Available fields' listed, including @timestamp, agent.ephemeral\_id, agent.hostname, agent.id, agent.name, agent.type, agent.version, cloud.account\_id, cloud.availability\_zone, cloud.image.id, cloud.instance.id, cloud.machine.type, cloud.provider, cloud.region, and cloud.service.name. The main area has a histogram at the top with 'Auto interval' and 'No breakdown' options. Below it, a table shows 'Documents (240)' and 'Field statistics'. The first document entry is: Jun 7, 2024 @ 07:51:12.498 @timestamp Jun 7, 2024 @ 07:51:12.498 agent.ephemeral\_id t... The right side features a 'Quick select' dropdown with a yellow border, showing options like 'Last 15 Minutes' and 'Commonly used' items such as 'Today', 'This week', 'Last 1 minute', 'Last 15 minutes', 'Last 30 minutes', and 'Last 1 hour'. There are also buttons for 'Refresh every 60 Seconds' and 'Apply'.

# And we can search particular logs with keywords of log

The screenshot shows the Elasticsearch Discover interface again. A search bar at the top contains the query 'ku'. The results table shows a single document entry: Jun 7, 2024 @ 08:01:22.498 @timestamp Jun 7, 2024 @ 08:01:22.498 agent.ephemeral\_id b5e148b2-a5b7-4d99-a07c-f7d5ae3780e2 agent.hostname kubernetes worker.agent.id 5d617631-7496-432e-8632-9ecdd7a6aed4 agent.name kubernetesworker.agent.type filebeat.agent.version 8.14.0 cloud.account.id 654088701472 cloud.availability\_zone ap-south-1a cloud.image.id ami-0f58b397bc5c1f2e... The right side of the interface shows a timeline from 21:30 to 22:30, and a detailed view of the selected log entry with fields like ty\_zone, us-cent, 2024 @ 08:01:22.498, 244cc08f-e92a-4, ty\_zone, us-cent, inerhos..., feef9f22-5e71-4, and rali-a ecs.version 1.12.0 elasticsearch.gc.tags safepoint event.category database event.created Jun 7, 2024 @ 08:01:22.498.

# Filtered with particular logs

The screenshot shows the Elasticsearch Discover interface. The search bar contains the query: "kubernetes.pod.name : \"ingress-nginx-controller-66bdb4887d-v6jnr\" or kubernetes.pod.name : \"dashboard-metrics-scraper-795895d745-9cxv2\"". Below the search bar, there are two filter panels: one for "Available fields" and one for "Selected filters". The "Selected filters" panel has a dropdown menu set to "and" and a time range from Jun 7, 2024 @ 00:00:00.000 to Jun 7, 2024 @ 23:59:59.999 (interval: Auto - 30 minutes). The results section shows 163 documents, with the first few entries listed:

- Jun 7, 2024 @ 08:06:52.497 kubernetes.pod.name dashboard-metrics-scraper-795895d745-9cxv2 @timestamp Jun 7, 2024 @ 08:06:52.497 agent.ephemeral\_id b5e148b2-a5b7-4d99-a07c-f7dsae3780e2 agent.hostname kubernetesworker.agent.id 5d617631-7496-432e 8632-9ecdd7a6aed4 agent.name kubernetesworker.agent.type filebeat.agent.version 8.14.0 cloud.account.id 65498878147...
- Jun 7, 2024 @ 08:06:42.498 kubernetes.pod.name dashboard-metrics-scraper-795895d745-9cxv2 @timestamp Jun 7, 2024 @ 08:06:42.498 agent.ephemeral\_id b5e148b2-a5b7-4d99-a07c-f7dsae3780e2 agent.hostname kubernetesworker.agent.id 5d617631-7496-432e 8632-9ecdd7a6aed4 agent.name kubernetesworker.agent.type filebeat.agent.version 8.14.0 cloud.account.id 65498878147...
- Jun 7, 2024 @ 08:06:34.198 kubernetes.pod.name dashboard-metrics-scraper-795895d745-9cxv2 @timestamp Jun 7, 2024 @ 08:06:34.198 agent.ephemeral\_id b5e148b2-a5b7-4d99-a07c-f7dsae3780e2 agent.hostname kubernetesworker.agent.id 5d617631-7496-432e 8632-9ecdd7a6aed4 agent.name kubernetesworker.agent.type filebeat.agent.version 8.14.0 cloud.account.id 65498878147...
- Jun 7, 2024 @ 08:06:33.873 kubernetes.pod.name dashboard-metrics-scraper-795895d745-9cxv2 @timestamp Jun 7, 2024 @ 08:06:33.873

Like explore it. To learn

The screenshot shows the Elasticsearch Discover interface. The search bar contains the query: "kubernetes.pod.name : \"ingress-nginx-controller-66bdb4887d-v6jnr\" or kubernetes.pod.name : \"dashboard-metrics-scraper-795895d745-9cxv2\"". The results section shows 163 documents. On the right side, a modal window titled "Document" is open, showing the details of the 20th document. The modal includes a "Actions" section with "View single document" and "View surrounding documents". The document content is as follows:

log.file.fingerprint 58e3fc09987ba9b5561837557d379aa522021e8fc2  
log.file.inode 96088a240484d34a1a03f2  
log.file.path /var/log/containers/dashboard-metrics-scraper-795895d745-9cxv2.kubernetes-dashboard.  
dashboard-metrics-scraper-862ea337af50e952  
5d0808ad6968825a1e3a38a0e3120d4901fdc31efc  
5e8849.log  
message 48,858  
("level": "error", "msg": "Error scraping node metrics: the server could not find the requested resource (get nodes.metrics.k8s.io)", "time": "2024-06-07T02:35:33Z"}  
orchestrator.cluster.name kubernetes  
stream stderr

**ELK is a powerful combination of three open-source tools: Elasticsearch, Logstash, and Kibana.**

**Elasticsearch**: Acts as a search and analytics engine, storing and indexing data for fast retrieval and analysis.

**Logstash**: Collects, processes, and forwards log data from various sources to Elasticsearch.

**Kibana** : Provides a user-friendly interface for visualizing and analyzing data stored in Elasticsearch.

Together, ELK helps organizations manage and analyze large volumes of log data, enabling them to gain insights into system performance, troubleshoot issues, and monitor trends.

#### **Websites:**

- [Elasticsearch](<https://www.elastic.co/elasticsearch/>):

Official website for Elasticsearch, providing information, documentation, and resources for the search and analytics engine.

- [Logstash](<https://www.elastic.co/logstash/>):

Official website for Logstash, offering details, documentation, and resources for the log collection and processing tool.

- [Kibana](<https://www.elastic.co/kibana/>):

Official website for Kibana, providing information, documentation, and resources for the visualization and analytics platform.

**These websites offer resources for users to learn about and utilize each component of the ELK stack effectively.**