Below is a **simple step-by-step guide** to create an **EC2 instance** using **Terraform**.

Prerequisites:

1. **AWS account**.
2. **AWS CLI** installed and configured (for authentication).
3. **Terraform** installed on your local machine.
4. **IAM user with appropriate permissions** (EC2, VPC, IAM, S3 etc.).

Step 1: Set up AWS credentials (if not done already)

1. Run the following command to configure AWS CLI:
2. aws configure
    - **AWS Access Key ID**: Your IAM access key.
    - **AWS Secret Access Key**: Your IAM secret key.
    - **Default region**: Choose your region (e.g., ap-south-1).
    - **Output format**: json (or any other format you prefer).

Step 2: Create your Terraform configuration file

1. Create a folder for your Terraform configuration. For example:
2. mkdir terraform-ec2
3. cd terraform-ec2

Now make following terraform files:

Of course — here's a short, no-code explanation:

---

1. backend.tf
→ Tells Terraform where to **store the state file** safely (like in S3).

2. provider.tf
→ Defines **which cloud provider** (AWS) and **region** Terraform will use.

3. variables.tf
→ Lists **what inputs** (like instance type, key name) your project needs.

4. terraform.tfvars
→ **Provides values** for the variables you declared.

5. instance.tf
→ **Creates** the EC2 server (virtual machine) based on your inputs.

6. security.tf
→ **Creates security groups** (firewall rules) to allow ports like SSH and HTTP.

7. keypair.tf
→ **Creates or imports a key pair** (your SSH key) so you can connect to EC2.

8. outputs.tf
→ **Prints important details** (like EC2 IP address) after deployment.

9. user_data.sh
→ A shell script that **automatically installs software** (like Nginx) on the EC2 when it starts.

10. my-key and my-key.pub
→ **Your private and public SSH keys** to securely log into the EC2 instance.

Would you also like me to show you a simple folder structure how all these files are usually arranged? It'll be super clear!

Code:

backend.tf

```
terraform {
 backend "s3" {
  bucket = "terraform-ec2-bucket-deepak"
  key    = "ec2/terraform.tfstate"
  region = "ap-south-1"
 }
}
```

instance.tf

```
# Define the EC2 instance
resource "aws_instance" "example" {
  ami                    = "ami-06b6e5225d1db5f46"
  instance_type          = var.instance_type
  associate_public_ip_address = true
  key_name               = aws_key_pair.imported_key.key_name
  security_groups        = [aws_security_group.allow_ssh_http.name]

  user_data = file("user_data.sh") # Provision Apache

  tags = {
    Name = "Terraform-EC2"
  }
}
```

keypair.tf

```
resource "aws_key_pair" "imported_key" {
  key_name   = var.key_name
  public_key = file(var.public_key_path)
}
```

outputs.tf

```
output "instance_public_ip" {
  description = "Public IP of the EC2 instance"
```

```
 value      = aws_instance.example.public_ip
}

output "instance_id" {
 description = "ID of the EC2 instance"
 value      = aws_instance.example.id
}
```

provider.tf

```
provider "aws" {
 region = var.aws_region
}
```

security.tf

```
resource "aws_security_group" "allow_ssh_http" {
 name       = "allow_ssh_http"
 description = "Allow SSH and HTTP inbound traffic"

 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"]
  }

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

terraform.tfvars

```
aws_region     = "ap-south-1"
instance_type  = "t2.micro"
key_name       = "my-key"
public_key_path = "./my-key.pub"
```

variables.tf

```
variable "aws_region" {
  default = "ap-south-1"
}

variable "instance_type" {
  default = "t2.micro"
}
```

```
variable "key_name" {
  default = "my-key"
}


variable "public_key_path" {
  default = "./my-key.pub"
}
```

user_data.sh

```
#!/bin/bash
sudo apt update -y
sudo apt install -y nginx
systemctl start nginx
systemctl enable nginx
echo "<h1>Terraform EC2 with Nginx Webserver (Ubuntu)</h1>" > /var/www/html/index.html
```

Other Files:

```
ssh-key -t rsa -b 4096 -g my-key
```

my-key, my-key.pub

Commands:

```
terraform init
terraform plan
terraform apply
```

Other Resources:

Created bucket on S3 with public access.

Created IAM user with full access to EC2, VPC, IAM, S3. Then created access keys to configure aws cli.

For this purpose, we selected ap-south-1 region

Step 3: Initialize Terraform

Initialize Terraform to download the AWS provider and set up your configuration:

terraform init

This will download the necessary provider plugin for AWS.

Step 4: Preview the plan (Optional but recommended)

You can run the following command to **preview** the changes Terraform will make to your AWS account:

terraform plan

This command will show you what actions Terraform will perform when you apply the configuration.

Step 5: Apply the configuration to create the EC2 instance

Now, **apply** the Terraform configuration to actually create the EC2 instance in AWS:

terraform apply

Terraform will prompt you to confirm the actions:

Are you sure you want to perform these actions?

Type **yes** and hit Enter.

Step 6: Verify the EC2 instance

After applying, Terraform will create the EC2 instance. Once completed, you should see the following output:

The EC2 instance is now created. You can verify this by logging into the **AWS Management Console** and navigating to **EC2** → **Instances**.

Step 7: SSH into the EC2 instance (Optional)

To **SSH** into your EC2 instance (if you selected an Amazon Linux 2 AMI), you'll need:

- The **Public IP** or **Public DNS** of the instance (available in the EC2 dashboard).
- The **private key** (.pem file) you used when creating the instance.

Run the following command (replace with your instance's IP and path to your .pem file):

```
ssh -i /path/to/your-key.pem ec2-user@<your-instance-public-ip>
```

Step 8: Clean up (Optional)

If you are done and no longer need the EC2 instance, you can **destroy** it to avoid AWS charges:

```
terraform destroy
```

Terraform will ask for confirmation:

```
Do you want to destroy all resources? (y/n)
```

Type **yes** and hit Enter to destroy the resources.
