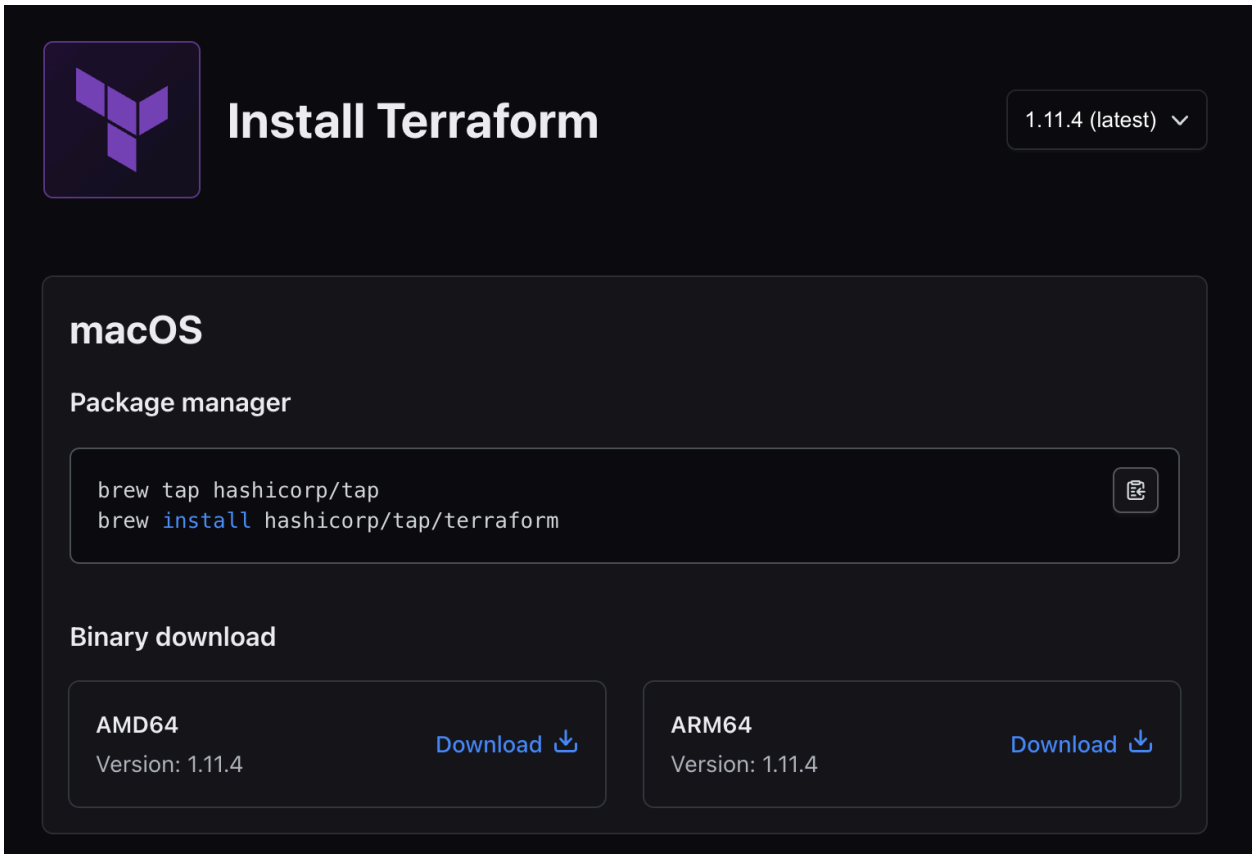
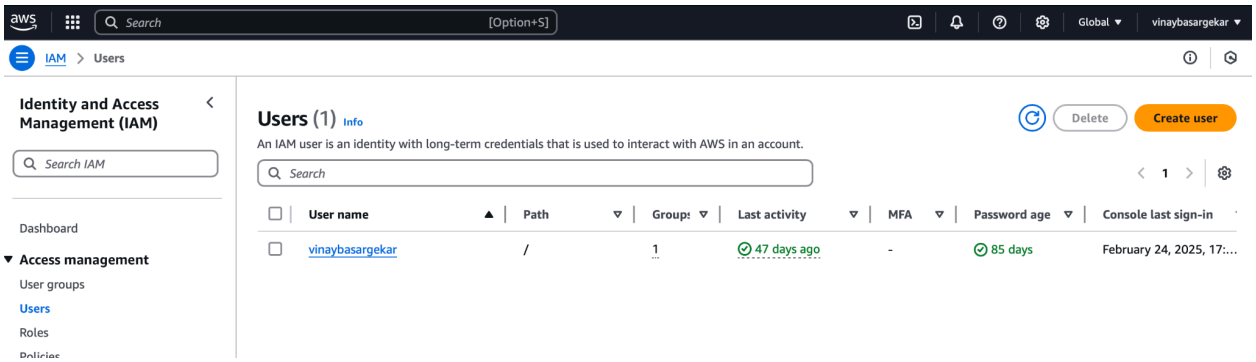
ASSIGNMENT NO.7

1. Install Terraform and verify with `terraform -v`

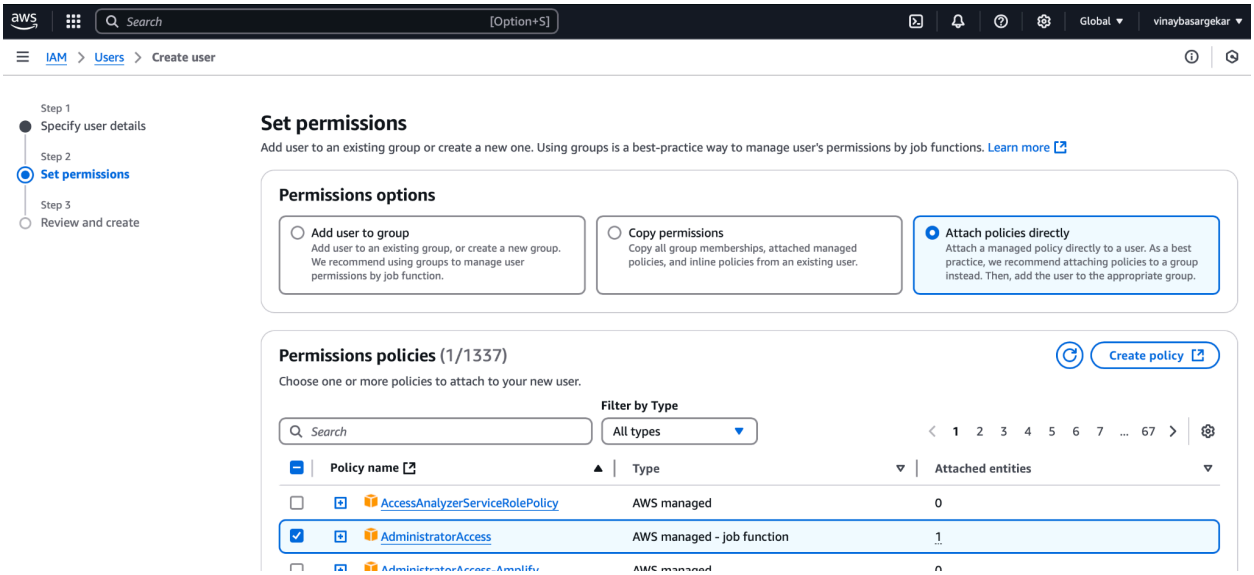


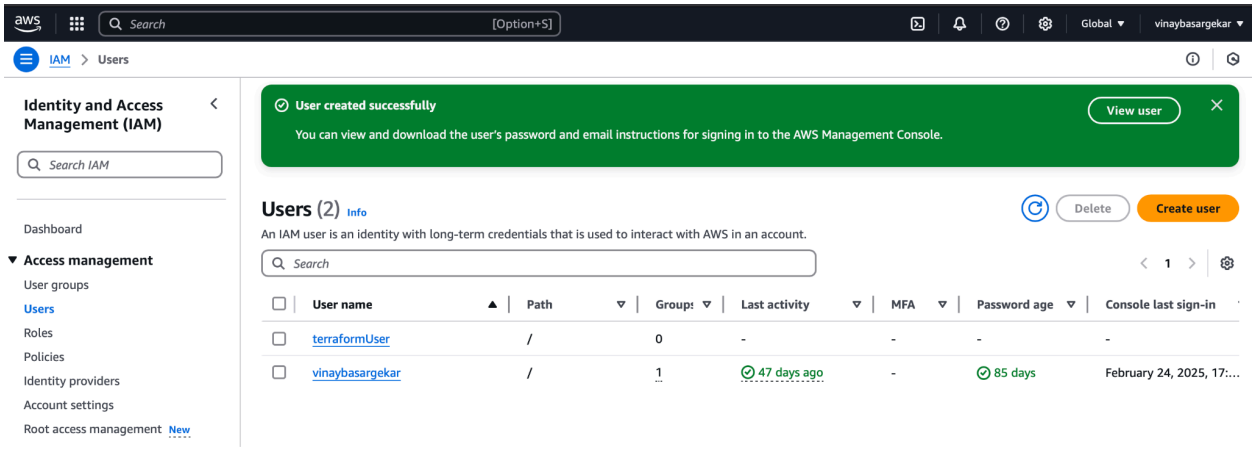
2. Install AWS CLI and configure with `aws configure`
   - *brew install awscli*
   - *aws --version*

3. Go to your AWS console and create a IAM user

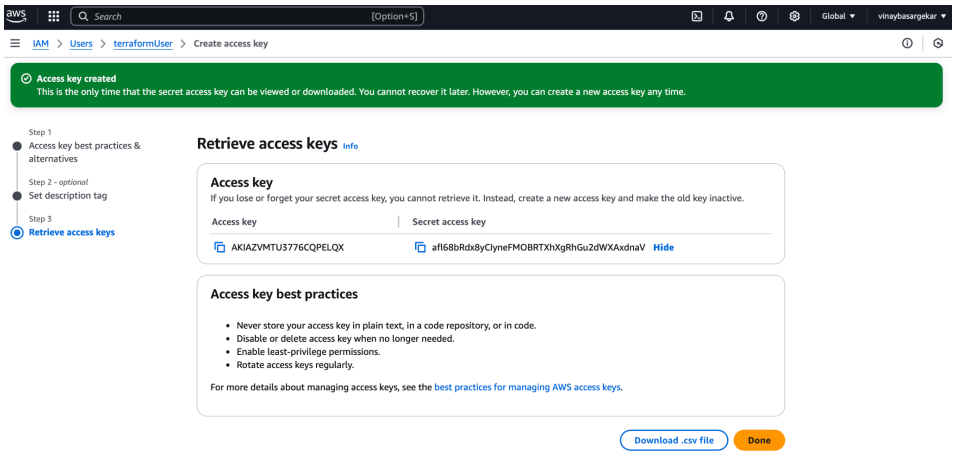


4. Make sure to add permission for administartive access in “attach policies directly”





5. Navigate to Security Credentials and Generate a new **Access Key**



Access key : AKIAZVMTU3776CQPELQX
Secret access key : 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

6. Configure AWS CLI using `aws configure`

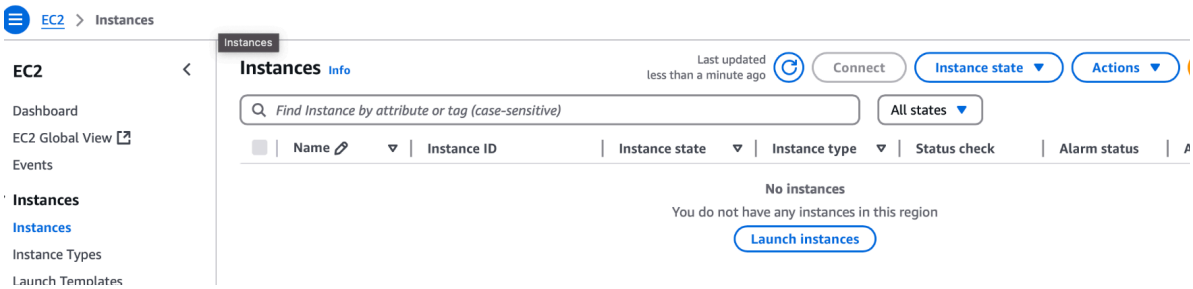
```
vinaybasargekar@Vinays-MacBook-Air ~ % aws configure

AWS Access Key ID [None]: AKIAZVMTU3776CQPELQX
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]: json
```

7. Create the Terraform Working Directory and Files

`main.tf` → The core configuration (provider + EC2 instance)

`variables.tf` → Input variables (dynamic values like AMI ID, instance type, etc.)

`outputs.tf` → Output values (like public IP, instance ID)



main.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.16"
    }
  }
  required_version = ">= 1.2.0"
}

provider "aws" {
  region = var.region
}

resource "aws_instance" "my_ec2" {
  ami           = var.ami_id
  instance_type = var.instance_type

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

variables.tf

```
variable "region" {
  description = "AWS region"
  type        = string
  default     = "us-east-1"
}

variable "ami_id" {
  description = "AMI ID for EC2"
  type        = string
}

variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t2.micro"
}
```

outputs.tf

```
output "instance_id" {
  description = "ID of the EC2 instance"
  value       = aws_instance.my_ec2.id
}

output "public_ip" {
  description = "Public IP address of the EC2 i
  value       = aws_instance.my_ec2.public_ip
}
```

terraform.tfvars

```
region         = "us-east-1"
ami_id         = "ami-00a929b66ed6e0de6"
instance_type  = "t2.micro"

```

8. Initialize Terraform

```
vinaybasargekar@Vinays-MacBook-Air terraform-ec2-setup % terraform init

Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.67.0

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.
```

9. Plan terraform

```
vinaybasargekar@Vinays-MacBook-Air terraform-ec2-setup % terraform plan


Terraform used the selected providers to generate the following execution plan. Resource actions ar
  + create

Terraform will perform the following actions:

  # aws_instance.my_ec2 will be created
  + resource "aws_instance" "my_ec2" {
      + ami                                  = "ami-00a929b66ed6e0de6"
      + 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)
```

10. Apply terraform

```
vinaybasargekar@Vinays-MacBook-Air terraform-ec2-setup % terraform apply

Terraform used the selected providers to generate the following execution plan. Resou
  + create

Terraform will perform the following actions:

  # aws_instance.my_ec2 will be created
  + resource "aws_instance" "my_ec2" {
      + ami                                  = "ami-00a929b66ed6e0de6"
      + 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)
```

```
Changes to Outputs:
  + instance_id = (known after apply)
  + public_ip   = (known after apply)

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

  Enter a value: yes

aws_instance.my_ec2: Creating...
aws_instance.my_ec2: Still creating... [10s elapsed]
aws_instance.my_ec2: Creation complete after 16s [id=i-029f2

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

Outputs:

instance_id = "i-029f2eef2535dd310"
public_ip = "54.242.17.51"
```

11. Check the instance running on AWS console