

Terraform

Install Terraform and AWS CLI	2
Run the first time	3
AWS resource	6
Create infrastructure on AWS	9

1. Install Terraform and AWS CLI

Để thực hành Terraform ta cần cài đặt AWS CLI và Terraform. Ta có thể cài đặt và sử dụng trên nhiều hệ điều hành nhưng trong bài thực hành này ta sẽ cài đặt và sử dụng trên Windows cho thuận tiện cho việc viết configure file trên visual code.

Để cài đặt Terraform trên Windows ta tải phần mềm trên trang chủ terraform.io:

https://releases.hashicorp.com/terraform/1.1.0/terraform_1.1.0_windows_amd64.zip

Để cài đặt AWS CLI ta tải bộ cài đặt MSI tại:

<https://awscli.amazonaws.com/AWSCLIV2.msi>

Sau khi cài đặt xong ta kiểm tra với lệnh sau:

```
C:\Users\ASUS GL552>aws --version
aws-cli/2.2.23 Python/3.8.8 Windows/10 exe/AMD64 prompt/off

C:\Users\ASUS GL552>_
```

Tiếp theo ta thêm access key vào bằng lệnh sau:

```
C:\Users\ASUS GL552>aws --version
aws-cli/2.2.23 Python/3.8.8 Windows/10 exe/AMD64 prompt/off

C:\Users\ASUS GL552>aws configure
AWS Access Key ID [*****3CGE]:
AWS Secret Access Key [*****jhKv]:
Default region name [ap-east-1]:
Default output format [json]: _
```

Làm tương tự như module AWS.

Ta cũng có thể tham khảo cách cài đặt Terraform và AWS CLI tại:

<https://cloudlinuxtech.com/install-terraform-on-ubuntu-uninstall-terraform/>

https://linuxhint.com/install_aws_cli_ubuntu/

2. Run the first time

Terraform hỗ trợ rất nhiều provider khác nhau tuy nhiên trong bài lab này ta tập trung tìm hiểu về provider aws để tạo infrastructure trên aws.

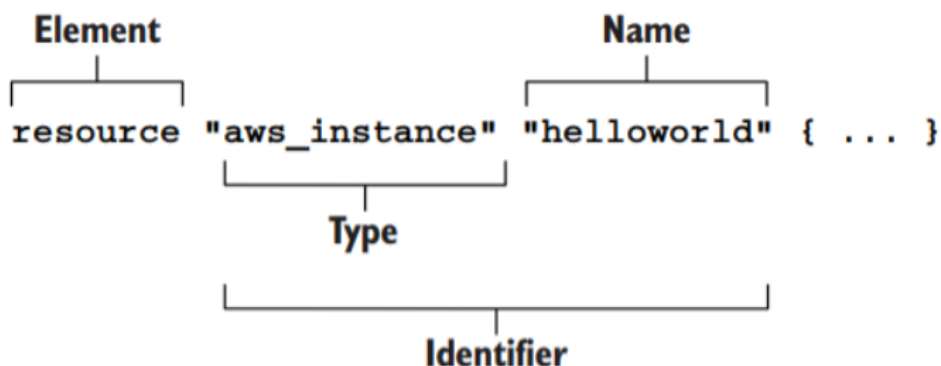
Tạo thư mục và mở bằng visual studio code. Khai báo file đầu tiên main.tf với nội dung:

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

Ở đây chúng ta khai báo provider là aws và vùng HongKong. Sau đó ta khai báo instance bằng đoạn code sau:

```
resource "aws_instance" "training" {  
  ami          = "ami-0b215afe809665ae5"  
  instance_type = "t3.micro"  
  tags = {  
    Name = "training"  
  }  
}
```

Ở trên ta sử dụng một block tên là resources, đây là block quan trọng nhất của terraform, ta sẽ sử dụng block này để tạo resource của chúng ta. Phía sau resources thì ta sẽ có thêm giá trị nữa là resource type mà ta muốn tạo (cái này phụ thuộc vào provider của chúng ta sẽ cung cấp những resource type nào), ở trên resource type của ta là **aws_instance**, và giá trị cuối cùng là tên của resource đó, này ta muốn đặt gì cũng được.




Để xem những thuộc tính của một resource nào đó, ta lên trang <https://registry.terraform.io/> để xem.


Trong đoạn code trên ta cần chú ý ami là loại instance ta muốn cài đặt nó phân biệt theo region nên ta có thể vào **launch instance** để tìm kiếm và **instance_type** ta chọn loại **free tier eligible** như trong module AWS.

nce Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review


Amazon Machine Image (AMI)


Amazon Linux
Free tier eligible


Amazon Linux 2 AMI (HVM) - Kernel 4.14, SSD Volume Type - ami-0a7fb2d401b6017e5 (64-bit x86) / ami-07fe2c4d7640f1b2d (64-bit Arm)
Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is now under maintenance only mode and has been removed from this wizard.
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes


Red Hat
Free tier eligible

Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-0ec87970e52e19867 (64-bit x86) / ami-047f2232912795fcf (64-bit Arm)
Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes


SUSE Linux
Free tier eligible

SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type - ami-0b4017973f2328b15 (64-bit x86) / ami-0748db038a2205f21 (64-bit Arm)
SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.5, PHP 5.4 and Ruby 1.8.7 available.
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes


Ubuntu
Free tier eligible

Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-0a9c1cc3697104990 (64-bit x86) / ami-0b112a0b6eb576c48 (64-bit Arm)
Ubuntu Server 20.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Xong khi ta viết config xong hết, thì ta mở terminal lên và gõ `terraform init`, bước này là bắt buộc khi ta viết một cấu hình cho một hạ tầng mới, nó sẽ tải code của provider xuống folder hiện tại mà ta viết file `main.tf`.

```
PS D:\Terraform\Test> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.69.0...
- Installed hashicorp/aws v3.69.0 (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.
PS D:\Terraform\Test> 
```

Sau khi init xong, ta gõ tiếp câu lệnh `apply` để nó tạo EC2 cho ta.

```
PS D:\Terraform\Test> terraform apply

Terraform used the selected providers to generate the following execution plan. Resources to be created:
+ create

Terraform will perform the following actions:

# aws_instance.training will be created
+ resource "aws_instance" "training" {
  + ami                  = "ami-0b215afe809665ae5"
  + 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_termination = (known after apply)
  + ebs_optimized         = (known after apply)
  + get_password_data      = false
  + host_id               = (known after apply)
  + id                    = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
}
```

Ta đọc cấu hình kiểm tra các thông tin và enter “yes” để terraform tạo instance.

Instances (1) Info								
<input type="text" value="Search"/>								
<input type="text" value="training"/> <input type="button" value="X"/> <input type="button" value="Clear filters"/>								
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	
<input type="checkbox"/>	training	i-0495fe4082d3cf8cf	Running	t3.micro	Initializing	No alarms	ap-east-1b	

Bây giờ nếu ta muốn xóa EC2 đi, ta chỉ cần chạy câu lệnh destroy.

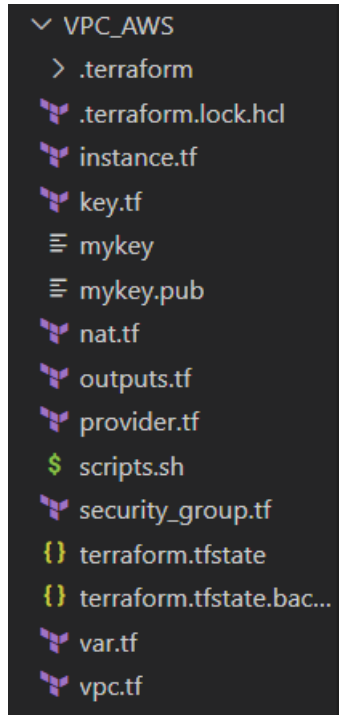
```
PS D:\Terraform\Test> terraform destroy --auto-approve
aws_instance.training: Refreshing state... [id=i-0495fe4082d3cf8cf]

Terraform used the selected providers to generate the following execution plan. Resources to be destroyed:
- destroy

Terraform will perform the following actions:

# aws_instance.training will be destroyed
- resource "aws_instance" "training" {
  - ami                  = "ami-0b215afe809665ae5" -> null
  - arn                  = "arn:aws:ec2:ap-east-1:179623033511:instance/i-0495fe4082d3cf8cf" -> null
  - associate_public_ip_address = true -> null
  - availability_zone     = "ap-east-1b" -> null
  - cpu_core_count        = 1 -> null
  - cpu_threads_per_core  = 2 -> null
  - disable_api_termination = false -> null
}
```

Để dễ dàng quản lý các resource ta có thể phân chia ra thành các file nhỏ như sau:



3. AWS resource

Một số resource AWS cơ bản

- aws_instance

```
resource "aws_instance" "name" {  
    ami                = "ami-"  
    instance_type      = "t3.micro"  
    subnet_id          = aws_subnet.  
    security_groups    = [aws_security_group.]  
    key_name            = aws_key_pair.  
    ebs_block_device {device_name, volume_size}  
    tags               = { Name}  
    provisioner "name" {}  
    connection {host, type, user, private_key}  
}
```

aws_instance.name.id

- variable

variable "name" {type, default}

var.name

- aws_key_pair

```
resource "aws_key_pair" "name" {key_name, public_key}
```

```
aws_key_pair.name.key_name
```

- output

```
output "name" {value}
```

- aws_vpc

```
resource "aws_vpc" "name" {  
  cidr_block      = "192.168.0.0/16"  
  instance_tenancy = "default"  
  enable_dns_support = "true"  
  enable_dns_hostnames = "true"  
  enable_classiclink = "false"  
  tags = {Name}  
}
```

```
aws_vpc.name.id
```

- aws_subnet

```
resource "aws_subnet" "name" {  
  vpc_id      = aws_vpc.name.id  
  cidr_block   = "192.168.1.0/24"  
  map_public_ip_on_launch = "true/false"  
  availability_zone = "ap-east-1a"  
  tags = { Name}  
}
```

```
aws_subnet.name.id
```

- aws_internet_gateway

```
resource "aws_internet_gateway" "name" {  
  vpc_id = aws_vpc.name.id  
  tags = { Name}  
}
```

```
aws_internet_gateway.name.id
```

- aws_route_table

```
resource "aws_route_table" "name" {
```

```
vpc_id = aws_vpc.name.id
route {
  cidr_block = "0.0.0.0/0"
  gateway_id = aws_internet_gateway.name.id
}
tags = { Name}
```

```
aws_route_table.name.id
```

- aws_route_table_association

```
resource "aws_route_table_association" "name" {
  subnet_id      = aws_subnet.name.id
  route_table_id = aws_route_table.name.id
}
```

```
aws_route_table_association.name.id
```

- aws_nat_gateway

```
resource "aws_eip" "nat" {
  vpc = true
}
resource "aws_nat_gateway" "nat_gateway" {
  allocation_id = aws_eip.nat.id
  subnet_id     = aws_subnet.public_subnet.id
  depends_on    = [aws_internet_gateway.name]
}
```

- aws_security_group

```
resource "aws_security_group" "name" {
  vpc_id      = aws_vpc.name.id
  name        = "allow-all"
  egress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```



```
tags = {Name}
}
```

```
aws_security_group.name.id
```

```
- data
```

```
data "type" "name" {}
```

```
data.type.name.
```

```
- module
```

```
module "name" {}
```

```
module.name.
```

```
- aws_ebs_volume
```

```
resource "aws_ebs_volume" "name" {
  availability_zone = "ap-east-1a"
  size             = 20
  type             = "gp2"
  tags = { Name}
}
resource "aws_volume_attachment" "name" {
  device_name = "/dev/xvdh"
  volume_id   = aws_ebs_volume.name.id
  instance_id = aws_instance.name.id
}
```

4. Create infrastructure on AWS

Viết configure file tạo hạ tầng trên AWS bao gồm: VPC, public subnet, private subnet, NAT, Internet Gateway, Route Table, Security Group, Instance.

```
//provider.tf
provider "aws" {
  profile = "default"
  region  = var.AWS_REGION
}
```

```
//vcp.tf
resource "aws_vpc" "VLAN" {
```

```

    cidr_block          = "192.168.0.0/16"
    instance_tenancy    = "default"
    enable_dns_support  = "true"
    enable_dns_hostnames = "true"
    enable_classiclink  = "false"
    tags = {
        Name = "ThuongDD"
        Environment = var.ENV
    }
}

resource "aws_subnet" "public_subnet" {
    vpc_id          = aws_vpc.VLAN.id
    cidr_block      = "192.168.0.0/24"
    map_public_ip_on_launch = "true"
    availability_zone = "ap-east-1a"
    tags = {
        Name = "public_subnet"
        Environment = var.ENV
    }
}

resource "aws_subnet" "public_subnet_1" {
    vpc_id          = aws_vpc.VLAN.id
    cidr_block      = "192.168.1.0/24"
    map_public_ip_on_launch = "true"
    availability_zone = "ap-east-1b"
    tags = {
        Name = "public_subnet_1"
        Environment = var.ENV
    }
}

resource "aws_subnet" "private_subnet" {
    vpc_id          = aws_vpc.VLAN.id
    cidr_block      = "192.168.2.0/24"
    map_public_ip_on_launch = "false"
    availability_zone = "ap-east-1a"
    tags = {
        Name = "private_subnet"
        Environment = var.ENV
    }
}

resource "aws_subnet" "private_subnet_1" {
    vpc_id          = aws_vpc.VLAN.id
    cidr_block      = "192.168.3.0/24"
    map_public_ip_on_launch = "false"
    availability_zone = "ap-east-1b"
    tags = {
        Name = "private_subnet_1"
    }
}

```

```

        Environment = var.ENV
    }
}
resource "aws_internet_gateway" "internet_gateway" {
    vpc_id = aws_vpc.VLAN.id
    tags = {
        Name = "internet_gateway_thuongdd"
        Environment = var.ENV
    }
}
resource "aws_route_table" "route_table" {
    vpc_id = aws_vpc.VLAN.id
    route {
        cidr_block = "0.0.0.0/0"
        gateway_id = aws_internet_gateway.internet_gateway.id
    }
    tags = {
        Name = "route_table_thuongdd"
        Environment = var.ENV
    }
}
resource "aws_route_table_association" "route_table_association" {
    subnet_id      = aws_subnet.public_subnet.id
    route_table_id = aws_route_table.route_table.id
}
resource "aws_route_table_association" "route_table_association_1" {
    subnet_id      = aws_subnet.public_subnet_1.id
    route_table_id = aws_route_table.route_table.id
}

```

```

//security_group.tf
resource "aws_security_group" "public" {
    vpc_id = aws_vpc.VLAN.id
    name   = "allow-all"
    egress {
        from_port = 0
        to_port   = 0
        protocol  = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        from_port = 0
        to_port   = 0
        protocol  = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }
    tags = {
        Name = "Security"
    }
}

```

```

    }
}
resource "aws_security_group" "private" {
  vpc_id = aws_vpc.VLAN.id
  name   = "allow_bastion_host"
  egress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "Security"
    Environment = var.ENV
  }
}
}

```

```

//nat.tf
resource "aws_eip" "nat" {
  vpc = true
}
resource "aws_nat_gateway" "nat_gateway" {
  allocation_id = aws_eip.nat.id
  subnet_id     = aws_subnet.public_subnet.id
  depends_on    = [aws_internet_gateway.internet_gateway]
}
resource "aws_route_table" "nat" {
  vpc_id = aws_vpc.VLAN.id
  route {
    cidr_block      = "0.0.0.0/0"
    nat_gateway_id = aws_nat_gateway.nat_gateway.id
  }
  tags = {
    Name = "nat_private"
    Environment = var.ENV
  }
}
resource "aws_route_table_association" "route_table_association_2" {
  subnet_id      = aws_subnet.private_subnet.id
  route_table_id = aws_route_table.nat.id
}
resource "aws_route_table_association" "route_table_association_3" {

```

```
    subnet_id      = aws_subnet.private_subnet_1.id
    route_table_id = aws_route_table.nat.id
}
```

```
//instance.tf
resource "aws_instance" "node_1" {
  ami           = "ami-0b215afe809665ae5"
  instance_type = "t3.small"
  subnet_id     = aws_subnet.public_subnet.id
  security_groups = [aws_security_group.private.id]
  key_name      = var.key_name
  tags = {
    Name        = "node_1"
    Environment = var.ENV
  }
}

resource "aws_instance" "node_2" {
  ami           = "ami-0b215afe809665ae5"
  instance_type = "t3.small"
  subnet_id     = aws_subnet.private_subnet.id
  security_groups = [aws_security_group.private.id]
  key_name      = var.key_name
  tags = {
    Name = "node_2"
  }
}
```

```
//output.tf
output "bastion_host" {
  value = aws_instance.bastion_host.public_ip
}

output "private_master" {
  value = aws_instance.master.private_ip
}

output "private_node_1" {
  value = aws_instance.node_1.private_ip
}

output "private_node_2" {
  value = aws_instance.node_2.private_ip
}
```

```
//var.tf
variable "AWS_REGION" {
  default = "ap-east-1"
}
```

```
}  
  
variable "instance_username" {  
    default = "ubuntu"  
}  
  
variable "ENV" {  
    default = "test"  
}
```