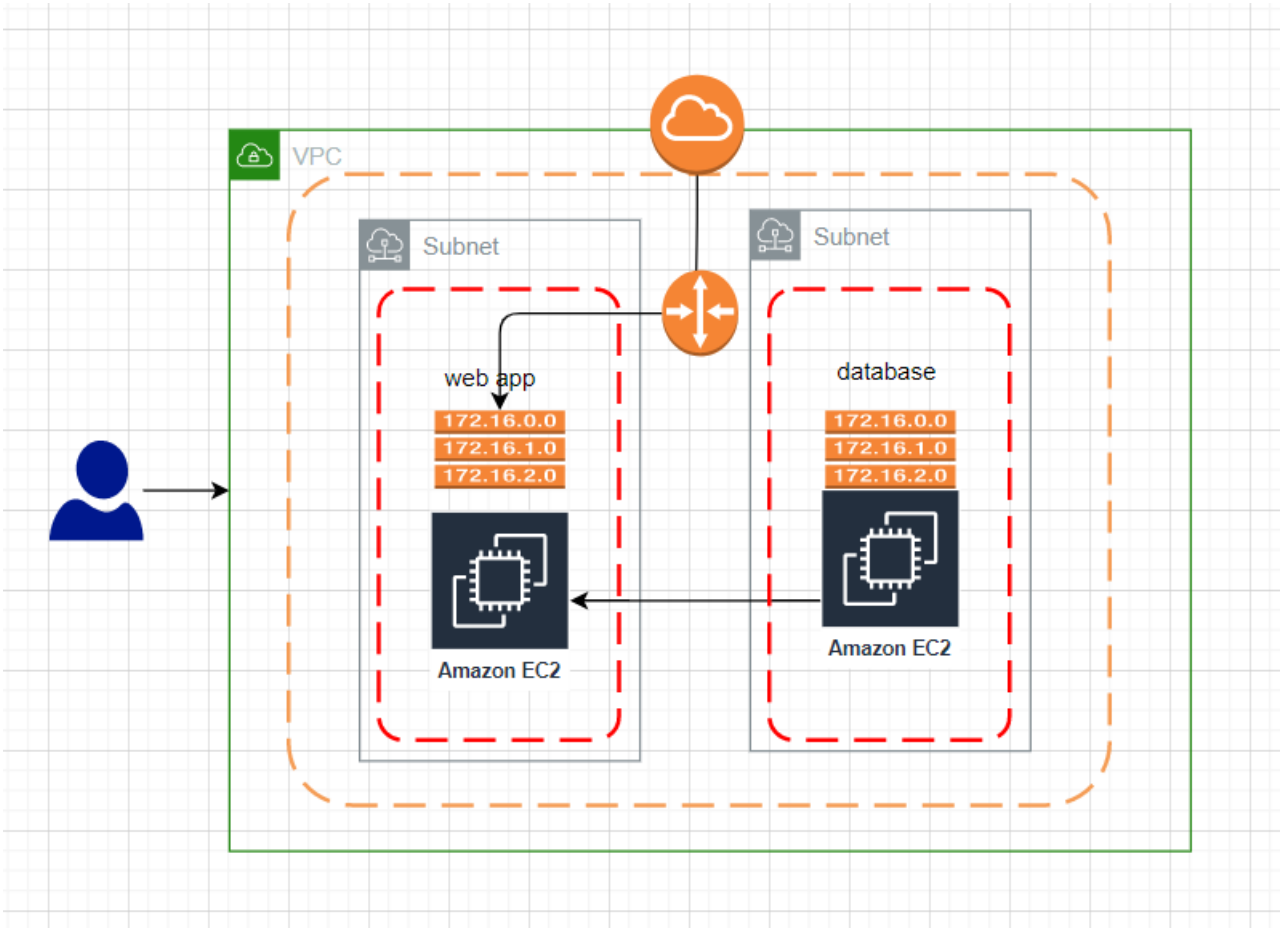
prepared by
Aya Rabih Mostafa

LinkedIn GitHub

What we will do in our project use terraform

- Create a VPC and subnets
- Create an internet gateway and route table to make the subnet public
- Create security groups
- Create an ec2 instance on a public subnet and install nginx
- Create an ec2 instance on private subnet and install database



Terraform and AWS

Terraform can be used with many providers like aws, azure, or google cloud. To use it with AWS, we first declare the aws provider with the region we're using to setup our infrastructure

We will use in our project AWS provider

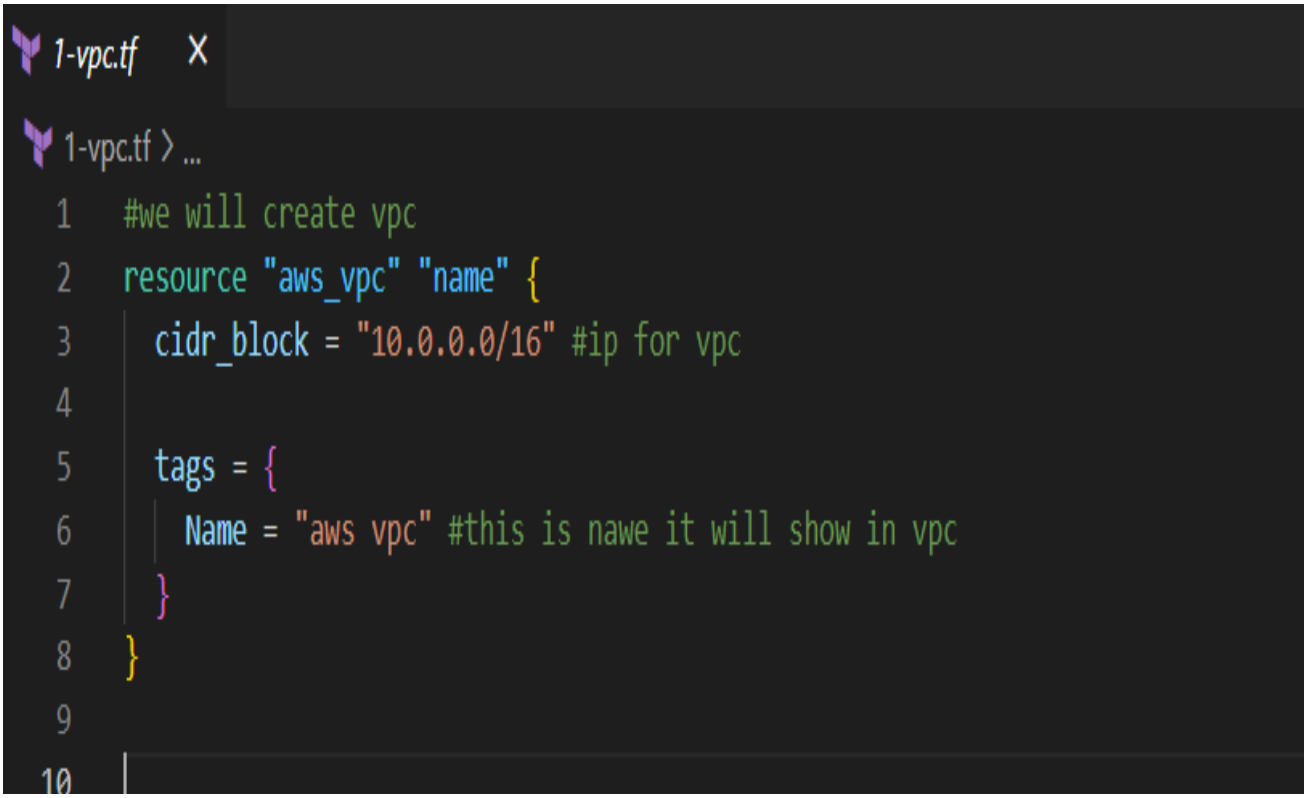
VPC and EC2 instance

When setting up a new VPC to deploy EC2 instances, we usually follow these basic steps.

1. Create a vpc
2. Create subnets for different parts of the infrastructure
3. Attach an internet gateway to the VPC
4. Create a route table for a public subnet
5. Create security groups to allow specific traffic
6. Create ec2 instances on the subnets

1. Create a vpc

This will setup a new VPC with the cidr block `10.0.0.0/16` and the name "Some Custom VPC". We can reference the VPC locally in the tf file using `some_custom_vpc`.



2. Create subnets for different parts of the infrastructure

This will create two new subnets in az 1a with the cidr blocks `10.0.1.0/24` and `10.0.2.0/24`. We need to use the VpcId from the previous step.

2-subnet.tf

```
#public-subnet
resource "aws_subnet" "some_public_subnet" { #in this line sould have two
  vpc_id            = aws_vpc.name.id
  cidr_block        = "10.0.1.0/24"
  availability_zone = "us-east-2a" #don't write 2a will give you error

  tags = {
    Name = "Some Public Subnet"
  }
}
#private-subnet
resource "aws_subnet" "some_private_subnet" {
  vpc_id            = aws_vpc.name.id
  cidr_block        = "10.0.2.0/24"
  availability_zone = "us-east-2a"

  tags = {
    Name = "Some Private Subnet"
  }
}
```

3. Attach an internet gateway to the VPC

This creates an internet gateway and attaches it to the custom VPC. Now we need a route table to handle routing to one or more of the subnets.

3-internet getway.tf

```
#we will create internet-getway to allow open internet
resource "aws_internet_gateway" "ig" {
  vpc_id = aws_vpc.name.id

  tags = {
    Name = " Internet Gateway" #this name will we show in dashbord
  }
}
```

4. Create a route table for a public subnet

This will create a new route table on the custom vpc. We can also specify the routes to route internet traffic through the gateway. So the route table and internet gateway are setup on The VPC, now we just need to associate any public subnets with the route table.

4-route table.tf

```
resource "aws_route_table" "public_rt" {
  vpc_id = aws_vpc.name.id

  route {
    cidr_block = "0.0.0.0/0" #this route will give you ipv4
    gateway_id = aws_internet_gateway.ig.id
  }

  route {
    ipv6_cidr_block = "::/0" #this route will give you ipv6
    gateway_id      = aws_internet_gateway.ig.id
  }

  tags = {
    Name = "Public Route Table"
  }
}

```

5.aws_route_table_association

Now some_public_subnet is accessible over the public internet.
create rote table attach to our subnet to attach new route table not table by default

5-route table attah.tf

```
#create rote table attach to our subnet to attach new route table not table by defult
resource "aws_route_table_association" "public_1_rt_a" {
  subnet_id      = aws_subnet.some_public_subnet.id
  route_table_id = aws_route_table.public_rt.id
}

```

6. Create security groups to allow specific traffic

Before we setup a new EC2 instance on the public subnet, we need to create a security group that allows internet traffic on port 80 and 22. We'll also allow outgoing traffic on all ports.

6-security group.tf

```
#we will create security-group to allow network
resource "aws_security_group" "web_sg" {
  name   = "HTTP and SSH"
  vpc_id = aws_vpc.name.id

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

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

  egress { #connect ip puplic
    protocol    = -1
      from_port   = 0
      to_port     = 0
      cidr_blocks = ["0.0.0.0/0"]
  }
  egress { #outbound
    from_port   = 0
    to_port     = 0
    protocol    = -1
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

7. First generate a new key pair

This will generate a new key pair and store the private key on your machine at ~/.ssh/MyKeyPair.pem

7-keypair.tf

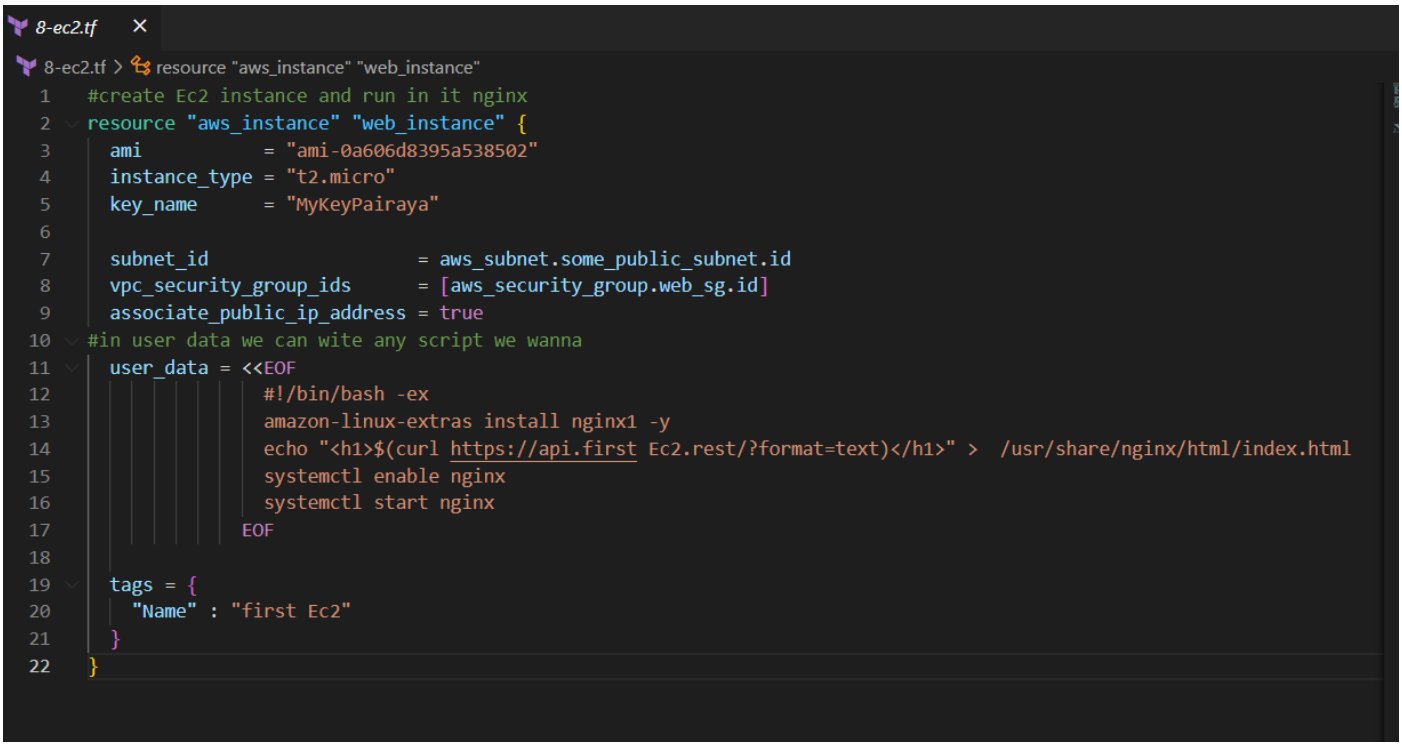
```
#we will create keypair we will use it when we lunch instance by this command
#don't forget to run it in your terminal before you start
# aws ec2 create-key-pair --key-name MyKeyPairaya --query 'KeyMaterial' --output text > ~/.ssh/MyKeyPairaya.pem
# chmod 400  ~/.ssh/MyKeyPairaya.pem

```

- ✓ And don't forget to hash it when you run to not give you error

8. Create ec2 instances on the subnets

Time to deploy an EC2 instance. If you already have an ssh keypair setup, you can just use that and skip the next step. If you haven't, or if you want to setup a new ssh key for this instance, run the following command using the aws cli



How we will run terraform Script

Here's what everything looks like as a single .tf file. Use the following commands to

- `terraform init`: Setup a new terraform project for this file.
- `terraform apply`: Setup the infrastructure as it's defined in the .tf file.
- `terraform destroy`: Tear down everything that terraform created.
- `terraform state list`: Show everything that was created by terraform.
- `terraform state show aws_instance.web_instance`: Show the details about the ec2 instance that was deployed

1-run command terraform init

```
aya@aya-Latitude-E6540:/media/aya/New Volume/project/terraform task$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

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.
aya@aya-Latitude-E6540:/media/aya/New Volume/project/terraform task$
```

2- if you wanna to see what you will deploy before you create it you will run comman terraform plan

```
  # aws_vpc.name will be created
  + resource "aws_vpc" "name" {
      + 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_classiclink                   = (known after apply)
      + enable_classiclink_dns_support       = (known after apply)
      + enable_dns_hostnames                 = (known after apply)
      + 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                                 = {
          + "Name" = "aws vpc"
        }
      + tags_all                             = {
          + "Name" = "aws vpc"
        }
    }

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

.

3- run command terraform apply

```
    }

Plan: 8 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_vpc.name: Creating...
aws_vpc.name: Creation complete after 5s [id=vpc-025eafb1de1c31c65]
aws_subnet.some_private_subnet: Creating...
aws_internet_gateway.ig: Creating...
aws_subnet.some_public_subnet: Creating...
aws_security_group.web_sg: Creating...
aws_subnet.some_private_subnet: Creation complete after 1s [id=subnet-0767ca91d1a9b88f1]
aws_internet_gateway.ig: Creation complete after 1s [id=igw-02ee5820dcc5c5530]
aws_route_table.public_rt: Creating...
aws_subnet.some_public_subnet: Creation complete after 1s [id=subnet-0090c0b09ba53c2e2]
aws_security_group.web_sg: Creation complete after 4s [id=sg-0cf03fc2d1bd9225b]
aws_instance.web_instance: Creating...
aws_route_table.public_rt: Creation complete after 3s [id=rtb-0b5d37e4d61ba64c9]
aws_route_table_association.public_1_rt_a: Creating...
aws_route_table_association.public_1_rt_a: Creation complete after 1s [id=rtbassoc-03a5285ff771b812e]
aws_instance.web_instance: Still creating... [10s elapsed]
aws_instance.web_instance: Still creating... [20s elapsed]
aws_instance.web_instance: Still creating... [30s elapsed]
aws_instance.web_instance: Creation complete after 39s [id=i-0960f939a343b5b97]

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.
aya@aya-Latitude-E6540:/media/aya/New Volume/project/terraform task$
```

4-one of our resource create it is Ec2

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Z...
first Ec2	i-0960f939a343b5b97	Running	t2.micro	2/2 checks passe	No alarms	us-east-2a

5-Now we will remove all resource by command terraform destroy

```
Plan: 0 to add, 0 to change, 8 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.public_1_rt_a: Destroying... [id=rtbassoc-03a5285ff771b812e]
aws_subnet.some_private_subnet: Destroying... [id=subnet-0767ca91d1a9b88f1]
aws_instance.web_instance: Destroying... [id=i-0960f939a343b5b97]
aws_route_table_association.public_1_rt_a: Destruction complete after 1s
aws_route_table.public_rt: Destroying... [id=rtb-0b5d37e4d61ba64c9]
aws_subnet.some_private_subnet: Destruction complete after 1s
aws_route_table.public_rt: Destruction complete after 1s
aws_internet_gateway.ig: Destroying... [id=igw-02ee5820dcc5c5530]
aws_instance.web_instance: Still destroying... [id=i-0960f939a343b5b97, 10s elapsed]
aws_internet_gateway.ig: Still destroying... [id=igw-02ee5820dcc5c5530, 10s elapsed]
aws_instance.web_instance: Still destroying... [id=i-0960f939a343b5b97, 20s elapsed]
aws_internet_gateway.ig: Still destroying... [id=igw-02ee5820dcc5c5530, 20s elapsed]
aws_instance.web_instance: Still destroying... [id=i-0960f939a343b5b97, 30s elapsed]
aws_internet_gateway.ig: Still destroying... [id=igw-02ee5820dcc5c5530, 30s elapsed]
aws_internet_gateway.ig: Destruction complete after 30s
aws_instance.web_instance: Destruction complete after 34s
aws_subnet.some_public_subnet: Destroying... [id=subnet-0090c0b09ba53c2e2]
aws_security_group.web_sg: Destroying... [id=sg-0cf03fc2d1bd9225b]
aws_subnet.some_public_subnet: Destruction complete after 1s
aws_security_group.web_sg: Destruction complete after 1s
aws_vpc.name: Destroying... [id=vpc-025eafb1de1c31c65]
aws_vpc.name: Destruction complete after 1s

Destroy complete! Resources: 8 destroyed.
aya@aya-Latitude-E6540:/media/aya/New Volume/project/terraform task$
```