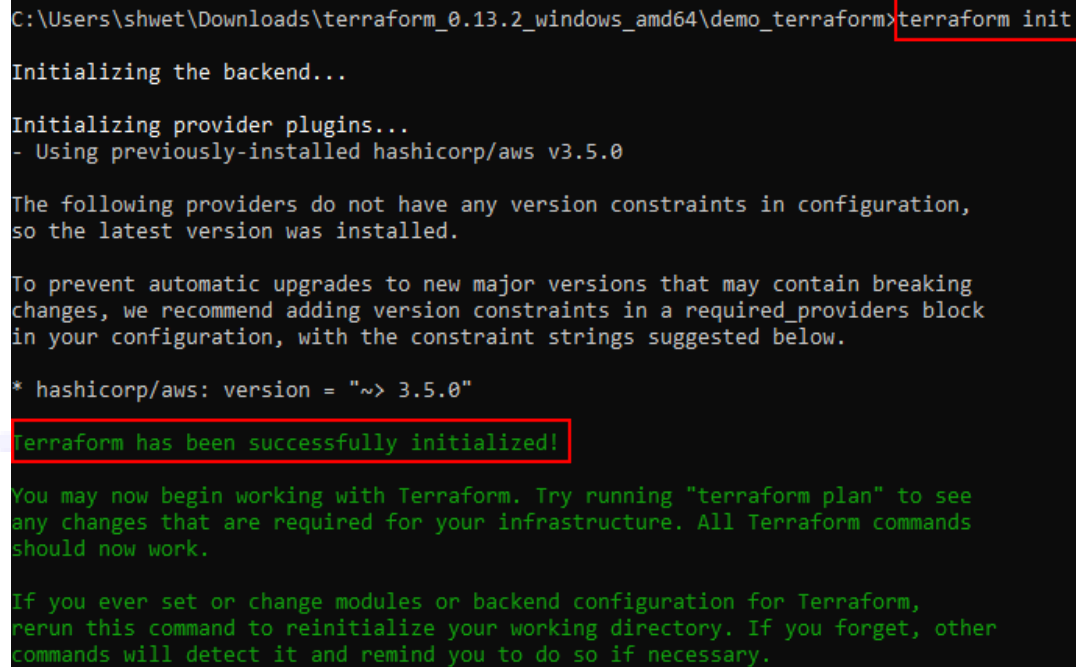


Deploying End-to-End Architecture on AWS

First, we will open terminal or command Prompt, go to the path where we have created the folder for terraform and type **terraform init** command. Whenever you are starting to use terraform, first you need to run this command to tell terraform which provider you are using.



```
C:\Users\shwet\Downloads\terraform_0.13.2_windows_amd64\demo_terraform>terraform init

Initializing the backend...

Initializing provider plugins...
- Using previously-installed hashicorp/aws v3.5.0

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, we recommend adding version constraints in a required_providers block
in your configuration, with the constraint strings suggested below.

* hashicorp/aws: version = "~> 3.5.0"

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.
```

Then the next step is to configure the providers which you want to use, here we use AWS. Create a file called **main.tf** and add the provider information to it. Save it in the location where terraform is saved.

```
provider "aws" {
  region = "us-west-2"
  access_key = "" #Access Key of your account
  secret_key = "" #Secret Key of Your Account
}
```

This code tells Terraform to create an AWS instance in us-west-2 Region and it specifies your account using the access key and secret Key.

For each provider, there are different resources like instances, databases, VPC etc. which can be created.

The syntax for creating a resource is

```
resource "<PROVIDER>_<TYPE>" "<NAME>" {  
    [configuration]  
}
```

Provider can be AWS, Azure, gcp etc. Type is the type of resource which we want to create (ex: Database, instance). Configuration will in Key-Value Pairs that are specific to that resource.

So, now we are going to create an architecture with VPC, subnets etc.

Creating a VPC:

First, we will create VPC, inside which we are going to create our instance.

```
resource "aws_vpc" "main" {  
    cidr_block = "192.168.0.0/16"  
    enable_dns_support = "1"  
    enable_dns_hostnames = "1"  
    tags = {  
        Name = "myfirstvpc"    # Name of your vpc  
    }  
}
```

Cidr_block specifies the IP range for the VPC. **Enable_dns_support** and **enable_dns_hostnames** are the attributes which specifies whether or not VPC has DNS support and DNS hostname support. The value 1 indicates it is True or it supports both DNS and DNS hostname.

cidr_block value can be anything between the range 0.0.0.0 to 255.255.255.255.

Name tag specifies the name of the VPC.

Creating a subnet:

Now inside VPC, we are going to create a subnet which enables you to ensure that information remains in the subnetted network.

```
resource "aws_subnet" "first" {
  availability_zone = "us-west-2a"
  cidr_block = "192.168.1.0/24"
  map_public_ip_on_launch = "1"
  vpc_id = "${aws_vpc.main.id}" #from the vpc resource aws_vpc main
  tags = {
    Name = "myfirstsubnet"
  }
}
```

This code creates a subnet inside the VPC with the help of vpc_id which is specified on the code. The vpc_id will be the aws_vpc.resource name.id. This we can get it from the above VPC resource code.

Creating an instance inside the subnet:

An instance is a virtual server in Amazon's EC2 which allows subscribers to run their application programs in the computing environment.

```
resource "aws_instance" "firstec2" {
  ami = "ami-0ba60995c1589da9d"
  instance_type = "t2.micro"
  key_name = "" #This can be downloaded from your AWS Account
  subnet_id = "${aws_subnet.first.id}" # from the subnet resource aws_subnet first
  tags = {
    Name = "knode"
  }
  user_data = file("./install.sh")
}
```

In this we specify the computing environment (i.e. ami) and the type of instance which we want to deploy. The Key_name can be downloaded from your AWS account.

Deploying internet getaway:

Internet gateway allows communication between your VPC and the internet. To enable access to or from the internet, attach an internet gateway to your VPC.

```
resource "aws_internet_gateway" "internet" {  
  vpc_id = "${aws_vpc.main.id}"  
  tags = {  
    Name = "myinternetgateway"  
  }  
}
```

The **vpc_id** specifies which vpc we are connecting the internet gateway and the **name tag** tells the name of the internet gateway.

In order to handle traffic, we are going to use route table. This helps the network packets to reach their destination and to connect internet gateway to public subnets.

```
resource "aws_route" "internet" {  
  route_table_id      = "${aws_vpc.main.default_route_table_id}"  
  destination_cidr_block = "0.0.0.0/0"  
  gateway_id = "${aws_internet_gateway.internet.id}"  
}  
  
resource "aws_route_table_association" "a" {  
  subnet_id    = "${aws_subnet.first.id}"  
  route_table_id = "${aws_vpc.main.default_route_table_id}"  
}
```

Deploying Security group:

If you don't want to specify security group, then Amazon takes it as default security group where everything will be enabled. The security group will be applied on instance level. Each instance inside a same subnet can have different security group.

```
resource "aws_default_security_group" "default_myfirst" {  
  ingress {  
    from_port = 0  
    to_port   = 0  
    protocol  = "-1"  
    cidr_blocks = ["0.0.0.0/0"]  
  }  
  egress {  
    from_port = 0  
    to_port   = 0  
    protocol  = "-1"  
    cidr_blocks = ["0.0.0.0/0"]  
  }  
  vpc_id = "${aws_vpc.main.id}"  
  tags = {  
    Name = "myfirstsecuritygroup"  
  }  
}
```

Ingress are the incoming traffic from the internet to instance. The from_port and to_port Specifies that it allows all the traffic from internet to uncommon ports i.e. the instance associated with the security group is not restricted. Adding default security code is optional. Even if you don't add it will take the default security Rules.

To create a virtual network inside the subnets we are going to use `aws_network_interface` and attach the network interface to the instance.

```
resource "aws_network_interface" "first" {
  subnet_id = "${aws_subnet.first.id}"
  tags = {
    Name = "mynetworkinterface"
  }
}

resource "aws_network_interface_attachment" "connect" {
  instance_id      = "${aws_instance.firstec2.id}"
  network_interface_id = "${aws_network_interface.first.id}"
  device_index     = 1
}
```

Now, to extract the value of an output variable from the state file, we are going to use this terraform output command.

```
output "IPs" {
  value = "kmaster - ${aws_instance.firstec2.public_ip}"
}
```

So, these are the configurations which we want our instance to have.

Now we need to run the **terraform plan** command. This command lets you see the changes before making any. This helps us to check the code before actually deploying it.

```
C:\Users\shwet\Downloads\terraform_0.13.2_windows_amd64\demo_terraform>terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

aws_instance.example: Refreshing state... [id=i-0fd65dd6829eef150]

-----

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.example will be created
+ resource "aws_instance" "example" {
  + ami              = "ami-0ba60995c1589da9d"
  + 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)
  + get_password_data  = false
  + host_id           = (known after apply)
  + id                = (known after apply)
  + instance_state     = (known after apply)
  + instance_type      = "t2.micro"
```

Anything with a (+) indicates that it will be created, anything with (-) will be deleted and anything with (~) will be modified. The specified ami and instance type is as mentioned and rest of the values will be taken as default.

To actually create an instance, we need to run the **terraform apply** command.

```
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_network_interface_attachment.connect: Creating...
aws_network_interface_attachment.connect: Still creating... [10s elapsed]
aws_network_interface_attachment.connect: Creation complete after 14s [id=eni-attach-0fd57e46c537d0f1a]

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

Outputs:

IPs = kmaster - 34.214.157.148
```

Now a VPC will be deployed in your AWS account using Terraform. Open your AWS Management Console and open VPC. Let's check whether it is created or not.

<input type="checkbox"/>	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
<input type="checkbox"/>	-	vpc-4b5a8233	Available	172.31.0.0/16	-
<input type="checkbox"/>	myfirstvpc	vpc-0a6e02410f3b32c03	Available	192.168.0.0/16	-

Now let's check the subnet and instance

<input type="checkbox"/>	Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4	IPv6 CIDR	Availability Zone	Availability Zone ID	Network
<input type="checkbox"/>	myfirstsubnet	subnet-0fd0ac82df7503e59	available	vpc-029f6b0d7f5593ed9 ...	192.168.1.0/24	249	-	us-west-2a	usw2-az2	us-west-2
<input type="checkbox"/>		subnet-79873852	available	vpc-4b5a8233	172.31.32.0/20	4090	-	us-west-2d	usw2-az4	us-west-2

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm St...	Availability zone	Public IPv4 DNS	Public IPv4 ...
<input type="checkbox"/>	knode	i-0e9bbdb78995c9a29	Running	t2.micro	2/2 checks ...	No alarms +	us-west-2a	ec2-34-214-157-148.u...	34.214.157.14
<input type="checkbox"/>	shwetha inst...	i-09ca0b558a3de57a9	Stopped	t2.micro	-	No alarms +	us-west-2c	-	-