

Home / AWS / Guided Lab / Build a Custom VPC and its components using Terraform

## Build a Custom VPC and its components using Terraform

Level: Fundamental

Amazon VPC   Amazon Web Services   Terraform



0h 59m 40s left



End Lab

Open Console

Validation

### Lab Credentials

User Name ⓘ

Whiz\_User\_80425.60918563



Password ⓘ

f28c3bed-4f8c-4125-b9e2-95ac6ba10d0f



Access Key ⓘ

AKIAYKGOODAZ5VLHQGXK



Secret Key ⓘ

M5oaZ+VMIPvO5omgK5WUyHGCMMKewQfETXsXnGNl






### Lab Resources

No Lab Resources Found

### Support Documents

No Support Documents Found

## Need help?

-  How to use Hands on Lab
-  Troubleshooting Lab
-  FAQs



 Cloud Network Engineer

 Networking, Infrastructure

# Lab Steps

## Task 1: Sign in to AWS Management Console

1. Click on the **Open Console** button, and you will get redirected to AWS Console in a new browser tab.
2. On the AWS sign-in page,
  - Leave the **Account ID** as default. Never edit/remove the 12 digit Account ID present in the AWS Console. otherwise, you cannot proceed with the lab.
  - Now copy your **User Name** and **Password** in the Lab Console to the **IAM Username** and **Password** in AWS Console and click on the **Sign in** button.
3. Once Signed In to the AWS Management Console, Make the default AWS Region as **US East (N. Virginia) us-east-1**.

## Task 2: Setup Visual Studio Code

1. Open the Visual Studio Code.
2. If you have already installed and using the Visual Studio code, open a new window.
3. A new window will open a new file and release notes page (only if you have installed or updated Visual Studio Code recently). Close the Release notes tab.

4. Open terminal by selecting View from the Menu bar and choose Terminal.
5. It may take up to 2 minutes to open the terminal window.
6. Once the terminal is ready, let us navigate to the **Desktop**.

```
cd Desktop
```



7. Create a new folder by running the below command:

```
mkdir task_10099_vpc
```



8. Change your present working directory to use the newly created folder by running the command:

```
cd task_10099_vpc
```



9. Get the location of the present working directory by running the below command:

```
pwd
```



10. Note down the location, as you will open the same in the next steps.

11. Now click on the first icon Explorer present on the left sidebar.

12. Click on the button called Open folder and navigate to the location of folder **task\_10099\_vpc**

13. Visual Studio code is now ready to use.

### Task 3: Create a variables file

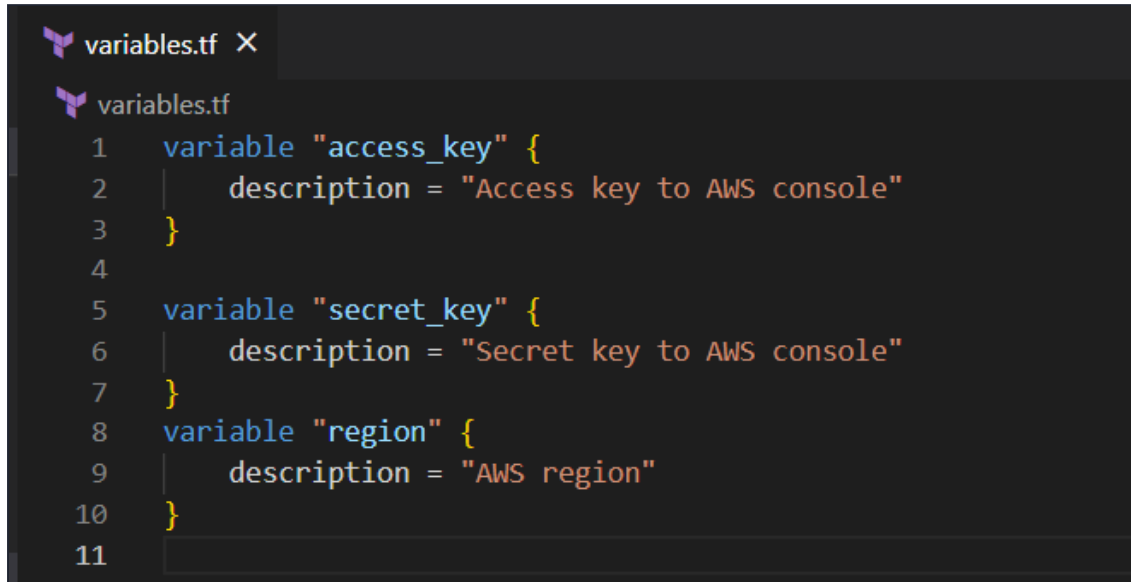
1. To create a variables file, expand the folder **task\_10099\_vpc** and click on the **New File** icon to add the file.
2. Name the file as **variables.tf** and press **Enter** to save it.
3. **Note:** Don't change the location of the new file, keep it default, i.e. inside the **task\_10099\_vpc** folder.
4. Paste the below contents in **variables.tf** file.

```
variable "access_key" {  
    description = "Access key to AWS console"  
}  
variable "secret_key" {  
    description = "Secret key to AWS console"  
}  
variable "region" {  
    description = "AWS region"}
```



```
}
```

5. In the above content, you are declaring a variable called, `access_key`, `secret_key`, and `region` with a short description of all 3.
6. After pasting the above contents, save the file by pressing **Ctrl + S**.



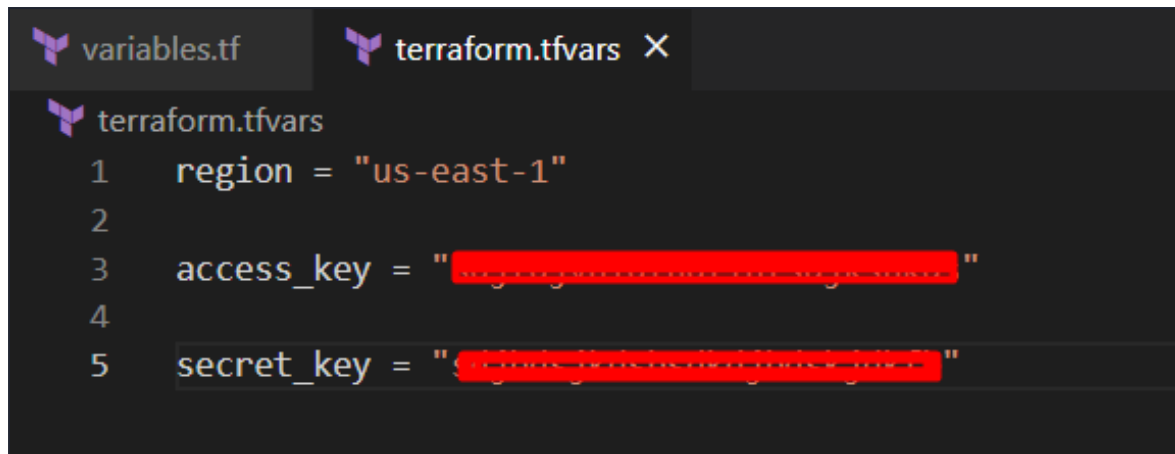
```
variables.tf X
variables.tf
1  variable "access_key" {
2      description = "Access key to AWS console"
3  }
4
5  variable "secret_key" {
6      description = "Secret key to AWS console"
7  }
8  variable "region" {
9      description = "AWS region"
10 }
11
```

7. Now expand the folder **task\_10099\_vpc** and click on the **New File** icon to add the file.
8. Name the file as **terraform.tfvars** and press **Enter** to save it.
9. Paste the below content into the **terraform.tfvars** file.

```
region = "us-east-1"
access_key = "YOUR_ACCESS_KEY"
secret_key = "YOUR_SECRET_KEY"
```



10. In the above code, you are defining the dynamic values of variables declared earlier.
11. Replace the values of `access_key` and `secret_key` by copying from the lab page.
12. After replacing the values of **`access_key`** and **`secret_key`**, save the file by pressing **Ctrl + S**.



```
variables.tf  terraform.tfvars X
terraform.tfvars
1  region = "us-east-1"
2
3  access_key = "[REDACTED]"
4
5  secret_key = "[REDACTED]"
```

#### Task 4: Create a VPC in the main.tf file

1. To create a **main.tf** file, expand the folder **task\_1009\_vpc** and click on the **New File** icon to add the file.
2. Name the file as **main.tf** and press **Enter** to save it.
3. Paste the below content into the **main.tf** file.

```
provider "aws" {
  region      = "${var.region}"
  access_key  = "${var.access_key}"
  secret_key  = "${var.secret_key}"
}
```



4. In the above code, you are defining the provider as **aws**.
5. Next, we want to tell Terraform to create a VPC named as **MyVPC**.
6. Paste the below content into the **main.tf** file after the provider.

```
resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "MyVPC"
  }
}
```



7. In the above code we are telling terraform to create a VPC with CIDR block **10.0.0.0/16**.
8. The tags are used for naming the VPC. The name of the VPC will be **MyVPC**.

A screenshot of a code editor window titled 'main.tf'. The code is Terraform configuration for an AWS VPC. It starts with a 'provider "aws"' block that uses variables for region, access\_key, and secret\_key. Then, it defines an 'aws\_vpc' resource named 'main' with a cidr\_block of '10.0.0.0/16' and a tag 'Name = "MyVPC"'.

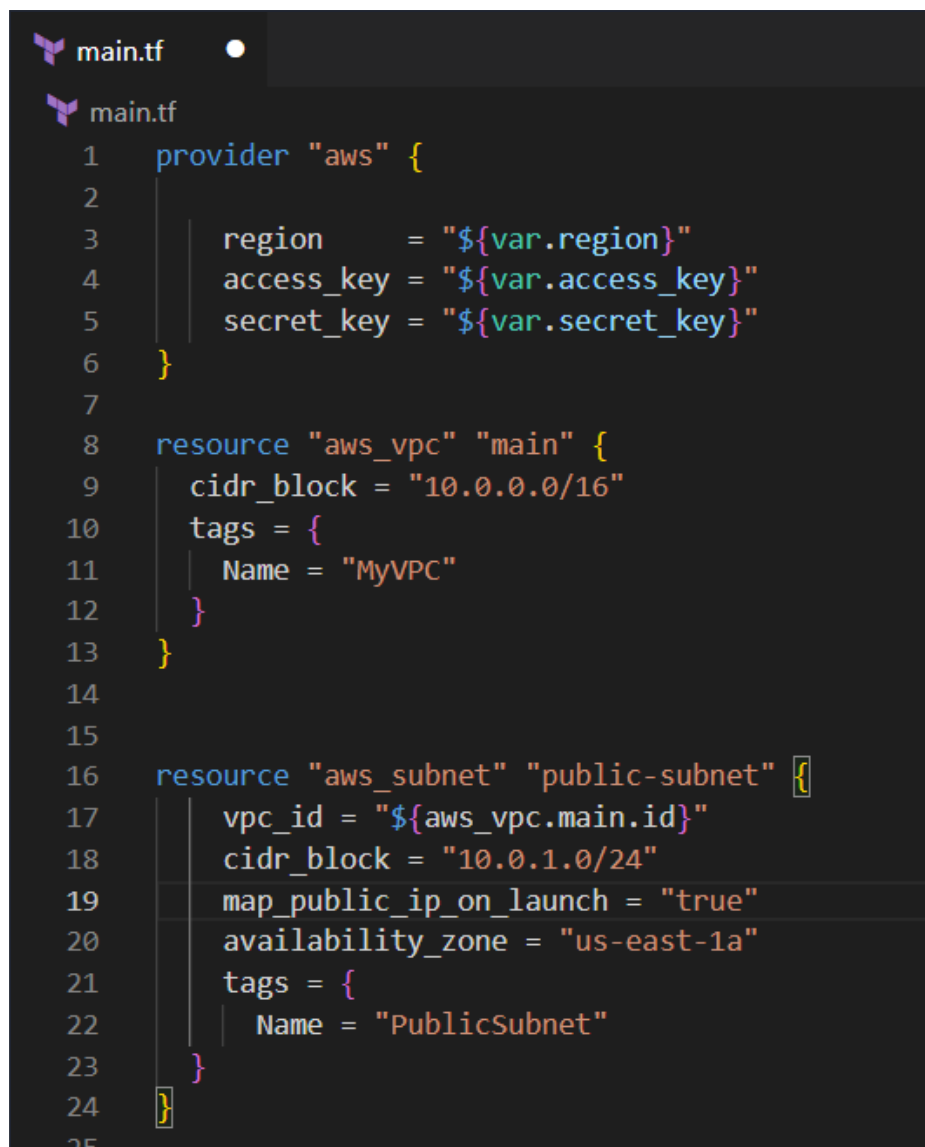
```
main.tf
1  provider "aws" {
2
3      region      = "${var.region}"
4      access_key  = "${var.access_key}"
5      secret_key  = "${var.secret_key}"
6  }
7
8  resource "aws_vpc" "main" {
9      cidr_block = "10.0.0.0/16"
10     tags = {
11         Name = "MyVPC"
12     }
13 }
14
```

## Task 5: Adding subnets to the VPC in the main.tf file

1. To add public subnet to the VPC , paste the following content in the **main.tf**.

```
resource "aws_subnet" "public-subnet" {
    vpc_id = "${aws_vpc.main.id}"
    cidr_block = "10.0.1.0/24"
    map_public_ip_on_launch = "true"
    availability_zone = "us-east-1a"
    tags = {
        Name = "PublicSubnet"
    }
}
```





```

main.tf
1  provider "aws" {
2
3      region      = "${var.region}"
4      access_key  = "${var.access_key}"
5      secret_key  = "${var.secret_key}"
6  }
7
8  resource "aws_vpc" "main" {
9      cidr_block = "10.0.0.0/16"
10     tags = {
11         Name = "MyVPC"
12     }
13 }
14
15
16 resource "aws_subnet" "public-subnet" {
17     vpc_id = "${aws_vpc.main.id}"
18     cidr_block = "10.0.1.0/24"
19     map_public_ip_on_launch = "true"
20     availability_zone = "us-east-1a"
21     tags = {
22         Name = "PublicSubnet"
23     }
24 }
25

```

2. In the above code, we have used the resource type as **aws\_subnet** and associated it with our VPC. The CIDR block assigned is **10.0.1.0/24**. To make it public , we have used `map_public_ip_on_launch` as `true`. The availability zone assigned is the **us-east-1a**.
3. You can similarly add a private subnet. Paste the following content to add the private subnet:

```

resource "aws_subnet" "private-subnet" {
    vpc_id = "${aws_vpc.main.id}"
    cidr_block = "10.0.2.0/24"
    availability_zone = "us-east-1b"
    tags = {
        Name = "Private Subnet"
    }
}

```



```
main.tf
main.tf

5      secret_key = "${var.secret_key}"
6    }
7
8    resource "aws_vpc" "main" {
9      cidr_block = "10.0.0.0/16"
10     tags = {
11       Name = "MyVPC"
12     }
13   }
14
15
16   resource "aws_subnet" "public-subnet" {
17     vpc_id = "${aws_vpc.main.id}"
18     cidr_block = "10.0.1.0/24"
19     map_public_ip_on_launch = "true"
20     availability_zone = "us-east-1a"
21     tags = {
22       Name = "PublicSubnet"
23     }
24   }
25
26
27   resource "aws_subnet" "private-subnet" {
28     vpc_id = "${aws_vpc.main.id}"
29     cidr_block = "10.0.2.0/24"
30     availability_zone = "us-east-1b"
31     tags = {
32       Name = "Private Subnet"
33     }
34   }
```

## Task 6: Add internet gateway and route tables in the main.tf file

In this task, you will add an internet gateway and associate it with the VPC created earlier. Also, you will create route tables for the subnets.

1. To add **internet gateway** and associate it with the VPC, Paste the below contents in **main.tf** file



```
resource "aws_internet_gateway" "MyIGW" {  
  vpc_id = "${aws_vpc.main.id}"  
  tags = {  
    Name = "MyInternetGateway"  
  }  
}
```



2. To create a public route table ,add the following code in the **main.tf** file

```
resource "aws_route_table" "publicrt" {  
  vpc_id = "${aws_vpc.main.id}"  
  route {  
    cidr_block = "0.0.0.0/0"  
    gateway_id = "${aws_internet_gateway.MyIGW.id}"  
  }  
  tags = {  
    Name = "PublicRouteTable"  
  }  
}
```



3. In the above code, we are creating a **public route table** describing the VPC , its route and name.

4. Similarly, to create a private route table, add the following code in the **main.tf** file

```
resource "aws_route_table" "privatert" {  
  vpc_id = "${aws_vpc.main.id}"  
  tags = {  
    Name = "PrivateRouteTable"  
  }  
}
```



5. **Kindly note that we haven't included the route while creating private route table.**

## Task 7: Associate route tables with the subnets

In this task , we will associate the route table with their respective subnets.

1. To associate the public route table with the public subnet, add the following code in the **main.tf** file.

```
resource "aws_route_table_association" "public-association"{  
  subnet_id = "${aws_subnet.public-subnet.id}"  
  route_table_id = "${aws_route_table.publicrt.id}"  
}
```



2. To associate the private subnet with the private subnet , add the following code in the **main.tf** file.

```
resource "aws_route_table_association" "private-association"{
  subnet_id = "${aws_subnet.private-subnet.id}"
  route_table_id = "${aws_route_table.privatert.id}"
}
```



## Task 8: Create an output file

1. To create an **output.tf** file, expand the folder **task\_10099\_vpc** and click on the **New File** icon to add the file.
2. Name the file as **output.tf** and press **Enter** to save it.
3. Paste the below content into the **output.tf** file.

```
output "vpc_id" {
  value= aws_vpc.main.id
}
output "public_subnet"{
  value = aws_subnet.public-subnet.id
}
output "private_subnet"{
  value = aws_subnet.private-subnet.id
}
```



4. In the above code ,we will extract the VPC ID , Public subnet ID, and Private subnet ID.

## Task 9: Applying terraform configurations

1. Initialize Terraform by running the below command,

```
terraform init
```



```
PS C:\Users\... \Desktop\task_10099 _vpc> terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.40.0

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.
```

2. To generate the action plans run the below command,

```
terraform plan
```



```
PS C:\Users\... \Desktop\task_10099 _vpc> terraform plan

Terraform used the selected providers to generate the following execution
+ create

Terraform will perform the following actions:

# aws_internet_gateway.MyIGW will be created
+ resource "aws_internet_gateway" "MyIGW" {
  + arn          = (known after apply)
  + id           = (known after apply)
  + owner_id     = (known after apply)
  + tags         = {
    + "Name" = "MyInternetGateway"
  }
  + tags_all     = {
    + "Name" = "MyInternetGateway"
  }
  + vpc_id       = (known after apply)
}
```

3. To create all the resources declared in **main.tf** configuration file, run the following command,

```
terraform apply
```



4. Enter **yes** and the resources will be created in 2-3 minutes.



```

Enter a value: yes

aws_vpc.main: Creating...
aws_vpc.main: Creation complete after 6s [id=vpc-07383294a9634b182]
aws_subnet.private-subnet: Creating...
aws_route_table.privatert: Creating...
aws_subnet.public-subnet: Creating...
aws_internet_gateway.MyIGW: Creating...
aws_subnet.private-subnet: Creation complete after 2s [id=subnet-0772ff6575581b3b7]
aws_route_table.privatert: Creation complete after 2s [id=rtb-07266f0428a6ffb69]
aws_route_table_association.private-association: Creating...
aws_internet_gateway.MyIGW: Creation complete after 2s [id=igw-0aa95f89ea3f6f644]
aws_route_table.publicrt: Creating...
aws_route_table_association.private-association: Creation complete after 1s [id=rtbassoc-0a139b2092fea2b61]
aws_route_table.publicrt: Creation complete after 4s [id=rtb-0a3c775babec5087f]
aws_subnet.public-subnet: Still creating... [10s elapsed]
aws_subnet.public-subnet: Creation complete after 14s [id=subnet-07d61bf4e6c26c19e]
aws_route_table_association.public-association: Creating...
aws_route_table_association.public-association: Creation complete after 1s [id=rtbassoc-00764d05ecaccef63]

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

Outputs:

private_subnet = "subnet-0772ff6575581b3b7"
public_subnet = "subnet-07d61bf4e6c26c19e"
vpc_id = "vpc-07383294a9634b182"

```

## Task 10: Check the resources in the AWS Console

1. Navigate to the **VPC** page by clicking on the **Services** menu at the top. **VPC** is available under the **Networking and Content Delivery** section.
2. Click on **Your VPCs** on the left navigation panel.
3. You can view that the VPC is created successfully.

Your VPCs (2) <a href="#">Info</a>						
<input type="text" value="Filter VPCs"/> <span>&lt; 1 &gt;</span>						
<input type="checkbox"/>	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	
<input type="checkbox"/>	MyVPC	vpc-07383294a9634b182	Available	10.0.0.0/16	-	

4. Similarly Click on **Subnets** on the left navigation panel. You can see that both the public and private subnets are created.

<input type="checkbox"/>	Name	Subnet ID	State	VPC	IPv4 CIDR
<input type="checkbox"/>	PublicSubnet	subnet-07d61bf4e6c26c19e	Available	vpc-07383294a9634b182   M...	10.0.1.0/24
<input type="checkbox"/>	Private Subnet	subnet-0772ff6575581b3b7	Available	vpc-07383294a9634b182   M...	10.0.2.0/24

5. Click on the **Route tables** on the left navigation panel. You can see both the public and private route tables are created.

<input type="checkbox"/>	Name	Route Table ID	Subnet ID	State	Associations	VPC
<input type="checkbox"/>	PrivateRouteTable	rtb-07266f0428a6ffb69	subnet-0772ff6575581...	Available	1	vpc-07383294a9634b182
<input type="checkbox"/>	PublicRouteTable	rtb-0a3c775babec5087f	subnet-07d61bf4e6c26...	Available	1	vpc-07383294a9634b182

6. **Select** the Public route table and Click on **Routes** in the details and you can see that the rule for internet gateway is associated with it

Details	Routes	Subnet associations	Edge associations	Route propagation	Tags
<b>Routes (2)</b> <input type="text" value="Filter routes"/> <span>Both ▼</span>					
Destination	Target	Status	Propagated		
0.0.0.0/0	<a href="#">igw-0aa95f89ea3f6f644</a>	✓ Active	No		
10.0.0.0/16	local	✓ Active	No		

7. Click on **Subnet Associations** and you can see that the public route table is associated with the public subnet.

Details	Routes	Subnet associations	Edge associations	Route propagation	Tags
<b>Explicit subnet associations (1)</b> <input type="text" value="Find subnet association"/>					
Subnet ID	IPv4 CIDR	IPv6 CIDR			
<a href="#">subnet-07d61bf4e6c26c19e</a> / PublicSubnet	10.0.1.0/24	–			

8. Click on the **Internet Gateways** on the left navigation panel. You can view the internet gateway created successfully.

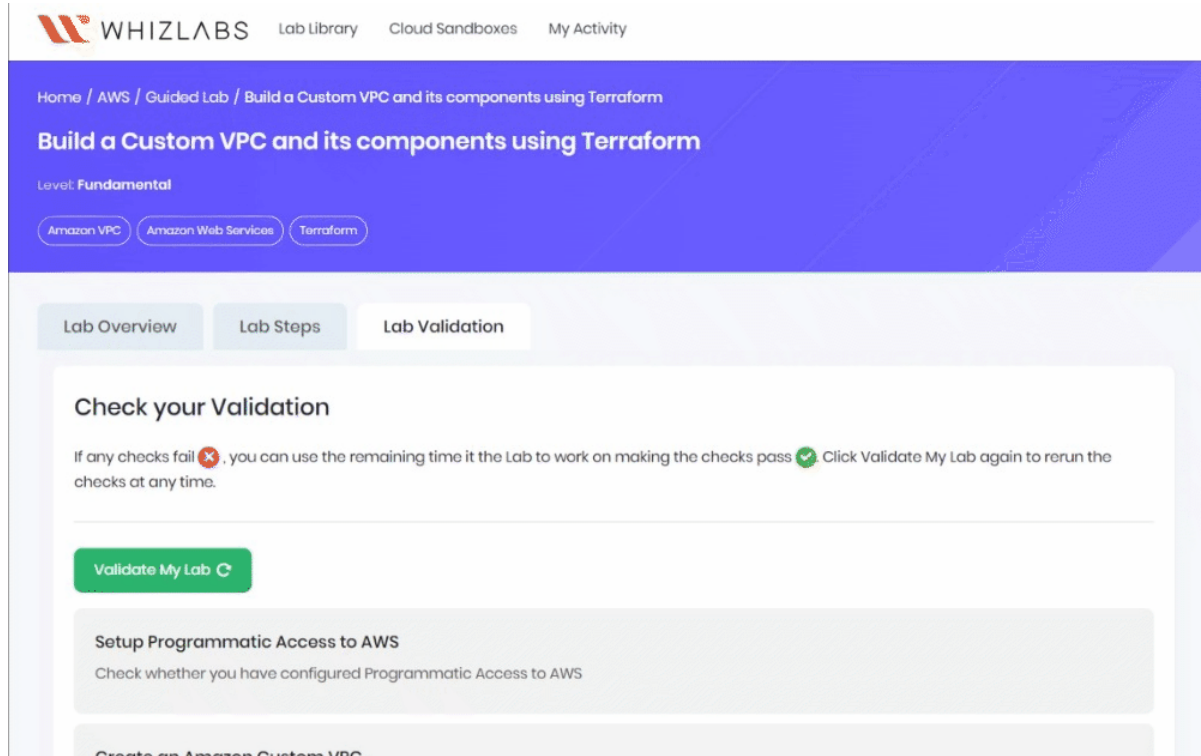
Internet gateways (2) <a href="#">Info</a>					<a href="#">Refresh</a>	<a href="#">Actions</a> ▼	<a href="#">Create</a>
<input type="text" value="Filter internet gateways"/>							
<input type="checkbox"/>	Name	Internet gateway ID	State	VPC ID			
<input type="checkbox"/>	my-igw	<a href="#">igw-0a3d648ed1b879390</a>	✓ Attached	<a href="#">vpc-0c4300e686542abb8</a>			
<input type="checkbox"/>	MyInternetGateway	<a href="#">igw-0aa95f89ea3f6f644</a>	✓ Attached	<a href="#">vpc-07383294a9634b182</a>   MyVPC			

## Do You Know ?

Terraform's **aws\_vpc** resource can automatically calculate and allocate CIDR blocks for your VPC subnets using the **cidr\_block** argument in a unique way known as "subnet sizing".

### Task 11: Validation of the Lab

1. Once the lab steps are completed, please click on the **Validation** button on the left side panel.
2. This will validate the resources in the AWS account and displays whether you have completed this lab successfully or not.
3. Sample output :



## Task 12: Delete AWS Resources

1. To delete the resources, open **Terminal** again.
2. Run the below command to delete all the resources.

```
terraform destroy
```



3. Enter **yes** to confirm the deletion.

```
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.private-association: Destroying... [id=rtbassoc-0a139b2092fea2b61]  
aws_route_table_association.public-association: Destroying... [id=rtbassoc-00764d05ecaccef63]  
aws_route_table_association.private-association: Destruction complete after 1s  
aws_route_table.privatert: Destroying... [id=rtb-07266f0428a6ffb69]  
aws_subnet.private-subnet: Destroying... [id=subnet-0772ff6575581b3b7]  
aws_route_table_association.public-association: Destruction complete after 1s  
aws_subnet.public-subnet: Destroying... [id=subnet-07d61bf4e6c26c19e]  
aws_route_table.publicrt: Destroying... [id=rtb-0a3c775babec5087f]  
aws_subnet.private-subnet: Destruction complete after 1s  
aws_subnet.public-subnet: Destruction complete after 1s  
aws_route_table.privatert: Destruction complete after 2s  
aws_route_table.publicrt: Destruction complete after 2s  
aws_internet_gateway.MyIGW: Destroying... [id=igw-0aa95f89ea3f6f644]  
aws_internet_gateway.MyIGW: Destruction complete after 1s  
aws_vpc.main: Destroying... [id=vpc-07383294a9634b182]  
aws_vpc.main: Destruction complete after 1s  
  
Destroy complete! Resources: 8 destroyed.
```

## Completion and Conclusion

1. You have set up the Visual Studio Code
2. You have successfully created variables.tf and terraform.tfvars files.
3. You have successfully created a main.tf file.
4. You have executed the terraform configurations commands to create the resources.
5. You have checked the resources created by opening the AWS Console.
6. You have deleted all the resources created.

## End Lab

1. Sign out of the AWS Account
2. You have successfully completed the lab.
3. Click on **End Lab** button from Whizlabs Labs console and wait till the process gets completed.

