



HashiCorp

# Terraform

Write, Plan, and Create Infrastructure as Code

GET STARTED

DOWNLOAD 0.10.7



## Provision any infrastructure for any application

Terraform provides a consistent workflow for operators to provision infrastructure across public cloud, private cloud, and external services.

Request Enterprise Demo

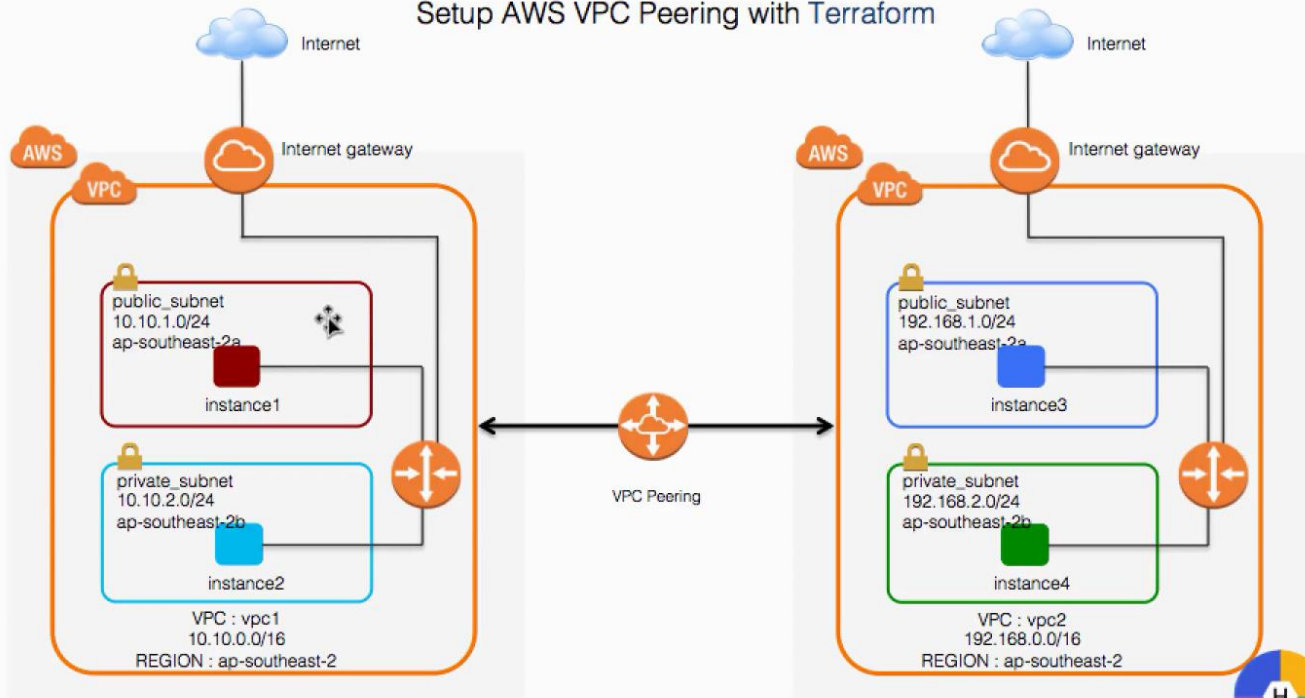
Download Open Source



Source: terraform.io

HashiCorp

## Setup AWS VPC Peering with Terraform



## Sample Terraform Code

### vpc\_peering\_project.tf

```
provider "aws" {  
  region          = "ap-southeast-2"  
  shared_credentials_file = "/Users/hellocloud/.aws/credentials"  
  profile         = "terraform"  
}  
  
resource "aws_vpc" "vpc1" {  
  cidr_block = "10.10.0.0/16"  
  enable_dns_support = true  
  enable_dns_hostnames = true  
  tags = { Name = "vpc1" }  
}  
  
resource "aws_vpc" "vpc2" {  
  cidr_block = "192.168.0.0/16"  
  enable_dns_support = true  
  enable_dns_hostnames = true  
  tags = { Name = "vpc2" }  
}
```



# Resource Providers > 60+



Source : <https://www.terraform.io/docs/providers/index.html>

## WRITE

### INFRASTRUCTURE AS CODE



## PLAN

### PREVIEW CHANGES BEFORE APPLYING

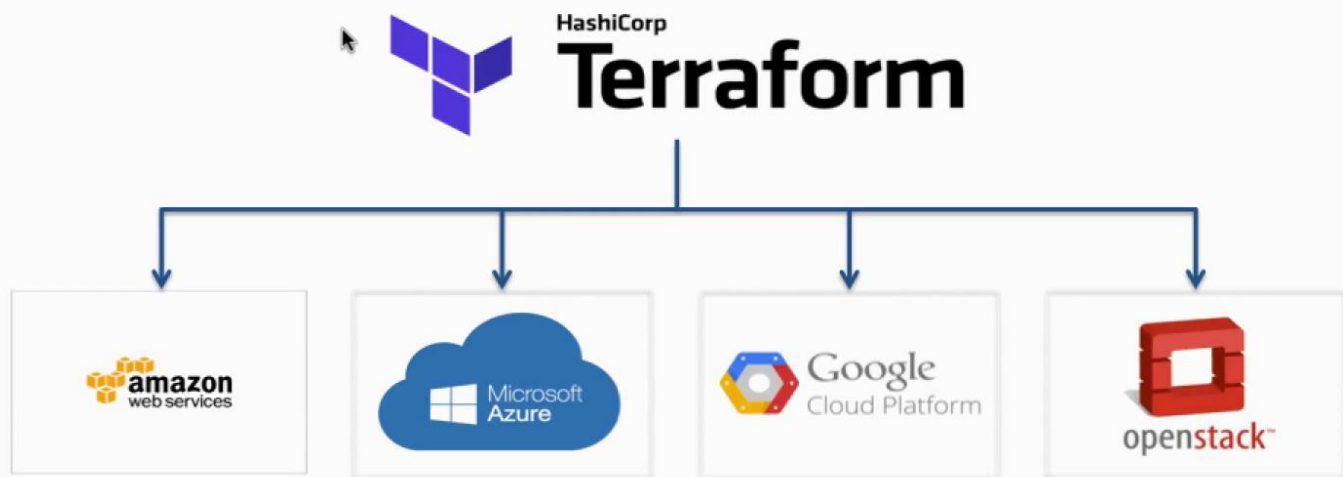


## CREATE

### REPRODUCIBLE INFRASTRUCTURE



# Use Terraform to manage Multi Clouds



## Shape your infrastructure with Terraform

Robert Firek  
**codurance**  
craft at heart



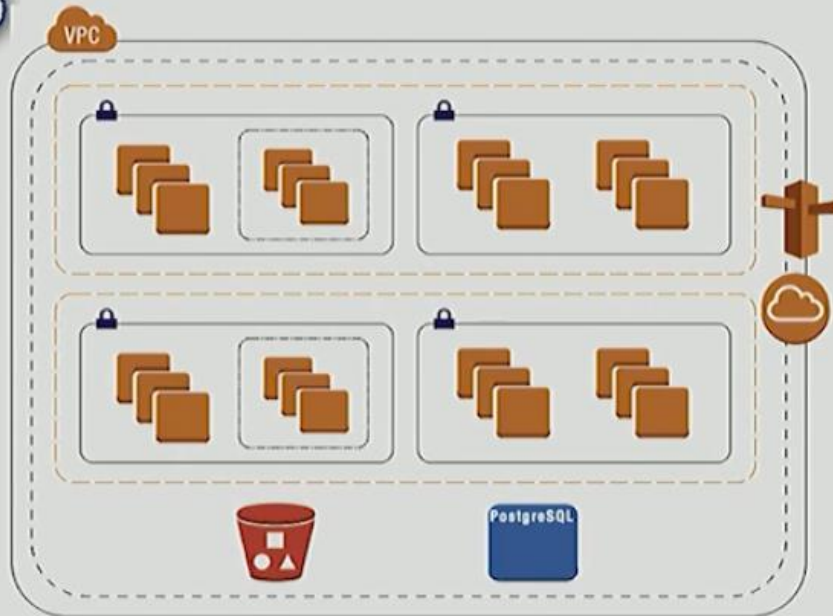
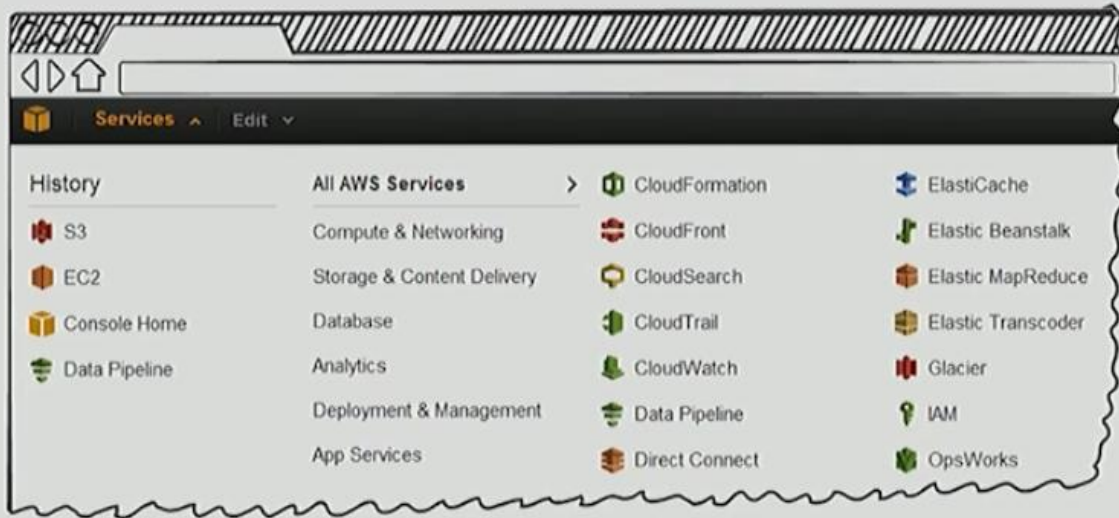
IaaS



Google Cloud Platform

CLOUD **F**OUNDRY





You need to automate this entire process





ANSIBLE



**BASH**  
THE BOURNE-AGAIN SHELL



**CloudFormation**

You can use **Bash** or **Ansible** with the **AWS CLI**, or use **CloudFormation** using **JSON** files that describe our infrastructure



 **HashiCorp**

**Terraform** allows you to describe your infrastructure as code easily and versionable in a declarative manner using APIs.

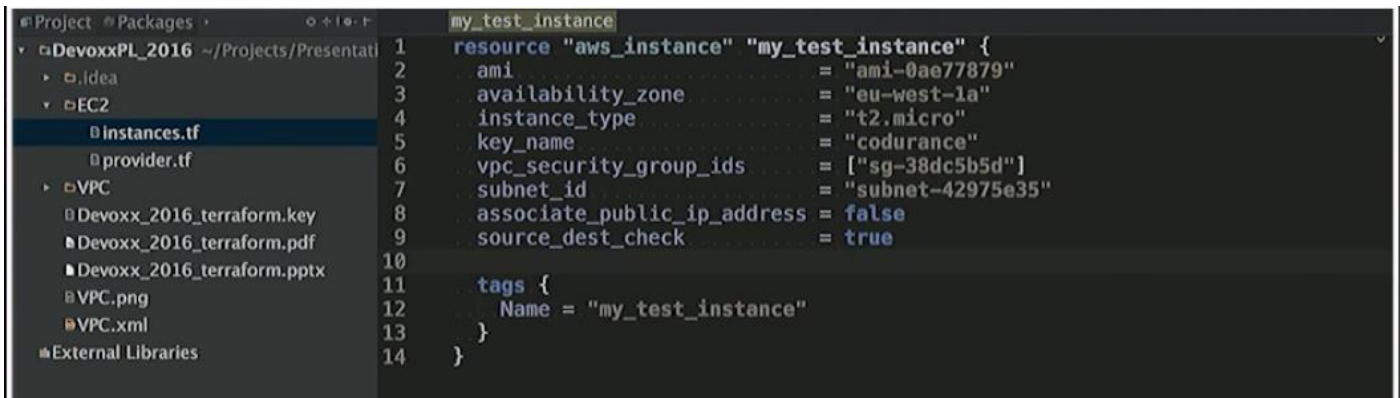
Hands-on session



```
1 provider "aws" {
2     region = "eu-west-1"
3     profile = "shape_your_infrastructure"
4 }
```

The screenshot shows an IDE with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'DevoxxPL\_2016' with a subdirectory 'EC2' containing 'instances.tf' and 'provider.tf'. The 'provider.tf' file is selected and its content is shown in the code editor. The code defines the AWS provider with the region 'eu-west-1' and the profile 'shape\_your\_infrastructure'.

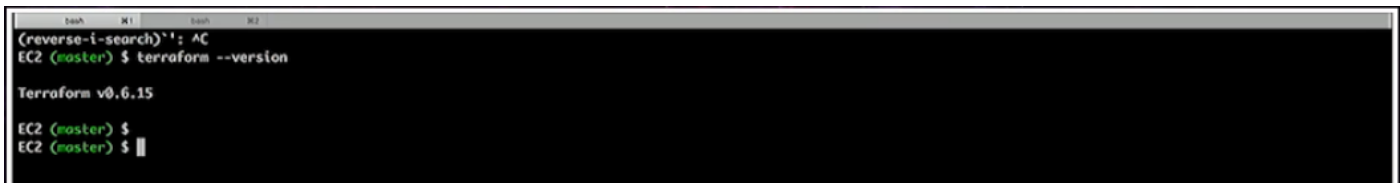
With **Terraform**, we need to specify how we want to connect to our cloud provider using a **provider.tf** file as above. Terraform will use the credentials on your local machine and the provider.tf file to connect to AWS and the specific region.



```
1 resource "aws_instance" "my_test_instance" {
2     ami           = "ami-0ae77879"
3     availability_zone = "eu-west-1a"
4     instance_type    = "t2.micro"
5     key_name         = "codurance"
6     vpc_security_group_ids = ["sg-38dc5b5d"]
7     subnet_id        = "subnet-42975e35"
8     associate_public_ip_address = false
9     source_dest_check = true
10
11     tags {
12         Name = "my_test_instance"
13     }
14 }
```

The screenshot shows the same IDE with the 'instances.tf' file selected in the file explorer. The code editor shows the configuration for an 'aws\_instance' resource named 'my\_test\_instance'. The configuration includes attributes for ami, availability zone, instance type, key name, security groups, subnet, and tags.

You then need to define what resource you want to create. This file will create an EC2 instance resource for us. This is the simplest form of terraform resource.



```
(reverse-i-search)~: AC
EC2 (master) $ terraform --version

Terraform v0.6.15
EC2 (master) $
EC2 (master) $
```

The screenshot shows a terminal window with the command 'terraform --version' being executed. The output is 'Terraform v0.6.15'. The prompt is 'EC2 (master) \$'.

You can run the above **\$ terraform - -version** command after installing Terraform on your local machine.



```

EC2 (master) $
EC2 (master) $ terraform plan
Refreshing Terraform state prior to plan...

The Terraform execution plan has been generated and is shown below.
Resources are shown in alphabetical order for quick scanning. Green resources
will be created (or destroyed and then created if an existing resource
exists), yellow resources are being changed in-place, and red resources
will be destroyed.

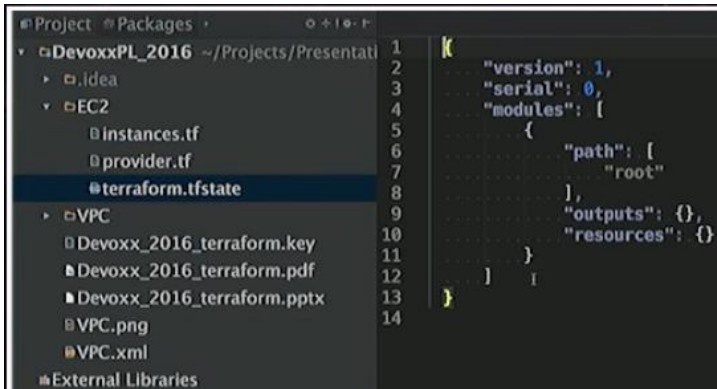
Note: You didn't specify an "-out" parameter to save this plan, so when
"apply" is called, Terraform can't guarantee this is what will execute.

+ aws_instance.my_test_instance
  ami:                               "" => "ami-0ae77879"
  associate_public_ip_address:       "" => "0"
  availability_zone:                  "" => "eu-west-1a"
  ebs_block_device.#:                 "" => "<computed>"
  ephemeral_block_device.#:           "" => "<computed>"
  instance_state:                     "" => "<computed>"
  instance_type:                      "" => "t2.micro"
  key_name:                           "" => "codurance"
  placement_group:                    "" => "<computed>"
  private_dns:                        "" => "<computed>"
  private_ip:                         "" => "<computed>"
  public_dns:                         "" => "<computed>"
  public_ip:                          "" => "<computed>"
  root_block_device.#:                "" => "<computed>"
  security_groups.#:                  "" => "<computed>"
  source_dest_check:                  "" => "1"
  subnet_id:                          "" => "subnet-42975e35"
  tags.#:                             "" => "1"
  tags.Name:                          "" => "my_test_instance"
  tenancy:                            "" => "<computed>"
  vpc_security_group_ids.#:            "" => "1"
  vpc_security_group_ids.4122177715: "" => "sg-38dc5b5d"

Plan: 1 to add, 0 to change, 0 to destroy.
EC2 (master) $

```

The **\$ terraform plan** commit does not create any resource for us but shows us what it will perform when executing the command.



The screenshot shows a file explorer on the left with the following structure:

- Project
  - Packages
    - DevoxxPL\_2016 ~/Projects/Presentati
      - idea
      - EC2
        - instances.tf
        - provider.tf
        - terraform.tfstate**
      - VPC
        - Devoxx\_2016\_terraform.key
        - Devoxx\_2016\_terraform.pdf
        - Devoxx\_2016\_terraform.pptx
        - VPC.png
        - VPC.xml
      - External Libraries

The right pane shows the content of the **terraform.tfstate** file:

```

1 {
2   "version": 1,
3   "serial": 0,
4   "modules": [
5     {
6       "path": [
7         "root"
8       ],
9       "outputs": {},
10      "resources": {}
11    }
12  ]
13 }
14

```

The **terraform.tfstate** file describes what we have so far in our infrastructure, this is empty at the moment

```

(reverse-i-search)`: AC
EC2 (master) $ terraform apply
aws_instance.my_test_instance: Creating...
ami:                               ==> "ami-0ae77879"
associate_public_ip_address:      ==> "0"
availability_zone:                 ==> "eu-west-1a"
ebs_block_device.#:               ==> "0"
ephemeral_block_device.#:         ==> "0"
instance_state:                   ==> "pending"
instance_type:                    ==> "t2.micro"
key_name:                          ==> "codurance"
placement_group:                  ==> "0"
private_dns:                      ==> "0"
private_ip:                       ==> "0"
public_dns:                       ==> "0"
public_ip:                        ==> "0"
root_block_device.#:              ==> "1"
security_groups.#:                ==> "1"
source_dest_check:                ==> "1"
subnet_id:                        ==> "subnet-42975e35"
tags.#:                           ==> "1"
tags.Name:                        ==> "my_test_instance"
tenancy:                          ==> "default"
vpc_security_group_ids.#:         ==> "1"
vpc_security_group_ids.4122177715: ==> "sg-38dc5b5d"
aws_instance.my_test_instance: Still creating... (10s elapsed)
aws_instance.my_test_instance: Still creating... (20s elapsed)
aws_instance.my_test_instance: Creation complete

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

The state of your infrastructure has been saved to the path
below. This state is required to modify and destroy your
infrastructure, so keep it safe. To inspect the complete state
use the 'terraform show' command.

State path: terraform.tfstate
EC2 (master) $

```

We then can use the **\$ terraform apply** command to create the infrastructure in AWS

The screenshot shows the AWS Management Console for the 'eu-west-1' region. The 'Launch Instance' button is highlighted. Below it, a table lists the instance 'my\_test\_instance' with the following details:

Name	Host Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Public IP	Monitoring
my_test_instance		i-52d1b77	t2.micro	eu-west-1a	running	Initializing	52.208.146.162	disabled

Below the table, the 'Description' tab is selected, showing the following details for instance 'i-52d1b77':

Property	Value
Instance ID	i-52d1b77
Instance state	running
Instance type	t2.micro
Private DNS	ip-172-31-38-159.eu-west-1.compute.internal
Private IP	172.31.38.159
Secondary private IP	
VPC ID	vpc-03a95266
Subnet ID	subnet-42975e35
Public DNS	ec2-52-208-146-162.eu-west-1.compute.amazonaws.com
Public IP	52.208.146.162
Elastic IP	-
Availability zone	eu-west-1a
Security groups	launch-wizard-1, view rules
Scheduled events	No scheduled events
AMI ID	ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20160610 [ami-0ae77879]
Platform	-

We now have the instance running in AWS

```
Project Packages
DevovxPL_2016 ~/Projects/Presentati
  .idea
  EC2
    instances.tf
    provider.tf
    terraform.tfstate
    terraform.tfstate.backup
  VPC
    Devovx_2016_terraform.key
    Devovx_2016_terraform.pdf
    Devovx_2016_terraform.pptx
    VPC.png
    VPC.xml
  External Libraries

1 {
2   "version": 1,
3   "serial": 1,
4   "modules": [
5     {
6       "path": [
7         "root"
8       ],
9       "outputs": {},
10      "resources": {
11        "aws_instance.my_test_instance": {
12          "type": "aws_instance",
13          "primary": {
14            "id": "i-fd2dlb77",
15            "attributes": {
16              "ami": "ami-0ae77879",
17              "associate_public_ip_address": "false",
18              "availability_zone": "eu-west-1a",
19              "disable_api_termination": "false",
20              "ebs_block_device.#": "0",
21              "ebs_optimized": "false",
22              "ephemeral_block_device.#": "0",
23              "iam_instance_profile": "",
24              "id": "i-fd2dlb77",
25              "instance_state": "running",
26              "instance_type": "t2.micro",
27              "key_name": "codurance",
28              "monitoring": "false",
29              "private_dns": "ip-172-31-38-159.eu-west-1.compute.internal",
30              "private_ip": "172.31.38.159",
31              "public_dns": "ec2-52-208-146-162.eu-west-1.compute.amazonaws.com",
32              "public_ip": "52.208.146.162",
33              "root_block_device.#": "1",
34              "root_block_device.0.delete_on_termination": "true",
35              "root_block_device.0.iops": "100",
```

```
Project Packages
DevovxPL_2016 ~/Projects/Presentati
  .idea
  EC2
    instances.tf
    provider.tf
    terraform.tfstate
    terraform.tfstate.backup
  VPC
    Devovx_2016_terraform.key
    Devovx_2016_terraform.pdf
    Devovx_2016_terraform.pptx
    VPC.png
    VPC.xml
  External Libraries

my_test_instance
1 resource "aws_instance" "my_test_instance" {
2   ami           = "ami-0ae77879"
3   availability_zone = "eu-west-1a"
4   instance_type   = "t2.micro"
5   key_name        = "codurance"
6   vpc_security_group_ids = ["sg-38dc5b5d"]
7   subnet_id       = "subnet-42975e35"
8   associate_public_ip_address = false
9   source_dest_check = true
10 }
11 tags {
12   Name = "my_test_instance"
13 }
14 }
```

We can increase the number of instance by changing the file above

```
Project Packages
DevovxPL_2016 ~/Projects/Presentati
  .idea
  EC2
    instances.tf
    provider.tf
    terraform.tfstate
    terraform.tfstate.backup
  VPC
    Devovx_2016_terraform.key
    Devovx_2016_terraform.pdf
    Devovx_2016_terraform.pptx
    VPC.png
    VPC.xml
  External Libraries

my_test_instance tags Name
1 resource "aws_instance" "my_test_instance" {
2   ami           = "ami-0ae77879"
3   availability_zone = "eu-west-1a"
4   instance_type   = "t2.micro"
5   key_name        = "codurance"
6   vpc_security_group_ids = ["sg-38dc5b5d"]
7   subnet_id       = "subnet-42975e35"
8   associate_public_ip_address = false
9   source_dest_check = true
10   count          = 3
11   tags {
12     Name = "my_test_instance ${count.index}"
13   }
14 }
```

```

terramark M1 bash M2
EC2 (master) $ terraform apply
aws_instance.my_test_instance: Creating...
ami: "" => "ami-0ae77879"
associate_public_ip_address: "" => "0"
availability_zone: "" => "eu-west-1a"
ebs_block_device.#: "" => "<computed>"
ephemeral_block_device.#: "" => "<computed>"
instance_state: "" => "<computed>"
instance_type: "" => "t2.micro"
key_name: "" => "codurance"
placement_group: "" => "<computed>"
private_dns: "" => "<computed>"
private_ip: "" => "<computed>"
public_dns: "" => "<computed>"
public_ip: "" => "<computed>"
root_block_device.#: "" => "<computed>"
security_groups.#: "" => "<computed>"
source_dest_check: "" => "1"
subnet_id: "" => "subnet-42975e35"
tags.#: "" => "1"
tags.Name: "" => "my_test_instance"
tenancy: "" => "<computed>"
vpc_security_group_ids.#: "" => "1"
vpc_security_group_ids.4122177715: "" => "sg-38dc5b5d"
aws_instance.my_test_instance: Still creating... (10s elapsed)
aws_instance.my_test_instance: Still creating... (20s elapsed)
aws_instance.my_test_instance: Creation complete

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

The state of your infrastructure has been saved to the path
below. This state is required to modify and destroy your
infrastructure, so keep it safe. To inspect the complete state
use the 'terraform show' command.

State path: terraform.tfstate
EC2 (master) $ terraform plan
Refreshing Terraform state prior to plan...

aws_instance.my_test_instance.0: Refreshing state... (ID: i-fd2d1b77)

```

We can now run **\$ terraform plan** command again to see what changes will be applied

```

bash M1 bash M2
Resources are shown in alphabetical order for quick scanning. Green resources
will be created (or destroyed and then created if an existing resource
exists), yellow resources are being changed in-place, and red resources
will be destroyed.

Note: You didn't specify an "-out" parameter to save this plan, so when
"apply" is called, Terraform can't guarantee this is what will execute.
}
~ aws_instance.my_test_instance.0
  tags.Name: "my_test_instance" => "my_test_instance 0"

+ aws_instance.my_test_instance.1
  ami: "" => "ami-0ae77879"
  associate_public_ip_address: "" => "0"
  availability_zone: "" => "eu-west-1a"
  ebs_block_device.#: "" => "<computed>"
  ephemeral_block_device.#: "" => "<computed>"
  instance_state: "" => "<computed>"
  instance_type: "" => "t2.micro"
  key_name: "" => "codurance"
  placement_group: "" => "<computed>"
  private_dns: "" => "<computed>"
  private_ip: "" => "<computed>"
  public_dns: "" => "<computed>"
  public_ip: "" => "<computed>"
  root_block_device.#: "" => "<computed>"
  security_groups.#: "" => "<computed>"
  source_dest_check: "" => "1"
  subnet_id: "" => "subnet-42975e35"
  tags.#: "" => "1"
  tags.Name: "" => "my_test_instance 1"
  tenancy: "" => "<computed>"
  vpc_security_group_ids.#: "" => "1"
  vpc_security_group_ids.4122177715: "" => "sg-38dc5b5d"

+ aws_instance.my_test_instance.2
  ami: "" => "ami-0ae77879"
  associate_public_ip_address: "" => "0"
  availability_zone: "" => "eu-west-1a"
  ebs_block_device.#: "" => "<computed>"
  ephemeral_block_device.#: "" => "<computed>"
  instance_state: "" => "<computed>"

```



```

private_ip:      "" => "computed"
public_dns:      "" => "computed"
public_ip:      "" => "computed"
root_block_device.#: "" => "computed"
security_groups.#: "" => "computed"
source_dest_check: "" => "1"
subnet_id:      "" => "subnet-42975e35"
tags.#:         "" => "1"
tags.Name:      "" => "my_test_instance 1"
tenancy:        "" => "computed"
vpc_security_group_ids.#: "" => "1"
vpc_security_group_ids.4122177715: "" => "sg-38dc5b5d"

+ aws_instance.my_test_instance.2
ami:      "" => "ami-0ae77879"
associate_public_ip_address: "" => "0"
availability_zone: "" => "eu-west-1a"
ebs_block_device.#: "" => "computed"
ephemeral_block_device.#: "" => "computed"
instance_state:      "" => "computed"
instance_type:      "" => "t2.micro"
key_name:      "" => "codurance"
placement_group:      "" => "computed"
private_dns:      "" => "computed"
private_ip:      "" => "computed"
public_dns:      "" => "computed"
public_ip:      "" => "computed"
root_block_device.#: "" => "computed"
security_groups.#: "" => "computed"
source_dest_check: "" => "1"
subnet_id:      "" => "subnet-42975e35"
tags.#:         "" => "1"
tags.Name:      "" => "my_test_instance 2"
tenancy:        "" => "computed"
vpc_security_group_ids.#: "" => "1"
vpc_security_group_ids.4122177715: "" => "sg-38dc5b5d"

Plan: 2 to add, 1 to change, 0 to destroy.
EC2 (master) $ terraform apply
aws_instance.my_test_instance.0: Refreshing state... (ID: i-fd2d1b77)

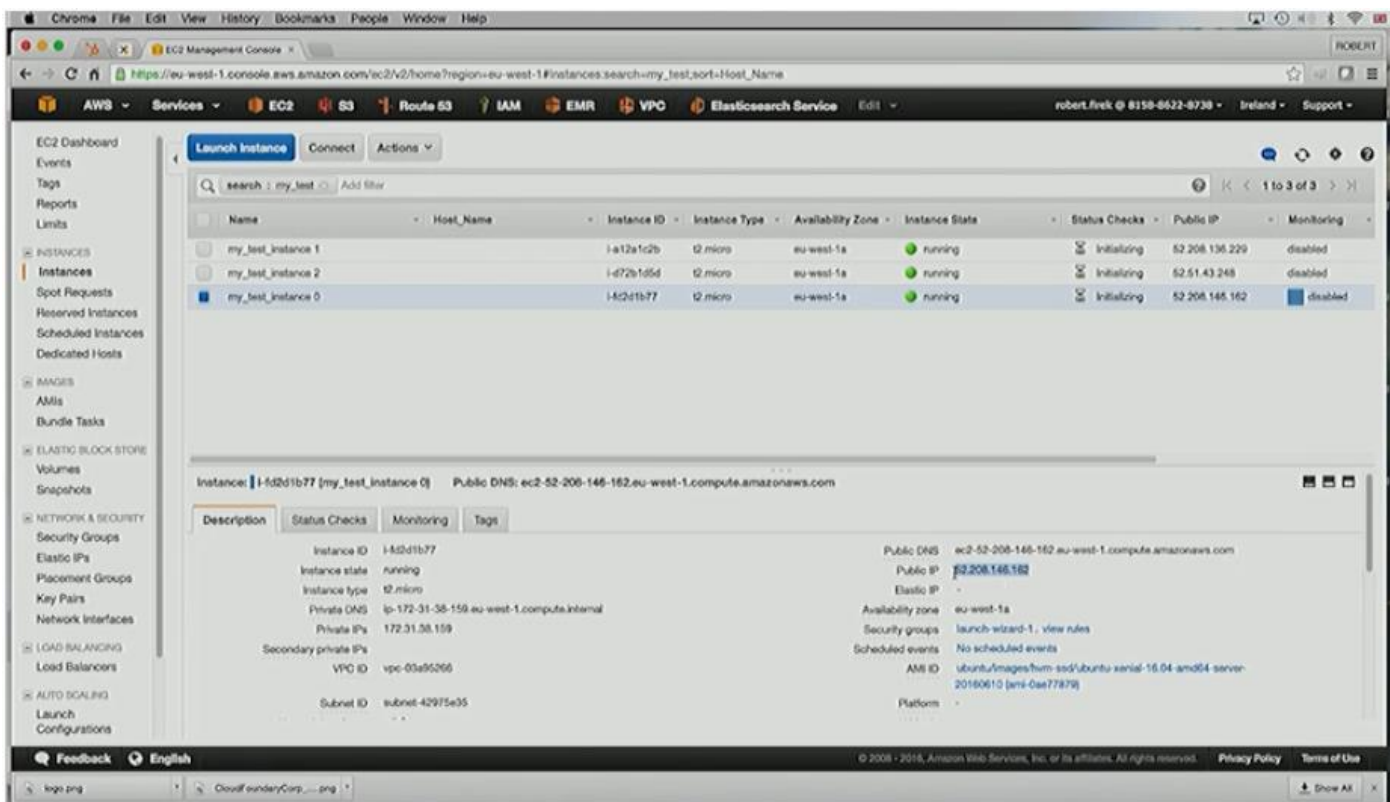
```

We can then run the **\$ terraform apply** command again

```

ephemeral_block_device.#: "" => "computed"
instance_state:      "" => "computed"
instance_type:      "" => "t2.micro"
key_name:      "" => "codurance"
placement_group:      "" => "computed"
private_dns:      "" => "computed"
private_ip:      "" => "computed"
public_dns:      "" => "computed"
public_ip:      "" => "computed"
root_block_device.#: "" => "computed"
security_groups.#: "" => "computed"
source_dest_check: "" => "1"
subnet_id:      "" => "subnet-42975e35"
tags.#:         "" => "1"
tags.Name:      "" => "my_test_instance 1"
tenancy:        "" => "computed"
vpc_security_group_ids.#: "" => "1"
vpc_security_group_ids.4122177715: "" => "sg-38dc5b5d"
aws_instance.my_test_instance.2: Creating...
ami:      "" => "ami-0ae77879"
associate_public_ip_address: "" => "0"
availability_zone: "" => "eu-west-1a"
ebs_block_device.#: "" => "computed"
ephemeral_block_device.#: "" => "computed"
instance_state:      "" => "computed"
instance_type:      "" => "t2.micro"
key_name:      "" => "codurance"
placement_group:      "" => "computed"
private_dns:      "" => "computed"
private_ip:      "" => "computed"
public_dns:      "" => "computed"
public_ip:      "" => "computed"
root_block_device.#: "" => "computed"
security_groups.#: "" => "computed"
source_dest_check: "" => "1"
subnet_id:      "" => "subnet-42975e35"
tags.#:         "" => "1"
tags.Name:      "" => "my_test_instance 2"
tenancy:        "" => "computed"
vpc_security_group_ids.#: "" => "1"
vpc_security_group_ids.4122177715: "" => "sg-38dc5b5d"

```



We now have 3 instances running

```

EC2 (master) $ terraform destroy
Do you really want to destroy?
Terraform will delete all your managed infrastructure.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_instance.my_test_instance.0: Refreshing state... (ID: i-fd2d1b77)
aws_instance.my_test_instance.2: Refreshing state... (ID: i-d72b1d5d)
aws_instance.my_test_instance.1: Refreshing state... (ID: i-a12a1c2b)
aws_instance.my_test_instance.2: Destroying...
aws_instance.my_test_instance.1: Destroying...
aws_instance.my_test_instance.0: Destroying...
aws_instance.my_test_instance.2: Still destroying... (10s elapsed)
aws_instance.my_test_instance.1: Still destroying... (10s elapsed)
aws_instance.my_test_instance.0: Still destroying... (10s elapsed)
aws_instance.my_test_instance.2: Still destroying... (20s elapsed)
aws_instance.my_test_instance.1: Still destroying... (20s elapsed)
aws_instance.my_test_instance.0: Still destroying... (20s elapsed)
aws_instance.my_test_instance.1: Still destroying... (30s elapsed)
aws_instance.my_test_instance.2: Still destroying... (30s elapsed)
aws_instance.my_test_instance.0: Still destroying... (30s elapsed)
aws_instance.my_test_instance.1: Still destroying... (40s elapsed)
aws_instance.my_test_instance.0: Still destroying... (40s elapsed)
aws_instance.my_test_instance.2: Still destroying... (40s elapsed)
aws_instance.my_test_instance.1: Still destroying... (50s elapsed)
aws_instance.my_test_instance.0: Still destroying... (50s elapsed)
aws_instance.my_test_instance.1: Destruction complete
aws_instance.my_test_instance.2: Destruction complete
aws_instance.my_test_instance.0: Still destroying... (1m0s elapsed)
aws_instance.my_test_instance.0: Destruction complete

Apply complete! Resources: 0 added, 0 changed, 3 destroyed.
EC2 (master) $ cd

```

We can use the **\$ terraform destroy** command to remove the infrastructure as above



Chrome File Edit View History Bookmarks People Window Help

EC2 Management Console

https://eu-west-1.console.aws.amazon.com/ec2/v2/home?region=eu-west-1#Instances:search=my\_test,sort=Host\_Name

AWS Services EC2 S3 Route 53 IAM EMR VPC Elasticsearch Service

robert.freek @ 8158-8622-8738 Ireland Support

EC2 Dashboard Events Tags Reports Limits

INSTANCES

Instances

Spot Requests Reserved Instances Scheduled Instances Dedicated Hosts

IMAGES

AMIs Bundle Tasks

ELASTIC BLOCK STORE

Volumes Snapshots

NETWORK & SECURITY

Security Groups Elastic IPs Placement Groups Key Pairs Network Interfaces

LOAD BALANCING

Load Balancers

AUTO SCALING

Launch Configurations

Launch Instance Connect Actions

search my\_test Add filter

1 to 3 of 3

Name	Host Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Public IP	Monitoring
my_test_instance 1		i-a12a1c2b	t2.micro	eu-west-1a	shutting-down			disabled
my_test_instance 2		i-d72b1d5d	t2.micro	eu-west-1a	shutting-down			disabled
my_test_instance 0		i-fd2d1b77	t2.micro	eu-west-1a	shutting-down			disabled

Instance: i-fd2d1b77 [my\_test\_instance 0] Public DNS: -

Description Status Checks Monitoring Tags

Instance ID i-fd2d1b77 Public DNS -

Instance state shutting-down Public IP -

Instance type t2.micro Elastic IP -

Private DNS - Availability zone eu-west-1a

Private IPs - Security groups -

Secondary private IPs - Scheduled events -

VPC ID - AMI ID ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20160610 [ami-0ae77879]

Subnet ID - Platform -

Feedback English

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Chrome File Edit View History Bookmarks People Window Help

EC2 Management Console

https://eu-west-1.console.aws.amazon.com/ec2/v2/home?region=eu-west-1#Instances:search=my\_test,sort=Host\_Name

AWS Services EC2 S3 Route 53 IAM EMR VPC Elasticsearch Service

robert.freek @ 8158-8622-8738 Ireland Support

EC2 Dashboard Events Tags Reports Limits

INSTANCES

Instances

Spot Requests Reserved Instances Scheduled Instances Dedicated Hosts

IMAGES

AMIs Bundle Tasks

ELASTIC BLOCK STORE

Volumes Snapshots

NETWORK & SECURITY

Security Groups Elastic IPs Placement Groups Key Pairs Network Interfaces

LOAD BALANCING

Load Balancers

AUTO SCALING

Launch Configurations

Launch Instance Connect Actions

search my\_test Add filter

1 to 3 of 3

Name	Host Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Public IP	Monitoring
my_test_instance 1		i-a12a1c2b	t2.micro	eu-west-1a	terminated			disabled
my_test_instance 2		i-d72b1d5d	t2.micro	eu-west-1a	terminated			disabled
my_test_instance 0		i-fd2d1b77	t2.micro	eu-west-1a	terminated			disabled

Instance: i-fd2d1b77 [my\_test\_instance 0] Public DNS: -

Description Status Checks Monitoring Tags

Instance ID i-fd2d1b77 Public DNS -

Instance state terminated Public IP -

Instance type t2.micro Elastic IP -

Private DNS - Availability zone eu-west-1a

Private IPs - Security groups -

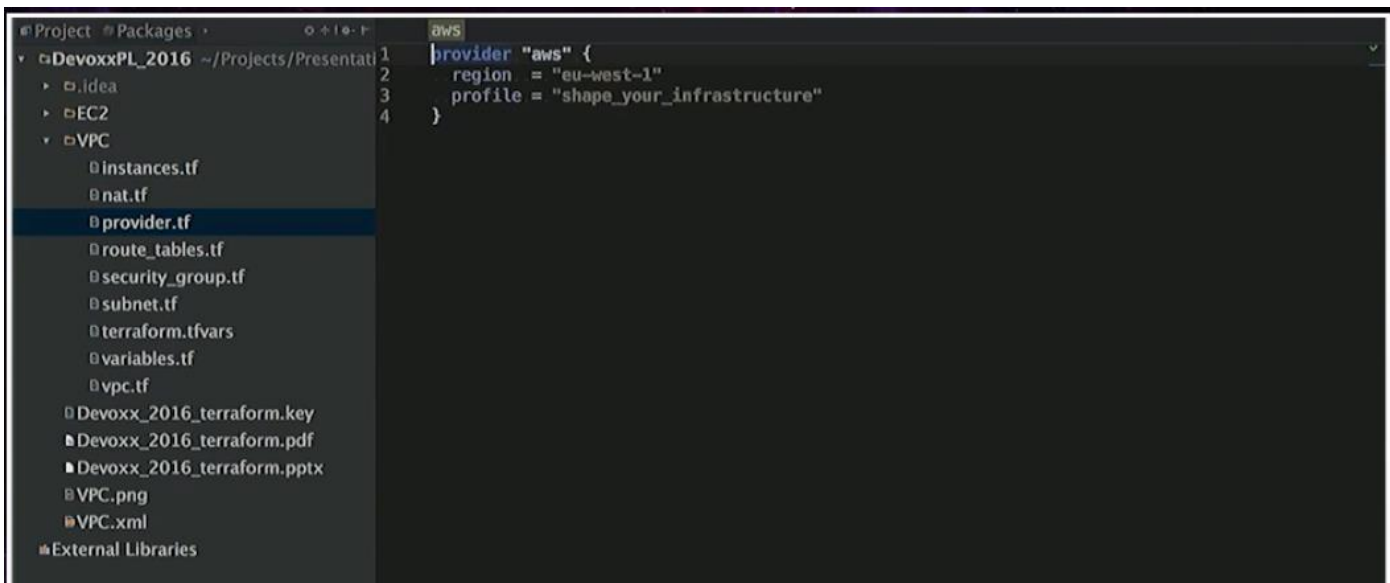
Secondary private IPs - Scheduled events -

VPC ID - AMI ID ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20160610 [ami-0ae77879]

Subnet ID - Platform -

Feedback English

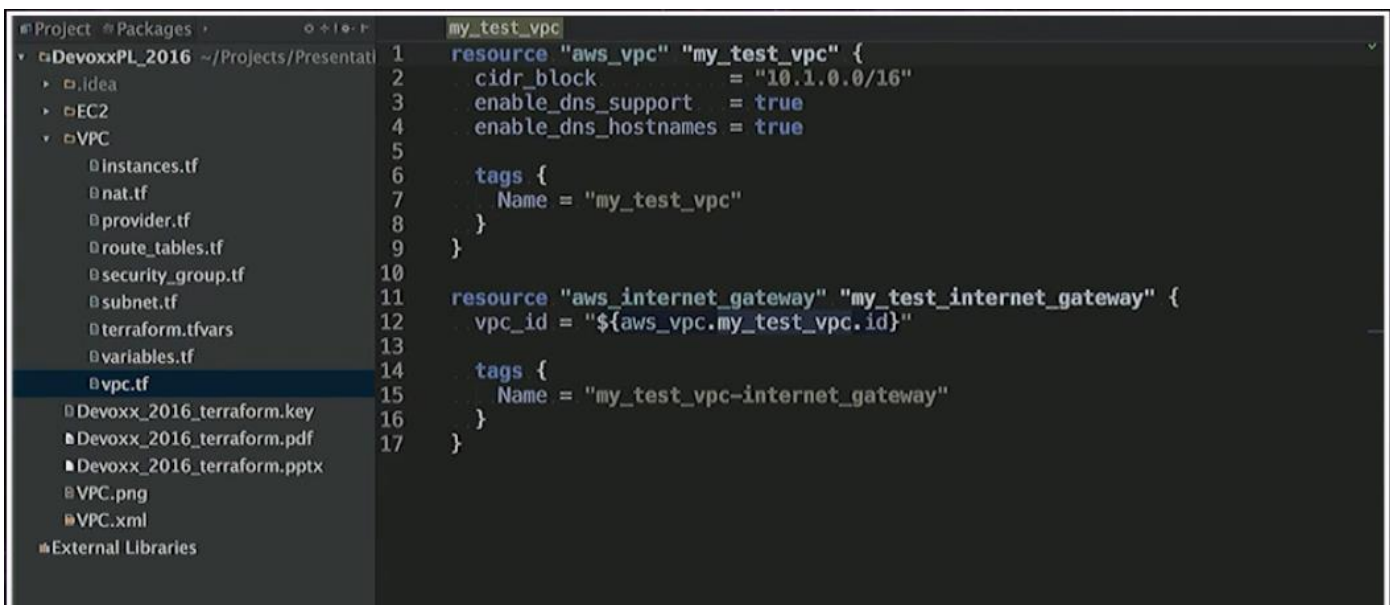
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



The screenshot shows an IDE with a file explorer on the left and a code editor on the right. The file explorer is expanded to show a directory structure for a project named 'DevoxxPL\_2016'. The 'VPC' directory is selected, and the 'provider.tf' file is highlighted. The code editor displays the contents of 'provider.tf', which is a Terraform configuration for the AWS provider. The configuration sets the provider to 'aws', the region to 'eu-west-1', and the profile to 'shape\_your\_infrastructure'.

```
1 provider "aws" {
2   region = "eu-west-1"
3   profile = "shape_your_infrastructure"
4 }
```

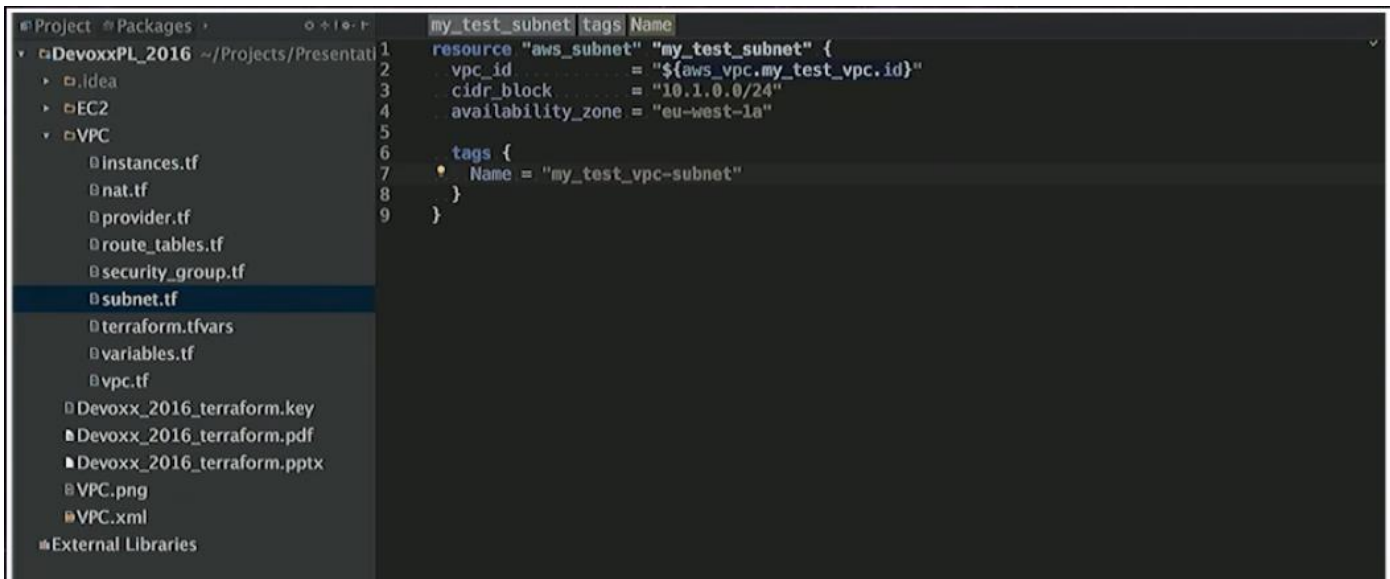
Let us look at a more sophisticated example of using Terraform to build out our AWS infrastructure like creating a VPC.



The screenshot shows the same IDE as before, but now the 'vpc.tf' file is selected in the file explorer. The code editor displays the contents of 'vpc.tf', which is a Terraform configuration for creating an AWS VPC and an Internet Gateway. The VPC configuration sets the cidr\_block to '10.1.0.0/16', enables dns\_support and dns\_hostnames, and tags it with 'my\_test\_vpc'. The Internet Gateway configuration sets the vpc\_id to the id of the VPC and tags it with 'my\_test\_vpc-internet\_gateway'.

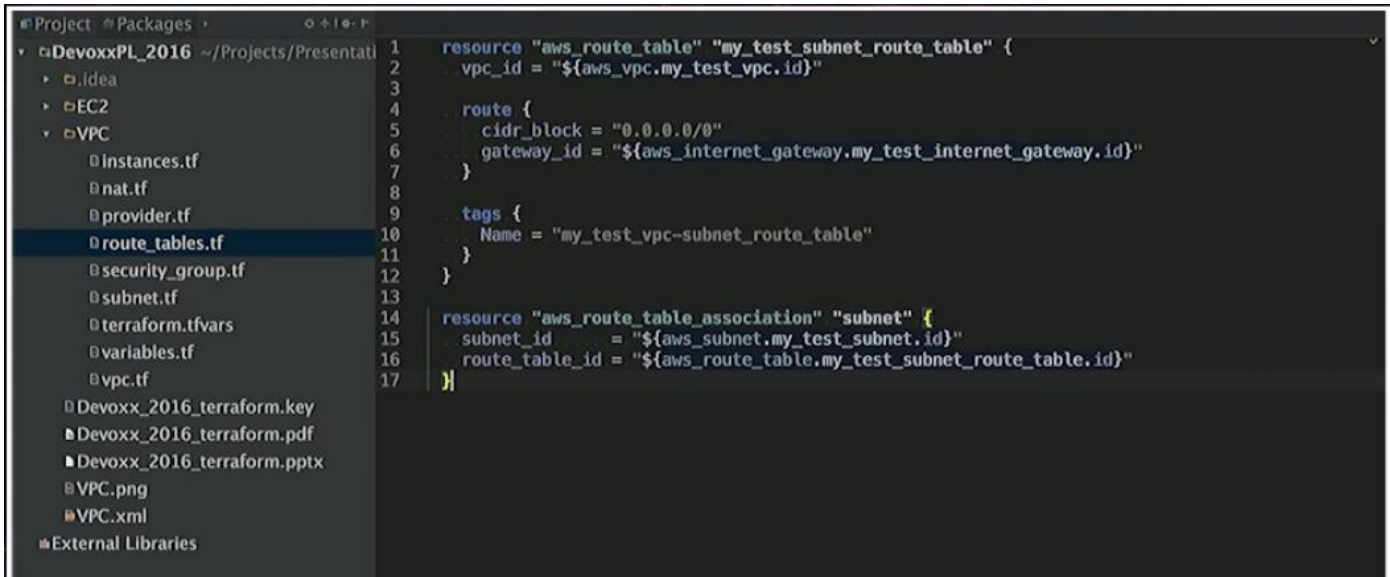
```
1 resource "aws_vpc" "my_test_vpc" {
2   cidr_block      = "10.1.0.0/16"
3   enable_dns_support = true
4   enable_dns_hostnames = true
5
6   tags {
7     Name = "my_test_vpc"
8   }
9 }
10
11 resource "aws_internet_gateway" "my_test_internet_gateway" {
12   vpc_id = "${aws_vpc.my_test_vpc.id}"
13
14   tags {
15     Name = "my_test_vpc-internet_gateway"
16   }
17 }
```

We can describe the VPC and resources within it that we want to create using the **vpc.tf** file above



```
1 resource "aws_subnet" "my_test_subnet" {
2   vpc_id = "${aws_vpc.my_test_vpc.id}"
3   cidr_block = "10.1.0.0/24"
4   availability_zone = "eu-west-1a"
5
6   tags {
7     Name = "my_test_vpc-subnet"
8   }
9 }
```

To describe the subnet in the **subnet.tf** file, we need to specify which VPC we want to create that subnet in and what route tables we want to use as below



```
1 resource "aws_route_table" "my_test_subnet_route_table" {
2   vpc_id = "${aws_vpc.my_test_vpc.id}"
3
4   route {
5     cidr_block = "0.0.0.0/0"
6     gateway_id = "${aws_internet_gateway.my_test_internet_gateway.id}"
7   }
8
9   tags {
10    Name = "my_test_vpc-subnet_route_table"
11  }
12 }
13
14 resource "aws_route_table_association" "subnet" {
15   subnet_id = "${aws_subnet.my_test_subnet.id}"
16   route_table_id = "${aws_route_table.my_test_subnet_route_table.id}"
17 }
```

We describe the route tables to use in the **route\_tables.tf** file above.

```
Project Packages
DevoxxPL_2016 ~/Projects/Presentation
├── .idea
├── EC2
├── VPC
│   ├── instances.tf
│   ├── nat.tf
│   ├── provider.tf
│   ├── route_tables.tf
│   ├── security_group.tf
│   ├── subnet.tf
│   ├── terraform.tfvars
│   ├── variables.tf
│   └── vpc.tf
├── Devoxx_2016_terraform.key
├── Devoxx_2016_terraform.pdf
├── Devoxx_2016_terraform.pptx
├── VPC.png
└── VPC.xml
External Libraries

security_group
1 resource "aws_security_group" "security_group" {
2   name = "my_test_vpc-security_group"
3
4   ingress {
5     from_port = 22
6     to_port   = 22
7     protocol  = "tcp"
8     cidr_blocks = ["0.0.0.0/0"]
9   }
10
11  ingress {
12    from_port = 80
13    to_port   = 80
14    protocol  = "tcp"
15    cidr_blocks = ["0.0.0.0/0"]
16  }
17
18  ingress {
19    from_port = 443
20    to_port   = 443
21    protocol  = "tcp"
22    cidr_blocks = ["0.0.0.0/0"]
23  }
24
25  egress {
26    from_port = 0
27    to_port   = 0
28    protocol  = "-1"
29    cidr_blocks = ["0.0.0.0/0"]
30  }
31
32  vpc_id = "${aws_vpc.my_test_vpc.id}"
33  tags {
34    Name = "my_test_vpc-security_group"
35  }
36 }
```

You also describe the security groups you want to create in the *security\_group.tf* file as above

```
Project Packages
DevoxxPL_2016 ~/Projects/Presentation
├── .idea
├── EC2
├── VPC
│   ├── instances.tf
│   ├── nat.tf
│   ├── provider.tf
│   ├── route_tables.tf
│   ├── security_group.tf
│   ├── subnet.tf
│   ├── terraform.tfvars
│   ├── variables.tf
│   └── vpc.tf
├── Devoxx_2016_terraform.key
├── Devoxx_2016_terraform.pdf
├── Devoxx_2016_terraform.pptx
├── VPC.png
└── VPC.xml
External Libraries

security_group
5   from_port = 22
6   to_port   = 22
7   protocol  = "tcp"
8   cidr_blocks = ["0.0.0.0/0"]
9 }
10
11 ingress {
12   from_port = 80
13   to_port   = 80
14   protocol  = "tcp"
15   cidr_blocks = ["0.0.0.0/0"]
16 }
17
18 ingress {
19   from_port = 443
20   to_port   = 443
21   protocol  = "tcp"
22   cidr_blocks = ["0.0.0.0/0"]
23 }
24
25 egress {
26   from_port = 0
27   to_port   = 0
28   protocol  = "-1"
29   cidr_blocks = ["0.0.0.0/0"]
30 }
31
32 vpc_id = "${aws_vpc.my_test_vpc.id}"
33 tags {
34   Name = "my_test_vpc-security_group"
35 }
36 }
```

```

1 resource "aws_eip" "my_test_nat_eip" {
2   vpc = true
3 }
4
5 resource "aws_nat_gateway" "my_test_nat" {
6   allocation_id = "${aws_eip.my_test_nat_eip.id}"
7   subnet_id = "${aws_subnet.my_test_subnet.id}"
8
9   depends_on = ["aws_internet_gateway.my_test_internet_gateway"]
10 }

```

You can also describe a NAT instance using the **nat.tf** file that will be used to connect instances in your VPC to the internet.

```

aws_instance.my_test_instance.0: Refreshing state... (ID: i-fd2d1b77)
aws_instance.my_test_instance.2: Refreshing state... (ID: i-d72b1d5d)
aws_instance.my_test_instance.1: Refreshing state... (ID: i-a12o1c2b)
aws_instance.my_test_instance.2: Destroying...
aws_instance.my_test_instance.1: Destroying...
aws_instance.my_test_instance.0: Destroying...
aws_instance.my_test_instance.2: Still destroying... (10s elapsed)
aws_instance.my_test_instance.1: Still destroying... (10s elapsed)
aws_instance.my_test_instance.0: Still destroying... (10s elapsed)
aws_instance.my_test_instance.2: Still destroying... (20s elapsed)
aws_instance.my_test_instance.1: Still destroying... (20s elapsed)
aws_instance.my_test_instance.0: Still destroying... (20s elapsed)
aws_instance.my_test_instance.1: Still destroying... (30s elapsed)
aws_instance.my_test_instance.2: Still destroying... (30s elapsed)
aws_instance.my_test_instance.0: Still destroying... (30s elapsed)
aws_instance.my_test_instance.1: Still destroying... (40s elapsed)
aws_instance.my_test_instance.0: Still destroying... (40s elapsed)
aws_instance.my_test_instance.2: Still destroying... (40s elapsed)
aws_instance.my_test_instance.2: Still destroying... (50s elapsed)
aws_instance.my_test_instance.1: Still destroying... (50s elapsed)
aws_instance.my_test_instance.0: Still destroying... (50s elapsed)
aws_instance.my_test_instance.1: Destruction complete
aws_instance.my_test_instance.2: Destruction complete
aws_instance.my_test_instance.0: Still destroying... (1m0s elapsed)
aws_instance.my_test_instance.0: Destruction complete

Apply complete! Resources: 0 added, 0 changed, 3 destroyed.
EC2 (master) $ cd ../VPC
VPC (master) $ ll
total 64
-rw-r--r-- 1 robertfirek staff 82 20 Jun 19:37 provider.tf
-rw-r--r-- 1 robertfirek staff 200 20 Jun 19:57 subnet.tf
-rw-r--r-- 1 robertfirek staff 646 20 Jun 20:06 security_group.tf
-rw-r--r-- 1 robertfirek staff 447 20 Jun 20:11 route_tables.tf
-rw-r--r-- 1 robertfirek staff 489 20 Jun 20:37 instances.tf
-rw-r--r-- 1 robertfirek staff 263 20 Jun 20:43 nat.tf
-rw-r--r-- 1 robertfirek staff 48 20 Jun 20:54 variables.tf
-rw-r--r-- 1 robertfirek staff 0 20 Jun 20:54 terraform.tfvars
-rw-r--r-- 1 robertfirek staff 333 20 Jun 21:01 vpc.tf
drwxr-xr-x 11 robertfirek staff 374 20 Jun 21:04 .
drwxr-xr-x 12 robertfirek staff 408 23 Jun 16:04 ..
VPC (master) $

```

```

VPC (master) $ terraform plan
Refreshing Terraform state prior to plan...


```

We then run the **\$ terraform plan** command to see everything that will be created for the VPC



```

terraform 0.12.24 ubuntu@ip-172-31-1-103:~$ terraform plan
aws_vpc.my_test_vpc:
  ingress.2541437006.from_port: "" => "22"
  ingress.2541437006.protocol: "" => "tcp"
  ingress.2541437006.security_groups.#: "" => "0"
  ingress.2541437006.self: "" => "0"
  ingress.2541437006.to_port: "" => "22"
  ingress.2617001939.cidr_blocks.#: "" => "1"
  ingress.2617001939.cidr_blocks.0: "" => "0.0.0.0/0"
  ingress.2617001939.from_port: "" => "443"
  ingress.2617001939.protocol: "" => "tcp"
  ingress.2617001939.security_groups.#: "" => "0"
  ingress.2617001939.self: "" => "0"
  ingress.2617001939.to_port: "" => "443"
  name: "" => "my_test_vpc-security_group"
  owner_id: "" => "<computed>"
  tags.#: "" => "1"
  tags.Name: "" => "my_test_vpc-security_group"
  vpc_id: "" => "${aws_vpc.my_test_vpc.id}"

+ aws_subnet.my_test_subnet
  availability_zone: "" => "eu-west-1a"
  cidr_block: "" => "10.1.0.0/24"
  map_public_ip_on_launch: "" => "0"
  tags.#: "" => "1"
  tags.Name: "" => "my_test_vpc-subnet"
  vpc_id: "" => "${aws_vpc.my_test_vpc.id}"

+ aws_vpc.my_test_vpc
  cidr_block: "" => "10.1.0.0/16"
  default_network_acl_id: "" => "<computed>"
  default_security_group_id: "" => "<computed>"
  dhcp_options_id: "" => "<computed>"
  enable_classiclink: "" => "<computed>"
  enable_dns_hostnames: "" => "1"
  enable_dns_support: "" => "1"
  instance_tenancy: "" => "<computed>"
  main_route_table_id: "" => "<computed>"
  tags.#: "" => "1"
  tags.Name: "" => "my_test_vpc"

Plan: 9 to add, 0 to change, 0 to destroy.
VPC (master) $

```

```

terraform 0.12.24 ubuntu@ip-172-31-1-103:~$ terraform plan
tags.Name: "" => "my_test_vpc-subnet"
vpc_id: "" => "${aws_vpc.my_test_vpc.id}"

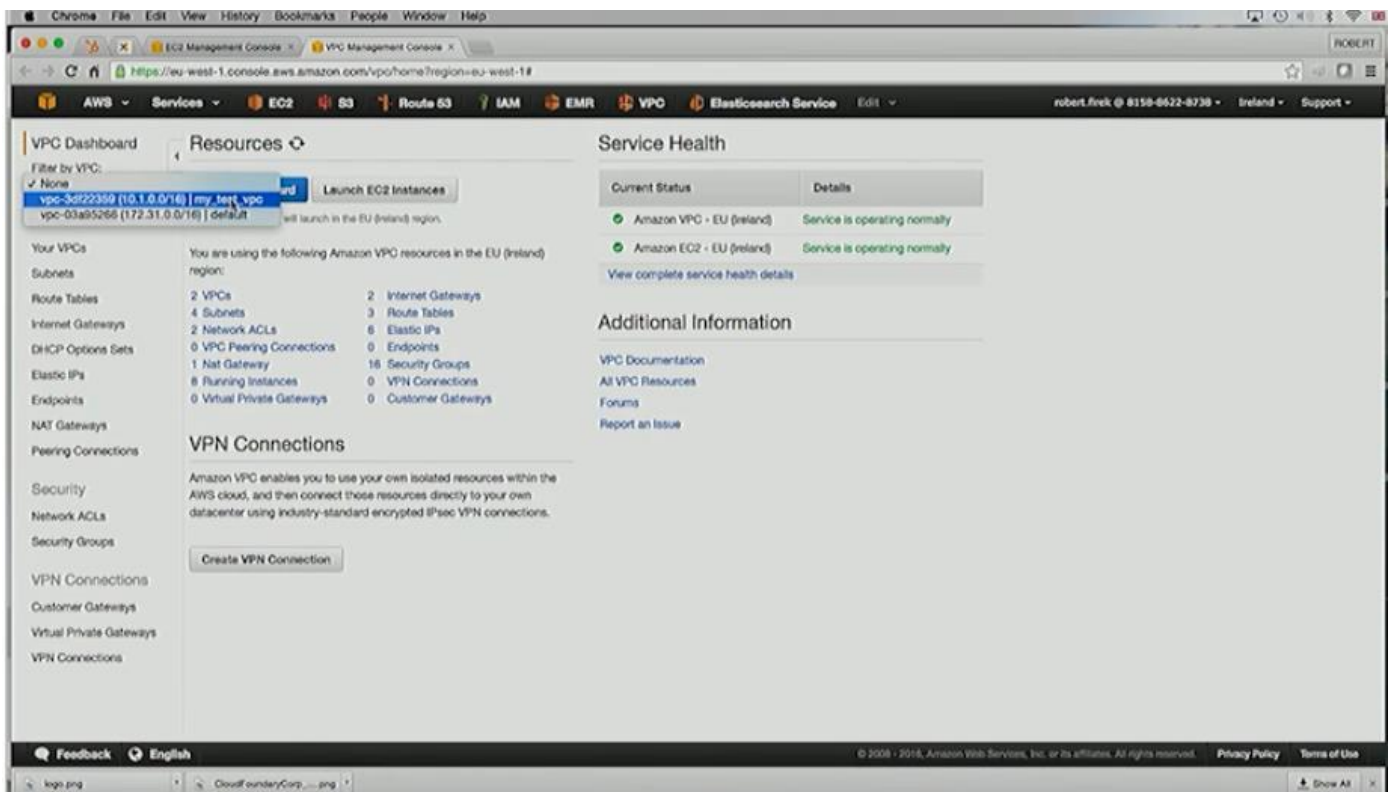
+ aws_vpc.my_test_vpc
  cidr_block: "" => "10.1.0.0/16"
  default_network_acl_id: "" => "<computed>"
  default_security_group_id: "" => "<computed>"
  dhcp_options_id: "" => "<computed>"
  enable_classiclink: "" => "<computed>"
  enable_dns_hostnames: "" => "1"
  enable_dns_support: "" => "1"
  instance_tenancy: "" => "<computed>"
  main_route_table_id: "" => "<computed>"
  tags.#: "" => "1"
  tags.Name: "" => "my_test_vpc"

Plan: 9 to add, 0 to change, 0 to destroy.
VPC (master) $ terraform apply
aws_vpc.my_test_vpc: Creating...
cidr_block: "" => "10.1.0.0/16"
default_network_acl_id: "" => "<computed>"
default_security_group_id: "" => "<computed>"
dhcp_options_id: "" => "<computed>"
enable_classiclink: "" => "<computed>"
enable_dns_hostnames: "" => "1"
enable_dns_support: "" => "1"
instance_tenancy: "" => "<computed>"
main_route_table_id: "" => "<computed>"
tags.#: "" => "1"
tags.Name: "" => "my_test_vpc"
aws_elb.my_test_elb: Creating...
allocation_id: "" => "<computed>"
association_id: "" => "<computed>"
domain: "" => "<computed>"
instance: "" => "<computed>"
network_interface: "" => "<computed>"
private_ip: "" => "<computed>"
public_ip: "" => "<computed>"
vpc: "" => "1"
aws_elb.my_test_elb: Creation complete

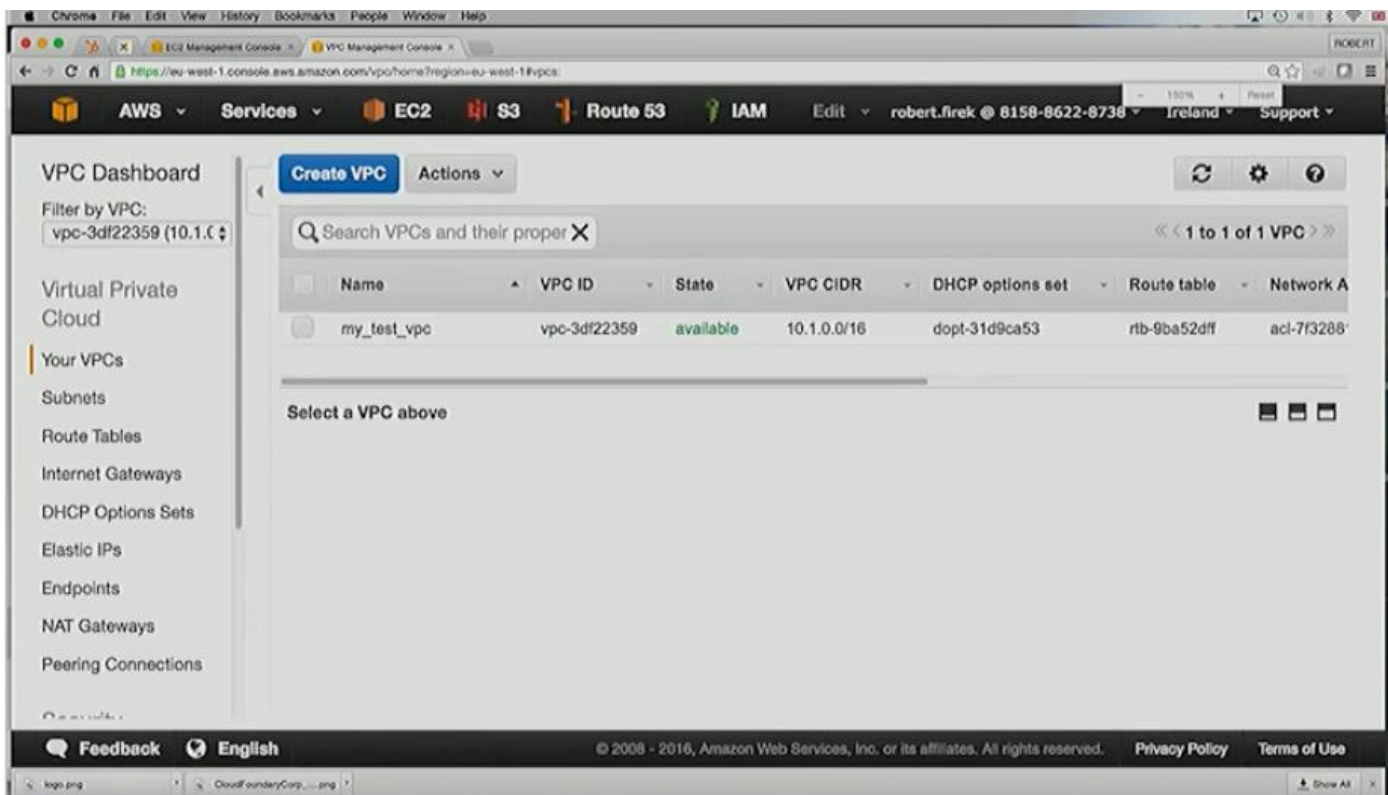
```

We then run the **\$ terraform apply** command to start creating the VPC





This is the new VPC being created



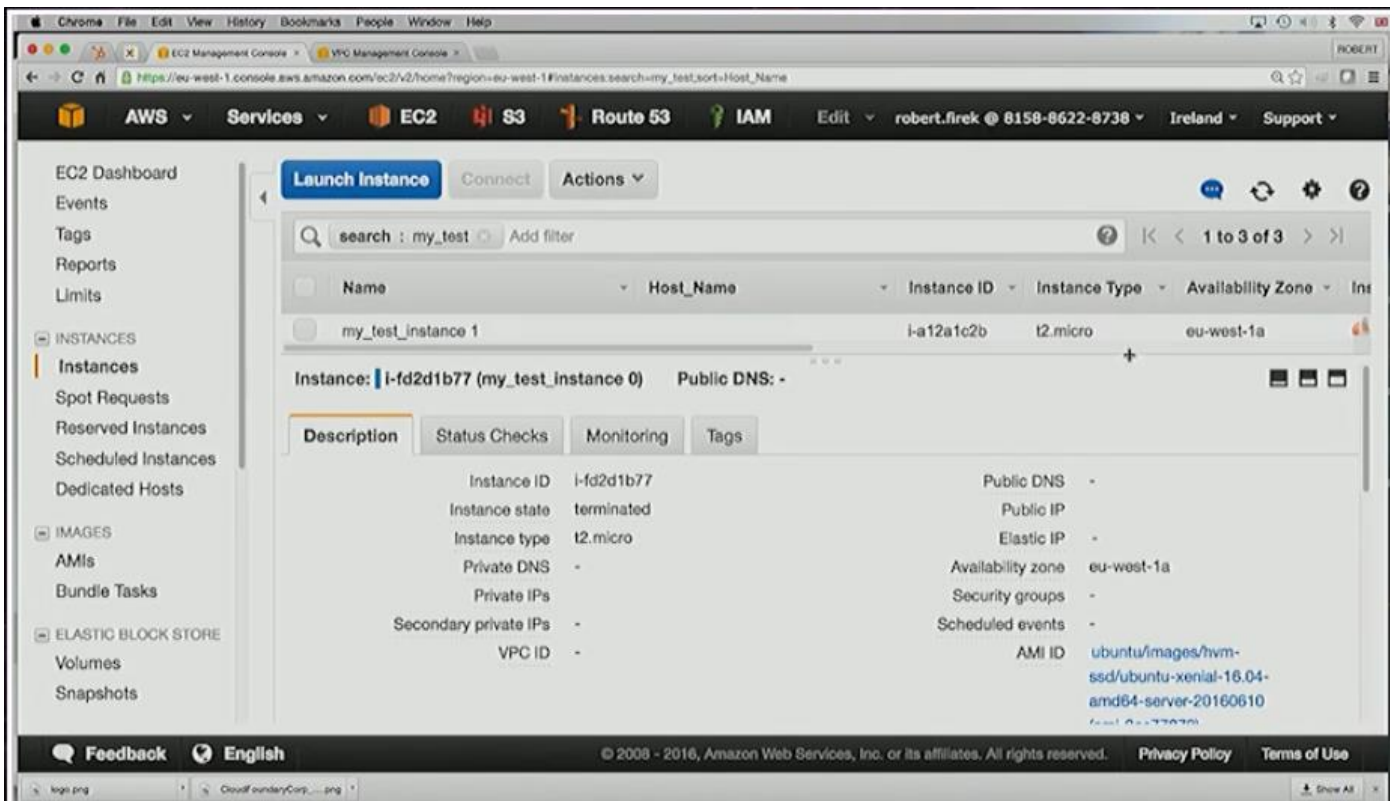
We now have a completely new VPC environment to start deploying EC2 instances into

```
task X1 - ubuntu (7/21) X2
public_dns: "" => "<computed>"
public_ip: "" => "<computed>"
root_block_device.#: "" => "<computed>"
security_groups.#: "" => "<computed>"
source_dest_check: "" => "1"
subnet_id: "" => "subnet-2b50f25d"
tags.#: "" => "1"
tags.Name: "" => "my_test_vpc-instance"
tenancy: "" => "<computed>"
vpc_security_group_ids.#: "" => "1"
vpc_security_group_ids.313707658: "" => "sg-6c38ab0b"
aws_route_table.my_test_subnet_route_table: Creation complete
aws_route_table_association.subnet: Creating...
  route_table_id: "" => "rtb-62a62a06"
  subnet_id: "" => "subnet-2b50f25d"
aws_route_table_association.subnet: Creation complete
aws_nat_gateway.my_test_nat: Still creating... (10s elapsed)
aws_instance.my_test_instance: Still creating... (10s elapsed)
aws_nat_gateway.my_test_nat: Still creating... (20s elapsed)
aws_instance.my_test_instance: Still creating... (20s elapsed)
aws_instance.my_test_instance: Creation complete
aws_nat_gateway.my_test_nat: Still creating... (30s elapsed)
aws_nat_gateway.my_test_nat: Still creating... (40s elapsed)
aws_nat_gateway.my_test_nat: Still creating... (50s elapsed)
aws_nat_gateway.my_test_nat: Still creating... (1m0s elapsed)
aws_nat_gateway.my_test_nat: Still creating... (1m10s elapsed)
aws_nat_gateway.my_test_nat: Still creating... (1m20s elapsed)
aws_nat_gateway.my_test_nat: Still creating... (1m30s elapsed)
aws_nat_gateway.my_test_nat: Still creating... (1m40s elapsed)
aws_nat_gateway.my_test_nat: Still creating... (1m50s elapsed)
aws_nat_gateway.my_test_nat: Still creating... (2m0s elapsed)
aws_nat_gateway.my_test_nat: Creation complete

Apply complete! Resources: 9 added, 0 changed, 0 destroyed.

The state of your infrastructure has been saved to the path
below. This state is required to modify and destroy your
infrastructure, so keep it safe. To inspect the complete state
use the 'terraform show' command.

State path: terraform.tfstate
VPC (master) $
```



Chrome File Edit View History Bookmarks People Window Help

EC2 Management Console x VPC Management Console x

https://eu-west-1.console.aws.amazon.com/ec2/v2/home?region=eu-west-1#instances:search=my\_test:sort=Host\_Name

AWS Services EC2 S3 Route 53 IAM EMR VPC Edit robert.firek 8158-8622-8738 Ireland Support

EC2 Dashboard Events Tags Reports Limits

INSTANCES

Instances

Spot Requests

Reserved Instances

Scheduled Instances

Dedicated Hosts

IMAGES

AMIs

Bundle Tasks

ELASTIC BLOCK STORE

Volumes

Snapshots

NETWORK & SECURITY

Launch Instance Connect Actions

search : my\_test Add filter

1 to 4 of 4

Name	Host Name	Instance ID	Instance Type	Availability Zone	Instance State
my_test_instance 1		i-a12a1c2b	t2.micro	eu-west-1a	terminated
my_test_vpc-instance		i-d7291f5d	t2.micro	eu-west-1a	running
my_test_instance 2		i-d72b1d5d	t2.micro	eu-west-1a	terminated
my_test_instance 0		i-fd2d1b77	t2.micro	eu-west-1a	terminated

Instance: i-fd2d1b77 (my\_test\_instance 0) Public DNS: -

Description Status Checks Monitoring Tags

Instance ID	i-fd2d1b77	Public DNS	-
Instance state	terminated	Public IP	-
Instance type	t2.micro	Elastic IP	-
Private DNS	-	Availability zone	eu-west-1a
Private IPs	-	Security groups	-

Feedback English

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

After Terraform provisions your VPC infrastructure, you can start using Ansible to deploy instance to the VPC.