HashiCorp

Terraform

Write, Plan, and Create Infrastructure as Code

GET STARTED | DOWNLOAD 0.10.7

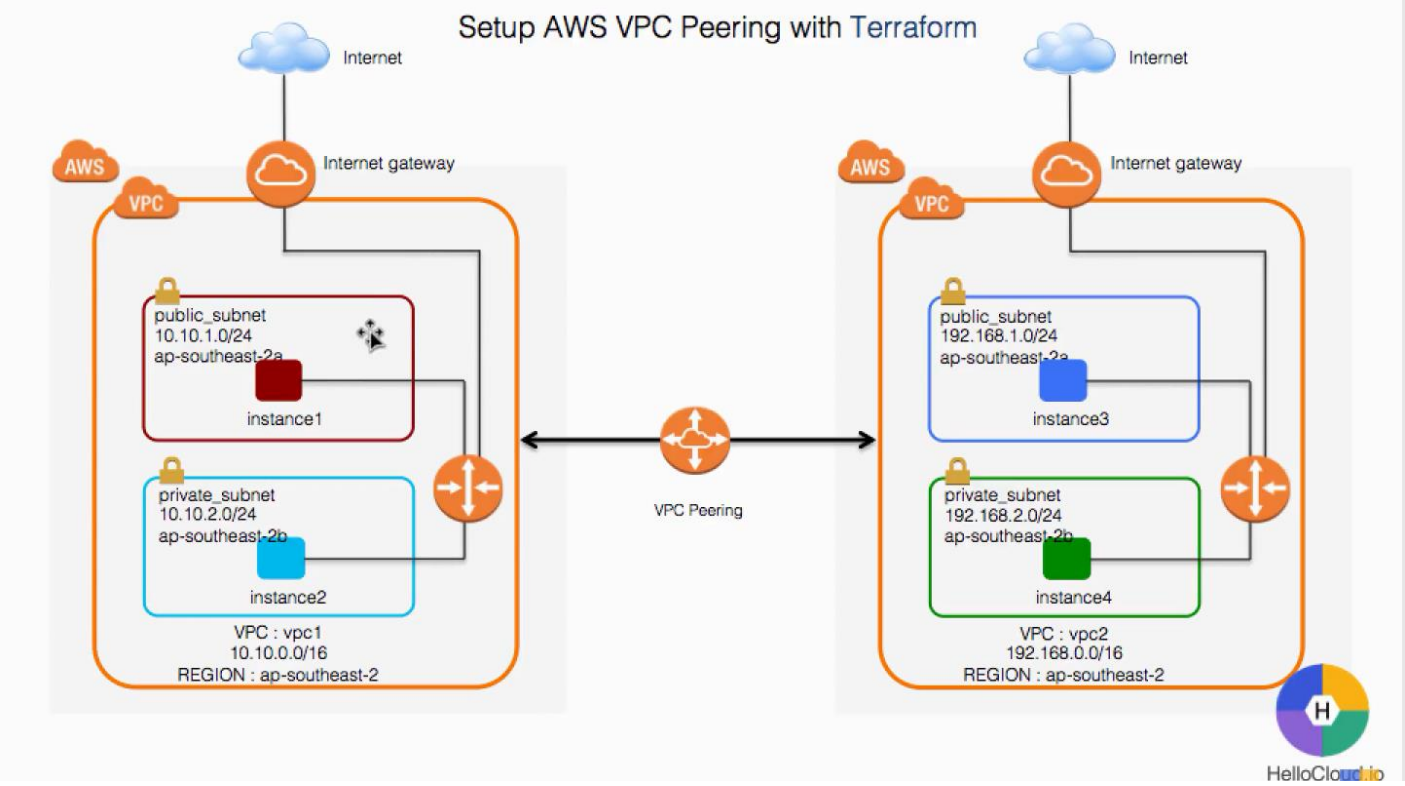
Terraform ENTERPRISE

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



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"}
}
```

HelloCloud.io

Resource Providers > 60+

amazon web services
Microsoft
Google Cloud
vmware
ORACLE
Alibaba Cloud
IBM
GitHub
Bitbucket
GitLab
DATADOG
librato solarwinds

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

HelloCloud.io

WRITE

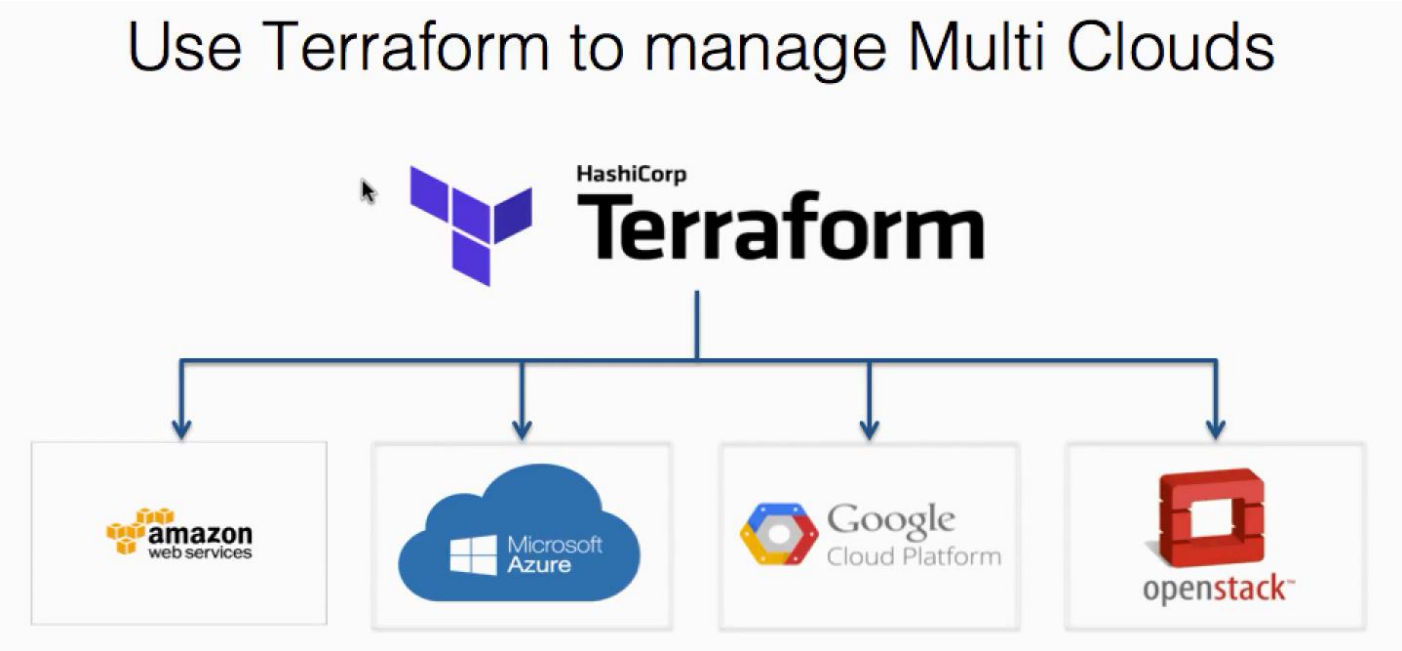
INFRASTRUCTURE AS CODE

PLAN

PREVIEW CHANGES BEFORE APPLYING

CREATE

REPRODUCIBLE INFRASTRUCTURE

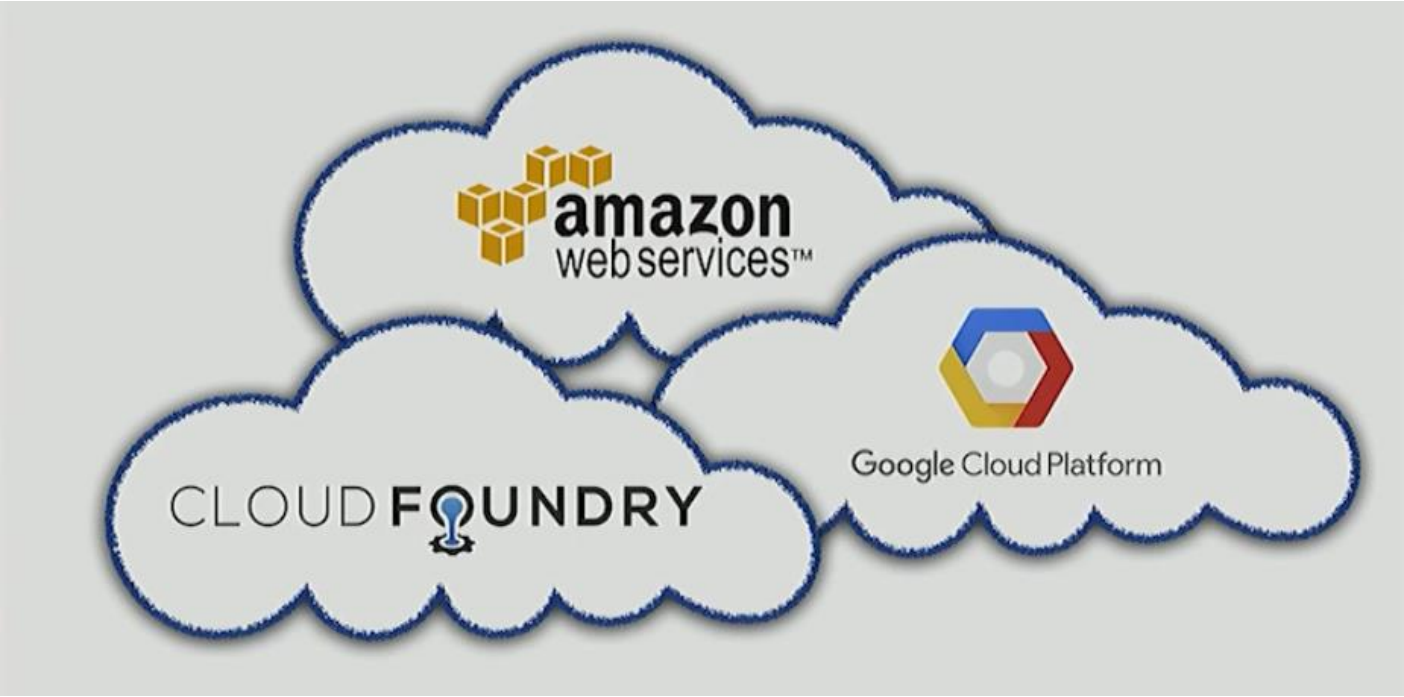


Shape your infrastructure

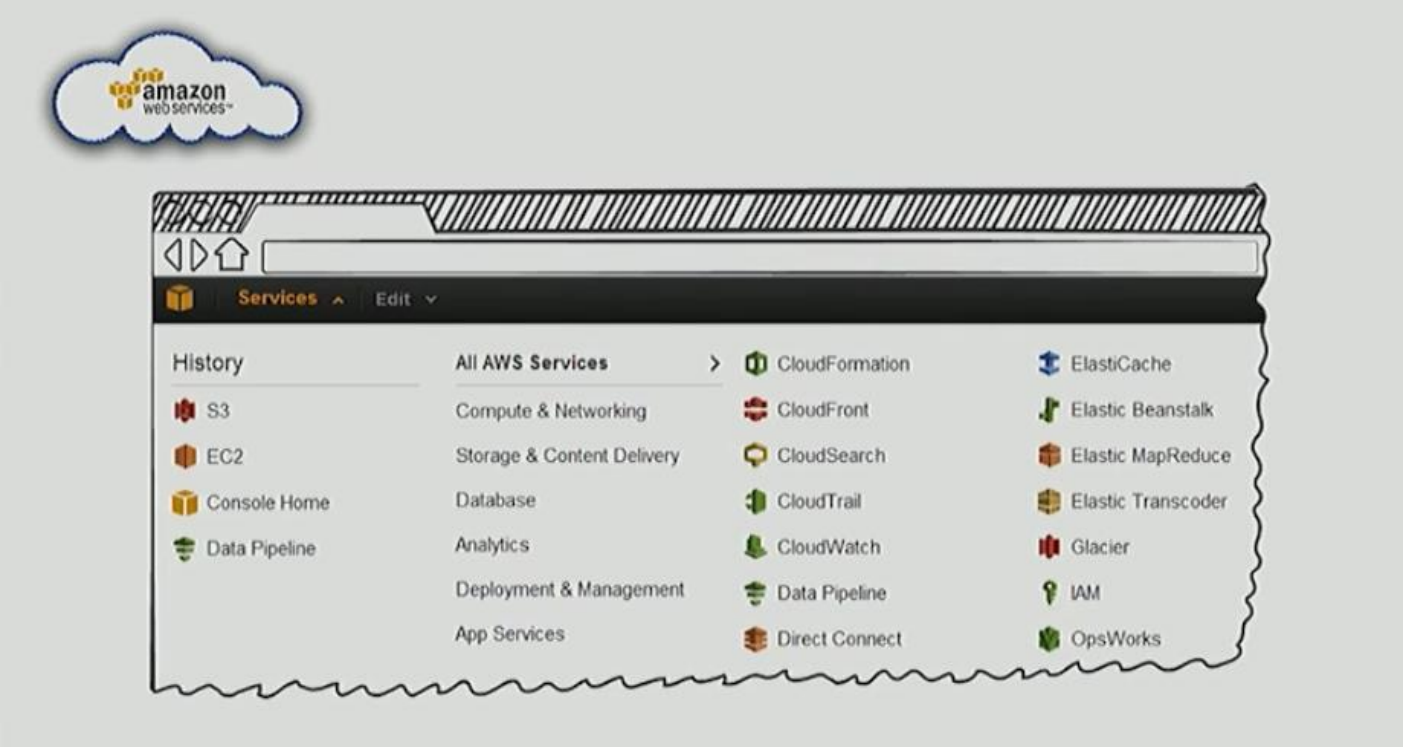
with Terraform

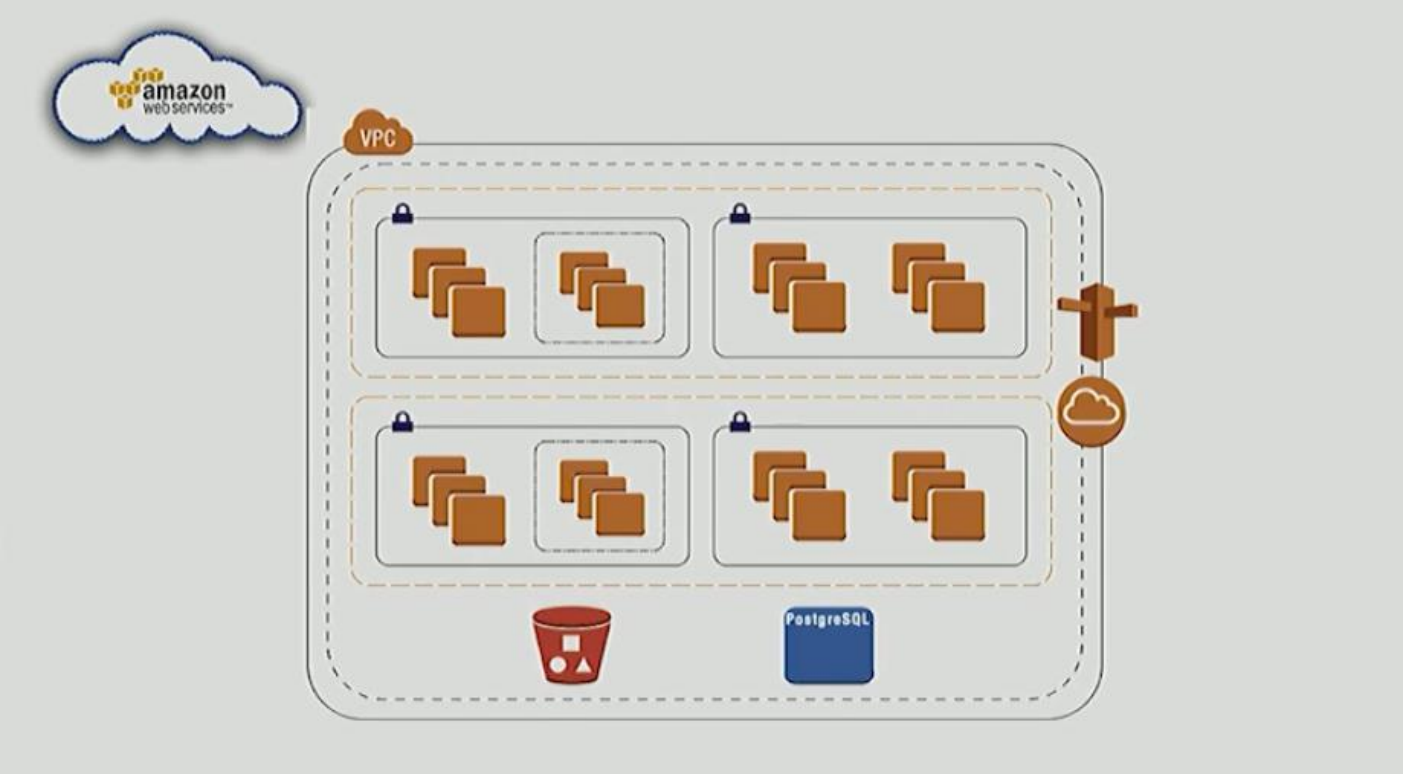
Robert Firek
codurance
craft at heart


IaaS


amazon
web services™
CLOUD FOUNDRY
Google Cloud Platform


amazon
web services™





You need to automate this entire process

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

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

```
provider "aws" {
  region  = "eu-west-1"
  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.

```
resource "aws_instance" "my_test_instance" {
  ami                         = "ami-0ae77879"
  availability_zone           = "eu-west-1a"
  instance_type               = "t2.micro"
  key_name                    = "codurance"
  vpc_security_group_ids      = ["sg-38dc5b5d"]
  subnet_id                   = "subnet-42975e35"
  associate_public_ip_address = false
  source_dest_check           = true

  tags {
    Name = "my_test_instance"
  }
}
```

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)`': ^C
EC2 (master) $ terraform --version

Terraform v0.6.15

EC2 (master) $
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.

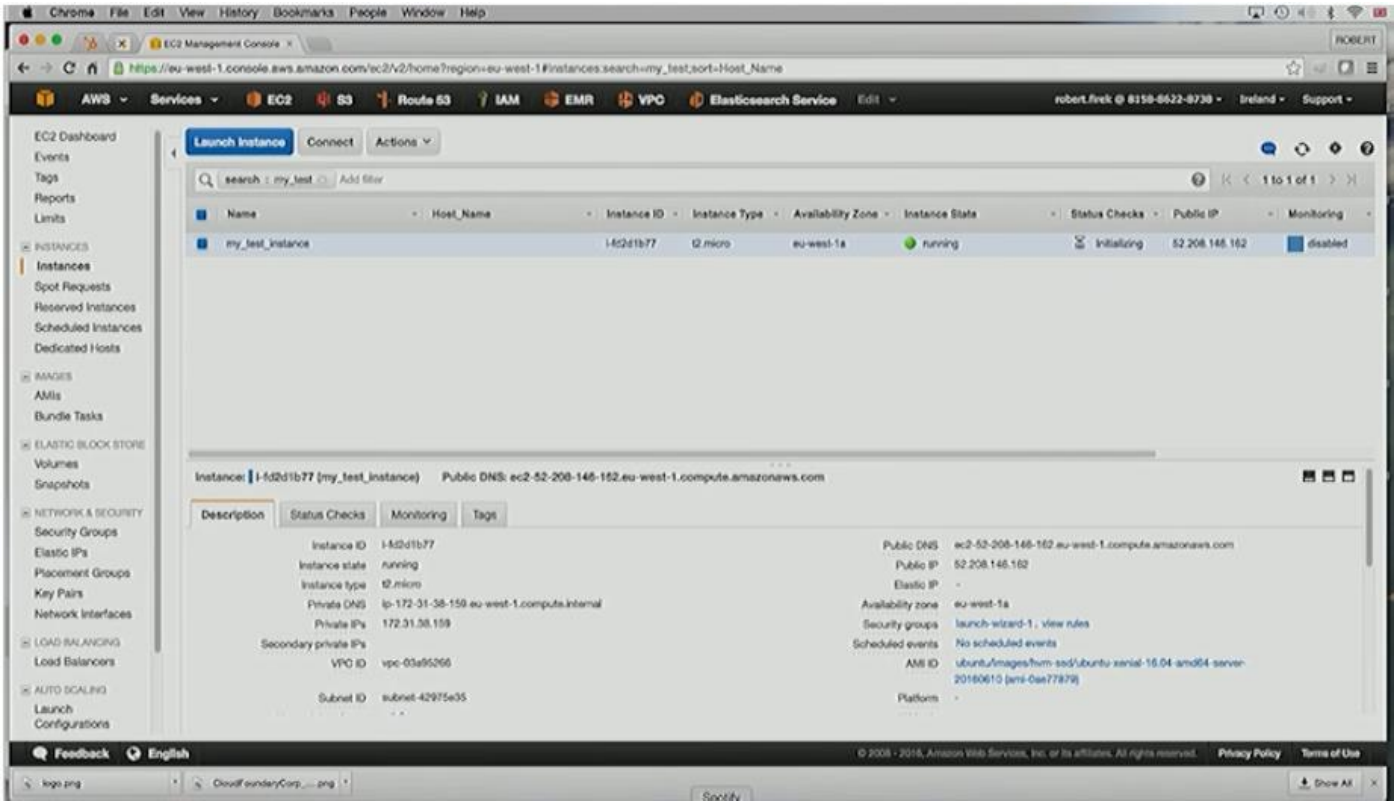
```
Project  Packages
DevoxxPL_2016 ~/Projects/Presentati
  .idea
  EC2
    instances.tf
    provider.tf
    terraform.tfstate
  VPC
  Devoxx_2016_terraform.key
  Devoxx_2016_terraform.pdf
  Devoxx_2016_terraform.pptx
  VPC.png
  VPC.xml
External Libraries
```

```
{
    "version": 1,
    "serial": 0,
    "modules": [
        {
            "path": [
                "root"
            ],
            "outputs": {},
            "resources": {}
        }
    ]
}
```

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

```
(reverse-i-search)`': ^C
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) $
```

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



We now have the instance running in AWS

```
Project  Packages
DevoxxPL_2016 ~/Projects/Presentati
  .idea
  EC2
    instances.tf
    provider.tf
    terraform.tfstate
    terraform.tfstate.backup
  VPC
  Devoxx_2016_terraform.key
  Devoxx_2016_terraform.pdf
  Devoxx_2016_terraform.pptx
  VPC.png
  VPC.xml
External Libraries
```

```
{
    "version": 1,
    "serial": 1,
    "modules": [
        {
            "path": [
                "root"
            ],
            "outputs": {},
            "resources": {
                "aws_instance.my_test_instance": {
                    "type": "aws_instance",
                    "primary": {
                        "id": "i-fd2d1b77",
                        "attributes": {
                            "ami": "ami-0ae77879",
                            "associate_public_ip_address": "false",
                            "availability_zone": "eu-west-1a",
                            "disable_api_termination": "false",
                            "ebs_block_device.#": "0",
                            "ebs_optimized": "false",
                            "ephemeral_block_device.#": "0",
                            "iam_instance_profile": "",
                            "id": "i-fd2d1b77",
                            "instance_state": "running",
                            "instance_type": "t2.micro",
                            "key_name": "codurance",
                            "monitoring": "false",
                            "private_dns": "ip-172-31-38-159.eu-west-1.compute.internal",
                            "private_ip": "172.31.38.159",
                            "public_dns": "ec2-52-208-146-162.eu-west-1.compute.amazonaws.com"
                            "public_ip": "52.208.146.162",
                            "root_block_device.#": "1",
                            "root_block_device.0.delete_on_termination": "true",
                            "root_block_device.0.iops": "100",
```

```
Project  Packages
DevoxxPL_2016 ~/Projects/Presentati
  .idea
  EC2
    instances.tf
    provider.tf
    terraform.tfstate
    terraform.tfstate.backup
  VPC
  Devoxx_2016_terraform.key
  Devoxx_2016_terraform.pdf
  Devoxx_2016_terraform.pptx
  VPC.png
  VPC.xml
External Libraries
```

```
my_test_instance
resource "aws_instance" "my_test_instance" {
  ami                         = "ami-0ae77879"
  availability_zone           = "eu-west-1a"
  instance_type               = "t2.micro"
  key_name                    = "codurance"
  vpc_security_group_ids      = ["sg-38dc5b5d"]
  subnet_id                   = "subnet-42975e35"
  associate_public_ip_address = false
  source_dest_check           = true

  tags {
    Name = "my_test_instance"
  }
}
```

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

```
Project  Packages
DevoxxPL_2016 ~/Projects/Presentati
  .idea
  EC2
    instances.tf
    provider.tf
    terraform.tfstate
    terraform.tfstate.backup
  VPC
  Devoxx_2016_terraform.key
  Devoxx_2016_terraform.pdf
  Devoxx_2016_terraform.pptx
  VPC.png
  VPC.xml
External Libraries
```

```
my_test_instance tags Name
resource "aws_instance" "my_test_instance" {
  ami                         = "ami-0ae77879"
  availability_zone           = "eu-west-1a"
  instance_type               = "t2.micro"
  key_name                    = "codurance"
  vpc_security_group_ids      = ["sg-38dc5b5d"]
  subnet_id                   = "subnet-42975e35"
  associate_public_ip_address = false
  source_dest_check           = true
  count                       = 3
  tags {
    Name = "my_test_instance ${count.index}"
  }
}
```

```
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)
```

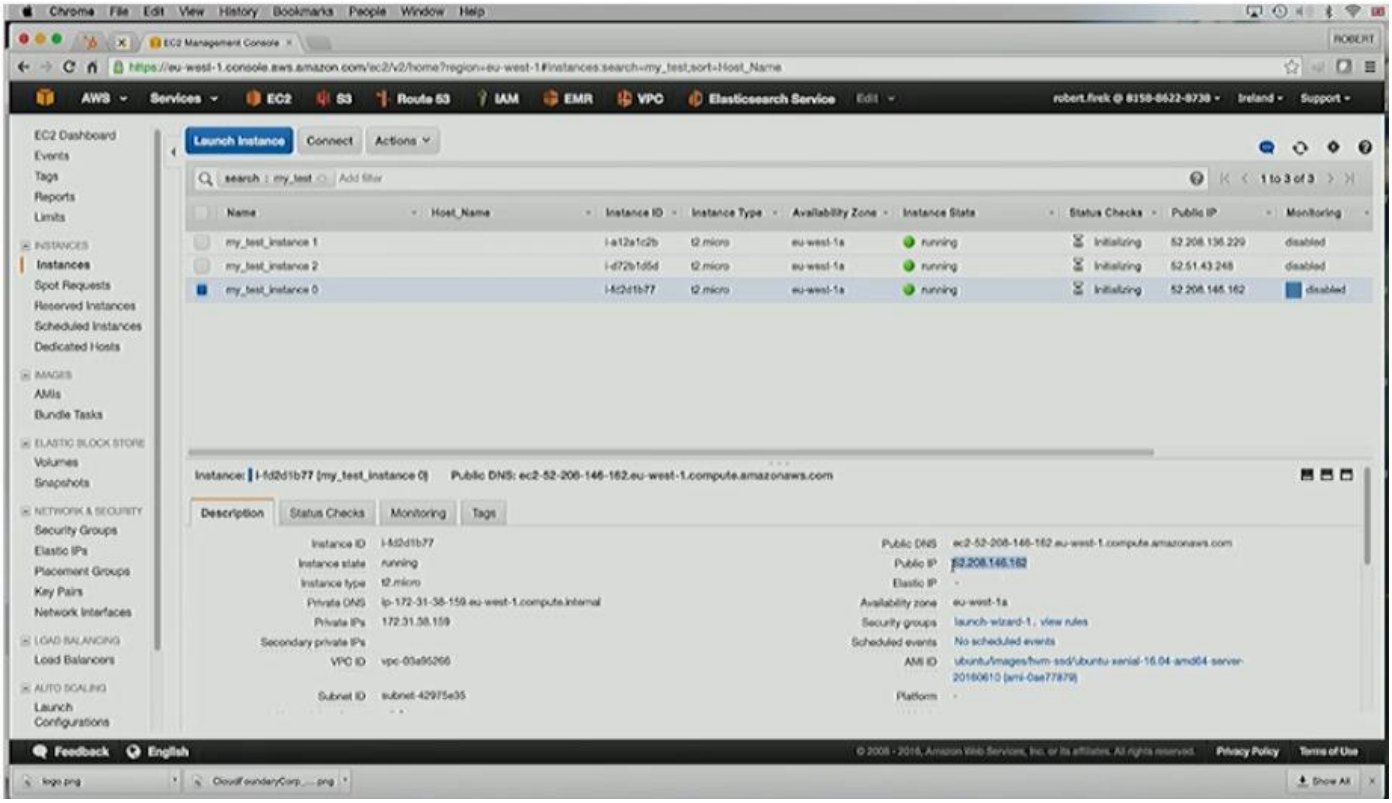
We can now run ***$ terraform plan*** command again to see what changes will be applied

```
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

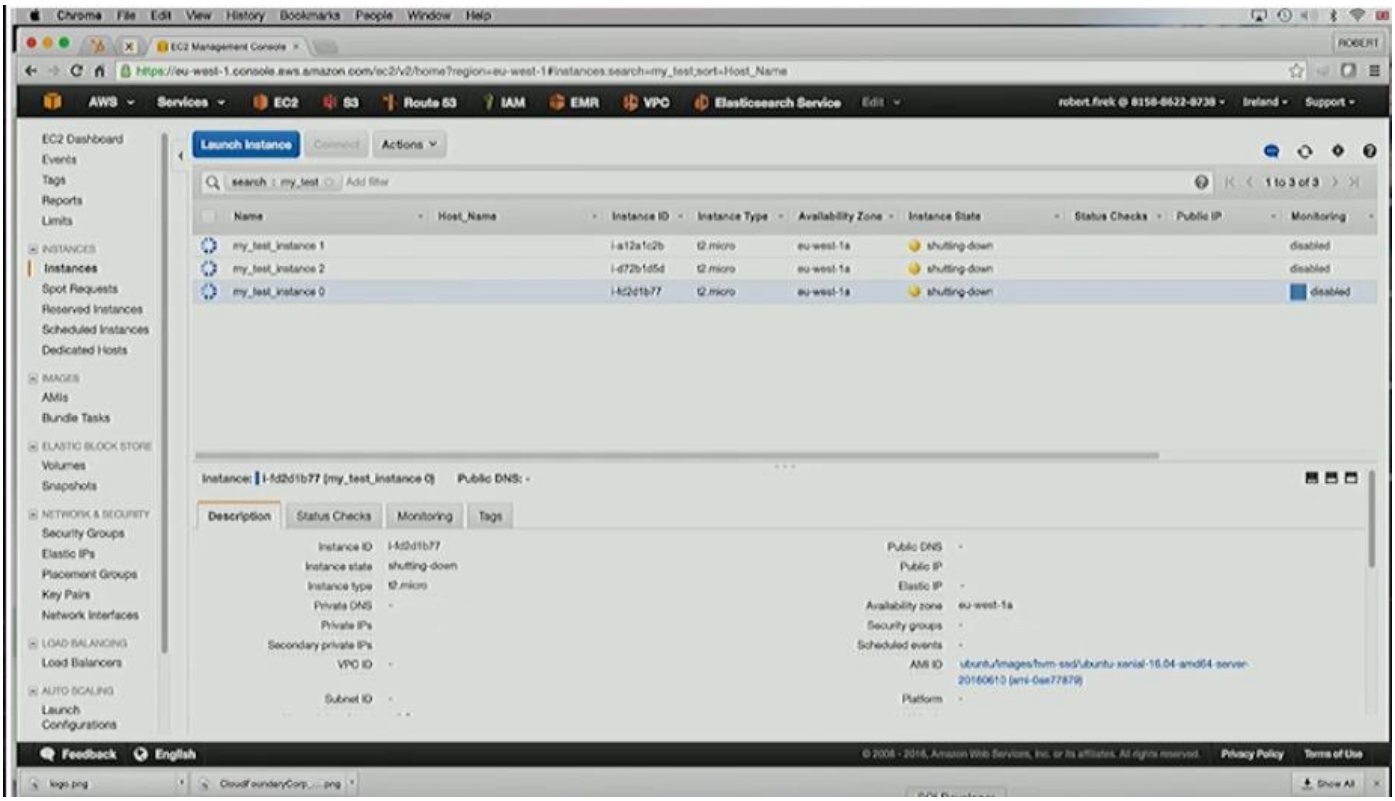
```
  ephemeral_block_device.#:         "" => "<computed>"
  instance_state:                   "" => "<computed>"
  instance_type:                    "" => "t2.micro"
  key_name:                         "" => "codurance"
  placement_group:                  "" => "<computed>"
  private_dns:                      "" => "<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"
```

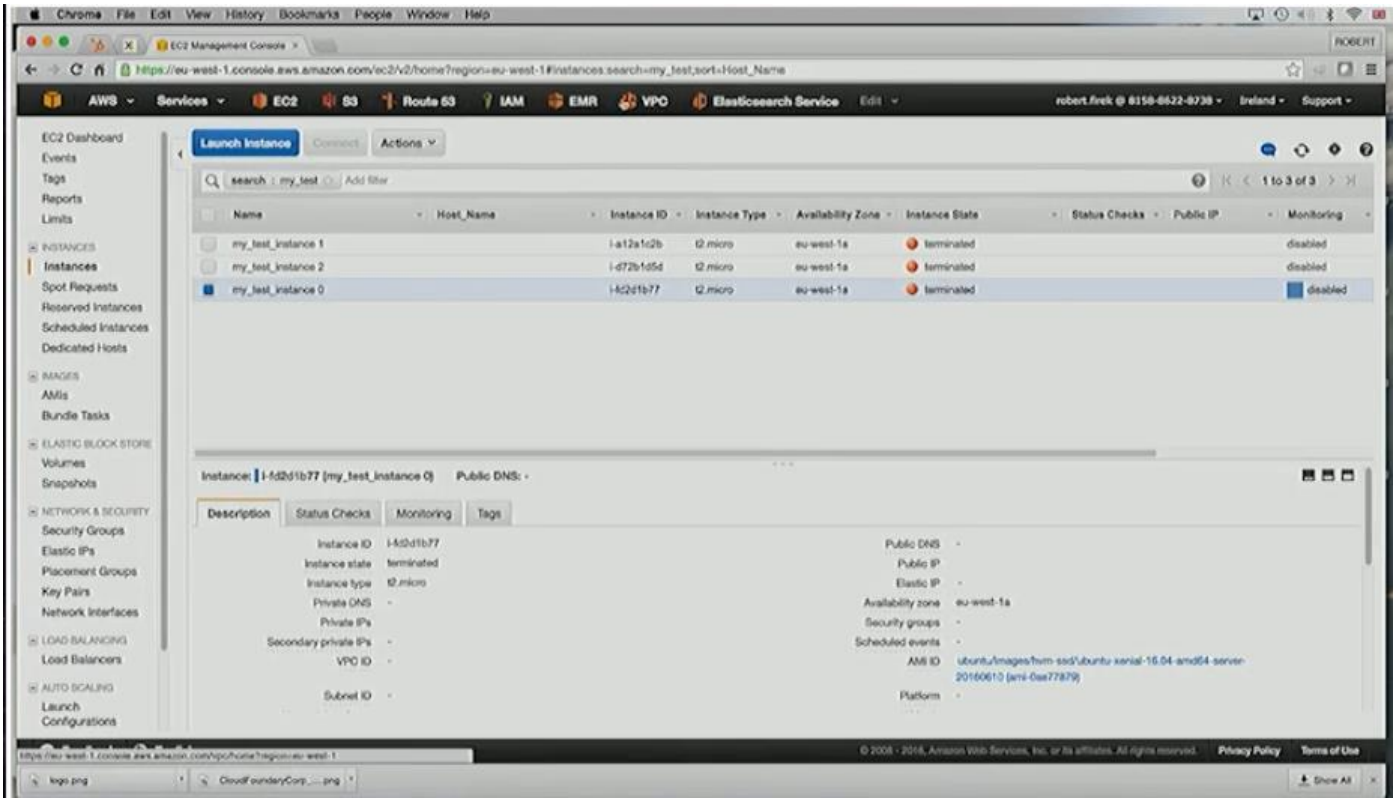


We now have 3 instances running

```
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.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
```

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 Edit
robert.frek @ 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
Instance state shutting-down
Instance type t2.micro
Private DNS -
Private IPs
Secondary private IPs
VPC ID -
Subnet ID -
Public DNS -
Public IP
Elastic IP -
Availability zone eu-west-1a
Security groups -
Scheduled events -
AMI ID ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20160610 (ami-0ae77879)
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 Edit
robert.frek @ 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
Instance state terminated
Instance type t2.micro
Private DNS -
Private IPs
Secondary private IPs -
VPC ID -
Subnet ID -
Public DNS -
Public IP
Elastic IP -
Availability zone eu-west-1a
Security groups -
Scheduled events -
AMI ID ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20160610 (ami-0ae77879)
Platform -
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

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

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

```
my_test_vpc
resource "aws_vpc" "my_test_vpc" {
  cidr_block           = "10.1.0.0/16"
  enable_dns_support   = true
  enable_dns_hostnames = true

  tags {
    Name = "my_test_vpc"
  }
}

resource "aws_internet_gateway" "my_test_internet_gateway" {
  vpc_id = "${aws_vpc.my_test_vpc.id}"

  tags {
    Name = "my_test_vpc-internet_gateway"
  }
}
```

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

```
my_test_subnet tags Name
resource "aws_subnet" "my_test_subnet" {
  vpc_id            = "${aws_vpc.my_test_vpc.id}"
  cidr_block        = "10.1.0.0/24"
  availability_zone = "eu-west-1a"

  tags {
    Name = "my_test_vpc-subnet"
  }
}
```

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

```
resource "aws_route_table" "my_test_subnet_route_table" {
  vpc_id = "${aws_vpc.my_test_vpc.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_internet_gateway.my_test_internet_gateway.id}"
  }

  tags {
    Name = "my_test_vpc-subnet_route_table"
  }
}

resource "aws_route_table_association" "subnet" {
  subnet_id      = "${aws_subnet.my_test_subnet.id}"
  route_table_id = "${aws_route_table.my_test_subnet_route_table.id}"
}
```

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

```
security_group
resource "aws_security_group" "security_group" {
  name = "my_test_vpc-security_group"

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

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

  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    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.my_test_vpc.id}"
  tags {
    Name        = "my_test_vpc-security_group"
  }
}
```

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


```
security_group
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

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

  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    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.my_test_vpc.id}"
  tags {
    Name        = "my_test_vpc-security_group"
  }
}
```

```
my_test_nat_eip
resource "aws_eip" "my_test_nat_eip" {
  vpc = true
}

resource "aws_nat_gateway" "my_test_nat" {
  allocation_id = "${aws_eip.my_test_nat_eip.id}"
  subnet_id = "${aws_subnet.my_test_subnet.id}"

  depends_on = ["aws_internet_gateway.my_test_internet_gateway"]
}
```

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-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.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  209 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) $
```

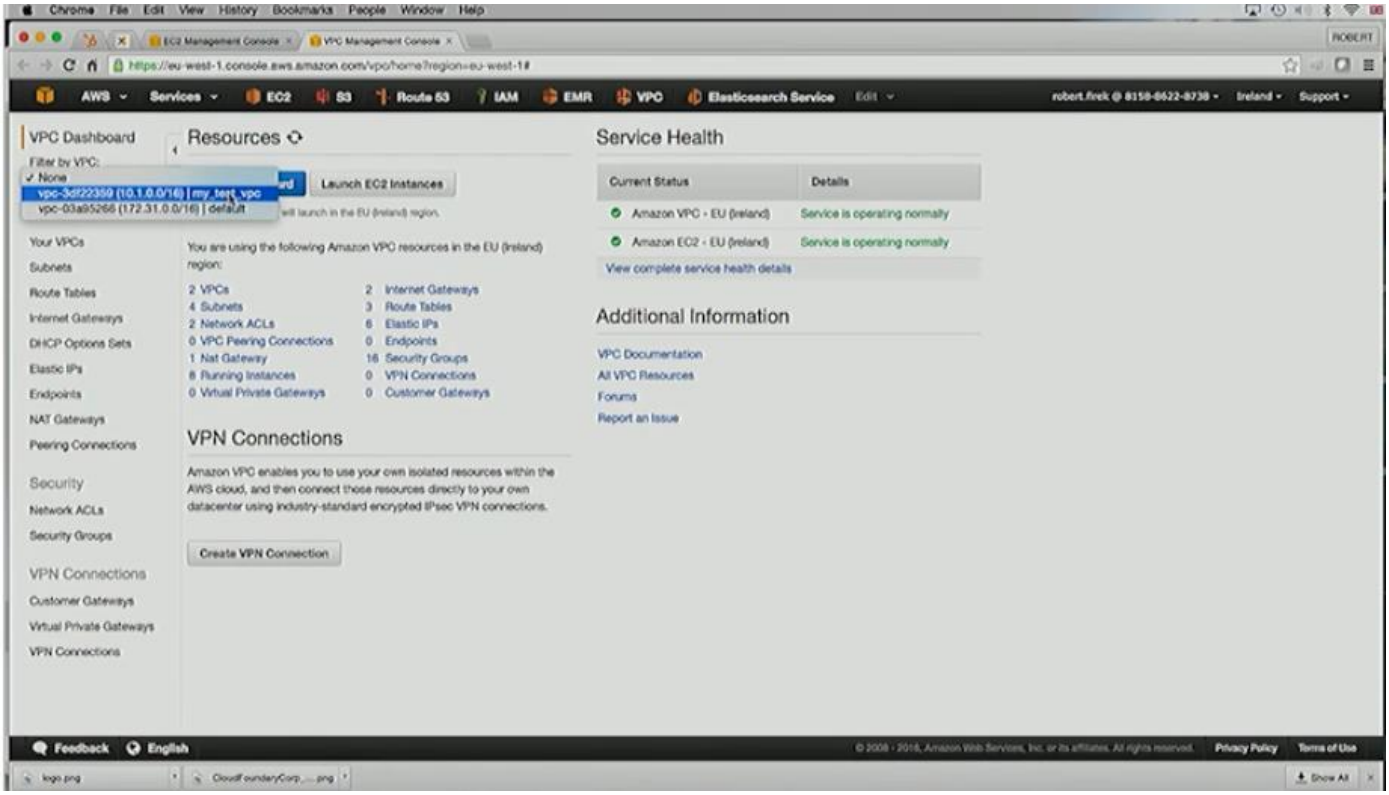
```
VPC (master) $ terraform plan
Refreshing Terraform state prior to plan...
```

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

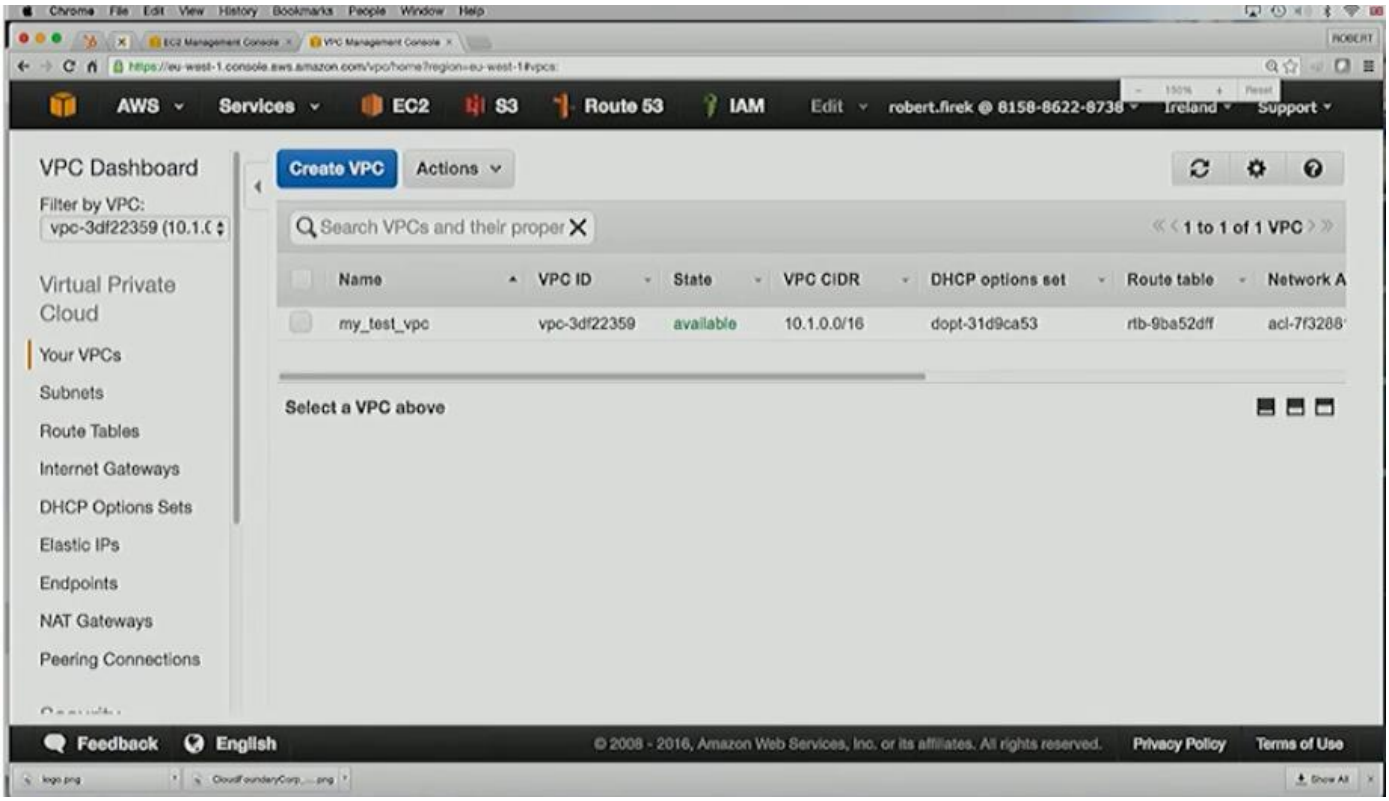
```
    ingress.2541437006.from_port:         "" => "22"
    ingress.2541437006.protocol:          "" => "tcp"
    ingress.2541437006.security_groups.#: "" => "0"
    ingress.2541437006.self:              "" => "0"
    ingress.2541437006.to_port:           "" => "22"
    ingress.2617001939.cidr_blocks.#:     "" => "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) $
```

```
    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_eip.my_test_nat_eip: Creating...
  allocation_id:     "" => "<computed>"
  association_id:    "" => "<computed>"
  domain:            "" => "<computed>"
  instance:          "" => "<computed>"
  network_interface: "" => "<computed>"
  private_ip:        "" => "<computed>"
  public_ip:         "" => "<computed>"
  vpc:               "" => "1"
aws_eip.my_test_nat_eip: Creation complete
```

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

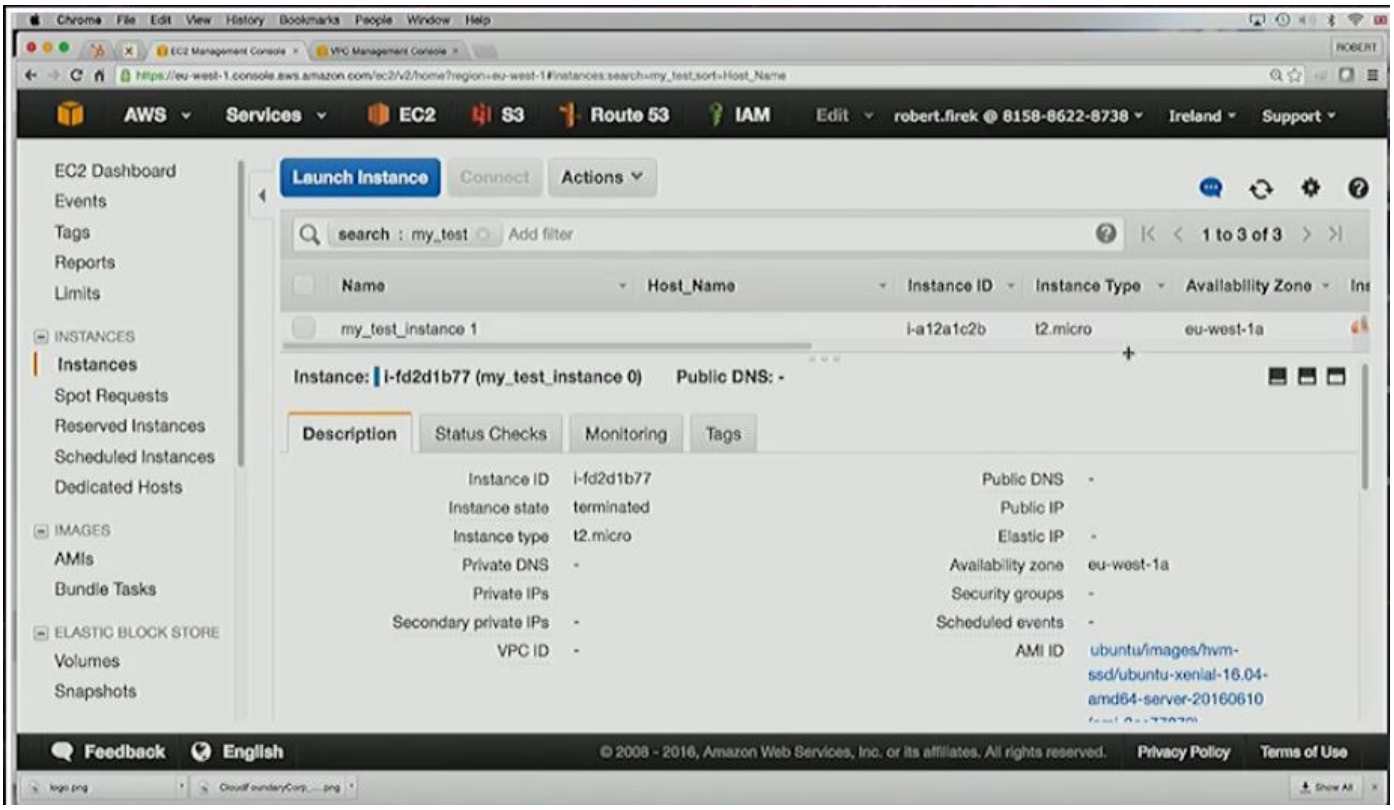


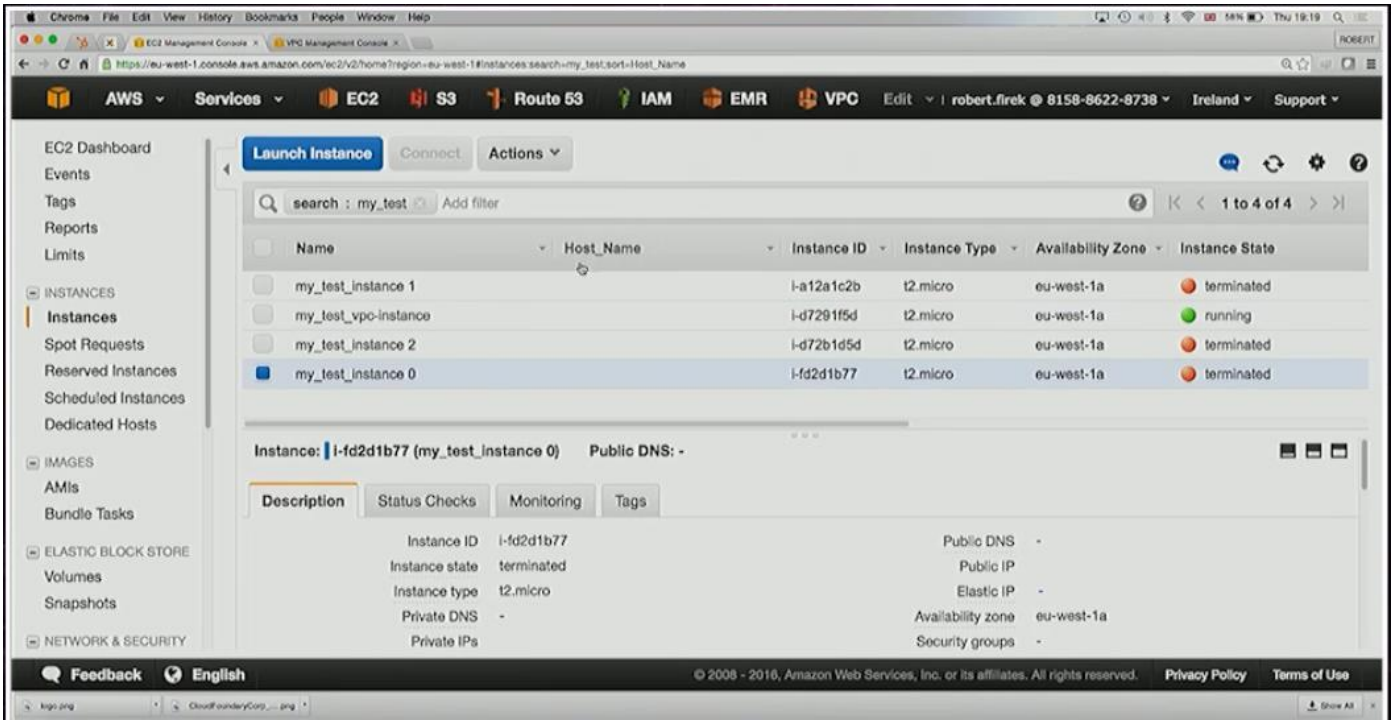
This is the new VPC being created



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

```
  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-62a62e06"
  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) $
```





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