Amazon EKS Terraform Workshop

1. Workshop Introduction
2. Introduction to Kubernetes
3. Start the Workshop
4. Terraform Primer (Optional)
   1. Building a VPC with Terraform
      - Terraform Networking - Part 1
      - **Terraform Networking - Part 2**
      - Terraform Networking - Part 3
   2. Validating Connectivity
      - Testing - Part 1
   3. Creating a Second VPC - with one character !
      - Second VPC - with one Character !
   - Optional - Automatic generation of Terraform

Terraform Networking - Part 2

Allocating an Elastic IP address

In the same directory tflab1

Create a new file named "my-eip.tf"

```
resource "aws_eip" "my-eip" {
  public_ipv4_pool = "amazon"
  tags             = {}
  domain           = "vpc"
  timeouts {}
}
```

File - Save As.... Use the file name **my-eip.tf**

Perform the same steps as before including terraform plan and apply

```
terraform fmt
terraform validate
terraform plan -out tfplan
terraform apply tfplan
```

Amazon EKS Terraform Workshop

1. Workshop Introduction
2. Introduction to Kubernetes
3. Start the Workshop
4. Terraform Primer (Optional)
   1. Building a VPC with Terraform
      - Terraform Networking - Part 1
      - **Terraform Networking - Part 2**
      - Terraform Networking - Part 3
   2. Validating Connectivity
      - Testing - Part 1
   3. Creating a Second VPC - with one character !
      - Second VPC - with one Character !
   - Optional - Automatic generation of Terraform

Next we're going to create a subnet - this part also introduces an important feature in Terraform showing how it references other existing resources by name.

Create a new file called "subnets.tf" with the following contents:

```
resource "aws_subnet" "myprivsubnet" {
  assign_ipv6_address_on_creation = false
  availability_zone               = "eu-west-1a"
  cidr_block                      = "10.1.4.0/24"
  map_public_ip_on_launch         = false
  tags = {
    "Name" = "Private subnet 10.1"
  }
  vpc_id = aws_vpc.vpc-10-1.id

  timeouts {}
}

resource "aws_subnet" "mypubsubnet" {
  assign_ipv6_address_on_creation = false
  availability_zone               = "eu-west-1a"
  cidr_block                      = "10.1.1.0/24"
  map_public_ip_on_launch         = false
  tags = {
    "Name" = "Public subnet 10.1"
  }
  vpc_id = aws_vpc.vpc-10-1.id

  timeouts {}
}
```

`File - Save As` .... use the file name **subnets.tf**

Notice two thing about this file:

1. It contain the definition for two AWS resources (two subnets) - Is this a good idea, having lots of Terraform resources in one file ?

Amazon EKS Terraform Workshop

1. Workshop Introduction
2. Introduction to Kubernetes
3. Start the Workshop
4. Terraform Primer (Optional)
   1. Building a VPC with Terraform
      - Terraform Networking - Part 1
      - **Terraform Networking - Part 2**
      - Terraform Networking - Part 3
   2. Validating Connectivity
      - Testing - Part 1
   3. Creating a Second VPC - with one character !
      - Second VPC - with one Character !
   - Optional - Automatic generation of Terraform

2. Note how "vpc_id" is defined, it refers to an existing resources attribute (the "id") be referencing it's full terraform name "aws_vpc.vpc-10-1.id".

The name aws_vpc.vpc-10-1.id came from your own definition of the VPC - look at the first line in the file vpc-10-1.tf

resource **"aws_vpc" "vpc-10-1"**

Notice how this mapped into terraform as a Terraform resource name

```
1   terraform state list | grep vpc
```

aws_vpc.vpc-10-1

And also note "id" is one of the attributes of the VPC

```
1   terraform state show aws_vpc.vpc-10-1
```

Output:

```
aws_vpc.vpc-10-1:
resource "aws_vpc" "vpc-10-1" {
    arn                              = "arn:aws:ec2:eu-west-1:665389187423:vpc/vpc-081
    assign_generated_ipv6_cidr_block = false
    cidr_block                       = "10.1.0.0/16"
    default_network_acl_id           = "acl-05aad0388ebbc5caa"
    default_route_table_id           = "rtb-04263d67204e83417"
    default_security_group_id        = "sg-0da6ec21c5317ec67"
    dhcp_options_id                  = "dopt-086f476e"
    enable_classiclink               = false
    enable_classiclink_dns_support   = false
    enable_dns_hostnames             = false
    enable_dns_support               = true
```

```
    id                                = "vpc-0817cafa92c07b435"
    instance_tenancy                  = "default"
    main_route_table_id               = "rtb-04263d67204e83417"
    owner_id                          = "665389187423"
    tags                              = {
        "Name" = "vpc-10-1"
    }
}
```


- 2. Introduction to Kubernetes
- 3. Start the Workshop
- 4. Terraform Primer (Optional)
  - 1. Building a VPC with Terraform
    - Terraform Networking - Part 1
    - **Terraform Networking - Part 2**
    - Terraform Networking - Part 3
  - 2. Validating Connectivity
    - Testing - Part 1
  - 3. Creating a Second VPC - with one character !
    - Second VPC - with one Character !
  - Optional - Automatic generation of Terraform

Now proceed to provision our subnets:

```
terraform fmt
terraform validate
terraform plan -out tfplan
```

And if your happy with the plan .....

```
terraform apply tfplan
```

Next use the console to check all the resources exist

You may need to hit refresh if your console window was still open

Look for:

- A new VPC vpc-10-1
- The new Elastic IP
- The new Subnets

1. Workshop Introduction
2. Introduction to Kubernetes
3. Start the Workshop
4. Terraform Primer (Optional)
1. Building a VPC with Terraform
Terraform Networking - Part 1
Terraform Networking - Part 2
Terraform Networking - Part 3
2. Validating Connectivity
Testing - Part 1
3. Creating a Second VPC - with one character !
Second VPC - with one Character !
Optional - Automatic generation of Terraform

Having done all of that - lets destroy what we have created

Yes really do this!

enter "yes" when prompted.

Terraform has supporting IP networking functions to help you calculate and specify CIDR ranges see: https://www.terraform.io/docs/configuration/functions/cidrsubnet.html

Now proceed to part 3