

Terraform scripts to Provision AWS EKS

Steps to be followed:

1. Create an IAM (Identity Access Management) user in AWS and add user to the user_group

- Sign in into AWS console with your credentials
- IAM -> Add user -> name -> add to Group -> add tags -> ok and save.
- After adding user ,Download **new_user_credentials.csv** file .
- Go to the file location and copy the credentials
- Open the terminal

\$ cd Downloads/

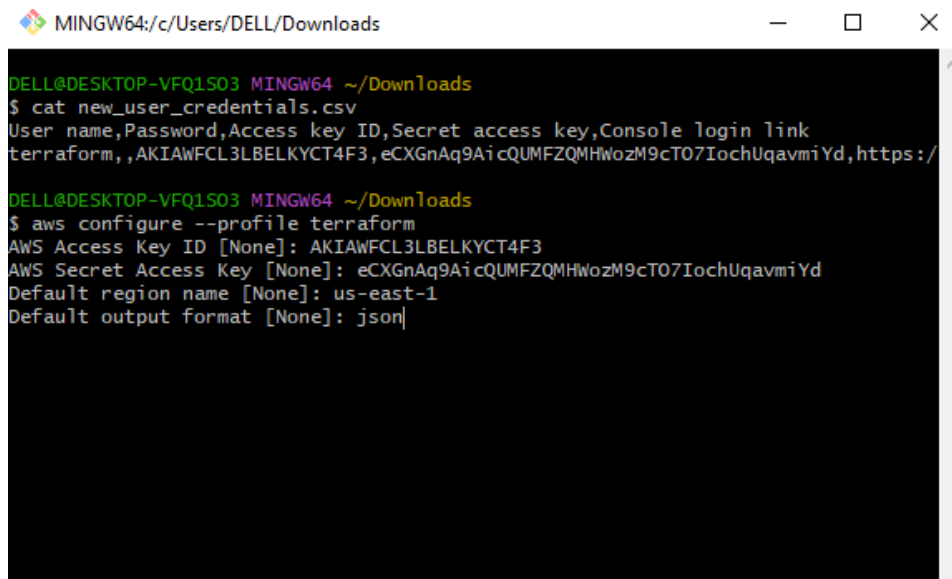
\$ aws configure --profile <profile name>

\$ AWS Access Key ID : <paste from the csv file>

\$ AWS Secret Access Key: <paste from the csv file>

\$ Default region name: us-east-1

\$ Default output format: json



```
MINGW64:/c:/Users/DELL/Downloads
DELL@DESKTOP-VFQ1S03 MINGW64 ~/Downloads
$ cat new_user_credentials.csv
User name,Password,Access key ID,Secret access key,Console login link
terraform,,AKIAWFCL3LBELKYCT4F3,eCXGnAq9AicQUMFZQMHWozM9cT07IochUqavmiYd,https://
DELL@DESKTOP-VFQ1S03 MINGW64 ~/Downloads
$ aws configure --profile terraform
AWS Access Key ID [None]: AKIAWFCL3LBELKYCT4F3
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
```

i

2. Create the following Terraform Scripts:

- provider.tf
- vpc.tf
- internet-gateway.tf
- subnets.tf
- eips.tf
- nat-gateways.tf

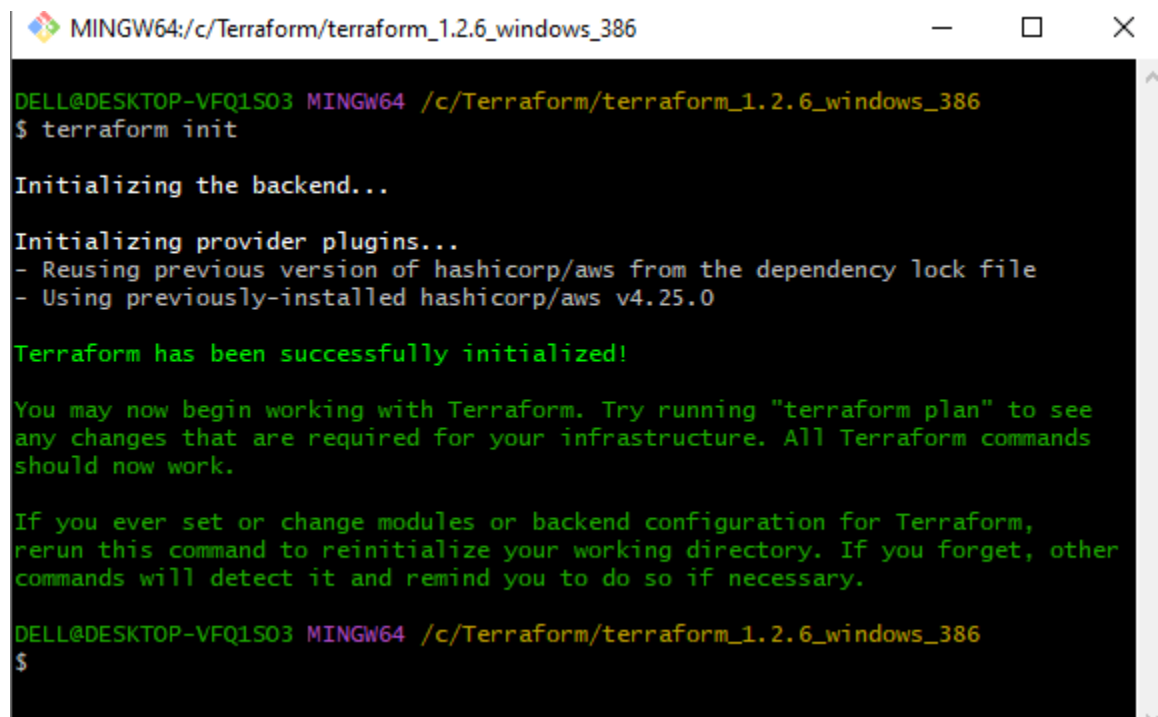
- route-tables.tf
- route-table-association.tf
- eks.tf
- eks-nodegroup.tf

3. Please find below url of my Github Repository for the Source code , where you can find all the Terraform Scripts as mentioned above.

<https://github.com/akshaykumart/two-tier.git>

4. Go to the Terminal :

- `$ cd Terraform` //Location where your Terraform files are located in local
- `$ terraform version` //validate the terraform version
- `$ aws --version` // Validate the AWS version
- `$ terraform init` // Initialize the terraform Environment



```

MINGW64:/c/Terraform/terraform_1.2.6_windows_386
DELL@DESKTOP-VFQ1S03 MINGW64 /c/Terraform/terraform_1.2.6_windows_386
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.25.0

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.

DELL@DESKTOP-VFQ1S03 MINGW64 /c/Terraform/terraform_1.2.6_windows_386
$

```

- `$ terraform plan`
- `$ terraform apply`

MINGW64:/c/Terraform/terraform_1.2.6_windows_386

```
DELL@DESKTOP-VFQ1S03 MINGW64 /c/Terraform/terraform_1.2.6_windows_386
$ terraform plan
aws_instance.example: Refreshing state... [id=i-05fd33384fd34f84]
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

```
# aws_vpc.main will be created
+ resource "aws_vpc" "main" {
  + arn                               = (known after apply)
  + assign_generated_ipv6_cidr_block = false
  + cidr_block                       = "10.0.0.0/16"
  + default_network_acl_id          = (known after apply)
  + default_route_table_id         = (known after apply)
  + default_security_group_id      = (known after apply)
  + dhcp_options_id                = (known after apply)
  + enable_classiclink              = false
  + enable_classiclink_dns_support = false
  + enable_dns_hostnames            = true
  + enable_dns_support              = true
  + id                             = (known after apply)
  + instance_tenancy                = "default"
  + ipv6_association_id             = (known after apply)
  + ipv6_cidr_block                 = (known after apply)
  + ipv6_cidr_block_network_border_group = (known after apply)
  + main_route_table_id            = (known after apply)
  + owner_id                       = (known after apply)
  + tags                           = {
    + "Name" = "main"
  }
  + tags_all                       = {
    + "Name" = "main"
  }
}
```

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:

```
+ vpc_id = (known after apply)
```

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

```
DELL@DESKTOP-VFQ1S03 MINGW64 /c/Terraform/terraform_1.2.6_windows_386
$
```

MINGW64:/c/Terraform/terraform_1.2.6_windows_386

```
+ resource "aws_vpc" "main" {
  + arn                               = (known after apply)
  + assign_generated_ipv6_cidr_block = false
  + cidr_block                       = "10.0.0.0/16"
  + default_network_acl_id          = (known after apply)
  + default_route_table_id         = (known after apply)
  + default_security_group_id      = (known after apply)
  + dhcp_options_id                = (known after apply)
  + enable_classiclink              = false
  + enable_classiclink_dns_support = false
  + enable_dns_hostnames            = true
  + enable_dns_support              = true
  + id                             = (known after apply)
  + instance_tenancy                = "default"
  + ipv6_association_id             = (known after apply)
  + ipv6_cidr_block                 = (known after apply)
  + ipv6_cidr_block_network_border_group = (known after apply)
  + main_route_table_id            = (known after apply)
  + owner_id                       = (known after apply)
  + tags                           = {
    + "Name" = "main"
  }
  + tags_all                       = {
    + "Name" = "main"
  }
}
```

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:

```
+ vpc_id = (known after apply)
```

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_vpc.main: Creating...

aws_vpc.main: Still creating... [10s elapsed]

aws_vpc.main: Creation complete after 15s [id=vpc-036b565f3f25adbc3]

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

Outputs:

```
vpc_id = "vpc-036b565f3f25adbc3"
```

```
DELL@DESKTOP-VFQ1S03 MINGW64 /c/Terraform/terraform_1.2.6_windows_386
$
```

MINGW64:/c/Terraform/terraform_1.2.6_windows_386

```
DELL@DESKTOP-VFQ1S03 MINGW64 /c/Terraform/terraform_1.2.6_windows_386
$ terraform fmt
```

```
DELL@DESKTOP-VFQ1S03 MINGW64 /c/Terraform/terraform_1.2.6_windows_386
$ terraform plan
aws_vpc.main: Refreshing state... [id=vpc-036b565f3f25adbc3]
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

```
# aws_internet_gateway.main will be created
+ resource "aws_internet_gateway" "main" {
  + arn          = (known after apply)
  + id           = (known after apply)
  + owner_id     = (known after apply)
  + tags         = {
    + "Name" = "main"
  }
  + tags_all     = {
    + "Name" = "main"
  }
  + vpc_id       = "vpc-036b565f3f25adbc3"
}
```

Plan: 1 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

```
DELL@DESKTOP-VFQ1S03 MINGW64 /c/Terraform/terraform_1.2.6_windows_386
$ |
```

MINGW64:/c/Terraform/terraform_1.2.6_windows_386

```
+ "Name" = "public-us-east-1a"
+ "kubernetes.io/cluster/eks" = "shared"
+ "kubernetes.io/role/elb" = "1"
}
+ tags_all = {
  + "Name" = "public-us-east-1a"
  + "kubernetes.io/cluster/eks" = "shared"
  + "kubernetes.io/role/elb" = "1"
}
+ vpc_id = "vpc-036b565f3f25adbc3"

# aws_subnet.public_2 will be created
+ resource "aws_subnet" "public_2" {
  + arn = (known after apply)
  + assign_ipv6_address_on_creation = false
  + availability_zone = "us-east-1b"
  + availability_zone_id = (known after apply)
  + cidr_block = "192.168.64.0/18"
  + enable_dns64 = false
  + enable_resource_name_dns_a_record_on_launch = false
  + enable_resource_name_dns_aaaa_record_on_launch = false
  + id = (known after apply)
  + ipv6_cidr_block_association_id = (known after apply)
  + ipv6_native = false
  + map_public_ip_on_launch = true
  + owner_id = (known after apply)
  + private_dns_hostname_type_on_launch = (known after apply)
  + tags = {
    + "Name" = "public-us-east-1b"
    + "kubernetes.io/cluster/eks" = "shared"
    + "kubernetes.io/role/elb" = "1"
  }
  + tags_all = {
    + "Name" = "public-us-east-1b"
    + "kubernetes.io/cluster/eks" = "shared"
    + "kubernetes.io/role/elb" = "1"
  }
  + vpc_id = "vpc-036b565f3f25adbc3"
}
```

Plan: 4 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

```
DELL@DESKTOP-VFQ1S03 MINGW64 /c/Terraform/terraform_1.2.6_windows_386
$ |
```

MINGW64/c/Terraform/terraform_1.2.6_windows_386

```
+ cidr_block = "10.0.64.0/24"
+ enable_dns64 = false
+ enable_resource_name_dns_a_record_on_launch = false
+ enable_resource_name_dns_aaaa_record_on_launch = false
+ id = (known after apply)
+ ipv6_cidr_block_association_id = (known after apply)
+ ipv6_native = false
+ map_public_ip_on_launch = true
+ owner_id = (known after apply)
+ private_dns_hostname_type_on_launch = (known after apply)
+ tags = {
+   "Name" = "public-us-east-1b"
+   "kubernetes.io/cluster/eks" = "shared"
+   "kubernetes.io/role/elb" = "1"
+ }
+ tags_all = {
+   "Name" = "public-us-east-1b"
+   "kubernetes.io/cluster/eks" = "shared"
+   "kubernetes.io/role/elb" = "1"
+ }
+ vpc_id = "vpc-036b565f3f25adbc3"
```

Plan: 4 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?

Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

```
aws_subnet.public_2: Creating...
aws_subnet.private_1: Creating...
aws_subnet.public_1: Creating...
aws_subnet.private_2: Creating...
aws_subnet.private_2: Creation complete after 2s [id=subnet-05bdca032100a1ca4]
aws_subnet.private_1: Creation complete after 2s [id=subnet-0c9221c4cf96dc3c3]
aws_subnet.public_2: Still creating... [10s elapsed]
aws_subnet.public_1: Still creating... [10s elapsed]
aws_subnet.public_2: Creation complete after 12s [id=subnet-0ca656bc5290f05a0]
aws_subnet.public_1: Creation complete after 13s [id=subnet-06e817fcdce6fde02]
```

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

vpc_id = "vpc-036b565f3f25adbc3"

DELL@DESKTOP-VFQ1S03 MINGW64 /c/Terraform/terraform_1.2.6_windows_386
\$

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

```
# aws_eip.nat1 will be created
+ resource "aws_eip" "nat1" {
+   allocation_id = (known after apply)
+   association_id = (known after apply)
+   carrier_ip = (known after apply)
+   customer_owned_ip = (known after apply)
+   domain = (known after apply)
+   id = (known after apply)
+   instance = (known after apply)
+   network_border_group = (known after apply)
+   network_interface = (known after apply)
+   private_dns = (known after apply)
+   private_ip = (known after apply)
+   public_dns = (known after apply)
+   public_ip = (known after apply)
+   public_ipv4_pool = (known after apply)
+   tags_all = (known after apply)
+   vpc = (known after apply)
+ }

# aws_eip.nat2 will be created
+ resource "aws_eip" "nat2" {
+   allocation_id = (known after apply)
+   association_id = (known after apply)
+   carrier_ip = (known after apply)
+   customer_owned_ip = (known after apply)
+   domain = (known after apply)
+   id = (known after apply)
+   instance = (known after apply)
+   network_border_group = (known after apply)
+   network_interface = (known after apply)
+   private_dns = (known after apply)
+   private_ip = (known after apply)
+   public_dns = (known after apply)
+   public_ip = (known after apply)
+   public_ipv4_pool = (known after apply)
+   tags_all = (known after apply)
+   vpc = (known after apply)
+ }

# aws_nat_gateway.gw1 will be created
+ resource "aws_nat_gateway" "gw1" {
+   allocation_id = (known after apply)
```

MINGW64/C:/Terraform/terraform_1.2.6_windows_386

```
+ public_dns      = (known after apply)
+ public_ip       = (known after apply)
+ public_ipv4_pool = (known after apply)
+ tags_all        = (known after apply)
+ vpc             = (known after apply)
}

# aws_nat_gateway.gw1 will be created
+ resource "aws_nat_gateway" "gw1" {
+   allocation_id = (known after apply)
+   connectivity_type = "public"
+   id            = (known after apply)
+   network_interface_id = (known after apply)
+   private_ip     = (known after apply)
+   public_ip      = (known after apply)
+   subnet_id      = "subnet-06e817fcdce6fde02"
+   tags          = {
+     "Name" = "NAT 1"
+   }
+   tags_all    = {
+     "Name" = "NAT 1"
+   }
}

# aws_nat_gateway.gw2 will be created
+ resource "aws_nat_gateway" "gw2" {
+   allocation_id = (known after apply)
+   connectivity_type = "public"
+   id            = (known after apply)
+   network_interface_id = (known after apply)
+   private_ip     = (known after apply)
+   public_ip      = (known after apply)
+   subnet_id      = "subnet-0ca656bc5290f05a0"
+   tags          = {
+     "Name" = "NAT 2"
+   }
+   tags_all    = {
+     "Name" = "NAT 2"
+   }
}
```

Plan: 4 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

DELL@DESKTOP-VFQ1S03 MINGW64 /c/Terraform/terraform_1.2.6_windows_386
\$

MINGW64/C:/Terraform/terraform_1.2.6_windows_386

```
+ "Name" = "NAT 2"
}
+ tags_all = {
+   "Name" = "NAT 2"
+ }
}
```

Plan: 4 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

```
aws_eip.nat2: Creating...
aws_eip.nat1: Creating...
aws_eip.nat1: Creation complete after 1s [id=eipalloc-05c93923b158742c0]
aws_eip.nat2: Creation complete after 1s [id=eipalloc-08238c12f96f259e9]
aws_nat_gateway.gw1: Creating...
aws_nat_gateway.gw2: Creating...
aws_nat_gateway.gw1: Still creating... [10s elapsed]
aws_nat_gateway.gw2: Still creating... [10s elapsed]
aws_nat_gateway.gw1: Still creating... [20s elapsed]
aws_nat_gateway.gw2: Still creating... [20s elapsed]
aws_nat_gateway.gw1: Still creating... [30s elapsed]
aws_nat_gateway.gw2: Still creating... [30s elapsed]
aws_nat_gateway.gw1: Still creating... [40s elapsed]
aws_nat_gateway.gw2: Still creating... [40s elapsed]
aws_nat_gateway.gw1: Still creating... [50s elapsed]
aws_nat_gateway.gw2: Still creating... [50s elapsed]
aws_nat_gateway.gw1: Still creating... [1m0s elapsed]
aws_nat_gateway.gw2: Still creating... [1m0s elapsed]
aws_nat_gateway.gw1: Still creating... [1m10s elapsed]
aws_nat_gateway.gw2: Still creating... [1m10s elapsed]
aws_nat_gateway.gw1: Still creating... [1m20s elapsed]
aws_nat_gateway.gw2: Still creating... [1m20s elapsed]
aws_nat_gateway.gw1: Creation complete after 1m28s [id=nat-0f89a25c82c413b05]
aws_nat_gateway.gw2: Still creating... [1m30s elapsed]
aws_nat_gateway.gw2: Still creating... [1m40s elapsed]
aws_nat_gateway.gw2: Creation complete after 1m49s [id=nat-08ec15eec07ca9709]
```

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

vpc_id = "vpc-036b565f3f25adbc3"

DELL@DESKTOP-VFQ1S03 MINGW64 /c/Terraform/terraform_1.2.6_windows_386
\$

```

MINGW64/c/Terraform/terraform_1.2.6_windows_386
+ id = (known after apply)
+ route_table_id = (known after apply)
+ subnet_id = "subnet-05bdca032100a1ca4"
}

# aws_route_table_association.public1 will be created
+ resource "aws_route_table_association" "public1" {
+ id = (known after apply)
+ route_table_id = (known after apply)
+ subnet_id = "subnet-06e817fcdce6fde02"
}

# aws_route_table_association.public2 will be created
+ resource "aws_route_table_association" "public2" {
+ id = (known after apply)
+ route_table_id = (known after apply)
+ subnet_id = "subnet-0ca656bc5290f05a0"
}

Plan: 7 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

Enter a value: yes

aws_route_table.private1: Creating...
aws_route_table.public: Creating...
aws_route_table.private2: Creating...
aws_route_table.public: Creation complete after 3s [id=rtb-0e96b6fa60a0d8434]
aws_route_table_association.public2: Creating...
aws_route_table_association.public1: Creating...
aws_route_table.private1: Creation complete after 3s [id=rtb-02fe4186172ac28c5]
aws_route_table_association.private1: Creating...
aws_route_table.private2: Creation complete after 4s [id=rtb-05d94f975a6177b6e]
aws_route_table_association.private2: Creating...
aws_route_table_association.public2: Creation complete after 1s [id=rtbassoc-0dbf12dc5afc75ebc]
aws_route_table_association.public1: Creation complete after 1s [id=rtbassoc-0c57a096dc7b34f71]
aws_route_table_association.private1: Creation complete after 1s [id=rtbassoc-0006b48cbc9ed1bfc]
aws_route_table_association.private2: Creation complete after 1s [id=rtbassoc-0ea10c4816fc611e7]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

vpc_id = "vpc-036b565f3f25adb3"

DELL@DESKTOP-VFQ1S03 MINGW64 /c/Terraform/terraform_1.2.6_windows_386
$ |

```

5. Validations:

- Login to AWS console and check whether the following things are created or not.

The screenshot shows the AWS Management Console interface. The top navigation bar includes the AWS logo, a search bar, and the user's name 'Akshay kumar T'. The left sidebar shows the 'Virtual private cloud' section with options like 'Your VPCs', 'Subnets', 'Route tables', etc. The main content area displays 'Your VPCs (1/2)' with a table listing VPCs. The 'main' VPC is selected, and its details are shown in the 'Details' tab.

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
default_vpc	vpc-08b8e0cb6e9d65a57	Available	172.31.0.0/16	-
main	vpc-036b565f3f25adb3	Available	10.0.0.0/16	-

The 'main' VPC details are shown below the table. The 'Details' tab is active, displaying the VPC ID 'vpc-036b565f3f25adb3' and the IPv4 CIDR '10.0.0.0/16'.

- Validate own VPC created as per the Terraform script as shown above

The screenshot shows the AWS Management Console for the us-east-1 region. The left sidebar contains navigation links for VPC dashboard, EC2 Global View, and various VPC resources. The main content area is titled 'Internet gateways (1/3)' and shows a table of existing gateways. The 'main' gateway is selected, and its details are displayed below the table.

Name	Internet gateway ID	State	VPC ID	Owner
-	igw-042402ea20fb8d80b	Attached	vpc-08b8e0cb6e9d65a57 default_vpc	42321335
main	igw-0c2285d210c798241	Attached	vpc-036b565f3f25adb3 main	42321335
-	igw-0d3442bf3939ab162	Detached	-	42321335

The details for the 'main' gateway (igw-0c2285d210c798241) are shown below the table.

- Validate IGW (Internet Gate Way) created as per the Terraform script as shown above

The screenshot shows the AWS Management Console for the us-east-1 region. The left sidebar contains navigation links for VPC dashboard, EC2 Global View, and various VPC resources. The main content area is titled 'Subnets (4/10)' and shows a table of existing subnets. The 'public-us-east-1a' and 'private-us-east-1a' subnets are selected.

Name	Subnet ID	State	VPC	IPv4 CIDR
-	subnet-09a60f39d9fbfc952	Available	vpc-08b8e0cb6e9d65a57 def...	172.31.32.0/20
-	subnet-07948be8d856596a	Available	vpc-08b8e0cb6e9d65a57 def...	172.31.0.0/20
public-us-east-1a	subnet-06e817fcdce6fde02	Available	vpc-036b565f3f25adb3 main	10.0.1.0/24
public-us-east-1b	subnet-0ca656bc5290f05a0	Available	vpc-036b565f3f25adb3 main	10.0.64.0/24
-	subnet-0c72047a346cbe2b1	Available	vpc-08b8e0cb6e9d65a57 def...	172.31.80.0/20
private-us-east-1a	subnet-0c9221c4cf96dc3c3	Available	vpc-036b565f3f25adb3 main	10.0.128.0/24
-	subnet-030f65c79eb10ad07	Available	vpc-08b8e0cb6e9d65a57 def...	172.31.16.0/20
private-us-east-1b	subnet-05bdca032100a1ca4	Available	vpc-036b565f3f25adb3 main	10.0.192.0/24
-	subnet-02c6e05ae732270af	Available	vpc-08b8e0cb6e9d65a57 def...	172.31.64.0/20
-	subnet-0ef35b3359fbc8353	Available	vpc-08b8e0cb6e9d65a57 def...	172.31.48.0/20

- Validate Subnets created as per the Terraform script as shown above.

us-east-1.console.aws.amazon.com/vpc/home?region=us-east-1#Addresses:

Services Search for services, features, blogs, docs, and more [Alt+S]

N. Virginia Akshay kumar T

Your VPCs
Subnets
Route tables
Internet gateways
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections

▼ Security
Network ACLs

Elastic IP addresses (2)

Filter Elastic IP addresses

<input type="checkbox"/>	Name	Allocated IPv4 add...	Type	Allocation ID	Reverse DNS n
<input type="checkbox"/>	-	35.169.29.189	Public IP	eipalloc-08238c12f96f259e9	-
<input type="checkbox"/>	-	35.171.38.9	Public IP	eipalloc-05c93923b158742c0	-

Feedback Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

new_user_credenti...csv Show all

- Validate Elastic IP's created as per the Terraform Script as shown above.

us-east-1.console.aws.amazon.com/vpc/home?region=us-east-1#NatGateways:

Services Search for services, features, blogs, docs, and more [Alt+S]

N. Virginia Akshay kumar T

gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections

▼ Security
Network ACLs
Security groups

▼ Network Analysis
Reachability Analyzer
Network Access Analyzer

▼ DNS Firewall

NAT gateways (1/2) Info

Filter NAT gateways

<input type="radio"/>	Name	NAT gateway ID	Connectivit...	State	State message	Elastic IP addr
<input type="radio"/>	NAT 2	nat-08ec15eec07ca9709	Public	Available	-	35.169.29.189
<input checked="" type="radio"/>	NAT 1	nat-0f89a25c82c413b05	Public	Available	-	35.171.38.9

nat-0f89a25c82c413b05 / NAT 1

Details Monitoring Tags

Details

Feedback Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

- Validate NAT Gateways created as per the Terraform Script as shown above.

us-east-1.console.aws.amazon.com/vpc/home?region=us-east-1#RouteTables:

Search for services, features, blogs, docs, and more [Alt+S]

Select a VPC

Virtual private cloud

- Your VPCs
- Subnets
- Route tables**
- Internet gateways
- Egress-only internet gateways
- Carrier gateways
- DHCP option sets
- Elastic IPs
- Managed prefix lists
- Endpoints
- Endpoint services
- NAT gateways
- Peering connections

Route tables (3/5) Info

Filter route tables

<input checked="" type="checkbox"/>	private1	rtb-02fe4186172ac28c5	subnet-0c9221c4cf96dc...	-	No	vpc-036b565f3
<input type="checkbox"/>	-	rtb-0ba32dd68ccf78d41	-	-	Yes	vpc-036b565f3
<input type="checkbox"/>	-	rtb-051ff2ac3c17e7e3a	-	-	Yes	vpc-08b8e0cb6
<input checked="" type="checkbox"/>	private2	rtb-05d94f975a6177b6e	subnet-05bdca032100a...	-	No	vpc-036b565f3
<input checked="" type="checkbox"/>	public	rtb-0e96b6fa60a0d8434	2 subnets	-	No	vpc-036b565f3

Feedback Looking for language selection? Find it in the new Unified Settings [?] © 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

- Validate the Routing Tables as per the Terraform script as shown above.

us-east-1.console.aws.amazon.com/eks/home?region=us-east-1#/clusters

Search for services, features, blogs, docs, and more [Alt+S]

Amazon Elastic Kubernetes Service

Clusters

Related services

- Amazon ECR
Container storage for EKS
- Documentation [?]
- Submit feedback

EKS > Clusters

Clusters (1) Info

Filter cluster by name, status, kubernetes version, or provider

Cluster name	Status	Kubernetes version	Provider
eks	Creating	1.22	EKS

Feedback Looking for language selection? Find it in the new Unified Settings [?] © 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

- Validate the EKS Cluster as per the Terraform Script as shown above.

Deploying a Two Tier Application on EKS Cluster

Steps to be followed:

I. Install AWS CLI:

- **\$ sudo apt-get update**
- **\$ sudo apt install python3-pip -y**
- **\$ sudo apt install awscli -y**
- **\$ aws configure**
access key id: <provide your IAM user access key>
secret key id: <provide your IAM user secret key>
region: <region of aws resources>
format: <file format>

II. Install AWS Authenticator :

- **\$ curl -Lo aws-iam-authenticator https://github.com/kubernetes-sigs/aws-iam-authenticator/releases/download/v0.5.9/aws-iam-authenticator_0.5.9_linux_amd64**
- **\$ chmod +x ./aws-iam-authenticator**
- **\$ mkdir -p \$HOME/bin && cp ./aws-iam-authenticator \$HOME/bin/aws-iam-authenticator && export PATH=\$PATH:\$HOME/bin**

III. Install Kubectl to communicate with EKS Cluster :

- **\$ curl -o kubectl https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.7/2022-06-29/bin/linux/amd64/kubectl**
- **\$ chmod +x ./kubectl**
- **\$ mkdir -p \$HOME/bin && cp ./kubectl \$HOME/bin/kubectl && export PATH=\$PATH:\$HOME/bin**

IV. To communicate with EKS cluster :

- **\$ aws eks --region <region> update-kubeconfig --name <EKS Cluster name>**
[In region, mention your region where eks is created]
[In name , mention your EKS Cluster name]
- **\$ more /home/ubuntu/.kube/config**

- **\$ export KUBECONFIG=~/.kube/config**
- **\$ kubectl get svc** //verification

V. Deploying a Wordpress and Mysql tier on to EKS Cluster :

- Create a docker compose file called docker-compose.yml

<https://github.com/akshaykumart/two-tier/blob/main/docker-compose.yml>

[Refer the above Github Repository for the scripts and code]

```
$ docker-compose up -d           //creating an image from script
$ docker ps                     //validate the image created
$ docker login                  //docker hub login credentials
    username : <provide your dockerhub account username>
    Password: <provide your dockerhub account password>
$ docker tag <image> <username/repo>:tag //tagging a image to username
$ docker push <username/repo>:tag       pushing image to dockerhub
```

The screenshot shows the Docker Hub interface for the repository 'akshaykumart/test'. The 'Tags' tab is selected, displaying a list of tags. The first tag is 'db', pushed a minute ago, with a digest of '64dc581e69a2' and a compressed size of 137.65 MB. The second tag is 'wordpress', pushed 5 minutes ago, with a digest of '63c4100a4cd0' and a compressed size of 204.73 MB. The interface includes a search bar, navigation links, and a 'Delete' button.

TAG	DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE
db	64dc581e69a2	linux/amd64	---	137.65 MB
wordpress	63c4100a4cd0	linux/amd64	---	204.73 MB

- Create a deployment file and service file for wordpress app called deployment1.yml

<https://github.com/akshaykumart/two-tier/blob/main/deployment1.yml>

[Refer the above Github Repository for the scripts and code]

\$ kubectl apply -f deployment1.yml

- Create a deployment file and service file for database called deployment2.yml

<https://github.com/akshaykumart/two-tier/blob/main/deployment2.yml>

[Refer the above Github Repository for the scripts and code]

\$ kubectl apply -f deployment2.yml

- Verify that the deployments are created or not

\$ kubectl get deployments

```
ubuntu@ip-172-31-21-252:~/two-tier$ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
database      2/2     2            2           15s
wordpress-app 2/2     2            2           4m40s
ubuntu@ip-172-31-21-252:~/two-tier$
```

\$ kubectl get svc

```
ubuntu@ip-172-31-21-252:~/two-tier$ kubectl get svc
NAME          TYPE          CLUSTER-IP      EXTERNAL-IP
database      ClusterIP     172.20.125.145  <none>
kubernetes    ClusterIP     172.20.0.1      <none>
wordpress-app LoadBalancer  172.20.134.96   ab7c2e740ecd34efdafae1fdbe9213af-7042
28348.us-east-1.elb.amazonaws.com 80:30001/TCP    2m12s
ubuntu@ip-172-31-21-252:~/two-tier$
```

\$ kubectl get nodes

```
ubuntu@ip-172-31-21-252:~/two-tier$ kubectl get nodes
NAME                                STATUS    ROLES    AGE   VERSION
ip-10-0-128-111.ec2.internal        Ready     <none>   80m   v1.23.9-eks-ba74326
ip-10-0-192-130.ec2.internal        Ready     <none>   80m   v1.23.9-eks-ba74326
ubuntu@ip-172-31-21-252:~/two-tier$
```

\$ kubectl get pods

```
ubuntu@ip-172-31-21-252:~/two-tier$ kubectl get pods
NAME                                READY    STATUS    RESTARTS   AGE
database-74497879cd-762dd          1/1      Running   0           87s
database-74497879cd-kt76m          1/1      Running   0           87s
wordpress-app-7848478467-kfn8s     1/1      Running   0           5m52s
wordpress-app-7848478467-rbg85     1/1      Running   0           5m52s
ubuntu@ip-172-31-21-252:~/two-tier$
```

- Access the application using external ip from service loadbalancer :

<External Ip from svc>:<port>

