V.P.C :- → Networking ← 25/04/22

Virtual Private cloud :- Isolated cloud resource.

IPv4 CIDR.

ex:- 10.0.0.0 / 16 → Default 2^{32}

IP series — No. of hosts

Here $2^{32-16} = 2^{16}$

$2^{32} = 0 = 1$ hosts

| 10.0. | → Static

		Address range
Class A	2^{24} Hosts	1 to 126
Class B	2^{16} Hosts	128 to 191
Class C	2^{8} Hosts	192 to 223

After the creation of V.P.C, default we will also get 1 Route table, 1 Network ACL (access control list) and 1 security group

Now, for server security we need to create subnets.

Create the Subnet.

V.P.C ID :-

Take the V.P.C that you created not the default one

IPv4 CIDR Block

Choose the same as IPv4 CIDR

10.0.1.0/24

$2^{32-24} = 2^{8} = 256$

We will get 256 Hosts.

Similarly create another subnet with name.

Private-Subnet.

10.0.2.0/24 (for private). 256 IP's.

but we will get only 251, because first 5 are reserved.

1.0.0.0.0
10.0.0.1
10.0.0.2
10.0.0.3
10.0.0.4

} These are reserved for every subnet

* If created a subnet, How many IP's I will get!

→ -5

Now, create a 'INTERNET GATEWAY' separately and then attach this to 'VPC' from "Actions" [attach VPC]

Now, the request can enter into V.P.C but can't access to the subnets servers because the subnets are blocking. Now to access the servers to the subnets we have to go through 'Route Table'. So, create a 'Route Table' with specified name, Now attach the 'Route Table' to 'Gateway Internet'

Create route-table

→ select the route-table.

→ routes
↳ edit routes.

Add route.

ex:-

Destination	Target	Status	propagated
10.0.0.0/16	local	Active	NO.
0.0.0.0/0	internet-gateway.	-	NO.

↓

This is for public.

Save

Subnet associations.

Edit

☑ Public

☐ Private.

* Internet Gateway → Route Table → Server Public.

NOW, TO get the ip address for ec2 public server we need to

enable

→ Subnets

→ public-subnet

→ Actions

→ Edit-Subnet setting

→ Auto-assign IP settings

☑ Enable auto-assign public IPV4 address

Now,

when creating the EC2 instance in the configuration section we need to select the V.P.C which we created

Network — our V.P.C Name.

Subnet — Public Subnet

Similarly for private-subnet can be created.

and for the public-subnet instance create the security from Launch wizard.

We can connect to the public servers as usual but for the private servers we can't connect normally. So, to connect to the private servers we need to connect from via 'Baston Hosts' (Public server)

Now, we need to place 'Pem' file in 'Baston Host (public) Server for that we use 'WinSCP'.

Laptop

Download.

Host name
public server IP → User name
ec2-user → login

drag and drop the pem file to right side.

Now, connected (still few steps).

change permissions.

ls -l

chmod 400 (pem file).

ls -l

Now, connected.

still Internet access is not there for private, so, there we will provide secure internet for private server called 'NAT Gate way'

NAT → Network address Translator

→ V.P.C

→ NAT Gateways.

create NAT Gateway. and this access is given to the public subnet.

→ Subnet

Public

→ connectivity type.

public

Allocate elastic IP

create NAT gateway

Now, this is created under public.

~~NAT~~ Gateway.

→ Route - Table.

select one private route table.

Routes

Edit routes

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	NO.
0.0.0.0/0	Nat-Gateway.		

↓

Elastic Ip will look like

Save

→ Subnet associations

→ Edit Subnet associations

☑ private-subnet

Save

Private → Nat gateway → Route Table → Subnet associations.

"Network ACL" this will block the request coming on the servers which are inside subnets.

(ACL - Access control list)

(1) Network ACL is default created with V.P.C.

Now, we have to allow the traffic.

→ By default they will Deny the access to the servers placed inside subnets.

Create NACL

V.P.C (created V.P.C)

Create

Now, on created NACL.

→ Inbound rules.

→ Edit

Add rules.

→ similarly create the outbound rules.

→ Edit Subnet associations

☑ public-subnet.

→ NACL :- Apply routing on Subnet level (Group) (Allow/Deny) option

→ Security Group :- Apply routing server level (single) No option.

Important components :-

→ V.P.C
→ Subnets
→ NACl
→ NAT gateway
→ Internet gateway
→ Route-Tables
→ security groups

We, can also have V.P.C logs.

Flow logs.

[Create]

Filter - All.

Destination - Cloud watch (attach here).

cloud watch
log
log group
create.
we can IAM role
create policies and roles.
giving cloud watch access.

NOW, TO analyse my V.P.C we have V.P.C flow logs.

V.P.C endpoints are used to connect to S3 buckets without Network gateways and internet.

26/04/22

→ V.P.C Peering.

TO connect V.P.C (2) in same regions, in different account are different regions.

→ Route 53:-

DNS:- Domain name service.

⇒ TO Map (or) associate Domain name to the server.

ex:- GoDaddy → Paid service.

AWS → Paid service. (Route 53)

↓

Route 53.

Domain registration

Freenom.Dom → Free domain service.

↓

register and get desired name

ex:- abc-zn.tk.

↓

Route 53

↓

DNS management

↓

Create hosted zone

Domain name - abczn.tk

Public hosted zone

After creating the hosted zone NS (Name server) and (SOA) A start of Authority.

Now, Take this name server 'Names' and copy somewhere.

Now, create record (A records)

→ Record Name ---- blog abczn.tk.

→ Value : IP address form EC2.

TTL (second) (Time to leave). Routing policy.

300 Simple routing.

create record

Now, similarly give blog (www) for that create another record.

Here we created two records without www and with www (abczn.tk)

Now, place this "NS" (Names) in the domain manager in "freenom" configure with names in 'AWS' Route 53

Different types of routings:-

1* Simple Routing policy.

→ Here there's no load balancing it will send request to servers randomly.

2* Weighted Routing policy:-

Here, we can have servers on different regions and one region has high configured server and other has low config.

we will give more weight of load to high config server min to low weighted server

3* Latency Routing policy:-

Redirect to the server that has the least latency close to Region.

Request will go to the nearest (Region) servers.

4* Failure over routing:-

we can keep servers on different regions and send the request to the servers with good health but, by default we need to set the primary and secondary servers.

→ Route 53 will also check the Health of servers (Health check).

→ we can set the health check by providing

Protocol
Ip address
Host name
port
path

Geo Location Routing policy:-

on the country level
if we get request from particular country we will coping with the nearest server.

multi-value Routing policy:-

it is similar to Load Balancer.

Route 53 Routing policies:-

→ Simple routing
→ weighted routing
→ Latency
→ Fail over
→ Geo Location
→ multi-level

Route 53

Hosted Zones

Value

Alias

↓

Choose endpoint

in this we can choose Load Balancer.

} we can route the request not only to server Ip, but route the request to different endpoints (VPC ELB etc)

** Server are placed in different regions; now I want to route traffic to different regions?

A:- Route 53.

27/04/22

Terraform :-

IAT - infrastructure as a Tool "Hashicorp"

EC2
create an EC2 instance then integrating with AWS using.

'IAM' (Giving access)

IAM
users
Add users
Terrayorm
Access :- Administrator access (full access)

Download . scv

on EC2 Server
aws configure

Access key :
secret key :-
Region : us-east-2

Now, Download Terrayorm :- linux
{(pdf pg:-3) link} → Amazon linux

copy the commands

→ sudo yum install -y yum-utils
→ sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/Amazon linux/hashicorp.repo
→ sudo yum -y install terrayorm
→ terrayorm -V

terraform provider :- registry.terraform.io

Providers :-

→ aws

→ Documentation

Now, create a directory and place the code.

```
mkdir vps
ls
cd vpc/
ls
vi provider.tf
```

⇒ Paste the code (pdf).

```
terraform init.
```

{ ansible = modules.
Terraform = resource }

```
vi main.tf
```

⇒ Paste the code (pdf).

logical name.

```
resource "aws_vpc" "main" {
  cidr_block =
  instance_tenancy =
  tags = {
    Name = "main"
  }
}
```

```
terraform validate
```

→ To validate the code.

* terraform plan.

→ information of resource.

* terrform apply.

→ creating the resource.

→ in the Aws we can see "vpc" is created using the terraform.

Now, in main.tf add subnet code to create the subnet for the vpc.

* Vi main.tf.

→ Add code from pdf

"${aws_vpc.main.id}"

aws_vpc → resource. main → logical name. id → property.

* terraform plan.

* terraform apply.

terraform will maintain two files statefile and backup file

statefile - Latest state of the file

Backup file - Old state of the file.

* terraform destroy

→ To destroy all the resources.

Creating Variables :-

* vi vars.tf
  - ⇒ key word should be "variable"

* vi provider.tf
  - ⇒ syntax to variable. "${var. }" (variable name)

* vi main.tf.
  - ⇒ cidr block = "${var. }".

* terraform plan.
* terraform apply!
* vi vars.tf
  - ⇒ Paste the code.
* vi main.tf.
  - ⇒ remove existing and place fresh code for subnet-
* terraform plan
* terraform apply

FOR each :- for changing data

28/04/22

mkdir users.

cd users/

Vi main.tf

→ copy the code. (Creating IAM users).

terrapom plan

terrapom apply

Vi main.tf

→ add another user in first position

terrapom ~~destroy~~ plan

tarrapom apply

→ Error.

tarrapom destroy

.vi main.tf

→ delete the index based code & replace "resource" for each code

for_each = toset (var.names).

terrapom plan.

terrapom apply.

→ "map"

→ mkdir ec2

cd ec2/

vi provider.tf

vi vars.tf

vi main.tf

→ past code.

map = lookup.

keyname = "vamshi_devops".

⇒ terrafom init after creation of every new directory.

terrafom init

terrafom apply.

⇒ After the server created then if want to connect and execute some code!

connection and provision

connection :- connecting to server

provisioner :- executing command in server

S3 bucket :-

mkdir S3

cd S3/

vi main.tf

⇒ past code

terrafom init

terrafom plan

terrafom apply.

vi main.tf

⇒ past code (connection (S3 and ec2, Glacier))

lifecycle

* terraform console

→ To know the properties on the terraform.

Project - 3

→ Creating the infrastructure by using Terraform.

VPC, Subnets, Route Tables, ELB, security Group and Apache webserver.

Mkdir project

cd project/

ls

* vi provider.tf

→ Paste code.

* Vi vars.tf.

* Vi install_httpd.sh

→ #!/bin/bash (shibang line).

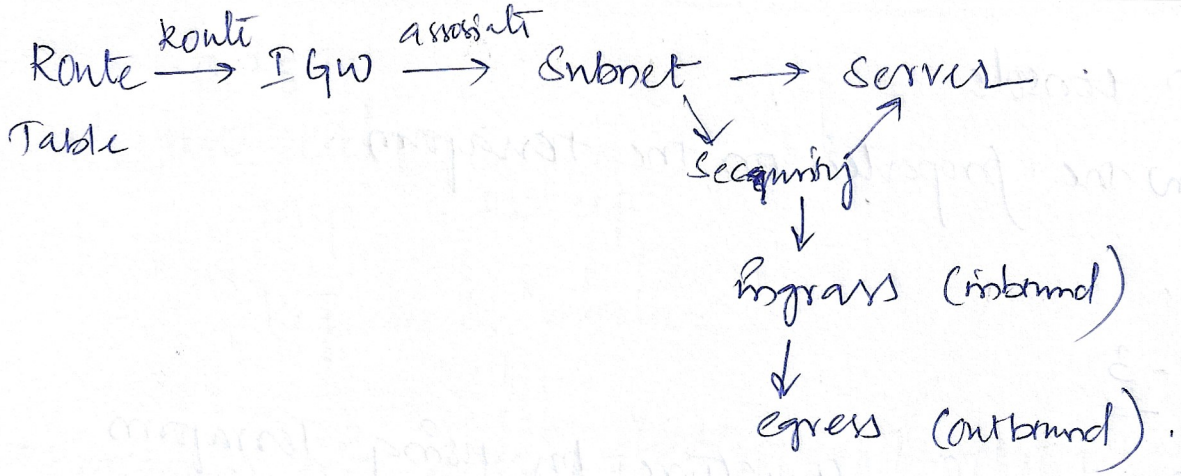
sudo yum install -y.

* Vi ec2.tf

* Vi vpc.tf

* Vi elb.tf.

ls



terraform init.

vi vars.tf

⇒ "list" = list.

vi ec2.tf.

⇒ under ami add key_name = "ramshi-devops".

terraform init

terraform plan.

terraform apply.