**VIRTUAL PRIVATE CLOUD**

**VPC Service – Network Engineering:**

* Stands for Virtual Private Cloud
* An isolated environment within a cloud
* Is a service that allows its users to launch their virtual machines in a protected as well as isolated virtual environment defined by them. You have complete control over your VPC, from creation to customization and even deletion.
* VPC customers can run code, store data, host websites, and do anything else they could do in an ordinary private cloud, but the private cloud is hosted remotely by a public cloud provider
* You can launch AWS resources into a defined virtual network using VPS

**IP Address:**

* Internet Protocol Address
* A unique numerical identifier for every device or network that connects to the internet

**Types of IP Address (Based on version):**

* IPv4 (32-bit address)
* IPv6 (128-bit address)

**Types of IP Address:**

* Public IP Address:
  + Globally unique
  + No duplicates
* Private IP Address:
  + Only among an organisation
  + Duplicates

**Conditions (IPv4):**

* Four numbers separated by 3 full stops
* Numbers should range from 0-255
* Example:
  + 10.0.0.0 (Mostly private VPC has IP address starting with 10)
  + 172.54.9.10 (AWS private VPC always starts with 172)

**VPC**

|  |
| --- |
| **Default VPC:**   * When you start using Amazon VPC, you have a default VPC in each AWS Region. * A default VPC comes with a public subnet in each Availability Zone, an internet gateway, and settings to enable DNS resolution. Therefore, you can immediately start launching Amazon EC2 instances into a default VPC * A default VPC is suitable for getting started quickly and for launching public instances such as a blog or simple website. * You can modify the components of your default VPC as needed. * You can add subnets to your default VPC. |

**How to create a VPC?**

* Go to ‘networking & content delivery’ -> VPC -> Your VPCS -> Create VPC
* Enter tag name (say my-vpc)
* Enter IPv4 CIDR block (say 10.0.0.0/16)
* Your VPC will be created successfully

**Classless Inter-Domain Routing (CIDR)**

* A technique for allocating IP addresses and for IP routing is called classless [Inter-Domain Routing (CIDR](https://www.geeksforgeeks.org/classless-inter-domain-routing-cidr/)), and its range is 0-32.
* When setting up a VPC, we must specify a set of IPv4 addresses using [classless Inter-Domain Routing (CIDR)](https://www.geeksforgeeks.org/classless-inter-domain-routing-cidr/), for (**Example:**10.0.0.0/16 For our VPC, this will serve as the main CIDR block).

**Internet Gateway:**

* With the help of **IGW** (Internet Gateway), the resources present (e.g: EC2) in the VPC will enable to access the Internet.
* One VPC can’t have more than one IGW
* If resources are running in a certain VPC then IGW cannot be detached from that particular VPC.

**How to create a Internet gateway:**

* Go to ‘Internet Gateways’ -> Create internet gateway
* Enter name tag (say: my-igw)
* Click ‘create internet gateway’
* Your Internet Gateway created successfully

**How to attach Internet gateway to VPC:**

* Once the Internet gateway is created, Click on ‘Actions’
* Click on ‘Attach to VPC’
* Select your VPC (say my-vpc)

**Subnets:**

* A range of IP addresses in your VPC
* Every VPC must have minimum 2 subnets
* A subnet is a smaller portion of the network that typically includes all the machines in a certain area.
* We can add as many as subnets we need in one availability zone. Each subnet must reside entirely within one availability zone.
* The public subnets will be attached to Internet Gateway which enables Internet access.
* The private subnets will not have internet access.
* Each and every subnet which is presented in VPC must be associated with the routing table.
* 200 subnets can be created in one VPC
* Each subnet should be associated with a route table

**How to create a subnet within a VPC?**

* Go to subnets -> Create subnet
* Select you VPC ID (say my-vpc)

**Subnet 1 (Public subnet)**

* Enter subnet name (say public-subnet)
* Select availability zone
* Enter CIDR value (Selecting the range of IPs) (say 10.0.1.0/24)

**Subnet 2 (Private subnet)**

* Enter subnet name (say private-subnet)
* Select availability zone (Better to select different availability zone for disaster recovery)
* Enter CIDR value (say 10.0.2.0/24)

**Route Tables:**

* Route Table contains a set of rules, called route which helps us to route the network traffic.
* A single VPC can have as many as route tables it requires.
* If the dependencies are attached to the route table, then they can’t be deleted.

**How to create a Route table?**

**Public route table:**

* Go to route table -> Create route table
* Enter name for route-table (say public-route-table)
* Select VPC (say my-vpc)
* Create route table

**Private route table:**

* Go to route table -> Create route table
* Enter name for route-table (say private-route-table)
* Select VPC (say my-vpc)
* Create route table

**How to connect route table with subnet:**

**Subnet association –** Connecting route table with subnet

**For public route table:**

* Select your route table (say public-route-table)
* Go to ‘subnet associations’ -> ‘Edit subnet associations’
* Select the subnet (say public-subnet)
* Click on ‘save associations’

**For private route table:**

* Select your route table (say private-route-table)
* Go to ‘subnet associations’ -> ‘Edit subnet associations’
* Select the subnet (say private-subnet)
* Click on ‘save associations’

**How to connect route table with Internet Gateway:**

**Routes -** Connecting route table with Internet gateway

* Select your route table (say public-route-table)
* Go to ‘Routes’
* Click on ‘Edit routes’ -> Add route
* Enter destination (say ’0.0.0.0/0’ which denotes all the IPs)
* Enter the Internet gateway in the target (say ‘my-igw’)
* Click on ‘save changes’

**Security group:**

Instance level security

**Network Access Control Lists:**

* Subnet level security
* The NACL security layer for VPC serves as a firewall to manage traffic entering and leaving one or more subnets.
* The NACL for the default VPC is active and connected to the default subnets.

**NACL vs Security group:**

|  |  |
| --- | --- |
|  |  |
| Instance level |  |
| Allow rules only | Allow and deny rule |
|  |  |
|  |  |

**NAT Gateways (Chargeable):**

* Network Address Translation

**How to gateways:**

* Go to ‘NAT gateways’
* Enter name of the NAT gateway
* Select the subnet (say public-subnet)
* To get Elastic IP Allocation ID, click ‘Allocate Elastic IP’
* Create NAT gateway
* A NAT gateway created successfully

GO to route table

Add target NAT gateway

**Elastic IP (Chargeable):**

**VPC Peering connection:**

* You can connect two VPCs in your own AWS account, between AWS accounts, or between AWS regions

**How to implement VPC peering?**

**Direct connect:**

* To connect VPC with corporate Data center directly

**AWS Transit gateway:**

* To connect multiple VPCs

**How to delete VPC?**

* Delete NAT Gate way
* Release Elastic IPs
* Terminate the instances inside VPC
* Delete VPC

**User data:**

* The scripts written in user data are executed automatically during the booting of instance
* It is a bootstrap script to configure the instance at the first launch
* Bootstrapping means launching commands when the machine starts
* So, that EC2 User data script is only run once and when it first starts, and then will never be run again
* User data shell scripts must start with the #! characters and the path to the interpreter you want to read the script (commonly **/bin/bash)**

Linux command:

#!/bin/bash

yum install httpd -y

service httpd start

echo “<h1>Hello Skcet</h1>” > /var/www/html/index.html

The “Hello Skcet” command will be created in the deployment path

**How to create user data?**

* Create a new instance (say ‘InstanceWithUserData’)
* Create the instance as usual
* Select HTTP and HTTPS in security group
* Go to Additional Settings
* Paste the linux command in user
* Now when this instance is launched, the commands are executed automatically and “Hello Skcet” command can be seen in the public IP address successfully

**Network Engineering**

**Load Balancer**

Let us consider t2.micro(1GB RAM, 1vCPU) can handle 1000 requests. Now when the requests are increased, the server will be crashed. So we must increase the CPU or RAM of the server. This process of increasing or decreasing is called Scaling

Instance->Stop->Change Instance settings

**Types of scaling:**

* **Vertical Scaling**

It is a manual scaling method

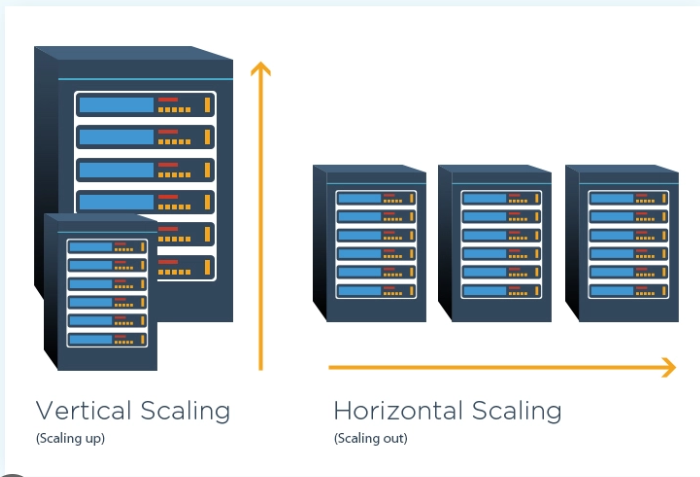
Changing the instance type of an existing instance

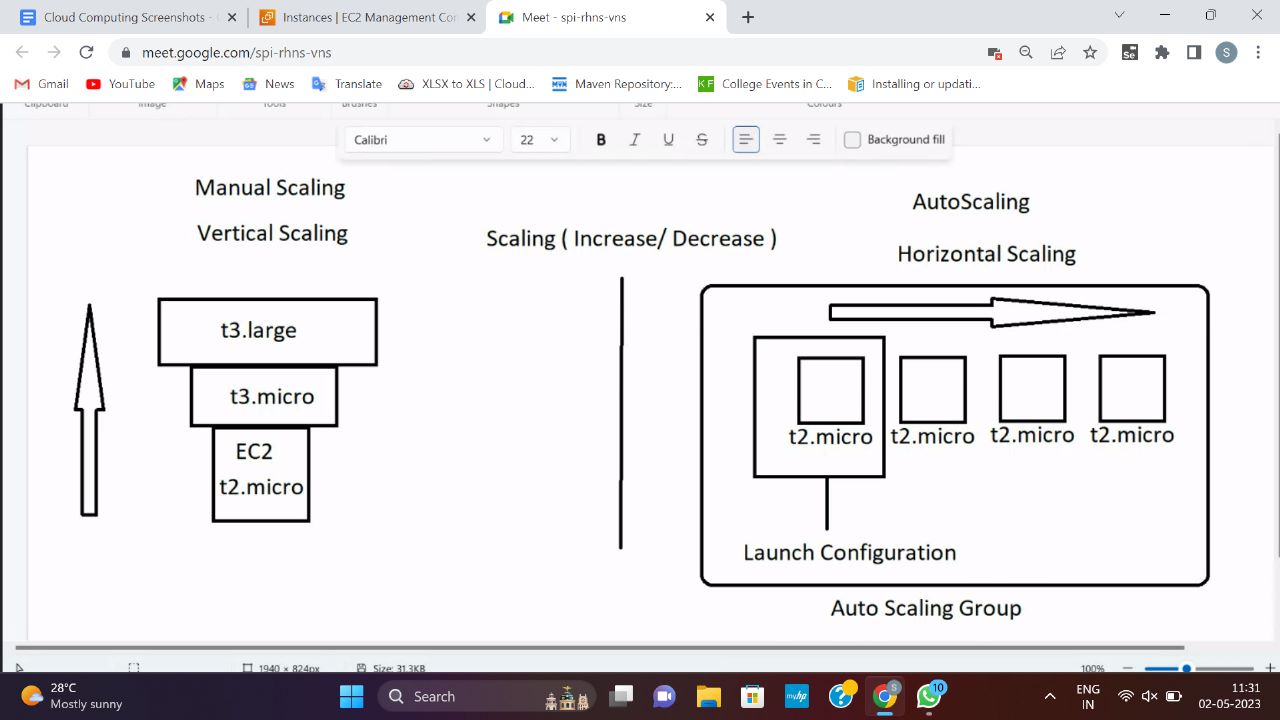
* **Horizontal Scaling**

It is auto scaling

Creating other copies of same instance

Increasing the number of instances





**Auto Scaling Groups (ASG):**

* Where the launch configurations are created
* Desired capacity
* Maximum-6

**Launch Configuration:**

* Instances are launched based on launch configurations

**Load Balancer**

automatically distributes incoming application traffic across multiple targets and virtual appliances in one or more Availability Zones (AZs).

* Health check
* Distribute the traffics

**Types of Load Balancer:**

1. **Classic Load Balancer**

Designed to run in monolithic architecture

Pre-requisite:

**Have a security group**

* Create a security group (say ‘LoadBalancer+ASG’) in EC2
* Add inbound rules (Type-HTTP, HTTPS, SSH/RDP, Source-Anywhere IPv4)

**How to create a Classic Load Balancer?**

* Go to EC2 -> Load balancing ->Load balancers->Create load balancer
* Go to Classic Load Balancer
* Click on create

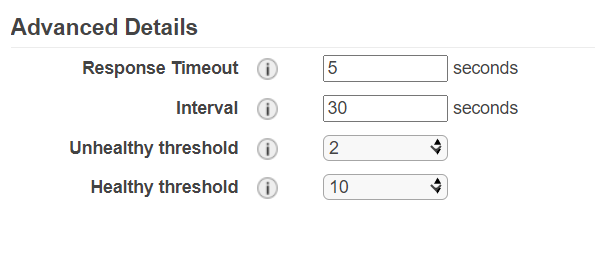
Step-1

* Enter LoadBalancer name (say ‘ClassicLoadBalancer-Frankfurt’)
* Click Next

Step-2

* Deselect default security group and select your existing security group
* Click Next

Step-4



**Response Timeout** – The number of seconds the instance is checked

**Interval** – The number of seconds of gap between each checking

**Unhealthy threshold** – The number of consecutive unhealthy response to be considered as the instance to be unhealthy

**Healthy threshold** – The number of consecutive healthy response to be considered as the instance to be healthy

* Step 6:
* Add key name (say ‘Name’) and value (say ‘ClassicLB’)

**Launch Configuration:**

* Go to EC2->Auto Scaling->Launch Configurations

**Launch configuration name**

* Enter launch configuration name (say ‘myLaunchConfiguration’)

**Amazon machine image (AMI)**

* Select AMI (Copy the AMI id and paste it here)(say AMI ID of Amazon Linux)

**Instance type**

* Select Instance Type (say ‘t2.micro’)

**Additional configuration**

* Go Advanced details
* Enter the user data

|  |
| --- |
| #!/bin/bash  yum install httpd -y  service httpd start  echo “<h1>Classic Load Balancer</h1>” > /var/www/html/index.html |

**Security group**

* Select an existing security group we created earlier (say ‘LoadBalancer+ASG’)

**Key pair**

* Select key pair
* Create launch configuration

**Auto Scaling Group**

* Go to EC2 -> Auto Scaling -> Auto Scaling Groups -> Create Auto Scaling groups

**Step 1 – Choose launch template or configuration**

**Name**

* Enter group name (say ‘MyASG’)

**Launch configuration**

* Switch to launch configuration
* Select the launch configuration we created earlier (say ‘MyLaunchConfiguration’)
* Click Next

**Step 2 – Choose instance launch option**

**Network**

* Select the VPC (say default VPC)
* Select the Availability zones and subnets (say : select all the zones for maximum availability)
* Click next

**Step 3 – Configure advanced options**

**Load Balancing**

* Select ‘Attach to an existing load balancer’

**Attach to an existing load balancer**

* Select ‘Choose from Classic Load Balancers’
* Select classic load balancer (say ‘ClassicLoadBalancer-Frankfurt’)
* Click Next

**Step 4 – Configure group size and scaling policies**

**Group size**

* Enter desired capacity (the number of instance that will be created during launch)(say ‘3’)
* Enter minimum capacity
* Enter maximum capacity (the maximum number of instances that can be scaled out)

**Step 5 – Add Tags**

**Tags**

* Add key (say ‘Name’) and value (say ‘AutoScalingApp’)

**Step 7 – Review**

* Click ‘Create auto scaling group’
* Now n number of instances (where n=desired capacity ) will be created in the EC2 instances
* After the instances are launched, each public IPs of the instances will display the content given in the user data.
* If any one instance is stopped, then alternative another instance will be launched automatically and the stopped instance will be terminated

**Application Load Balancer**

* Create an instance “Facebook” with userdata

Select existing security group

#!/bin/bash

yum install httpd -y

service httpd start

mkdir /var/www/html/facebook

echo “<h1>Facebook Application is running</h1>” > /var/www/html/facebook/index.html

To view the content, after launching open the public IP and add “/facebook” in the url

* Create another instance “Instagram”

Select existing security group

#!/bin/bash

yum install httpd -y

service httpd start

mkdir /var/www/html/instagram

echo “<h1>Facebook Application is running</h1>” > /var/www/html/instagram/index.html

To view the content, after launching open the public IP and add “/facebook” in the url

**Target Group**

* Go to EC2-> Load Balancing -> Target Groups -> Create target group
* Select target type as ‘instances’
* Enter target name (say ‘TargetGroup1’)
* In health check, enter health check path (say ’/facebook/index.html’)
* Click next
* Select the target group and click on ‘Include as pending below’

(target will be added)

* Click ‘Create target group’

Similarly create another target group ‘TargetGroup2’, with health check path ‘/Instagram/index.html’ and add the target group to ‘Instagram’ instance

**Load Balancer**

* Go to EC2-> Load Balancing -> Load Balancers -> Create load balancer
* Select application load balancer
* Enter load balancer name (say ‘MyApplicationLoadBalancer’)
* In network mapping, select availability zones ( say select all instances)
* In Security group, remove default security group and add the existing security group we created (say ‘LoadBalancer+ASG’)
* In Listeners and routing, select target group in forward to (say ‘TargetGroup1’)
* Click create load balancer
* Click view
* Wait till the state become ‘Active’
* Copy the DNS name and paste it in URL to view ‘It works’
* Add ‘/facebook’ with the URL to get the content written in facebook/index.html

**Adding listeners:**

* Go to Load balancers
* Select your load balancer
* Go to listeners
* Click on ‘1 rule’
* Click ‘Manage rules’
* Click on ‘+’ button
* Click on ‘insert rule’
* Click on ‘add condition’ -> ‘path ’-> Enter path (say ‘/facebook\*’)
* Click on ‘add action’ -> ‘forward to’ -> Select target group (say ‘TargetGroup1’)
* Click save

Similarly add another rule for instagram instance

Now when then DNS URL is opened with /facebook, facebook content will be displayed.

When the DNS URL is opened with /instagram, instagram content will be displayed

**How to delete target group?**

1. Delete instance
2. Delete load balancer
3. Delete target group