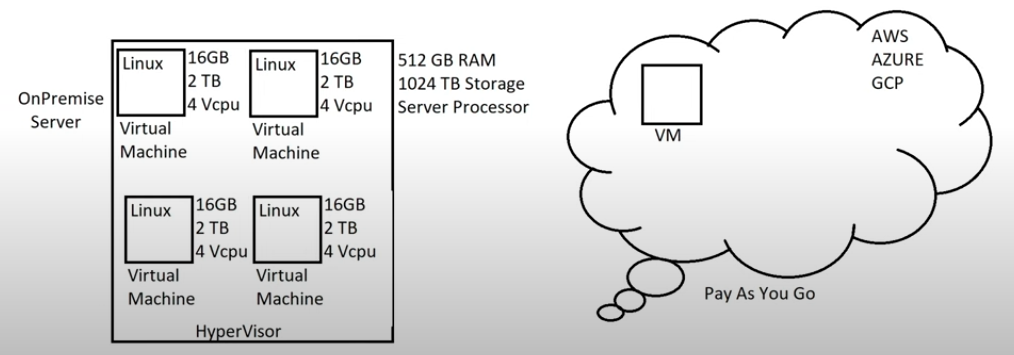
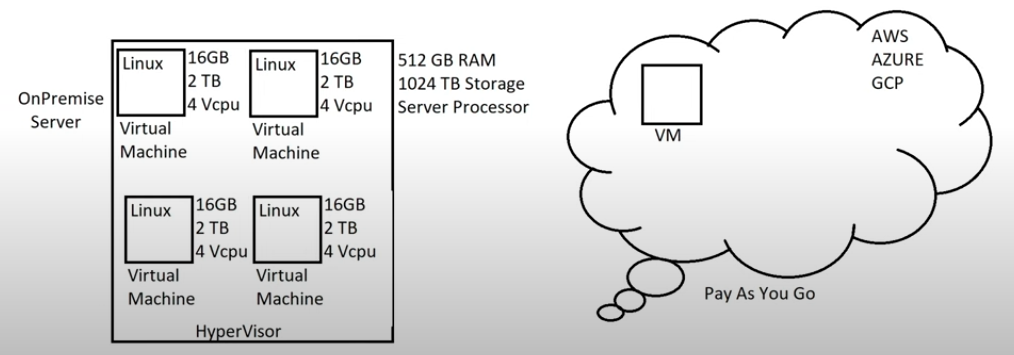
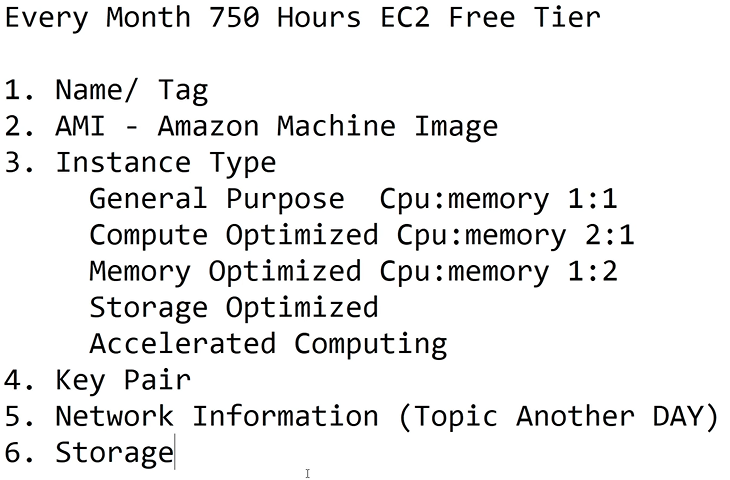
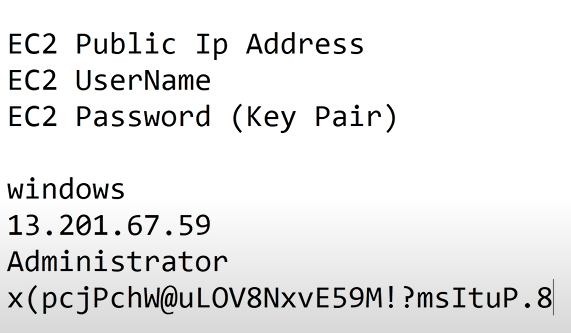
**PART-1 --> Intro to AWS EC2**

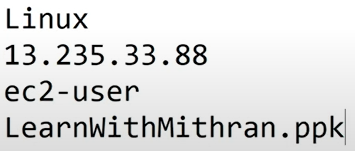


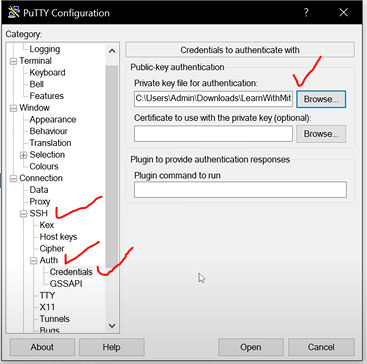


* **Windows -(RDP) Remote Desktop Protocol --> (RDC) Remote Desktop Connection .**
* **PEM --> Privacy Enhanced Mail**



* **Linux Putty connection**
* **PPK --> Putty private key**





* In putty up right click change settings --> Appearance change font size 14 bold -->colours use system colours apply.

**PART-2 --> How to host a website in aws ec2.**

* **Ec2 --> Elastic Compute Cloud.**
* Product based:- own product --> working for own company
* Service based:- working for another company
* Linux --> MobaXterm

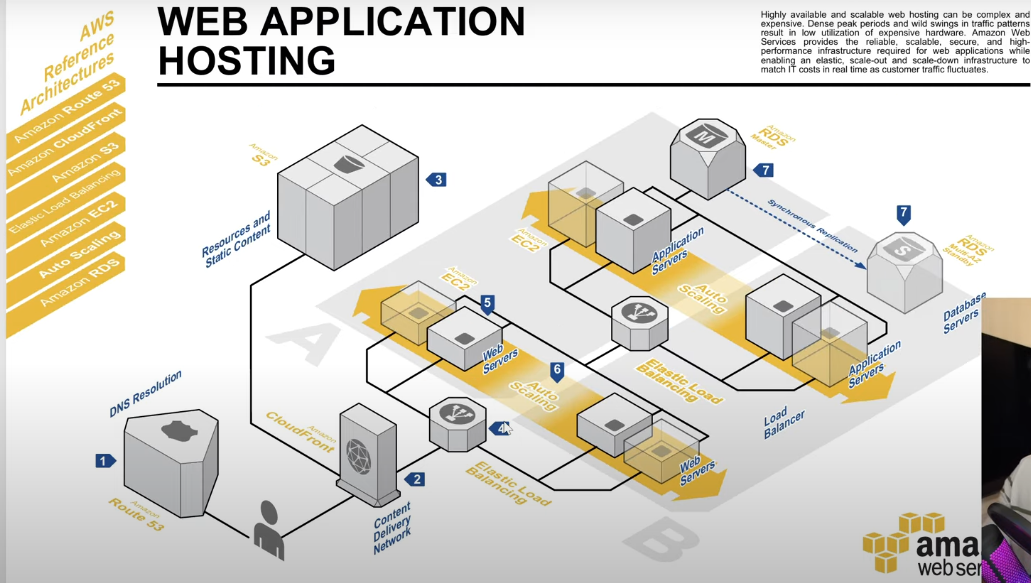
**Website ?**

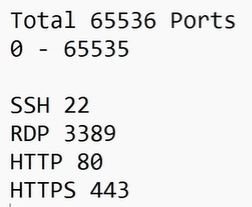
* Static website --> same content for all --> front end only
* Dynamic website --> different content to person to person --> Flipkart, Amazon, facebook, Instagram --> full stack development

**Tech Stack ?**

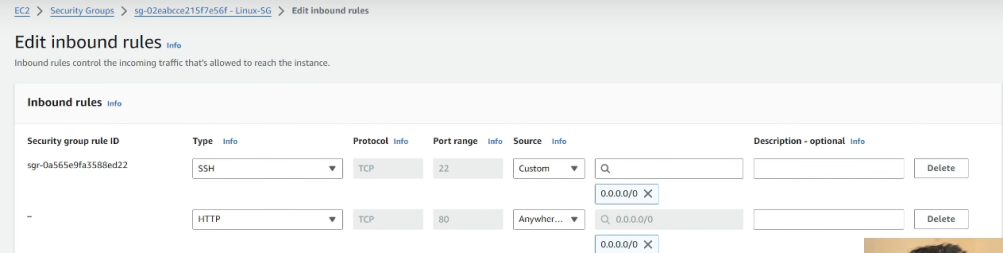
* Front End --> Html, css, Javascript
* BackEnd --> Python
* Database --> MySQL

**Three tire architecture?**





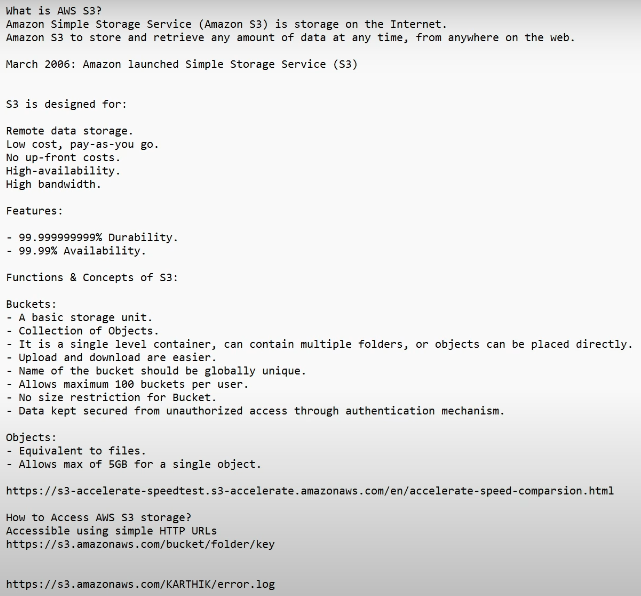
* Search in google --> Sample html templates --> sample static project.
* Software Linux --> Apache httpd
* Software Windows --> IIS
* Windows to windows file transfer --> copy and paste
* Server Manager in virtual ec2 machine.
* Add Role and feature --> next --> next --> next --> web server(IIS) --> next --> next ….. --> Install
* In Window server --> C:\inetpub\wwwroot --> copy and paste
* In linux start with ec2-user
* Sudo su - root --> sudo su - --> switch to root user
* Yum install httpd -y --> in ubantu (yum - apt)
* Exit --> from rootuser to ec2-user
* Windows --> linux (WinScp) --> download in goggle
* To edit instance :- security groups.in ec2z



* Service httpd status --> to check apache status
* Service httpd start or start httpd.service --> to start apache server
* Vi index.html --> right click --> data enter
* chmod -Rf 777 /var/www/html/ --> permision to all
* In Winscp --> public ip, username, password --> advanced --> Authentation --> private key --> ppk file .
* Sudo su - root
* Cp -r /home/ec2-user/\* /var/www/html/ --> -r means copy directories recursively --> from ec2-user to html folder.

**PART-4 --> S3 Buckets and Objects**

* **S3 --> Simple storage service**



* Amazon S3 --> Store and retrieve any amount of data from anywhere.
* S3 --> Scalable Storage in the Cloud.
* **Features** --> High Available -->Unlimited data in s3 bucket

**limitations**

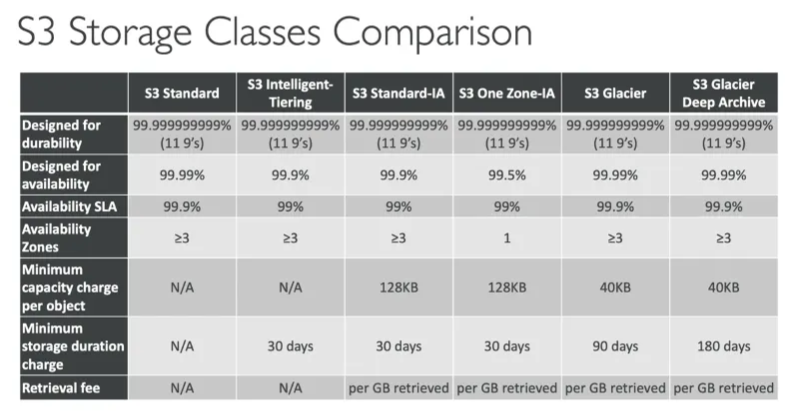
* Per aws account you can have 100 buckets
* More than 100 buckets possible we need to contact AWS or rise ticket 100 buckets not enogh for me I need 200 buckets.
* Single file cannot be more than 5TB
* 15GB free in s3
* Bucket name globally unique
* Aws charge data transfer cost.
* Bucket --> logical group
* Object --> files and folders
* Interview question:- If I delete a file from an S3 bucket, is it possible to retrieve it?
  + Yes, it is possible to retrieve a deleted file from an S3 bucket if versioning is enabled.  
    If versioning is not enabled, the file is permanently deleted and cannot be recovered.

**Create Bucket:-**

* Bucket name:- Bucket name should be globally unique namespace.
* Object Ownership:
  + ACLs disabled:- Own Aws Account
  + ACLs enabled:- Other Aws Accounts
* Block Public Access settings for this bucket
  + Default Disable--> Block all public access
  + Enable --> Public access to all
* Bucket Versioning
  + Default --> disable
  + Enable --> to be Versioning
* Tags --> Key-value pair information --> Youtube - learnwithmitran
* Default encryption --> like pdf protection with password
* Advanced settings --> object lock --> write once read many --> once file upload we cant delete if enabled.

**Upload File:-**

* Permissions --> Access control list(ACL)
  + Choose from predefined ACLs --> private or public
  + Specify individual ACL permiss
* Storage Class:-
  + - Durability means --> Extremely safe.
    - Availability means --> Immediate access.
  + Standard --> frequently accessed files.
  + Intelligent tiering --> Intelligently monitoring --> frequently access data cost bill --> rare access low cost bill.
  + Express one zone --> High performance storage for your most frequently accessed data.
  + Standard-1A --> intelligent tiering keep frequently access files in standard and infrequent access files in standard-1A.
  + standard frequently access files we can change to any other storage class but in standard-1A infrequent access files we cant change to any other storage class - Min 30 days should be in 1A.
  + One zone-1A :-Infrequent access data --> cheaper then standard-1A
  + Glacier instant retrieval --> long lived files --> min 90 days --> few times in a year
  + Glacier flexible retrieval --> getting data in minutes and hours
  + Glacier deep Archive --> rarely accessed and very low cost --> getting files in hours

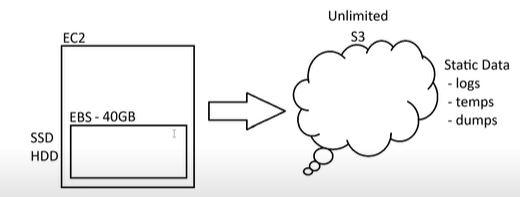




* Server side encryption
* Additional Checksums:- it will check file or folder successfully uploaded or not

Difference between s3 bucket and EBS volume, why I need to choose s3 not EBS.

* Both are Important
* We use EBS for frequent access files or programs or applications.
* We use s3 for permanent long term data storage.



**Bucket Level:-**

* Objects --> files and Folders
* Properties
  + Bucket versioning
  + Tags --> key value pair
  + Default encryption --> KMS keys
  + Intelligent tiering archive configurations --> Behavour noted --> Automate keep storages intelligently.
  + Server access login --> who are accessing servers --> cloud watch --> who write or read or modify .. etc.
  + Event notification --> create notification --> automation --> lambda.
  + Transfer acceleration --> cloud front --> High internet speed to upload a large file if enabled.
  + Object lock --> if enabled we can’t delete or modify file.
  + Requester pays --> In my aws account I keep my friend bucket --> he access or download or data transfer charges he will pay the bill not me.
  + Static website hosting
* Permissions
  + Bucket policies --> IAM
* Metrices
  + Total bucket size
  + Total number of objects
  + It is integrated to cloudwatch it is a monitoring solution.
* Management
  + Life cycle rules
    - Automatic delete files before 60 days.
    - Automatic move to cheaper storage class.
  + Replication rules
    - Backup bucket in another region.
* Access points
  + Vpc end points

**PART-5 --> IAM Users, Groups, Roles & Policies:-**

* IAM - Identity and access Management --> least Privilege access.
* IAM - Manage access to AWS resources.

**Create User:-**

* Username
* Provide user access
  + I want to create an IAM user
  + Console password
  + Create new password at next sign-in
* Permissions Options --> Attach policies directly --> Policy --> power --> Aws policies-1253
* Permissions boundary:- to control the maximum permissions for this user --> limit

**IAM signin:-**

* Account id or account alias --> create alias in aws account
* IAM username
* Password

**Create Group:-** Collection of IAM users.

* Group name
* Add users
* Attach permission policies

**Create Own Policies:-** Customer Managed Policies

**Create Inline Policies :-** user defined policies --> createing policies --> customer inline

* Select a service --> EC2
* Actions allowed --> Allow all
  + List
  + Read
  + Write
  + Permissions management
  + Tagging
  + Dependent Permissions not selected --> not ec2 supporting service like VPC
* Recources
  + All
  + Specific
* Request conditions
  + User is MFA Authenticated
  + Request from IP
* Policy name --> finally creation done

**Aws CLI for windows install --> google**

**Linux have already CLI**

**CLI have 2 ways to connect with seffvice to service**

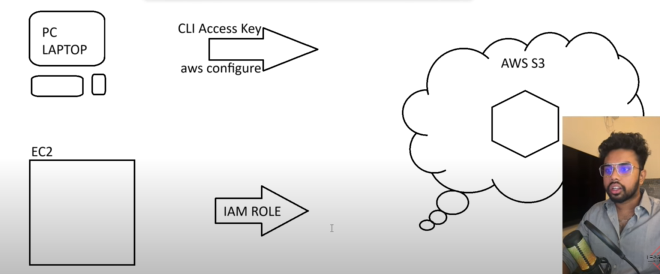
* **Access keys**
* **IAM role**

**CLI -->** user -->security credentials --> Access keys

Root user --> security credentials --> Access key

**Access keys**

* Use access keys to send programmatic calls to AWS from the AWS CLI
* Create Access key --> Command line interface(CLI) --> add s3 policies to user
* It will create 2 things
  + Access key
  + Secret access key
* In aws ec2-user --> commands
  + Aws
  + Aws s3 ls --> list s3 buckets
  + Aws configure
    - Access key
    - Secret key
    - Ap-south-1 --> region
    - Json --> output



**IAM ROLES --> Create Roles --> ec2 and s3**

* Select trusted entity --> ec2
* Attach policy --> s3
* Role name
* Go to ec2 instance --> actions --> security --> modify IAM role --> choose IAM role --> update IAM role --> success
* We need to attach policy for particular ec2 instance.

Identity based policy:- for creating a user or service --> IAM level policy

Resource based policy:- bucket level policy --> Bucket level policy

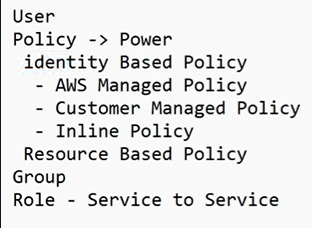
ARN --> Amazon Resource Name --> unique id --> Bucket ARN

**Cross account access** :- EC2 - S3 --> ARN

* Principal --> Role ARN
* Resource --> Bucket ARN

**TASK:-**

* User in account A
* S3 bucket in account B



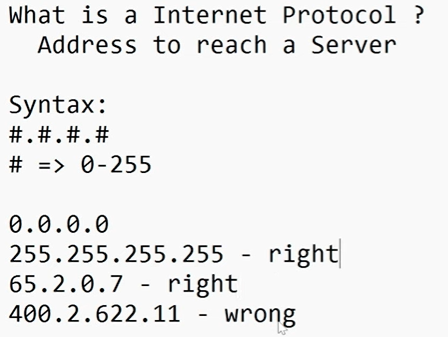


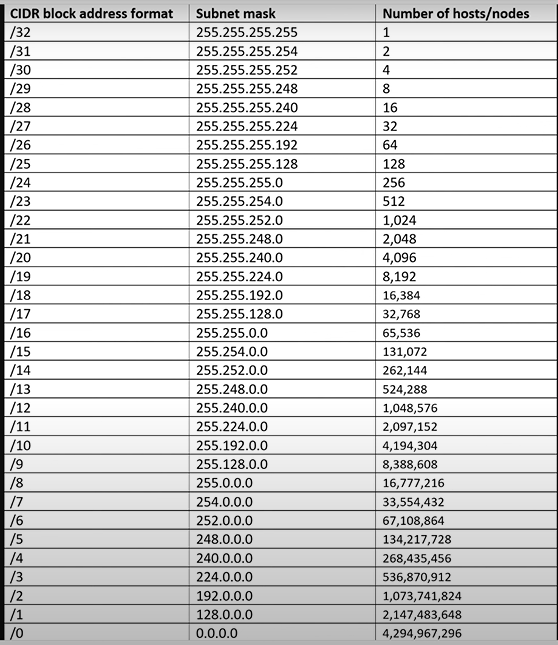
**Practice:-**

* User1 --> s3fullaccess
* User2 --> s3 fullaccess +boundry s3readonlyaccess
* User3 --> ec2fullaccess
* Access key :- AKIA4IWGK4ZZSFVALNU2
* Secret Access key:- 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

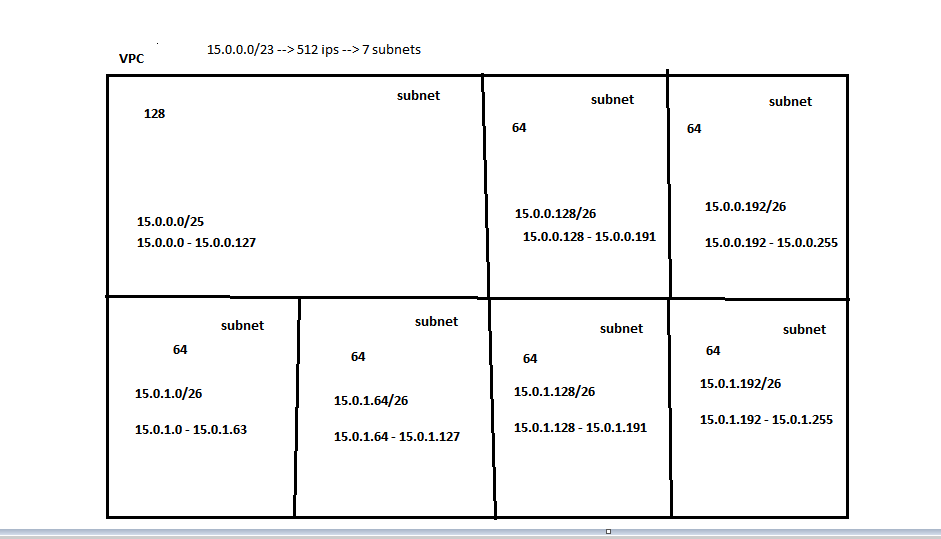
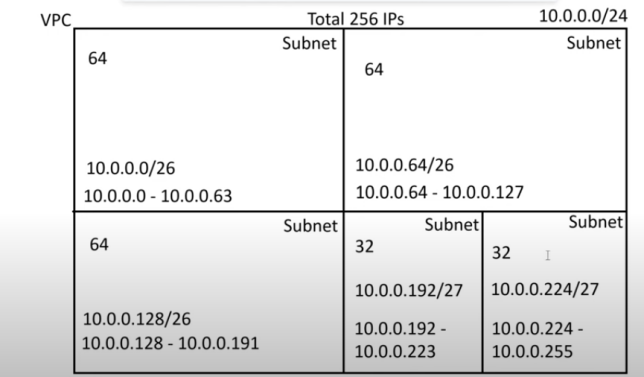
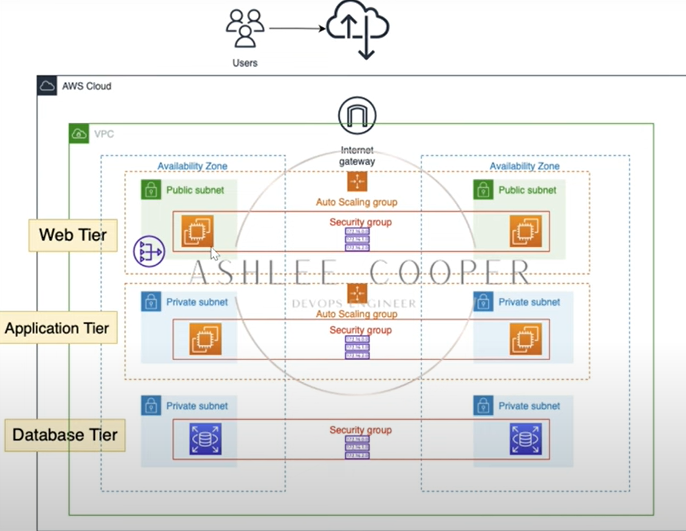
**PART-6 --> VPC-Public and private IP, CIDR Range:-**

* VPC - Virtual Private Cloud
* CIDR - Classless Inter-Domain Routing - Range of Ips



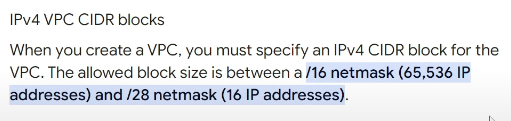


* 10.0.0.0/24 --> 10.0.0.0 - 10.0.0.255 (Total 256) --> 32-24=8 --> 2\*\*8 -->256 ips
* 10.0.0.0/23 --> 10.0.0.0 - 10.0.1.255(Total 512) --> 32-23=9 --> 2\*\*9 -->512 ips
* Public ip: reachable from internet
* Private ip: used within our office network
  + 10.0.0.0/8 --> ip addresses: 10.0.0.0 - 10.255.255.255
  + 172.16.0.0/12 --> ip addresses: 172.16.0.0 - 172.31.255.255
  + 192.168.0.0/16 --> ip addresses: 192.168.0.0 - 192.168.255.255



# **PART-7 --> Build AWS VPC Subnets Route Table Internet Gateway & Nat Gateway:-**

* VPC --> Isolated cloud resources



* Biggest vpc size - 16
* Smallest vpc size -28

**Create VPC**

* Resources to create --> VPC only
* Name tag --> vpc name
* IPv4 CIDR block --> 10.0.0.0/16 --> 65536 total IPs --> CIDR block size must be between /16 and /28
* Tenancy --> rack in data centers
* Tags --> key value pair

**Create Subnets**

* VPC id
* Subnet settings
  + Subnet name
  + Availability zone --> ex:- mumbai have 3 zones 1a,1b & 1c
  + IPv4 VPC CIDR block -->VPC CIDR --> 10.0.0.0/16
  + Ipv4 subnet CIDR block --> small part in VPC --> 256 Ips
  + Add new subnet --> if we need another subnet

**Create internet gateway**

* Name
* Select gateway --> actions --> attach to VPC
* Attaching gateway to subnet it will become public subnet

**Create Route table**

* Name
* Select a VPC

**Connect Route table to subnet**

* Select a Subnet --> route table --> edit route table association --> Route table id.

(or)

* Select route table --> subnet associations -->edit subnet association --> select subnet.

**Connect Route table to internet gateway**

* Select a route table --> edit routes
  + 0.0.0.0/0 --> Anywhere from the world --> any IP
  + Internet gateway --> select IGW

**For Testing**

* Launch 2 ec2 instance
  + Public
  + Private
* Network settings --> not default VPC --> created VPC
* Select subnet --> public or private
* Auto-assign public ip --> enabled --> if private disabled
* Select security group

**Create security group**

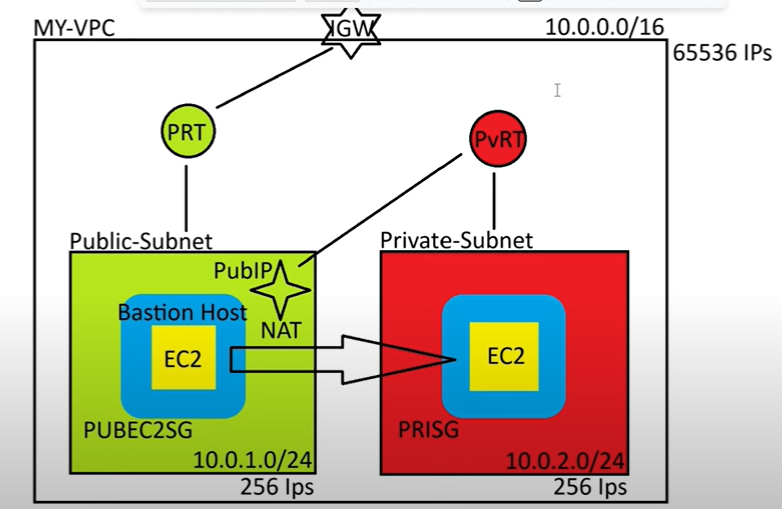
* Group name
* Vpc --> select VPC
* Add inbound rule --> for windows RDP --> for linux SSH --> HTTP and HTTPS
* Outbound rule --> any traffic anywhere

**Create NAT gateway**

* Name
* Select subnet --> having internet
* Connectivity type --> public
* Elastic ip allocation ID --> allocate elastic IP --> elastic public ip

**Connect Route table to NAT gateway**

* Select a route table --> edit routes
  + 0.0.0.0/0 --> Anywhere from the world --> any IP
  + NAT gateway --> select NAT



**Linux to Linux connection:-**

* Ssh /root/Sample/Window.pem ec2-user@191.16.4.5 --> password, username and IP.
* Chmod 400 /root/Sample/Window.pem --> 400 private readonly--> permissions

# **PART-9 --> Master AWS Load Balancers & Auto Scaling Groups:-**

* **Load Balancer**:- It is used to distribute the incoming trafic

**Create Clasic Load Balancer:-**

* Load Balancer name
* Scheme
  + Internet-facing
  + Internal
* Select VPC
* Select Availability Zones
* Select security group
* Listeners and routing
* Health checks --> ping path
* Add instances
* Attributes --> default
* Load balancer tags --> optional

**DNS name is the Load Balancer Ip address**

**Load balancer algorithms:-**

* **Static load balancing**
  + Round-robin --> 123 123 123 --> one chance
  + Weighted round-robin --> priority like 60% remaining 40%
  + IP hash
* **Dinamic load balancing**
  + Least connection --> new request to least server
  + Weighted least connection --> priority
  + Least response time --> immediate response
  + Resource-based

**Application Load balancing(ALB)**

* ALB will work on http and https
* ALB will support Path based routing, host based routing, sticky sessions ex:- 5min after logout
* SSL cerificate will attach only in ALB

**Network Load balancing(NLB)**

* NLB will work on TCP and UDP
* High performance load balancer
* Static ip and elastic ip we can attach only in NLB

**Create Target Group:- After createing instances**

* Target Type - instances
* Target group name
* Protocol : port
* Ip address type - IPV4
* Select VPC
* Protocol version
* Health checks --> path --> /green/index.html
* Select instance

**Create Application Load Balancer:-**

* Load Balancer name
* Scheme
  + Internet-facing
  + Internal
* Load balancer ip address type --> IPV4
* Select VPC
* Select Availability Zones
* Select security group
* Listeners and routing --> default action green target group

**Path based routing configuration:-**

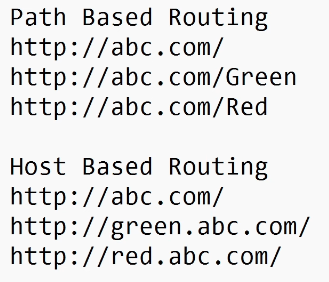
* Click load balancer --> click listeners and rules --> manage rules --> edit rule --> add rule
  + Rule name --> green
  + Add condition --> path --> /green\*
  + Actions --> forward to Green target group
  + Priority --> 1 to 50,000
  + Create
* Add rule --> red

**Auto Scaling -** Automatically it will create new Ec2 instances

* Horizontal scaling --> Increasing the servers --> ex. 2 4 6 8 instances
* Vertical scaling --> upgrading the hardware configuration--> ex. 16 / i5 - 32 / i7 --> increasing the capacity of server
* AWS supports only Horizontal scaling

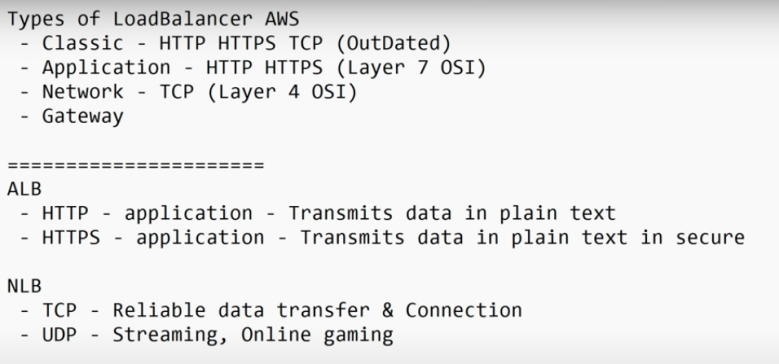
**Create Autoscaling Groups - Highly available**

* Autoscaling Groups Name
* **Create Launch template**
  + Launch template Name
  + Description
  + Quick start --> choose AMI
  + Instance type --> t2 micro
  + Key pair --> private key
  + Select security group
  + User data --> shell script
* Select launch template
* If need override launch template
* Select VPC - default
* Select Availability Zones and subnets
* If need load balancer --> attach
* Desired capacity --> servers
* Scaling --> min servers and max servers --> min<desired and max>desired
* Remaining all - default





# 

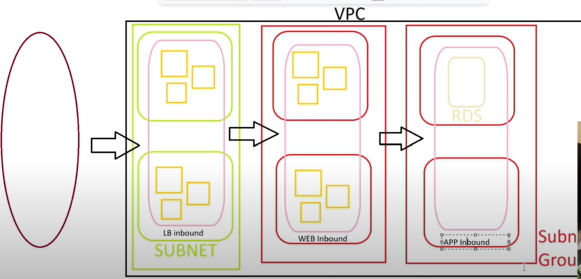


# **PART-10 --> Relational Database Service (RDS)**

* RDS - Managed Relational Database Service

**Create Database**

* Choose a database creation method
* Database engine --> MySQL
* Templates --> Free tier
* Availability and durability
  + Multi-AZ Cluster --> 2 or more than 2 instances
  + Multi-AZ DB instance --> one standby instance
  + Single DB instance --> create single DB instance with no standby DB instance.
* DB instance identifier --> RDBMS name
* Username
* Credentials or password
* Instance configuration
* Storage
* If needed enable storage autoscaling
* Connectivity
  + If needed connect to ec2 instance
  + Network type - IPV4
  + VPC -default --> after db created you cannot change VPC
  + DB subnet group --> default
  + Public access
  + VPC security group
    - New VPC security group name
    - Availability Zone - no preference
* Tags
* Database authentication --> password authentication
* Additional configuration
  + Database options --> initial database name
  + DB parameter group
  + Option group
  + Backup --> off
  + Encryption --> off
  + Log exports --> all off
  + Maintenance --> off
  + Maintenance window --> no preference
  + Delete protection --> off --> if on I am not able to delete database
* Create --> done
* Click database --> Connectivity and security --> end point
* If we need to connect ec2 we need
  + Endpoint
  + Port
  + User
  + Password
* EC2 and RDS should be in same VPC or diffent using VPC peering
* In EC2 switch to root user connect to RDBMS
  + Change security group add inbound rule --> type-mysql and source-ec2 private ip/32 or security group id --> save rule
  + Yum install mysql -y
  + **Mysql -h endpoint -P port 3306 -u username -p** enter **password**
    - h for endpoint
    - capital P for port number
    - U means user
    - Small p means password
  + Mysql opened
* Multi-AZ is for high availability, whereas a Read Replica is primarily for handling or reducing the load on the master.



**Create DB subnet group:-**

* Name
* Vpc
* Availability zone
* Select subnets

**Create parameter group:-**

* Name
* Engine type --> mysql
* Parameter group family

**Create Option group:-**

* Name
* Engine type --> mysql
* Engine version

**EC2 FTP Connection**

* Sudo su --> switch ec2 to root user
* Yum update -y
* Yum install vsftpd -y
* Vi /etc/vsftpd/vsftpd.conf
  + anonymous\_enable=YES
  + ascii\_upload\_enable=YES
  + ascii\_download\_enable=YES
* Systemctl start vsftpd
* Systemctl status vsftpd
* Useradd susmitha\_ftp
* Passwd susmitha\_ftp --> new password and confirm password
* Security groups in ec2 should be enabled
  + Custom TCP --> 20-21
  + Custom TCP --> 1024-1048
* Sudo yum remove -y vsftpd --> to uninstall package
* Vi /etc/vsftpd/vsftpd.conf
  + Pasv\_enable=YES
  + Pasv\_min\_port=1024
  + Pasv\_max\_port=1048
  + Port\_enabled=YES
  + Pasv\_addr\_resolve=NO
* Systemctl restart vsftpd
* Check win WinSCP
* sudo chmod 777 /home/susmitha\_ftp --> full control

# **PART-11 --> AWS Route 53**