**EC2:**

**Static hosting using EC2:**

**Connecting Linux Instance =>**

1. Save keypair in .pem format

2. Convert .pem to n in PuttyGen and Save file

3. Go to Putty and add hostname

4. Go to "Auth" => "Credentials" => Select the .ppk file that is converted in step 2

5. Click Open

Launching a website =>

1. yum update -y

2. yum install httpd -y

3. cd /var/www/html

4. Add index.html

5. service httpd start

6. chkconfig on

#!/bin/bash

sudo su

yum update -y

yum install httpd -y

systemctl start httpd

systemctl enable httpd

echo "<h1>Hello World from $(hostname -f)</h1>" > /var/www/html/index.html

Copy public IP in browser

**AMI:**

AMIs are built for a specific AWS Region, they're unique for each AWS Region. You can't launch an EC2 instance using an AMI in another AWS Region, but you can copy the AMI to the target AWS Region and then use it to create your EC2 instances.

**EBS:**

**To encrypt an unencrypted EBS volume**

Create an EBS snapshot of the volume

Encrypt the EBS snapshot using copy

Create a new EBS volume from the snapshot (volume will also be encrypted

Attach the encrypted volume to the original instance

**S3:**

**S3 Replication:**

1. After you enable Replication, only new objects are replicated
2. You can replicate existing objects using S3 Batch Replication.

Replicates existing objects and objects that failed replication

1. For delete operation, can replicate delete markers from source to target(optional)
2. NO chaining of replication
3. **Create a bucket**

**aws s3 mb s3://my-useless-bucket-101**

1. **Copying a local file to S3**

**aws s3 cp devops.pdf s3://my-useless-bucket-101**

1. **Copying a file from S3 to S3**

aws s3 cp s3://mybucket/test.txt s3://mybucket/test2.txt

1. **Copying an S3 object to a local file**

aws s3 cp s3://mybucket/test.txt test2.txt

1. **Delete bucket**

aws s3 rb s3://mybucket

1. **Delete an S3 object**

aws s3 rm s3://mybucket/test2.txt

1. **Delete all contents in a bucket**

aws s3 rm s3://mybucket \

--recursive

**To create EC2 instance**

aws ec2 run-instances --image-id ami-051f8a213df8bc089 --instance-type t2.micro --count 1 --subnet-id subnet-0eaf568399e81080b --key-name KP-101 --security-group-ids sg-01aa873de7b8b606b --credit-specification CpuCredits=unlimited

**To describe the EC2 instance**

aws ec2 describe-instances --instance-ids i-09a142af690624276

**Policies:**

**Trust Policy: Which principal(Service, granting third-party) can assume role, under which conditions**

**Identity based: What permissions a user/role/group is allowed or denied**

**Resource based: What permissions we can perform on resource**

**Service control policy:** Service control policies (SCPs) enable central administration over the permissions that determine which services and actions that all identities (users and roles) can use across the accounts in your organization.

**Route53**

1. **Domain Name Purchasing in Route53**
2. **Hosting Website =>** Mapping Domain Name Records - A, AAAA, CNAME
3. **Subdomain Configuration with CNAME**
4. **Alias =>** used when we don’t have an IP address to the resource and we can’t use CNAME in that case as it can be used with subdomain only.
5. **How to use TXT record**
6. **Health check of the domains**
7. **Routing Policy =>** When you create a record, you choose a routing policy, which determines how Amazon Route 53 responds to queries

* Simple Routing Policy
* Weighted (Weight is added to the record according to that request is serverd
* Geolocation
* Latency
* Failover
* Multivalue

1. Private hosted zone

**Route53**

Purchasing domain name

Hosting website or resource using different types of records(A, AAAA, CNAME, Alias)

Health Checks

Types of routing policies

**Load Balancer:**

1. Classic Load Balancer
2. Application Load Balancer

ALB hands on

ALB supports targets such as : EC2, Lambda, ECS, authenticate users before routing

ALB does routing based on rules added as listeners:

1. Path in url - example.com/users
2. Hostname in url - one.exmple.com
3. Query String and Heards - [example.com/users?id=1234&platform=Mobile](http://example.com/users?id=1234&platform=Mobile)

3. Network Load Balancer

1. Gateway Load Balancer

Cross Zone Load Balancing

**Auto Scaling Group**

Launch Template

Auto Scaling Hands on

Auto Scaling Policies

1. Predictive
2. Dynamic - simple, target, step

Auto Scaling + ALB Hands on

**CloudFront**

CloudFront Hands On

Invalidations in CloudFront

Accessing the EC2/ELB through CloudFront only => VPC => Managed prefix list => cloudfront.origin facing

Creating origins and adding behavior for path based routing with multiple origins

Path based routing => (/api => EC2 , /images => s3

1. Create a distribution with origin as EC2 instance.
2. After deployed state go to origin and add origin as S3 bucket
3. Go to Behaviour and add path and origin in it. Do it for EC2 and S3 both

**Two EC2 instances => one bucket => get image from internet and add it to the bucket => Only one instance should be able to access it.**

**User Scripts:**

#!/bin/bash

sudo su

yum update -y

yum install httpd -y

systemctl start httpd

systemctl enable httpd

echo "<h1>Hello World from $(hostname -f)</h1>" > /var/www/html/index.html

#!/bin/bash

wget https://img.freepik.com/free-photo/vibrant-colors-nature-close-up-wet-purple-daisy-generated-by-artificial-intellingence\_25030-63819.jpg

aws configure set aws\_access\_key\_id "AKIAZQ3DT6ERHVXJXUHT" && aws configure set aws\_secret\_access\_key "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" && aws configure set region "us-east-1" && aws configure set output "text"

aws s3 cp /home/ec2-user/vibrant-colors-nature-close-up-wet-purple-daisy-generated-by-artificial-intellingence\_25030-63819.jpg s3://my-aws-bucket-100

#!/bin/bash

sudo su

yum update -y

mkdir hello

touch hiii

**IAM**

**EC2**

**S3**

**Load Balancer and ASG**

**Databases**

**Route53**

**CloudFront**

**VPC**

**Note:**

1. VPC is region oriented
2. Subnet is AZ oriented
3. Security Group is associated with VPC. Use the security group and ec2 instances should have the same security group.
4. Where internet gateway is attached is called public subnet
5. If an instance is in a private subnet no command will work like apt-get update and all.
6. IgW allows instances with public IPs to access the internet whereas NAT Gateway allows instances with private IPs to access the internet.

**Creating a private subnet and public subnet and enabling communication between them.**

1. Create four subnets, 2 private, 2 public.
2. Create a public route table with internet gateway attached and a private route table with no internet gateway.
3. Edit the route table association of private subnet and change route table to private one.

**Accessing the private instance from public instance**

1. Connect to public instance using ssh -i key.pem ec2-user@private-ip
2. Create a .pem file and copy paste the secret key in it
3. Run ssh command
4. Internet is still not accessible

**Accessing internet in private instance (Preconfigured AMI, Must have Elastic IP and Stop src/dest check) ami-0780b09c119334593 (amzn-ami-vpc-nat-2018.03.0.20220503.0-x86\_64-ebs)**

1. Create nat instance using amzn-nat-ami, add it to public subnet and attach security group as All traffic and enable public ip
2. Create Elastic IP and associate it to nat instance
3. Edit the route table of private instances. Add rule 0.0… and select nat instance
4. Go to nat instance and stop src/dest check

We can avoid creating NAT instances and instead use NAT gateway directly.

**NACL**

1. NACL controls traffic in/out of subnet
2. A subnet can only be associated with one NACL at a time.
3. You can associate a network ACL with multiple subnets.
4. A network ACL has inbound rules and outbound rules. Each rule can either allow or deny traffic. Each rule has a number from 1 to 32766.
5. NACLs are stateless, which means that information about previously sent or received traffic is not saved. If, for example, you create a NACL rule to allow specific inbound traffic to a subnet, responses to that traffic are not automatically allowed.

**NAT Gateway**

Instances in a private subnet can connect to services outside your VPC but external services cannot initiate a connection with those instances.

1. Create a NAT Gateway in a different public subnet with internet gateway
2. Allocate Elastic IP to NAT Gateway
3. Edit route table of private instance from 0.0…. To nat gateway

**VPC Peering**

1. A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses.
2. You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account. The VPCs can be in different Regions (also known as an inter-Region VPC peering connection).
3. Create two VPC with a public subnet in one VPC and private subnet in another VPC
4. Add one instance in the public subnet and one in private subnet.
5. Create VPC Peering connection
6. Edit route table of both VPCs
7. Ping from public to private

**Transit Gateway**

A transit gateway is a network transit hub that you can use to interconnect your virtual private clouds (VPCs) and on-premises networks.

**VPC Flow Logs**

1. VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC.
2. Flow log data can be published to the following locations: Amazon CloudWatch Logs, Amazon S3, or Amazon Data Firehose

**Route Table**

* Destination—The range of IP addresses where you want traffic to go (destination CIDR). For example, an external corporate network with the CIDR 172.16.0.0/12.
* Target—The gateway, network interface, or connection through which to send the destination traffic; for example, an internet gateway.

The following is an example of the route tables that enables communication between instances in two peered VPCs, VPC A and VPC B. Each table has a local route and a route that sends traffic for the peer VPC to the VPC peering connection.

| **Route table** | **Destination** | **Target** |
| --- | --- | --- |
| VPC A | *VPC A CIDR* | Local |
| *VPC B CIDR* | pcx-*11112222* |
| VPC B | *VPC B CIDR* | Local |
| *VPC A CIDR* | pcx-*11112222* |

**Security Group**

Inbound - source (From which source traffic is allowed to the instance)

Outbound - destination (To which destination we can send traffic to)

**VPC Endpoint**

1. A VPC endpoint enables customers to privately connect to supported AWS services and VPC endpoint services powered by AWS PrivateLink. (Connection between EC2 and S3 privately)
2. Amazon VPC instances do not require public IP addresses to communicate with resources of the service.
3. Traffic between an Amazon VPC and a service does not leave the Amazon network.

Two types of endpoints:

1. Interface endpoints - creates ENI
2. Gateway endpoints - adds route to route table

**Site-to-Site VPN**

1. VPN connection refers to the connection between your VPC and your own on-premises network.
2. By default, instances that you launch into an Amazon VPC can't communicate with your own (remote) network.
3. You can enable access to your remote network from your VPC by creating an AWS Site-to-Site VPN (Site-to-Site VPN) connection, and configuring routing to pass traffic through the connection.

Three things to configure:

1. Customer Gateway (Customer side)
2. Virtual Private Gateway (AWS side) - Need to attach it to VPC - Enable Route Propagation in the route table
3. VPN Connection (Connecting both)

**Egress Only Internet Gateway**

1. Same as Internet Gateway but for IPv6 traffic
2. An egress-only internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows outbound communication over IPv6 from instances in your VPC to the internet, and prevents the internet from initiating an IPv6 connection with your instances.

**SNS**

**SQS**

**Kinesis**

**Lambda**

**ECS**

**EKS**

**Cloud Watch**

**Cloud Trail**

**Change port of SSH from 22 to 222**

1. Add security group rule
2. Sudo vi /etc/ssh/sshd\_config
3. Change port from 22 to 222
4. Sudo service sshd restart
5. Ssh -i key.pem -p 222 ubuntu@public-ip

Creating public and private key => ssh-keygen

**Encryption and decryption using public and private key:**

1. Create private key

openssl genrsa -out private\_key.key 512

1. Create public key

openssl rsa -in private.key -pubout -out public\_key.key

1. Encrypt the file with public key

openssl rsautl -encrypt -inkey public\_key.key -pubin -in myfile.txt -out encrypted\_file.enc

1. Decrypt the file using private key

openssl rsautl -decrypt -inkey private\_key.key -in encrypted\_file.enc -out decrypted.txt

Hands on - VPC Endpoint

VPN

Egress only IG

Direct connect

**Amazon Linux Mysql-client installation:**

sudo wget https://dev.mysql.com/get/mysql80-community-release-el9-1.noarch.rpm

sudo dnf install mysql80-community-release-el9-1.noarch.rpm -y

sudo rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2023

sudo dnf install mysql-community-client -y

mysql -h my-mariadb-instance.ctkuw0i8gtw8.us-east-1.rds.amazonaws.com -P 3306 -u admin -p

**Session Manager**

Use case: Assume that you have an EC2 Instance in a private subnet and need to connect to the Instance without using SSH over the internet.

How will you do it?

Solution: Create an IAM role for EC2 Instance and attach the AmazonSSMManagedInstanceCore policy.

**Ephemeral Ports**

**WHy do we need to set up Ephemeral Ports:**

<https://remy-nts.medium.com/aws-nacl-why-the-need-to-set-ephemeral-ports-range-for-outbound-rules-50ee93986555>

<https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html>

DSR:08/05/2024

Hello,

**i. What I did today:**

**Did Linux Commands:**

Edit files in Linux - vi, nano

Working with folders

Linux head & tail commands

Comparing files in Linux - cmp and diff

Compress and Decompress files in linux

Download files in Linux - wget & curl

Linux Package Management - yum, rpm

Linux Service Control - systemctl

Switch user and sudo

Access Remote Linux Server (scp, ssh)

Linux Files Permissions (chmod)

**ii. What will I do tomorrow-**

Practical on SQS and SNS.

iii. Any roadblocks -

No roadblocks

**Deploying Web Application with RDS**

1. Update Ubuntu

Apt update

1. Install apache, php, mysqli, mysql server

Apt install apache2

Apt install php

Apt install php-mysqli

Apt instal mysql-server

1. Remove index.html from /var/www/html and Add web-app files to /var/www/html

1. Change permissions

sudo chmod -R 755 /var/www/html

1. Login to mysql

sudo mysql -u root -p

1. Create database

CREATE DATABASE registration;

1. Create a root user and grant privileges (NO NEED TO DO THIS IF USING RDS)

CREATE USER root@'localhost' IDENTIFIED BY password;

GRANT ALL PRIVILEGES ON registration.\* TO root@'localhost';

FLUSH PRIVILEGES;

1. Create table ‘user’
2. Restart server

systemctl restart apache2

1. If anything goes wrong check error log of server

sudo cat /var/log/apache2/error.log

**Linux commands**

1. **ls -lt**
2. **Ls -ltr**
3. **ls -lh**
4. **Less => to display and search in file**
5. **More => display file page by page**
6. **Sort**
7. **Grep “pattern” filename => searching in file**
8. **Ls x\* => all files starting with x**
9. **Ls \*.txt => files ending with .txt**
10. **Shuf filename => shuffle**
11. **Wc filename => count of line, word and character**
12. **Cmp file1 file2**
13. **Diff -u file1 file2**
14. **find path -name filename**
15. **Cal => Calendar**
16. **Date**
17. **Bc => calculator**
18. **Script => record commands in typescript file && Ctrl+D to stop script**
19. **Alias l=”ls -ltr” => creating shortcut of a command**
20. **Gzip -k file-name => To compress file**
21. **Gzip -d file-name => Decompress file**
22. **Tar -czf filename.tar.gz folder-name/ => Compress a folder**
23. **Tar -xzf filename.tar.gz => Decompress folder**
24. **Zip myfiles.zip file1 file2 => Compressing multiple files in one zip file**
25. **Unzip myfiles.zip => Decompress**
26. **Unzip -l myfiles.zip => listing files in compressed folder**
27. **Wget url => download file from internet**
28. **Wget -o new\_name url => so save file from different name**
29. **Curl url => call an API**
30. **Apt or yum/dnf => install an application, ubuntu = apt, linux = yum**
31. **Rpm -qa | grep app-name => check if application is installed or not**
32. **Dnf list installed => check if application is installed or not**
33. **Apt search package-name => list available packages to install on ubuntu**
34. **yum/dnf list available => list available packages to install on linux**
35. **Systemctl => start, stop, restart, check status, enable/disable service**
36. **Systemctl start/stop service\_name => start/stop service on linux**
37. **Systemctl status service\_name => check status of service**
38. **Service service\_name status => check status of service**
39. **Systemctl list-units –type=service –all => List all services**
40. **Printenv => listing environment variables**
41. **Export JAVA\_HOME = “/ur/lib/jvm/java\_v” => adding environment variable (temporary)**
42. **Add env variable in ./bashrc file = source ~/.bash\_profile => adding environment variable (permanently)**
43. **Su => changing to root user**
44. **Ssh user@hostname => accessing remote linux user**
45. **Scp file user@hostname:/home/paul => uploading files to remote server**
46. **Chmod a+rwx filename => changing permissions of file**
47. **Chown user file => change owner**
48. **Free => check free RAM space**
49. **Top => %memory and cpu utilization**
50. **Du => disk utilization**
51. **Df => filesystem available and disk space allocated**
52. **Lscpu => check cpu/core/thread info**
53. **Arch => type of architecture**
54. **Lsblk => list of storage devices, disk partition**
55. **Uname -a => os name**
56. **Pgrep process-name => return process ids**
57. **Kill -9 process-id =>kill process**
58. **Jobs => list active jobs**

**1 ls**

**2 sudo su**

**3 pwd**

**4 whoami**

**5 date**

**6 date +%D**

**7 date +%T**

**8 ls**

**9 ssh-keygen**

**10 ls -lt**

**11 ls -ltr**

**12 ls -lh**

**13 mkdir new-folder**

**14 ls -lh**

**15 vi names.txt**

**16 cat names.txt**

**17 less names.txt**

**18 more names.txt**

**19 vi linux.txt**

**20 cat linux.txt**

**21 ls**

**22 rmdir new-folder**

**23 ls**

**24 cp names.txt csv\_files**

**25 cd csv\_files**

**26 ls**

**27 cp ../linux.txt .**

**28 ls**

**29 cp linux.txt myfile**

**30 cat myfile**

**31 mv myfile my-new-file**

**32 ls**

**33 vi names.txt**

**34 vi sort-demo.txt**

**35 sort sort-demo.txt**

**36 vi sort-demo.txt**

**37 sort sort-demo.txt | uniq**

**38 cat sort-demo.txt**

**39 split -l 3 sort-demo.txt**

**40 ls**

**41 cat xaa**

**42 grep "Neha" names.txt**

**43 grep "Neha"|"book" names.txt**

**44 grep "Neha | book" names.txt**

**45 grep "Neha|books" names.txt**

**46 ls \*x**

**47 ls x\***

**48 clear**

**49 ls xa\***

**50 ls \*.txt**

**51 touch file{1..2}**

**52 ls**

**53 shuf sort-demo.txt**

**54 clear**

**55 wc -l names.txt**

**56 wc names.txt**

**57 wc -w names.txt**

**58 wc -c names.txt**

**59 cmp names.txt sort-demo.txt**

**60 diff -u names.txt sort-demo.txt**

**61 find names.txt**

**62 cd ..**

**63 cd**

**64 clear**

**65 ls**

**66 find ./ -name names.txt**

**67 find ./ -name \*.csv**

**68 find ./ -name \*.txt**

**69 clear**

**70 locate names.txt**

**71 updatedb**

**72 clear**

**73 history**

**74 ls --help**

**75 ls -R**

**76 man sort**

**77 man ls**

**78 which ls**

**79 which cat**

**80 cd /usr/bin**

**81 ls**

**82 cat ls**

**83 clear**

**84 sudo su**

**85 exit**

**86 pwd**

**87 echo "hello"**

**88 ls**

**89 history**

**90 sudo su**

**91 ls**

**92 ls -lh**

**93 ls -d**

**94 ls -D**

**95 ls -lD**

**96 ls -R**

**97 bc**

**98 sudo apt install bc**

**99 bc**

**100 clear**

**101 cal**

**102 uptime**

**103 script**

**104 ls**

**105 cat typescript**

**106 vi typescript**

**107 clear**

**108 alias l="ls -ltr"**

**109 l**

**110 gzip -k names.txt**

**111 ls**

**112 gzip -d names.txt.gz**

**113 ls**

**114 ls csv\_files**

**115 tar -czf compress.tar.gz csv\_files**

**116 ls**

**117 rmdir csv\_files**

**118 rm -rf csv\_files**

**119 tar -xzf compress.tar.gz**

**120 ls**

**121 curl http://numbersapi.com/random**

**122 yum install nginx**

**123 sudo su**

**124 sudo yum install nginx**

**125 sudo apt install nginx**

**126 clear**

**127 dnf list installed**

**128 rpm -qa | grep sql**

**129 sudo apt install rpm**

**130 rpm -qa | grep sql**

**131 apt search nginx**

**132 systemctl status nginix**

**133 systemctl status nginx**

**134 service status nginx**

**135 systemctl status nginx.service**

**136 service ngnix status**

**137 service nginx status**

**138 systemctl status firewalld.service**

**139 service firewalld.service status**

**140 service ngnix.service status**

**141 service nginx.service status**

**142 clear**

**143 service nginx status**

**144 systemctl stop nginx**

**145 systemctl status nginx.service**

**146 systemctl stop nginx.service**

**147 systemctl status nginx.service**

**148 sudo systemctl stop nginx.service**

**149 systemctl status nginx.service**

**150 systemctl list-units --type=service --all**

**151 clear**

**152 printenv**

**153 clear**

**154 history**

**155 free**

**156 free -h**

**157 free -th**

**158 top**

**159 du -h**

**160 df -h**

**161 hostname**

**162 lscpu**

**163 arch**

**164 lsblk**

**165 uname -a**

**166 history**

**167 cat /etc/os-release**

**168 ps -ef | grep ngnix**

**169 ps -ef | grep httpd**

**170 sudo systemctl start ngnix**

**171 sudo systemctl start nginx**

**172 ps -ef | grep httpd**

**173 ps -ef | grep ngnix**

**174 ps -ef | grep nginx**

**175 ls**

**176 pgrep nginx**

**177 kill -9 58902**

**178 sudo kill -9 58902**

**179 service nginx status**

**180 jobs**

**181 sleep 5s**

**182 jobs**

**183 ifconfig**

**184 ipconfig**

**185 www.google.com**

**186 ping www.google.com**

**187 netstat**

**188 sudo apt install net-tools**

**189 ifconfig**

**190 ipconfig**

**191 netstat**

**192 tracert**

**193 tracert google.com**

**194 clear**

**195 free**

**196 free -th**

**197 traceroute www.google.com**

**198 sudo apt install traceroute**

**199 traceroute www.google.com**

**200 clear**

**201 useradd rosy**

**202 su**

**203 sudo su**

**204 su rosy**

**205 history**

**206 sudo su**

**207 exit**

**208 history**