
SERVER
HARDENING
FOR SECURITY
AND AVAILABILITY

Content

1) Server Hardening Introduction

2) Server Hardening Types

3) Server Hardening Best Practices

4) EC2 Instance Hardening measure 1

5) EC2 Instance Hardening measure 2

6) EC2 Instance Hardening measure 3

7) EC2 Instance Hardening measure 4

What is Server Hardening

- ✓ Systems hardening is a collection of tools, techniques, and best practices to reduce vulnerability in technology applications, systems, infrastructure, firmware, and other areas. The goal of systems hardening is to reduce security risk by eliminating potential attack vectors and condensing the system's attack surface. By removing superfluous programs, accounts functions, applications, ports, permissions, access, etc. attackers and malware have fewer opportunities to gain a foothold within your IT ecosystem.

- ✓ Systems hardening demands a methodical approach to audit, identify, close, and control potential security vulnerabilities throughout your organization.

- ✓ Although the principles of system hardening are universal, specific tools and techniques do vary depending on the type of hardening you are carrying out. System hardening is needed throughout the lifecycle of technology, from initial installation, through configuration, maintenance, and support, to end-of-life decommissioning.

Types of system hardening activities

- Application hardening
- Operating system hardening
- Server hardening
- Database hardening
- Network hardening

What is the attack surface

- The aim of server hardening is to reduce the attack surface of the server. The attack surface is all the different points where an attacker can to attempt to access or damage the server. This includes all network interfaces and installed software. By removing software that is not needed and by configuring the remaining software to maximise security the attack surface can be reduced. As a result, an attacker has fewer opportunities to compromise the server.

Systems Hardening to Reduce the “Attack Surface”

The “attack surface” is the combination of all the potential flaws and backdoors in technology that can be exploited by hackers. These vulnerabilities can occur in multiple ways, including:

- Default and hardcoded passwords
- Passwords and other credentials stored in plain text files
- Unpatched software and firmware vulnerabilities
- Poorly configured BIOS, firewalls, ports, servers, switches, routers, or other parts of the infrastructure
- Unencrypted network traffic or data at rest
- Lack, or deficiency, of privileged access controls

Best Practices for Systems Hardening

- **Audit your existing systems:** Carry out a comprehensive audit of your existing technology. Use penetration testing, vulnerability scanning, configuration management, and other security auditing tools to find flaws in the system and prioritize fixes. Conduct system hardening assessments against resources using industry standards from NIST, Microsoft, CIS, DISA, etc.

- **Create a strategy for systems hardening:** You do not need to harden all of your systems at once. Instead, create a strategy and plan based on risks identified within your technology ecosystem, and use a phased approach to remediate the biggest flaws.

- **Patch vulnerabilities immediately:** Ensure that you have an automated and comprehensive vulnerability identification and patching system in place.

- **Network hardening:** Ensure your firewall is properly configured and that all rules are regularly audited; secure remote access points and users; block any unused or unneeded open network ports; disable and remove unnecessary protocols and services; implement access lists; encrypt network traffic.

Best Practices for Systems Hardening

- **Server hardening:** Put all servers in a secure datacenter; never test hardening on production servers; always harden servers before connecting them to the internet or external networks; avoid installing unnecessary software on a server; segregate servers appropriately; ensure superuser and administrative shares are properly set up, and that rights and access are limited in line with the principle of least privilege.

- **Application hardening:** Remove any components or functions you do not need; restrict access to applications based on user roles and context (such as with application control); remove all sample files and default passwords. Application passwords should then be managed via an application password management/privileged password management solution, that enforces password best practices (password rotation, length, etc.). Hardening of applications should also entail inspecting integrations with other applications and systems, and removing, or reducing, unnecessary integration components and privileges.

.

Best Practices for Systems Hardening

- **Database hardening:** Create admin restrictions, such as by controlling privileged access, on what users can do in a database; turn on node checking to verify applications and users; encrypt database information—both in transit and at rest; enforce secure passwords; introduce role-based access control (RBAC) privileges; remove unused accounts;

- **Operating system hardening:** Apply OS updates, service packs, and patches automatically; remove unnecessary drivers, file sharing, libraries, software, services, and functionality; encrypt local storage; tighten registry and other systems permissions; log all activity, errors, and warnings; implement privileged user controls.

- **Eliminate unnecessary accounts and privileges:** Enforce least privilege by removing unnecessary accounts (such as orphaned accounts and unused accounts) and privileges throughout your IT infrastructure

Benefits of Systems Hardening

- **Enhanced system functionality:** Since fewer programs and less functionality means there is less risk of operational issues, misconfigurations, incompatibilities, and compromise.

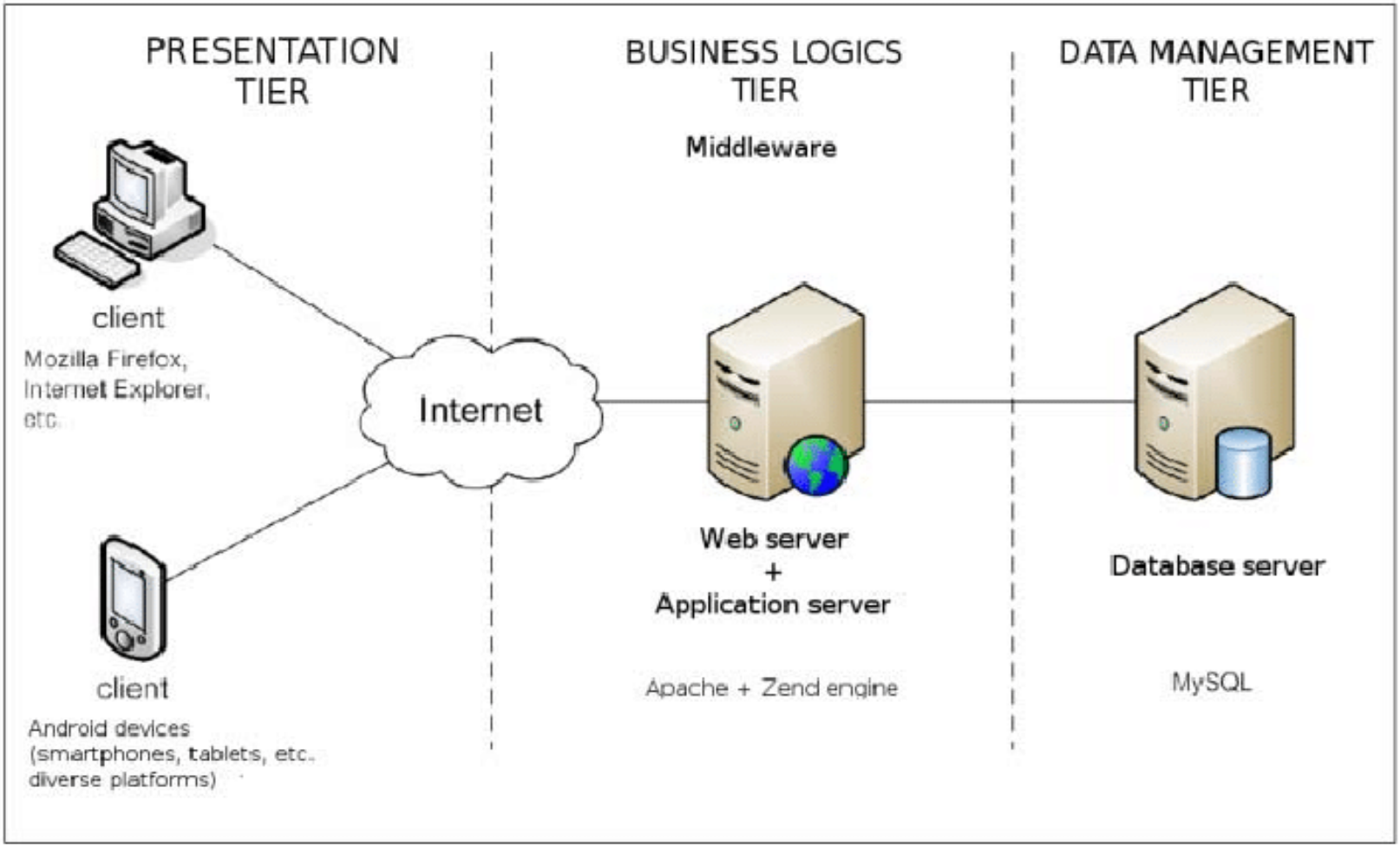
- **Significantly improved security:** A reduced attack surface translates into a lower risk of data breaches, unauthorized access, systems hacking, or malware.

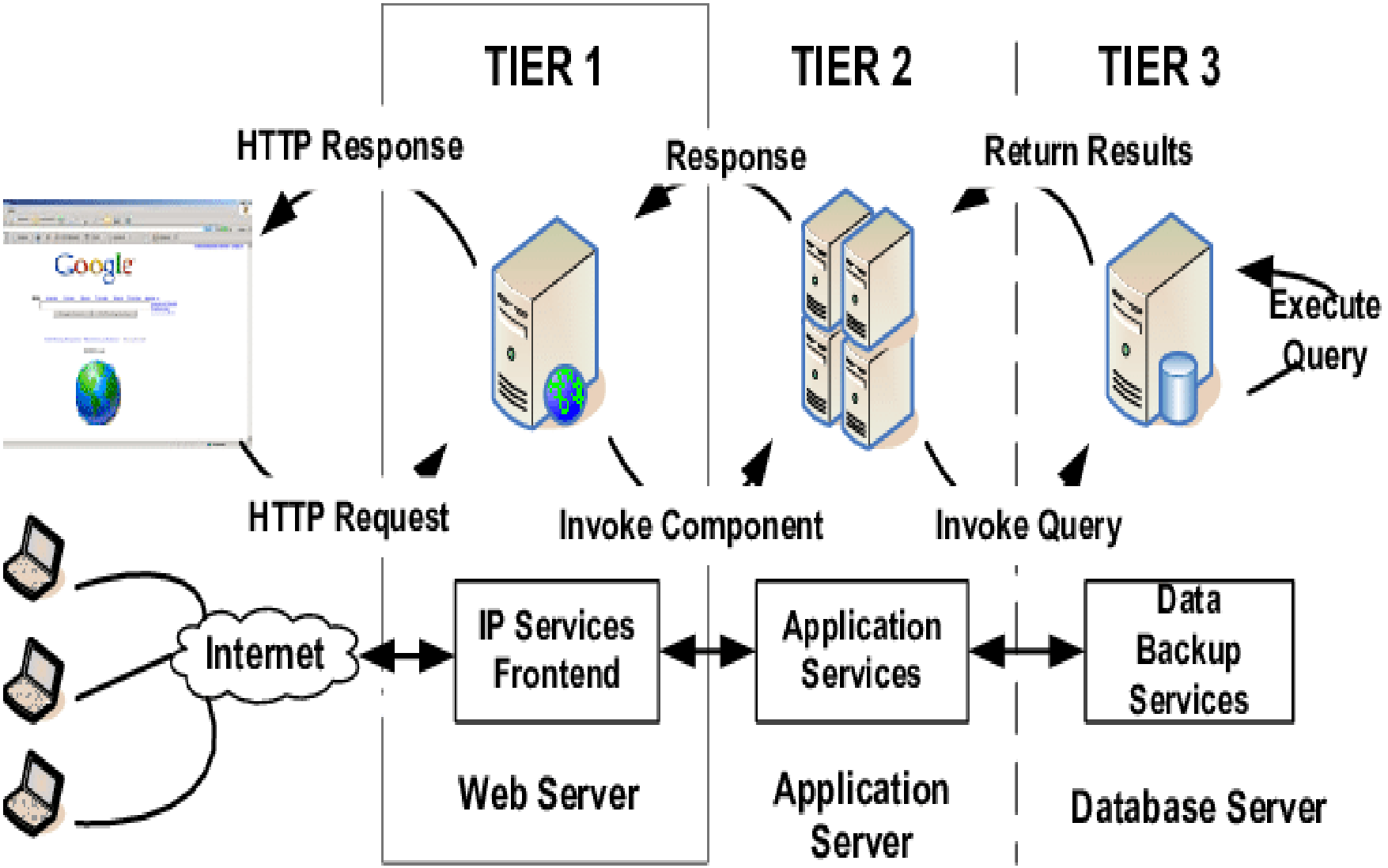
- **Simplified compliance and auditability:** Fewer programs and accounts coupled with a less complex environment means auditing the environment will usually be more transparent and straightforward.

EC2 Instance Hardening – 1(Public and Private Network)

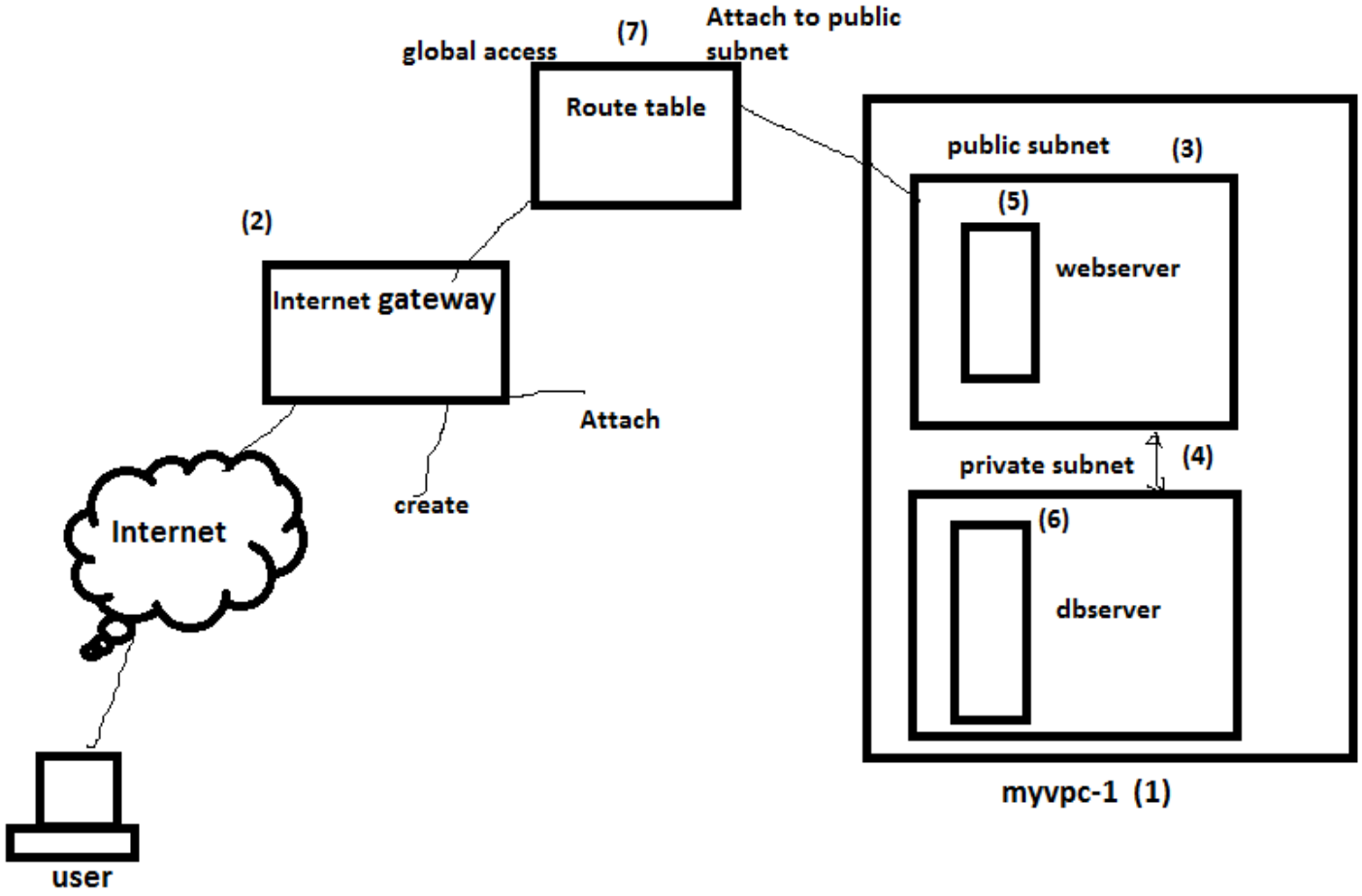
How separating server roles improves security

- The goal of sever hardening is to remove all unnecessary components and access to the server in order to maximise its security. This is easiest when a server has a single job to do such as being either a web server or a database server. A web server needs to be visible to the internet whereas a database server needs to be more protected, it will often be visible only to the web servers or application servers and not directly connected to the internet.
- If a single server is hosting both a webserver and a database there is clearly a conflict in the security requirements of the two different applications


PRESENTATION TIER
BUSINESS LOGICS TIER
Middleware
DATA MANAGEMENT TIER
client
Mozilla Firefox, Internet Explorer, etc.
Internet
Web server + Application server
Apache + Zend engine
Database server
MySQL
client
Android devices (smartphones, tablets, etc. diverse platforms)


TIER 1
TIER 2
TIER 3
HTTP Response
Response
Return Results
Google
Execute Query
HTTP Request
Invoke Component
Invoke Query
Internet
IP Services Frontend
Application Services
Data Backup Services
Web Server
Application Server
Database Server

Hands on – Configure custom VPC with public and private network



VPC Configuration Steps

Region	Mumbai	
VPC ID	10.100.0.0/16	
Public Subnet ID	10.100.1.0/24	
Private Subnet ID	10.100.2.0/24	

In Mumbai

1) Open AWS Console –Services – VPC – Your VPC – Create VPC- Type name : project1-vpc – IP CIDR block -10.100.0.0/16 – Create VPC

VPC Configuration Steps

2) Subnets – Create Subnets –Select VPC ID – subnet name: Public-subnet –

Availiblity zone : ap-south-1a – IPV4 CIDR block:10.100.1.0/24 – Create Subnet

3) Subnets – Create Subnets –Select VPC ID – subnet name: Private-subnet –

Availiblity zone : ap-south-1b – IPV4 CIDR block:10.100.1=2.0/24 – Create Subnet

VPC Configuration Steps

3) Internet gateway – Create Internet gateway –Tag – project1-int-gtw -- Create Internet gateway

Then go to action –Attach to VPC – Available VPCs –select project1-vpc – Attach Internet gateway

4) Route table –Create Route table – Name tag: Project1-RT1 – VPC - project1-vpc – Create

After creating select it – subnet association –edit –select public-subnet ---save

Go to Routes –Edit –Add route – 0.0.0.0/0 --- Target – Internet gateway - project1-int-gtw – save routes

Launching Web and DB instance

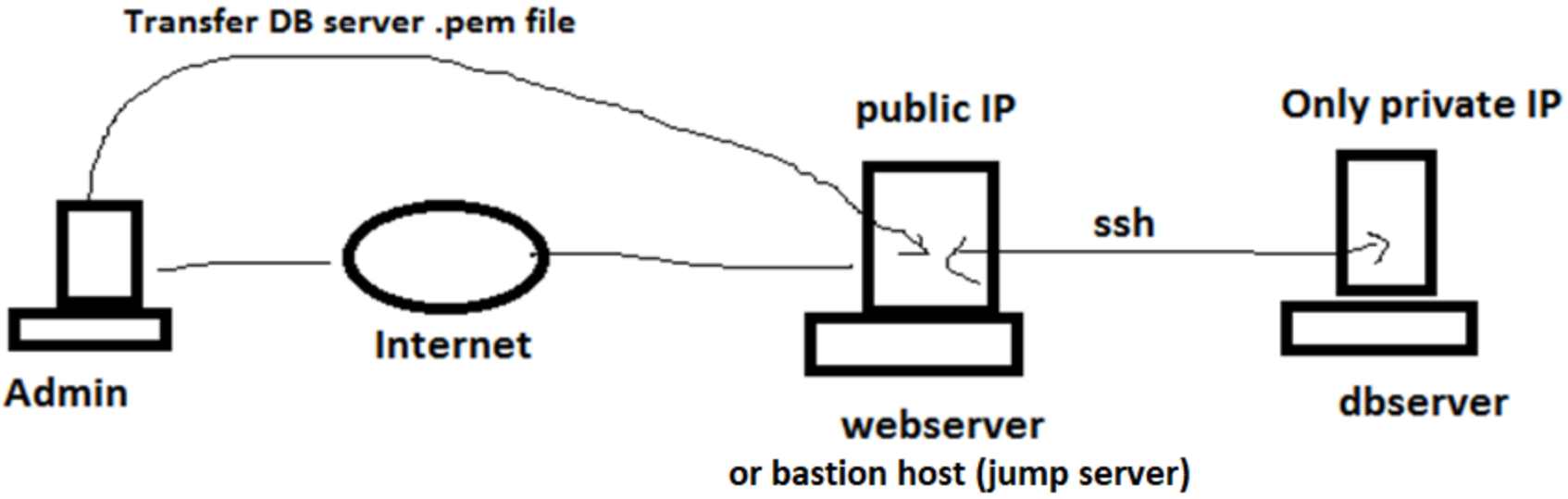
1) Launch instance ----- Network – Project1-VPC, Subnet – public subnet , Auto assign public IP – Enabled ----next ------Security group –select ssh and http –next – next –Launch

2) Launch instance ----- Network – Project1-VPC, Subnet – private subnet , Auto assign public IP – Disabled ----next ------Security group –select -ssh –next – next –Launch

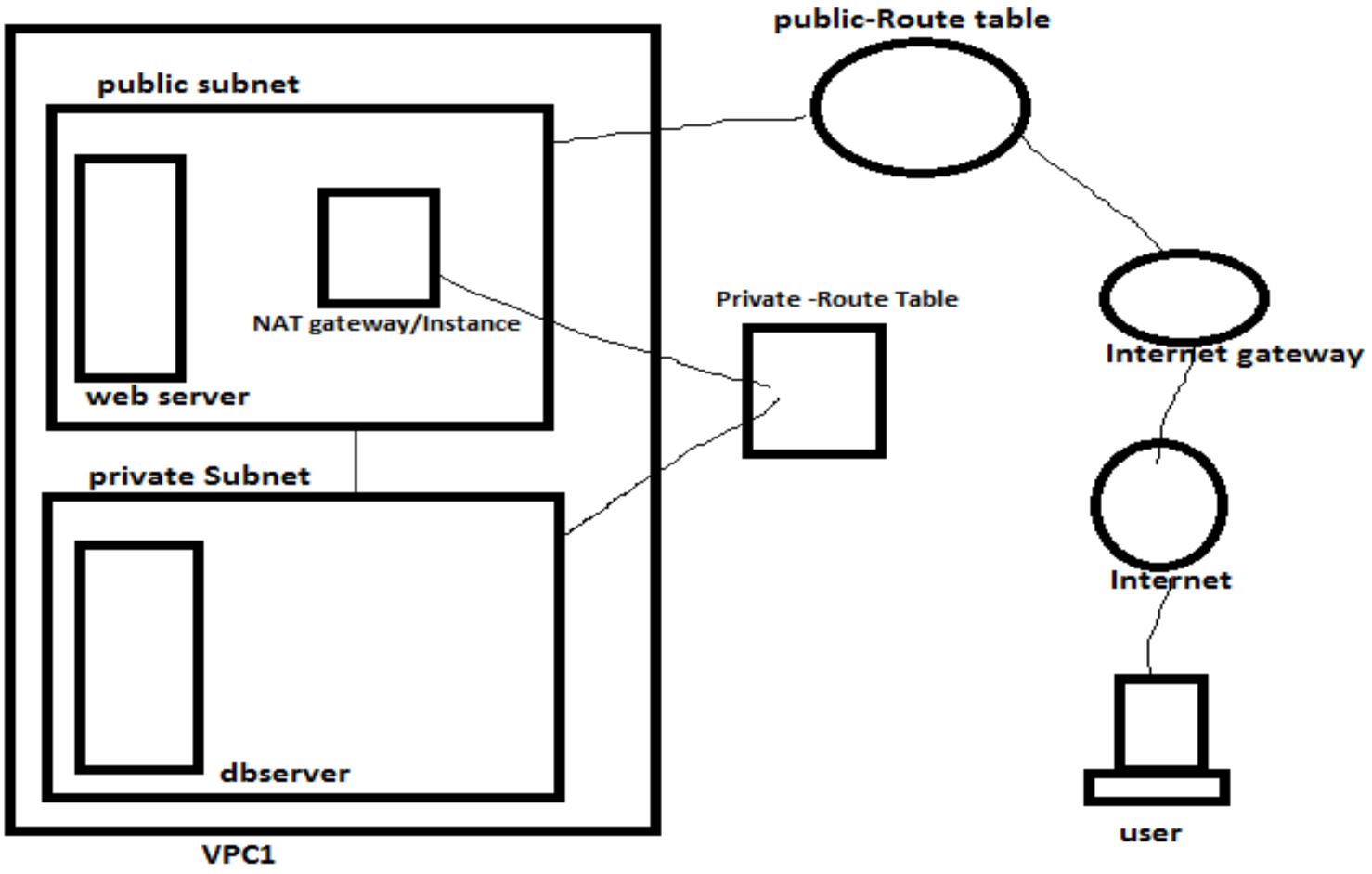
Try to connect both server.

Web server connectivity possible but not for DB server

Connecting Dbserver through Webserver



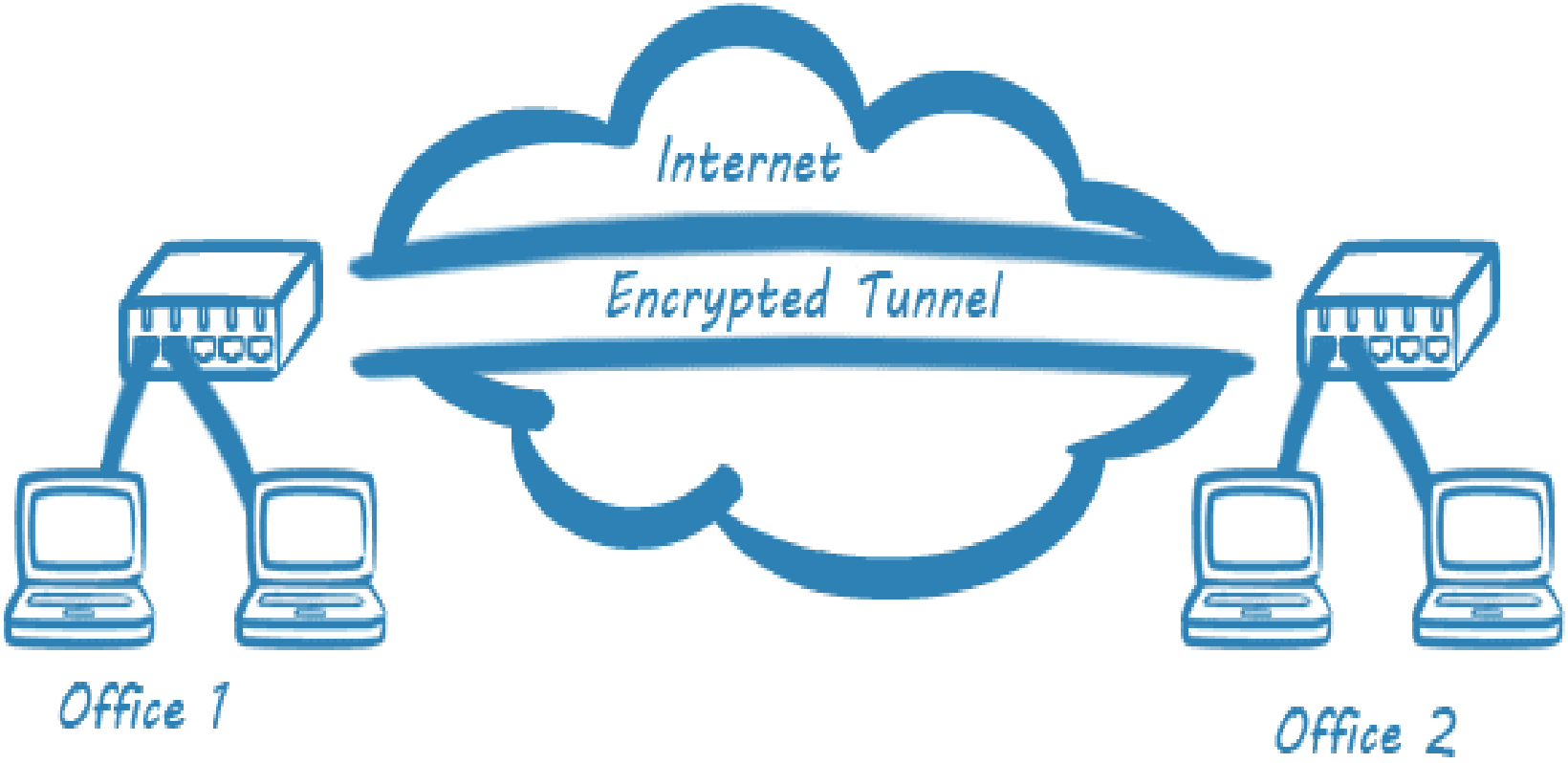
Provide Internet Connectivity (Outbound) to Private Subnet



	NAT Gateway	NAT Instance
Managed	Managed by AWS	Managed by you
Availability	Highly available within an AZ	Not highly available (would require scripting)
Bandwidth	Up to 45 Gbps	Depends on the bandwidth of the EC2 instance type selected
Maintenance	Managed by AWS	Managed by you
Performance	Optimized for NAT	Amazon Linux AMI configured to perform NAT
Public IP	Elastic IP that cannot be detached	Elastic IP that can be detached
Security Groups	Cannot associate with a Security Group	Can associate with a Security Group
Bastion Host	Not supported	Can be used as a bastion host

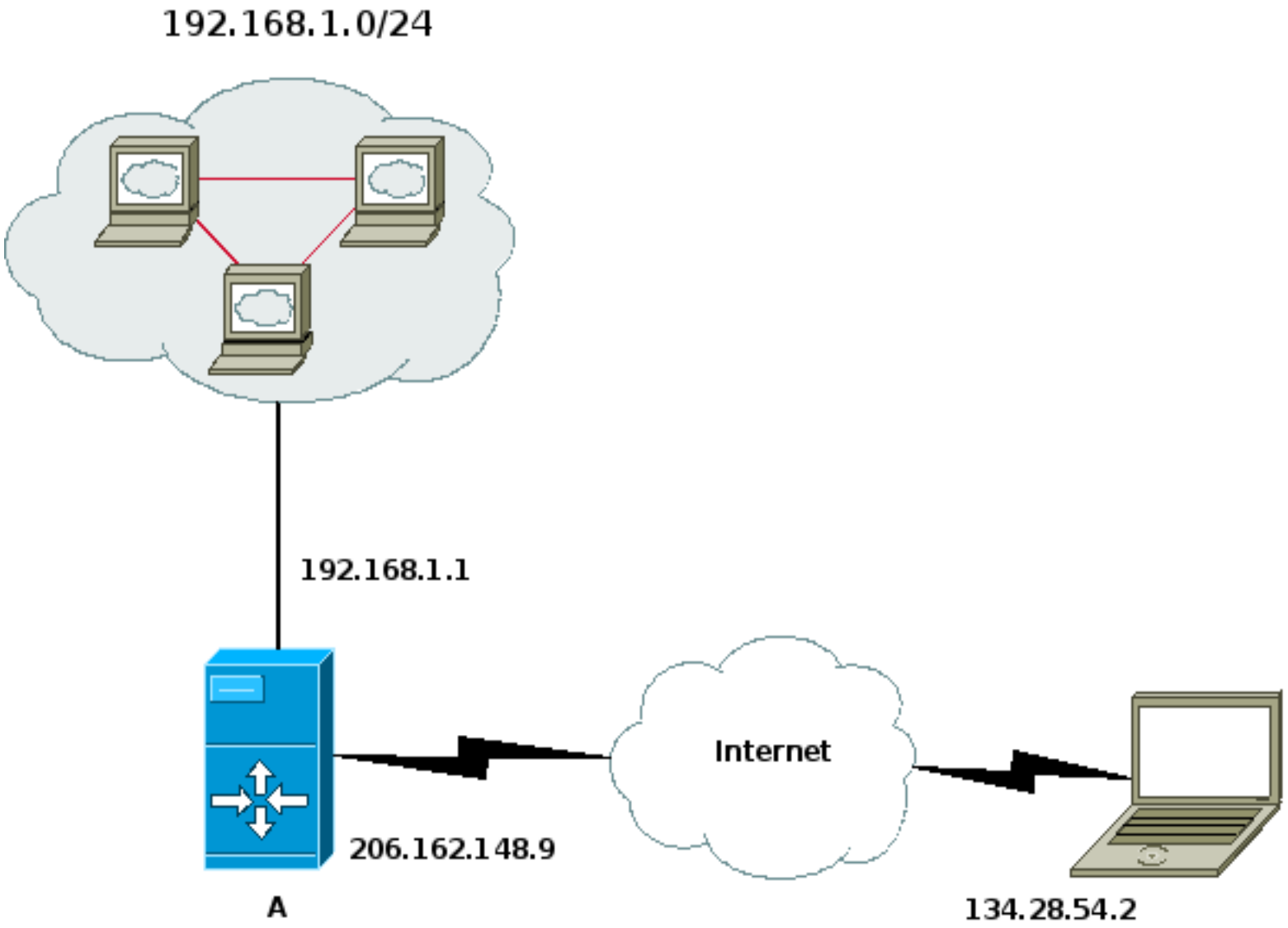
Instance Hardening -2 : VPN connection

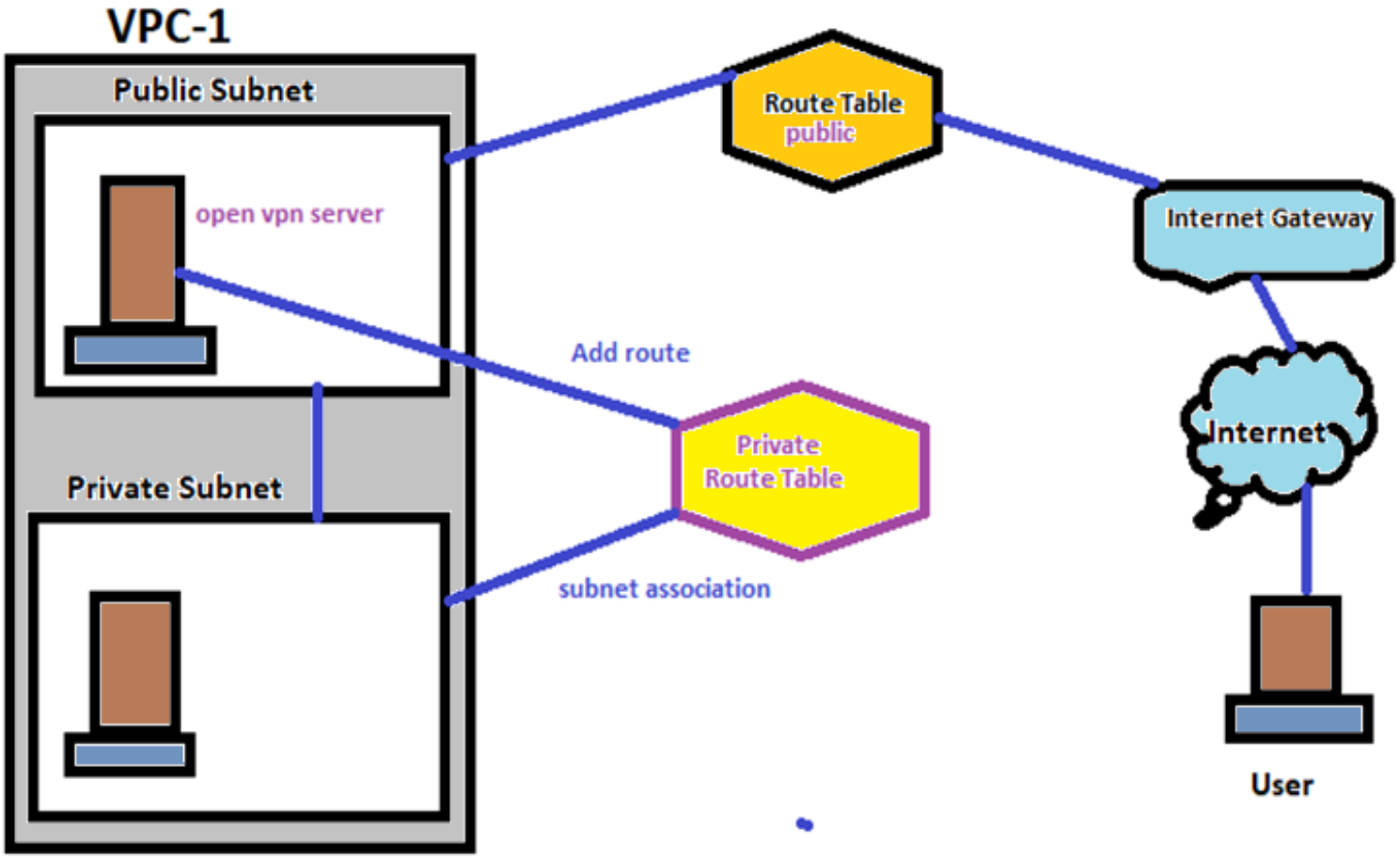
VPN connection between Client to Server Instance



VPN

- ✓ AWS Virtual Private Network solutions establish secure connections between your on-premises networks, remote offices, client devices, and the AWS global network. AWS VPN is comprised of two services: **AWS Site-to-Site VPN** and **AWS Client VPN**. Together, they deliver a highly-available, managed, and elastic cloud VPN solution to protect your network traffic.
- ✓ AWS Site-to-Site VPN creates encrypted tunnels between your network and your Amazon Virtual Private Clouds or AWS Transit Gateways. For managing remote access, AWS Client VPN connects your users to AWS or on-premises resources using a VPN software client.


192.168.1.0/24
192.168.1.1
Internet
206.162.148.9
A
134.28.54.2


VPC-1
Public Subnet
open vpn server
Private Subnet
Route Table
public
Internet Gateway
Add route
Private
Route Table
subnet association
Internet
User

VPN Client Lab Steps --openvpn

1) Create one VPC with Public and Private Subnet.

2) Create openvpn server from aws marketplace-select public subnet-next –SG-keep default ---launch

3) Open this server with user name : openvpnas

4) Follow the instruction ----yes-yes—etc ---passwd openvpn –give new password

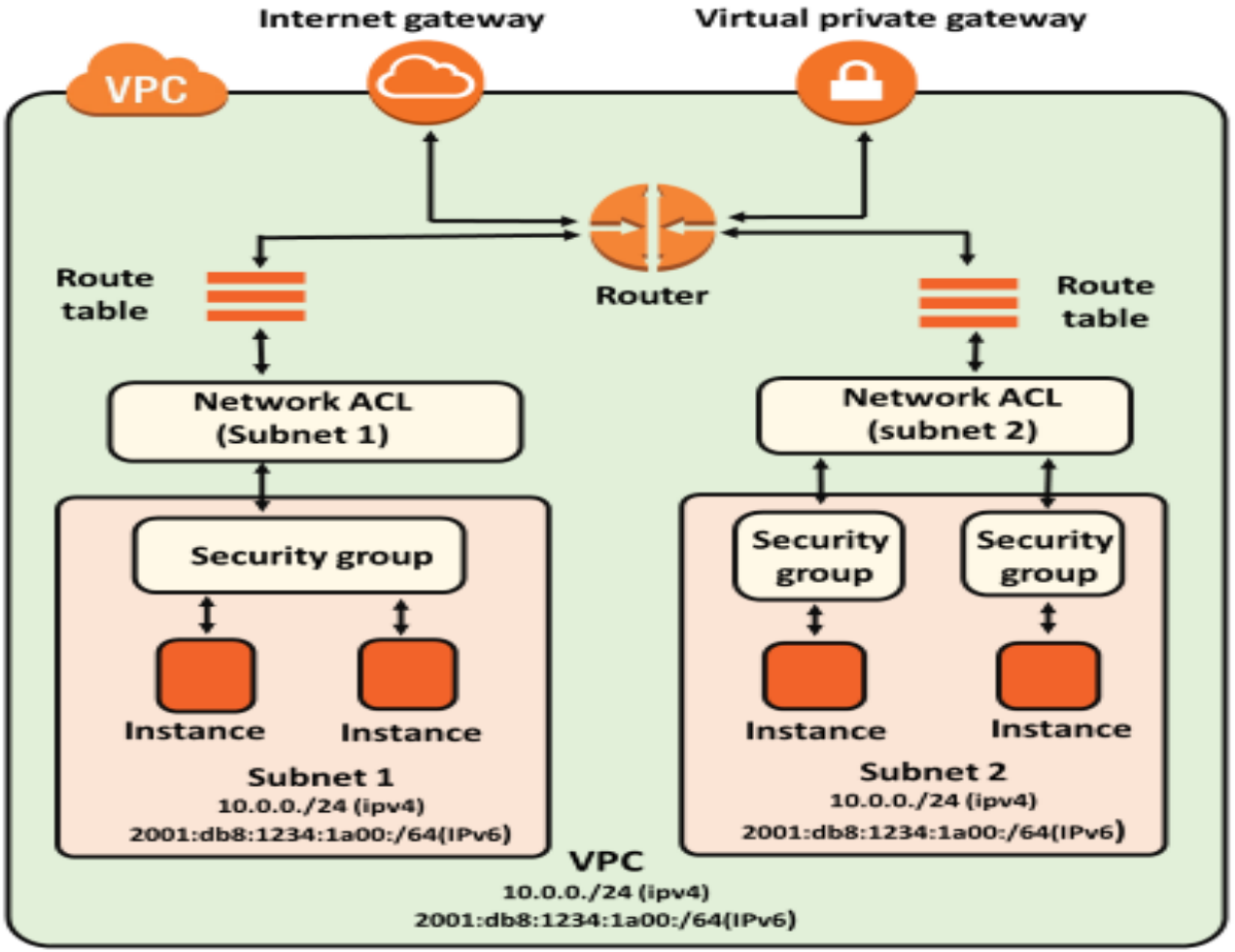
Now copy –admin UI and Client UI and paste in browser

5) Open client UI –download openvpn for windows –install it-and connect to openvpn server –user: openvpn, password –new

6) Now connect to your DB server directly using private IP

Instance Hardening -3 : Security Group and NACL

Firewall concept in VPC -- Security Group – Instance level, NACL – Subnet level security



Instance Hardening -4 : Instance Termination Protection

What are some options that AWS offers that can help me protect my data against accidental termination?

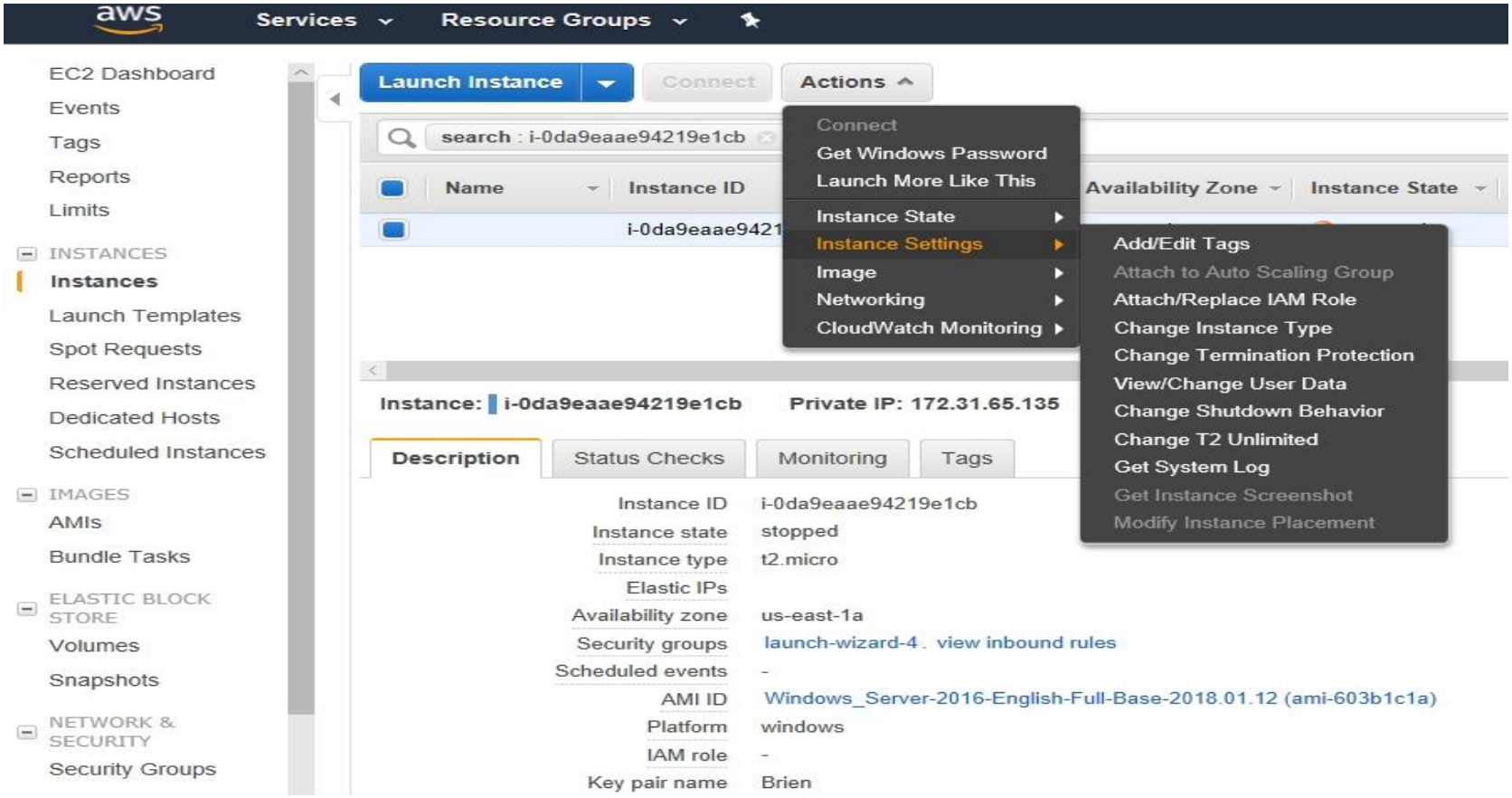
- ✓ **Enable termination protection**. Termination protection prevents an instance from being accidentally terminated by requiring that you disable the protection before terminating the instance. For more information, see Enabling termination protection.

- ✓ **Regularly back up your data.** Back up your instance by creating an Amazon Machine Image (AMI), using Amazon EBS snapshots, or using a third-party backup program. An AMI can capture the data on all attached EBS volumes, and it can be used to launch a new instance.

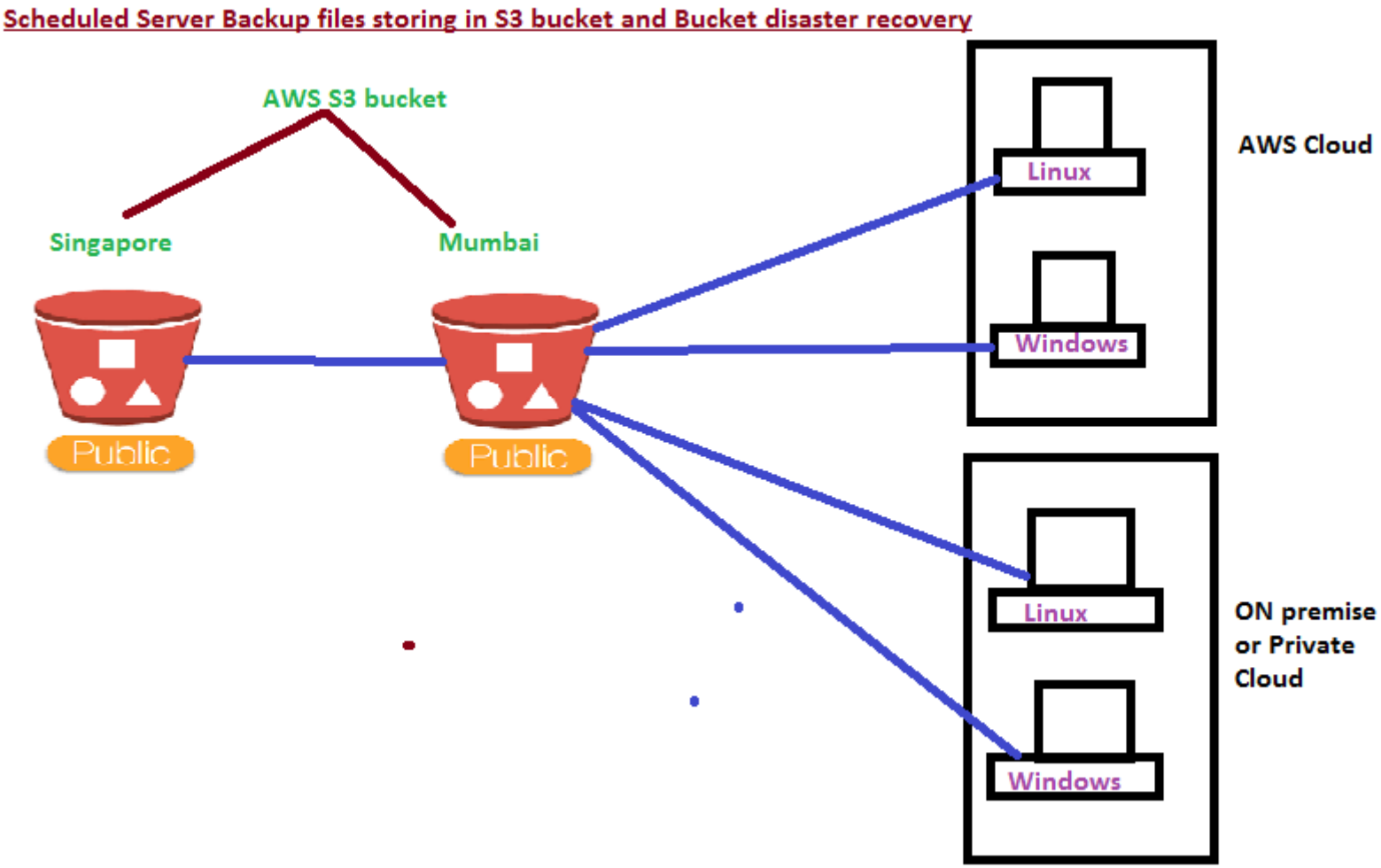
- ✓ **Output data to another AWS service or source**. For example, you might consider using Amazon S3, Amazon RDS, or DynamoDB to store the outputs of workflows you run on your instance.

Enable termination protection.

Instance ------ Action ----- Instance Setting – Termination Protect ----- Enable



Sync Instances with S3 bucket (Instance Data Backup to S3 bucket)



Sync S3 bucket with EC2 instance (Linux) --Steps

1) Open IAM –create an user and assign Amazons3fullaccess policy there.

2) Open this user –Security credential –create access key –download access key.csv file –open it with excel and note down access key and secret access key.

3) Open S3 and create one bucket (indiabucket) –open the bucket—create 3 folder there(lab1, lab2, lab3)

4)Launch Amazon linux2 AMI ---Open it – type command “aws configure” and give the data.

Access key:

Secret access key:

Default region: ap-south-1

Output format: JSON

$ mkdir lab11

$ cd lab11

$ touch file1 file2 file2

$ aws s3 sync /home/ec2-user/lab11 s3://indiabucket/lab1

Now check the data in S3 bucket

Upload some files in S3 bucket—Come to EC2 instance

$ aws s3 sync s3://indiabucket/lab1 /home/ec2-user/lab11

$ ls

Sync S3 bucket with EC2 instance (Windows) --Steps

1) Open IAM –create an user and assign Amazons3fullaccess policy there.

2) Open this user –Security credential –create access key –download access key.csv file –open it with excel and note down access key and secret access key.

3) Open S3 and create one bucket (indiabucket) –open the bucket—create 3 folder there(lab1, lab2, lab3)

4)Launch Windows instance ---Open it – Download and install “Amazon CLI “

Open cmd– type command “aws configure” and give the data.

Access key:

Secret access key:

Default region: ap-south-1

Output format: JSON

C:\ mkdir lab22

C:\ cd lab22

C:\ touch file1 file2 file2

C:\ aws s3 sync c:\lab22 s3://indiabucket/lab2

Now check the data in S3 bucket

Upload some files in S3 bucket—Come to EC2 instance

C:\ aws s3 sync s3://indiabucket/lab2 c:\lab22

C:\ ls

How to Schedule the File sync (Auto Backup)(Windows)

Windows

1) Create one batch file and write the command there

aws s3 sync c:\awsdata2\ s3://awsbatch100/windows1

Save file--- file name: s3sync.bat, save as type: all files

2) Open task Scheduler – create basic task—name:test1 –next—select daily-next-set time –next—start a program– browse and select created batch file –ok—finish.

3) Now create some files in c:\awsdata2\ folder and wait for that scheduled time –then check the output in s3 bucket.

How to Schedule the File sync (Auto Backup)(Linux)

Linux

1) Create one shell script file and write the command there

$ pwd

/home/ec2-user

$nano test1.sh

aws s3 sync /home/ec2-user/linux11 s3://awsbatch100/linux1

Save and exit

2) Chmod u+x test1.sh

3) crontab –e

- * * * * sh /home/ec2-user/test1.sh

3) Now create some files in /home/ec2-user/linux11 folder and wait for that scheduled time –then check the output in s3 bucket.

Note: the given time is to run the script in every one minue