Installation and Use of Virtual Machines

Virtual machines (VMs) are pivotal in modern computing, enabling the creation of isolated, customizable environments on a single physical host. By emulating hardware, VMs allow multiple operating systems to run concurrently, making them ideal for testing, development, and cloud computing applications. This assignment explores the practical implementation of VMs using VMware Workstation, a leading hypervisor, to install and configure CentOS, Windows Vista, and OpenStack in All-in-One mode. Each VM is tailored to demonstrate specific functionalities, such as network configuration, internet access, and cloud service deployment. Additionally, the assignment extends to leveraging Amazon Web Services (AWS) for public cloud server setup, network and security configuration, and storage management. Through these tasks, we demonstrate the versatility of VMs in local and cloud environments, ensuring robust network connectivity and resource management. The following sections detail the step-by-step processes, configurations, testing results, and a comprehensive report summarizing the outcomes, challenges, and solutions encountered during the setup.

Install VMware Workstation:

VMware Workstation is a desktop hypervisor that allows you to create and manage multiple virtual machines on a single physical computer. It supports various guest operating systems and provides robust networking capabilities for testing cloud environments.

Download VMware Workstation:

- Download the latest version of VMware Workstation Pro (e.g., version 17.x as of 2025). You may need a license or a trial version for educational purposes.
- Ensure your system meets the requirements: 64-bit Windows/Linux, 4 GB RAM (16 GB recommended), 2 GHz multi-core processor, and ~2 GB free disk space for the application.

vmware
India
Communities | Store | Login
VMware Cloud
Products
Solutions
Support
Professional Services
Downloads
Partners
Company
Products > Workstation Pro > Try VMware Workstation Pro
VMWARE WORKSTATION PRO™ 15.5
Workstation 15.5 Pro for Windows
Download Now >
Workstation 15.5 Pro for Linux
Download Now >

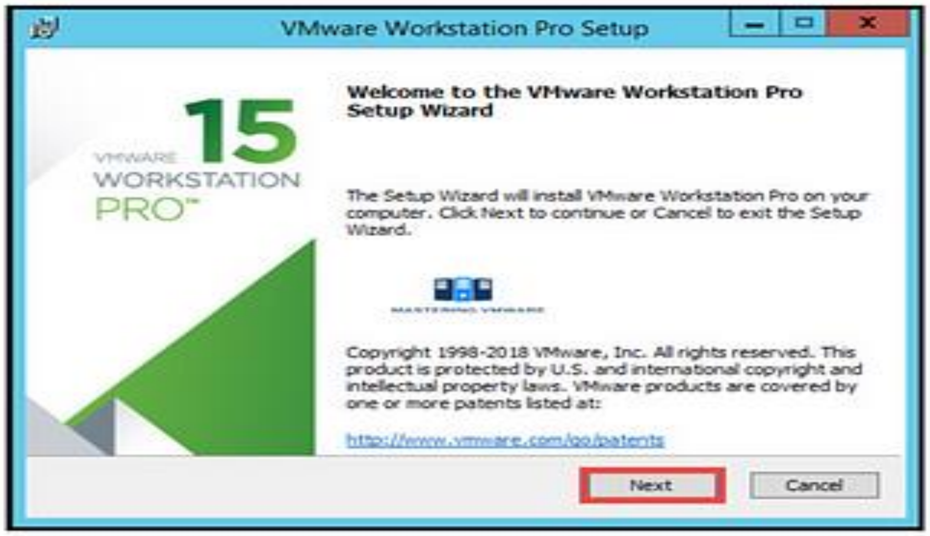
VMware Workstation Pro is the industry standard for running multiple operating systems on a single Linux or Windows PC.

Workstation 15.5 Pro improves on the leading desktop with new user interface controls, Jumbo Frame support, support for the latest Windows and Linux operating systems, and more...

Install VMware Workstation:

- Run the installer (e.g., VMware-workstation-full-17.x.x.exe).
- Follow the installation wizard:
  - Accept the license agreement.
  - Choose the installation directory (default is fine).
  - Enable or disable product updates and VMware Customer Experience Improvement Program as per your preference.
  - Install VMware Workstation Server for shared VMs (optional for this assignment).

- Complete the installation and enter your license key if prompted. If using a trial, skip this step.
- Restart your computer if required.

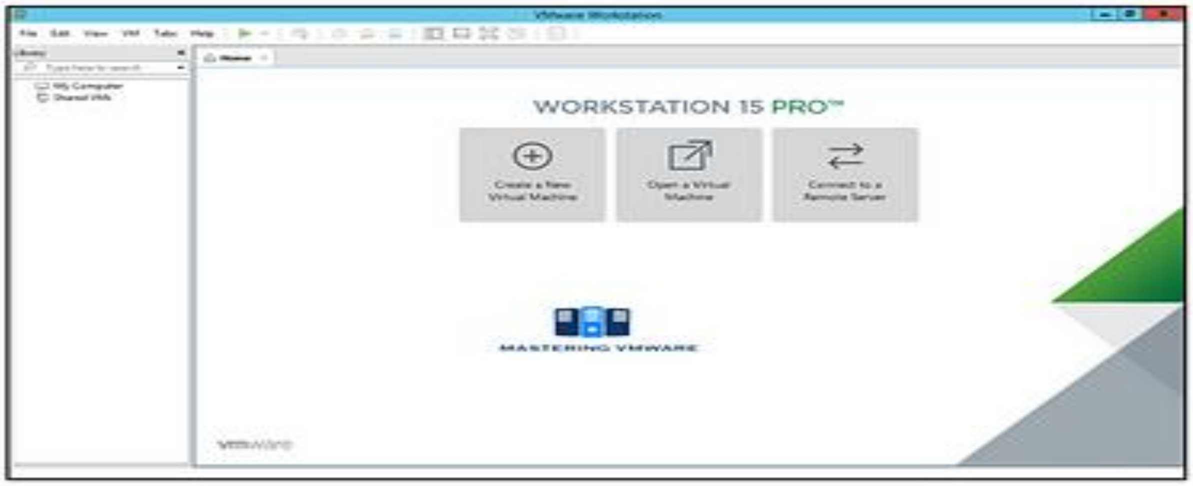

VMware Workstation Pro Setup

Welcome to the VMware Workstation Pro Setup Wizard

The Setup Wizard will install VMware Workstation Pro on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Copyright 1998-2018 VMware, Inc. All rights reserved. This product is protected by U.S. and international copyright and intellectual property laws. VMware products are covered by one or more patents listed at:

http://www.vmware.com/go/patents



Verify Installation:

- Launch VMware Workstation from the Start menu.
- Ensure the interface opens, showing options to create a new virtual machine.
  Artifact: VMware Workstation Installation Steps

  - 64-bit Windows 10/11 or Linux host.
  - Minimum 4 GB RAM (16 GB recommended).
  - 2 GHz multi-core processor.
  - ~2 GB free disk space.

Installation Steps

1. Run the installer and follow the wizard:
   - Accept the license agreement.
   - Choose the installation directory.
   - Configure optional settings (updates, VMware Workstation Server).

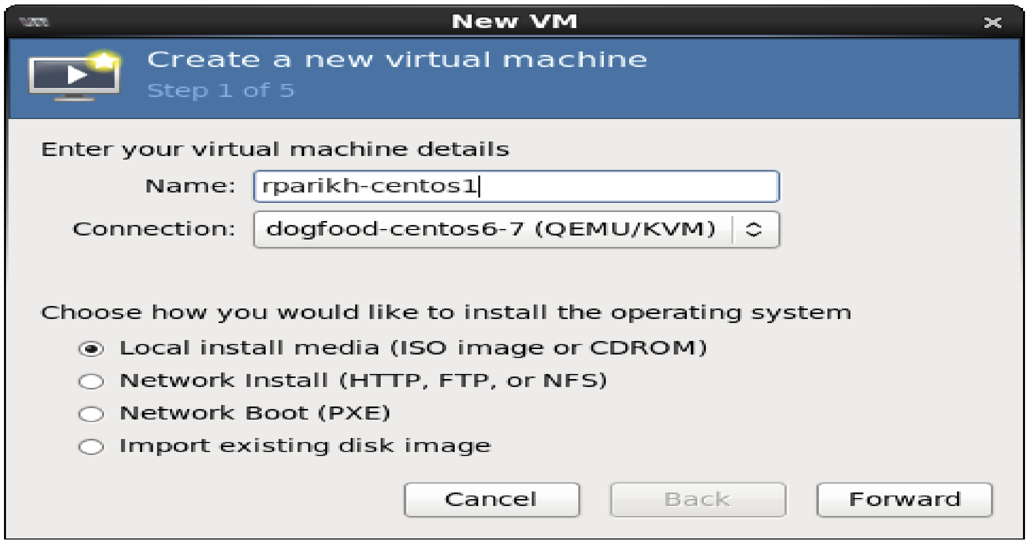
2. Enter a license key or use trial mode.
3. Restart the system if prompted.
4. Launch VMware Workstation to verify installation.

Step 2: Create a Virtual Machine and Install CentOS.

CentOS is a stable, open-source Linux distribution commonly used in cloud computing. Create a VM in VMware Workstation and install CentOS 7 (or the latest available version, e.g., CentOS Stream 9, as CentOS 7 reached EOL in 2024).

```
cd /var/lib/libvirt/images
wget http://mirror.hmc.edu/centos/6.6/isos/x86_64/CentOS-6.6-x86_64-minimal.iso
```

- Download the CentOS Stream 9 ISO
- Save the ISO to a local directory (e.g., C:\ISOs).

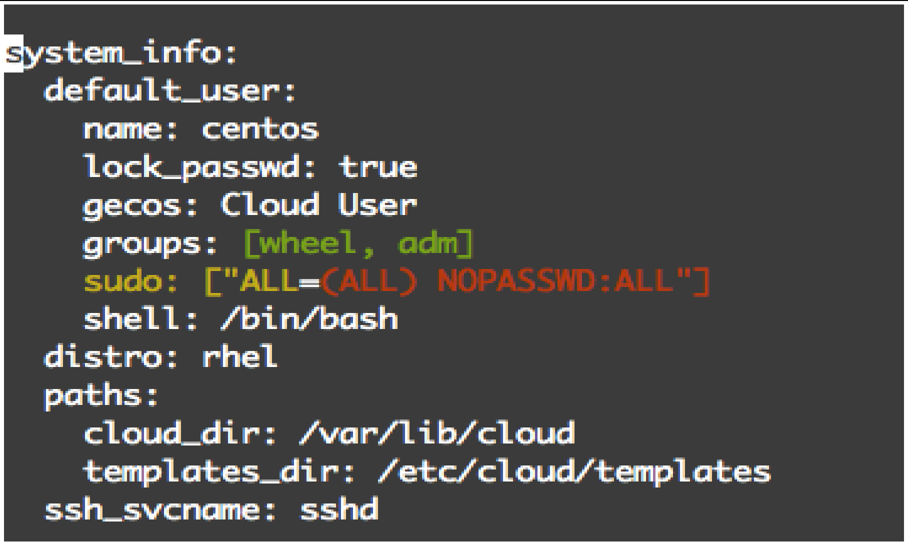


Enter your virtual machine details
Name: rparikh-centos1
Connection: dogfood-centos6-7 (QEMU/KVM)

Choose how you would like to install the operating system
- [✓] Local install media (ISO image or CDROM)
- Network Install (HTTP, FTP, or NFS)
- Network Boot (PXE)
- Import existing disk image

Create CentOS VM:

- Open VMware Workstation.
- Click **File > New Virtual Machine**.
- Select **Typical (recommended)** and click **Next**.
- Choose **Installer disc image file (iso)** and browse to the CentOS ISO.
- Select **Linux** as the guest OS and **CentOS 64-bit** (or CentOS Stream 9 64-bit) as the version.
- Name the VM (e.g., CentOS-VM) and set the location (e.g., C:\Virtual Machines\CentOS-VM).
- Allocate disk space (e.g., 20 GB, store as a single file) and click **Next**.
- Customize hardware:
  - Memory: 4 GB (minimum 2 GB).
  - Processors: 2 CPUs, 1 core each.
  - Network Adapter: Set to **NAT** for internet access.
- Click **Finish**.



```
system_info:
  default_user:
    name: centos
    lock_passwd: true
    gecos: Cloud User
    groups: [wheel, adm]
    sudo: ["ALL=(ALL) NOPASSWD:ALL"]
    shell: /bin/bash
  distro: rhel
  paths:
    cloud_dir: /var/lib/cloud
    templates_dir: /etc/cloud/templates
  ssh_svcname: sshd
```

Install CentOS:

- Power on the VM by clicking **Power on this virtual machine**.
- The CentOS installer boots. Select **Install CentOS** and press Enter.
- **Configure:**
  - Language: English (or your preference).
  - Network & Hostname: Enable the network interface (e.g., ens33) and set to **On** for DHCP.
  - Installation Destination: Select the 20 GB disk and use automatic partitioning.
  - Root Password: Set a strong password (e.g., Password123!).
  - User Creation: Create a user (e.g., centosuser, password: Password123!).
- Click **Begin Installation** and wait for completion.
- Reboot the VM when prompted.

Network Configuration:

- Log in as root or the created user.
- Verify network settings:

  nmcli device status

  Ensure ens33 (or similar) is connected.
- Edit the network configuration file if needed:

  sudo vi /etc/sysconfig/network-scripts/ifcfg-ens33

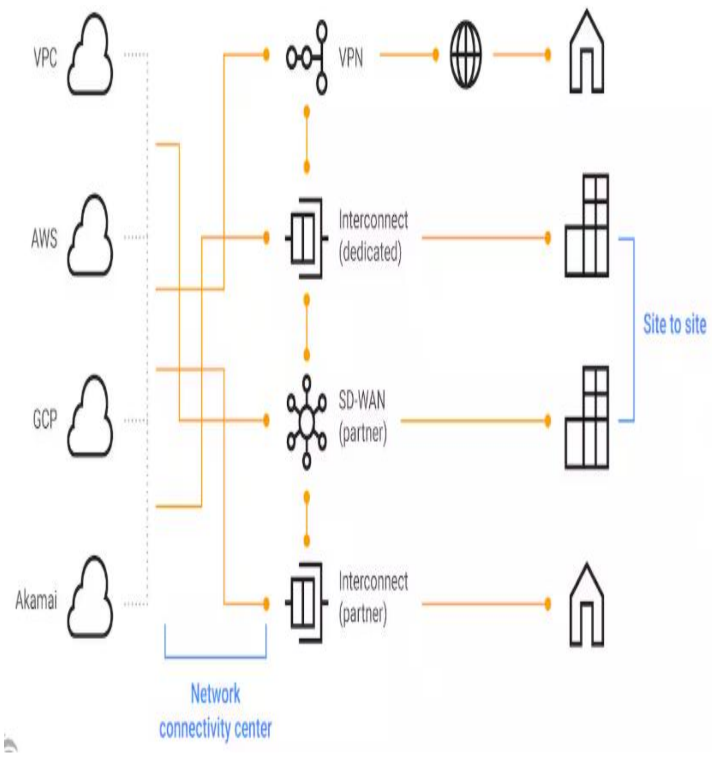
  BOOTPROTO=dhcp

  ONBOOT=yes
- Restart the network service:

  sudo systemctl restart network
- Check IP address:

  ip addr show ens33

  the IP (e.g., 192.168.1.1).



Test Internet Access:

- Open a terminal and test connectivity:

  ```
  ping -c 4 google.com
  ```

  Expected output: Successful ping responses (e.g., 64 bytes from...).
- Install a browser (e.g., Firefox) if GUI is enabled:

  ```
  sudo yum install firefox
  ```

  Launch Firefox and visit a website (e.g., https://www.google.com).

Ping Test from Host:

- On the host (Windows), open Command Prompt.
- Ping the VM's IP:

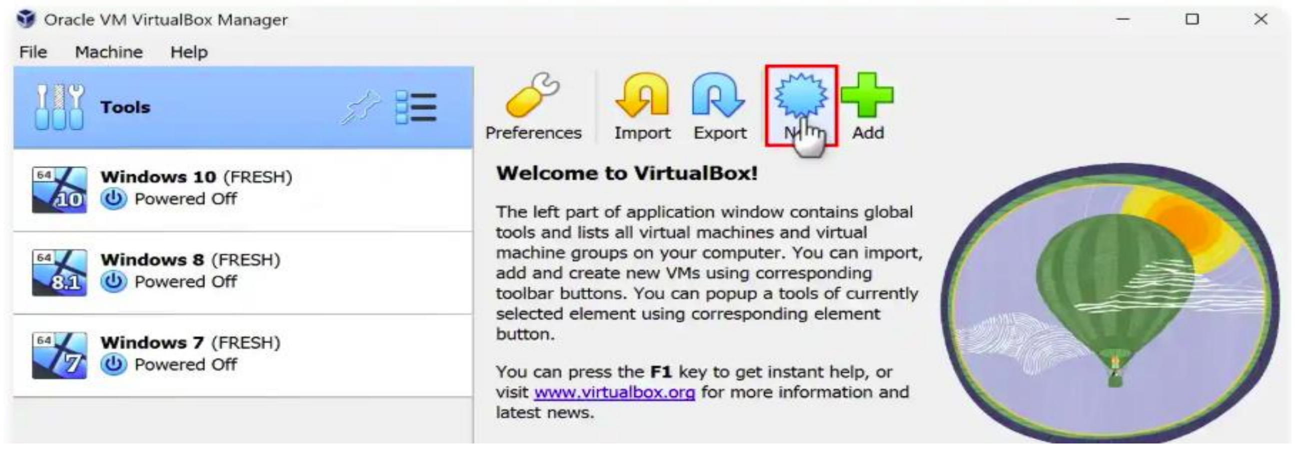
  cmd

  ```
  ping 192.168.1.1
  ```

  Artifact: CentOS VM Installation and Configuration

Step 3: Create a Virtual Machine and Install Windows Vista

Windows Vista, though outdated, may be specified for legacy testing. We'll create a VM and install it, ensuring network connectivity.



Welcome to VirtualBox!

The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

Obtain Windows Vista ISO:

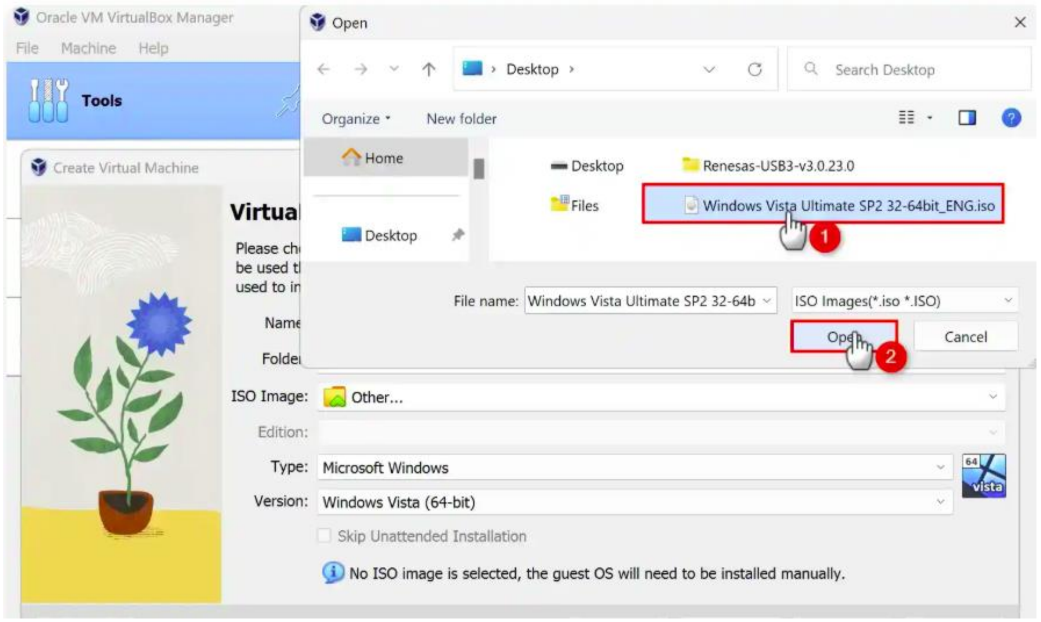
- Source a legitimate Windows Vista ISO (e.g., from an educational license or archive). Ensure you have a valid product key.
- Save the ISO to C:\ISOs.

Create Windows Vista VM:

- In VMware Workstation, click **File > New Virtual Machine**.
- Select **Typical** and choose the Windows Vista ISO.
- Set guest OS to **Microsoft Windows > Windows Vista (64-bit or 32-bit)**.
- Name the VM (e.g., Vista-VM) and set location (e.g., C:\Virtual Machines\Vista-VM).
- Allocate 20 GB disk space.
- Customize hardware:
  - Memory: 2 GB (minimum 1 GB).
  - Processors: 1 CPU, 1 core.
  - Network Adapter: **NAT**.
- Click **Finish**.

Install Windows Vista:

- Power on the VM.
- Follow the installer:
  - Select language and edition (e.g., Home Premium).
  - Enter the product key.
  - Choose the 20 GB disk for installation.
  - Complete the setup, setting a username and password.
- Install VMware Tools after booting:
  - In VMware, click **VM > Install VMware Tools**.
  - Mount the virtual CD, run the installer, and follow prompts.



File name: Windows Vista Ultimate SP2 32-64b

ISO Images(*.iso *.ISO)

ISO Image: Other...

Edition:

Type: Microsoft Windows

Version: Windows Vista (64-bit)

No ISO image is selected, the guest OS will need to be installed manually.

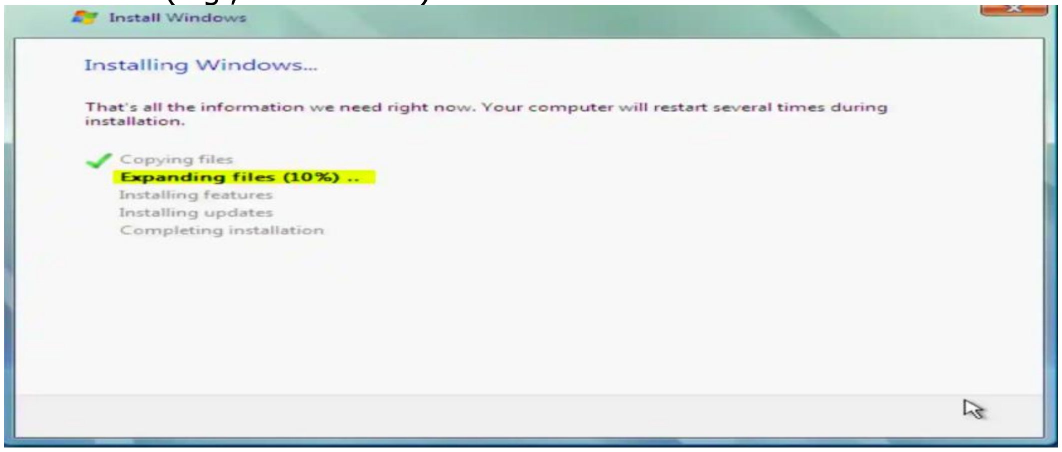
Network Configuration:

- After booting, check network status in **Control Panel > Network and Sharing Center**.
- Ensure the network adapter is set to obtain an IP automatically (DHCP).
- Verify IP address:

  cmd

  ipconfig

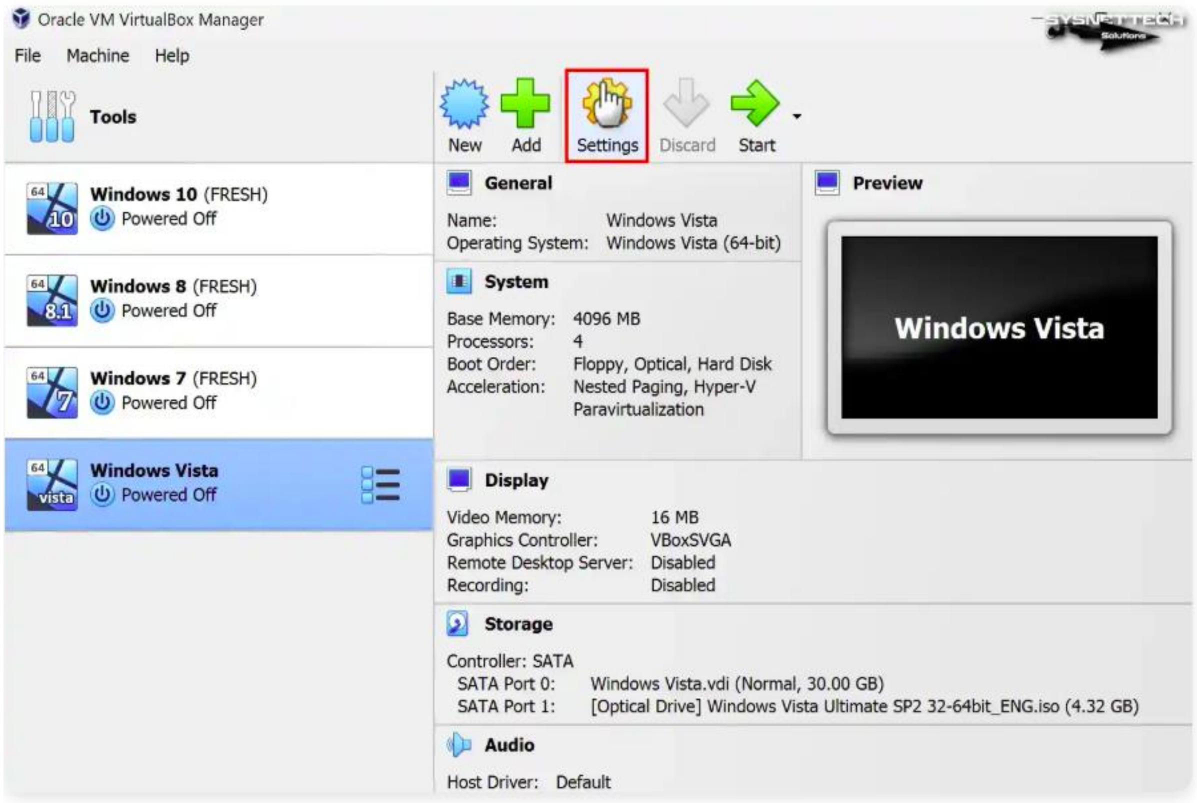
  the IPv4 address (e.g., 192.168.1.1).



Installing Windows...

That's all the information we need right now. Your computer will restart several times during installation.

✓ Copying files
Expanding files (10%) ..
Installing features
Installing updates
Completing installation



Oracle VM VirtualBox Manager
File Machine Help
Tools
New Add Settings Discard Start
Windows 10 (FRESH) Powered Off
Windows 8 (FRESH) Powered Off
Windows 7 (FRESH) Powered Off
Windows Vista Powered Off
Preview
Windows Vista

General

Name: Windows Vista
Operating System: Windows Vista (64-bit)

System

Base Memory: 4096 MB
Processors: 4
Boot Order: Floppy, Optical, Hard Disk
Acceleration: Nested Paging, Hyper-V Paravirtualization

Display

Video Memory: 16 MB
Graphics Controller: VBoxSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage

Controller: SATA
SATA Port 0: Windows Vista.vdi (Normal, 30.00 GB)
SATA Port 1: [Optical Drive] Windows Vista Ultimate SP2 32-64bit_ENG.iso (4.32 GB)

Audio

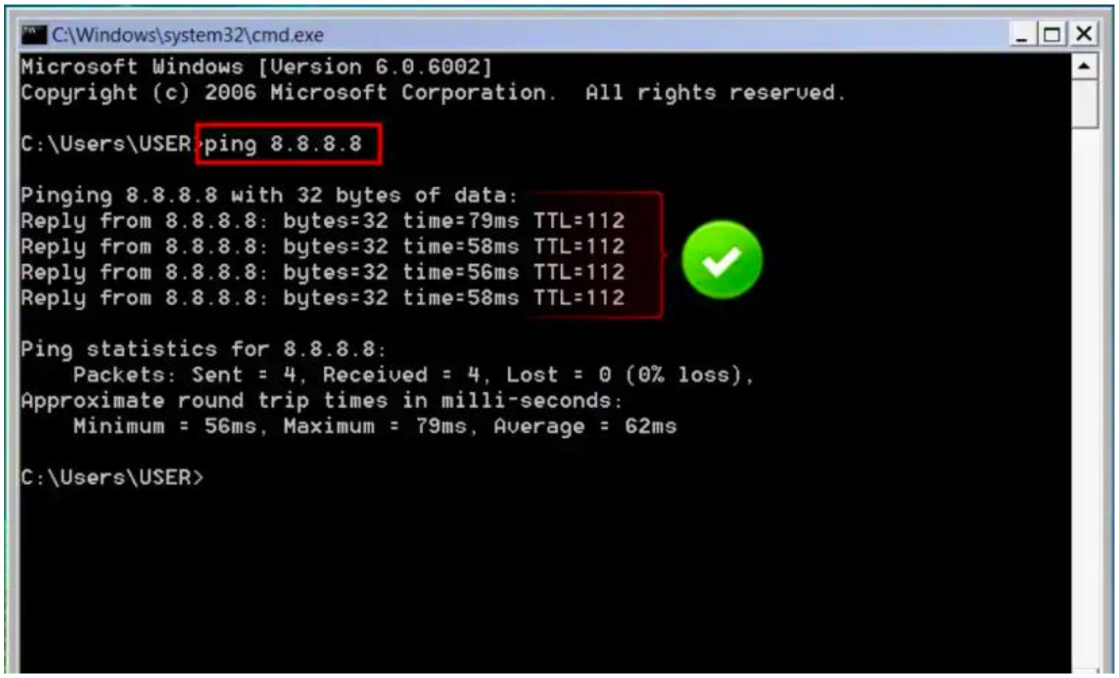
Host Driver: Default

Test Internet Access:

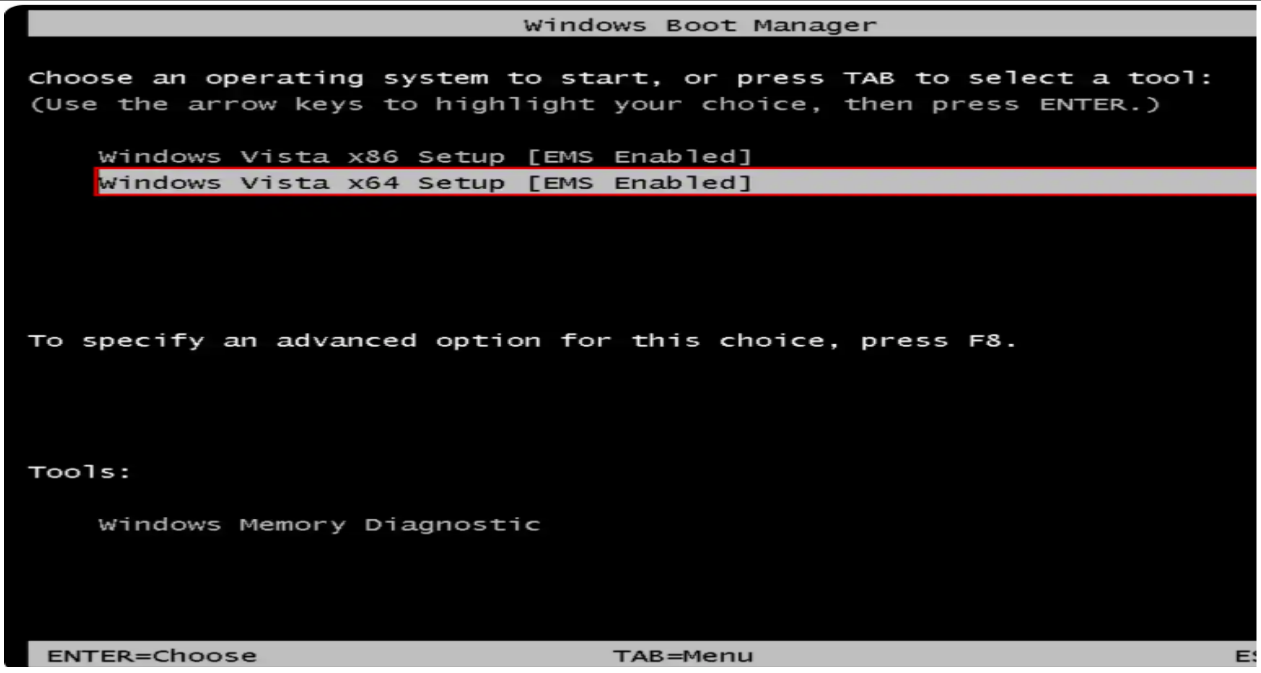
- Test connectivity:

  cmd

  ping google.com
  Expected result: Successful replies.



```
Microsoft Windows [Version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\USER>ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=79ms TTL=112
Reply from 8.8.8.8: bytes=32 time=58ms TTL=112
Reply from 8.8.8.8: bytes=32 time=56ms TTL=112
Reply from 8.8.8.8: bytes=32 time=58ms TTL=112

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 56ms, Maximum = 79ms, Average = 62ms

C:\Users\USER>
```



```
Windows Boot Manager

Choose an operating system to start, or press TAB to select a tool:
(Use the arrow keys to highlight your choice, then press ENTER.)

    Windows Vista x86 Setup [EMS Enabled]
    Windows Vista x64 Setup [EMS Enabled]



To specify an advanced option for this choice, press F8.



Tools:

    Windows Memory Diagnostic



 ENTER=Choose                        TAB=Menu                        ES
```

Ping Test from Host:

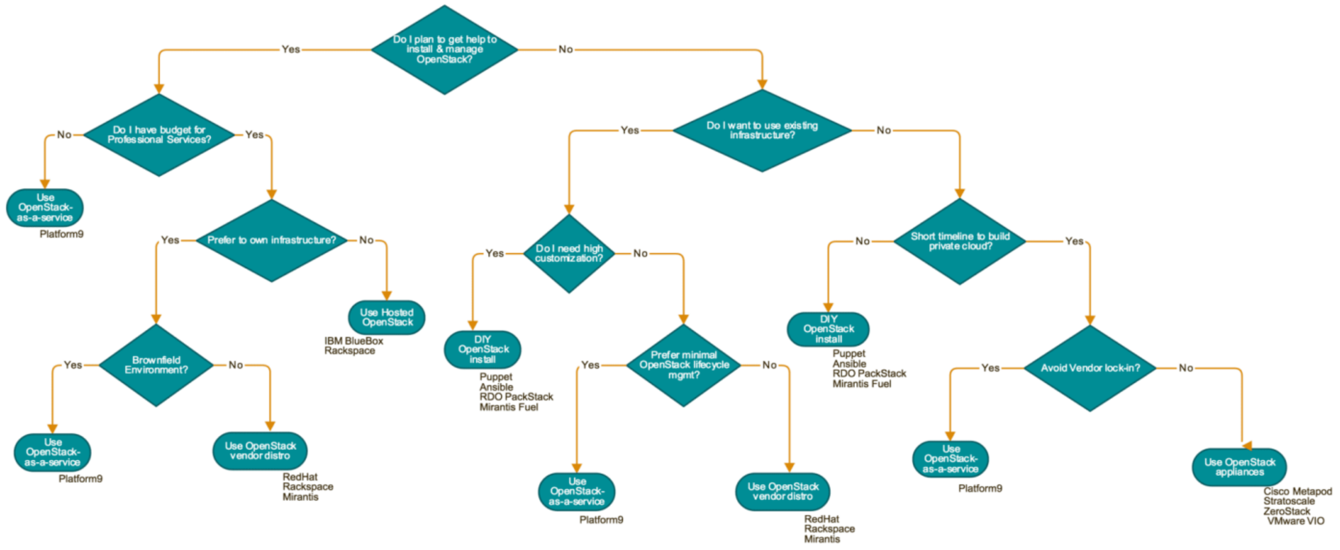
- From the host Command Prompt:
  cmd

  ping <Vista_VM_IP>
  <u>Expected result</u>: Replies from the VM. If the firewall blocks ICMP, disable it:
  - **Control Panel > Windows Firewall > Turn Windows Firewall on or off > Off**.

    Artifact: Windows Vista VM Installation and Configuration

<u>Create a Virtual Machine and Install OpenStack All-in-One Mode</u>

OpenStack is a cloud computing platform for managing virtual machines. The All-in-One mode, typically deployed via DevStack, installs all OpenStack components on a single VM for development and testing. We'll use Ubuntu 24.04 LTS as the base OS due to its compatibility with DevStack.

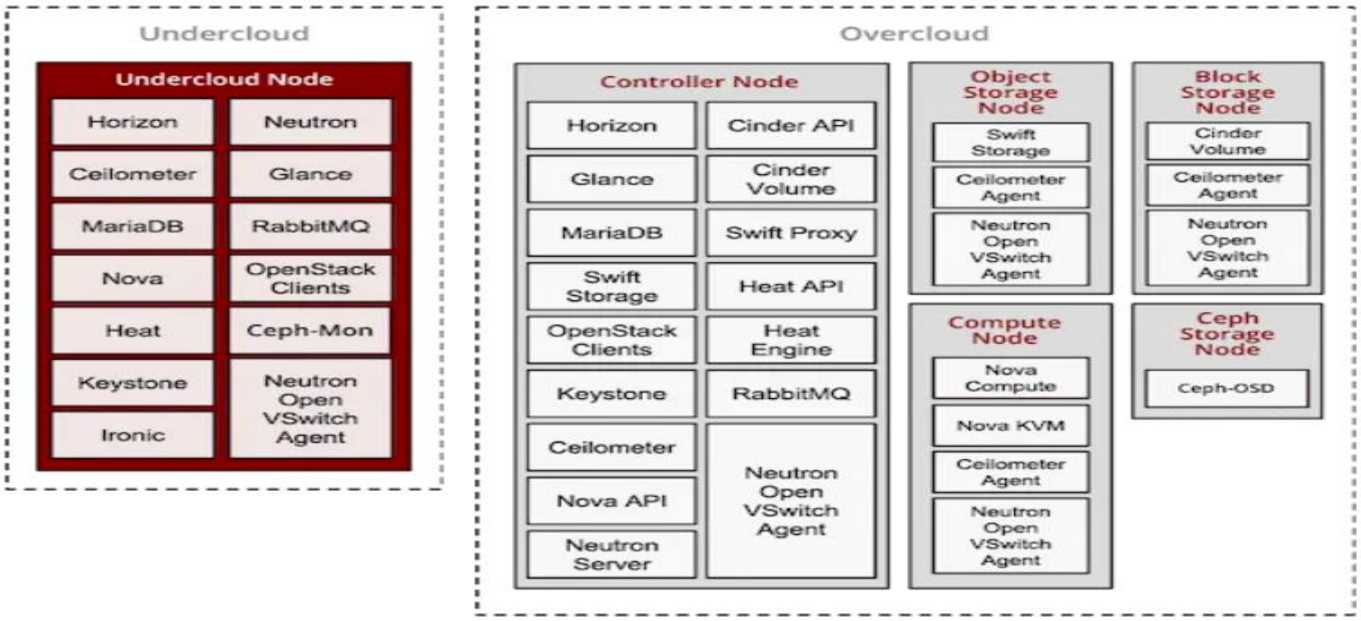

Do I plan to get help to install & manage OpenStack?
Yes
No
Do I have budget for Professional Services?
No
Yes
Use OpenStack-as-a-service
Platform9
Prefer to own infrastructure?
Yes
No
Use Hosted OpenStack
IBM BlueBox
Rackspace
Brownfield Environment?
Yes
No
Use OpenStack-as-a-service
Platform9
Use OpenStack vendor distro
RedHat
Rackspace
Mirantis
Do I want to use existing infrastructure?
Yes
No
Do I need high customization?
Yes
No
DIY OpenStack install
Puppet
Ansible
RDO PackStack
Mirantis Fuel
Prefer minimal OpenStack lifecycle mgmt?
Yes
No
Use OpenStack-as-a-service
Platform9
Use OpenStack vendor distro
RedHat
Rackspace
Mirantis
Short timeline to build private cloud?
No
Yes
DIY OpenStack install
Puppet
Ansible
RDO PackStack
Mirantis Fuel
Avoid Vendor lock-in?
Yes
No
Use OpenStack-as-a-service
Platform9
Use OpenStack appliances
Cisco Metapod
Stratoscale
ZeroStack
VMware VIO

Download Ubuntu ISO:

- Download the Ubuntu 24.04 LTS ISO (e.g., ubuntu-24.04-live-server-amd64.iso).

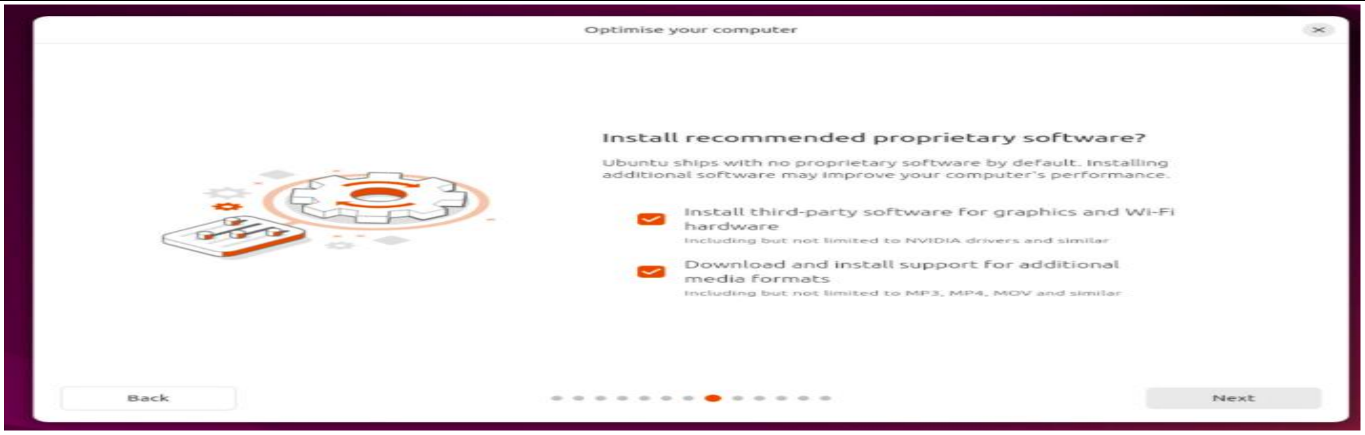
Create OpenStack VM:

- In VMware Workstation, click **File > New Virtual Machine**.
- Select **Typical**, choose the Ubuntu ISO, and set guest OS to **Ubuntu 64-bit**.
- Name the VM (e.g., OpenStack-VM) and set location (e.g., C:\Virtual Machines\OpenStack-VM).
- Allocate 100 GB disk space (OpenStack requires significant storage).
- Customize hardware:
  - Memory: 16 GB (minimum 8 GB).
  - Processors: 4 CPUs, 2 cores each.
  - Network Adapter: **NAT**.
  - Enable **Expose hardware assisted virtualization to the guest OS** (in CPU settings) for nested virtualization, as OpenStack uses KVM/QEMU.



Install Ubuntu:

- Power on the VM and follow the installer:
  - Choose language and keyboard.
  - Enable SSH server during installation.
  - Set hostname (e.g., openstack) and create a user (e.g., openstackuser, password: Password123!).
  - Use automatic partitioning for the 100 GB disk.
- Reboot after installation.


Optimise your computer

Install recommended proprietary software?

Ubuntu ships with no proprietary software by default. Installing additional software may improve your computer's performance.

[✓] Install third-party software for graphics and Wi-Fi hardware
Including but not limited to NVIDIA drivers and similar

[✓] Download and install support for additional media formats
Including but not limited to MP3, MP4, MOV and similar

Install DevStack for OpenStack:

- Log in as openstackuser.
- Update the system:

  sudo apt update && sudo apt upgrade -y

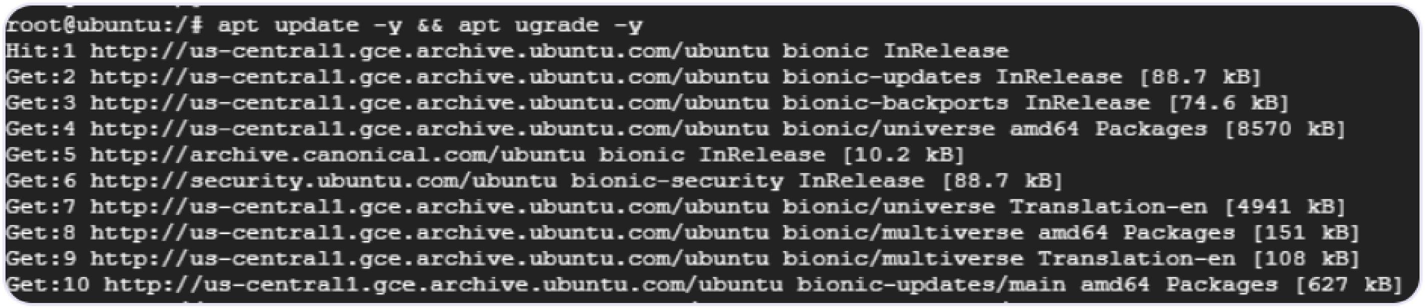
- Create a stack user: *sudo useradd -s /bin/bash -d /opt/stack -m stack*

  echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack

- Switch to the stack user: *sudo -u stack -i*
- Clone DevStack:
  CollapseWrapRun

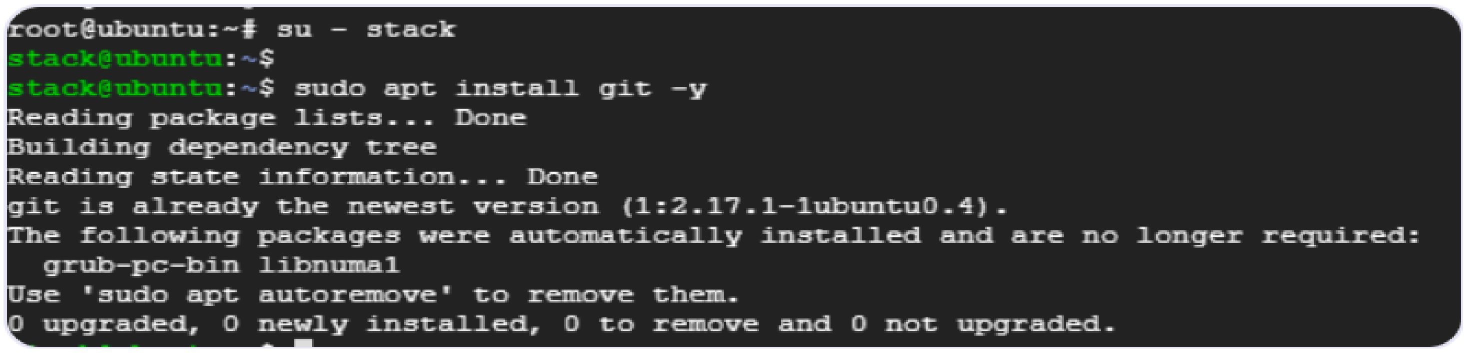
  git clone https://opendev.org/openstack/devstack

  cd devstack



```
root@ubuntu:/# apt update -y && apt ugrade -y
Hit:1 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [8570 kB]
Get:5 http://archive.canonical.com/ubuntu bionic InRelease [10.2 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:7 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/universe Translation-en [4941 kB]
Get:8 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [151 kB]
Get:9 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/multiverse Translation-en [108 kB]
Get:10 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [627 kB]
```

- **Create a minimal local.conf:**
  - *cat > local.conf << EOL   [[local|localrc]]*
  - *ADMIN_PASSWORD=secret*
  - *DATABASE_PASSWORD=secret*
  - *RABBIT_PASSWORD=secret*
  - *SERVICE_PASSWORD=secret*
  - *HOST_IP=<VM_IP>*
  - *FLOATING_RANGE=192.168.1.224/27*
  - *Q_FLOATING_ALLOCATION_POOL=start=192.168.1.225,end=192.168.1.254*
  - *PUBLIC_NETWORK_GATEWAY=192.168.1.1*
  - *PUBLIC_INTERFACE=eth0*
  - *ENABLED_SERVICES=nova,neutron,glance,keystone,horizon*
  - *Replace <VM_IP> with the VM's IP (check with ip addr show eth0)*



```
root@ubuntu:~# su - stack
stack@ubuntu:~$
stack@ubuntu:~$ sudo apt install git -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.17.1-1ubuntu0.4).
The following packages were automatically installed and are no longer required:
  grub-pc-bin libnuma1
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

- **Run DevStack:**

./stack.sh
This may take 10–30 minutes. Monitor for errors.

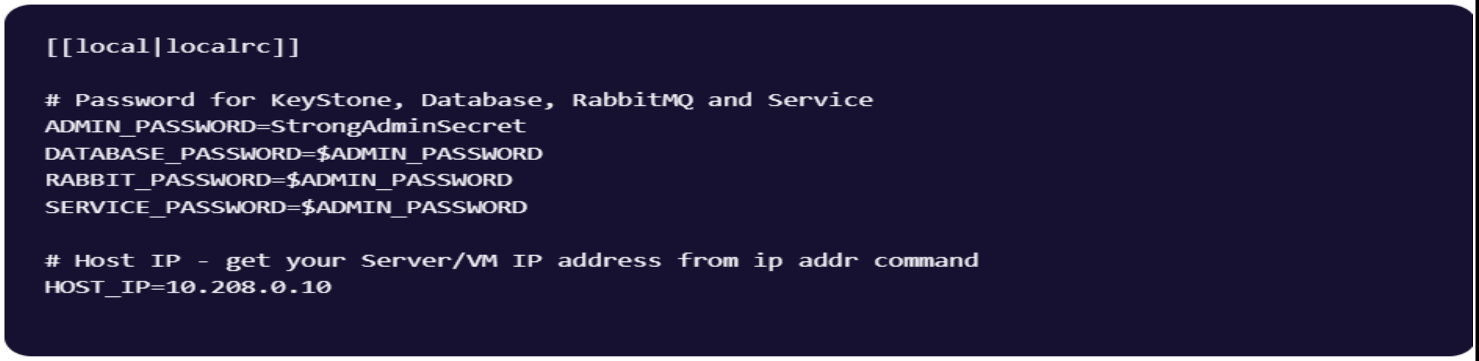
Network Configuration:

- DevStack configures an external network (public) and a private network with DHCP.

  Verify network settings: *openstack network list*

Log in to the Horizon dashboard (http://<VM_IP>/dashboard, user: admin, password: secret) to confirm networks.

Ensure the VM has internet access: ***ping -c 4 google.com***



```
[[local|localrc]]

# Password for KeyStone, Database, RabbitMQ and Service
ADMIN_PASSWORD=StrongAdminSecret
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD

# Host IP - get your Server/VM IP address from ip addr command
HOST_IP=10.208.0.10
```

Test Internet Access:

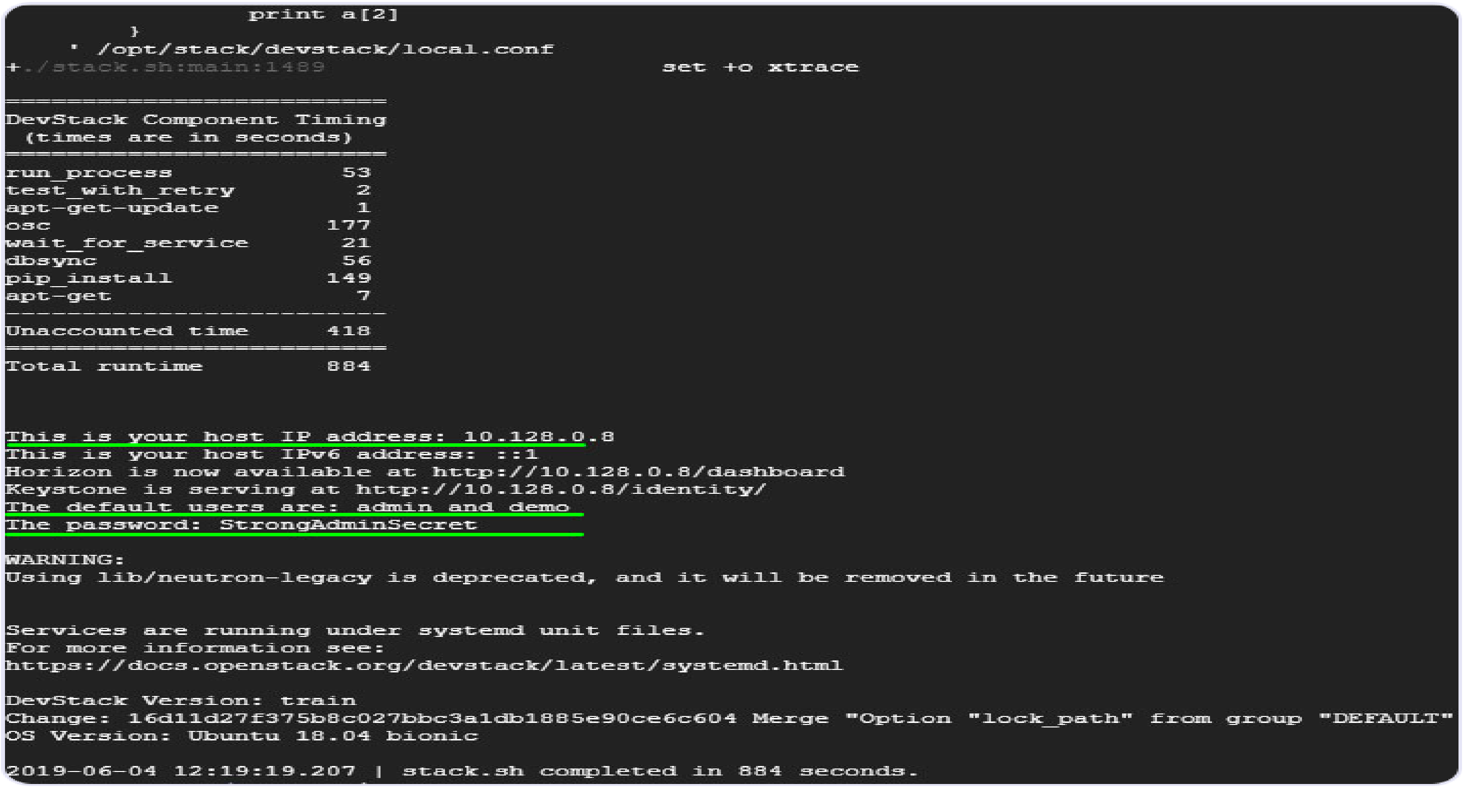
- Open a browser on the host and access the Horizon dashboard at http://<VM_IP>/dashboard.
- Log in and verify the dashboard loads.
  - From the VM, test: *ping -c 4 google.com*

Ping Test from Host:

- From the host Command Prompt:
  cmd

  ping <OpenStack_VM_IP>

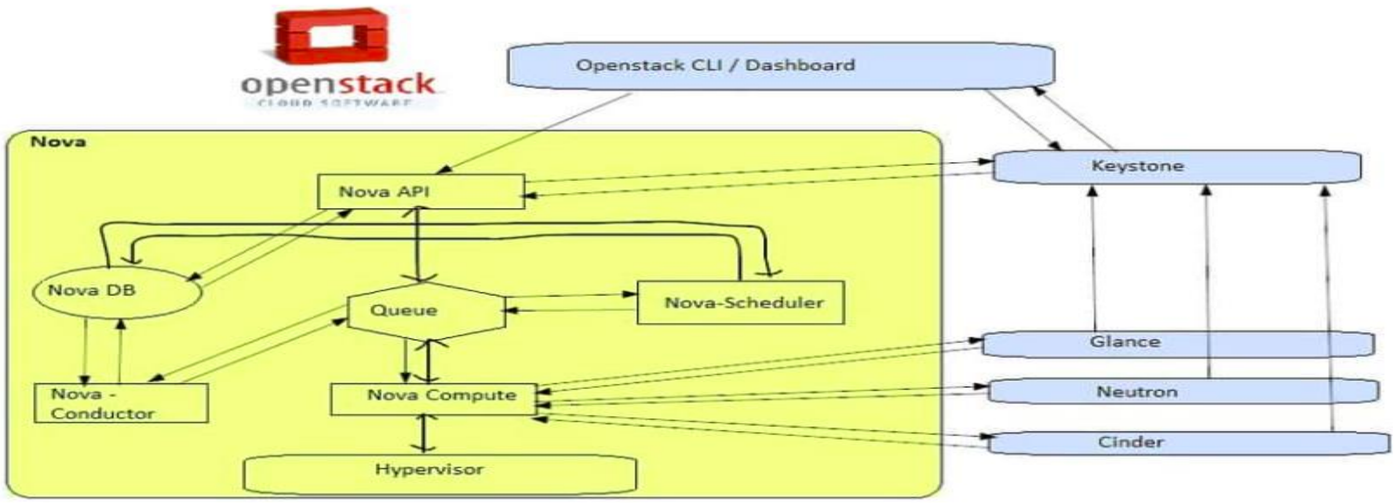
  Expected result: Successful replies



```
                    print a[2]
            }
        ' /opt/stack/devstack/local.conf
+./stack.sh:main:1489                           set +o xtrace

=========================
DevStack Component Timing
 (times are in seconds)
=========================
run_process            53
test_with_retry         2
apt-get-update          1
osc                   177
wait_for_service       21
dbsync                 56
pip_install           149
apt-get                 7
-------------------------
Unaccounted time      418
=========================
Total runtime         884



This is your host IP address: 10.128.0.8
This is your host IPv6 address: ::1
Horizon is now available at http://10.128.0.8/dashboard
Keystone is serving at http://10.128.0.8/identity/
The default users are: admin and demo
The password: StrongAdminSecret

WARNING:
Using lib/neutron-legacy is deprecated, and it will be removed in the future


Services are running under systemd unit files.
For more information see:
https://docs.openstack.org/devstack/latest/systemd.html

DevStack Version: train
Change: 16d11d27f375b8c027bbc3a1db1885e90ce6c604 Merge "Option "lock_path" from group "DEFAULT"
OS Version: Ubuntu 18.04 bionic

2019-06-04 12:19:19.207 | stack.sh completed in 884 seconds.
```

.

Create a VM Instance in OpenStack:



- Log in to Horizon (http://<VM_IP>/dashboard).
- **Upload a test image:**
  wget http://download.cirros-cloud.net/0.5.2/cirros-0.5.2-x86_64-disk.img
  openstack image create --disk-format qcow2 --file cirros-0.5.2-x86_64-disk.img cirros
- **Create a security group to allow SSH and ICMP:**
  openstack security group create allow_ssh_icmp
  openstack security group rule create --protocol tcp --dst-port 22 --remote-ip 0.0.0.0/0 allow_ssh_icmp
  openstack security group rule create --protocol icmp allow_ssh_icmp

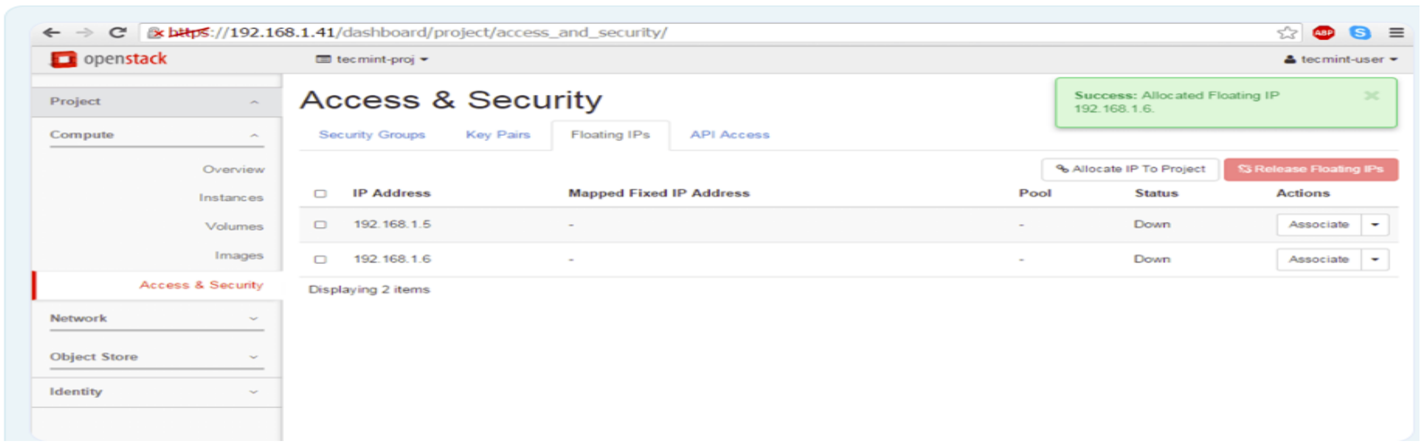
- ***Launch an instance:***

  openstack server create --image cirros --flavor m1.tiny --network private --security-group allow_ssh_icmp test-instance

- **Assign a floating IP:**

  openstack floating ip create public

  openstack server add floating ip test-instance <FLOATING_IP>
  Note the floating IP (e.g., 192.168.1.225).



Access & Security

Success: Allocated Floating IP 192.168.1.6.

	IP Address	Mapped Fixed IP Address	Pool	Status	Actions
☐	192.168.1.5	-	-	Down	Associate
☐	192.168.1.6	-	-	Down	Associate

Displaying 2 items

- **Test instance connectivity:**
  ping -c 4 <FLOATING_IP>

  ssh cirros@<FLOATING_IP>

  Use password gocubsgo for Cirros.

  Artifact: OpenStack All-in-One Installation and Configuration

Compile Results into a Report

The report summarizes the installation, configuration, and testing results for all VMs, including network connectivity and OpenStack instance creation.

Report: Cloud Computing Assignment Report

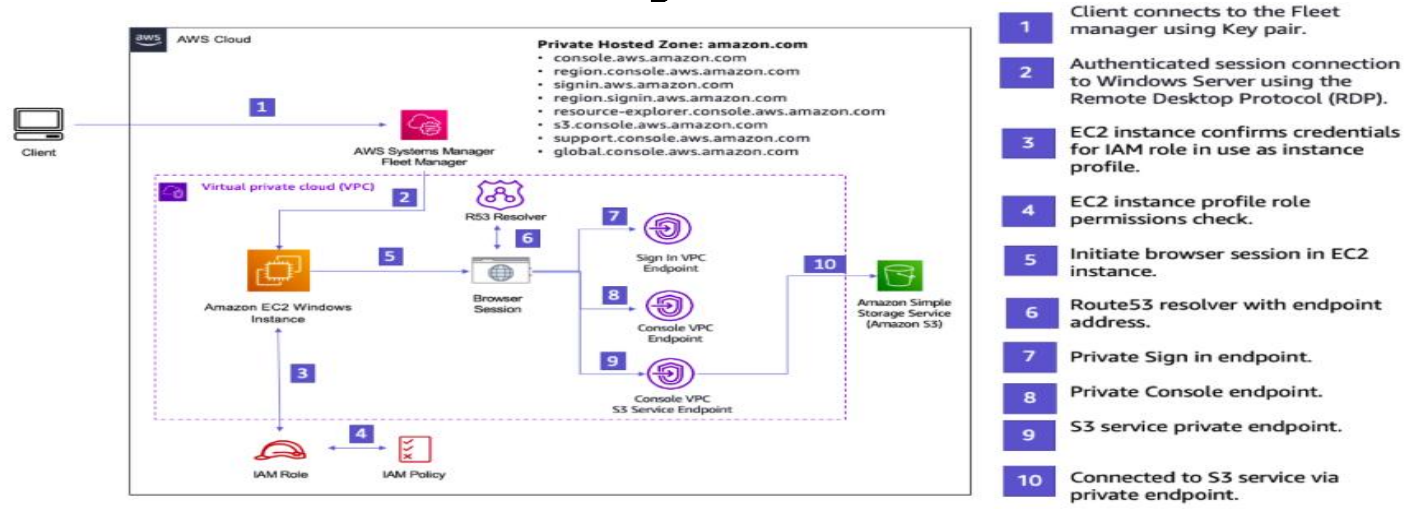
Possible Issues & Solutions

- **No Internet in VM**: Check VMware NAT service in *services.msc* on the host and ensure it's running.
- **Ping Fails**: Disable guest firewalls or allow ICMP. For OpenStack, ensure security group rules permit ICMP.
- **OpenStack Errors**: Verify nested virtualization is enabled and sufficient resources (RAM, CPU) are allocated. Check DevStack logs in /opt/stack/logs.
- **Horizon Not Accessible**: Ensure the VM's IP is correct and the host can access it. Check ufw status in Ubuntu and allow port 80.

<u>PART II Use of Cloud Computing Platforms</u>

Cloud computing platforms, such as Amazon Web Services (AWS), provide scalable, on-demand resources for building and managing virtualized environments. This section of the assignment focuses on utilizing AWS to create and configure an EC2 instance, establish network and security settings through a Virtual Private Cloud (VPC) and security groups, and manage storage using Elastic Block Store (EBS) volumes, snapshots, and Simple Storage Service (S3). These tasks demonstrate the power of cloud infrastructure for deploying secure, accessible, and efficient computing resources. The following sections outline the detailed steps, configurations, testing outcomes, encountered challenges, and solutions, compiled into a comprehensive report.

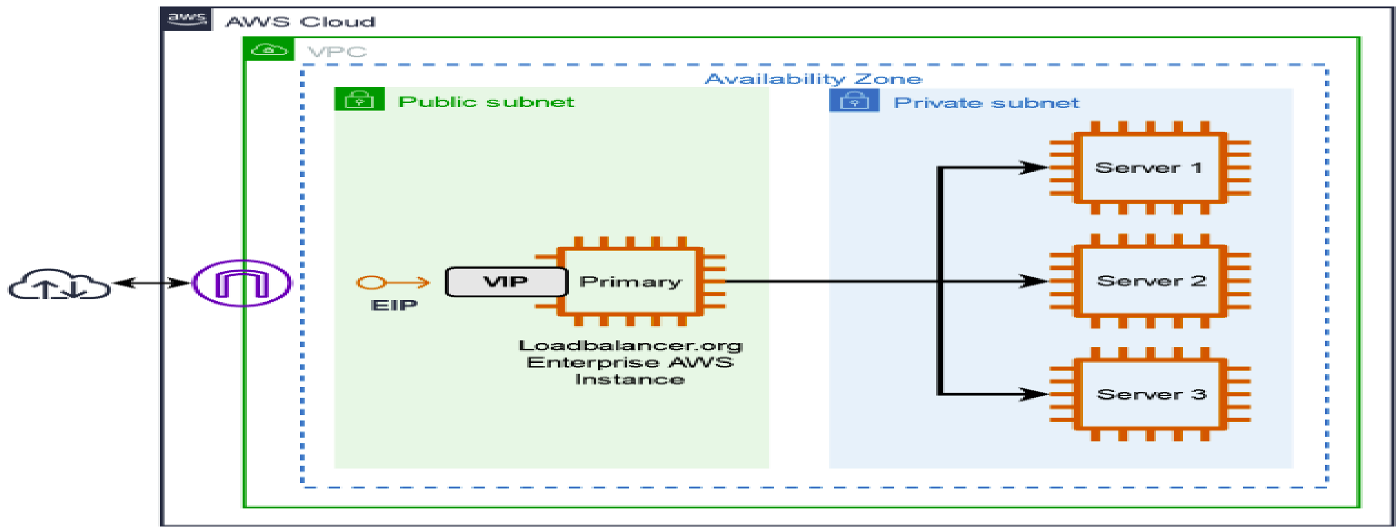
AWS Public Cloud Server Configuration

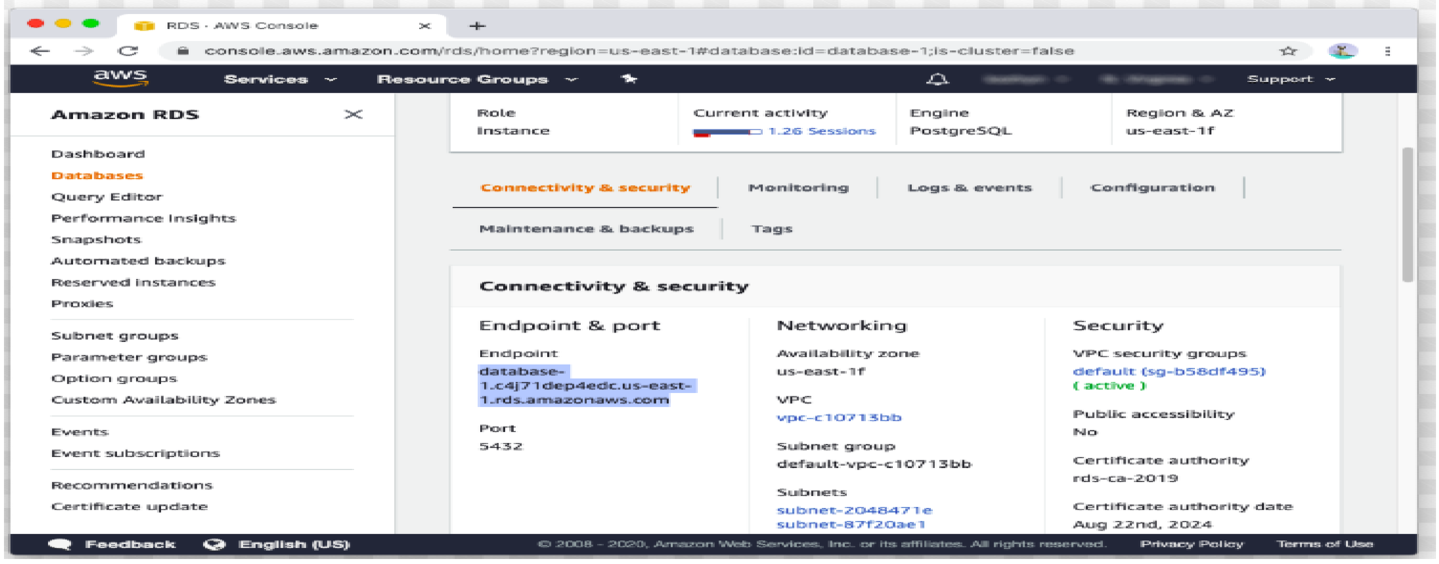


AWS Cloud
Private Hosted Zone: amazon.com
• console.aws.amazon.com
• region.console.aws.amazon.com
• signin.aws.amazon.com
• region.signin.aws.amazon.com
• resource-explorer.console.aws.amazon.com
• s3.console.aws.amazon.com
• support.console.aws.amazon.com
• global.console.aws.amazon.com
Client
1
AWS Systems Manager Fleet Manager
Virtual private cloud (VPC)
2
R53 Resolver
6
7
Sign In VPC Endpoint
5
Amazon EC2 Windows Instance
Browser Session
8
Console VPC Endpoint
10
Amazon Simple Storage Service (Amazon S3)
9
Console VPC S3 Service Endpoint
3
4
IAM Role
IAM Policy

1	Client connects to the Fleet manager using Key pair.
2	Authenticated session connection to Windows Server using the Remote Desktop Protocol (RDP).
3	EC2 instance confirms credentials for IAM role in use as instance profile.
4	EC2 instance profile role permissions check.
5	Initiate browser session in EC2 instance.
6	Route53 resolver with endpoint address.
7	Private Sign in endpoint.
8	Private Console endpoint.
9	S3 service private endpoint.
10	Connected to S3 service via private endpoint.

Amazon Elastic Compute Cloud (EC2) provides scalable virtual servers in the cloud. Creating an instance involves selecting an Amazon Machine Image (AMI), instance type, and configuring networking and security settings.





Role	Current activity	Engine	Region & AZ
Instance	1.26 Sessions	PostgreSQL	us-east-1f

Connectivity & security

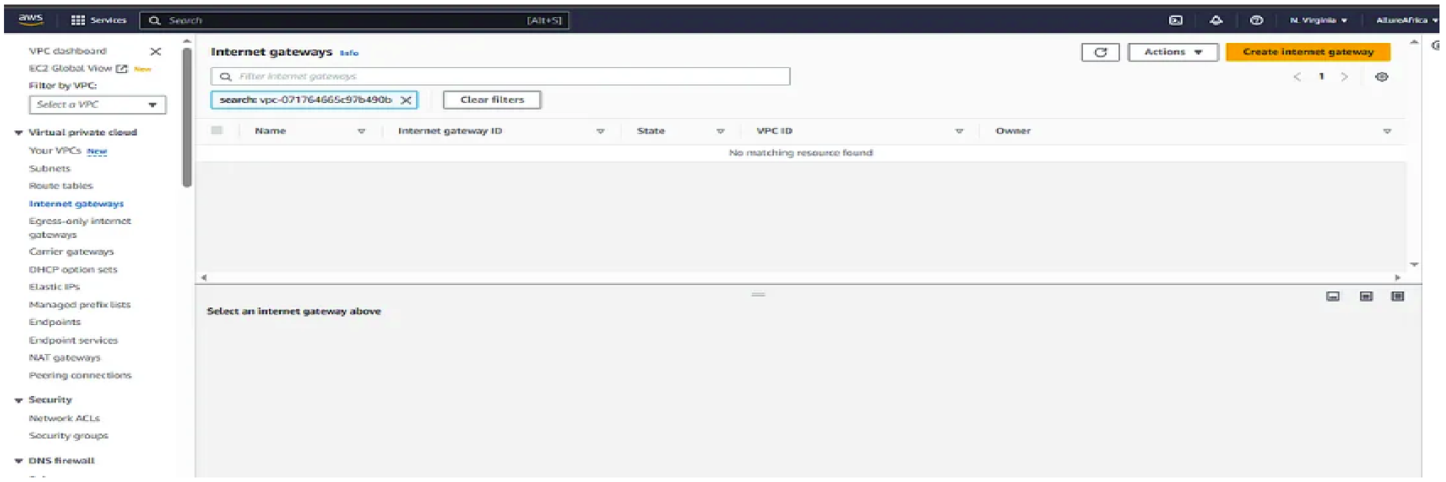
Endpoint & port
Endpoint
database-1.c4j71dep4edc.us-east-1.rds.amazonaws.com
Port
5432

Networking
Availability zone
us-east-1f
VPC
vpc-c10713bb
Subnet group
default-vpc-c10713bb
Subnets
subnet-2048471e
subnet-87f20ae1

Security
VPC security groups
default (sg-b58df495)
(active)
Public accessibility
No
Certificate authority
rds-ca-2019
Certificate authority date
Aug 22nd, 2024

Sign in to AWS Management Console:

- Select a region (e.g., US East (N. Virginia) us-east-1) from the top-right corner.



search: vpc-071764665c97b490b

Name	Internet gateway ID	State	VPC ID	Owner

No matching resource found

Select an internet gateway above

- ## Create an EC2 Instance:
- From the AWS Management Console, click **Services > EC2**.
- Under **Instances**, click **Launch Instances**.
- **Name and Tags**: Name the instance (e.g., Test-Instance).
- **Application and OS Images (AMI)**: Select **Amazon Linux 2 AMI (HVM), SSD Volume Type** (Free Tier eligible).
- **Instance Type**: Choose **t2.micro** (Free Tier eligible, 1 vCPU, 1 GB RAM).
- **Key Pair**: For now, select **Proceed without a key pair** (we'll create one later in the network section). Note: This is temporary to test instance creation; SSH requires a key pair.
- ## Network Settings:
  - Use the **default VPC** and a **public subnet** (auto-selected).
  - Click **Edit** under **Firewall (security groups)** and select **Create security group**.
  - Name: Test-SG.
  - Description: "Security group for Test-Instance".

- **Add inbound rules:**
- Type: **SSH**, Protocol: TCP, Port: 22, Source: **My IP** (your current IP, e.g., 203.0.113.25/32) to restrict access. Avoid 0.0.0.0/0 for production.
- Type: **HTTP**, Protocol: TCP, Port: 80, Source: **Anywhere-IPv4** (0.0.0.0/0) for web access testing.
  - **Configure Storage**: Keep default (1x 8 GB gp2 volume, Free Tier eligible).
  - Click **Launch Instance**.
  - Wait for the instance state to show **Running** and note the **Public IPv4 address** (e.g., 34.242.148.128).

Configure Security Group:

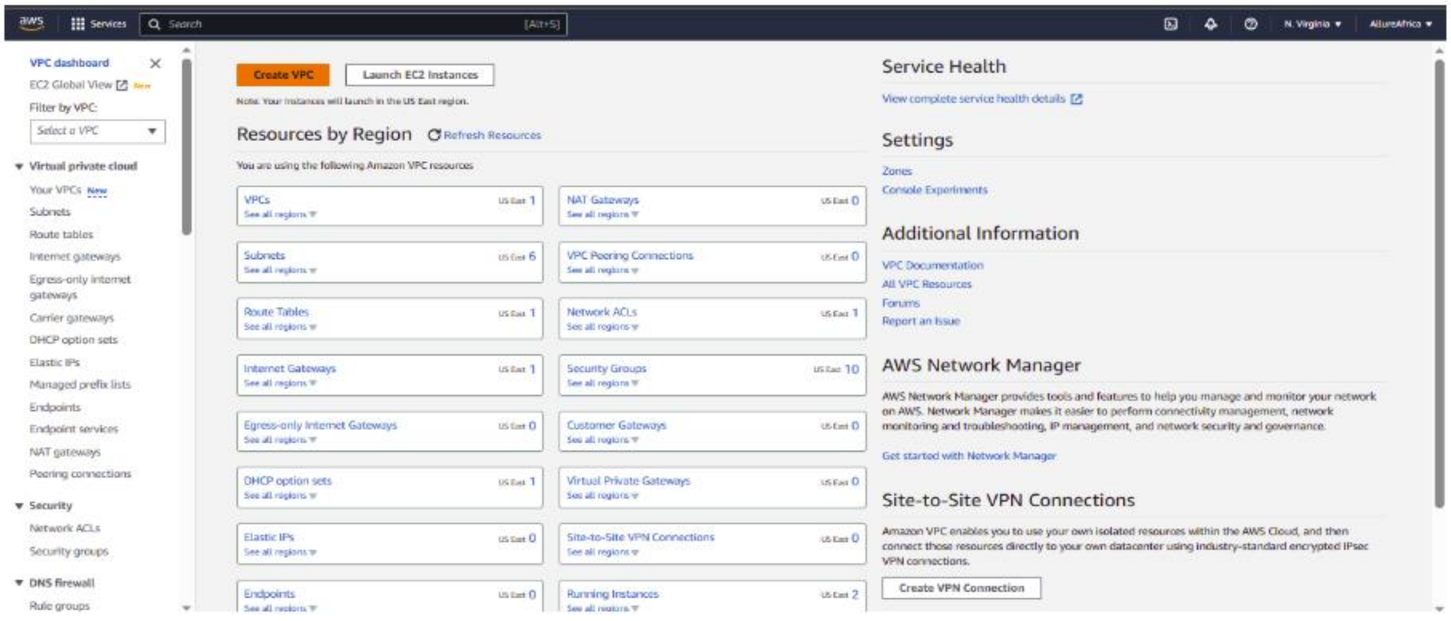
- Go to **EC2 > Security Groups**.
- Select Test-SG and verify the inbound rules:
  - SSH: Port 22, Source: Your IP.
  - HTTP: Port 80, Source: 0.0.0.0/0.
- Outbound rules: Default allows all traffic (0.0.0.0/0, all protocols).

View and Access the Instance:

- **View**: In **EC2 > Instances**, select Test-Instance. Check details like Instance ID, Public IP, VPC ID, and Subnet ID.
- **Web Access**: Since no web server is installed yet, accessing http://<Public_IP> won't work. We'll set up a web server in the network section for SSH and HTTP testing.
- **SSH Access**: Without a key pair, SSH isn't possible yet. This will be addressed in the next section.

<u>Problems and Solutions</u>:

- **Problem**: Instance launch fails due to Free Tier limits.
  - **Solution**: Ensure you haven't exceeded Free Tier limits (750 hours/month for t2.micro). Terminate unused instances. Check usage in **Billing > Bills**.
- **Problem**: Security group rules block access.
  - **Solution**: Verify your IP is correctly set in the SSH rule. Use https://checkip.amazonaws.com to confirm your public IP.



Note: Your instances will launch in the US East region.

Resources by Region

You are using the following Amazon VPC resources

VPCs	US East 1	NAT Gateways	US East 0
Subnets	US East 6	VPC Peering Connections	US East 0
Route Tables	US East 1	Network ACLs	US East 1
Internet Gateways	US East 1	Security Groups	US East 10
Egress-only Internet Gateways	US East 0	Customer Gateways	US East 0
DHCP option sets	US East 1	Virtual Private Gateways	US East 0
Elastic IPs	US East 0	Site-to-Site VPN Connections	US East 0
Endpoints	US East 0	Running Instances	US East 2

Service Health

View complete service health details

Settings

Additional Information

AWS Network Manager

AWS Network Manager provides tools and features to help you manage and monitor your network on AWS. Network Manager makes it easier to perform connectivity management, network monitoring and troubleshooting, IP management, and network security and governance.

Site-to-Site VPN Connections

Amazon VPC enables you to use your own isolated resources within the AWS Cloud, and then connect those resources directly to your own datacenter using industry-standard encrypted IPsec VPN connections.

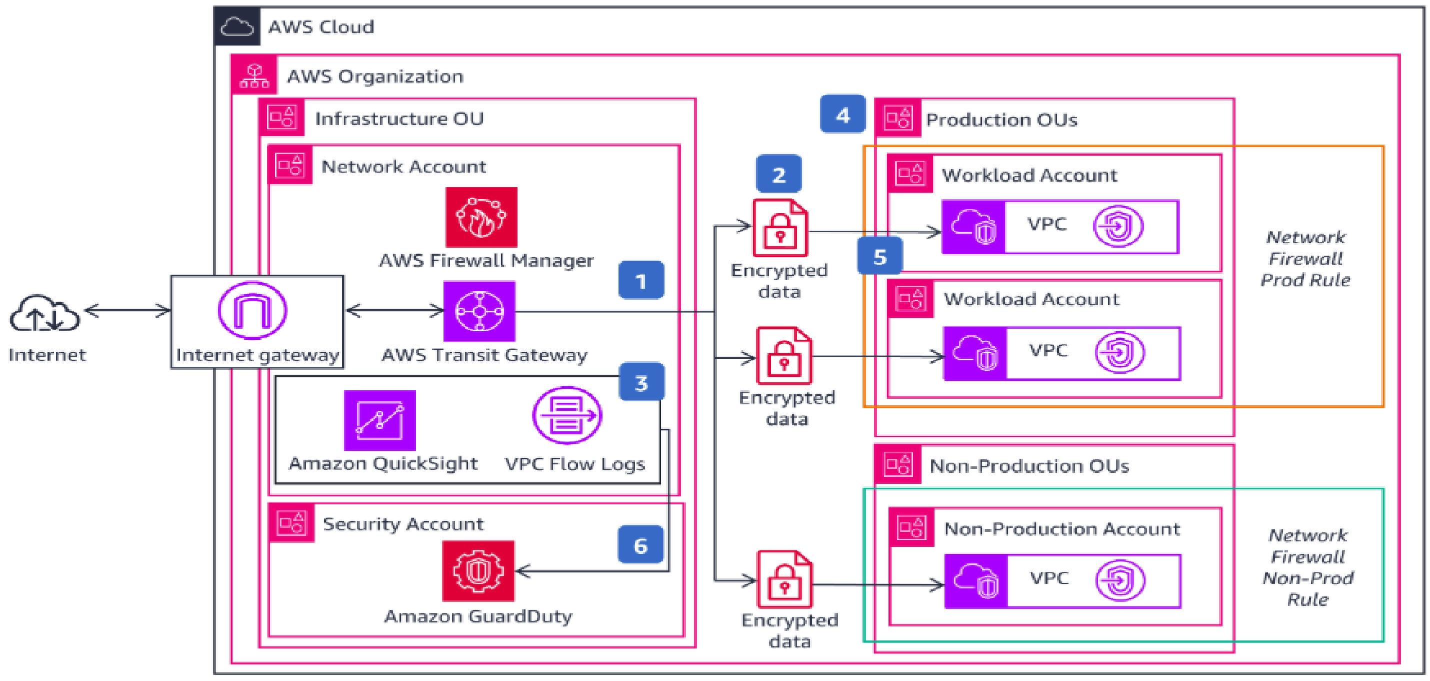
AWS EC2 Instance Creation

- AWS account with Free Tier eligibility.
- Access to AWS Management Console.

1. Sign in to AWS Console, select region (e.g., us-east-1).
2. Navigate to **EC2 > Launch Instances**.
3. Configure:
   - Name: Test-Instance.
   - AMI: Amazon Linux 2 (Free Tier).
   - Instance Type: t2.micro.
   - Key Pair: None (temporary).
   - Network: Default VPC, public subnet.
   - Security Group: Create Test-SG with:
     - SSH (Port 22, My IP).
     - HTTP (Port 80, Anywhere-IPv4).
   - Storage: 8 GB gp2 volume.
4. Launch instance and note Public IP.
5. Verify security group rules in **EC2 > Security Groups**.
6. View instance details in **EC2 > Instances**.

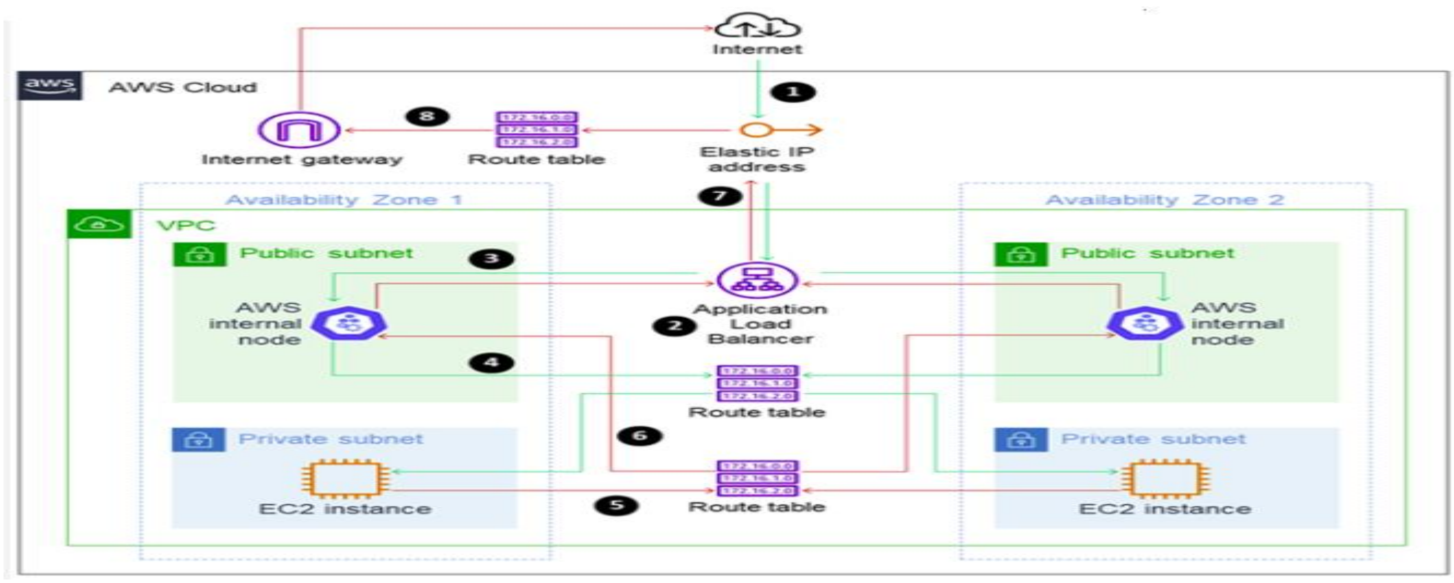
AWS Network and Security Configuration:

This section involves creating a key pair for secure SSH access, setting up a custom VPC with subnets, creating a VPC-specific security group, launching an instance in the custom VPC, and enabling web and SSH access. A VPC is a logically isolated virtual network where you can define IP ranges, subnets, and routing. A key pair ensures secure authentication, and the security group controls traffic.



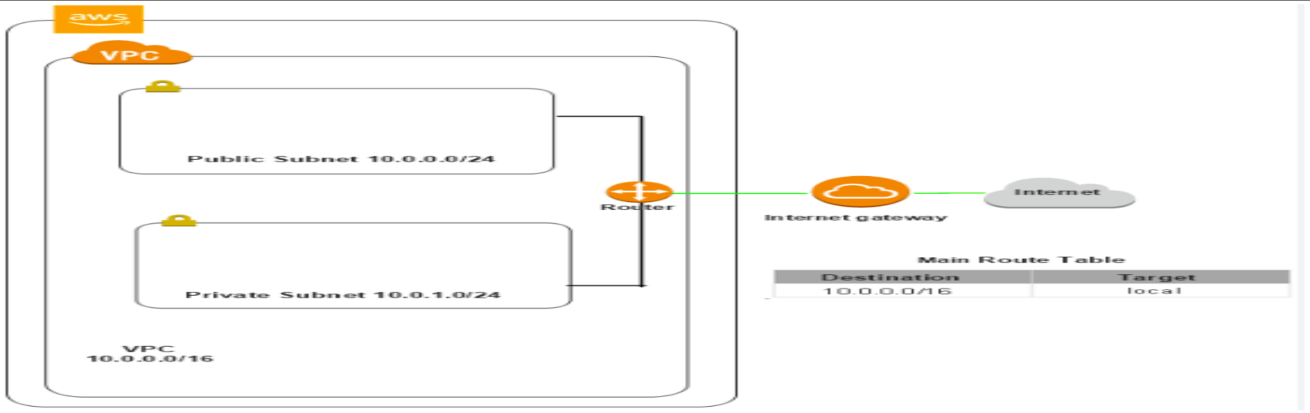
Create a Key Pair:

- In **EC2 > Key Pairs**, click **Create key pair**.
- Name: MyKeyPair.
- Key pair type: **RSA**.
- Private key file format: **.pem** (for SSH clients like OpenSSH).
- Click **Create key pair**. The .pem file (e.g., MyKeyPair.pem) downloads automatically.
- Secure the file (e.g., move to C:\Users\YourName\.ssh and set permissions on Linux/Mac: chmod 400 MyKeyPair.pem).



Create a VPC:

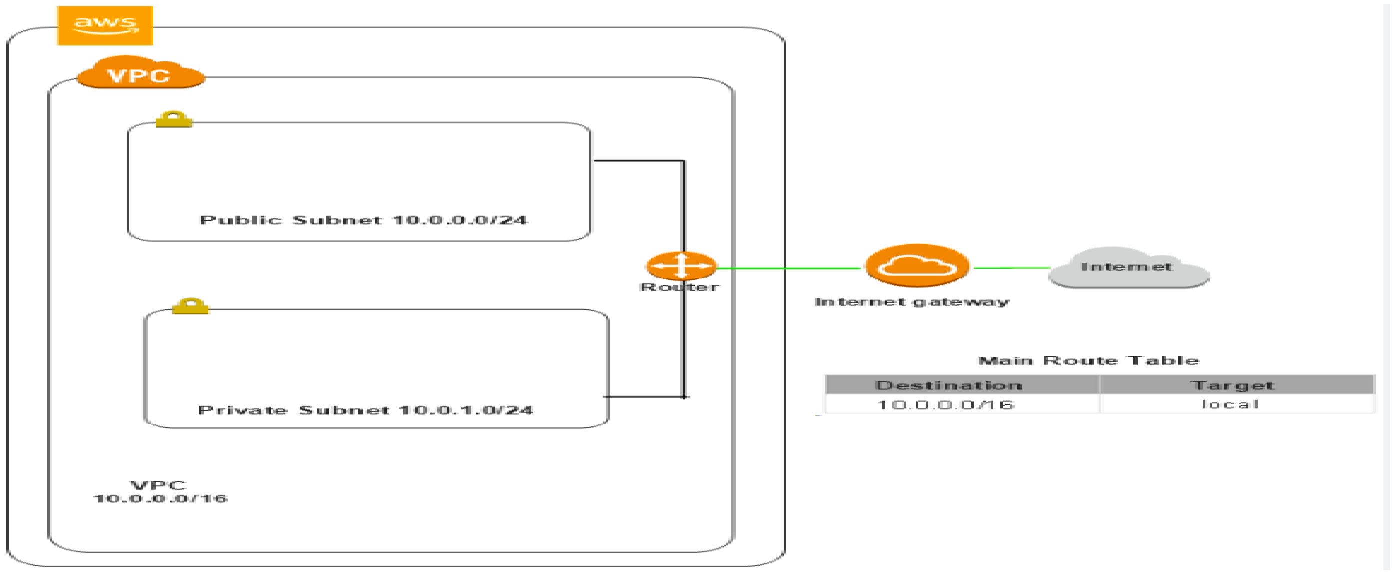
- Go to **VPC > Your VPCs > Create VPC**.
- Name: Test-VPC.
- IPv4 CIDR block: 10.0.0.0/16 (65,536 IP addresses).
- Click **Create VPC**.
- **Create Subnets**:
  - Go to **VPC > Subnets > Create subnet**.
  - VPC: Select Test-VPC.
  - Subnet 1: Name: Public-Subnet-1, Availability Zone: us-east-1a, CIDR: 10.0.1.0/24 (256 IPs).
  - Subnet 2: Name: Public-Subnet-2, Availability Zone: us-east-1b, CIDR: 10.0.2.0/24.
  - Click **Create subnet**.
- **Enable Auto-Assign Public IP**:
  - Select Public-Subnet-1, click **Actions > Edit subnet settings**.
  - Check **Enable auto-assign public IPv4 address** and save.
- **Create Internet Gateway**:
  - Go to **VPC > Internet Gateways > Create internet gateway**.
  - Name: Test-IGW.



Main Route Table

Destination	Target
10.0.0.0/16	local

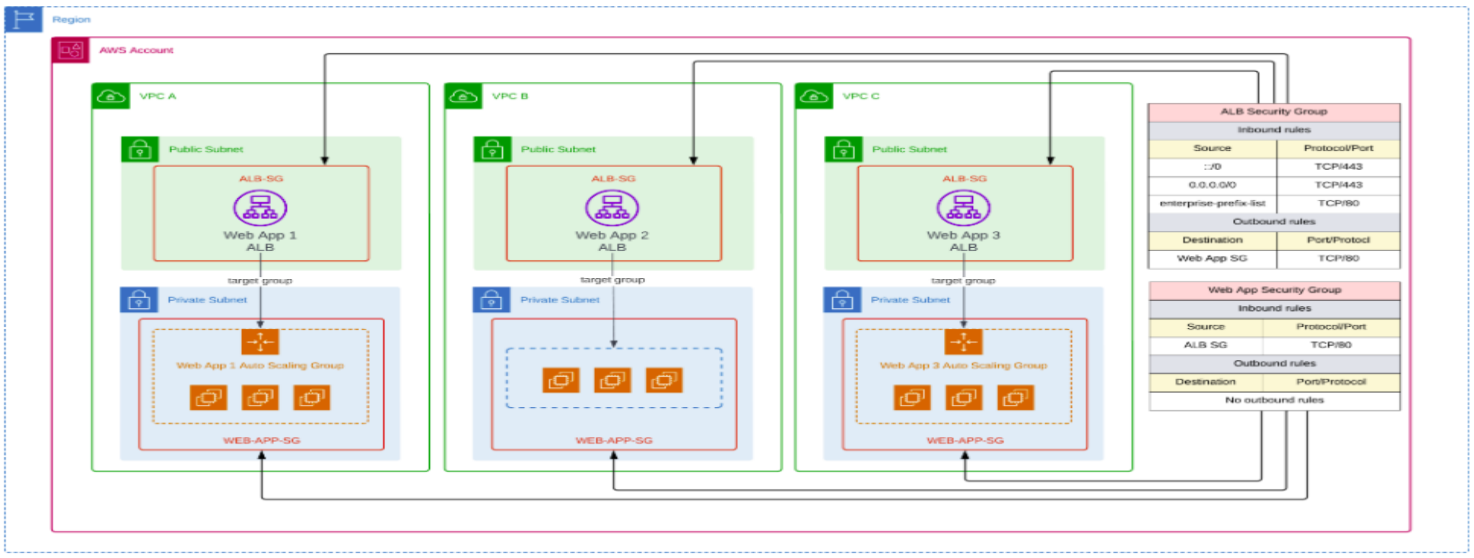
- Attach to Test-VPC via **Actions > Attach to VPC**.



Main Route Table

Destination	Target
10.0.0.0/16	local

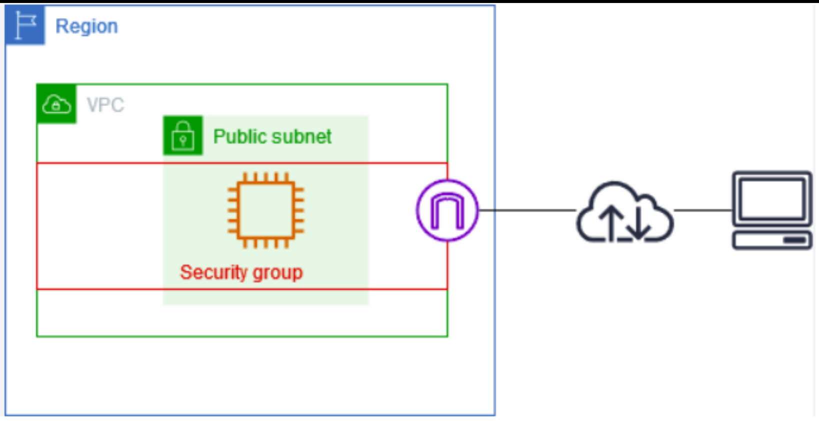
- ## Update Route Table:
  - Go to **VPC > Route Tables**, select the route table for Test-VPC.
  - Click **Routes > Edit routes > Add route**.
  - Destination: 0.0.0.0/0, Target: igw-xxxxxxxx (Test-IGW ID).
  - Save routes.
  - Associate Public-Subnet-1 and Public-Subnet-2 with this route table via **Subnet associations**.



ALB Security Group	
Inbound rules	
Source	Protocol/Port
::/0	TCP/443
0.0.0.0/0	TCP/443
enterprise-prefix-list	TCP/80
Outbound rules	
Destination	Port/Protocl
Web App SG	TCP/80

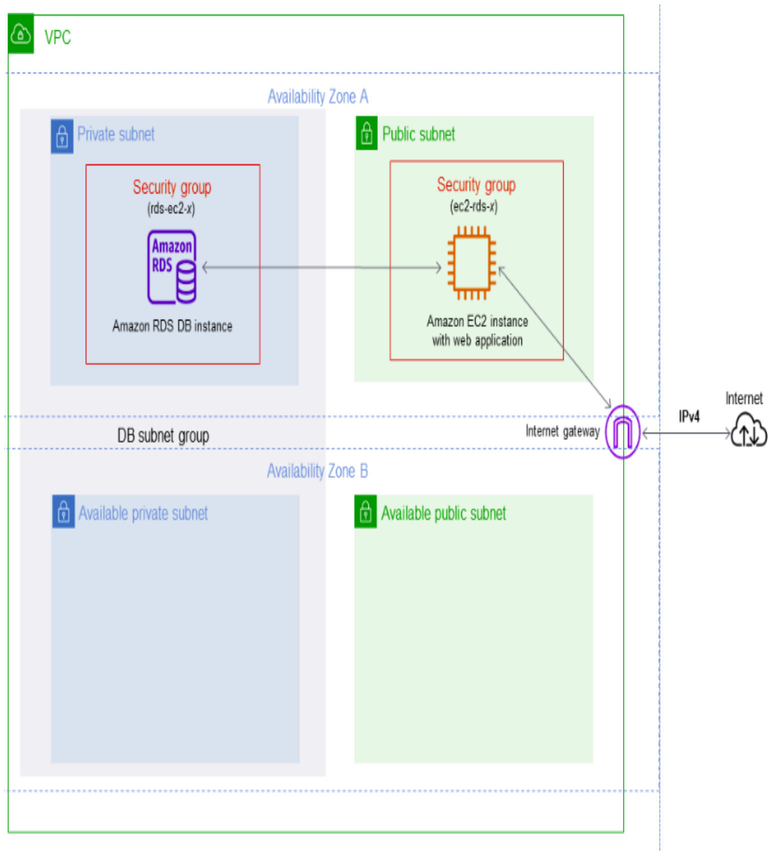
Web App Security Group	
Inbound rules	
Source	Protocol/Port
ALB SG	TCP/80
Outbound rules	
Destination	Port/Protocol
No outbound rules	

- **Create a VPC Security Group**:
  - In **VPC > Security Groups**, click **Create security group**.
  - Name: VPC-SG.
  - Description: "Security group for Test-VPC instance".
  - VPC: Select Test-VPC.
  - Inbound rules:
    - Type: **SSH**, Port: 22, Source: **My IP**.
    - Type: **HTTP**, Port: 80, Source: **Anywhere-IPv4** (0.0.0.0/0).
  - Outbound rules: Keep default (all traffic allowed).
  - Click **Create security group**.



Create an Instance in the VPC:

- Go to **EC2 > Launch Instances**.
- Name: VPC-Instance.
- AMI: Amazon Linux 2.
- Instance Type: t2.micro.
- Key Pair: Select MyKeyPair.
- Network:
  - VPC: Test-VPC.
  - Subnet: Public-Subnet-1.
  - Auto-assign Public IP: Enable.
- Firewall: Select existing security group VPC-SG.
- Storage: 8 GB gp2 volume.



- **User Data** (to install and start a web server):

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
echo "<h1>Hello from AWS EC2</h1>" > /var/www/html/index.html
```

- Click **Launch Instance**.
- Note the Public IP (e.g., 54.123.45.67).

Perform Web and SSH Access:

- **Web Access**:
  - Open a browser and navigate to http://<Public_IP>.
  - Expected result: Displays "Hello from AWS EC2".
- **SSH Access**:
  - On Windows, use an SSH client like PuTTY or Windows Terminal.
  - Command (Linux/Mac or Windows Terminal):

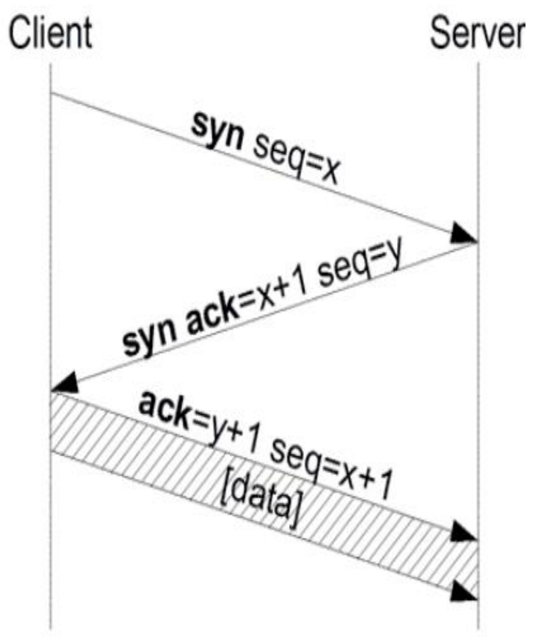
```
ssh -i ~/.ssh/MyKeyPair.pem ec2-user@<Public_IP>
```

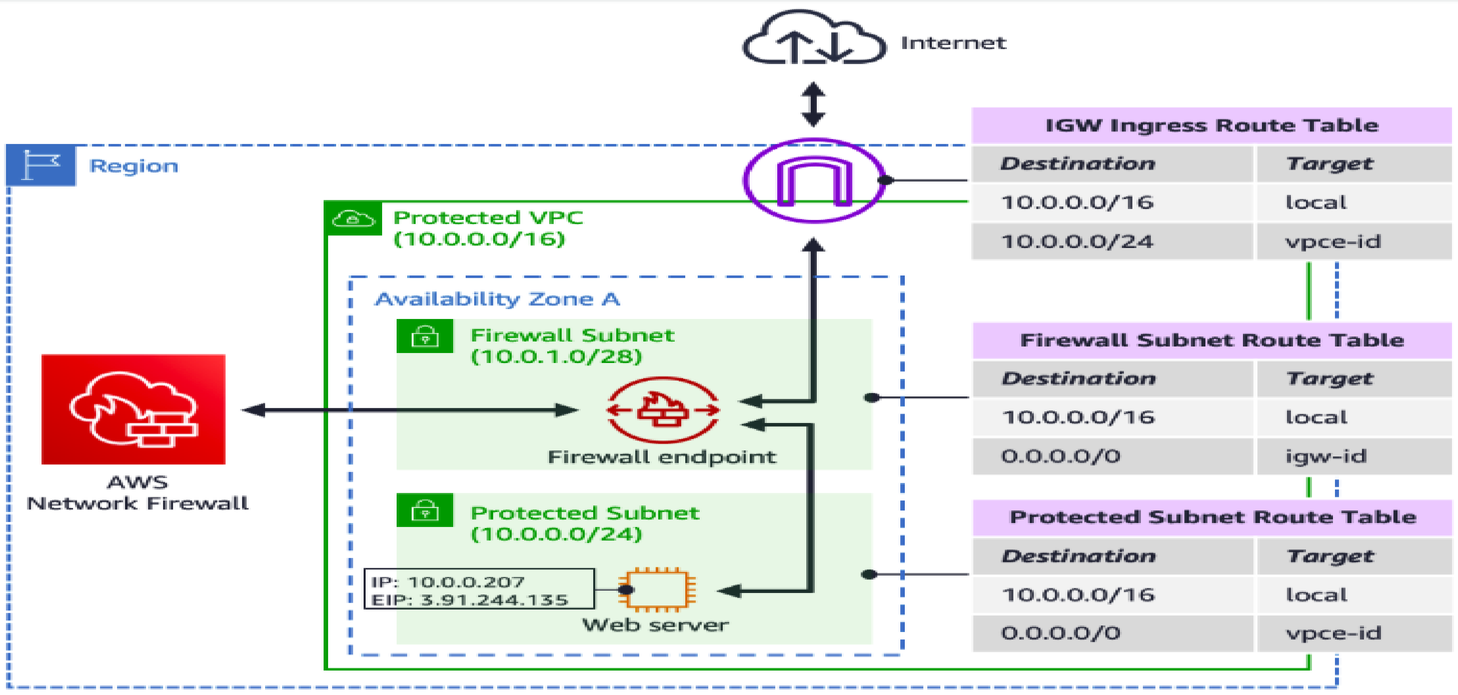
  - **Expected result:** Connects to the instance’s shell.
  - If using PuTTY, convert .pem to .ppk using PuTTYgen and configure the session with ec2-user@<Public_IP>.

Problems and Solutions:

- **Problem**: SSH connection refused or timed out.
  - **Solution**: Verify security group allows SSH from your IP. Check if the instance is in a public subnet with an Internet Gateway. Ensure the .pem file permissions are correct (chmod 400 on Linux/Mac).
- **Problem**: Web page not accessible.
  - **Solution**: Confirm HTTP rule in security group. Verify httpd is running (systemctl status httpd on the instance). Check route table for 0.0.0.0/0 to Internet Gateway.
- **Problem**: VPC instance has no public IP.
  - **Solution**: Ensure **Auto-assign public IPv4 address** is enabled for the subnet.
    AWS Network and Security Configuration Steps

- AWS account with Free Tier eligibility.
- AWS Management Console access.





IGW Ingress Route Table

Destination	*Target*
10.0.0.0/16	local
10.0.0.0/24	vpce-id

Firewall Subnet Route Table

Destination	*Target*
10.0.0.0/16	local
0.0.0.0/0	igw-id

Protected Subnet Route Table

Destination	*Target*
10.0.0.0/16	local
0.0.0.0/0	vpce-id

Create Key Pair:

- In **EC2 > Key Pairs**, create MyKeyPair (RSA, .pem).
- Download and secure the .pem file.

Create VPC:

- In **VPC > Your VPCs**, create Test-VPC (CIDR: 10.0.0.0/16).
- Create subnets: Public-Subnet-1 (10.0.1.0/24, us-east-1a), Public-Subnet-2 (10.0.2.0/24, us-east-1b).
- Enable auto-assign public IP for Public-Subnet-1.
- Create and attach Test-IGW to Test-VPC.
- Update route table: Add 0.0.0.0/0 to igw-xxxxxxxx.

Create Security Group:

- In **VPC > Security Groups**, create VPC-SG for Test-VPC.
- Inbound: SSH (Port 22, My IP), HTTP (Port 80, Anywhere-IPv4).

Launch Instance:

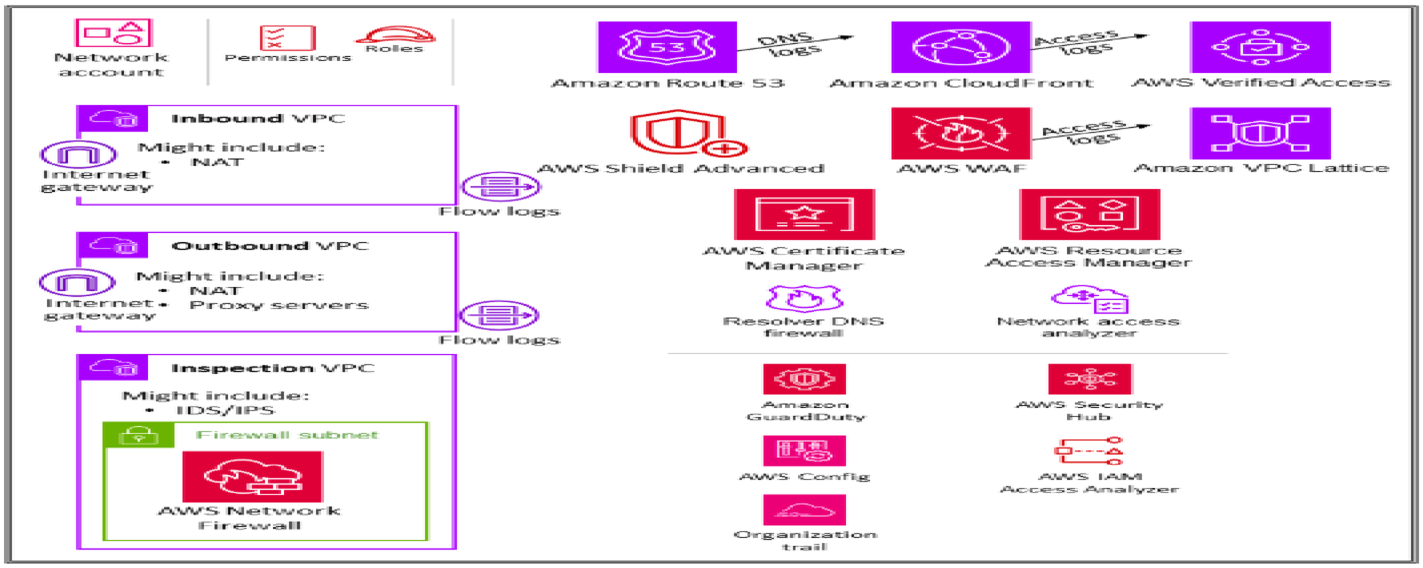
- In **EC2 > Launch Instances**, create VPC-Instance:
  - AMI: Amazon Linux 2.
  - Type: t2.micro.
  - Key Pair: MyKeyPair.
  - Network: Test-VPC, Public-Subnet-1, enable public IP.
  - Security Group: VPC-SG.
  - User Data: Install httpd and create index.html.

Access Instance:

- *Web: http://<Public_IP> (displays "Hello from AWS EC2").*
- *SSH: ssh -i MyKeyPair.pem ec2-user@<Public_IP>.*

Step 3: AWS Storage Configuration

AWS storage services include EBS for block storage, EBS snapshots for backups, and S3 for object storage. We'll create and attach an EBS volume to the VPC-Instance, create a snapshot, and use S3 to upload and download files.



Create and Mount EBS Volume:

- Go to **EC2 > Volumes > Create Volume**.
- Size: 1 GB (Free Tier eligible).
- Availability Zone: us-east-1a (match VPC-Instance subnet).
- Click **Create Volume**.
- **Attach Volume**:
  - Select the volume, click **Actions > Attach Volume**.
  - Choose VPC-Instance (e.g., i-xxxxxxxx).
- **Mount Volume**:
  - *SSH into VPC-Instance:*

    ssh -i ~/.ssh/MyKeyPair.pem ec2-user@<Public_IP>
  - **Check for the new volume:**

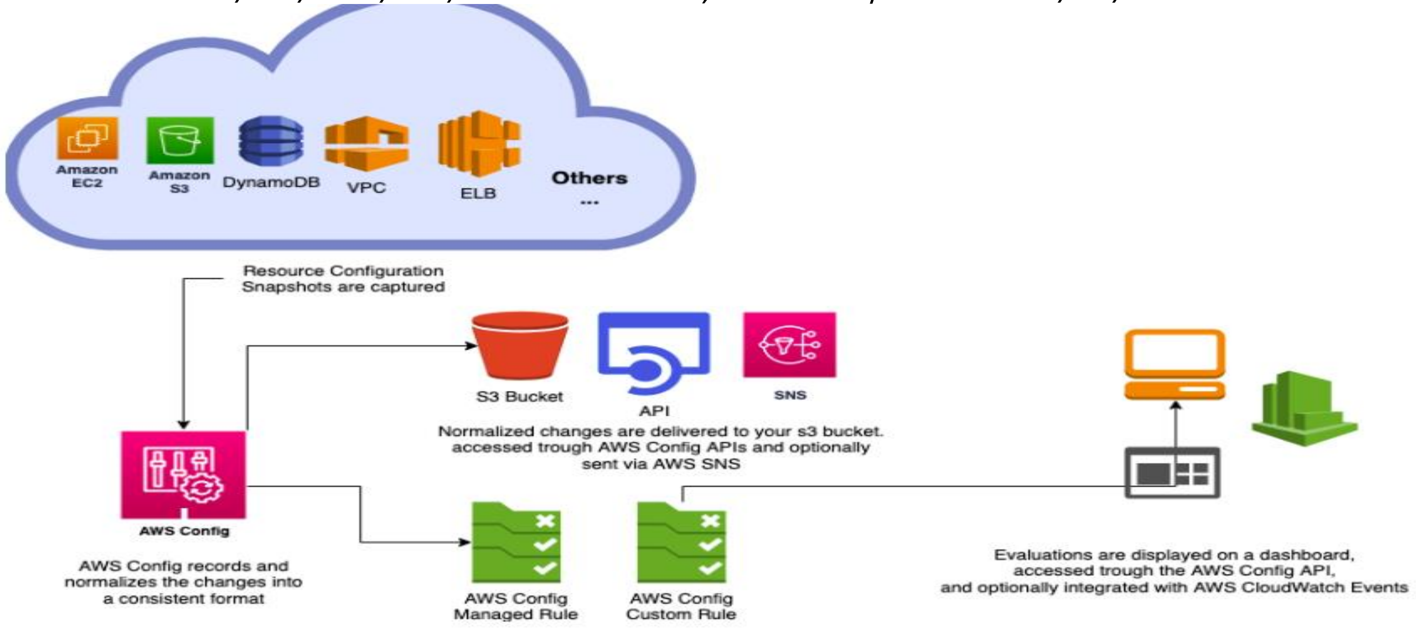
    lsblk
    Expected: /dev/xvdf (or similar) as unmounted.
  - **Format and mount:**

    sudo mkfs.ext4 /dev/xvdf

    sudo mkdir /mnt/ebs

    sudo mount /dev/xvdf /mnt/ebs
  - **Verify:**

    df -h
    Expected: /mnt/ebs shows 1 GB.
  - Make persistent
  - *echo "/dev/xvdf /mnt/ebs ext4 defaults,nofail 0 2" | sudo tee -a /etc/fstab*


Amazon EC2
Amazon S3
DynamoDB
VPC
ELB
Others ...
Resource Configuration Snapshots are captured
S3 Bucket
API
SNS
Normalized changes are delivered to your s3 bucket. accessed trough AWS Config APIs and optionally sent via AWS SNS
AWS Config
AWS Config records and normalizes the changes into a consistent format
AWS Config Managed Rule
AWS Config Custom Rule
Evaluations are displayed on a dashboard, accessed trough the AWS Config API, and optionally integrated with AWS CloudWatch Events

Create and Use Snapshots:

- In **EC2 > Snapshots > Create Snapshot**.
- Resource Type: Volume, select the 1 GB volume attached to VPC-Instance.
- Description: "Snapshot of EBS volume".
- Click **Create Snapshot**.

- Wait for status to change to **completed**.
- **Use Snapshot** (create a new volume):
  - Select the snapshot, click **Actions > Create Volume from Snapshot**.
  - Size: 1 GB, Availability Zone: us-east-1a.
  - Click **Create Volume**.
  - Attach to VPC-Instance and mount as above (e.g., to /mnt/ebs2).

Use S3 for Object Storage:

- **Create S3 Bucket**:
  - Go to **S3 > Buckets > Create bucket**.
  - Bucket Name: my-unique-bucket-2025 (must be globally unique).
  - Region: us-east-1.
  - Keep default settings (block public access for security).
  - Click **Create bucket**.
- **Upload File**:
  - SSH into VPC-Instance.
  - Create a test file:

    echo "Test file for S3" > test.txt
  - Install AWS CLI:

    sudo yum install -y awscli
  - Configure AWS CLI (use IAM user credentials with S3 permissions):

    aws configure

    Enter Access Key ID, Secret Access Key, region (us-east-1), and output format (json).
  - Upload:

    aws s3 cp test.txt s3://my-unique-bucket-2025/

- ## Download File:
  - Download to a new location:

    aws s3 cp s3://my-unique-bucket-2025/test.txt /home/ec2-user/test-download.txt
  - **Verify:**

    cat test-download.txt
    Expected: "Test file for S3".

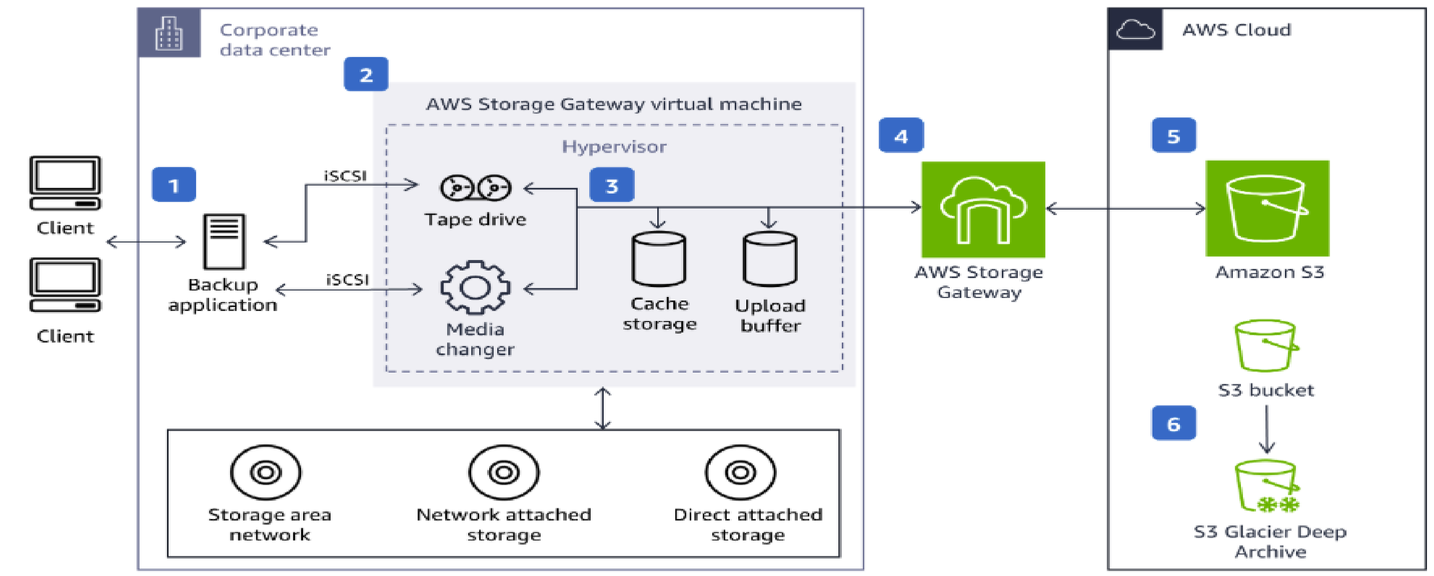
Problems and Solutions:

- **Problem**: EBS volume not visible in lsblk.
  - **Solution**: Ensure the volume is attached to the correct instance and Availability Zone. Reboot the instance if necessary.
- **Problem**: S3 upload fails with access denied.
  - **Solution**: Verify IAM user has AmazonS3FullAccess policy attached. Check bucket permissions.
- **Problem**: Snapshot creation stuck in pending.
  - **Solution**: Wait longer (large volumes take time). Ensure the volume is not heavily used during snapshot creation.

AWS Storage Configuration Steps

Prerequisites

- Running EC2 instance (VPC-Instance) with SSH access.



Create and Mount EBS Volume:

- In **EC2 > Volumes**, create a 1 GB volume in us-east-1a.
- Attach to VPC-Instance.
- SSH into instance and run:

  sudo mkfs.ext4 /dev/xvdf

  sudo mkdir /mnt/ebs

  sudo mount /dev/xvdf /mnt/ebs
- *Verify: df -h.*

Create and Use Snapshot:

- In **EC2 > Snapshots**, create a snapshot of the 1 GB volume.
- Create a new volume from the snapshot and attach to VPC-Instance.
- Mount to /mnt/ebs2.

S3 Operations:

- Create bucket my-unique-bucket-2025 in **S3 > Buckets**.
- SSH into instance, install AWS CLI: sudo yum install -y awscli.
- Configure AWS CLI with IAM credentials.
- *Upload: aws s3 cp test.txt s3://my-unique-bucket-2025/.*
- *Download: aws s3 cp s3://my-unique-bucket-2025/test.txt test-download.txt.*

Compile Results into a Report

The report summarizes the key steps, configurations, problems encountered, and solutions for the AWS configurations. It's written in LaTeX for professional formatting and includes all required elements.

```
|documentclass{article}
|usepackage[utf8]{inputenc}
|usepackage{geometry}
|geometry{a4paper, margin=1in}
|usepackage{listings}
|usepackage{xcolor}

|lstset{
  basicstyle=|ttfamily|small,
    breaklines=true,
    frame=single,
    numbers=left,
    numberstyle=|tiny,
    keywordstyle=|color{blue},
  }

  |title{AWS Cloud Computing Assignment Report}
  |author{Your Name}
  |date{May 25, 2025}

  |begin{document}

  |maketitle
```

This report details the configuration of an AWS public cloud server, network, security, and storage components as part of a cloud computing assignment. Tasks include creating an EC2 instance, configuring security groups, setting up a VPC, enabling SSH and web access, and managing EBS volumes, snapshots, and S3 storage. Key steps, configurations, problems, and solutions are summarized.

```
|section{AWS Public Cloud Server Configuration}
|subsection{Key Steps}
|begin{itemize}
  |item Signed into AWS Console, selected |texttt{us-east-1}.
  |item Launched EC2 instance (|texttt{Test-Instance}):
    |begin{itemize}
      |item AMI: Amazon Linux 2.
      |item Type: t2.micro.
      |item Security Group: |texttt{Test-SG} with SSH (Port 22, My IP) and HTTP (Port 80, Anywhere).
      |item Storage: 8 GB gp2 volume.
    |end{itemize}
  |item Viewed instance details (Public IP, Instance ID).
|end{itemize}

|subsection{Problems and Solutions}
|begin{itemize}
  |item |textbf{Problem}: Instance launch failed due to Free Tier limits.
    |textbf{Solution}: Terminated unused instances, verified Free Tier eligibility in Billing.
  |item |textbf{Problem}: Security group blocked access.
    |textbf{Solution}: Confirmed My IP in SSH rule using |texttt{https://checkip.amazonaws.com}.
|end{itemize}

|section{AWS Network and Security Configuration}
|subsection{Key Steps}
|begin{itemize}
  |item Created key pair |texttt{MyKeyPair} (.pem, RSA).
  |item Created |texttt{Test-VPC} (CIDR: |texttt{10.0.0.0/16}):
    |begin{itemize}
      |item Subnets: |texttt{Public-Subnet-1} (|texttt{10.0.1.0/24}), |texttt{Public-Subnet-2} (|texttt{10.0.2.0/24}).
      |item Enabled auto-assign public IP for |texttt{Public-Subnet-1}.
      |item Attached Internet Gateway (|texttt{Test-IGW}).
      |item Updated route table: |texttt{0.0.0.0/0} to |texttt{igw-xxxxxxxx}.
    |end{itemize}
  |item Created security group |texttt{VPC-SG} for |texttt{Test-VPC} with SSH (Port 22, My IP) and HTTP (Port 80,
Anywhere).
  |item Launched |texttt{VPC-Instance} in |texttt{Test-VPC}:
    |begin{itemize}
      |item Used |texttt{MyKeyPair}, |texttt{Public-Subnet-1}, |texttt{VPC-SG}.
      |item User Data: Installed |texttt{httpd} and created |texttt{index.html}.
    |end{itemize}
  |item Accessed via:
    |begin{itemize}
      |item Web: |texttt{http://<Public_IP>} displayed "Hello from AWS EC2".
      |item SSH: |texttt{ssh -i MyKeyPair.pem ec2-user@<Public_IP>}.
    |end{itemize}
|end{itemize}

|subsection{Problems and Solutions}
|begin{itemize}
  |item |textbf{Problem}: SSH connection timed out.
    |textbf{Solution}: Verified security group, subnet public IP settings, and .pem file permissions.
  |item |textbf{Problem}: Web page not accessible.
    |textbf{Solution}: Confirmed |texttt{httpd} running and HTTP rule in security group.
|end{itemize}

|section{AWS Storage Configuration}
|subsection{Key Steps}
|begin{itemize}
  |item Created 1 GB EBS volume, attached to |texttt{VPC-Instance}, and mounted:
    |begin{lstlisting}
sudo mkfs.ext4 /dev/xvdf
sudo mkdir /mnt/ebs
sudo mount /dev/xvdf /mnt/ebs
    |end{lstlisting}
  |item Created snapshot of EBS volume and restored to a new volume.
  |item Created S3 bucket |texttt{my-unique-bucket-2025}.
  |item Installed AWS CLI, uploaded/downloaded |texttt{test.txt}:
    |begin{lstlisting}
aws s3 cp test.txt s3://my-unique-bucket-2025/
aws s3 cp s3://my-unique-bucket-2025/test.txt test-download.txt
    |end{lstlisting}
|end{itemize}

|subsection{Problems and Solutions}
|begin{itemize}
  |item |textbf{Problem}: EBS volume not visible.
    |textbf{Solution}: Ensured correct Availability Zone and rebooted instance.
  |item |textbf{Problem}: S3 access denied.
    |textbf{Solution}: Attached |texttt{AmazonS3FullAccess} policy to IAM user.
|end{itemize}

|section
```

Conclusion

All AWS tasks were completed successfully, including EC2 instance creation, VPC setup, security configuration, and storage management. Web and SSH access were verified, and storage operations (EBS, snapshots, S3) functioned as expected. Proper security practices (e.g., restricting SSH to My IP) were followed to enhance safety.

\end