

3-Tier Architecture

This project explains how a 3-tier architecture is implemented on Microsoft Azure to create a secure and scalable web application. The architecture is divided into three layers: the Web Server, the Application Server, and the Database Server. Each layer is deployed on a separate Azure Virtual Machine to improve security and make management easier.

The Web Server is the first point of contact for users and handles incoming requests. These requests are then forwarded to the Application Server, which runs Apache Tomcat on port 8080 and processes the application logic. The Database Server uses MySQL on port 3306 to store and manage data. It is placed in a private network so that only the Application Server can access it, keeping the data safe.

Azure services such as Virtual Networks (VNets), subnets, and Network Security Groups (NSGs) are used to control communication between the servers and block unauthorized access.

Requirements:

Azure Free Trail Subscription

Resource Group

Virtual Network (VNet)

Subnets-WebServer , AppServer , DB server

Network Security Group (NSG)

Steps to build 3 – tier Architecture:

Step 1: Create a Resource group:

The screenshot shows the 'Create a resource group' wizard in the Microsoft Azure portal. The 'Basics' tab is selected. The 'Subscription' field is set to 'Azure subscription 1'. The 'Resource group name' field contains 'RG-GANESH01'. The 'Region' field is set to '(Asia Pacific) Australia East'. At the bottom, there are 'Previous', 'Next', and 'Review + create' buttons.

Select a region of your choice and assign a name to your Resource Group (RG) and then click on **Review + Create**.

Step 2: Create a Virtual Network (VNet):

Type **Virtual Network** in the search box, then click **Create** to start setting it up.

The screenshot shows the 'Create virtual network' wizard in the Microsoft Azure portal. The 'Basics' tab is selected. In the 'Project details' section, the subscription is set to 'Azure subscription 1' and the resource group is 'RG-GANESH01'. In the 'Instance details' section, the virtual network name is 'VNET-GANESH-WEB' and the region is '(Asia Pacific) Australia East'. At the bottom, there are 'Previous', 'Next', and 'Review + create' buttons.

Select the Resource Group in which the Virtual Network will be deployed. Provide a name for the Virtual Network, choose a region, and then click **Next** to proceed to the IP Address section.

Note: The region of the Resource Group and the Virtual Network does not need to be the same. Using different regions does not impact architecture.

The screenshot shows the 'Create virtual network' wizard in the Microsoft Azure portal. The 'IP addresses' tab is selected. An IPv4 address space '10.1.0.0/16' is defined, covering the range 10.1.0.0 - 10.1.255.255 with 65,536 addresses. Two subnets are configured: 'WEB-SNET' with IP range 10.1.1.0 - 10.1.1.255 and size /24 (256 addresses), and 'APP-SNET' with IP range 10.1.2.0 - 10.1.2.255 and size /24 (256 addresses). There is also an option to 'Add IPv4 address space'. At the bottom, there are 'Previous', 'Next', and 'Review + create' buttons.

Configure the IP address space and CIDR range as per the architecture requirements. Update the subnet name and settings accordingly.

In above image we configured two subnets, one for webserver and another one for appserver.

The screenshot shows the 'Create virtual network' wizard in the Azure portal. The 'Basics' tab is selected. In the 'Project details' section, 'Subscription' is set to 'Azure subscription 1' and 'Resource group' is set to 'RG-GANESH01'. In the 'Instance details' section, 'Virtual network name' is 'VNET-GANESH-DB' and 'Region' is '(Asia Pacific) Australia Central'. At the bottom, there are 'Previous', 'Next', and 'Review + create' buttons.

Choose a region and click **Next** to navigate to the IP Address configuration page.

The screenshot shows the 'Create virtual network' wizard in the Azure portal. The 'IP addresses' tab is selected. Under 'Allocate using IP address pools', there is a checkbox labeled 'Allocate using IP address pools. Learn more'. Below it, there is a table for adding subnets:

Subnets	IP address range	Size	NAT gateway
DB-SNET	10.2.1.0 - 10.2.1.255	/24 (256 addresses)	-

At the bottom, there are 'Add IPv4 address space' and 'Review + create' buttons.

Note: We are creating two VNet's in two different regions because, Azure Free trial doesn't provide **1vcpu** service (Total of 4vcpu's). And here we need three Servers so we use two different VNets in two different regions

Step 3: Create Virtual Machines

Now its time to create VM's.

WEB-SERVER(VM):

Search for Virtual Machines and click on create and select first option (virtual machine).

This screenshot shows the 'Create a virtual machine' wizard on the Microsoft Azure portal. The user is on the 'Set instance details' step. They have selected 'Help me create a low cost VM'. Under 'Subscription', 'Azure subscription 1' is chosen. Under 'Resource group', 'RG-GANESH01' is selected. In the 'Instance details' section, the 'Virtual machine name' is set to 'GANESH-WEBSERVER', 'Region' is '(Asia Pacific) Australia East', 'Availability options' is 'No infrastructure redundancy required', 'Security type' is 'Standard', and 'Image' is 'Ubuntu Server 24.04 LTS - x64 Gen2 (free services eligible)'. The 'VM architecture' is 'Arm64'. At the bottom, there are buttons for '< Previous', 'Next : Disks >', and 'Review + create'.

Choose the Resource Group, assign a name to the virtual machine, and select the region in which the initial Virtual Network (GANESH-web-vnet) was created. Configure the necessary availability options, select the Ubuntu image, and scroll down to proceed.

This screenshot shows the 'Create a virtual machine' wizard on the Microsoft Azure portal. The user is on the 'Set administrator account' step. They have selected 'Help me choose the right VM size for my workload'. Under 'Enable Hibernation', there is a note: 'Hibernate is not supported by the size that you have selected. Choose a size that is compatible with Hibernate to enable this feature.' Under 'Administrator account', 'Authentication type' is set to 'Password', and the 'Username' is 'GANESH'. The 'Password' and 'Confirm password' fields both contain masked text. Under 'Inbound port rules', it says 'Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.' The 'Public inbound ports' setting is 'Allow selected ports'. At the bottom, there are buttons for '< Previous', 'Next : Disks >', and 'Review + create'.

Choose the virtual machine size with at least 2 vCPUs. Select the preferred authentication method—SSH or password (password authentication is used here with a configured username and password). Allow inbound access on the SSH port to connect to the server, and then navigate

Create a virtual machine

OS disk

OS disk size: Image default (30 GiB)

OS disk type: Standard SSD (locally-redundant storage)

Key management: Platform-managed key

Data disks for GANESH-WEBSERVER

Next: Networking > Review + create

Choose the required disk type (SSD is selected here) and click **Next** to proceed.

Create a virtual machine

Virtual network: VNET-GANESH-WEB

Subnet: WEB-SNET (10.1.1.0/24)

Public IP: (new) GANESH-WEBSERVER-ip

NIC network security group: Basic

Public inbound ports: Allow selected ports

Select inbound ports: SSH (22)

< Previous | Next: Management > | Review + create

Select the VNet named VNET-GANESH-WEB (created for the web server), choose the appropriate **subnet range**, then click **Next** to proceed to the **Advanced settings** page.

The screenshot shows the 'Create a virtual machine' wizard in Microsoft Azure. The 'Advanced' tab is selected. In the 'Custom data and cloud init' section, there is a text input field containing the command `#!/bin/bash`. Below the input field are navigation buttons: '< Previous', 'Next : Tags >', and 'Review + create'.

write the commands as shown in custom data:

```
#!/bin/bash
```

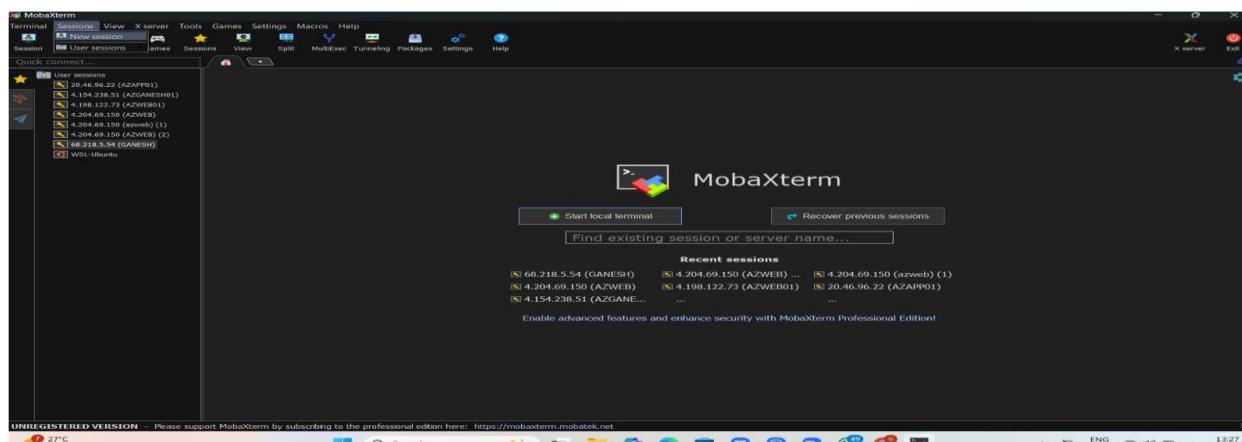
```
sudo su
```

```
apt update
```

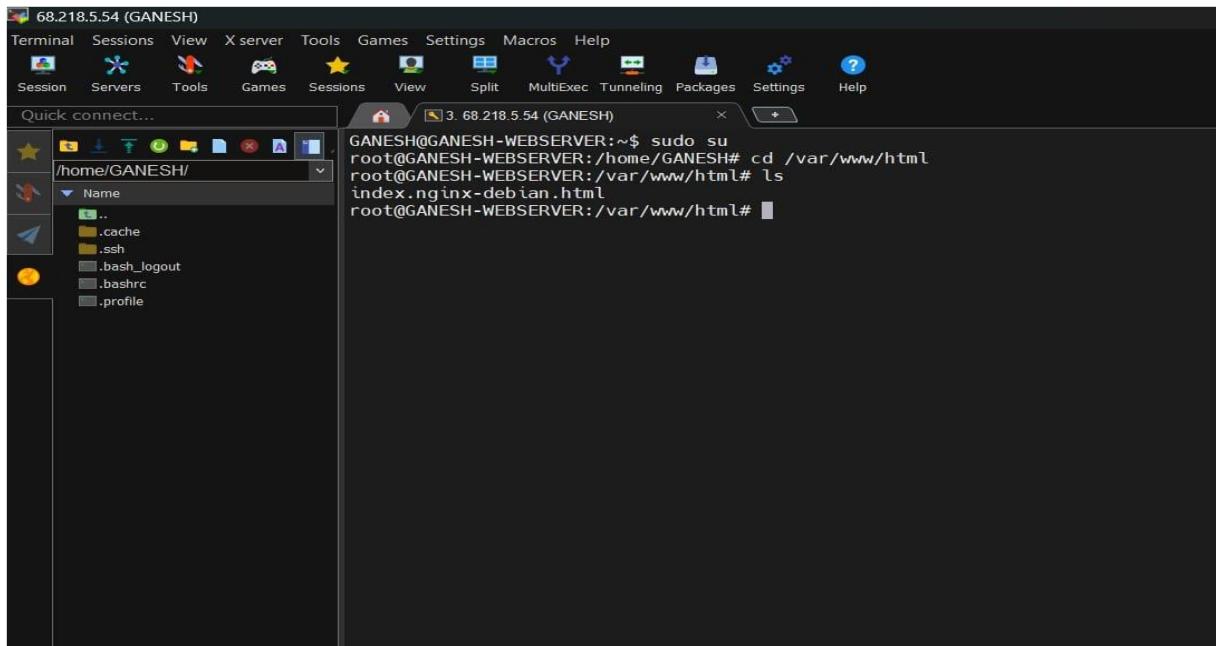
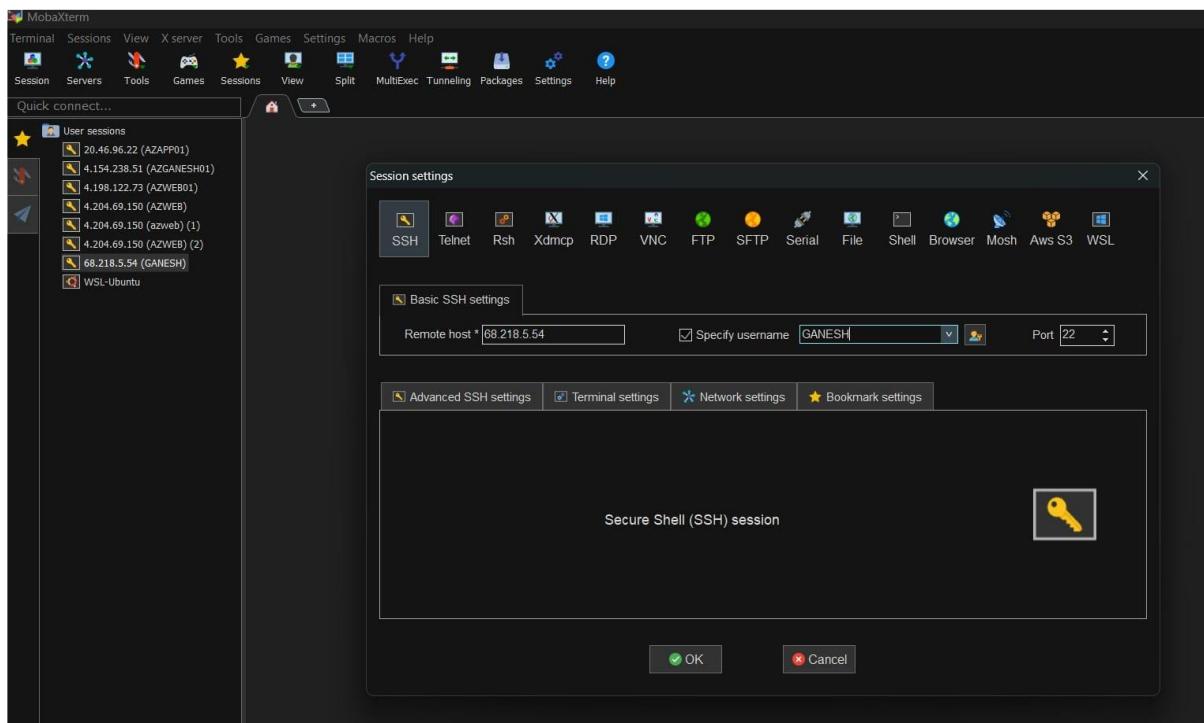
```
apt install nginx -y
```

This will automatically install nginx server while creating the VM, and then click on review + create.

Now lets connect to the server and check whether the nginx server is installed or not. Here i used **Mobaxterm**, to connect to the Vm's using SSH and provide the **public ip** of the machine and enter password.

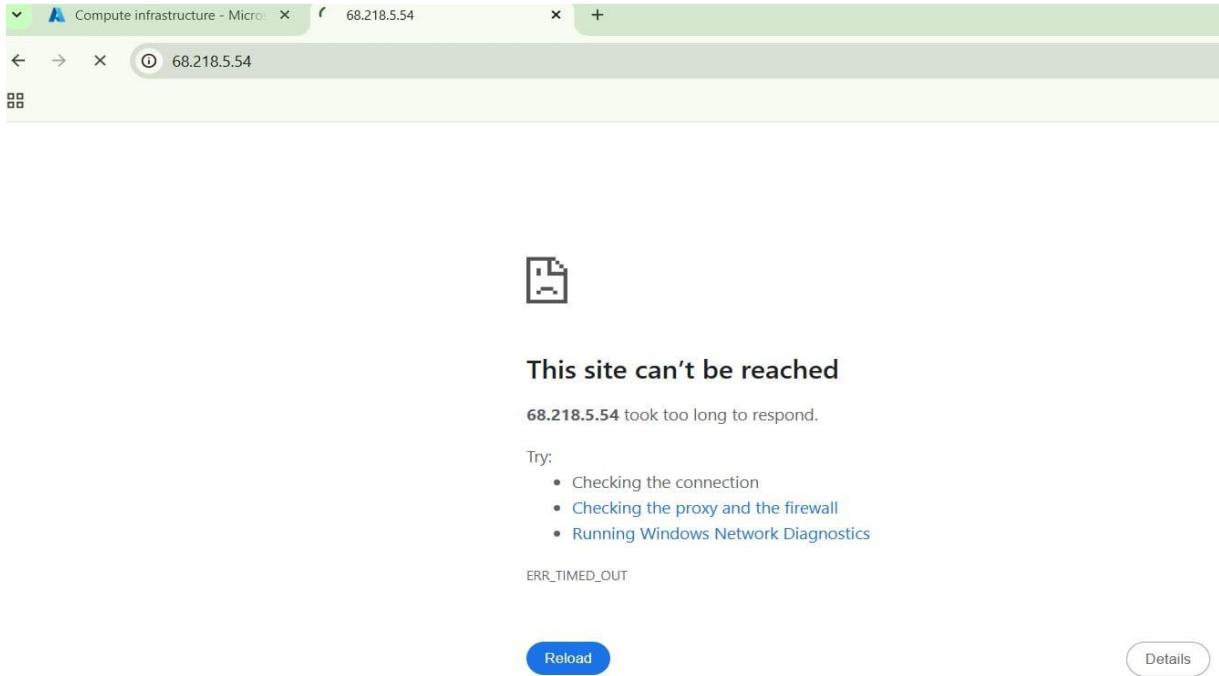


So, in mobaxtreme go to sessions → new session → session settings(click on ssh) and enter remote host and username.



Run the command `cd /var/www/html` to navigate to the web server's root directory, then use the `ls` command to verify whether Nginx is installed. As shown in the screenshot, the Nginx server files

are present. Next, access the server by entering its **public IP address** in a web browser such as Chrome.



Since the server is currently inaccessible, it is necessary to configure **inbound rules** to allow traffic on **HTTP port 80**.

To do this:

1. Navigate to the **web server** in the Azure portal.
2. Go to **Network Settings** and scroll down to the **Inbound and Outbound Rules** section.
3. Configure the inbound rule to allow **HTTP traffic on port 80** as required, then click **Add** to apply the rule.

Prio...	Name	Port	Protocol	Source
300	SSH	22	TCP	Any
65000	AllowVNetInBound	Any	Any	Virtu...
65001	AllowAzureLoadBalancerInB...	Any	Any	Azur...
65500	DenyAllInbound	Any	Any	Any

Prio...	Name	Port	Protocol	Source
300	SSH	22	TCP	Any
65000	AllowVNetInBound	Any	Any	Virtu...
65001	AllowAzureLoadBalancerInB...	Any	Any	Azur...
65500	DenyAllInbound	Any	Any	Any

In web server virtual network → go to networking → networking settings → NSG(create port rule) → enter the rule and then save.

The screenshot shows two overlapping dialog boxes from the Azure portal:

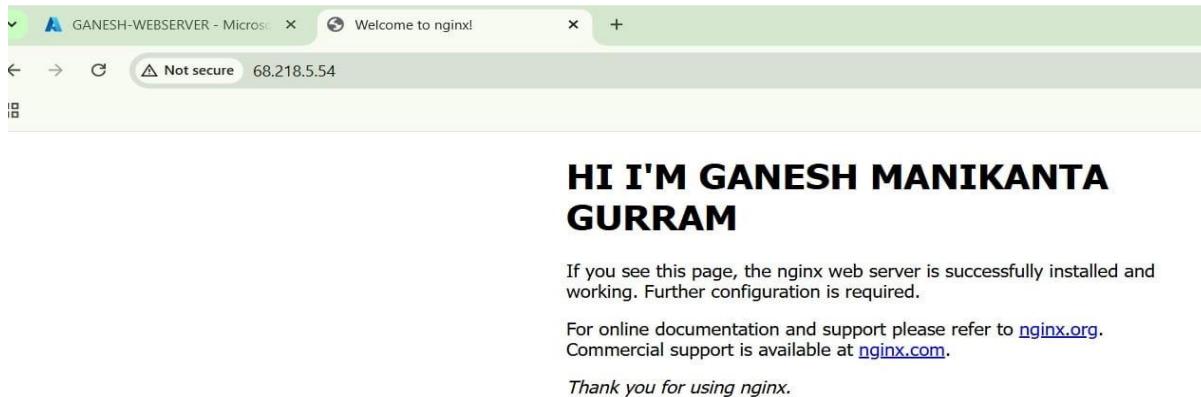
- Left Dialog (Main):** "Add inbound security rule" for "GANESH-WEBSERVER-nsg". It includes fields for Source (Any), Destination (Any), Service (HTTP), and Destination port ranges (80). It also has sections for Protocol (TCP selected) and Action (Allow selected).
- Right Dialog (Overlay):** "Add inbound security rule" for "GANESH-WEBSERVER-nsg". It shows the selected protocol (TCP), action (Allow), priority (310), name ("allow_http"), and a description field.

The screenshot shows the Azure portal interface with the following components:

- Left Side Navigation:** Shows the "Virtual machines" section under "Compute infrastructure | Virtual machines".
- Middle Panel:** Shows the "GANESH-WEBSERVER | Network settings" for the "GANESH-WEBSERVER" VM. It lists rules under "Inbound port rules (5)" and "Outbound port rules (3)".
- Right Panel:** Shows the detailed view of the "Inbound port rules (5)", listing the following rules:

Prio...	Name	Port	Protocol	Source
300	SSH	22	TCP	Any
310	allow_http	80	TCP	Any
65000	AllowVnetInBound	Any	Any	Virtu...
65001	AllowAzureLoadBalancerInB...	Any	Any	Azur...
65500	DenyAllInBound	Any	Any	Any

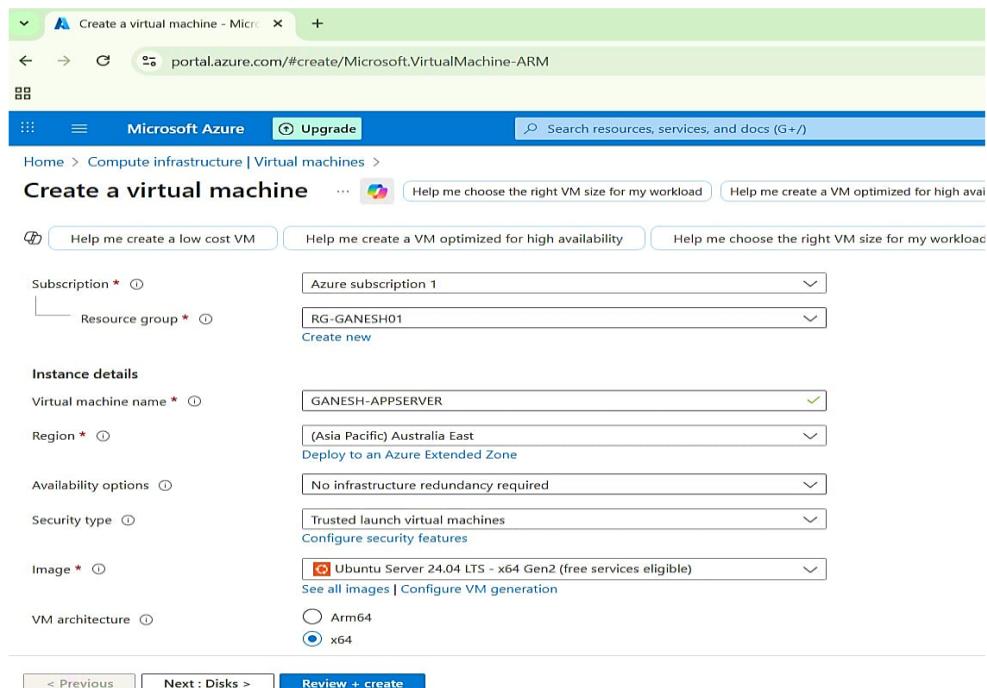
The Nginx server is now publicly accessible. To verify, open a web browser (e.g., Chrome) and enter the server's **public IP address** in the address bar. The default Nginx welcome page should appear, confirming successful access to the server.



The Nginx server is now accessible.

APP SERVER:

Next, we will create the App Server. Search for Virtual Machines in the Azure portal, click Create, and select the first option (Virtual Machine). The creation process is similar to that of the Web Server VM, The region and VNet are also same for the App-server, region(RG-GANESH-01),virtual network(VNET-GANESH-WEB).



Create a virtual machine

Administrator account

Authentication type: Password

Username *: GANESH-APPSERVER

Password *:

Confirm password *:

Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports *: None

Networking

Virtual network *: VNET-GANESH-WEB

Subnet *: APP-SNET (10.1.2.0/24)

Public IP: (new) GANESH-APPSERVER-ip

Public inbound ports *: None

< Previous | Next : Disks > | Review + create

At this stage, all unnecessary ports should be **denied**. Specific inbound rules will be added later as required to allow controlled access. This ensures that the server remains secure during the initial setup.

Create a virtual machine

Network interface

When creating a virtual machine, a network interface will be created for you.

Virtual network *: VNET-GANESH-WEB

Subnet *: APP-SNET (10.1.2.0/24)

Public IP: (new) GANESH-APPSERVER-ip

NIC network security group: Basic

Public inbound ports *: None

< Previous | Next : Management > | Review + create

Select the **VNET-GANESH-WEB** Virtual Network for this server, choose the **APP-SNET(10.1.2.0/24)** subnet, and then click **Review + Create** to proceed with the VM deployment.

Here also I'm selecting my disk as standard sdd. Because it is enough for me.

The screenshot shows the 'Create a virtual machine' wizard on the Microsoft Azure portal. The current step is 'Create a virtual machine'. The 'OS disk' section is selected, showing the following configuration:

- OS disk size: Image default (30 GiB)
- OS disk type: Standard SSD (locally-redundant storage) (selected)
- Key management: Platform-managed key
- Enable Ultra Disk compatibility: Ultra disk is supported in Availability Zone(s) 1,2,3 for the selected VM size Standard_B2s_v2.

Below this, under 'Data disks for GANESH-APPSERVER', it says: 'You can add and configure additional data disks for your virtual machine or attach existing disks. This VM also comes with a temporary disk.'

At the bottom, there are navigation buttons: '< Previous', 'Next : Networking >', and 'Review + create'.

The screenshot shows the 'Add inbound security rule' dialog box for a network security group (NSG) named 'GANESH-APPSERVER-nsg'. The configuration is as follows:

- Source: IP Addresses
- Source IP addresses/CIDR ranges: 10.1.1.4
- Source port ranges: *
- Destination: IP Addresses
- Destination IP addresses/CIDR ranges: 10.1.2.4
- Service: Custom
- Destination port ranges: 8080
- Protocol: Any (selected)

At the bottom, there are 'Add' and 'Cancel' buttons, and a 'Give feedback' link.

DB SERVER:

Next, create the **Database Server (DB Server)** VM following the same steps as the App Server. Select the **VNET-GANESH-DB** and assign it to a **DB-SNET(subnet), 10.2.1.0/24**.

Here we are choosing the another VNET rather than to WEB-SERVER AND APP-SERVER because in our free trial we are not able to create multiple(3-machines)machines. So, we choose another virtual network in that region and then we create our machine and later we do virtual network peering to connect.

Virtual machine name * (Required)
Region * (Required)
Availability options (Required)
Security type (Required)
Image * (Required)

< Previous | Next : Disks > | Review + create

Authentication type (Required)
Username * (Required)
Password * (Required)
Confirm password * (Required)

Inbound port rules
Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports * (Required)
Select inbound ports

< Previous | Next : Disks > | Review + create

The screenshot shows the Microsoft Azure portal with the URL portal.azure.com/#create/Microsoft.VirtualMachine-ARM. The page title is "Create a virtual machine". The main section is titled "OS disk". Under "OS disk size", "Image default (30 GiB)" is selected. Under "OS disk type", "Standard SSD (locally-redundant storage)" is selected, with a note: "The selected VM size supports premium disks. We recommend Premium SSD for high IOPS workloads. Virtual machines with Premium SSD disks qualify for the 99.9% connectivity SLA.". There are checkboxes for "Delete with VM" (checked) and "Key management" (Platform-managed key). A note at the bottom says: "Data disks for GANESH-DBSERVER You can add and configure additional data disks for your virtual machine or attach existing disks. This VM also comes with a temporary disk." Navigation buttons at the bottom include "< Previous", "Next : Networking >", and "Review + create".

The screenshot shows the Microsoft Azure portal with the URL portal.azure.com/#create/Microsoft.VirtualMachine-ARM. The page title is "Create a virtual machine". The main section is titled "Network interface". Configuration includes: "Virtual network" set to "VNET-GANESH-DB" (Create new); "Subnet" set to "DB-SNET (10.2.1.0/24)" (Manage subnet configuration); "Public IP" set to "(new) GANESH-DBSERVER-ip" (Create new), with a note: "Public IP addresses have a nominal charge. Estimate price"; "NIC network security group" set to "Basic"; and "Public inbound ports" set to "Allow selected ports". Navigation buttons at the bottom include "< Previous", "Next : Management >", and "Review + create".

Here we done creating our machine(GANESH-DBSERVER)

PEERING :

The screenshot shows two side-by-side Azure blades. The left blade is titled 'Network foundation | Virtual networks' and shows a list of virtual networks: 'VNET-GANESH-DB' and 'VNET-GANESH-WEB'. The right blade is titled 'VNET-GANESH-WEB | Peering' and shows a list of peering options: 'Resource visualizer', 'Settings', 'Address space', 'Connected devices', 'Subnets', 'Bastion', 'DDoS protection', 'Firewall', 'Microsoft Defender for Cloud', 'Network manager', 'DNS', 'Peerings' (which is selected), 'Service endpoints', and 'Private endpoints'. Both blades have a search bar at the top and a sidebar with various navigation links.

Setting up Peering:

Since the servers are deployed across **two different VNets**, it is essential to enable communication between all three servers. To achieve this, we configure **Network Peering**.

Network Peering:

Network Peering acts as a **bridge between two virtual networks**, allowing resources in different VNets to communicate securely and efficiently as if they were part of the same network.

The screenshot shows the 'Add peering' configuration page. It includes sections for 'Remote virtual network summary' and 'Remote virtual network peering settings'. In the 'Remote virtual network summary' section, fields include 'Peer link name' (set to 'peer-vnetweb-vnetdb'), 'I know my resource ID' (unchecked), 'Subscription' (set to 'Azure subscription 1'), and 'Virtual network' (set to 'VNET-GANESH-DB (RG-GANESH01)'). In the 'Remote virtual network peering settings' section, there are two checkboxes: 'Allow 'VNET-GANESH-DB' to access 'VNET-GANESH-WEB'' (checked) and 'Allow 'VNET-GANESH-DB' to receive forwarded traffic from 'VNET-GANESH-WEB'' (unchecked). At the bottom are 'Add' and 'Cancel' buttons.

Local virtual network summary

Peering link name *

Local virtual network peering settings

Allow 'VNET-GANESH-WEB' to access 'VNET-GANESH-DB'

Allow 'VNET-GANESH-WEB' to receive forwarded traffic from 'VNET-GANESH-DB'

Allow gateway or route server in 'VNET-GANESH-WEB' to forward traffic to 'VNET-GANESH-DB'

Enable 'VNET-GANESH-WEB' to use 'VNET-GANESH-DB's remote gateway or route server

Add **Cancel**

To establish communication between the servers deployed across two different VNets, we configure VNet peering. For example, navigate to ganesh-web-vnet and select the Peering option from the left-hand menu. Provide a name for the peering connection and select the destination VNet, vnet-ganesh-db, then click Add to create the peering. Once the peering is established, connectivity between the servers can be verified using the ping command, ensuring that all servers can communicate across the VNet's.

```

GANESH@GANESH-WEBSERVER:~$ sudo su
root@GANESH-WEBSERVER:/home/GANESH# cd /var/www/html
root@GANESH-WEBSERVER:/var/www/html# ls
index.nginx-debian.html
root@GANESH-WEBSERVER:/var/www/html# ls
index.nginx-debian.html
root@GANESH-WEBSERVER:/var/www/html# vi ^C
root@GANESH-WEBSERVER:/var/www/html# vi index.nginx-debian.html
root@GANESH-WEBSERVER:/var/www/html# ls
index.nginx-debian.html
root@GANESH-WEBSERVER:/var/www/html# ping 10.2.1.4
PING 10.2.1.4 (10.2.1.4) 56(84) bytes of data.
64 bytes from 10.2.1.4: icmp_seq=1 ttl=64 time=15.3 ms
64 bytes from 10.2.1.4: icmp_seq=2 ttl=64 time=4.82 ms
64 bytes from 10.2.1.4: icmp_seq=3 ttl=64 time=4.84 ms
64 bytes from 10.2.1.4: icmp_seq=4 ttl=64 time=4.83 ms
64 bytes from 10.2.1.4: icmp_seq=5 ttl=64 time=5.09 ms
64 bytes from 10.2.1.4: icmp_seq=6 ttl=64 time=4.82 ms
64 bytes from 10.2.1.4: icmp_seq=7 ttl=64 time=4.83 ms
64 bytes from 10.2.1.4: icmp_seq=8 ttl=64 time=4.89 ms
64 bytes from 10.2.1.4: icmp_seq=9 ttl=64 time=4.86 ms
64 bytes from 10.2.1.4: icmp_seq=10 ttl=64 time=4.75 ms

```

Adding inbound and outbound rules:

Inbound and outbound rules are configured based on the role of each server. The Web Server (Frontend) is intended to be publicly accessible, as it hosts the frontend web page, and therefore the required inbound rules have already been added. The App Server and db server (Backend), should remain private and must not be accessible from the public internet. It should only communicate based on the rules. This can be done as follows:

The screenshot shows the Microsoft Azure portal interface. The left sidebar is under 'Compute infrastructure | Virtual machines' and has 'Virtual machines' selected. The main pane shows 'GANESH-DBSERVER' under 'Network settings'. The 'Rules' section displays three inbound port rules:

Prio...	Name	Port	Protocol	Source
65000	AllowVnetInBound	Any	Any	Virt
65001	AllowAzureLoadBalancerInBound	Any	Any	Azur
65500	DenyAllInBound	Any	Any	Any

The screenshot shows the configuration dialog for a Network Security Group (NSG). The 'Source' section is set to 'IP Addresses' with the value '10.1.1.4'. The 'Destination' section is set to 'IP Addresses' with the value '10.2.1.4'. The 'Service' section is set to 'Custom'. The 'Protocol' section is set to 'Any'. The 'Protocol' dropdown has a radio button next to 'Any'.

deny_all
GANESH-DBSERVER-nsg

Protocol: Any
 TCP
 UDP
 ICMPv4
 ICMPv6

Action: Deny
 Allow
 Deny

Priority * ⓘ

Name

Description

Save **Cancel** **Give feedback**

Since the **Database Server** communicates only with the **App Server**, inbound and outbound rules must be configured accordingly. The database handles requests and responses from the App Server; therefore, access to the **MySQL port (3306)** is allowed by specifying the **App Server's private IP address as the source** and the **Database Server's private IP address as the destination**. This configuration ensures that only the App Server can communicate with the Database Server over port 3306.

allow_ap_db
GANESH-DBSERVER-nsg

Net
Source ⓘ IP Addresses
Source IP addresses/CIDR ranges * ⓘ 10.1.2.4

Source port ranges * ⓘ *

Destination ⓘ IP Addresses
Destination IP addresses/CIDR ranges * ⓘ 10.2.1.4

Service ⓘ Custom
Destination port ranges * ⓘ 3306

Protocol: Any
 TCP
 UDP
 ICMPv4
 ICMPv6

Action: Allow
 Deny

Priority * ⓘ 105

Name

Description

Save **Cancel** **Give feedback**

allow_ap_db
GANESH-DBSERVER-nsg

Net
Protocol: Any
 TCP
 UDP
 ICMPv4
 ICMPv6

Action: Allow
 Deny

Priority * ⓘ 105

Name

Description

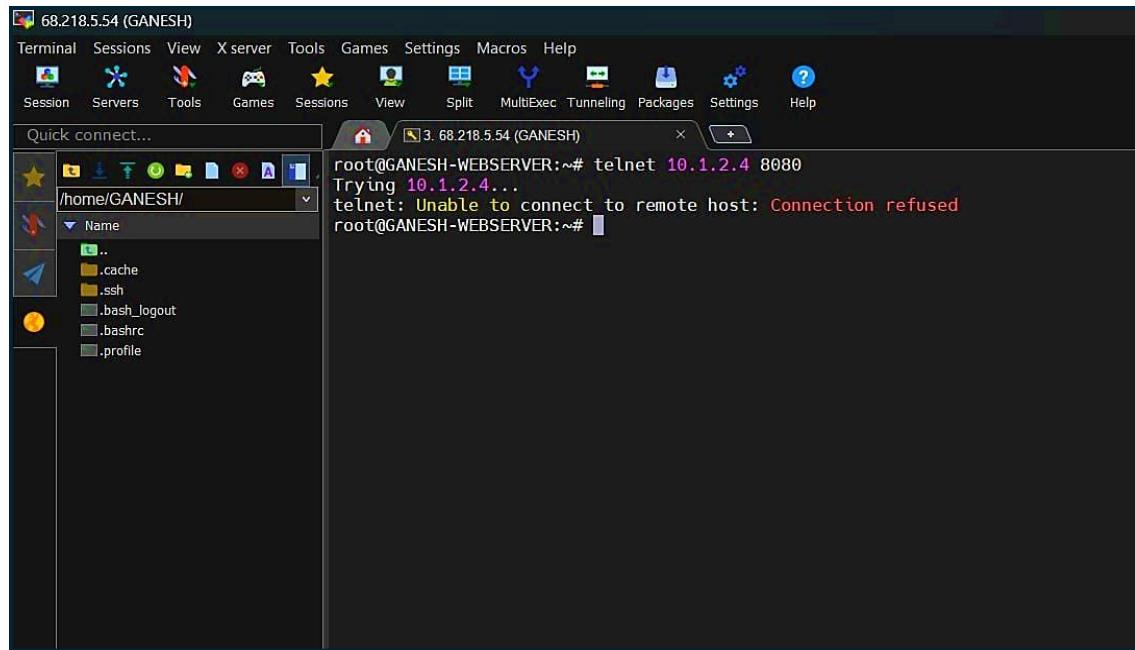
Save **Cancel** **Give feedback**

Installing and Configuring Tomcat on the App Server:

With the setup of all servers completed, the next step is to verify connectivity between the Web Server and the App Server. This can be tested using the command:

```
telnet 10.1.2.4 8080
```

At this stage, the connection attempt will fail because Tomcat has not yet been installed on the App Server. This confirms that the application service is not currently running on port 8080 and must be installed and configured before communication can be established.



After verifying connectivity between the Web Server and the App Server, the next step is to install and configure **Apache Tomcat** on the App Server. The App Server can be accessed directly from the Web Server using the SSH command:

```
ssh username & ip address(private ip) (ssh GANESH-APPSERVER@10.1.2.4)
```

Once connected, switch to the **root user** and execute the following commands to install and configure Tomcat:

1. Update the package list: `apt update`
2. Install Java (OpenJDK 11): `apt install openjdk-11-jdk -y`
3. Create a Tomcat user: `useradd -m -U -d /opt/tomcat -s /bin/false tomcat`

This command creates a dedicated Tomcat user for security purposes.

4. Download Apache Tomcat: cd /tmp
wget https://archive.apache.org/dist/tomcat/tomcat10/v10.1.19/bin/apache-tomcat-10.1.19.tar.gz
5. Create the Tomcat installation directory: mkdir -p /opt/tomcat
6. Extract Tomcat files: tar -xzf /tmp/apache-tomcat-10.1.19.tar.gz -C /opt/tomcat --strip-components=1

This extracts the downloaded Tomcat files into the newly created /opt/tomcat directory.

```
root@GANESH-WEBSERVER:~# ssh GANESH-APPSERVER@10.1.2.4
GANESH-APPSERVER@10.1.2.4's password:
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1017-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Fri Dec 19 09:21:10 UTC 2025

System load:  0.08      Processes:           133
Usage of /:   5.7% of 28.02GB  Users logged in:     0
Memory usage: 36%          IPv4 address for eth0: 10.1.2.4
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

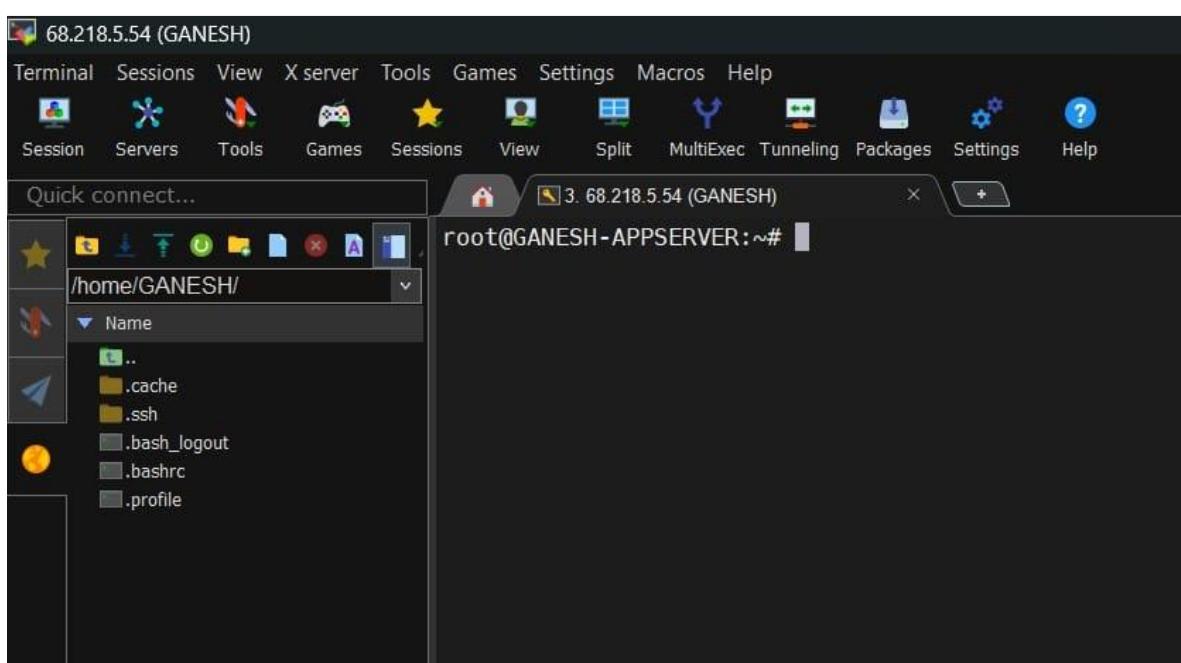
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

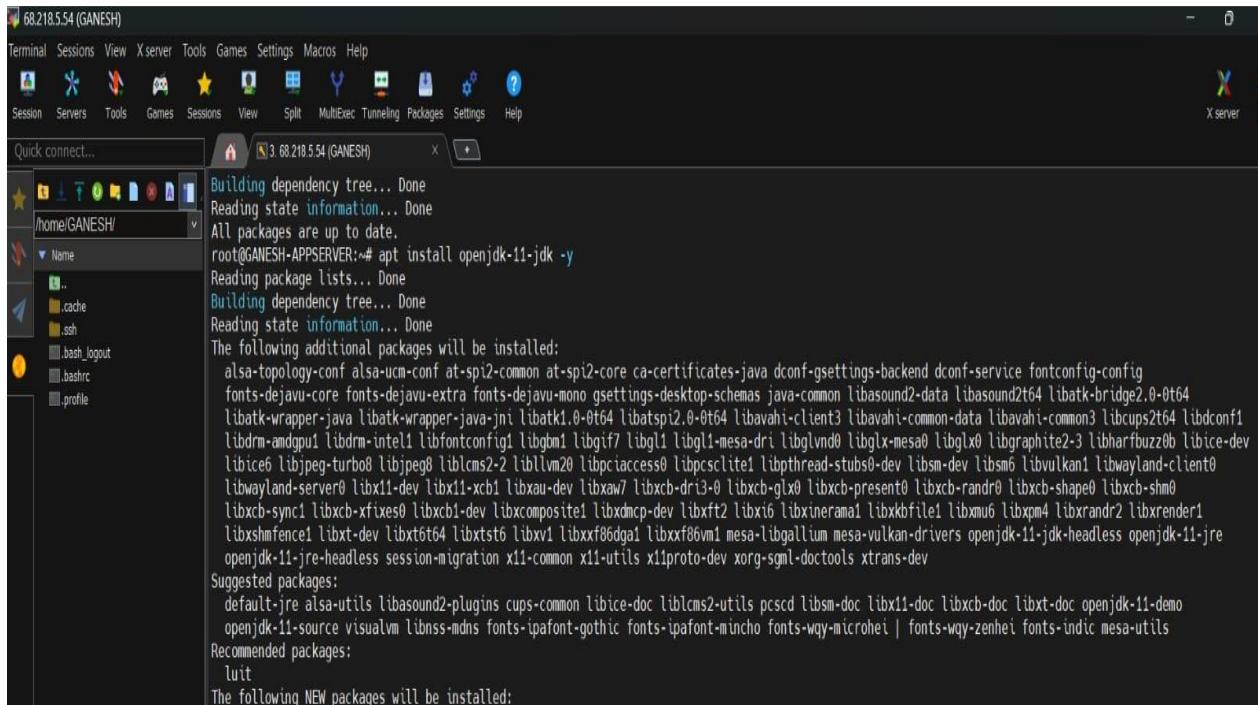
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

GANESH-APPSERVER@GANESH-APPSERVER:~$
```



Now we are in app server, and here we need to install tomcat.



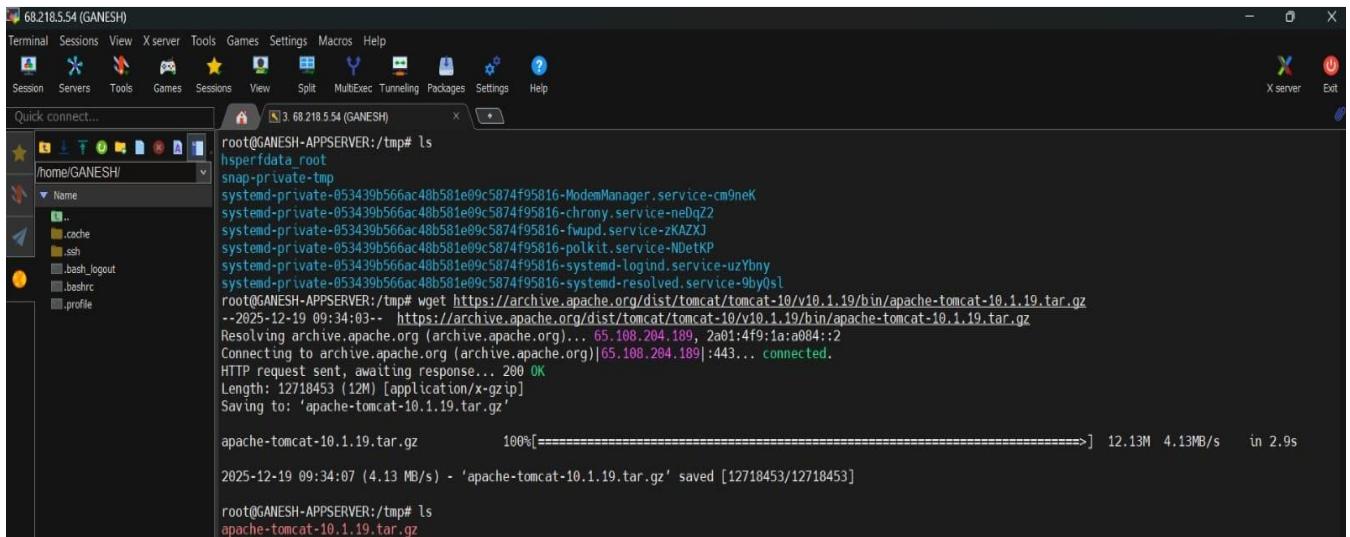
The screenshot shows a terminal window with the following commands and output:

```
root@GANESH-APPSERVER:/tmp# ls
hsperfdata_root
snap-private-tmp
systemd-private-053439b566ac48b581e09c5874f95816-ModemManager.service-cm9neK
systemd-private-053439b566ac48b581e09c5874f95816-chrony.service-neDqZ2
systemd-private-053439b566ac48b581e09c5874f95816-fwupd.service-zKAZXJ
systemd-private-053439b566ac48b581e09c5874f95816-polkit.service-NDetKP
systemd-private-053439b566ac48b581e09c5874f95816-systemd-logind.service-uZYbny
systemd-private-053439b566ac48b581e09c5874f95816-systemd-resolved.service-9byQsl
root@GANESH-APPSERVER:/tmp# wget https://archive.apache.org/dist/tomcat/tomcat-10/v10.1.19/bin/apache-tomcat-10.1.19.tar.gz
--2025-12-19 09:34:03-- https://archive.apache.org/dist/tomcat/tomcat-10/v10.1.19/bin/apache-tomcat-10.1.19.tar.gz
Resolving archive.apache.org (archive.apache.org)... 65.108.204.189, 2a01:4f9:1a:a084::2
Connecting to archive.apache.org (archive.apache.org)|65.108.204.189|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12718453 (12M) [application/x-gzip]
Saving to: 'apache-tomcat-10.1.19.tar.gz'

apache-tomcat-10.1.19.tar.gz          100%[=====] 12.13M  4.13MB/s   in 2.9s

2025-12-19 09:34:07 (4.13 MB/s) - 'apache-tomcat-10.1.19.tar.gz' saved [12718453/12718453]
```

7. ls /opt/tomcat - must see (bin conf lib logs webapps work temp)



```
root@GANESH-APPSERVER:~# cd /tmp
root@GANESH-APPSERVER:/tmp# ls
apache-tomcat-10.1.19.tar.gz
hisperfdata_root
snap-private-tmp
systemd-private-053439b566ac48b581e09c5874f95816-ModemManager.service-cm9neK
systemd-private-053439b566ac48b581e09c5874f95816-chrony.service-neDqZ2
systemd-private-053439b566ac48b581e09c5874f95816-fwupd.service-zKAZXJ
systemd-private-053439b566ac48b581e09c5874f95816-polkit.service-NDetKP
systemd-private-053439b566ac48b581e09c5874f95816-systemd-logind.service-uzYbny
systemd-private-053439b566ac48b581e09c5874f95816-systemd-resolved.service-9byQsl
root@GANESH-APPSERVER:/tmp# wget https://archive.apache.org/dist/tomcat/tomcat-10/v10.1.19/bin/apache-tomcat-10.1.19.tar.gz
--2025-12-19 09:34:03-- https://archive.apache.org/dist/tomcat/tomcat-10/v10.1.19/bin/apache-tomcat-10.1.19.tar.gz
Resolving archive.apache.org (archive.apache.org)... 65.108.204.189, 2a01:4f9:1a:a084::2
Connecting to archive.apache.org (archive.apache.org)|65.108.204.189|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12718453 (12M) [application/x-gzip]
Saving to: 'apache-tomcat-10.1.19.tar.gz'

2025-12-19 09:34:07 (4.13 MB/s) - 'apache-tomcat-10.1.19.tar.gz' saved [12718453/12718453]

root@GANESH-APPSERVER:/tmp# ls
apache-tomcat-10.1.19.tar.gz
```

```
root@GANESH-APPSERVER:~# cd /tmp
root@GANESH-APPSERVER:/tmp# ls
apache-tomcat-10.1.19.tar.gz
hisperfdata_root
snap-private-tmp
systemd-private-053439b566ac48b581e09c5874f95816-ModemManager.service-cm9neK
systemd-private-053439b566ac48b581e09c5874f95816-chrony.service-neDqZ2
systemd-private-053439b566ac48b581e09c5874f95816-fwupd.service-zKAZXJ
systemd-private-053439b566ac48b581e09c5874f95816-polkit.service-NDetKP
systemd-private-053439b566ac48b581e09c5874f95816-systemd-logind.service-uzYbny
systemd-private-053439b566ac48b581e09c5874f95816-systemd-resolved.service-9byQsl
root@GANESH-APPSERVER:/tmp# mkdir -p /opt/tomcat
root@GANESH-APPSERVER:/tmp# tar -xzf /tmp/apache-tomcat-10.1.19.tar.gz -C /opt/tomcat --strip-components=1
tar: --: Not found in archive
tar: strip-components=1: Not found in archive
tar: Exiting with failure status due to previous errors
root@GANESH-APPSERVER:/tmp# ^C
root@GANESH-APPSERVER:/tmp# tar -xzf /tmp/apache-tomcat-10.1.19.tar.gz -C /opt/tomcat --strip-components=1
root@GANESH-APPSERVER:/tmp# ls /opt/tomcat
BUILDING.txt CONTRIBUTING.md LICENSE NOTICE README.md RELEASE-NOTES RUNNING.txt bin conf lib logs temp webapps work
root@GANESH-APPSERVER:/tmp#
```

8. chown -R tomcat:tomcat /opt/tomcat - change ownership

9. chmod +x /opt/tomcat/bin/*.sh - change permission

10. nano /etc/systemd/system/tomcat.service - to create systemed file

[Unit]

Description=Apache Tomcat

After=network.target

[Service]

Type=forking

User=tomcat

Group=tomcat

Environment=JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

Environment=CATALINA_HOME=/opt/tomcat

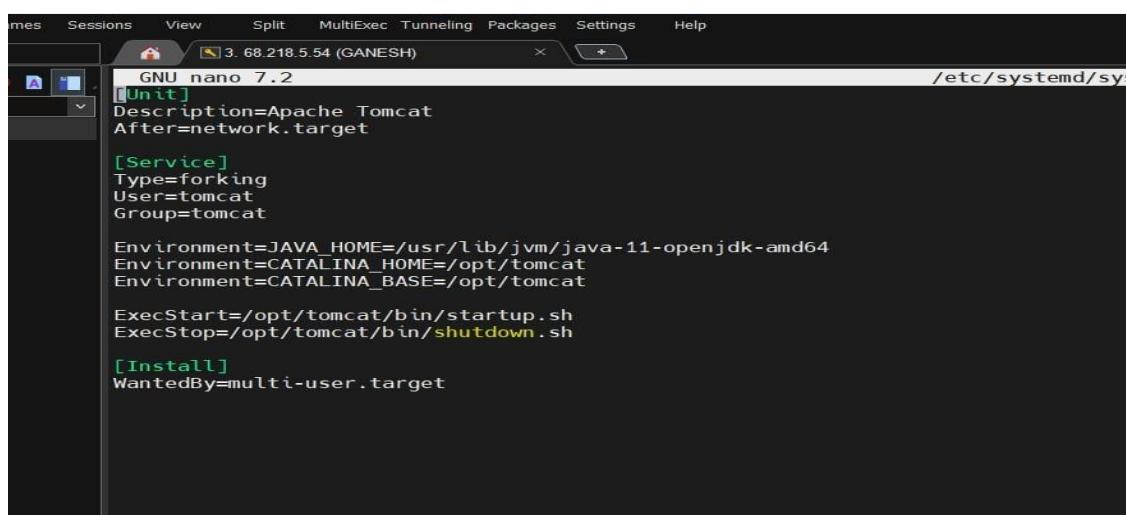
Environment=CATALINA_BASE=/opt/tomcat

ExecStart=/opt/tomcat/bin/startup.sh

ExecStop=/opt/tomcat/bin/shutdown.sh

[Install]

WantedBy=multi-user.target



The screenshot shows a terminal window titled "3 68.218.5.54 (GANESH)". The content of the terminal is a systemd service unit file for Apache Tomcat. It includes sections for [Unit], [Service], Environment variables, ExecStart, ExecStop, [Install], and WantedBy.

```
[Unit]
Description=Apache Tomcat
After=network.target

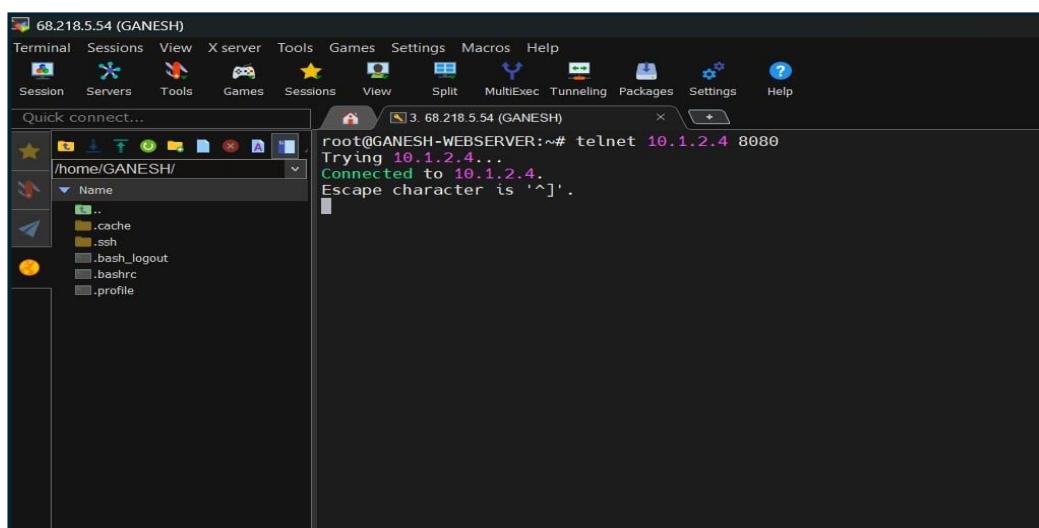
[Service]
Type=forking
User=tomcat
Group=tomcat

Environment=JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh

[Install]
WantedBy=multi-user.target
```

- systemctl daemon-reload
- systemctl start tomcat
- systemctl enable tomcat
- ss -tulnp | grep 8080 — to check 8080 port is open or not
- Now go to webserver root address and retype the telnet command: telnet 10.1.2.4 8080



The screenshot shows a terminal window titled "3 68.218.5.54 (GANESH)". The terminal output shows a root shell connected via telnet to the IP address 10.1.2.4 on port 8080. The user has typed "Connected to 10.1.2.4." and is awaiting further input.

```
root@GANESH-WEBSERVER:~# telnet 10.1.2.4 8080
Trying 10.1.2.4...
Connected to 10.1.2.4.
Escape character is '^]'.
```

Installing and Configuring MySql database in DB Server:

Now we need to install MySql database in DB Server and try to connect from appserver to dbserver using: telnet 10.2.1.4

Now lets connect to app server from web server using SSH command and then from app server we connect to DB server using SSH command:

```
ssh ganesh@10.2.1.4
```

We cannot directly connect to dbserver from webserver because we denied all the ports communication of dbserver with webserver.

Execute the following commands to install and configure MySql database:

- apt update
- apt install mysql-server -y
- systemctl start mysql
- systemctl enable mysql
- systemctl status mysql

```
done!
update-alternatives: using /var/lib/mecab/dic/ipadic-utf8 to provide /var/lib/mecab/dic/debian (mecab-dictionary) in auto mode
Setting up libhtml-parser-perl:amd64 (3.81-1build3) ...
Setting up libhttp-message-perl (6.45-1ubuntu1) ...
Setting up mysql-server (8.0.44-0ubuntu0.24.04.2) ...
Setting up libcgi-pm-perl (4.63-1) ...
Setting up libhtml-template-perl (2.97-2) ...
Setting up libcgi-fast-perl (1:2.17-1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.6) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@GANESH-DBSERVER:~#
```

```

NO VR guests are running outdated hypervisor (qemu) that lies on this host.
root@GANESH-DBSERVER:~# systemctl start mysql
root@GANESH-DBSERVER:~# systemctl enable mysql
Synchronizing state of mysql.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable mysql
root@GANESH-DBSERVER:~# systemctl status mysql
● mysql.service - MySQL Community Server
   Loaded: loaded (/usr/lib/systemd/system/mysql.service; enabled; preset: enabled)
   Active: active (running) since Fri 2025-12-19 10:36:52 UTC; 1min 28s ago
     Main PID: 2997 (mysqld)
       Status: "Server is operational"
        Tasks: 37 (limit: 1037)
      Memory: 354.8M (peak: 378.9M)
        CPU: 1.096s
      CGroup: /system.slice/mysql.service
              └─2997 /usr/sbin/mysqld

Dec 19 10:36:51 GANESH-DBSERVER systemd[1]: Starting mysql.service - MySQL Community Server...
Dec 19 10:36:52 GANESH-DBSERVER systemd[1]: Started mysql.service - MySQL Community Server.
root@GANESH-DBSERVER:~# 
```

- mysql_secure_installation - and press yes for all
- ss -tulnp | grep 3306:

It is used to check whether MySQL (port 3306) is running and listening on the server.

Now try to connect db server from app server, but the connection will be refused.Because in above grep command the ip is 127.0.0.1:3306, so it will connect to only that ip, so we need to chane it to 0.0.0.0:3306 to connect using any ip address. Go to db server again and execute.

- nano /etc/mysql/mysql.conf.d/mysqld.cnfex:
- execute this cmd you will get THIS image

```

# The MySQL database server configuration file.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.
#
# For explanations see
# http://dev.mysql.com/doc/mysql/en/server-system-variables.html
#
# Here are entries for some specific programs
# The following values assume you have at least 32M ram

[mysqld]
#
# * Basic Settings
#
user          = mysql
# pid-file     = /var/run/mysqld/mysqld.pid
# socket      = /var/run/mysqld/mysqld.sock
# port         = 3306
# datadir      = /var/lib/mysql

# If MySQL is running as a replication slave, this should be
# changed. Ref https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tmpdir
# tmpdir        = /tmp
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address    = 0.0.0.0
mysqlx-bind-address = 127.0.0.1
#
# * Fine Tuning
#
key_buffer_size     = 16M

```

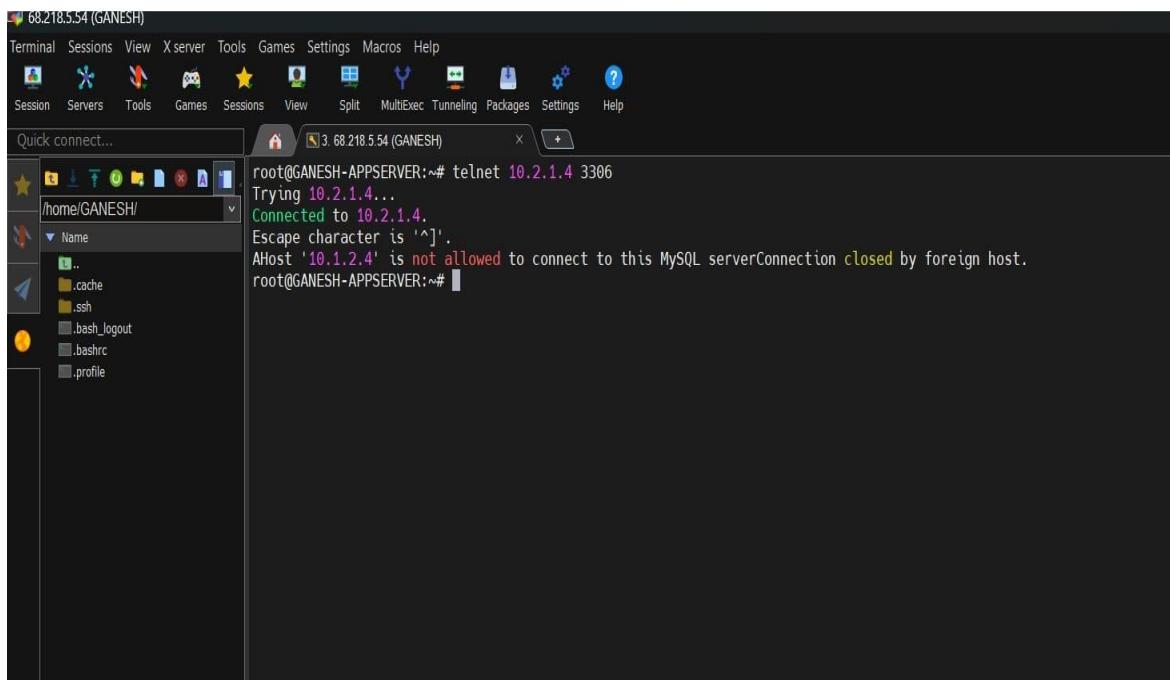
Here change the bind-address from 127.0.0.1 to 0.0.0.0

- again execute: ss -tulnp | grep 3306

```
All done!
root@GANESH-DBSERVER:~# ss -tulnp | grep 3306
tcp  LISTEN  0      151      127.0.0.1:3306    0.0.0.0:*      users:(("mysqld",pid=2997,fd=25))
tcp  LISTEN  0      70       127.0.0.1:33060   0.0.0.0:*      users:(("mysqld",pid=2997,fd=21))
root@GANESH-DBSERVER:~# nano /etc/mysql/mysql.conf.d/mysqld.cnf
root@GANESH-DBSERVER:~# systemctl restart mysql
root@GANESH-DBSERVER:~# ss -tulnp | grep 3306
tcp  LISTEN  0      151      0.0.0.0:3306    0.0.0.0:*      users:(("mysqld",pid=3331,fd=23))
tcp  LISTEN  0      70       127.0.0.1:33060   0.0.0.0:*      users:(("mysqld",pid=3331,fd=21))
root@GANESH-DBSERVER:~#
```

127.0.0.1:3306 is changed to 0.0.0.0:3306(Now, we connect from any ip).

Now try to connect to database server from appserver,



So, the connection is established between appserver and dbserver.

CONCLUSION :

This project shows how a three-tier architecture is implemented on Microsoft Azure. The application is divided into three layers: Web Server, Application Server, and Database Server. This separation helps in better performance, easy management, and improved security.

Azure Virtual Networks (VNets), subnets, and Network Security Groups (NSGs) are used to allow secure communication between the servers and to block unauthorized access. The Web Server is used to handle user requests, the Application Server runs the business logic using Apache Tomcat, and the Database Server stores data securely using MySQL in a private subnet.

Overall, this project represents a real-world cloud architecture and provides a basic understanding of how multi-tier applications are deployed on Azure.