### ****Session 4: Networking Essentials for DevOps****

### ****Learning Objectives****:

* Understand basic networking concepts relevant to DevOps.
* Identify different VirtualBox network modes.
* Use essential networking tools on the command line.

## ****Part 1: Networking Basics Overview****

### ****1. IP Address****

An **IP (Internet Protocol) address** uniquely identifies a device on a network.

#### IPv4:

* Format: xxx.xxx.xxx.xxx (e.g., 192.168.1.1)
* 32-bit address → approx. 4.3 billion addresses
* Example classes:
  + Class A: 1.0.0.0 – 126.255.255.255
  + Class B: 128.0.0.0 – 191.255.255.255
  + Class C: 192.0.0.0 – 223.255.255.255
* **Private ranges**:
  + 10.0.0.0/8
  + 172.16.0.0/12
  + 192.168.0.0/16

#### IPv6:

* Format: 8 groups of 4 hexadecimal digits, separated by colons (e.g., 2001:0db8:85a3::8a2e:0370:7334)
* 128-bit address → ≈ 3.4×10³⁸ possible addresses
* Built to replace IPv4 due to address exhaustion

**Why DevOps cares**: You’ll often configure services to bind to specific IPs, or whitelist IP ranges for access.

### ****2. Ports and Common Port Numbers****

A **port** is a logical channel through which services listen for requests. They range from **0 to 65535**.

#### Port Types:

* **Well-known ports** (0–1023): Used by standard services
  + 22: SSH (remote login)
  + 80: HTTP (web)
  + 443: HTTPS (secure web)
  + 3306: MySQL
  + 5432: PostgreSQL
* **Registered ports** (1024–49151): Used by vendors/applications
* **Dynamic/private ports** (49152–65535): Used temporarily (e.g., by browsers or APIs)

**Why DevOps cares**: You’ll open/close ports in firewalls, expose container ports, or debug services bound to wrong ports.

### ****3. DNS: Domain Name System****

DNS maps **human-readable names** to **IP addresses**.

* google.com → 142.250.182.46
* Works like a **phonebook for the internet**
* Tools: nslookup, dig

#### DNS Query Flow:

1. Browser → OS → Local DNS cache
2. If not found, → DNS Resolver (usually from ISP)
3. Resolver → Root → TLD (.com) → Authoritative server
4. IP address is returned → Browser connects

**Why DevOps cares**: You may host your own DNS, modify /etc/hosts, or troubleshoot resolution errors.

### ****4. NAT: Network Address Translation****

NAT allows **multiple devices to share a single public IP** by rewriting private IPs in outgoing packets.

* Common in home routers
* E.g., your device 192.168.1.5 goes out via public IP 116.203.100.25
* Port numbers help NAT map responses back to the correct private device

#### Types:

* **SNAT** (Source NAT): Change sender IP (e.g., private → public)
* **DNAT** (Destination NAT): Port forwarding, change destination IP (e.g., public\_ip:80 → local\_vm:8080)

**Why DevOps cares**: You configure NAT when exposing internal services or testing VM networking modes.

### ****5. Firewalls****

A **firewall** filters traffic based on rules.

* **Host-based**: Runs on your machine (e.g., ufw, firewalld)
* **Network-based**: Hardware or cloud firewall (e.g., AWS Security Groups)

#### What they filter:

* Source/destination IP
* Port numbers
* Protocol (TCP/UDP)

#### Basic Commands (UFW example):

sudo ufw enable

sudo ufw allow 22

sudo ufw deny 80

sudo ufw status

**Why DevOps cares**: A common cause of “my service doesn’t work” is a blocked port!

## ****Part 2: VirtualBox Network Modes****

VirtualBox allows you to simulate various network environments by configuring how your VM connects to other machines, the host, and the internet.

Understanding these modes is crucial when:

* Setting up local test environments
* Hosting services inside VMs
* Creating isolated virtual networks

### 1. ****NAT (Network Address Translation)****

🔹 **Default mode** for new VMs  
🔹 VM can **access the internet** through the host  
🔹 **Outside devices cannot access** the VM unless you use port forwarding

**Analogy**: Like your phone accessing the internet via a home Wi-Fi router. The outside world doesn’t know your phone’s private IP.

#### Use Cases:

* Browsing the web or downloading packages from inside VM
* Safe for general purpose use
* Great for initial setup, software installations

#### Limitations:

* No inbound access to the VM (e.g., can’t SSH into the VM from host)
* No direct communication between two NAT VMs unless configured with port forwarding

### 2. ****Bridged Networking****

🔹 VM acts like a **real machine** on the **same network** as your host  
🔹 Gets its **own IP from the LAN’s DHCP** (like your physical machine)  
🔹 Fully visible on the local network

**Analogy**: It’s as if your VM is connected to the LAN by its own Ethernet cable.

#### Use Cases:

* When you want to access the VM **from other devices on your network**
* Hosting local servers or test environments
* SSH/HTTP/FTP access from other physical devices

#### Notes:

* May not work well on Wi-Fi or restricted networks (e.g., public cafes, universities)
* Exposes VM to LAN traffic, so security is more important

### 3. ****Host-Only Networking****

🔹 VM can **only talk to the host**, **not to the internet**  
🔹 Creates a **private virtual network** between host and VM  
🔹 Host gets a virtual adapter (e.g., vboxnet0)

**Analogy**: Like connecting two laptops directly with a cable, isolated from others.

#### Use Cases:

* Host ↔ VM development/testing
* Secure, isolated environments
* SSH or file transfer between host and VM only

#### Notes:

* No external internet unless combined with NAT
* You need to configure IP addresses manually sometimes

### 4. ****Internal Network****

🔹 VM can talk to **other VMs in the same internal network name**  
🔹 **No internet**, **no host access**  
🔹 Perfect for **multi-VM simulations** and **lab setups**

**Analogy**: Like a private room where only selected VMs are allowed in.

#### Use Cases:

* Simulate a small isolated network (e.g., 2 VMs: app server + database)
* Test networking setups like DNS, DHCP, and firewall rules
* Teaching isolated networking concepts

#### Notes:

* All VMs must be on the same internal network name
* Needs manual IP configuration unless you run a DHCP server inside

### Summary Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mode** | **Internet Access** | **Host Access** | **VM ↔ VM** | **Use Case** |
| **NAT** | ✅ Yes | ❌ No (unless forwarded) | ❌ No (by default) | General use, package install |
| **Bridged** | ✅ Yes | ✅ Yes | ✅ Yes | Server on LAN, reachable from other PCs |
| **Host-Only** | ❌ No | ✅ Yes | ❌ No (unless configured) | Dev/test between host ↔ VM |
| **Internal** | ❌ No | ❌ No | ✅ Yes | Fully isolated VM-only networks |

### ****Part 3 Network Tools****.

#### **Basic Commands**

|  |  |  |
| --- | --- | --- |
| **Tool** | **Sample Command** | **Purpose** |
| ping | ping google.com | Test connectivity |
| ifconfig/ip | ifconfig or ip a | View network config |
| netstat | netstat -tulnp | View open ports/services |
| curl | curl ifconfig.me or curl google.com | HTTP requests |
| nslookup | nslookup google.com | DNS resolution |

Introduce UFW (Uncomplicated Firewall)

sudo ufw status

sudo ufw enable

sudo ufw allow 8080

sudo ufw deny 22