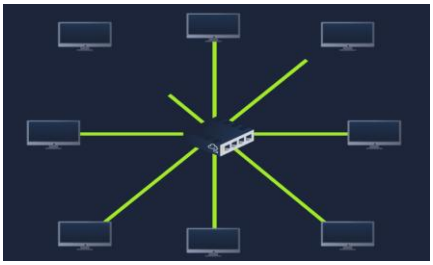
🔰 **Welcome to Pre-Security**

In these notes, we'll cover the basics you need to understand before starting your journey into cybersecurity. You'll learn about how computers, networks, and the internet work, as well as key security concepts. No prior experience needed — just curiosity and the will to learn!

Local Area Network (LAN) Topologies

Over the years, there has been experimentation and implementation of various network designs. In reference to networking, when we refer to the term "topology", we are actually referring to the design or look of the network at hand. Let's discuss the advantages and disadvantages of these topologies below.



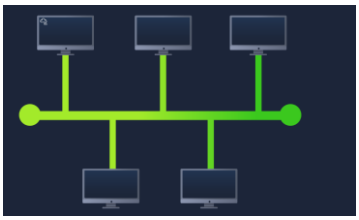
Star Topology

The main premise of a star topology is that devices are individually connected via a central networking device such as a switch or hub. This topology is the most commonly found today because of its reliability and scalability - despite the cost.

Any information sent to a device in this topology is sent via the central device to which it connects. Let's explore some of these advantages and disadvantages of this topology below:

Because more cabling and the purchase of dedicated networking equipment is required for this topology, it is more expensive than any of the other topologies. However, despite the added cost, this does provide some significant advantages. For example, this topology is much more scalable in nature, which means that it is very easy to add more devices as the demand for the network increases.

Unfortunately, the more the network scales, the more maintenance is required to keep the network functional. This increased dependence on maintenance can also make troubleshooting faults much harder. Furthermore, the star topology is still prone to failure - albeit reduced. For example, if the centralised hardware that connects devices fails, these devices will no longer be able to send or receive data. Thankfully, these centralised hardware devices are often robust.



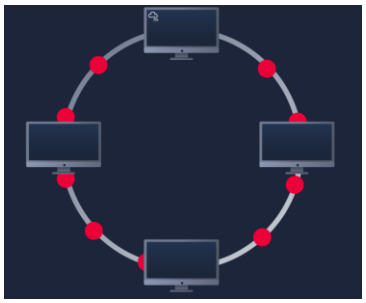
Bus Topology

This type of connection relies upon a single connection which is known as a backbone cable. This type of topology is similar to the leaf off of a tree in the sense that devices (leaves) stem from where the branches are on this cable.

Because all data destined for each device travels along the same cable, it is very quickly prone to becoming slow and bottlenecked if devices within the topology are simultaneously requesting data. This bottleneck also results in very difficult troubleshooting because it quickly becomes difficult to identify which device is experiencing issues with data all travelling along the same route.

However, with this said, bus topologies are one of the easier and more cost-efficient topologies to set up because of their expenses, such as cabling or dedicated networking equipment used to connect these devices.

Lastly, another disadvantage of the bus topology is that there is little redundancy in place in case of failures. This disadvantage is because there is a single point of failure along the backbone cable. If this cable were to break, devices can no longer receive or transmit data along the bus.



Ring Topology

The ring topology (also known as token topology) boasts some similarities. Devices such as computers are connected directly to each other to form a loop, meaning that there is little cabling required and less dependence on dedicated hardware such as within a star topology.

A ring topology works by sending data across the loop until it reaches the destined device, using other devices along the loop to forward the data. Interestingly, a device will only send received data from another device in this topology if it does not have any to send itself. If the device happens to have data to send, it will send its own data first before sending data from another device.

Because there is only one direction for data to travel across this topology, it is fairly easy to troubleshoot any faults that arise. However, this is a double-edged sword because it isn't an efficient way of data travelling across a network, as it may have to visit many multiple devices first before reaching the intended device.

Lastly, ring topologies are less prone to bottlenecks, such as within a bus topology, as large amounts of traffic are not travelling across the network at any one time. The design of this topology does, however, mean that a fault such as cut cable, or broken device will result in the entire networking breaking.

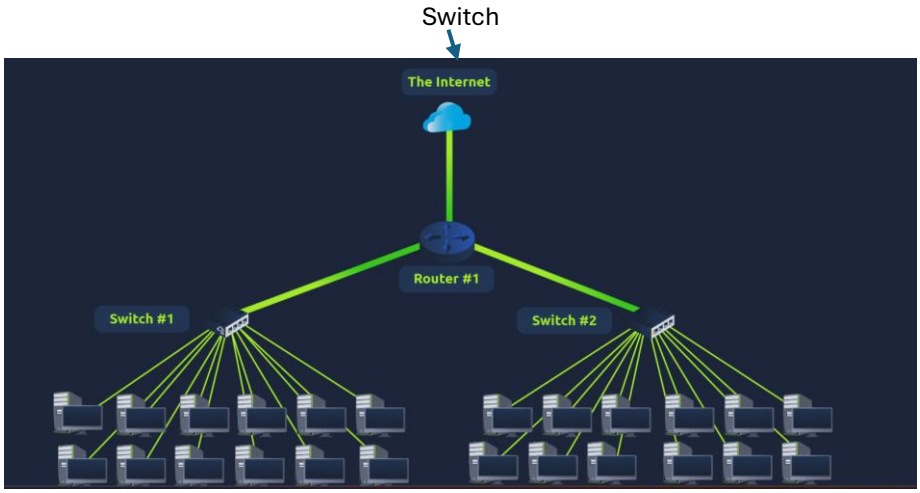
What is a Switch?

A **switch** is a network device used to connect multiple devices (like computers, printers, etc.) within the **same local network** using Ethernet. It operates at **Layer 2 (Data Link Layer)** of the OSI model and uses **MAC addresses** to identify and forward data to the correct device.

Key points:

- Found in larger networks (businesses, schools, etc.).
- Available in various port counts: 4, 8, 16, 24, 32, or 64.

- **Smarter than hubs**: Switches send packets only to the intended recipient (not to all devices), reducing traffic and improving efficiency.

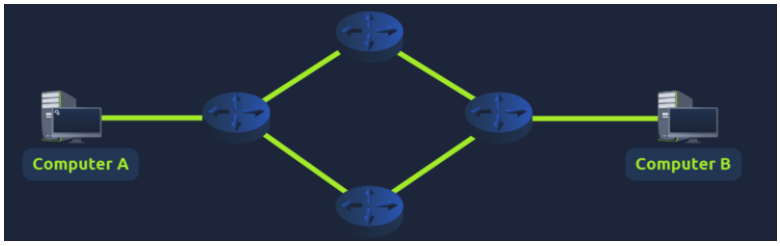


What is a Router?

A **router** is a device that connects **different networks** together and directs data between them. It operates at **Layer 3 (Network Layer)** of the OSI model and uses **IP addresses** to route traffic.

Key points:

- Essential for internet access (e.g., home router connects your home network to your ISP).
- Routes data between local networks (LANs) and wide-area networks (WANs).
- Makes routing decisions to find the best path for data to travel across networks.



Switches vs. Routers

Feature	Switch	Router
Purpose	Connects devices within a network	Connects different networks
OSI Layer	Layer 2 (Data Link)	Layer 3 (Network)
Uses	MAC addresses	IP addresses

Feature	Switch	Router
Traffic Control	Sends data only to intended device	Routes data across networks
Example Use	Office devices sharing a printer	Home network connecting to ISP

Redundancy & Reliability

Switches and routers can be connected together in various configurations to provide **redundancy**. This ensures that if one path fails, another can take over—improving **network reliability** with minimal downtime, even if it slightly affects performance.

What is Subnetting?

Subnetting is the process of dividing a larger network into smaller, more manageable **sub-networks (subnets)**. Think of it like slicing a cake: there's only so much cake (IP space) to go around, and subnetting lets you decide who gets what portion.

Why Subnet?

Organizations often have multiple departments (e.g., Accounting, Finance, HR), and just like in real life you'd send mail to the right department, networks need a way to route traffic correctly. Subnetting allows **network administrators** to group and manage devices efficiently.

IP Addresses & Subnet Masks

- An **IP address** is made of 4 **octets** (e.g., 192.168.1.100)
- A **subnet mask** is also 4 octets, used to define how many bits represent the **network** vs the **host**
    - Example: 255.255.255.0 = 24 bits for network, 8 bits for host

---

Subnet Address Types

Type	Purpose	Example
Network Address	Identifies the beginning of the subnet.	192.168.1.0
Host Address	Identifies individual devices within the subnet.	192.168.1.100
Default Gateway	Used to send data *outside* the subnet (to other networks). Usually .1 or .254.	192.168.1.254

When is Subnetting Used?

- **Home Networks**: Usually only use **one subnet**, since devices rarely exceed 254.
- **Business Networks**: Require **multiple subnets** to manage printers, PCs, cameras, etc.

Benefits of Subnetting

- ✅ **Efficiency** – Allocates IP ranges based on department/device count
- 🔒 **Security** – Isolates sensitive systems (e.g., POS devices vs. public Wi-Fi)
- 🧠 **Control** – Fine-tuned access and routing control between subnets

Real-World Example: A Café

- **Subnet 1**: Internal devices (registers, employee PCs)
- **Subnet 2**: Public Wi-Fi for customers

With subnetting, these two networks remain **separate**, yet both can access the **Internet**, enhancing both **security and functionality**.

🧠 **What is ARP (Address Resolution Protocol)?**

ARP is a protocol used to **map IP addresses to MAC addresses** in a local network.

🔁 It allows devices to find out the **physical (MAC) address** that corresponds to a known **IP address**.

📌 **Why is ARP Important?**

- Devices on a network use **IP addresses** to identify each other logically.
- But actual communication happens using **MAC addresses** (hardware addresses).
- ARP **links these two** so devices can communicate.

🔧 **How ARP Works**

Step-by-Step:

1. **Device A** wants to talk to **Device B** using its **IP address**.
2. It checks its **ARP cache** to see if it already knows Device B's **MAC address**.
3. If **not found**, Device A sends an **ARP Request**:

- This is a **broadcast** asking, “Who has IP address X.X.X.X?”

4. The device that owns that IP address (**Device B**) responds with an **ARP Reply**:
    - “I have that IP, and my MAC address is YY:YY:YY:YY:YY:YY.”
5. **Device A** stores the mapping (IP ↔ MAC) in its **ARP cache** for future communication.

📁 ARP Cache

- A local **table** that stores recent IP-to-MAC address mappings.
- Prevents the need to repeat ARP requests constantly.

📬 Types of ARP Messages

Message Type	Purpose
ARP Request	Asks: "Who has IP address X.X.X.X?" (Broadcast)
ARP Reply	Responds: "I do, and here’s my MAC address"

🧰 Real-World Use

- Enables communication in local networks (e.g., from your laptop to a printer).
- Used in **Ethernet** and **IPv4** networks.
- Can be **abused** in attacks like **ARP spoofing** (used in man-in-the-middle attacks).

How IP Addresses Are Assigned

IP addresses can be assigned in two primary ways:

1. **Manually** (Static IP):
    - The IP address is physically configured on the device by a user or administrator.
    - Often used for servers, printers, and other devices that need a consistent IP address.
2. **Automatically** (Dynamic IP via DHCP):
    - The **Dynamic Host Configuration Protocol (DHCP)** is used to automatically assign IP addresses to devices on a network.
    - This is the most common method, especially in home and enterprise networks.

DHCP

IP addresses can be assigned either manually, by entering them physically into a device, or automatically and most commonly by using a DHCP (Dynamic Host Configuration Protocol) server. When a device connects to a network, if it has not already been manually assigned an IP address, it sends out a request (DHCP Discover) to see if any DHCP servers are on the network. The DHCP server then replies back with an IP address the device could use (DHCP Offer). The device then sends a reply confirming it wants the offered IP Address (DHCP Request), and then lastly, the DHCP server sends a reply acknowledging this has been completed, and the device can start using the IP Address (DHCP ACK).

DHCP Process (DORA)

When a device connects to a network and doesn't have an assigned IP, it goes through the **DORA** process:

1. **D**iscover:
    - The device sends a **DHCP Discover** message to locate any DHCP servers on the network.
2. **O**ffer:
    - A DHCP server responds with a **DHCP Offer**, suggesting an available IP address and related configuration.
3. **R**equest:
    - The device responds with a **DHCP Request**, indicating it would like to use the offered IP.
4. **A**cknowledgement:
    - The DHCP server sends a **DHCP ACK**, confirming the lease of the IP address and other network settings (e.g., subnet mask, gateway, DNS).

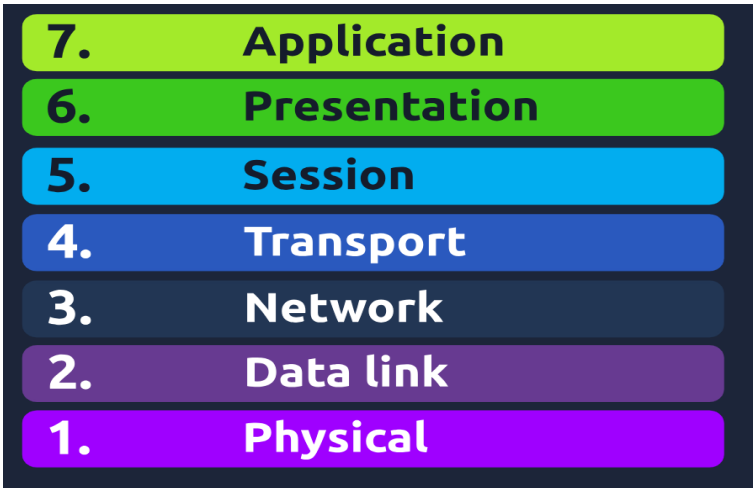
OSI Model

The OSI model (or Open Systems Interconnection Model) is an essential model used in networking. This critical model provides a framework dictating how all networked devices will send, receive and interpret data.

One of the main benefits of the OSI model is that devices can have different functions and designs on a network while communicating with other devices. Data sent across a network that follows the uniformity of the OSI model can be understood by other devices.

The OSI model consists of seven layers which are illustrated in the diagram below. Each layer has a different set of responsibilities and is arranged from Layer 7 to Layer 1.

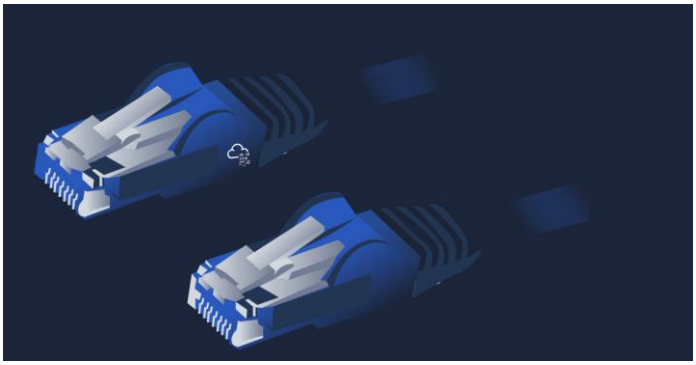
At every individual layer that data travels through, specific processes take place, and pieces of information are added to this data, which is what we'll come to discuss in the upcoming tasks within this room. However, for now, we only need to understand that this process is called encapsulation and what the OSI model looks like in the diagram below:



Physical layer (Layer 1 of the OSI Model)

This layer is one of the easiest layers to grasp. Put simply, this layer references the physical components of the hardware used in networking and is the lowest layer that you will find. Devices use electrical signals to transfer data between each other in a binary numbering system (1's and 0's).

For example, ethernet cables connecting devices, such as in the picture below:



Data Link (Layer 2 of the OSI Model)

The data link layer focuses on the physical addressing of the transmission. It receives a packet from the network layer (including the IP address for the remote computer) and adds in the physical **MAC** (Media Access Control) address of the receiving endpoint. Inside every network-enabled computer is a **N**etwork **I**nterface Card (**NIC**) which comes with a unique MAC address to identify it.

MAC addresses are set by the manufacturer and literally burnt into the card; they can't be changed – although they can be spoofed. When information is sent across a network, it's actually the physical address that is used to identify where exactly to send the information.

Additionally, it's also the job of the data link layer to present the data in a format suitable for transmission.

Network Layer (Layer 3 of the OSI Model)

The **Network Layer** is responsible for **routing** and **reassembling data** from smaller chunks (packets) into the original data.

Routing

Routing is the process of determining the most optimal path for data to travel through a network to reach its destination. The decision is influenced by:

- **Shortest path** (least number of devices or hops)
- **Reliability** (based on packet loss history on certain routes)
- **Speed of the connection** (e.g., fiber is faster than copper)

Routing Protocols

At this stage, you only need to be aware of the existence of protocols that help with routing decisions:

- **OSPF** (Open Shortest Path First)
- **RIP** (Routing Information Protocol)

OSPF (Open Shortest Path First)

- **Type:** Link-state routing protocol
- **Function:** Calculates the **shortest and most efficient path** using Dijkstra's algorithm.
- **Updates:** Sends updates only when changes occur, making it faster and more scalable for large networks.
- **Use case:** Preferred in enterprise and large networks due to its efficiency and speed.

RIP (Routing Information Protocol)

- **Type:** Distance-vector routing protocol
- **Function:** Chooses routes based on **hop count** (maximum of 15 hops).
- **Updates:** Sends full routing table updates every 30 seconds, which can cause delays and more bandwidth usage.
- **Use case:** Simple and easy to configure, but only suitable for small networks.

These protocols evaluate the network to determine the best possible route for data.

IP Addresses

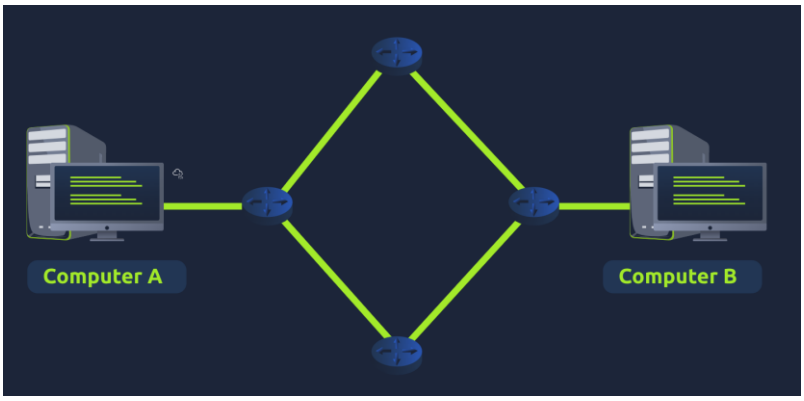
This layer deals with IP addresses like 192.168.1.100 to identify devices on the network.

Layer 3 Devices

Devices like **routers** operate at this layer. They are capable of:

- Reading IP addresses
- Making routing decisions
- Delivering packets across networks

These are referred to as **Layer 3 devices** because they function at the third layer of the OSI model.



Transport Layer (Layer 4 of the OSI Model)

Layer 4 of the OSI model plays a vital part in transmitting data across a network and can be a little bit difficult to grasp. When data is sent between devices, it follows one of two different protocols that are decided based upon several factors:

- TCP
- UDP

TCP

The **T**ransmission **C**ontrol **P**rotocol (**TCP**). Potentially hinted by the name, this protocol is designed with reliability and guarantee in mind. This protocol reserves a constant connection between the two devices for the amount of time it takes for the data to be sent and received.

Not only this, but TCP incorporates error checking into its design. Error checking is how TCP can guarantee that data sent from the small chunks in the session layer (layer 5) has then been received and reassembled in the same order.

Let's summarise the advantages and disadvantages of TCP in the table below:

Advantages of TCP	**Disadvantages of TCP**
Guarantees accurate and complete data delivery	Requires a reliable connection between devices
Synchronizes devices to avoid data flooding	If any data chunk is lost, the entire transmission may be affected

Advantages of TCP	Disadvantages of TCP
Ensures data is delivered in order	A slow receiver can cause bottlenecks, holding the connection open
Built-in error checking and recovery mechanisms	Slower performance compared to protocols like UDP due to overhead and extra processes

TCP is used for situations such as file sharing, internet browsing or sending an email. This usage is because these services require the data to be accurate and complete (no good having half a file!).

In the diagram below, we can see how a picture of a cat is broken down into small pieces of data (known as packets) from the "webserver", where the "computer" re-constructs the picture of the cat into the correct order.

UDP

Now let's move onto the **U**ser **D**atagram **P**rotocol (or **UDP** for short). This protocol is not nearly as advanced as its brother - the TCP protocol. It doesn't boast the many features offered by TCP, such as error checking and reliability. In fact, any data that gets sent via UDP is sent to the computer whether it gets there or not. There is no synchronisation between the two devices or guarantee; just hope for the best, and fingers crossed.

Whilst this sounds disadvantageous, it does have its merits, which we'll layout in the table below:

Advantages of UDP	Disadvantages of UDP
Much faster than TCP	No guarantee that data is received
Does not maintain a continuous connection (connectionless)	Poor performance over unstable or unreliable connections
Leaves flow control to the application layer	No built-in mechanisms for packet ordering or error correction
Offers flexibility for software developers	Increased complexity for developers who need reliable transmission

Using the same example as before, we can now see that only Packets #1 and #3 have been received by the "Computer", meaning that half of the image is missing.

UDP is useful in situations where there are small pieces of data being sent. For example, protocols used for discovering devices (*ARP* and *DHCP)* or larger files such as video streaming (where it is okay if some part of the video is pixelated. Pixels are just lost pieces of data!)

Session Layer (Layer 5 of the OSI Model)

The **session layer (Layer 5)** of the OSI model is responsible for starting, managing, and ending communication between two devices. When two devices connect and start talking to each other, a **session** is created — think of it like a temporary chat room just for them. While the session is active, the devices can send and receive data. If the connection is lost or inactive for too long, the session layer will close it. Sessions can also include **checkpoints**, which help resume data transfers from where they left off if something goes wrong, instead of starting over. In short, a **session** is a controlled, organized conversation between two devices.

when you visit a webpage, a session is usually created between your computer (the client) and the server.

This session helps the server remember things like:

- Who you are (if you logged in)
- What's in your shopping cart
- Your preferences or activity on the page

This is especially true for **web applications** and dynamic websites (like Gmail, Amazon, etc.).

Presentation Layer (Layer 6 of the OSI Model)

What does Layer 6 (Presentation Layer) do?

- It **translates** data between different formats so that both the sender and receiver can understand it — even if they're using different software.
- It ensures that **data looks the same** no matter what program is used (e.g. two people using different email apps can still read the same email correctly).

- It also handles **encryption and decryption**, like when you visit a secure website using HTTPS.

✅ **Key Roles:**

- Data **formatting** and **conversion**
- **Translation** between application-level data types
- **Compression** (e.g. reducing file size)
- **Encryption/Decryption** for security

So in short:
Layer 6 is the translator and security guard of network data. It makes sure the message gets through in the right format — and stays safe along the way.

Application Layer (Layer 7 of the OSI Model

The application layer is the part of the OSI model you're most likely familiar with. This is because it's where protocols and rules exist to guide how users interact with data being sent or received.

Everyday applications like email clients, web browsers, or file transfer programs such as FileZilla provide a user-friendly graphical interface (GUI) to help you manage this data. Additionally, protocols like DNS (Domain Name System) work behind the scenes to translate website addresses into IP addresses, making it easier to access websites.

Packets and Frames

Packets and Frames: Understanding the Difference

Packets and frames are small chunks of data that come together to form larger messages or pieces of information. Although they might seem similar, they belong to different layers of the OSI model and have distinct roles.

A **frame** exists at Layer 2 — the Data Link layer. This means it doesn't contain IP addresses. To visualize this, imagine putting an envelope inside another envelope before sending it. The outer envelope is like a **packet** that you mail, and when that envelope is opened, you find the inner envelope — the **frame** — which still contains data.

This process is called **encapsulation** (which we discussed earlier in Room 3: the OSI model). When we talk about IP addresses, we are referring to packets, because IP addressing happens at Layer 3 — the Network layer. When the outer packet is stripped away, what remains is the frame.

Why Packets Are Efficient

Packets allow devices on a network to communicate efficiently by breaking data into small pieces. Sending smaller chunks reduces the chance of network bottlenecks compared to sending large messages all at once.

For example, when you load an image from a website, the image isn't sent as a whole file. Instead, it's divided into multiple packets, which are then sent separately and reassembled on your computer. Imagine the picture of a cat being split into three packets — once all arrive, your computer reconstructs them into the full image.

Packet Structure and Standards

Packets have different structures depending on the protocol being used. Networking relies heavily on standards and protocols to ensure data is handled consistently, which is crucial when billions of devices are connected to the internet.

Taking the **Internet Protocol (IP)** as an example, each packet includes headers — pieces of information that help manage and route the data properly.

Some important IP packet headers include:

Header	Description
Time to Live	Sets an expiry timer so packets don't clog the network if they get lost or stuck.
Checksum	Checks data integrity — if the data changes during transit, this helps detect corruption.
Source Address	The IP address of the device sending the packet, so responses can be routed back correctly.
Destination Address	The IP address of the device meant to receive the packet, guiding its journey through the network.

TCP (or **T**ransmission **C**ontrol **P**rotocol for short) is another one of these rules used in networking.

This protocol is very similar to the OSI model that we have previously discussed in room three of this module so far. The TCP/IP protocol consists of four layers and is arguably just a summarised version of the OSI model. These layers are:

- Application
- Transport
- Internet
- Network Interface

Very similar to how the OSI model works, information is added to each layer of the TCP model as the piece of data (or packet) traverses it. As you may recall, this process is known as encapsulation - where the reverse of this process is decapsulation.

One defining feature of TCP is that it is **connection-based**, which means that TCP must establish a connection between both a client and a device acting as a server **before** data is sent.

Because of this, TCP guarantees that any data sent will be received on the other end. This process is named the Three-way handshake, which is something we'll come on to discuss shortly. A table comparing the advantages and disadvantages of TCP is located below:

Advantages and Disadvantages of TCP

Advantages of TCP	Disadvantages of TCP
✅ Ensures the **integrity of data** — no data is accepted unless it's received correctly.	❌ Requires a **reliable connection**. If even one piece of data is lost, the entire segment may need to be re-sent.
✅ Can **synchronize devices** to prevent flooding or out-of-order data.	❌ **Slower** performance compared to UDP due to error-checking and connection management.
✅ Handles multiple **reliability mechanisms**, such as acknowledgements, error detection, and retransmission.	❌ A **slow connection** can occupy a connection slot on the receiving device, potentially causing bottlenecks.

TCP Packet Structure (Key Headers)

TCP packets include several headers that provide essential information during communication. These are added during the **encapsulation** process:

Header	Description
Source Port	The port used by the sender to send the data. Chosen randomly from the available range (0–65535).
Destination Port	The port used by the receiving service or application (e.g., port 80 for HTTP).

Header	Description
Source IP	IP address of the device sending the packet.
Destination IP	IP address of the device meant to receive the packet.
Sequence Number	A random number assigned to the first packet. Used to keep data in order and track it.
Acknowledgement Number	Confirms receipt of data by indicating the next expected sequence number.
Checksum	A value used to verify data integrity. If the result of a checksum calculation doesn't match, the packet is considered corrupted.
Data	The actual content (payload) being transferred — e.g., part of a file.
Flag	Indicates the purpose of the packet (e.g., starting, continuing, or ending a connection).

The TCP Three-Way Handshake

This is the process used to **establish a reliable connection** between two devices before data is exchanged.

Step	Message	Description
1	**SYN**	The client sends a SYN packet to the server to initiate the connection.
2	**SYN/ACK**	The server responds with a SYN/ACK packet to acknowledge the request and synchronize.
3	**ACK**	The client sends an ACK to confirm the connection has been established.
4	**DATA**	After the handshake, data (like file bytes or messages) is transferred.
5	**FIN**	Either side can send a FIN message to cleanly close the connection.
#	**RST**	If something goes wrong, a RST message is sent to immediately terminate the connection.

User Datagram Protocol (UDP)

UDP is another core protocol used to transmit data between devices — but it operates very differently from TCP.

Unlike its more reliable counterpart, **UDP is a stateless protocol**, meaning it doesn't require a constant connection between devices. There's **no Three-Way Handshake**, no synchronization, and no guarantees that the data will even be received.

This makes UDP ideal for situations where **speed matters more than reliability**, or when **occasional data loss is acceptable** — like in **video streaming, voice calls, online gaming**, or **broadcast messages**.

Advantages and Disadvantages of UDP

Advantages of UDP	Disadvantages of UDP
🚀 **Faster than TCP** — no overhead from handshakes or acknowledgements.	❌ **No delivery guarantee** — UDP doesn't check if the data actually arrived.
⚙️ **Leaves control to the application** — software decides how to handle data flow and retransmissions.	❌ **Less stable experience** on weak connections — packet loss or delay can affect performance noticeably.
📡 **No reserved connection** — less resource usage on the network and end devices.	❌ **No error correction** — there's no mechanism for detecting or fixing corrupted data.

UDP Packet Structure (Key Headers)

UDP packets are simpler than TCP packets — they contain fewer headers and skip reliability checks. However, some fields are shared between both protocols:

Header	Description
Time to Live (TTL)	Sets a lifespan for the packet — if it doesn't reach its destination in time, it's discarded to avoid network congestion.
Source Address	The IP address of the device sending the data — helps with responses if needed.
Destination Address	The IP address of the intended recipient device.
Source Port	The randomly chosen port used by the sender to transmit data (from the range 0–65535).
Destination Port	The specific port on the receiving device where the data should be delivered (e.g., port 80 for HTTP).
Data	The actual payload — the content being transmitted.

How UDP Works (Compared to TCP)

- **No connection setup:** UDP skips the handshake process used in TCP.
- **No acknowledgements:** The receiving device does not confirm whether it got the data.
- **No order guarantees:** Packets may arrive out of order — or not at all.
- **No congestion control:** UDP keeps sending packets, regardless of network condition.

This makes UDP very fast — but also **risky** in terms of reliability. That's why it's used in **real-time applications** where performance and speed are more important than accuracy.

What Are Ports in Networking?

Just like ships dock at specific ports in a harbour, **network ports** act as specific "doors" through which data enters or exits a device. These ports are numbered from **0 to 65535**, and each port is associated with a particular type of traffic or service.

Analogy: Ships and Harbours

Imagine:

- A **ship** = a data packet
- A **port** = a specific location at a harbour made for certain ship types

For example, a cruise liner (large data) can't dock at a fishing port (small service), and vice versa. Similarly, in networking, different types of data require different ports. If the wrong type of data tries to access an incompatible port, the communication is rejected.

How Ports Are Used in Networking

When a connection is made between devices (like a client and server), all communication flows through these **numbered ports**. These port numbers help the system know **which application or service should handle the incoming or outgoing data**.

To maintain order and consistency, the industry uses **standardized port numbers** for common services, especially those in the **range of 0 to 1023**, which are known as **well-known ports**.

Common Port Numbers and Their Protocols

Protocol	Port Number	Description
FTP (File Transfer Protocol)	21	Used for transferring files using a client-server setup.
SSH (Secure Shell)	22	Securely accesses and manages devices via a terminal (text-based).
HTTP (HyperText Transfer Protocol)	80	Used by web browsers to access and load websites.
HTTPS (Secure HTTP)	443	Same as HTTP, but encrypted for secure web browsing.
SMB (Server Message Block)	445	Used for file and printer sharing on local networks.
RDP (Remote Desktop Protocol)	3389	Allows you to remotely control a computer with a full graphical interface.

Custom Ports and Flexibility

Although these are **standard port numbers**, you **can change** them if needed. For example:

- A web server usually runs on **port 80**.
- You could configure it to use **port 8080** instead.

But be aware: users or software connecting to this service must be told explicitly (e.g., http://example.com:8080), because they'll assume the standard port by default if no port is specified.

Key Takeaways

- Ports help systems know **where to send or receive data**.
- Ports range from **0 to 65535**, with **0–1023 being standard (well-known) ports**.
- **Protocols are assigned standard ports**, but this is **not mandatory**.
- Changing ports is possible, but doing so requires **manual configuration and awareness** on the client side.

What Is Port Forwarding?

Port forwarding is a technique used to allow external devices on the **internet** to access services (like web servers, game servers, or CCTV systems) running on **private networks**.

Without port forwarding, services on a private network (like a local web server) are **only accessible within that same network** — this is known as an **intranet**.

Example: Internal-Only Access

Imagine a small network with this setup:

- **Web server IP:** 192.168.1.10
- **Service:** Web server running on **port 80**

In this case, only other devices on the same local network can access the website using 192.168.1.10:80. Devices **outside the network** (e.g., someone from the internet) **cannot reach it**.

Making It Public: Port Forwarding

To make that same web server **accessible from the internet**, you need to **forward port 80** on your **router** to 192.168.1.10.

Here's what happens after configuring port forwarding:

- **Public IP address of the network:** 82.62.51.70
- External users can now visit: http://82.62.51.70

- The router automatically **forwards traffic on port 80** to the internal server at 192.168.1.10

This makes the private server **publicly accessible**.

Port Forwarding vs. Firewalls

It's easy to confuse **port forwarding** with **firewall rules**, but here's the difference:

Feature	Port Forwarding	Firewall
What it does	Routes incoming traffic to a specific device	Allows or blocks traffic based on rules
Where it's set	On the router	On the router *or* individual devices (like servers)
Key role	Opens a path to a device on your network	Decides if traffic on that path is allowed

Important: Just because a port is forwarded doesn't mean traffic will reach it — if the firewall blocks that port, communication still won't happen.

Final Note

- Port forwarding is **configured on your router**.
- It's critical when you want to **host something on your local network** that needs to be accessible from the internet (like a game server, web server, or home surveillance system).

🔥 What Is a Firewall?

A **firewall** is a **security system**—either hardware, software, or both—that controls what network traffic is **allowed in or out** of a network. You can think of it as a **border security checkpoint** for data.

🛂 What Can a Firewall Decide?

Firewalls inspect network traffic and make decisions based on a set of rules, such as:

- **Source**: Where is the traffic coming from? (e.g., deny all traffic from a suspicious IP range)
- **Destination**: Where is the traffic going? (e.g., block traffic going to unauthorized internal servers)
- **Port number**: What service is the traffic trying to access? (e.g., only allow HTTP on port 80)
- **Protocol**: What kind of traffic is it? (e.g., allow TCP but block UDP)

To make these decisions, firewalls perform **packet inspection**—looking at the metadata and contents of network packets.

🔒 Types of Firewalls

Firewalls vary in complexity and performance depending on how they inspect traffic. Two common types are:

Type	Description
Stateful Firewall	- Tracks the **entire connection state**, not just individual packets. - Makes dynamic decisions based on the context of the traffic (e.g., whether a full TCP handshake is taking place). - **More secure**, but **resource-intensive**. - Can block entire devices if suspicious behavior is detected.
Stateless Firewall	- Makes decisions based on **predefined static rules** applied to each **individual packet**. - Doesn't track the full connection—faster, but less intelligent. - **Lighter on resources**, ideal for environments expecting large volumes of traffic (e.g., protection from DDoS attacks). - Only as good as the rules it's given.

A **stateless firewall** filters traffic based on fixed rules for individual packets without tracking the state of a connection, making it fast but less intelligent. In contrast, a **stateful firewall** monitors the entire connection's state and context, allowing it to make more informed decisions, but it uses more system resources.

🧱 Where Are Firewalls Used?

- **Enterprise firewalls**: Dedicated hardware appliances (e.g., Palo Alto, Fortinet) used in corporate networks.
- **Home routers**: Built-in basic firewall functionality (NAT, port filtering).
- **Software firewalls**: Programs like **Snort**, **iptables**, or **Windows Defender Firewall** that run on individual systems.
- **Cloud firewalls**: Provided by cloud platforms like AWS, Azure, GCP to protect cloud-based resources.

Virtual Private Network (VPN)

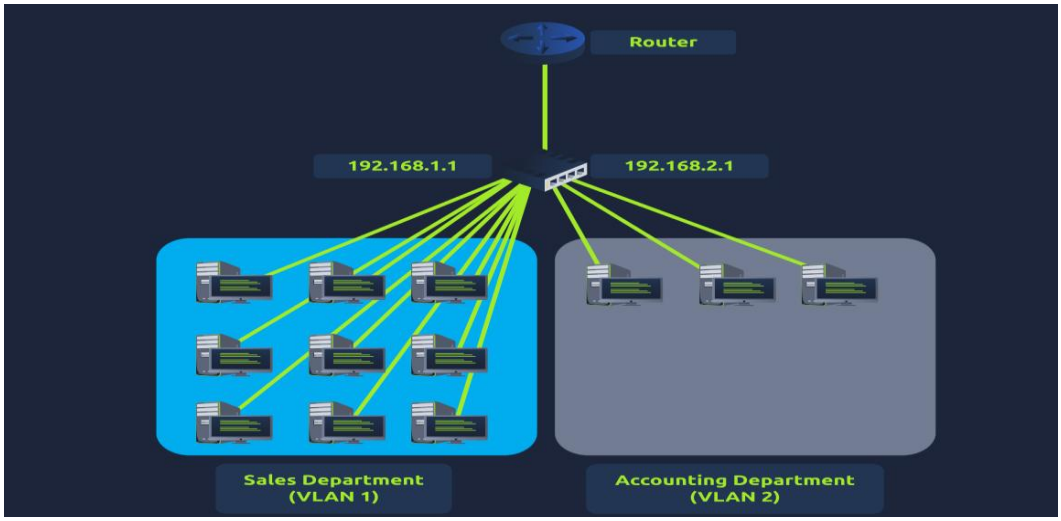
A **Virtual Private Network (VPN)** creates a secure, encrypted tunnel between devices over the Internet, allowing them to communicate as if they were on the same private network. This enables secure remote access, protects data from eavesdropping (especially on public WiFi), and can provide anonymity. VPNs are used in businesses to connect offices, by individuals for privacy, and by platforms like TryHackMe to safely connect users to remote machines without exposing them to the public internet. Technologies like **PPTP**, **PPP**, and **IPSec** are common in VPN implementations, each offering varying levels of encryption and compatibility.

Benefit	Description
Allows networks in different geographical locations to be connected	A business with multiple offices can use a VPN to connect their networks, allowing access to shared resources like servers and infrastructure from any location.
Offers privacy	VPNs use encryption to protect data, so only the sender and intended receiver can understand it. This is especially helpful on public WiFi, where traffic would otherwise be exposed to eavesdropping.
Offers anonymity	VPNs hide your internet activity from ISPs and other intermediaries. This is crucial for journalists and activists operating in regions with restricted freedom of speech. However, true anonymity depends on the VPN provider's privacy practices.

VPN Technology	Description
PPP (Point-to-Point Protocol)	Used by PPTP to provide authentication and encryption. Works with private keys and public certificates. Cannot leave the local network by itself (non-routable).
PPTP (Point-to-Point Tunneling Protocol)	Allows PPP data to travel over the Internet. Easy to set up and supported on most devices, but offers weak encryption.
IPSec (Internet Protocol Security)	Encrypts data within the IP framework. More complex to set up, but offers strong encryption and is widely supported.

VLAN

A technology called **VLAN** (**V**irtual **L**ocal **A**rea **N**etwork) allows specific devices within a network to be virtually split up. This split means they can all benefit from things such as an Internet connection but are treated separately. This network separation provides security because it means that rules in place determine how specific devices communicate with each other. This segregation is illustrated in the diagram below:

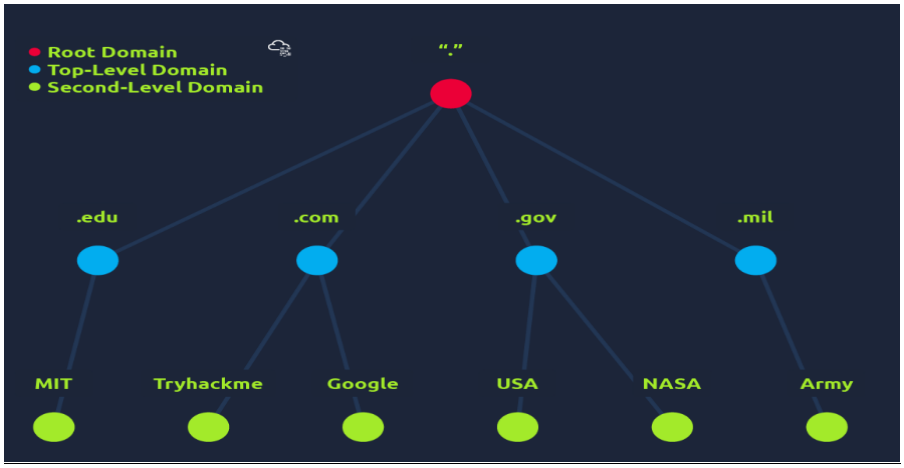


DNS – Domain Name System

What is DNS?

DNS (Domain Name System) provides a simple way for us to communicate with devices on the internet without remembering complex numbers. Much like every house has a unique address for sending mail directly to it, every computer on the internet has its own unique address to communicate with it called an IP address. An IP address looks like the following 104.26.10.229, 4 sets of digits ranging from 0 - 255 separated by a period. When you want to visit a website, it's not exactly convenient to remember this complicated set of numbers, and that's where DNS can help. So instead of remembering 104.26.10.229, you can remember google.com instead.

Domain Hierarchy



🌐 TLD (Top-Level Domain)

- The rightmost part of a domain name.
- Example: In tryhackme.com, the TLD is .com.

Types of TLDs:

- gTLD (Generic): Describes purpose
  Examples: .com (commercial), .org (organization), .edu (education), .gov (government), .online, .club, .biz
- ccTLD (Country Code): Indicates location
  Examples: .ca (Canada), .co.uk (United Kingdom), .de (Germany)

🔤 Second-Level Domain

- Comes directly before the TLD.
  Example: In tryhackme.com, tryhackme is the Second-Level Domain.

Rules:

- Max length: 63 characters (not including TLD)
- Allowed: a–z, 0–9, and hyphens (-)
- Cannot:
    - Start or end with a hyphen
    - Have consecutive hyphens

🧩 Subdomain

- Appears to the left of the Second-Level Domain, separated by dots.
  Example: In admin.tryhackme.com, admin is the subdomain.

Features & Rules:

- Same character rules as Second-Level Domains
- Max length per subdomain: 63 characters
- Full domain length (including all parts): Max 253 characters
- Unlimited number of subdomains allowed
  Example: jupiter.servers.tryhackme.com

DNS Record Types

🔵 **A Record (Address Record)**

An **A (Address) record** maps a **domain name** to an **IPv4 address**. When you type a domain like example.com into your browser, a DNS resolver queries the DNS records to find the A record, which returns the IPv4 address (like 104.26.10.229) associated with that domain. Your browser then uses this IP address to connect to the web server hosting the site.

Why it matters: Without an A record, domain names would not resolve to servers. A records are fundamental to how the internet works.

🟢 **AAAA Record (IPv6 Address Record)**

The **AAAA record** is very similar to the A record, but instead of mapping to an IPv4 address, it maps a domain to an **IPv6 address** (for example, 2606:4700:20::681a:be5).

Why it matters: IPv6 is the future of internet addressing due to the shortage of IPv4 addresses. Networks with modern infrastructure often have both A and AAAA records to support both address types.

🟡 CNAME Record (Canonical Name Record)

A **CNAME record** maps a domain name **to another domain name**, rather than to an IP address. For example, store.tryhackme.com might be a CNAME for shops.shopify.com. This means that when a user tries to access store.tryhackme.com, DNS first resolves the CNAME to shops.shopify.com, and then looks up the A or AAAA record for that target.

Why it matters: CNAMEs simplify DNS management. Instead of having to update the IP address in multiple records, you point to a canonical (master) domain and only maintain its IP address.

🔴 MX Record (Mail Exchange Record)

The **MX record** tells mail servers where to deliver **email for a domain**. For example, when you send an email to user@tryhackme.com, your mail server looks up the MX record for tryhackme.com to find out which server(s) should receive the message — e.g., alt1.aspmx.l.google.com.

Each MX record includes a **priority value** (lower is higher priority), so if one mail server is down, email delivery can fail over to another.

Why it matters: Misconfigured MX records can break email delivery. They're also a critical part of email security configurations, such as preventing spam and phishing.

🔵 TXT Record (Text Record)

TXT records are flexible DNS records that allow domain owners to store **free-form text** in DNS. While they can hold any text, they're commonly used for:

- **SPF records**: To specify which mail servers are allowed to send emails on behalf of your domain.
- **Domain verification**: Many services (like Google Workspace or Microsoft 365) require you to add a specific TXT record to prove ownership of a domain.
- **DKIM/DMARC**: Email authentication protocols also use TXT records to publish cryptographic keys and policies.

Why it matters: TXT records are key in fighting **email spoofing, phishing, and spam**. They also help validate domain ownership in automation systems and third-party integrations.

Summary:

- **A/AAAA**: Point a name to an IP address.
- **CNAME**: Point a name to another name.
- **MX**: Direct email traffic to the right servers.
- **TXT**: Store text data for security and verification.

Here's a detailed explanation of **what happens when you make a DNS request**, along with a step-by-step breakdown.

🌐 **Step-by-Step DNS Resolution Process**

1. **Client Cache Check**
    - When you type a domain like www.tryhackme.com in your browser, your **computer first checks its local DNS cache**.
    - If it finds a valid (not expired) record, it uses that IP address immediately.
    - ✅ If found → Request ends here (fastest resolution).
    - ❌ If not found → It forwards the request to the next step.

---

2. **Recursive Resolver (DNS Recursor)**
    - Your machine contacts a **Recursive DNS Server**, often provided by your **ISP** (like 8.8.8.8 for Google DNS or 1.1.1.1 for Cloudflare).
    - The recursive resolver **also checks its cache** for the answer.
    - ✅ If it has it → It returns the result to your computer.
    - ❌ If not → It starts querying other DNS servers to find the answer (this is where the recursion kicks in).

---

3. **Root DNS Server**
    - The recursive resolver contacts a **Root DNS Server** (one of 13 global root servers).
    - The root server doesn't know the exact IP of www.tryhackme.com, but it **knows where the Top-Level Domain (TLD) servers** for .com domains are.
    - It responds with a **referral** to the correct .com TLD server.

---

4. **TLD DNS Server**
    - The resolver now asks the **TLD server for .com** where to find the authoritative DNS server for tryhackme.com.
    - The TLD server responds with the **name of the authoritative nameserver**, e.g., kip.ns.cloudflare.com.

---

5. **Authoritative DNS Server**
    - Finally, the resolver queries the **Authoritative DNS Server** (Cloudflare in this case).

- This server **stores the actual DNS records** (A, CNAME, MX, etc.) for tryhackme.com.
- The authoritative server returns the IP address or DNS record requested (e.g., A record with 104.26.10.229).

---

6. **Caching and Response**
   - The recursive resolver:
     - **Caches the response** based on its TTL (Time To Live) value (e.g., 300 seconds).
     - **Returns the final result to your computer**.
   - Your computer also caches it for future use, reducing lookup time.

🧠 **Summary in Plain Terms:**

When you request a website, your system goes through several steps: local check → ISP DNS check → root server → TLD server → nameserver → and back with the IP. It caches the answer to avoid doing this every time.

HTTP & HTTPS

What is HTTP? (HyperText Transfer Protocol)

HTTP is what's used whenever you view a website, developed by Tim Berners-Lee and his team between 1989-1991. HTTP is the set of rules used for communicating with web servers for the transmitting of webpage data, whether that is HTML, Images, Videos, etc.

What is HTTPS? (HyperText Transfer Protocol Secure)

HTTPS is the secure version of HTTP. HTTPS data is encrypted so it not only stops people from seeing the data you are receiving and sending, but it also gives you assurances that you're talking to the correct web server and not something impersonating it.

🌐 **What is a URL? (Uniform Resource Locator)**

A **URL** is like an address for finding a resource on the internet. It tells your browser *how* to connect and *what* to connect to. A URL can include several parts:

📌 **Example URL:**

http://user:password@tryhackme.com:80/view-room?id=1#task3

Part	Description
Scheme (http)	Tells the browser what protocol to use, like HTTP or HTTPS.
User:Password (user:password)	Optional. Used for basic authentication (not common nowadays).

Part	Description
Host (tryhackme.com)	The domain name or IP address of the server you're trying to reach.
Port (80)	Optional. Specifies which port to connect to. HTTP uses 80, HTTPS uses 443 by default.
Path (/view-room)	The exact location or page/resource on the server.
Query String (?id=1)	Sends additional info to the server (e.g. article ID, search term).
Fragment (#task3)	Refers to a specific section of the page (used in navigation).

📬 Making a Request

When you visit a URL, your browser sends a **request** to the web server to fetch the page.

Example Request:

GET / HTTP/1.1

Host: tryhackme.com

User-Agent: Mozilla/5.0 Firefox/87.0

Referer: https://tryhackme.com/

Explanation:

- **Line 1:** GET / HTTP/1.1 → Asks for the homepage / using the GET method (HTTP version 1.1).
- **Line 2:** Host: tryhackme.com → Specifies which website to access (needed in shared hosting).
- **Line 3:** User-Agent: Mozilla/5.0 ... → Tells the server what browser/device is being used.
- **Line 4:** Referer: → Tells where you came from (previous page).
- **Line 5:** The blank line tells the server the request is complete.

📩 Getting a Response

The server responds with headers and content.

Example Response:

HTTP/1.1 200 OK

Server: nginx/1.15.8

Date: Fri, 09 Apr 2021 13:34:03 GMT

Content-Type: text/html

Content-Length: 98

```
<html>
<head>
  <title>TryHackMe</title>
</head>
<body>
  Welcome To TryHackMe.com
</body>
</html>
```

Explanation:

- **Line 1:** Status code 200 OK → Everything worked.
- **Line 2:** Server type/version (e.g. nginx).
- **Line 3:** Server's date/time.
- **Line 4:** Type of content (e.g. HTML, JSON, image).
- **Line 5:** Size of the content in bytes.
- **Line 6:** Blank line signals end of headers.
- **Lines 7–end:** The actual content (in this case, a simple HTML page).

HTTP Methods

HTTP methods are a way for the client to show their intended action when making an HTTP request. There are a lot of HTTP methods but we'll cover the most common ones, although mostly you'll deal with the GET and POST method.

GET Request - This is used for getting information from a web server.

POST Request - This is used for submitting data to the web server and potentially creating new records

PUT Request - This is used for submitting data to a web server to update information

DELETE Request - This is used for deleting information/records from a web server.

🌐 What Are HTTP Status Codes?

When your browser (the **client**) sends a request to a website's **server**, the server always replies with a **status code**. This 3-digit number tells you what happened with your request — whether it succeeded, failed, was redirected, or something else.

🧱 HTTP Status Code Categories

✅ 100–199: *Informational Responses*

These are rare today. They let the client know that the initial part of the request was received, and they can continue.

- **100 Continue** – The server received the headers, and the client can now send the body.
- **101 Switching Protocols** – The client requested a protocol switch (e.g., from HTTP to WebSocket).

🟢 200–299: *Success*

These codes mean your request worked!

- **200 OK** – Everything went well. The server is returning the requested content.
- **201 Created** – Used after creating something new, like a user account or a blog post. Common in APIs.
- **204 No Content** – The request was successful, but there's nothing to send back (e.g., a DELETE request).

🔁 300–399: *Redirection*

The content is not here. Your browser is redirected somewhere else.

- **301 Moved Permanently** – The resource has been permanently moved to another URL. Browsers and search engines update their stored location.
- **302 Found** – A temporary redirection. The resource has moved *for now*, but could return later.
- **304 Not Modified** – The resource hasn't changed since your last request. Useful for caching.

🟥 400–499: *Client Errors*

The problem is on **your side** — the browser or client made a bad request.

- **400 Bad Request** – The server couldn't understand your request (malformed syntax, missing data, etc.).
- **401 Unauthorized** – You're not logged in or didn't send valid credentials.
- **403 Forbidden** – You are **not allowed** to access this resource, even if you're logged in.
- **404 Not Found** – The page or resource doesn't exist.
- **405 Method Not Allowed** – You used the wrong HTTP method (e.g., sending a GET when the server expects POST).

🛑 500–599: *Server Errors*

The problem is on **the server's side**.

- **500 Internal Server Error** – Something went wrong, but the server isn't sure what.
- **502 Bad Gateway** – The server is acting as a gateway and got a bad response from another server.
- **503 Service Unavailable** – The server is down or too busy.
- **504 Gateway Timeout** – The upstream server didn't respond in time.

💡 Use Case Examples

Situation	Status Code
User signs up and account is created	**201 Created**
Visiting a broken link	**404 Not Found**
API fails to parse your JSON input	**400 Bad Request**
Server is undergoing updates	**503 Service Unavailable**
You forgot to log in before accessing a page	**401 Unauthorized**
You try to GET a form that only accepts POST	**405 Method Not Allowed**

🔍 What Are HTTP Headers?

HTTP headers are **key-value pairs** sent between the client (usually your browser) and the server. They give **extra information** about the request or the response — like the content type, size, language, and how to handle cookies or caching.

They are not required for every request, but without them, websites would not behave as expected or even render correctly.

📨 Request Headers (Client → Server)

These are sent **by the client** (e.g., your browser or a tool like curl or Postman) when making a request.

🏷 Host

- **Purpose:** Tells the server which website you're trying to access, especially useful when a server hosts **multiple domains**.
- **Example:**
  Host: tryhackme.com
- 💡 Without this, the server might return the wrong site (or a default page).

🧭 **User-Agent**

- **Purpose:** Identifies the client software (browser, version, OS).
- **Example:**
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)...
- 💡 Servers can tailor content based on this (like a mobile version for phones).

📦 **Content-Length**

- **Purpose:** Tells the server how many bytes of data are being sent (usually with POST, PUT requests).
- **Example:**
  Content-Length: 342
- 💡 Helps the server avoid incomplete uploads or data truncation.

🌀 **Accept-Encoding**

- **Purpose:** Tells the server what types of **compression** your browser can handle.
- **Example:**
  Accept-Encoding: gzip, deflate, br
- 💡 The server will respond with compressed content to save bandwidth and improve speed.

🍪 **Cookie**

- **Purpose:** Sends stored cookies back to the server (e.g., session IDs, login tokens).
- **Example:**
  Cookie: sessionid=abc123; theme=dark
- 💡 Essential for maintaining sessions across requests.

📬 **Response Headers (Server → Client)**

These are sent **by the server** after receiving a request, to give context about the returned data.

🍪 **Set-Cookie**

- **Purpose:** Tells the browser to store a cookie.
- **Example:**
  Set-Cookie: sessionid=abc123; Path=/; HttpOnly

- 💡 These cookies will be sent back on future requests to maintain login or preferences.

🗄️ **Cache-Control**

- **Purpose:** Instructs the browser how and for how long to **store (cache)** the response.
- **Example:**
  Cache-Control: max-age=3600 *(cache for 1 hour)*
  Cache-Control: no-store *(don't save this at all)*
- 💡 Helps speed up page load and reduce server load.

📄 **Content-Type**

- **Purpose:** Specifies the **MIME type** of the data being returned so the browser knows how to handle it.
- **Examples:**
  - Content-Type: text/html (regular webpage)
  - Content-Type: application/json (API response)
  - Content-Type: image/png (image)
- 💡 If this is wrong, the browser might misinterpret the content (e.g., try to render a JSON file as plain text).

🗜️ **Content-Encoding**

- **Purpose:** Tells the browser which **compression method** was used.
- **Example:**
  Content-Encoding: gzip
- 💡 The browser will decompress this before rendering the content.

🍪 **What Are Cookies in HTTP?**

Cookies are small pieces of data that a website stores on your browser. They help **maintain continuity** between requests, since HTTP by itself is **stateless** — it doesn't remember who you are from one page to the next.

💡 **Why Use Cookies?**

Without cookies, every time you visit a new page, the server would treat you as a **completely new visitor**. Cookies fix this by carrying data between requests.

Common uses:

- ✅ **Authentication** – keeping you logged in after entering your credentials.

- 🌐 **Session tracking** – remembering your activity or shopping cart on an e-commerce site.
- ⚙️ **Preferences** – saving things like language settings or dark mode.

🧭 Cookie Lifecycle

1. **You visit a website** for the first time.
2. **The server responds with a Set-Cookie header** to store a cookie in your browser.

Set-Cookie: session_id=abc123; Path=/; HttpOnly

3. Your browser **saves** that cookie.
4. On future requests to the same domain, your browser **automatically sends the cookie** back:

Cookie: session_id=abc123

5. The server reads the cookie to **identify your session or preferences**.

🔐 Cookie Format

Cookies look like this in headers:

From the Server (Response Header):

Set-Cookie: name=value; Path=/; Expires=Wed, 21 Jun 2025 12:00:00 GMT; HttpOnly; Secure

- name=value – The actual data being stored.
- Path=/ – Makes the cookie accessible for the entire site.
- Expires= – Tells the browser when to delete the cookie.
- HttpOnly – Makes it inaccessible to JavaScript (security feature).
- Secure – Only sent over HTTPS connections.

🍼 Viewing Cookies in Your Browser

You can inspect cookies using your browser's **Developer Tools**:

📱 For Chrome/Firefox/Edge:

1. **Right-click** on the webpage and choose **Inspect** (or press F12).
2. Go to the **Network** tab.
3. Refresh the page so the requests show up.
4. Click on the first request (usually the root /).
5. Find the **Cookies** tab or section:
    - Under **Request Headers**, you'll see what cookies your browser sent (Cookie:).

- Under **Response Headers**, you'll see any new cookies the server sent (Set-Cookie:).

Alternatively:

- In **Chrome**, go to **Application > Storage > Cookies** to view all current cookies for the site.

🔐 **Are Cookies Secure?**

Cookies themselves are **not encrypted**. Their security depends on:

- The Secure flag → Ensures cookies only travel over HTTPS.
- The HttpOnly flag → Prevents JavaScript from accessing the cookie.
- Session management on the server side → The server should store sensitive info securely and use secure session tokens.

👉 **Cookies should never store passwords or personal sensitive data in plain text.**

How websites work

When you visit a website, your browser (*like Safari or Google Chrome*) makes a request to a web server asking for information about the page you're visiting. It will respond with data that your browser uses to show you the page; a web server is just a dedicated computer somewhere else in the world that handles your requests.

There are two major components that make up a website:

1. Front End (Client-Side) - the way your browser renders a website.
2. Back End (Server-Side) - a server that processes your request and returns a response.

There are many other processes involved in your browser making a request to a web server, but for now, you just need to understand that you make a request to a server, and it responds with data your browser uses to render information to you.

Websites are primarily created using:

- HTML, to build websites and define their structure
- CSS, to make websites look pretty by adding styling options
- JavaScript, implement complex features on pages using interactivity

Hyper**T**ext **M**arkup **L**anguage (HTML) is the language websites are written in. Elements (also known as tags) are the building blocks of HTML pages and tells the browser how to display content. The code snippet below shows a simple HTML document, the structure of which is the same for every website:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>
        <h1>Example Heading</h1>
        <p>Example paragraph..</p>
    </body>
</html>
```

The HTML structure (as shown in the screenshot) has the following components:

- The <!DOCTYPE html> defines that the page is a HTML5 document. This helps with standardisation across different browsers and tells the browser to use HTML5 to interpret the page.
- The <html> element is the root element of the HTML page - all other elements come after this element.
- The <head> element contains information about the page (such as the page title)
- The <body> element defines the HTML document's body; only content inside of the body is shown in the browser.
- The <h1> element defines a large heading
- The <p> element defines a paragraph
- There are many other elements (tags) used for different purposes. For example, there are tags for buttons (<button>), images (<img>), lists, and much more.

Tags can contain attributes such as the class attribute which can be used to style an element (e.g. make the tag a different color) <p class="bold-text">, or the *src* attribute which is used on images to specify the location of an image: <img src="img/cat.jpg">.An element can have multiple attributes each with its own unique purpose, e.g., <p attribute1="value1" attribute2="value2">.

Elements can also have an id attribute (<p id="example">), which is unique to the element. Unlike the class attribute, where multiple elements can use the same class, an element must have different id's to identify them uniquely. Element id's are used for styling and to identify it by JavaScript.

You can view the HTML of any website by right-clicking and selecting "View Page Source" (Chrome) / "Show Page Source" (Safari).

JavaScript

JavaScript (JS) is one of the most popular coding languages in the world and allows pages to become interactive. HTML is used to create the website structure and content, while JavaScript is used to control the functionality of web pages - without JavaScript, a page would not have interactive elements and would always be static. JS can dynamically update the page in real-time, giving functionality to change the

style of a button when a particular event on the page occurs (such as when a user clicks a button) or to display moving animations.

JavaScript is added within the page source code and can be either loaded within <script> tags or can be included remotely with the src attribute: <script src="/location/of/javascript_file.js"></script>

The following JavaScript code finds a HTML element on the page with the id of "demo" and changes the element's contents to "Hack the Planet" : document.getElementById("demo").innerHTML = "Hack the Planet";

HTML elements can also have events, such as "onclick" or "onhover" that execute JavaScript when the event occurs. The following code changes the text of the element with the demo ID to Button Clicked: <button onclick='document.getElementById("demo").innerHTML = "Button Clicked";'>Click Me!</button> - onclick events can also be defined inside the JavaScript script tags, and not on elements directly.

Sensitive Data Exposure

Sensitive Data Exposure occurs when a website doesn't properly protect (or remove) sensitive clear-text information to the end-user; usually found in a site's frontend source code.

We now know that websites are built using many HTML elements (tags), all of which we can see simply by "viewing the page source". A website developer may have forgotten to remove login credentials, hidden links to private parts of the website or other sensitive data shown in HTML or JavaScript.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Fake Website</title>
    </head>
    <body>
       <form>
          <input type='text' name='username'>
          <input type='password' name='password'>
          <button>Login</button>
          <!-- TODO: remove test credentials admin:password123 -->
        </form>
    </body>
</html>
```

Sensitive information can be potentially leveraged to further an attacker's access within different parts of a web application. For example, there could be HTML comments with temporary login credentials, and if you viewed the page's source code and found this, you could use these credentials to log in elsewhere on the application (or worse, used to access other backend components of the site).

Whenever you're assessing a web application for security issues, one of the first things you should do is review the page source code to see if you can find any exposed login credentials or hidden links.

🔍 What is HTML Injection?

HTML Injection happens when a website takes **what a user types** (like in a form or a comment box) and shows it directly on the page **without checking or cleaning it first**. If the site doesn't filter or sanitize the input, a user can sneak in HTML code instead of just regular text.

💡 Why is this a problem?

Web browsers are designed to read and display HTML. So if someone enters something like a heading or a link instead of their name or comment, the browser will **display it as actual HTML**, not just plain text.

For example, instead of showing "Hi Shya," the website might display a big bold heading that says "Hacked!" if the user entered special HTML.

Even worse, an attacker could inject scripts that **run automatically in other people's browsers** when they visit the site. This is how some **cross-site scripting (XSS)** attacks start — and they can be used to steal data, redirect users, or deface a site.

🛡️ How do websites protect against it?

To stay secure, websites need to **sanitize** any user input — this means:

- Removing or disabling any HTML or script tags from the input.
- Making sure anything the user types is treated as **text**, not as code.
- Avoiding risky ways of showing user input on the page.

🎯 Why does this matter?

Because **HTTP is stateless**, the server doesn't remember you from one page to another unless you use tools like cookies — and it also means it relies heavily on what the browser sends. If a user can control part of the page by injecting their own code, they can **change how the page looks**, **trick other users**, or even **steal sensitive data**.

In short: HTML Injection is like giving a stranger a microphone without checking what they're about to say. If you don't filter what they say, they can cause chaos. Websites should never trust what users input — and always clean it before using it.

Putting It All Together

From the previous modules, you'll have learned that quite a lot of things go on behind the scenes when you request a webpage in your browser.

To summarise, when you request a website, your computer needs to know the server's IP address it needs to talk to; for this, it uses DNS. Your computer then talks to the web server using a special set of commands called the HTTP protocol; the webserver then returns HTML, JavaScript, CSS, Images, etc., which your browser then uses to correctly format and display the website to you.

🌐 Load Balancer

A **load balancer** is like a traffic manager for busy websites. Instead of sending every visitor to the same server, it spreads people out across several servers to:

- **Avoid crashing from too much traffic**
- **Keep the site working even if one server goes down**

It decides which server to send you to using smart rules — like taking turns (round-robin) or choosing the server with the least work.

🌍 CDN (Content Delivery Network)

A **CDN** helps speed things up. Instead of loading images or videos from the original website, it stores copies of them all over the world. When you open a site, you get the files from the **closest server to you**, which makes everything load faster and reduces pressure on the main website.

💾 Database

Websites often need a place to store information — like user accounts, messages, or product lists. This is where **databases** come in. The server talks to the database to **save and get** the information it needs. There are many kinds, but all help with storing and organizing data.

🔐 WAF (Web Application Firewall)

A **WAF** protects websites from attacks. It checks your request **before it even reaches** the website to block:

- Hackers trying to break in
- Bots pretending to be users
- Too many requests in a short time (flooding)

If it thinks your request looks dangerous, it blocks it and keeps the site safe.

These tools all work together behind the scenes to keep websites **fast**, **reliable**, and **secure** — especially when millions of people are using them at the same time.

🌐 What Is a Web Server?

A **web server** is a program (or software) that listens for incoming internet requests and sends back web content like websites, images, or videos using the **HTTP protocol**.

Some popular web servers include:

- **Apache**

- **Nginx**
- **IIS** (for Windows)
- **NodeJS** (can act as one)

The server stores website files in a **root folder**. For example:

- On Linux: /var/www/html
- On Windows (IIS): C:\inetpub\wwwroot

So if you visit a page like http://example.com/picture.jpg, the server checks that folder and gives you the picture.jpg file.

🏠 Virtual Hosts

Web servers can host **more than one website** at the same time — this is done using **virtual hosts**.

The server looks at the domain name you typed (from the request header) and matches it to a configuration file that points to the right folder on the server.

For example:

- one.com → /var/www/website_one
- two.com → /var/www/website_two

If there's no match, it just shows a default website.

📁 Static vs Dynamic Content

- **Static content** = Doesn't change. Examples: images, styles (CSS), scripts (JS), or even simple HTML files.
- **Dynamic content** = Changes based on input or updates. For example:
    - A news site that updates with new articles.
    - A search page that shows different results depending on what you type.

Dynamic content is generated using **backend code** — which runs on the server side.

⚙️ Backend and Scripting Languages

The **backend** is the "brain" behind a website. It's where the logic runs, like:

- Reading data from a database
- Processing forms or search queries
- Authenticating users

Common backend languages include:

- **PHP**
- **Python**
- **NodeJS**
- **Ruby**
- **Perl**

The **frontend** is what you actually see in the browser — the result of what the backend sends after doing all the behind-the-scenes work.

🔐 **Why It Matters**

Because the backend controls so much and users can't see how it works, **security becomes very important**. If a site doesn't filter or sanitize input correctly, attackers can abuse the backend to steal data, bypass login systems, or worse.

LINUX

Linux Fundamentals Part 1 – Summary for Notes

🔹 **What is Linux?**

Linux is an open-source Unix-like operating system widely used in servers, desktops, and embedded systems. It consists of a **kernel** and various **utilities** and offers flexibility, security, and control.

It's fair to say that Linux is a lot more intimidating to approach than Operating System's (OSs) such as Windows. Both variants have their own advantages and disadvantages. For example, Linux is considerably much more lightweight and you'd be surprised to know that there's a good chance you've used Linux in some form or another every day! Linux powers things such as:

- Websites that you visit
- Car entertainment/control panels
- Point of Sale (PoS) systems such as checkout tills and registers in shops
- Critical infrastructures such as traffic light controllers or industrial sensors

---

As we previously discussed, a large selling point of using Operating Systems such as Ubuntu is how lightweight they can be. This, of course, doesn't come without its disadvantages, where for example, often there is no GUI (Graphical User Interface) or what is also known as a desktop environment that we can use to interact with the machine (unless it has been installed). A large part of interacting with these systems is using the "Terminal".

The "Terminal" is purely text-based and is intimidating at first. However, if we break down some of the commands, after some time, you quickly become familiar with using the terminal!

1. Basic Terminal Commands

The **terminal** is a text-based interface to interact with the operating system.

- **pwd** – Print current directory.
- **ls** – List directory contents.
- **cd** – Change directory.
- **file** – Determine file type.
- **cat** – Display file contents.
- **clear** – Clear terminal screen.
- **echo** – Print to terminal or file.

The **pwd** command shows your current working directory, helping you know where you are in the filesystem. **ls** lists the contents of a directory, showing files and folders. **cd** is used to change from one directory to another. The **file** command identifies the type of a file (e.g., text, binary, image). **cat** displays the contents of a file directly in the terminal. **clear** clears all the previous output from the terminal screen. Lastly, echo prints text or variables to the terminal or can be used to write to a file.

---

2. Working with Directories and Files

- **mkdir** – Create a directory.
- **touch** – Create a new empty file.
- **cp** – Copy files or directories.
- **mv** – Move or rename files/directories.
- **rm** – Remove files/directories.
- **rmdir** – Remove empty directories.

---

3. Viewing and Manipulating Files

- **less** – View file content one page at a time.
- **head/tail** – View the first or last lines of a file.

- **grep** – Search for patterns in files.
- **wc** – Count lines, words, characters.
- **sort** – Sort lines in a file.
- **cut** – Extract columns from text files.

---

4. Permissions and Ownership

- Files have three permission groups: **owner**, **group**, and **others**.
- Permissions: **r (read)**, **w (write)**, **x (execute)**.
- Use **ls -l** to view permissions.
- **chmod** – Change file permissions (e.g., chmod +x script.sh).
- **chown** – Change file owner.
- **chgrp** – Change file group.

---

5. Getting Help

- **man** – Manual for a command (e.g., man ls).
- **--help** – Quick help for commands.
- Try command --help or man command to learn command syntax.

---

6. Wildcards and Searching

- Wildcards: * (matches any characters), ? (matches one character).
- Use with commands like ls, cp, or rm to target multiple files.

Operator "&"

This operator allows us to execute commands in the background. For example, let's say we want to copy a large file. This will obviously take quite a long time and will leave us unable to do anything else until the file successfully copies.

The "&" shell operator allows us to execute a command and have it run in the background (such as this file copy) allowing us to do other things!

Operator "&&"

This shell operator is a bit misleading in the sense of how familiar is to its partner "&". Unlike the "&" operator, we can use "&&" to make a list of commands to run for example command1 && command2. However, it's worth noting that command2 will only run if command1 was successful.

Operator ">"

This operator is what's known as an output redirector. What this essentially means is that we take the output from a command we run and send that output to somewhere else.

A great example of this is redirecting the output of the echo command that we learned in Task 4. Of course, running something such as echo howdy will return "howdy" back to our terminal — that isn't super useful. What we can do instead, is redirect "howdy" to something such as a new file!

Operator ">>"

This operator is also an output redirector like in the previous operator (>) we discussed. However, what makes this operator different is that rather than overwriting any contents within a file, for example, it instead just puts the output at the end.

Following on with our previous example where we have the file "welcome" that has the contents of "hey". If were to use echo to add "hello" to the file using the > operator, the file will now only have "hello" and not "hey".

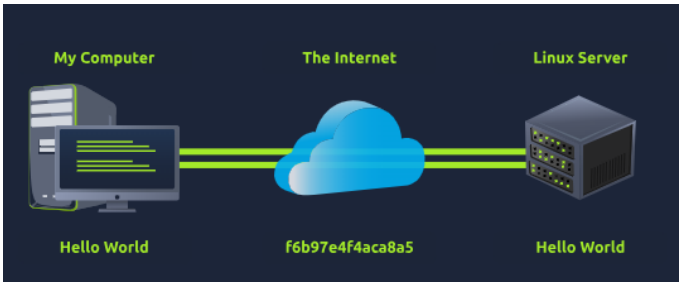
7. Command Chaining and Pipes

- **;** – Run multiple commands sequentially.
- **&&** – Run second command only if first succeeds.
- **||** – Run second command only if first fails.
- **|** – Pipe output from one command to another (e.g., ls | grep file).

The | symbol, called a pipe, is used in the terminal to connect two commands together. It takes the output from the command on the left side and passes it as input to the command on the right side. For example, ls | grep file means "list all files and then search through that list for anything with the word 'file' in it." This lets you combine commands to do more complex tasks in one line, without needing to save intermediate results.

SSH - Secure Shell

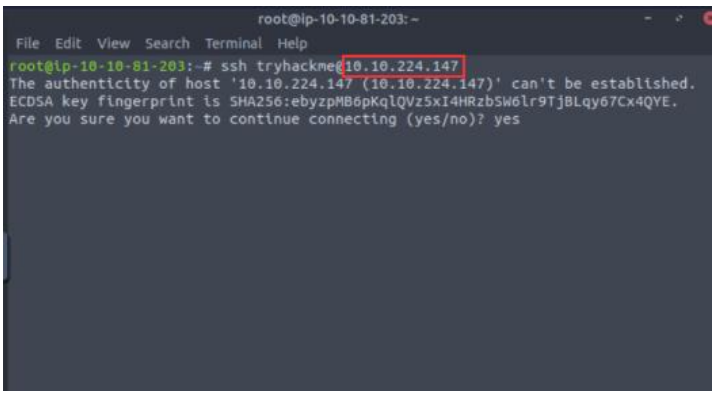
SSH (Secure Shell) is a protocol used to securely connect to remote machines over a network. It encrypts all data sent between the client and server, ensuring confidentiality and protection from attackers. With SSH, you can remotely execute commands on another device as if you were physically present. This is commonly used by system administrators and developers for managing servers. The encryption ensures that sensitive information, like login credentials and command output, remains private during transmission.



How to Use SSH

The command to do so is ssh and then the username of the account, @ the IP address of the machine.

Ex: ssh shya@192.10.10.10



Note: *When you type a password into an ssh login prompt there is no visible feedback -- you will not be able to see any text or symbols appear as you type the password. It is still working, however, so just type the password and press enter to login.*

Flags and Switches

A majority of commands allow for arguments to be provided. These arguments are identified by a hyphen and a certain keyword known as flags or switches.

When you run a command like ls, it performs its **default behavior**—listing visible files. But you can modify how commands work using **flags or switches**, which are preceded by a hyphen. For example, using -a with ls (ls -a) makes it show **hidden files** (those starting with a dot).

To discover which flags a command supports, use:

- command --help — Quick overview of flags with descriptions (e.g., ls --help shows options like -a, -l, -h, etc.).
- man command — Access the full **manual page**, containing detailed documentation, usage examples, and all supported options (e.g., man ls).

In summary:

- **Flags** (like -a, -l) allow you to customize command behavior.
- --help gives you quick guidance.
- man pages provide in-depth documentation for commands and their options.

We covered some of the most fundamental commands when interacting with the filesystem on the Linux machine. For example, we covered how to list and find the contents of folders using ls and find and navigating the filesystem using cd.

In this task, we're going to learn some more commands for interacting with the filesystem to allow us to:

- create files and folders
- move files and folders

- delete files and folders

More specifically, the following commands:

Command	Full Name	Purpose
touch	*Touch*	Creates a new empty file. Useful for quickly generating placeholder files.
mkdir	*Make Directory*	Creates a new folder (directory).
cp	*Copy*	Copies files or directories from one location to another.
mv	*Move*	Moves or renames files or directories.
rm	*Remove*	Deletes files or folders. Use -r for removing directories recursively.
file	*File*	Tells you the type of a file (e.g., text, image, binary).

Pro Tip:
You can use full paths when working with these commands. For example:

cp /home/user/docs/file.txt /home/user/backup/

This is especially helpful when working in deep or different directory structures.

Creating Files and Folders (touch, mkdir)

Creating files and folders on Linux is a simple process. First, we'll cover creating a file. The touch command takes exactly one argument -- the name we want to give the file we create. For example, we can create the file "note" by using touch note. It's worth noting that touch simply creates a blank file. You would need to use commands like echo or text editors such as nano to add content to the blank file.

Using touch to create a new file:

tryhackme@linux2:~$ touch note

tryhackme@linux2:~$ ls

folder1 note

This is a similar process for making a folder, which just involves using the mkdir command and again providing the name that we want to assign to the directory. For example, creating the directory "mydirectory" using mkdir mydirectory.

Creating a new directory (Folder) with mkdir:

tryhackme@linux2:~$ mkdir mydirectory

tryhackme@linux2:~$ ls

folder1 mydirectory note

Removing Files and Folders (rm)

rm is extraordinary out of the commands that we've covered so far. You can simply remove files by using rm. However, you need to provide the -R switch alongside the name of the directory you wish to remove.

Using rm to remove a file:

tryhackme@linux2:~$ rm note

tryhackme@linux2:~$ ls

folder1 mydirectory

Using rm recursively to remove a directory

tryhackme@linux2:~$ rm -R mydirectory

tryhackme@linux2:~$ ls

folder1

🗑️ **What does rm do?**

rm means **remove** — it's a command to delete files or folders.

🗂️ **But if you want to delete a *folder* with files inside it?**

You can't just type rm foldername, because that only works for empty folders or single files.

✅ **So you use the recursive option:**

rm -r foldername

- The -r (or --recursive) means:
  👉 **"Go inside the folder and delete everything in it too."**

So it's like saying:

"Remove this folder **and everything inside it**, even other folders or files."

⚠️ **Be careful!**

- Once you run this, **the folder and all its contents are gone**.
- There's no "undo" in the terminal.

🔧 **Example:**

If you have a folder called old_photos with a bunch of files inside:

rm -r old_photos

That will delete the **whole folder** and all the photos inside.

Copying and Moving Files and Folders (cp, mv)

Copying and moving files is an important functionality on a Linux machine. Starting with cp, this command takes two arguments:

1. the name of the existing file

2. the name we wish to assign to the new file when copying

cp copies the entire contents of the existing file into the new file. In the screenshot below, we are copying "note" to "note2".

Using cp to copy a file

tryhackme@linux2:~$ cp note note2

tryhackme@linux2:~$ ls

folder1 note note2

Moving a file takes two arguments, just like the cp command. However, rather than copying and/or creating a new file, mv will merge or modify the second file that we provide as an argument. Not only can you use mv to move a file to a new folder, but you can also use mv to rename a file or folder. For example, in the screenshot below, we are renaming the file "note2" to be named "note3". "note3" will now have the contents of "note2".

Using mv to move a file

tryhackme@linux2:~$ mv note2 note3

tryhackme@linux2:~$ ls

folder1 note note3

Determining File Type

What is often misleading and often catches people out is making presumptions from files as to what their purpose or contents may be. Files usually have what's known as an extension to make this easier. For example, text files usually have an extension of ".txt". But this is not necessary.

So far, the files we have used in our examples haven't had an extension. Without knowing the context of why the file is there -- we don't really know its purpose. Enter the file command. This command takes one argument. For example, we'll use file to confirm whether or not the "note" file in our examples is indeed a text file, like so file note.

Using file to determine the contents of a file

tryhackme@linux2:~$ file note

note: ASCII text

🔐 File and Folder Permissions in Linux

In Linux, not everyone can do everything with every file or folder. Some users have full access, and others may have no access at all. To figure out who can do what, we can use this command:

ls -l

This shows you:

- Who owns the file (the user)
- What group it belongs to
- And what kind of access they have (read, write, or execute)

🧍 User vs Group Permissions

Each file or folder has permissions for:

- The owner (user)
- The group it belongs to
- Everyone else

Each of these can be allowed to:

- Read (look at the file)
- Write (edit the file)
- Execute (run it like a program)

For example, this output:

-rw-r--r--

Means:

- The owner can read and write
- The group can only read
- Others can only read

🔄 Switching Between Users

Sometimes you need to become another user to do certain tasks. You do this using:

su username

This means “Switch User.”

You’ll be asked for that user’s password.

🏠 What's the Difference Between su and su -l?

- su user2: Switches to user2, but keeps you in the current folder.
- su -l user2: Switches to user2 and acts like you just logged in — puts you in user2’s home folder and loads their environment.

🧠 Why This Matters

- You need the right permissions to read or edit a file.
- You can switch users to access different files (if you have the password).
- This is very common in Linux systems — especially in cybersecurity and server management.

/etc

This root directory is one of the most important root directories on your system. The etc folder (short for etcetera) is a commonplace location to store system files that are used by your operating system.

For example, the sudoers file highlighted in the screenshot below contains a list of the users & groups that have permission to run sudo or a set of commands as the root user.

Also highlighted below are the "passwd" and "shadow" files. These two files are special for Linux as they show how your system stores the passwords for each user in encrypted formatting called sha512.

Some notable contents of the /etc directory

```
tryhackme@linux2:/etc$ ls
shadow passwd sudoers sudoers.d
```

/var

The "/var" directory, with "var" being short for variable data, is one of the main root folders found on a Linux install. This folder stores data that is frequently accessed or written by services or applications running on the system. For example, log files from running services and applications are written here (**/var/log**), or other data that is not necessarily associated with a specific user (i.e., databases and the like).

```
tryhackme@linux2:/var$ ls
backups log opt tmp
```

/root

Unlike the **/home** directory, the **/root** folder is actually the home for the "root" system user. There isn't anything more to this folder other than just understanding that this is the home directory for the "root" user. But, it is worth a mention as the logical presumption is that this user would have their data in a directory such as "**/home/root**" by default.

/tmp (Temporary) (Volatile)

This is a unique root directory found on a Linux install. Short for "**temporary**", the /tmp directory is volatile and is used to store data that is only needed to be accessed once or twice. Similar to the memory on your computer, once the computer is restarted, the contents of this folder are cleared out.

What's useful for us in pentesting is that any user can write to this folder by default. Meaning once we have access to a machine, it serves as a good place to store things like our enumeration scripts.

Throughout the series so far, we have only stored text in files using a combination of the echo command and the pipe operators (> and >>). This isn't an efficient way to handle data when you're working with files with multiple lines and the sorts!

Introducing terminal text editors

There are a few options that you can use, all with a variety of friendliness and utility. This task is going to introduce you to nano but also show you an alternative named VIM .

Nano

It is easy to get started with Nano! To create or edit a file using nano, we simply use nano filename -- replacing "filename" with the name of the file you wish to edit.

Once we press enter to execute the command, nano will launch! Where we can just begin to start entering or modifying our text. You can navigate each line using the "up" and "down" arrow keys or start a new line using the "Enter" key on your keyboard.

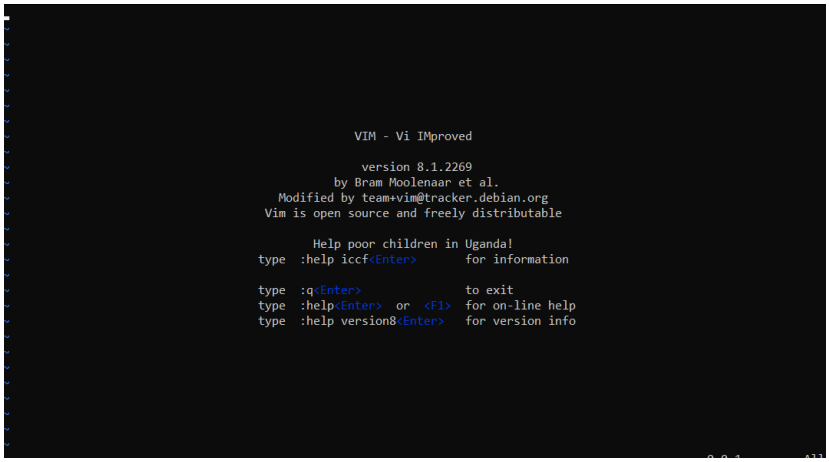
Nano has a few features that are easy to remember & covers the most general things you would want out of a text editor, including:

- Searching for text
- Copying and Pasting
- Jumping to a line number
- Finding out what line number you are on

You can use these features of nano by pressing the "**Ctrl**" key (which is represented as an ^ on Linux) and a corresponding letter. For example, to exit, we would want to press "**Ctrl**" and "**X**" to exit Nano.

VIM

VIM is a much more advanced text editor. Whilst you're not expected to know all advanced features, it's helpful to mention it for powering up your Linux skills.



Some of VIM's benefits, albeit taking a much longer time to become familiar with, includes:

- Customisable - you can modify the keyboard shortcuts to be of your choosing
- Syntax Highlighting - this is useful if you are writing or maintaining code, making it a popular choice for software developers
- VIM works on all terminals where nano may not be installed
- There are a lot of resources such as cheatsheets, tutorials, and the sorts available to you use.

🧲 Downloading Files with wget

A pretty fundamental feature of computing is the ability to transfer files. For example, you may want to download a program, a script, or even a picture. Thankfully for us, there are multiple ways in which we can retrieve these files.

We're going to cover the use of wget . This command allows us to download files from the web via HTTP -- as if you were accessing the file in your browser. We simply need to provide the address of the resource that we wish to download. For example, if I wanted to download a file named "myfile.txt" onto my machine, assuming I knew the web address it -- it would look something like this:

- wget is a command used to **download files from the internet**.
- You just give it a link (URL), and it grabs the file for you.
- Example:

wget https://example.com/file.txt

This saves file.txt to your current folder.

🔐 Transferring Files with scp (Secure Copy)

Secure copy, or SCP, is just that -- a means of securely copying files. Unlike the regular cp command, this command allows you to transfer files between two computers using the SSH protocol to provide both authentication and encryption.

- scp lets you **copy files between your computer and another computer over SSH**.
- It encrypts the transfer for safety.
- The command format is always:

scp [source] [destination]

📤 Example: Copy a file from *your* computer → *remote* machine:

scp important.txt ubuntu@192.168.1.30:/home/ubuntu/transferred.txt

📥 Example: Copy a file from *remote* machine → *your* computer:

scp ubuntu@192.168.1.30:/home/ubuntu/documents.txt notes.txt

🌐 Hosting Files with a Simple Web Server (python3 -m http.server)

Ubuntu machines come pre-packaged with python3. Python helpfully provides a lightweight and easy-to-use module called "HTTPServer". This module turns your computer into a quick and easy web server that you can use to serve your own files, where they can then be downloaded by another computing using commands such as curl and wget.

Python3's "HTTPServer" will serve the files in the directory where you run the command, but this can be changed by providing options that can be found within the manual pages. Simply, all we need to do is run python3 -m http.server in the terminal to start the module! In the snippet below, we are serving from a directory called "webserver", which has a single named "file".

- Python can turn any folder into a **temporary web server**.
- Run this in the folder where your file is:

python3 -m http.server

- By default, it runs on **port 8000** and shares everything in that folder.

🧲 Then, to download from another machine:

wget http://<IP-ADDRESS>:8000/filename

- Replace <IP-ADDRESS> with the IP of the server.
- You need to **keep the server terminal open**, and run wget in a **separate terminal**.

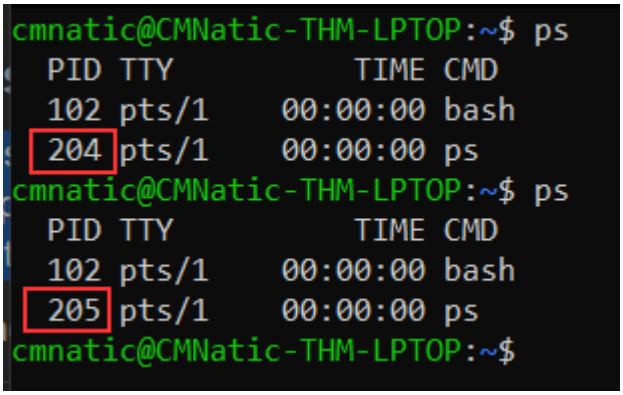
⚠️ **Notes:**

- You must know the **exact file name** to download it with wget.
- This method is great for quick file sharing in labs or over your network.

Processes are the programs that are running on your machine. They are managed by the kernel, where each process will have an ID associated with it, also known as its PID. The PID increments for the order In which the process starts. I.e. the 60th process will have a PID of 60.

Viewing Processes

We can use the friendly ps command to provide a list of the running processes as our user's session and some additional information such as its status code, the session that is running it, how much usage time of the CPU it is using, and the name of the actual program or command that is being executed:



Note how in the screenshot above, the second process ps has a PID of 204, and then in the command below it, this is then incremented to 205. In the context of **processes** in Linux (or any operating system), **"incremented"** means that the **Process ID (PID)** number is increased by one (or more) each time a **new process** is created.

🔍 **ps aux – Viewing All Running Processes**

- The ps command shows running processes. By default, it only shows **your** processes in the current session.
- If you run ps aux, it shows:
  - **All users' processes** (including the system or "root").
  - **Processes not started by a terminal** (like background/system services).
  - Details like who started the process, how much CPU or memory it uses, and what command was run.

👀 **Example:**

ps aux

You might see processes run by:

- root (the system)

- cmnatic (a normal user)

📈 top – Live View of Processes

- top is a powerful command that shows **live updates** of your system's activity.
- It automatically refreshes every 10 seconds.
- It shows the most resource-hungry processes at the top.
- You can scroll using arrow keys to explore all processes.

🧠 Useful for:

- Monitoring system performance in real time.
- Finding which process is using too much CPU or memory.
- The top command in Linux is very similar to the **Task Manager** in Windows.

🔧 Killing or Controlling Processes in Linux

When a program or process is running, you can send it a **signal** to tell it what to do — like stop, pause, or exit.

You use the command:

kill <PID>

Where PID is the **Process ID** of the running program you want to target.

🔬 Common Signals You Can Send

1. **SIGTERM** (kill <PID>)
    - **Meaning**: Politely asks the process to shut down.
    - **Use Case**: Safest way to stop a process — it gives the process time to **clean up** (e.g., save data, close files).
    - **Signal Number**: 15
    - **Example**: kill 1337
2. **SIGKILL** (kill -9 <PID>)
    - **Meaning**: Forcibly kills the process **immediately** — no chance to clean up.
    - **Use Case**: When a process is **frozen or unresponsive** and won't stop with SIGTERM.
    - **Signal Number**: 9
    - **Example**: kill -9 1337
3. **SIGSTOP** (kill -STOP <PID>)

- **Meaning**: **Pauses** the process.
- **Use Case**: Temporarily suspend a process without killing it (you can resume it later with SIGCONT).
- **Signal Number**: 19
- **Example**: kill -STOP 1337

🛠️ How Do Processes Start?

Let's start off by talking about namespaces. The Operating System (OS) uses namespaces to ultimately split up the resources available on the computer to (such as CPU, RAM and priority) processes. Think of it as splitting your computer up into slices -- similar to a cake. Processes within that slice will have access to a certain amount of computing power, however, it will be a small portion of what is actually available to every process overall.

Namespaces are great for security as it is a way of isolating processes from another -- only those that are in the same namespace will be able to see each other.

We previously talked about how PID works, and this is where it comes into play. The process with an ID of 0 is a process that is started when the system boots. This process is the system's init on Ubuntu, such as **systemd**, which is used to provide a way of managing a user's processes and sits in between the operating system and the user.

For example, once a system boots and it initialises, **systemd** is one of the first processes that are started. Any program or piece of software that we want to start will start as what's known as a child process of **systemd**. This means that it is controlled by **systemd**, but will run as its own process (although sharing the resources from **systemd**) to make it easier for us to identify and the likes.

1. **Namespaces = Private Workspaces**
   Imagine each department in the office has its own private space with certain tools (CPU, RAM, etc). These are **namespaces**.
   - Only workers in the same department (namespace) can see and interact with each other.
   - It keeps things **secure** and **organized**.
2. **The First Worker: systemd**
   When the office opens (your system boots), the **first process** to start is systemd.
   - This is like the **office manager** who keeps track of all workers (processes).
   - Every other process (like your browser, file manager, servers) is like a **child** of this manager.

🔁 Starting Services When Your System Boots

Some programs, like web servers or database tools, need to start automatically when your computer turns on. These are important programs that system administrators usually set to launch right at startup.

For example, let's say we want to use the Apache web server. First, we can start it manually. Then, we can tell the system to always start it automatically when the computer boots.

To do this, we use a command called systemctl, which helps us manage these types of services.

Some "workers" (like a web server or a database) need to be ready **as soon as the office opens**.

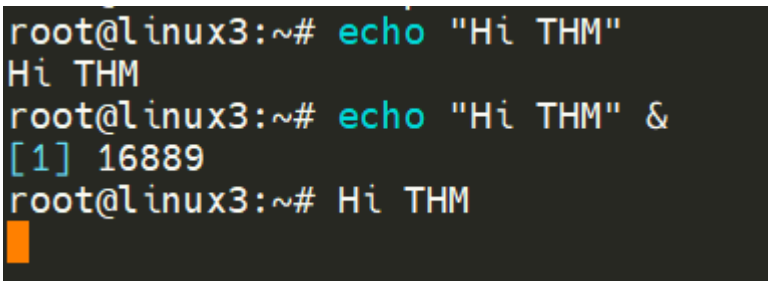
To manage these workers, we use the command:

systemctl [option] [service-name]

- systemctl start apache2 → Starts the Apache web server **now**
- systemctl stop apache2 → Stops it
- systemctl enable apache2 → Makes it start **automatically at boot**
- systemctl disable apache2 → Prevents it from auto-starting

🧠 Foreground vs Background Processes

Processes can run in two states: In the background and in the foreground. For example, commands that you run in your terminal such as "echo" or things of that sort will run in the foreground of your terminal as it is the only command provided that hasn't been told to run in the background. "Echo" is a great example as the output of echo will return to you in the foreground, but wouldn't in the background - take the screenshot below, for example.



Here we're running echo "Hi THM" , where we expect the output to be returned to us like it is at the start. But after adding the & operator to the command, we're instead just given the ID of the echo process rather than the actual output -- as it is running in the background.

This is great for commands such as copying files because it means that we can run the command in the background and continue on with whatever further commands we wish to execute (without having to wait for the file copy to finish first)

We can do the exact same when executing things like scripts -- rather than relying on the & operator, we can use Ctrl + Z on our keyboard to background a process.

Imagine you're on a phone call (process) at your desk:

- **Foreground process**: You're fully focused on that one call. You can't do anything else until it's done.
  Example: You type echo "Hello" → you see the result immediately.
- **Background process**: You put the call on speakerphone and start typing an email. You're doing both.
  Add & to the end of your command like this:

echo "Hello" &

Now the process runs **in the background**, and your terminal is free to do other things.

⏸️ Pausing a Process: Ctrl + Z

You can **pause** a process by pressing Ctrl + Z.

- This stops the process temporarily and puts it in the background.
- Great for pausing scripts or long commands without killing them.

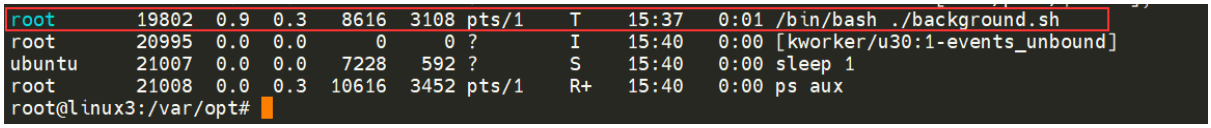
🔄 Bringing It Back: fg

Want to talk to that backgrounded worker again?

- Just type fg to bring the paused/background process **back to the foreground**.
- Now you're interacting with it again directly.

📋 Checking What's Running

Now that we have a process running in the background, for example, our script "background.sh" which can be confirmed by using the ps aux command, we can back-pedal and bring this process back to the foreground to interact with.



With our process backgrounded using either Ctrl + Z or the & operator, we can use fg to bring this back to focus like below, where we can see the fg command is being used to bring the background process back into use on the terminal, where the output of the script is now returned to us.

Use this to check all running processes:

ps aux

It's like checking the employee attendance list.

CronTabs – Cron Jobs

Users may want to schedule a certain action or task to take place after the system has booted. Take, for example, running commands, backing up files, or launching your favourite programs on, such as Spotify or Google Chrome.

We're going to be talking about the cron process, but more specifically, how we can interact with it via the use of crontabs . Crontab is one of the processes that is started during boot, which is responsible for facilitating and managing cron jobs.

A **crontab** is just a special file that tells your computer to run certain commands at specific times. It's used by a program called **cron**, which automatically reads this file and runs the commands as scheduled.

The **Windows equivalent of crontabs** is the **Task Scheduler**.

What is Task Scheduler?

Task Scheduler is a built-in Windows tool that allows you to **automate tasks** to run at specific times, on certain events, or when certain conditions are met — just like cron on Linux.

To set up a crontab job, you need to include **six parts**:

Part	What it Means
MIN	The minute (0–59) you want the task to run
HOUR	The hour (0–23) you want it to run
DOM	Day of the month (1–31)
MON	Month of the year (1–12)
DOW	Day of the week (0–7, where 0 and 7 = Sunday)
CMD	The command you want to run

Example:

Let's say you want to back up the user's **Documents** folder every **12 hours**. The crontab line would look like this:

0 */12 * * * cp -R /home/cmnatic/Documents /var/backups/

This means:

- **0** → run at minute 0
- ***/12** → every 12 hours
- The rest * * * means any day, any month, any weekday

The * (asterisk) means "every value" for that field. So if you don't care about the specific day or month, just use *.

If writing this manually seems hard, you can use tools like:

- Crontab Generator
- Cron Guru

To create or edit a crontab, use:

crontab -e

This opens your crontab in a text editor (like Nano), so you can add or change scheduled tasks easily.

Packages & Software Repositories in Linux — Easy Explanation

When developers create software for Linux, they often share it through **repositories** — special online places that store software packages. On Ubuntu (and many other Linux systems), these repositories work with the **apt** system to let users easily install and update software.

- The **apt** tool helps manage software: it installs, updates, and removes programs from your system.
- Ubuntu already has its official repositories, but you can add **extra repositories** from other sources to get more software.
- Sometimes these additional repositories are hosted by companies or communities, and they may even be closer to your location for faster downloads.

Adding & Removing Repositories

You usually add repositories using a simple command like add-apt-repository, but you can also add them manually.

Why add repositories?
Some software isn't in the default Ubuntu repositories. For example, the text editor **Sublime Text** isn't included by default, so you add its repository to get the latest version.

How does Ubuntu trust the software?
Before adding a repository, you add a **GPG key** (a kind of digital signature) to make sure the software is safe and authentic.

Example: Adding Sublime Text Repository

1. **Add the GPG key** (tells your system to trust the software source):

wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add -

2. **Add the repository** by creating a file /etc/apt/sources.list.d/sublime-text.list and adding the repository address inside it.
3. **Update apt** to recognize the new repository:

sudo apt update

4. **Install Sublime Text**:

sudo apt install sublime-text

Removing Repositories or Software

- You can remove a repository using:

sudo add-apt-repository --remove ppa:PPA_Name/ppa

- Or manually delete the repository file from /etc/apt/sources.list.d/.
- To uninstall software, use:

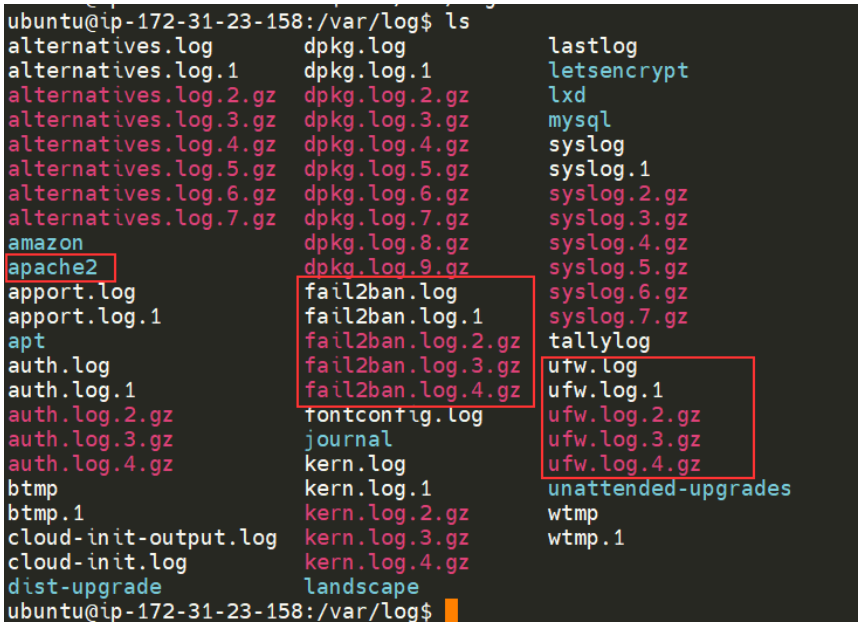
sudo apt remove [software-name]

System Logs

We briefly touched upon log files and where they can be found in Linux Fundamentals Part 1. However, let's quickly recap. Located in the /var/log directory, these files and folders contain logging information for applications and services running on your system. The Operating System (OS) has become pretty good at automatically managing these logs in a process that is known as "rotating".

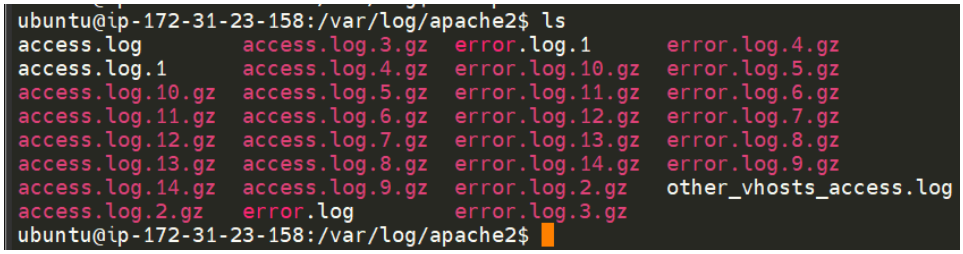
I have highlighted some logs from three services running on a Ubuntu machine:

- An Apache2 web server
- Logs for the fail2ban service, which is used to monitor attempted brute forces, for example
- The UFW service which is used as a firewall



These services and logs are a great way in monitoring the health of your system and protecting it. Not only that, but the logs for services such as a web server contain information about every single request - allowing developers or administrators to diagnose performance issues or investigate an intruder's activity. For example, the two types of log files below that are of interest:

- access log
- error log



There are, of course, logs that store information about how the OS is running itself and actions that are performed by users, such as authentication attempts.

Windows OS

The Windows operating system has a long history dating back to 1985, and currently, it is the dominant operating system in both home use and corporate networks. Because of this, Windows has always been targeted by hackers & malware writers.

Windows XP was a popular version of Windows and had a long-running. Microsoft announced Windows Vista , which was a complete overhaul of the Windows operating system. There were many issues with Windows Vista. It wasn't received well by Windows users, and it was quickly phased out.

When Microsoft announced the end-of-life date for Windows XP, many customers panicked. Corporations, hospitals, etc., scrambled and tested the next viable Windows version , which was Windows 7, against many other hardware and devices. Vendors had to work against the clock to ensure their products worked with Windows 7 for their customers. If they couldn't, their customers had to break their agreement and find another vendor that upgraded their products to work with Windows 7. It was a nightmare for many, and Microsoft took note of it.

Windows 7, as quickly as it was released soon after, was marked with an end of support date. Windows 8.x came and left and it was short-lived, like Vista.

Then arrived Windows 10 followed by Windows 11, which is the current Windows operating system version for desktop computers.

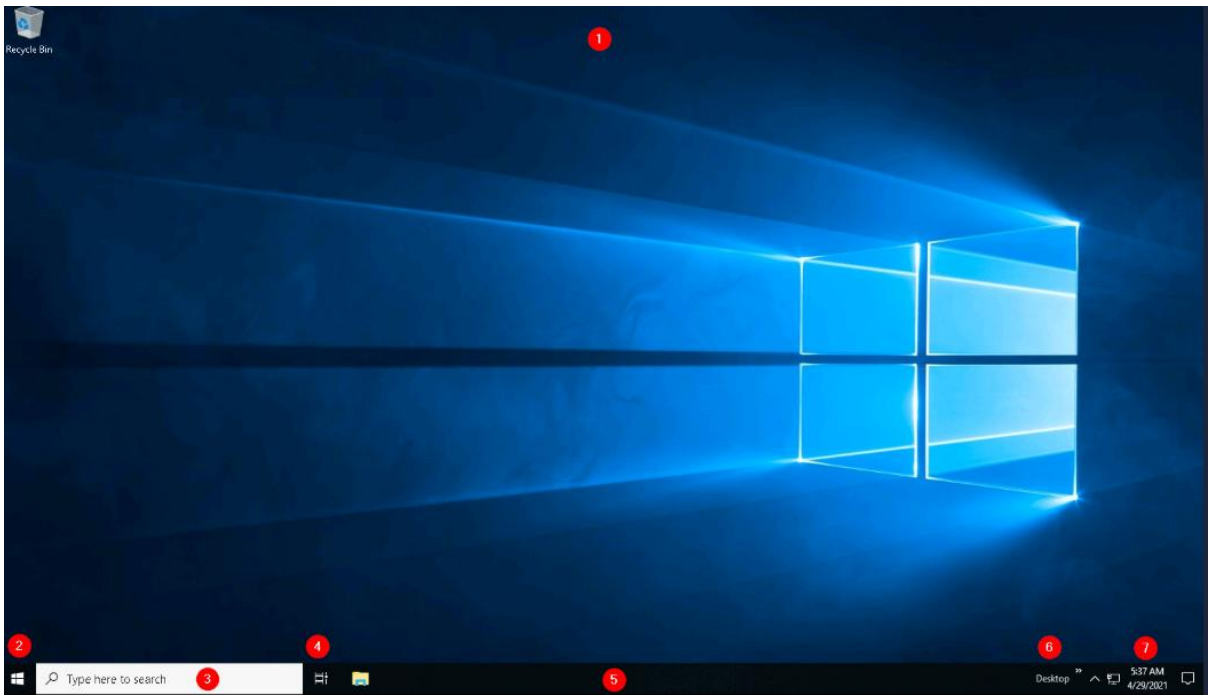
Windows 11 comes in 2 flavors, Home and Pro. You can read the difference between the Home and Pro here.

Even though we didn’t talk about servers, the current version of the Windows operating system for servers is Windows Server 2025.

Many critics like to bash on Microsoft, but they have made long strides to improve the usability and security with each new version of Windows

The Windows Desktop, aka the graphical user interface or GUI in short, is the screen that welcomes you once you log into a Windows 10 machine.

Traditionally, you need to pass the login screen first. The login screen is where you need to enter valid account credentials; usually, a username & password of a preexisting Windows account on that particular system or in the Active Directory environment (if it's a domain-joined machine).



The above screenshot is an example of a typical Windows Desktop. Each component that makes up the GUI is explained briefly below.

1. The Desktop
2. Start Menu
3. Search Box (Cortana)
4. Task View
5. Taskbar
6. Toolbars
7. Notification Area

🖥️ The Desktop

- **Purpose:** Quick access to shortcuts for programs, folders, files, etc.
- **Organization:** Items may be:
    - **Organized** in folders and sorted alphabetically.
    - **Unorganized**, scattered randomly.

Desktop Customization

- **Right-click Menu:**
    - Resize icons
    - Arrange or auto-arrange items
    - Copy/paste
    - Create new items (folder, shortcut, text document)

Display Settings

- Change **screen resolution** and **orientation**
- Configure **multi-monitor setups**

🔒 *Note:* In a **Remote Desktop session**, some display settings may be disabled.

Personalization

- Change **wallpaper**, **fonts**, **themes**, and **color scheme**
- Accessed by right-clicking the desktop and selecting **Personalize**

🪟 **The Start Menu**

Overview

- In Windows 10+, the **Start button** is a **Windows logo**, not the word "Start"
- Provides access to:
    - Apps and programs
    - Files
    - Settings
    - System tools

Start Menu Sections

1. User and System Shortcuts

- Options: Lock screen, Sign out, Account settings
- Quick access to:
    - **Documents**
    - **Pictures**
    - **Settings** (gear icon)
    - **Power** (shutdown, restart, log off)

2. App List

- **Recently Added:** Shows apps you've recently installed
- **All Apps:** Listed alphabetically
    - Click a letter heading to jump using an **alphabet grid**

📌 *Note:* Google Chrome will no longer appear as a "Recently Added" app in the VM.

3. Tiles (Right Side)

- **Pinned apps and utilities**
- Right-click options:
    - Resize
    - Unpin from Start
    - View Properties
- To add a tile: right-click app > **Pin to Start**

📌 **The Taskbar**

- Displays all **currently open apps/files**
- Hovering over an icon shows:
    - **Thumbnail preview**
    - **Tooltip** with name (helpful when multiple windows are open)
- Closing an app removes it from the Taskbar (unless pinned)

Customization

- **Right-click the Taskbar** for options like:
    - Lock/unlock taskbar
    - Turn toolbars on/off
    - Open Taskbar settings

🔔 **The Notification Area (System Tray)**

- Located in the **bottom-right corner**
- Displays:
    - Date & time
    - System icons (e.g., volume, network)
- Customize icons:

- Right-click Taskbar > Taskbar Settings
- Scroll to **Notification Area** to add/remove icons

📄 Refer to Microsoft Docs for more info on the **Start Menu** and **Notification Area**

🧠 **Quick Tip:**

Right-click any file, folder, or app icon to:

- View more information
- Access actions like "Open", "Rename", "Delete", "Properties", etc.

NTFS

The file system used in modern versions of Windows is the **New Technology File System** or simply NTFS .

Before NTFS, there was **FAT16/FAT32** (File Allocation Table) and **HPFS** (High Performance File System).

You still see FAT partitions in use today. For example, you typically see FAT partitions in USB devices, MicroSD cards, etc. but traditionally not on personal Windows computers/laptops or Windows servers.

NTFS is known as a journaling file system. In case of a failure, the file system can automatically repair the folders/files on disk using information stored in a log file. This function is not possible with FAT.

NTFS addresses many of the limitations of the previous file systems; such as:

- Supports files larger than 4GB
- Set specific permissions on folders and files
- Folder and file compression
- Encryption (Encryption File System or **EFS**)

If you're running Windows, what is the file system your Windows installation is using? You can check the Properties (right-click) of the drive your operating system is installed on, typically the C drive (C:\).

Let's speak briefly on some features that are specific to NTFS.

On NTFS volumes, you can set permissions that grant or deny access to files and folders.

The permissions are:

- **Full control**
- **Modify**
- **Read & Execute**
- **List folder contents**
- **Read**

- **Write**

The below image lists the meaning of each permission on how it applies to a file and a folder.
(credit Microsoft)

Permission	Meaning for Folders	Meaning for Files
Read	Permits viewing and listing of files and subfolders	Permits viewing or accessing of the file's contents
Write	Permits adding of files and subfolders	Permits writing to a file
Read & Execute	Permits viewing and listing of files and subfolders as well as executing of files; inherited by files and folders	Permits viewing and accessing of the file's contents as well as executing of the file
List Folder Contents	Permits viewing and listing of files and subfolders as well as executing of files; inherited by folders only	N/A
Modify	Permits reading and writing of files and subfolders; allows deletion of the folder	Permits reading and writing of the file; allows deletion of the file
Full Control	Permits reading, writing, changing, and deleting of files and subfolders	Permits reading, writing, changing and deleting of the file

How can you view the permissions for a file or folder?

- Right-click the file or folder you want to check for permissions.
- From the context menu, select Properties .
- Within Properties, click on the Security tab.
- In the Group or user names list, select the user, computer, or group whose permissions you want to view.

🔍 ADS Explanation 2 :

ADS lets you hide extra data inside a normal file, without changing how that file looks or behaves in File Explorer.

📦 Example:

Imagine a file called note.txt. Normally, it has one stream — the actual content of the file. But with ADS, you can secretly attach another "hidden" file to it like this:

echo HiddenData > note.txt:hidden.txt

- note.txt still opens normally.
- hidden.txt is an **alternate stream** that isn't visible in File Explorer.
- The file size of note.txt doesn’t appear to change.

🧠 Why Does ADS Exist?

- ✅ **Legit use**: Windows uses ADS to store metadata. For example, when you download a file, it gets a Zone.Identifier stream to track its origin (internet, local network, etc.).
- ❌ **Malicious use**: Hackers can hide malware, scripts, or tools in ADS so that they're not easily seen or detected.

🔐 Security Tip:

You can use **PowerShell** to see if a file has hidden streams:

Get-Item -Path .\file.txt -Stream *

🧪 Quick Test (Try it Yourself):

echo Hello > test.txt

echo SecretStuff > test.txt:hidden

more < test.txt:hidden

Now check the streams with PowerShell.

Alternate Data Streams (**ADS**)

Alternate Data Streams (ADS) is a file attribute specific to Windows **NTFS** (New Technology File System).

Every file has at least one data stream ($DATA), and ADS allows files to contain more than one stream of data. Natively Window Explorer doesn't display ADS to the user. There are 3rd party executables that can be used to view this data, but Powershell gives you the ability to view ADS for files.

From a security perspective, malware writers have used ADS to hide data.

Not all its uses are malicious. For example, when you download a file from the Internet, there are identifiers written to ADS to identify that the file was downloaded from the Internet.

🗂️ What is the C:\Windows Folder?

- The C:\Windows folder is where the **core files of the Windows operating system** are stored.
- It doesn't **have to** be in C: — Windows can be installed on a different drive or folder.
- Its **location** is referenced using a **system environment variable**:

%windir% → Points to the Windows directory (e.g., C:\Windows)

🌐 **What Are Environment Variables?**

- Special placeholders used by Windows to reference key system info.
- Examples:
    - %windir% → Windows directory
    - %temp% → Temporary files folder
    - %systemroot% → Same as %windir%
- Used in scripting, system configs, and automation tools to ensure compatibility.

🏛️ **What is the System32 Folder?**

- Located inside the Windows directory:

C:\Windows\System32 (or %windir%\System32)

- Contains **critical system files**, such as:
    - Core Windows system libraries (.DLL files)
    - Executables for built-in tools (e.g., cmd.exe, taskmgr.exe, regedit.exe)
    - Device drivers and control utilities
    - PowerShell, WMI, and administrative tools

🧨 **Why Is System32 So Sensitive?**

❗ **Deleting or modifying files in System32 can break Windows completely.**

- System won't boot properly.
- Programs will fail to launch.
- You might need to reinstall the OS if it gets corrupted.

⚙️ **Examples of Common Tools Inside System32**

Tool	Purpose
cmd.exe	Command Prompt
powershell.exe	PowerShell
regedit.exe	Registry Editor
taskmgr.exe	Task Manager
ipconfig.exe	Network Configuration

Tool	Purpose
ping.exe	Network connectivity testing
netstat.exe	Viewing network connections

These are **built into Windows** and live in System32, which is why you can run them from any path using just their name.

🔐 **Security Implications**

- **Malware** often tries to hide or replace files in System32 to stay undetected.
- **System administrators and cybersecurity professionals** often monitor this folder for unauthorized changes.

🧠 **Key Takeaways**

- **%windir%** is the variable pointing to the Windows install directory.
- **System32** is the beating heart of Windows — it contains essential files and tools.
- **Don't delete or edit anything** in this folder unless you're 100% sure.
- Many **Windows tools you'll use** in learning or working with Windows internals come from this directory.

User accounts can be one of two types on a typical local Windows system: Administrator & Standard User.

The user account type will determine what actions the user can perform on that specific Windows system.

An Administrator can make changes to the system: add users, delete users, modify groups, modify settings on the system, etc.

A Standard User can only make changes to folders/files attributed to the user & can't perform system-level changes, such as install programs.

You are currently logged in as an Administrator. There are several ways to determine which user accounts exist on the system.

The large majority of home users are logged into their Windows systems as local administrators. Remember from the previous task that any user with administrator as the account type can make changes to the system.

UAC – User Account Control

A user doesn't need to run with high (elevated) privileges on the system to run tasks that don't require such privileges, such as surfing the Internet, working on a Word document, etc. This elevated privilege increases the risk of system compromise because it makes it easier for malware to infect the system. Consequently, since the user account can make changes to the system, the malware would run in the context of the logged-in user.

To protect the local user with such privileges, Microsoft introduced **User Account Control** (UAC). This concept was first introduced with the short-lived Windows Vista and continued with versions of Windows that followed.

Note : UAC (by default) doesn't apply for the built-in local administrator account.

How does UAC work? When a user with an account type of administrator logs into a system, the current session doesn't run with elevated permissions. When an operation requiring higher-level privileges needs to execute, the user will be prompted to confirm if they permit the operation to run.

Settings and Control Panel

On a Windows system, the primary locations to make changes are the Settings menu and the Control Panel.For a long time, the Control Panel has been the go-to location to make system changes, such as adding a printer, uninstall a program, etc.

The Settings menu was introduced in Windows 8, the first Windows operating system catered to touch screen tablets, and is still available in Windows 10. As a matter of fact, the Settings menu is now the primary location a user goes to if they are looking to change the system.

There are similarities and differences between the two menus.

Control Panel is the menu where you will access more complex settings and perform more complex actions. In some cases, you can start in Settings and end up in the Control Panel.

Task Manager

The Task Manager provides information about the applications and processes currently running on the system. Other information is also available, such as how much CPU and RAM are being utilized, which falls under **Performance**.

System Configuration

The **System Configuration** utility (MSConfig) is for advanced troubleshooting, and its main purpose is to help diagnose startup issues.

The utility has five tabs across the top. Below are the names for each tab. We will briefly cover each tab in this task.

1. General
2. Boot
3. Services

4. Startup
5. Tools

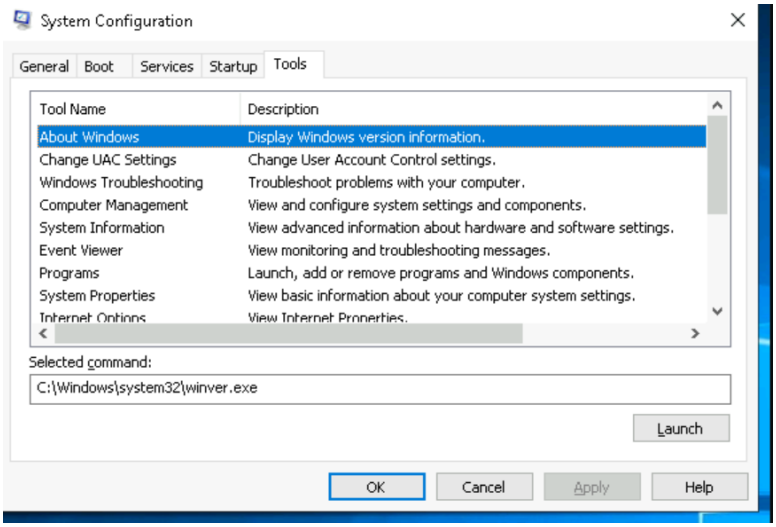
In the General tab, we can select what devices and services for Windows to load upon boot. The options are: Normal, Diagnostic, or Selective.

In the Boot tab, we can define various boot options for the Operating System.

The **Services** tab lists all services configured for the system regardless of their state (running or stopped). A service is a special type of application that runs in the background.

Startup - As you can see, Microsoft advises using **Task Manager (**taskmgr**)** to manage (enable/disable) startup items. The System Configuration utility is **NOT** a startup management program.

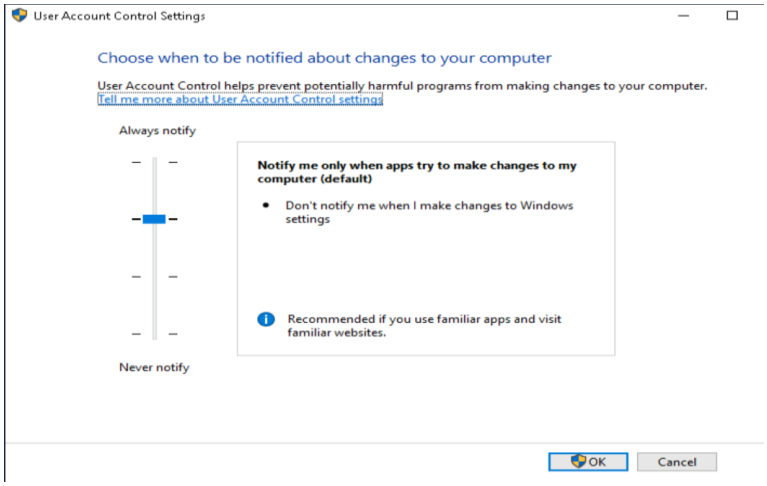
There is a list of various utilities (tools) in the Tools tab that we can run to configure the operating system further. There is a brief description of each tool to provide some insight into what the tool is for.



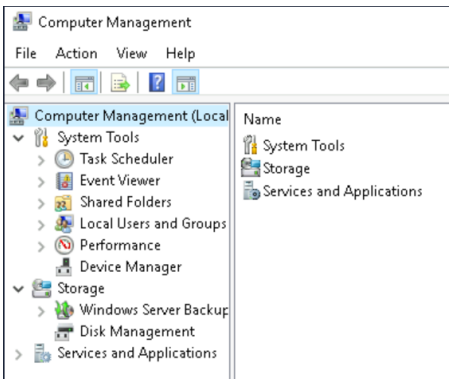
User Account Control (UAC) was covered in great detail in Windows Fundamentals 1.

The UAC settings can be changed or even turned off entirely (not recommended).

You can move the slider to see how the setting will change the UAC settings and Microsoft's stance on the setting.



The **Computer Management** (compmgmt) utility has three primary sections: **System Tools**, **Storage**, and **Services and Applications**.

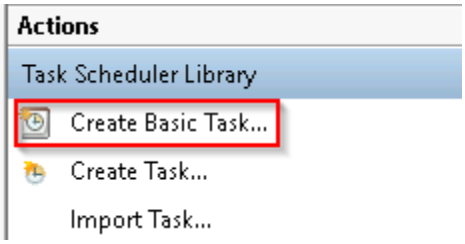


System Tools

Let's start with **Task Scheduler**. Per Microsoft, with Task Scheduler, we can create and manage common tasks that our computer will carry out automatically at the times we specify.

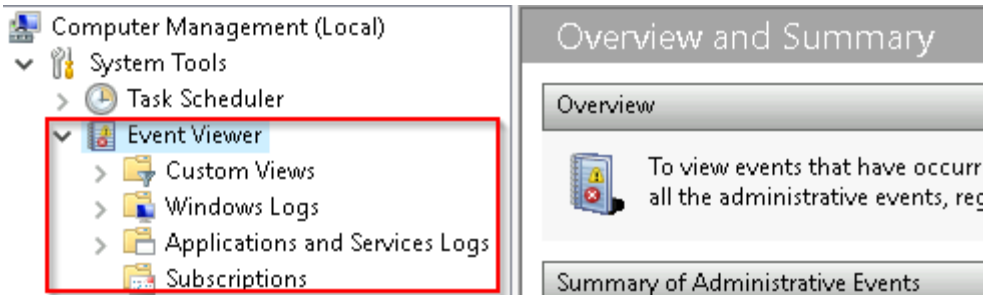
A task can run an application, a script, etc., and tasks can be configured to run at any point. A task can run at log in or at log off. Tasks can also be configured to run on a specific schedule, for example, every five mins.

To create a basic task, click on Create Basic Task under **Actions** (right pane).



Next is **Event Viewer**.

Event Viewer allows us to view events that have occurred on the computer. These records of events can be seen as an audit trail that can be used to understand the activity of the computer system. This information is often used to diagnose problems and investigate actions executed on the system.



Event Viewer has three panes.

1. The pane on the left provides a hierarchical tree listing of the event log providers. (as shown in the image above)
2. The pane in the middle will display a general overview and summary of the events specific to a selected provider.

3. The pane on the right is the actions pane.

There are five types of events that can be logged. Below is a table from docs.microsoft.com providing a brief description for each.

The following table describes the five event types used in event logging.

Event type	Description
Error	An event that indicates a significant problem such as loss of data or loss of functionality. For example, if a service fails to load during startup, an Error event is logged.
Warning	An event that is not necessarily significant, but may indicate a possible future problem. For example, when disk space is low, a Warning event is logged. If an application can recover from an event without loss of functionality or data, it can generally classify the event as a Warning event.
Information	An event that describes the successful operation of an application, driver, or service. For example, when a network driver loads successfully, it may be appropriate to log an Information event. Note that it is generally inappropriate for a desktop application to log an event each time it starts.
Success Audit	An event that records an audited security access attempt that is successful. For example, a user's successful attempt to log on to the system is logged as a Success Audit event.
Failure Audit	An event that records an audited security access attempt that fails. For example, if a user tries to access a network drive and fails, the attempt is logged as a Failure Audit event.

The standard logs are visible under **Windows Logs**. Below is a table from docs.microsoft.com providing a brief description for each.

The event log contains the following standard logs as well as custom logs:

Log	Description
Application	Contains events logged by applications. For example, a database application might record a file error. The application developer decides which events to record.
Security	Contains events such as valid and invalid logon attempts, as well as events related to resource use such as creating, opening, or deleting files or other objects. An administrator can start auditing to record events in the security log.
System	Contains events logged by system components, such as the failure of a driver or other system component to load during startup.
CustomLog	Contains events logged by applications that create a custom log. Using a custom log enables an application to control the size of the log or attach ACLs for security purposes without affecting other applications.

Shared Folders is where you will see a complete list of shares and folders shared that others can connect to.

Share Name	Folder Path	Type	# Client Connections	Description
ADMIN$	C:\Windows	Windows	0	Remote Admin
C$	C:\	Windows	0	Default share
IPC$		Windows	0	Remote IPC

Folders or shares with a **dollar sign ($)** at the end — like ADMIN$ — are **hidden administrative shares** used in Windows operating systems.

🔐 What is ADMIN$?

- ADMIN$ is a **default administrative share** that points to the **Windows directory**, usually C:\Windows.

- It's **hidden from normal users** — the dollar sign ($) makes it invisible in standard network browse lists.
- It is used **remotely by administrators** for:
    - Accessing system files
    - Remote management
    - Remote software installation or patching

🔧 **Common Admin Shares:**

Share Name	Path	Purpose
ADMIN$	C:\Windows	Remote admin tasks
C$	C:\	Full access to C drive for admins
D$, E$	D:\, E:\...	Admin access to other drives
IPC$	-	Inter-process communication (SMB)

📌 **Notes:**

- You typically need **administrator credentials** to access ADMIN$.
- It's used by tools like **PsExec**, **Group Policy**, **SCCM**, etc.
- Disabling these shares can improve security — but may break some management tools.

In the above image, under Shares, are the default share of Windows, C$, and default remote administration shares created by Windows, such as ADMIN$.

As with any object in Windows, you can right-click on a folder to view its properties, such as Permissions (who can access the shared resource).

Under Sessions, you will see a list of users who are currently connected to the shares. In this VM, you won't see anybody connected to the shares.

All the folders and/or files that the connected users access will list under Open Files.

The Local Users and Groups section you should be familiar with from Windows Fundamentals 1 because it's lusrmgr.msc.

In Performance, you'll see a utility called Performance Monitor (perfmon).

Perfmon is used to view performance data either in real-time or from a log file. This utility is useful for troubleshooting performance issues on a computer system, whether local or remote.

Device Manager allows us to view and configure the hardware, such as disabling any hardware attached to the computer.

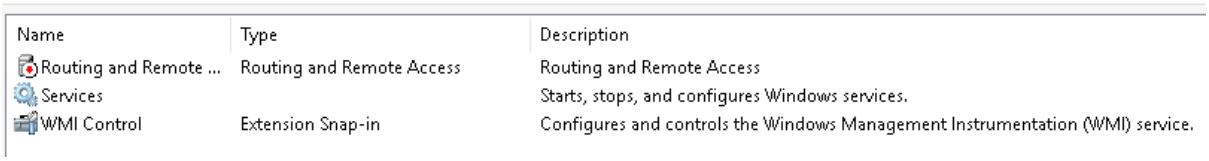
Storage

Under Storage is **Windows Server Backup** and **Disk Management**. We'll only look at Disk Management

Disk Management is a system utility in Windows that enables you to perform advanced storage tasks. Some tasks are:

- Set up a new drive
- Extend a partition
- Shrink a partition
- Assign or change a drive letter (ex. E:)

Services and Applications



Recall from the previous task; a service is a special type of application that runs in the background. Here you can do more than enable and disable a service, such as view the Properties for the service.

WMI Control configures and controls the **Windows Management Instrumentation** (WMI) service.

Per Wikipedia, "*WMI allows scripting languages (such as VBScript or Windows PowerShell) to manage Microsoft Windows personal computers and servers, both locally and remotely. Microsoft also provides a command-line interface to WMI called Windows Management Instrumentation Command-line (WMIC)*."

Note: The WMIC tool is deprecated in Windows 10, version 21H1. Windows PowerShell supersedes this tool for WMI.

We're continuing with Tools that are available through the **System Configuration** panel.

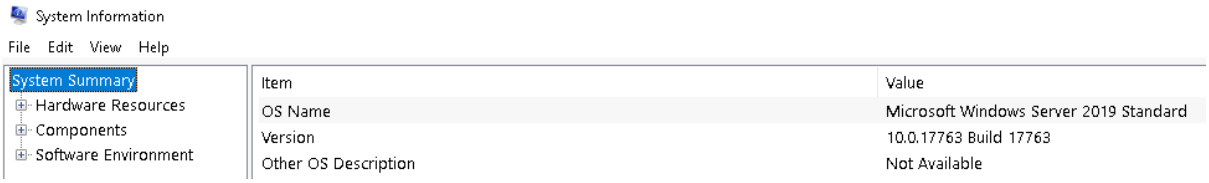
What is the **System Information** (msinfo32) tool?

Per Microsoft, "*Windows includes a tool called Microsoft System Information (Msinfo32.exe). This tool gathers information about your computer and displays a comprehensive view of your hardware, system components, and software environment, which you can use to diagnose computer issues.*"

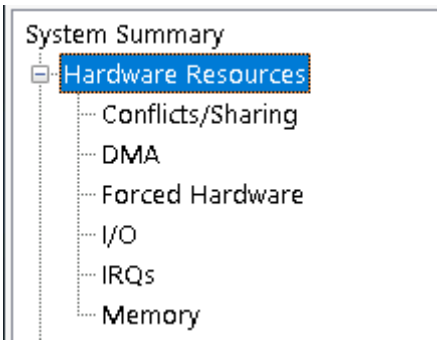
The information in **System Summary** is divided into three sections:

- **Hardware Resources**
- **Components**
- **Software Environment**

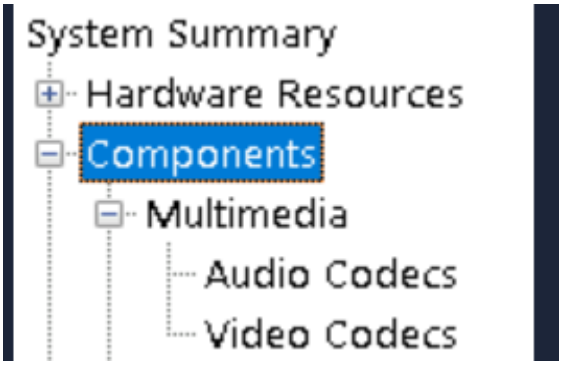
System Summary will display general technical specifications for the computer, such as processor brand and model.



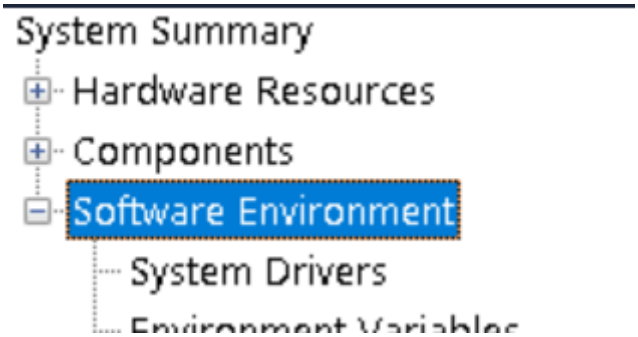
The information displayed in **Hardware Resources** is not for the average computer user. If you want to learn more about this section, refer to the official Microsoft page.



Under **Components**, you can see specific information about the hardware devices installed on the computer. Some sections don't show any information, but some sections do, such as **Display** and **Input.**



In the **Software Environment** section, you can see information about software baked into the operating system and software you have installed. Other details are visible in this section as well, such as the **Environment Variables** and **Network Connections**.



Recall from the Windows Fundamentals 1 room (The Windows\System32 Folder task) where **Environment Variables** was briefly touched on.

Per Microsoft, "*Environment variables store information about the operating system environment. This information includes details such as the operating system path, the number of processors used by the operating system, and the location of temporary folders.*

The environment variables store data that is used by the operating system and other programs. For example, the WINDIR environment variable contains the location of the Windows installation directory. Programs can query the value of this variable to determine where Windows operating system files are located".

🛠️ Resource Monitor (resmon)

Resource Monitor is a powerful **Windows diagnostic tool** available through the **System Configuration panel** or by typing resmon in the Start menu or Run box.

🔍 What It Does

Per Microsoft:

"Resource Monitor displays per-process and aggregate CPU, memory, disk, and network usage information, and details about which processes are using individual file handles and modules."

In simple terms, it lets you:

- See **which apps** are using your **CPU**, **memory**, **disk**, and **network**
- Monitor **real-time usage** via graphs
- **Kill frozen processes**, **analyze deadlocks**, and troubleshoot performance issues
- **Pause, resume, or stop services**
- Filter data by specific processes or resources

🧠 Who It's For

While anyone can open and use Resource Monitor, it is especially useful for:

- **Power users**
- **System administrators**
- **IT professionals**
- **Cybersecurity analysts** (for detecting unusual process behavior)

📊 Main Tabs in Resource Monitor

Each tab corresponds to a key system resource:

Tab	Description
Overview	Combined view of CPU, Disk, Network, and Memory usage
CPU	Lists processes, threads, services, and associated handles/modules
Memory	Shows used/available memory, process memory usage, and memory mapping

Tab	Description
Disk	Displays read/write activity per process, disk queues, and file access
Network	Shows process-level network usage, ports in use, and TCP connections

---

📈 Real-Time Graphs

On the **far-right pane**, real-time graphs display:

- CPU load per core
- Disk I/O activity
- Network throughput
- Memory usage trends

These update live as the system runs, helping to quickly identify bottlenecks or misbehaving processes.

🖱️ Quick Launch

You can open Resource Monitor via:

- Start > search **Resource Monitor**
- Run (Win + R) > type resmon > Enter
- Task Manager > **Performance tab** > click **Open Resource Monitor** at the bottom

🛠️ Tools Available Through the System Configuration Panel: Command Prompt (cmd)

💬 What Is the Command Prompt?

The **Command Prompt** (cmd) is a **text-based interface** that allows users to interact directly with the Windows operating system. It predates the graphical user interface (GUI) and was once the **only way** to operate a computer.

While the GUI makes things easier with buttons and icons, the command prompt still holds power — especially for **troubleshooting**, **advanced tasks**, and **automation**.

📌 Basic Commands to Know

Command	Purpose
hostname	Displays the name of the computer.
whoami	Shows the currently logged-in user.

Command	Purpose
cls	Clears the screen in the command prompt.

🛠️ Troubleshooting Commands

1. ipconfig

- Shows network configuration settings such as IP address, subnet mask, and default gateway.
- Helpful for checking network connectivity.

🔍 View Help Manual:

ipconfig /?

---

2. netstat

- Displays active network connections, listening ports, and protocol statistics.
- Useful for detecting open connections, ports in use, and potential unauthorized activity.

🔍 Syntax Example:

netstat -a :: Shows all active connections and listening ports

netstat -b :: Shows the executable associated with each connection

netstat /? :: Displays help information

---

3. net

- Manages **network resources**, such as users, shared folders, and sessions.
- Typing net alone will show a list of sub-commands like user, use, view, localgroup, share, and session.

🔍 Get Help for a Specific Sub-Command:

net help user

net help share

Note: Unlike other commands, net does **not** use /?" for help — it uses net help [sub-command].

💡 Tip: Every Command Has a Manual

Most commands support a built-in help option:

[command] /?

Examples:

- cls /?
- ping /?
- dir /?

This gives you the syntax, description, and available parameters for each command.

🧪 **Try It Yourself**

Use the command prompt to explore:

- Your system name (hostname)
- Your current user (whoami)
- Your IP and gateway (ipconfig)
- Network sessions (netstat)
- User accounts and groups (net user, net localgroup

We're continuing with Tools that are available through the **System Configuration** panel.

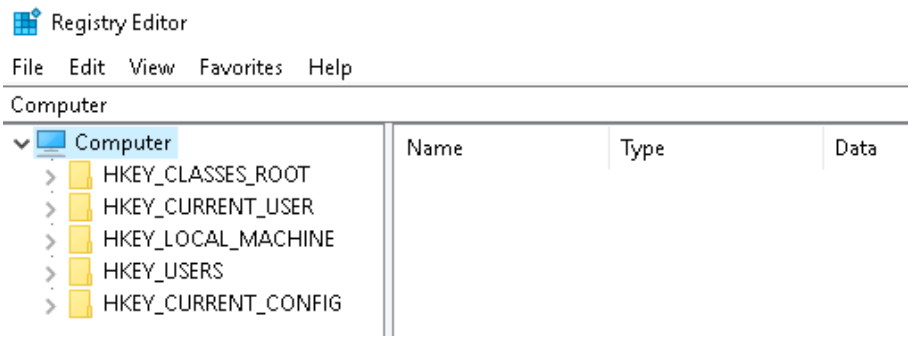
The **Windows Registry** (per Microsoft) is a central hierarchical database used to store information necessary to configure the system for one or more users, applications, and hardware devices.

The registry contains information that Windows continually references during operation, such as:

- Profiles for each user
- Applications installed on the computer and the types of documents that each can create
- Property sheet settings for folders and application icons
- What hardware exists on the system
- The ports that are being used.

Warning: The registry is for advanced computer users. Making changes to the registry can affect normal computer operations.

There are various ways to view/edit the registry. One way is to use the **Registry Editor** (regedit).



Windows Update

Windows Update is a service provided by Microsoft to provide security updates, feature enhancements, and patches for the Windows operating system and other Microsoft products, such as Microsoft Defender.

Updates are typically released on the 2nd Tuesday of each month. This day is called **Patch Tuesday**. That doesn't necessarily mean that a critical update/patch has to wait for the next Patch Tuesday to be released. If the update is urgent, then Microsoft will push the update via the Windows Update service to the Windows devices.

Windows Security *is your home to manage the tools that protect your device and your data*".

Windows Security is also available in **Settings**.

Virus & threat protection is divided into two parts:

- **Current threats**
- **Virus & threat protection settings**

Current threats

Scan options

- **Quick scan** - Checks folders in your system where threats are commonly found.
- **Full scan** - Checks all files and running programs on your hard disk. This scan could take longer than one hour.
- **Custom scan** - Choose which files and locations you want to check.

Threat history

- **Last scan** - Windows Defender Antivirus automatically scans your device for viruses and other threats to help keep it safe.
- **Quarantined threats** - Quarantined threats have been isolated and prevented from running on your device. They will be periodically removed.
- **Allowed threats** - Allowed threats are items identified as threats, which you allowed to run on your device.

Warning: Allow an item to run that has been identified as a threat only if you are **100%** sure of what you are doing.

Virus & threat protection settings

Manage settings

- **Real-time protection** - Locates and stops malware from installing or running on your device.
- **Cloud-delivered protection** - Provides increased and faster protection with access to the latest protection data in the cloud.
- **Automatic sample submission** - Send sample files to Microsoft to help protect you and others from potential threats.
- **Controlled folder access** - Protect files, folders, and memory areas on your device from unauthorized changes by unfriendly applications.
- **Exclusions** - Windows Defender Antivirus won't scan items that you've excluded.
- **Notifications** - Windows Defender Antivirus will send notifications with critical information about the health and security of your device.

Warning: Excluded items could contain threats that make your device vulnerable. Only use this option if you are **100%** sure of what you are doing.

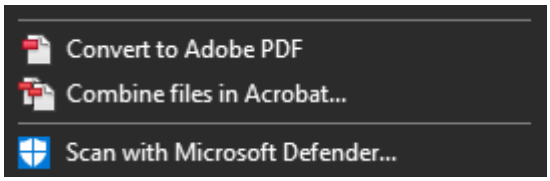
Virus & threat protection updates

- **Check for updates** - Manually check for updates to update Windows Defender Antivirus definitions.

Ransomware protection

- **Controlled folder access** - Ransomware protection requires this feature to be enabled, which in turn requires Real-time protection to be enabled.

Tip: You can perform on-demand scans on any file/folder by right-clicking the item and selecting 'Scan with Microsoft Defender'.



What is a **firewall**?

Per Microsoft, "*Traffic flows into and out of devices via what we call ports. A firewall is what controls what is - and more importantly isn't - allowed to pass through those ports. You can think of it like a security guard standing at the door, checking the ID of everything that tries to enter or exit*".

What is the difference between the 3 (**Domain**, **Private**, and **Public**)?

Per Microsoft, "*Windows Firewall offers three firewall profiles: domain, private and public*".

- **Domain** - *The domain profile applies to networks where the host system can authenticate to a domain controller.*

- **Private** - *The private profile is a user-assigned profile and is used to designate private or home networks.*
- **Public** - *The default profile is the public profile, used to designate public networks such as Wi-Fi hotspots at coffee shops, airports, and other locations.*

Configuring the **Windows Defender Firewall** is for advanced Windows users.

Command to open the Windows Defender Firewall is WF.msc.

The **Trusted Platform Module (TPM)** is a **hardware-based security device** built into many modern computers. It plays a critical role in securing sensitive data and protecting systems from unauthorized access, especially at the hardware level.

✅ Core Functions of TPM

TPM chips are **secure crypto-processors**, meaning they are designed to safely perform **cryptographic operations**, such as:

- **Key generation and management**
- **Secure storage of cryptographic keys, passwords, and certificates**
- **Platform integrity checks** (e.g., verifying the boot process hasn't been tampered with)
- **Digital signatures** and **encryption/decryption**

🔐 Why It's Important

- **Hardware-based security** is much harder to compromise than software-only solutions.
- TPMs have **tamper-resistant designs**, making it extremely difficult for attackers to extract secrets stored inside.
- **Operating systems like Windows use TPM** for features such as:
    - **BitLocker Drive Encryption**
    - **Windows Hello** for secure logins
    - **Measured Boot and Secure Boot**

🔄 TPM in Action

During system startup, the TPM:

1. Measures components like firmware and bootloader.
2. Compares them against known-good values (hashes).
3. If anything has been altered, it can prevent the system from booting or warn the user/admin.

🛠️ TPM Versions

- **TPM 1.2**: Older version, supports basic functions.
- **TPM 2.0**: More flexible and secure. Required for **Windows 11** installation.

💡 Real-World Use Cases

- **Enterprise endpoint protection**
- **Multi-factor authentication**
- **Secure VPN connections**
- **Certificate-based authentication**

In summary, **TPM is a foundational building block for modern device security**, especially in environments where **data protection and integrity** are essential.

A **cryptographic key** is a string of bits (like a password, but stronger) used in **cryptographic algorithms** to perform **encryption**, **decryption**, **authentication**, and **data integrity checks**.

🔑 What Does a Cryptographic Key Do?

Depending on the type of cryptography, keys are used for:

1. **Encrypting data** (turning readable text into unreadable text)
2. **Decrypting data** (making encrypted data readable again)
3. **Creating digital signatures** (proving who sent the data)
4. **Verifying digital signatures** (confirming the data hasn't been altered)

🔐 Types of Cryptographic Keys

1. Symmetric Key

- Same key is used to **encrypt and decrypt**.
- Fast, but both sender and receiver need to share the same secret.
- Example: AES (Advanced Encryption Standard)

Think: Like using one key to lock and unlock a door.

2. Asymmetric Key (Public/Private Key)

- Uses a **pair of keys**:
    - **Public key** (shared with others)
    - **Private key** (kept secret)
- One key encrypts, the other decrypts.

- Example: RSA, ECC

Think: Like someone giving you a locked mailbox (public key), and only they have the key to open it (private key).

🧠 Example in Action

Suppose Alice wants to send a secure message to Bob:

- **With symmetric encryption**: They both use the same secret key.
- **With asymmetric encryption**: Alice encrypts the message using Bob's **public key**. Only Bob can decrypt it with his **private key**.

🔄 Cryptographic Key Use in Daily Life

- HTTPS (websites with a lock symbol)
- Secure emails
- VPNs
- Messaging apps like Signal or WhatsApp
- Encrypted file storage

🧱 Summary

A **cryptographic key** is like the secret ingredient that enables secure digital communication and data protection. Without it, encryption would be meaningless.

🔐 What is BitLocker?

BitLocker is a security feature in Windows that **encrypts your hard drive** to protect your data.

If your computer is **lost or stolen**, BitLocker helps make sure no one can access your files without the correct password or key.

🧩 How it works with TPM

If your computer has a **TPM chip** (Trusted Platform Module), BitLocker works even better. The TPM helps:

- **Keep encryption keys safe**
- **Make sure your system hasn't been tampered with** before it starts

💾 What is Volume Shadow Copy Service (VSS)?

VSS is a Windows feature that lets the system take **snapshots of your data** at a specific point in time. These are also called **shadow copies** or **restore points**.

They're useful for:

- **Backing up data**
- **Restoring your system** to a previous state if something goes wrong

These copies are stored in a hidden folder called **System Volume Information** on each drive where protection is turned on.

⚙️ What Can You Do With VSS?

If VSS (System Protection) is enabled, you can:

- Create a **restore point**
- Perform a **system restore**
- Change **restore settings**
- **Delete old restore points**

⚠️ Security Note

Malware (especially ransomware) often tries to **delete shadow copies** so victims **can't recover** their files. That's why it's important to also have **offline or off-site backups** for better protection.

That's all for the pre security notes, designed especially for beginners. They cover the essential concepts you should know before diving into cybersecurity.

🙏 **Thank you** to anyone who takes the time to read and use these notes. I hope they help guide you on your journey into this exciting field!

Best wishes,

Shya Kaplan – Cyber Security Graduate