

# Computer Hardware Basics

## **1. Motherboard (Main Board of Computer)**

The motherboard is the main circuit board of the computer.

**Function:** Connects and allows communication between all components.

**Contains:**

- CPU socket
- RAM slots
- GPU slot (PCIe)
- Storage connectors (SATA/NVMe)

**Analogy:** Like the skeleton + nervous system of the human body — everything is connected through it.

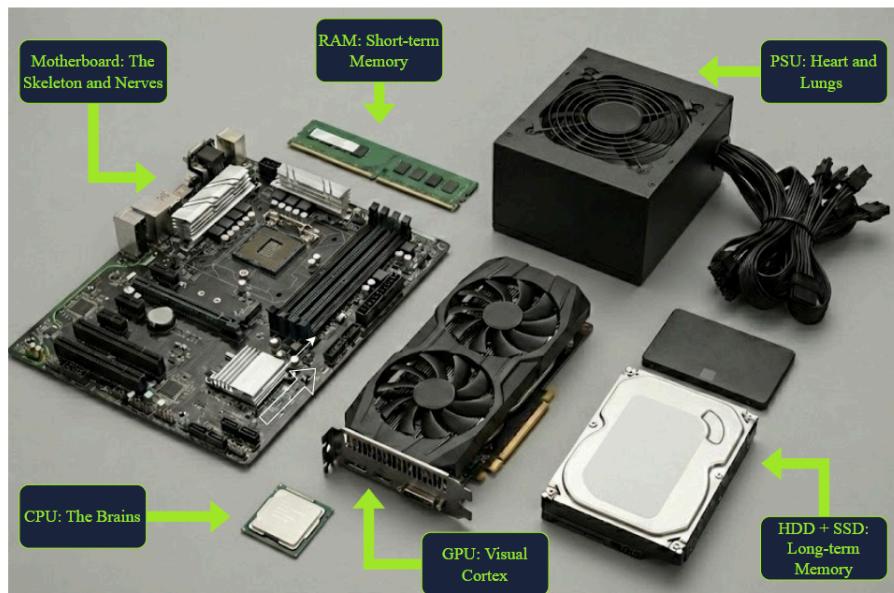
## **2. CPU (Central Processing Unit)**

Main processor of the computer.

**Function:**

- Executes instructions and performs calculations.
- Running programs
- Logic operations
- System control

**Analogy:** Brain of the computer.



## **3. Storage (HDD / SSD)**

**HDD (Hard Disk Drive)**

- Slower
- Mechanical parts
- Cheaper
- Larger storage

**SSD (Solid State Drive)**

- Very fast
- No moving parts
- Expensive but efficient
- Improves boot and loading time

#### **4. RAM (Random Access Memory)**

- Temporary memory used while the computer is running.
- Stores currently running programs and data.
- Volatile (data lost when power off).
- Short-term memory of the brain.

#### **5. GPU (Graphics Processing Unit)**

Processes graphics and visuals.

Function:

- Gaming
- Video editing
- Rendering
- AI/ML tasks

Analogy: Visual cortex of brain.

#### **6. PSU (Power Supply Unit)**

Provides power to all components.

Function:

- Converts AC power → DC power for PC.
- Supplies power to: Motherboard, CPU, GPU, Storage

Analogy: Heart and lungs (gives energy).

### **Types of Computers**

**Laptop:** Portable computer with built-in screen & keyboard. Used for daily tasks like study, browsing, coding.

Key: portability > performance.

**Desktop:** Fixed computer with separate monitor & keyboard. Better cooling and power for long tasks.

Key: performance & stability.

**Workstation:** High-performance computer for professional work (3D design, AI, editing, simulation).

Key: precision & heavy computing.

**Server:** Powerful computer that provides services/data to many users over a network. Runs 24/7.

Key: serves multiple users, no direct personal use.

### Everyday Computers:

**Smartphone:** Pocket computer with internet & apps.

Example: Android, iPhone.

**Tablet:** Touchscreen computer with bigger display.

Example: iPad, drawing tablet.

**IoT Device:** Small device connected to the internet for one task.

Example: smart doorbell, fitness tracker, thermostat.

**Embedded Computer:** Tiny computer inside another device.

Example: washing machine chip, coffee machine, automatic door.

### IoT vs Embedded

- IoT: connected to internet/network
- Embedded: works inside device, may not use internet

## Client–Server Model

### 1. Client & Server

**Client:** Device/app that sends a request.

Example: Browser.

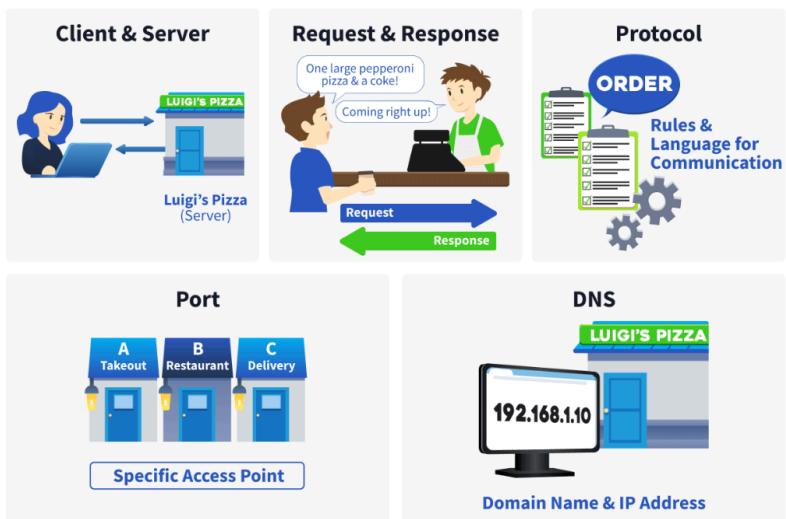
**Server:** System that provides the service or data.

The client always starts (initiates) the request.

### 2. Request & Response

**Request:** Client asks for something.

**Response:** Server replies with data or error.



**Example:**

1. Browser → asks for webpage
2. Server → sends webpage back
3. If something is wrong → server sends an error response.

### **3. Protocol**

**Protocol:** Rules & language for communication.

**Defines:**

- How request is structured
- What commands are used (GET, POST)
- What response should be returned

**Example:** HTTP

Protocol = communication rules

### **4. Port**

**Port:**

- Specific access point for a service on a server.
- A server can run multiple services using different ports.

**Example:**

- Web service → Port 80 / 443
- Other services → different ports

Port identifies a specific service.

### **5. DNS (Domain Name System)**

- Converts domain name → IP address.
- Works like GPS for the internet.

**Example:**

google.com → 142.x.x.x (IP address)

DNS finds the server's address.

## HTTP & GET

**HTTP:** Protocol used for web communication.

**HTTPS:** Secure version of HTTP (encrypted).

**Stateless:** Each request is independent (server doesn't remember previous requests).

### HTTP Methods (Commands)

- GET → Retrieve data
- POST → Send data
- PUT → Update/replace data
- DELETE → Remove data
- PATCH → Partially update
- HEAD → Get headers only
- OPTIONS → Show allowed methods
- CONNECT → Tunnel connection
- TRACE → Debug request

Most common: GET & POST

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time
200	GET	httpdemo.local:8080	/	document	html	737 B	551 B	7 ms
200	GET	httpdemo.local:8080	style.css	stylesheet	css	349 B	164 B	7 ms
200	GET	httpdemo.local:8080	script.js	script	js	355 B	163 B	7 ms
404	GET	httpdemo.local:8080	favicon.ico	img	html	520 B	335 B	7 ms

### GET Method

- Used to request a resource from the server.
  - Example:  
GET /index.html
- Server replies with:
  - Status code (e.g., 200 OK)
  - Requested content

### Important Request Fields

**Scheme:** HTTP or HTTPS

Request Headers (390 B)

Name	Value
Host	http://httpdemo.local:8080
User-Agent	curl/7.54.0
Accept	*/*
Accept-Encoding	gzip, deflate
Accept-Language	en-US,en;q=0.9
Connection	keep-alive

Response Headers (186 B)

Name	Value
Content-Type	text/html; charset=UTF-8
Content-Length	551
Date	Mon, 05 Jan 2026 07:00:41 GMT
Last-Modified	Fri, 19 Dec 2025 13:55:06 GMT
Server	SimpleHTTP/0.6 Python/3.12.3

**Host:** Website name (e.g., google.com)

**Filename/Path:** File requested (e.g., /index.html)

**Address:** Server IP address

**Status:** Shows result (200 OK, 404, etc.)

## Virtualization

### Before Virtualization

Old rule: 1 server = 1 application

#### Problems:

- High cost (many physical servers)
- Low usage (servers mostly idle)
- Slow setup (takes days/weeks)
- Hard to scale (need new server each time)



### Virtualization

Running multiple virtual computers on one physical server.

#### Purpose:

- Save cost
- Use hardware efficiently
- Easy scaling
- Faster deployment

### Hypervisor

Software that manages virtual machines and shares hardware safely.

Acts like: manager/referee between VMs and physical server.

Example: Virtual box / VMware



### Virtual Machine (VM)

Virtual computer inside a physical server.

#### Each VM:

- Has its own OS
- Apps and settings
- Works like real computer
- Shares same hardware

## **Types of Hypervisors:**

### **Type 1 (Bare-metal):**

- Runs directly on hardware
- Very fast and efficient
- Used in data centers & production servers
- Example: VMware ESXi, Hyper-V (server)

### **Type 2 (Hosted):**

- Runs inside an existing OS
- Easy to install and use
- Used for learning/testing
- Example: VirtualBox, VMware Workstation

## **Containers**

Lightweight isolated environment for running applications.

### **Function:**

- Share host OS kernel
- Do NOT need full OS
- Package app + dependencies together

### **Advantages:**

- Very fast startup
- Uses fewer resources
- Easy deployment
- Scalable
- Consistent on all machines

Example Tool: Docker

### **Limitation:**

- Must match host OS type
- (Linux container → Linux system)

## **Cloud Computing**

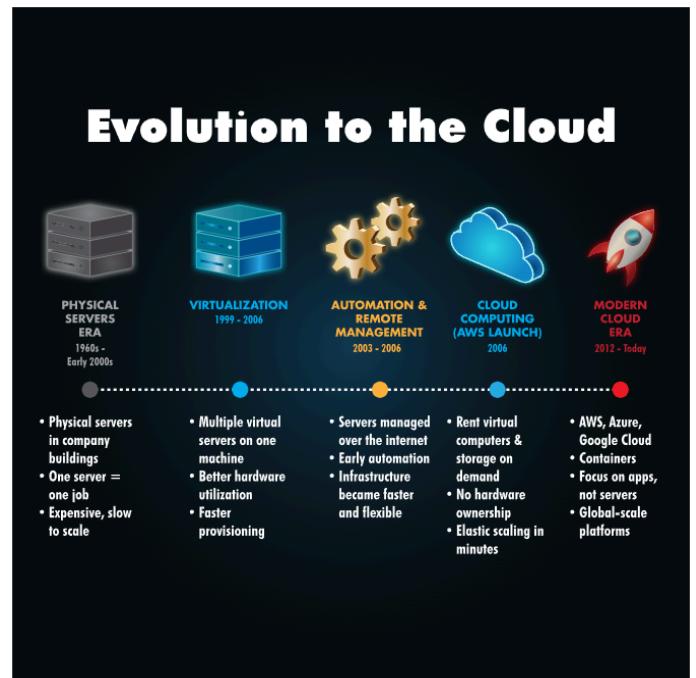
### **What is Cloud Computing**

Using computing resources (server, storage, apps) over the internet instead of your own computer.

Example: Google Drive, AWS, Azure

## Benefits of Cloud

- Scalability: Increase/decrease resources anytime
- On-demand: Create servers instantly
- Pay as you use: No upfront hardware cost
- High availability: Always online
- Security: Cloud provider protects infra
- Global access: Access from anywhere



## Types of Cloud (Deployment Models)

### Public Cloud

- Anyone can use
- Cheap and scalable
- Example: AWS, Google Cloud
- Used by startups & apps

### Private Cloud

- Only one organization
- More control & security
- Used by banks, govt

### Hybrid Cloud

- Mix of public + private
- Sensitive data private
- Other apps public

## Cloud Service Models

### 1. IaaS (Infrastructure as a Service)

You rent:

- Virtual machines
- Storage
- Network

You manage:

- OS
- Apps
- Example: AWS EC2

## 2. PaaS (Platform as a Service)

Provider manages:

- Hardware
- OS
- Runtime

You manage:

- Application code only
- Example: Heroku, Google App Engine

## 3. SaaS (Software as a Service)

Ready software via the internet.

You just use it:

- Gmail
- Zoom
- Facebook

The provider manages everything.

IaaS → control everything

PaaS → focus on coding

SaaS → just use software



## Major Cloud Providers

- AWS (most popular)
- Microsoft Azure
- Google Cloud
- Alibaba Cloud
- IBM Cloud
- Oracle Cloud

## Basic Cloud Terminology (AWS)

### **EC2 (Elastic Compute Cloud)**

- Virtual computer in cloud
- Like real PC: CPU, RAM, storage
- Used to run apps & servers
- Creating EC2 = creating a virtual machine

EC2 = cloud server

### **Instance Type**

Shows power of virtual machines.

### **Examples:**

- t2, t3, t3.micro → small/cheap
- m5.large → powerful/expensive
- Bigger instance → more power + more cost
- Smaller instance → less cost + less power

### **Region**

Location where cloud servers are stored.

### **Example:**

- US-East
- Asia
- Europe

Choose region close to users for:

- Better speed
- Lower delay

The screenshot shows the Twinklme Cloud Console interface. At the top, there's a navigation bar with 'OVERVIEW', 'Dashboard', 'COMPUTE', 'Virtual Machines (EC2)', 'COSTS', and 'Billing'. Below this is a summary card for 'Cloud Virtual Machines Console (EC2)'. It shows 'Region: us-east-1' with a yellow arrow pointing to it. The card displays four metrics: 'Virtual Machines (EC2)' with 'Total instances' (2), 'Running' (2), 'Stopped' (0), and 'Estimated Cost' (20.0 credits/month). Below the summary card is a table titled 'Instances' showing two running instances: 'web-1' and 'db-1', both of type 't3.micro' and status 'running'. A tip at the bottom says 'Tip: Stop unused VMs to reduce your bill.' On the right side, there's a 'Create Virtual Machine' form with fields for 'Name' (my-app-server), 'Type' (t3.micro), and 'Status' (running), with a 'Create VM' button.

### **Creating Virtual Machines**

To run an application in cloud:

1. Create EC2 instances
2. Give name
3. Choose instance type
4. Start (running state)

Each instance = one virtual computer.

The screenshot shows a 'Create Virtual Machine' dialog box. It has a title 'Create Virtual Machine' and a subtitle 'Launch a new Instance'. There are three main input fields: 'Name' (containing 'application-interface'), 'Type' (containing 't3.micro (10 credits / month)'), and 'Status' (containing 'running'). At the bottom is a large green 'Create VM' button.

## Billing

### Cloud cost depends on:

- Instance type
- Running time
- Number of servers

Running VM → costs money

Stopped VM → no cost (mostly)

The screenshot shows a 'Billing' interface with the following details:

**Billing**  
Only running VMs add cost

NAME	TYPE	STATUS	MONTHLY COST
web-1	t3.micro	running	10.0
db-1	t3.micro	running	10.0
application-interface	t3.micro	running	10.0
study-machine1	m5.large	running	70.0
study-machine2	m5.large	running	70.0

Total estimated: 170.0 credits / month