Docker

### 1️⃣ ****One Tier Application (1990)****

* All components (Frontend, Backend, Database) are combined in a single codebase.
* Example: A desktop application where everything runs together.
* 🧑‍💻 **Users:** Initially designed for ~1000 users.

### 2️⃣ ****Monolithic Three-Tier Architecture (2004)****

* **Tiers:**
  + **Frontend**: React, Angular, Next.js
  + **Backend**: Python, Java, .NET
  + **Database**: SQL Server, MySQL
* All components tightly coupled (Sab kuch juda hua hai).
* **Problems:**
  + A failure in one part can bring down the entire application.
  + Hard to scale specific components independently.
  + Computer restart time is high as everything is packaged together.

### 3️⃣ ****Microservice Three-Tier Architecture (2011)****

* Each functionality (Cart, Accounts, Orders, Wishlist) is its own service.
* Each service runs on separate computers or VMs (2 GB or 4 GB allocated).
* **Microfrontends:** Frontend divided per functionality (Cart Frontend, Accounts Frontend, etc.).
* **Problems:**
  + Memory wastage as each service often over-allocates resources.
  + Restart time for services is high → Leads to higher downtime.

## ⚡ **Scalability**

* As user load increases (10,000 → 100,000 users), the number of computers (VMs) increases.
* Unit job size grows → More DevOps engineers needed to manage.

## 🏗 **Virtualization**

* **Setup:**
  + Hardware: 250 GB RAM, 40 core processor
  + Host OS → Hypervisor (e.g., VirtualBox)
  + Multiple VMs (each with Guest OS + Middleware + Code)
* **Problems:**
  + Guest OS in every VM causes memory wastage.
  + Restarting VMs takes significant time.
  + Inefficient use of hardware resources.

## 🐳 **Containerization**

* **Setup:**
  + Host OS → Linux Kernel (Ubuntu) + Docker/Podman/Rocket (container engine)
  + Containers share the OS kernel but run isolated environments.
* **Advantages:**
  + No Guest OS overhead → Efficient memory use.
  + Very fast startup/shutdown (Slim, Trim, Fast Restart).
  + Minimal downtime.
* **Linux Feature:**
  + **C-groups (Control groups):** Used to allocate CPU/memory to specific processes.

## ⚙ **Automation Tools**

* **Imperative:** az CLI → Commands define how to achieve a state.
* **Declarative:** Terraform, Bicep, ARM templates → Define what the end state should be, tool figures out how.
* **Usage Guidance:**
  + Small tasks → Automation like az CLI
  + Big tasks → Use Terraform, Bicep, ARM for better manageability

## 📦 **Containers & Operations**

* Build a package (image) containing middleware + code (e.g., nginx + app code).
* Example:
  + **Dev Team:** Creates Dockerfile with nginx 1.65 + code → Builds image.
  + **Ops Team:** Deploys the image → Easy version management (e.g., nginx 2.0 image).

## ✅ **Summary of Key Problems (Before Containerization)**

* Wastage of memory due to unnecessary overhead.
* High restart time → More downtime.
* Hard to scale specific components without scaling entire app/VM.
* More operational complexity → Higher demand for DevOps resources.

## 📝 **Final Takeaways**

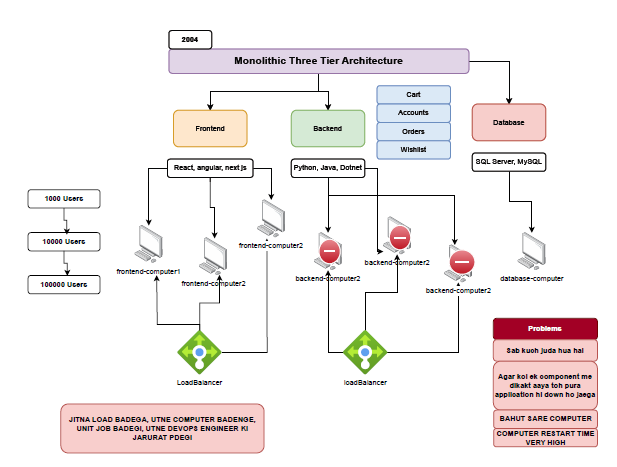
👉 **Containerization** provides:

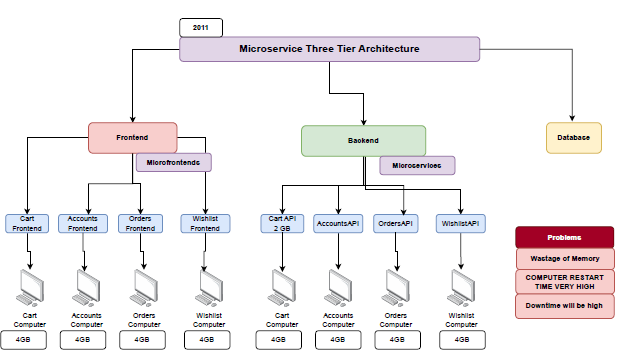
* Better resource utilization
* Faster deployments and scaling
* Easier management of microservices and micro frontends
* Lower operational cost and downtime

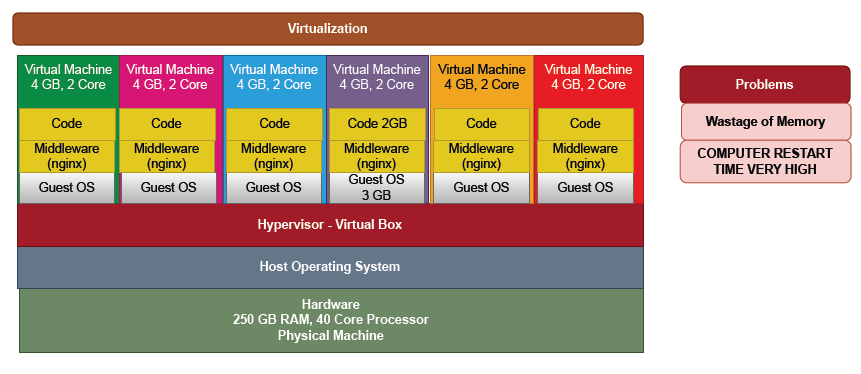
👉 **DevOps Tools:**

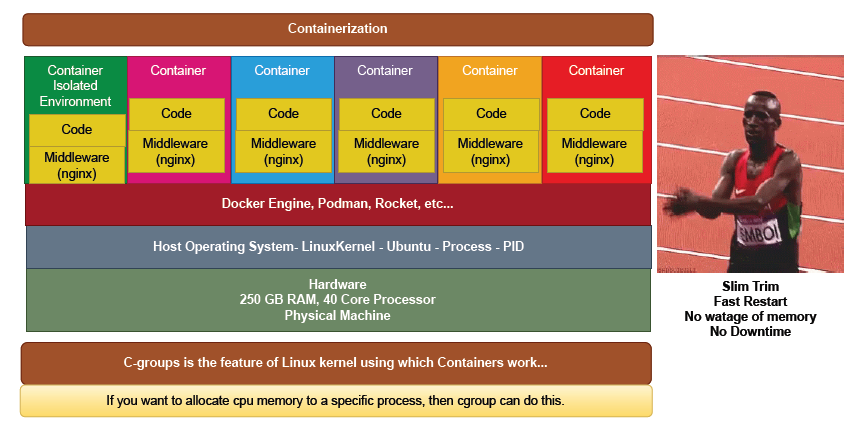
* Use **Imperative tools (az CLI)** for quick fixes or one-off tasks.
* Use **Declarative tools (Terraform, Bicep, ARM)** for managing large, complex environments consistently.











# 🚢 ****Docker Essentials — Batch 16-2****

### 📌 ****Where to Install Docker?****

* ✅ **Own Laptop**
  + Suitable for practice.
  + Install Docker Desktop (Windows) / Docker Engine (Linux).
* ✅ **Virtual Machine**
  + Recommended: 4-core CPU, 8 GB RAM minimum.
  + OS: Linux (preferred) or Windows.
  + Note: Small VM may struggle (performance issues).
* ✅ **Cloud**
  + Azure Subscription → Use VMs or services.
* ✅ **Free Playground**
  + [Play with Docker](https://labs.play-with-docker.com/)

### 📌 ****How to Install Docker?****

* **Windows:** Search and download Docker Desktop → Install.
* **Linux:** Search "Docker on Ubuntu" → Use apt to install.
* **Tip:** The Docker machine is resilient and designed for containers.

### 📌 ****What are Docker Images?****

* Read-only template used to create containers.
* Example: ubuntu, nginx, ubuntu-with-nginx
* Source:
  + **DockerHub:** Marketplace for Docker images.
  + Think of it like a library (images = books).

### 📌 ****What are Docker Containers?****

* Running instance of an image.
* Lightweight, isolated, fast to start/stop.
* Compared to VMs: Less resource usage (no full OS per container).

### 📌 ****Basic Docker Commands****

* **Download Image**
* docker pull <image-name>
* e.g., docker pull nginx
* **Run Container**
* docker run <image-name>
* e.g., docker run nginx
* **Manage Container**
* docker start <container-id>
* docker stop <container-id>
* docker delete <container-id>

### 📌 ****Dockerfile****

* Blueprint to build custom Docker images.
* Includes instructions for:
  + Base image
  + Commands to run
  + Copy files
  + Expose ports

### 📌 ****Advanced Topics****

* **Volumes:** Persistent storage for containers.
* **Networking:** Connect containers to each other and external systems.

### 📌 ****Troubleshooting****

* Identify errors through logs:
* docker logs <container-id>
* Check container status:
* docker ps -a

### 📌 ****Key Concepts****

* **Docker vs VM**
  + VM: Heavy, slow restart, wastes resources.
  + Docker: Lightweight, quick restart, efficient.
* **Kubernetes**
  + Acts as a steering wheel for the Docker ship (manages containers at scale).

### 📌 ****Requirement Example****

👉 Create a Docker container using nginx: 1️⃣ Pull the image:

docker pull nginx

2️⃣ Run the container:

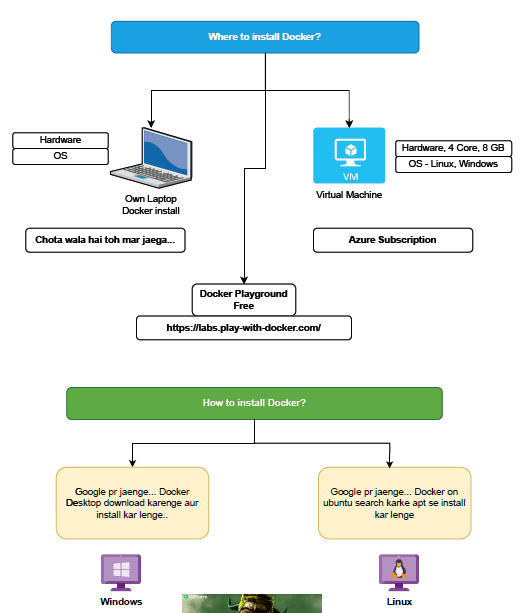
docker run nginx

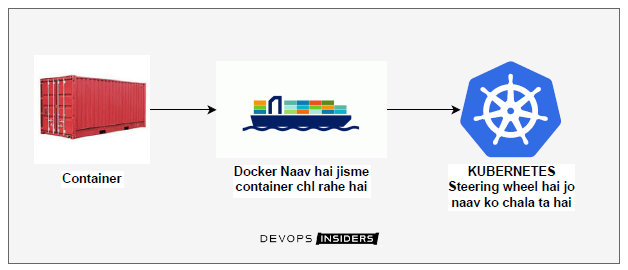
3️⃣ Access via browser: Visit: localhost:80

⚠ **Tip:** “Okhli pr sar nahi marna hai” — Don’t overcomplicate simple Docker tasks.

## 🚀 **Conclusion**

Docker simplifies app deployment by using lightweight, fast containers instead of heavy VMs, making it ideal for modern DevOps and cloud workflows.





# 🚀 ****Docker Detailed Notes — Batch 16-2****

### 📌 ****Docker Command Syntax****

* General format:
* docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
* docker run: Command to create and start a new container from an image.
* **IMAGE** → The base image (e.g., nginx, ubuntu).
* **OPTIONS** → Custom settings (e.g., container name, ports).
* **COMMAND [ARG...]** → Commands passed to the container.

👉 **Note:** Everything inside [] (square brackets) is **optional**.

### 📌 ****Example Command****

docker run --name pappu -d nginx

* --name pappu → Names the container pappu.
* -d → Detached mode (runs in background).
* nginx → The image used.

### 📌 ****Docker Command Conventions****

* 🔹 Small letters → Constants (e.g., run, pull, start)
* 🔹 CAPITAL letters → Variables/placeholders (e.g., IMAGE, COMMAND)

### 📌 ****Examples of Docker Run****

* Simple:
* docker run nginx

Runs an nginx container in foreground mode.

* With options:
* docker run --name mynginx -p 8080:80 -d nginx
  + Names the container mynginx
  + Maps host port 8080 to container port 80
  + Detached mode

### 📌 ****Docker Usage in Ubuntu Terminal****

* Example command you would type:
* docker run nginx
* Opens terminal container with nginx running.

## 📝 **Key Concepts**

✅ **Docker Run:** Launches a container from a specified image.

✅ **Options (Flags)**:

* --name → Give your container a custom name.
* -d → Run container in background (detached mode).
* -p → Map ports from host to container.

✅ **Optional Elements:** Indicated with [] in command syntax.

✅ **Constants vs Variables:**

* Constants → Written in small letters → Actual commands.
* Variables → Written in CAPITAL letters → Values to supply (e.g., IMAGE name).

Requirement: - How to pull an image and How to create a docker container using Nginx...?

* 1. Open docker desktop or VM where you want to run the docker container
  2. Open the docker hub where all docker images are present <https://hub.docker.com/>
  3. Search Nginx and copy its command >> docker pull nginx
  4. Open cmd /PowerShell and paste this command >> docker pull nginx
  5. To check this docker image has created or not >> docker images
  6. Ab is image se ek container banate hai >>
  7. docker run -d -p 8080:80 --name mycontainer nginx
  8. docker ps >> is command se check krte hai ki docker container bna ki nhi
  9. <http://localhost:8080> >> browser me hit krenge aur Nginx run kr rha hoga

## Requirement: - How to pull an image and How to create a docker container using **Netflix-clone**

...?

* 1. Open docker desktop or VM where you want to run the docker container
  2. Open the docker hub where all docker images are present <https://hub.docker.com/>
  3. Search Nginx and copy its command >> docker pull devopsinsiders/netflix-clone
  4. Open cmd /PowerShell and paste this command >> docker pull devopsinsiders/netflix-clone
  5. To check this docker image has created or not >> docker images
  6. Ab is image se ek container banate hai >>
  7. docker run -d -p 8085:80 --name mycontainer01 devopsinsiders/netflix-clone
  8. docker ps >> is command se check krte hai ki docker container bna ki nhi
  9. http://localhost:8085 >> browser me hit krenge aur Netflix run kr rha hoga