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**

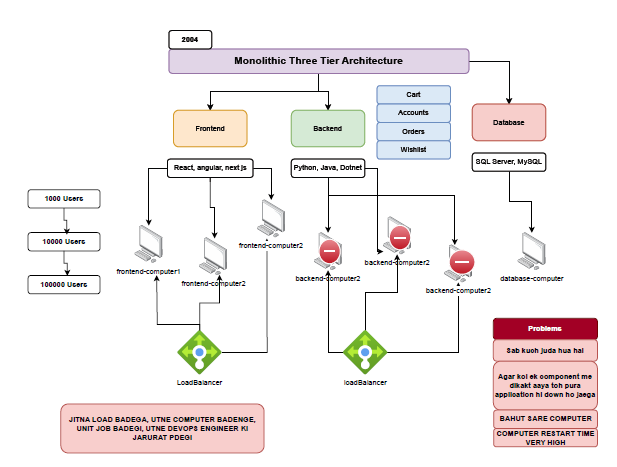
👉 **Containerization** provides:

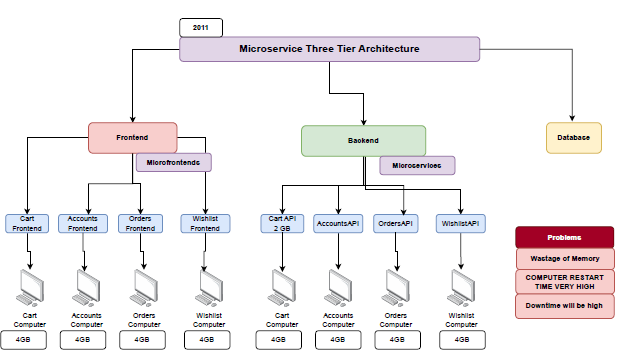
* Better resource utilization
* Faster deployments and scaling
* Easier management of microservices and micro frontends
* Lower operational cost and downtime

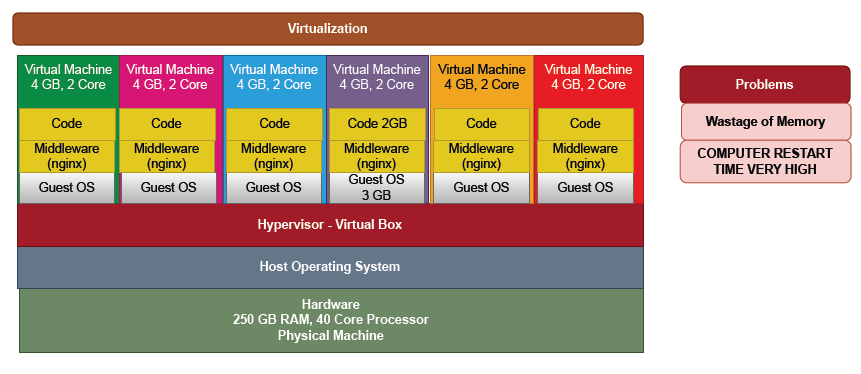
👉 **DevOps Tools:**

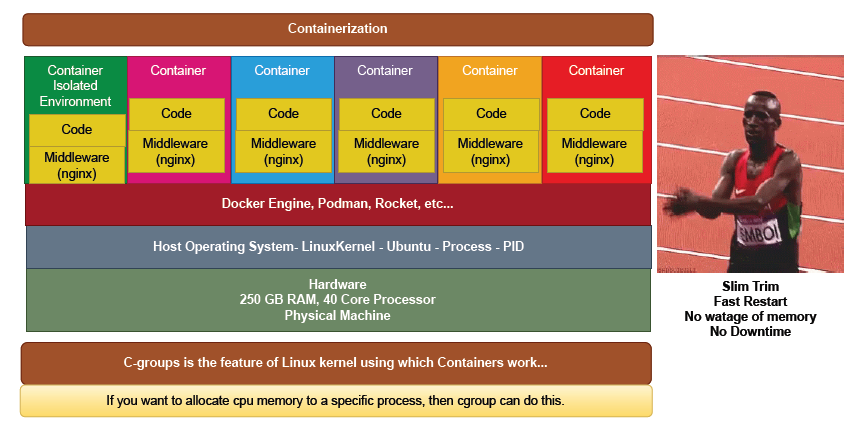
* Use **Imperative tools (az CLI)** for quick fixes or one-off tasks.
* Use **Declarative tools (Terraform, Bicep, ARM)** for managing large, complex environments consistently.











# 🚢 ****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:

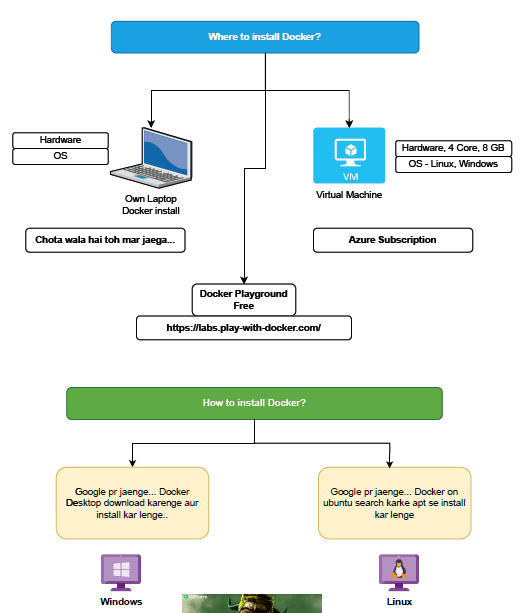
docker run nginx

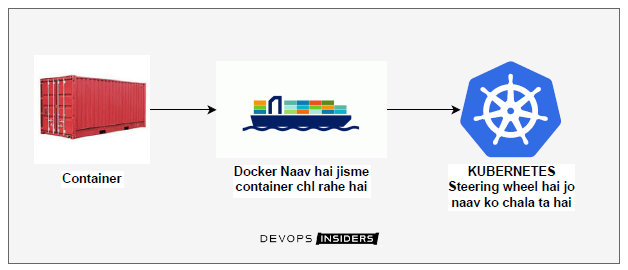
3️⃣ Access via browser: Visit: localhost:80

⚠ **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).

Website /application code

Docker File

Docker image

Docker container

Small VM

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

If we need to inspect (visit or khoj-been) inside the container then --------

* 1. docker inspect <container-id | container name>
  2. we will get the all information of the container inside this & IP information also
  3. docker network inspect bridge >> bridge is a default network >> subnet information

## 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

Requirements: Run a website (Starbucks) on docker container?

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. Open the docker hub where all docker images are present <https://hub.docker.com/>
4. Search Nginx latest and copy its command >> docker pull nginx:latest
5. Search Firefox latest and copy its command >> docker pull linuxserver/firefox
6. To check this docker image has created or not >> docker images
7. Ab is image se ek container banate hai >
8. docker run -d nginx
9. docker ps >> is command se check krte Hai Ki docker container bna ki nhi
10. Jaise hm vm ke Andar ssh kr k login krte the waise hi hm container ke Andar bhi ja sakte hai exec ke through
11. docker exec <container id> ls >> the output will show same as a inside the vm

bin

boot

dev

docker-entrypoint. d

docker-entrypoint.sh

etc

home

lib

lib64

media

mnt

opt

proc

root

run

sbin

srv

sys

tmp

usr

var

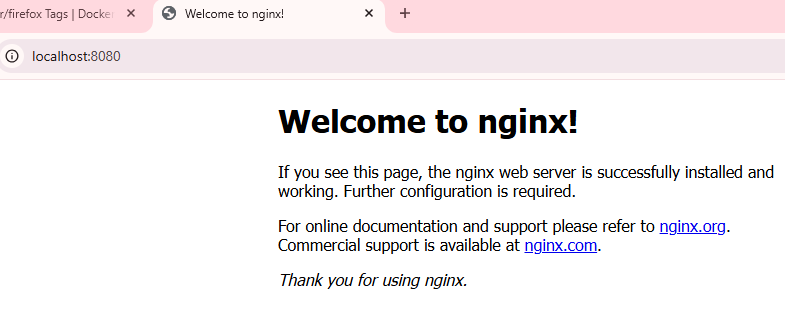
1. docker exec 93230a5ad764 pwd >> to check the path of container
2. docker exec -i -t <container id> bash >> to login that container
3. root@93230a5ad764:/# ls

bin dev docker-entrypoint.sh home lib64 mnt proc run srv tmp var

1. boot docker-entry point. d etc lib media opt root sbin sys usr
2. root@93230a5ad764:/# cd /usr/share/nginx/html >> ls >> path Jha per html file rkhi hai nginx ki

50x.html index.html

1. root@93230a5ad764:/# exit
2. Now create a container
3. docker run -d -p 8080:80 --name mycontainer01 nginx
4. docker ps
5. <http://localhost:8080> >> browser me hit krenge aur Nginx run kr rha hoga



Leking hme to website chalani hai apne docker ke Andar to uske liye hme artifacts le aana hoga

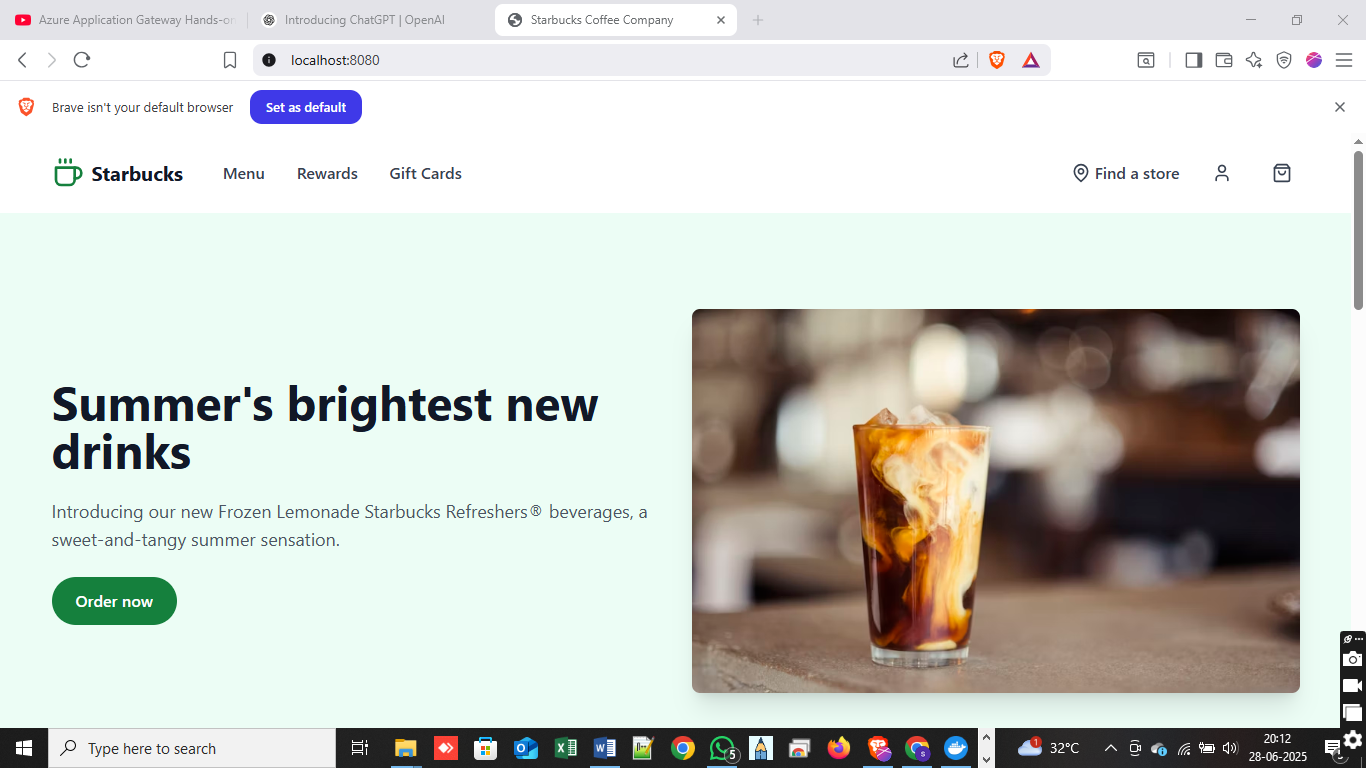
1. <https://github.com/devopsinsiders/starbucks-clone> >> artifacts yha pde hai
2. C:\Users\Sumantra>docker exec -it <container id> bash
3. root@f1a0e9fc66b8:/# git clone https://github.com/devopsinsiders/starbucks-clone.git
4. bash: git: command not found
5. root@f1a0e9fc66b8:/# apt update
6. root@f1a0e9fc66b8:/# apt install git
7. root@f1a0e9fc66b8:/# git –version >> now git installed

git version 2.39.5

1. root@f1a0e9fc66b8:/# git clone https://github.com/devopsinsiders/starbucks-clone.git
2. root@f1a0e9fc66b8:/# ls
3. root@f1a0e9fc66b8:/# cd starbucks-clone/

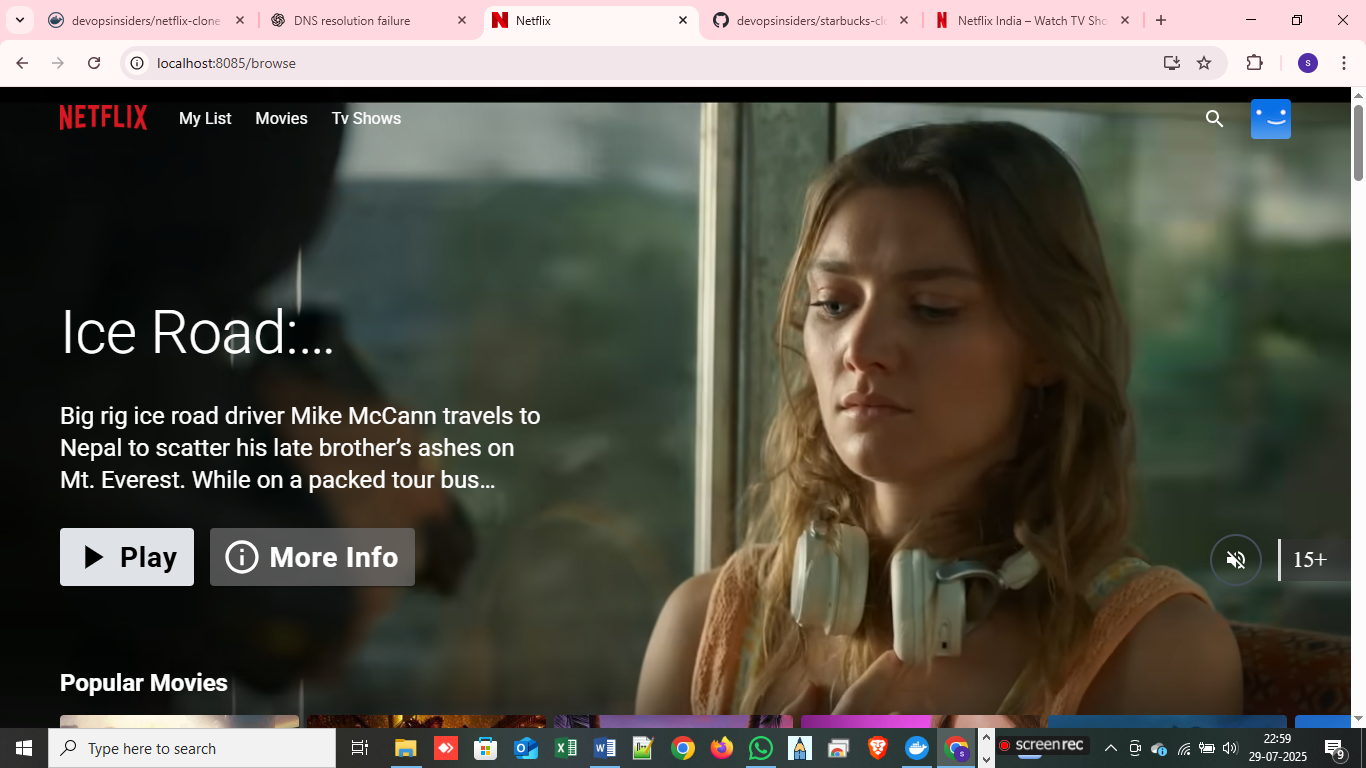
assets index.html

1. root@f1a0e9fc66b8:/starbucks-clone# cd ..
2. root@f1a0e9fc66b8:/# cp -r starbucks-clone/\* /usr/share/nginx/html >> copy html files to html folder
3. root@f1a0e9fc66b8:/# cd /usr/share/nginx/html
4. <http://localhost:8080> >> browser me hit krenge aur Starbucks run kr rha hoga



Agar hmko koi aur website chalani hai same container pr to kya krenge?

1. root@f1a0e9fc66b8:/usr/share/nginx/html# rm -r \*
2. root@f1a0e9fc66b8:/usr/share/nginx/html# ls >> all artifacts are deleted
3. root@f1a0e9fc66b8:/usr/share/nginx/html# cd /home
4. root@f1a0e9fc66b8:/home# git clone <https://github.com/devopsinsiders/StreamFlix.git>
5. root@f1a0e9fc66b8:/home# cp -r StreamFlix/\* /usr/share/nginx/html
6. root@f1a0e9fc66b8:/# cd /usr/share/nginx/html
7. <http://localhost:8080> >> browser me hit krenge aur Starbucks run kr rha hoga



To delete containers one by one

* 1. docker ps
  2. docker stop <container id of running container>
  3. docker ps -a >> to check stopped container details
  4. docker rm -f <container id of running container>

container will get removed one by one

Requirement: Using Firefox image run the container?

1. docker pull linuxserver/firefox
2. docker run -d -p 10000:3000 linuxserver/firefox
3. docker ps
4. <http://localhost:10000>
5. We can check that if another container is formed so we can just copy their host port and when we will hit inside the firefox it will run means the all are interconnected they can communicate without any public IP.

If we need to copy from the local computer to our docker container or container to local computer then------

Its working same as scp command which we were used before to copy artifacts from local to Linux vm

1. docker cp container: source path destination path
2. docker cp source path container: destination path

Docker file

https://docs.docker.com/reference/dockerfile/

Docker can build images automatically by reading the instructions from a Docker file. A Docker file is a text document that contains all the commands a user could call on the command line to assemble an image.

| **Instruction** | **Description** |
| --- | --- |
| [ADD](https://docs.docker.com/reference/dockerfile/#add) | Add local or remote files and directories. |
| [ARG](https://docs.docker.com/reference/dockerfile/#arg) | Use build-time variables. |
| [CMD](https://docs.docker.com/reference/dockerfile/#cmd) | Specify default commands. |
| [COPY](https://docs.docker.com/reference/dockerfile/#copy) | Copy files and directories. |
| [ENTRYPOINT](https://docs.docker.com/reference/dockerfile/#entrypoint) | Specify default executable. |
| [ENV](https://docs.docker.com/reference/dockerfile/#env) | Set environment variables. |
| [EXPOSE](https://docs.docker.com/reference/dockerfile/#expose) | Describe which ports your application is listening on. |
| [FROM](https://docs.docker.com/reference/dockerfile/#from) | Create a new build stage from a base image. |
| [HEALTHCHECK](https://docs.docker.com/reference/dockerfile/#healthcheck) | Check a container's health on startup. |
| [LABEL](https://docs.docker.com/reference/dockerfile/#label) | Add metadata to an image. |
| [MAINTAINER](https://docs.docker.com/reference/dockerfile/#maintainer-deprecated) | Specify the author of an image. |
| [ONBUILD](https://docs.docker.com/reference/dockerfile/#onbuild) | Specify instructions for when the image is used in a build. |
| [RUN](https://docs.docker.com/reference/dockerfile/#run) | Execute build commands. |
| [SHELL](https://docs.docker.com/reference/dockerfile/#shell) | Set the default shell of an image. |
| [STOPSIGNAL](https://docs.docker.com/reference/dockerfile/#stopsignal) | Specify the system call signal for exiting a container. |
| [USER](https://docs.docker.com/reference/dockerfile/#user) | Set user and group ID. |
| [VOLUME](https://docs.docker.com/reference/dockerfile/#volume) | Create volume mounts. |
| [WORKDIR](https://docs.docker.com/reference/dockerfile/#workdir) | Change working directory. |

* Multistage docker file in that we are using lighter images

If we need to write the docker file for the Starbucks clone file then we will clone the git repo where code is present.

1. Git clone <https://github.com/devopsinsiders/starbucks-clone.git> in your local computer
2. Open the clone code into the vs code
3. Create a new file file name will be >> Dockerfile
4. Now we will write the dockerfile

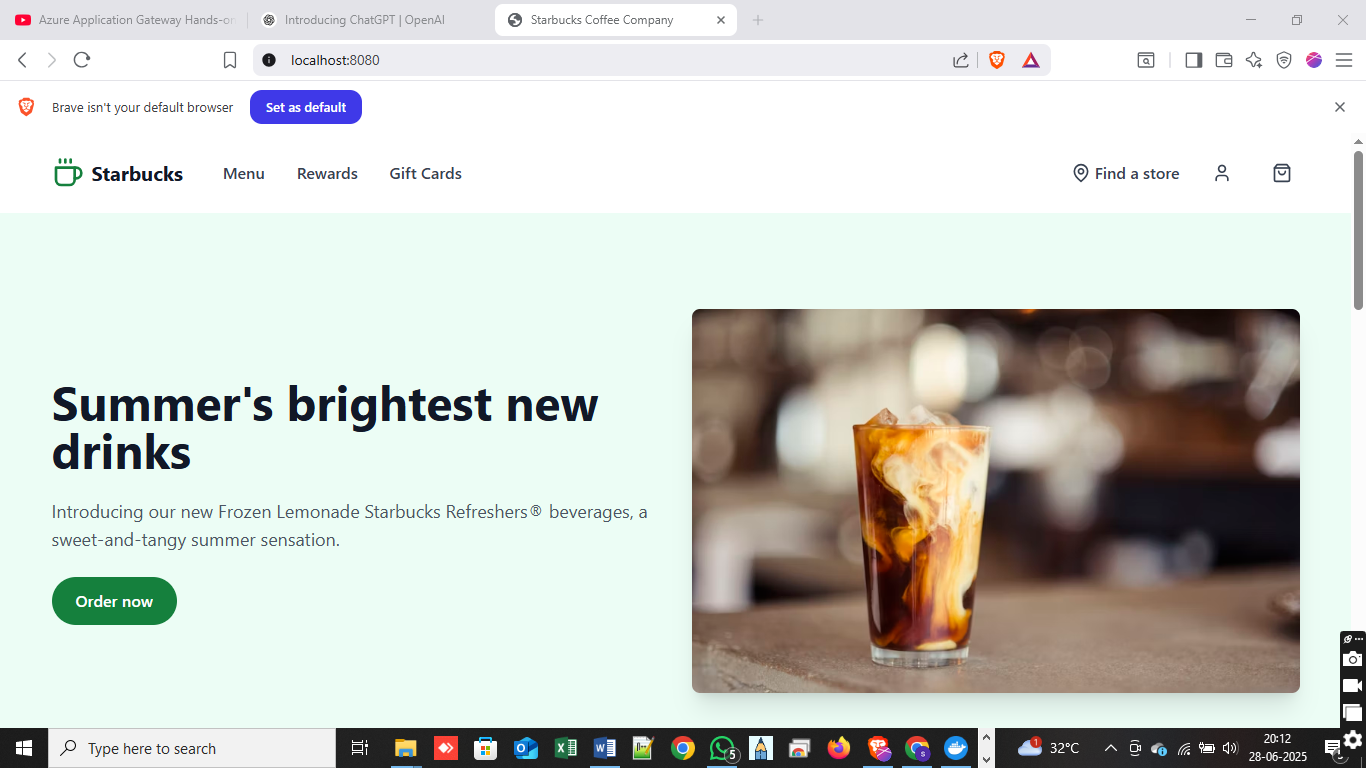
# we are using the official nginx image

FROM nginx:latest

# copy the starbucks clone into the container path file

COPY . /usr/share/nginx/html

1. docker buildx build PATH | URL | - >> For example to make the customize image command
2. docker build -t starbucks\_image:v1 . >> to build up the customize image for the container where **.** is current path
3. docker images
4. docker run -d - - name starbucks -p 9900:80 starbucks\_image:v1
5. docker ps
6. <http://localhost:9900>

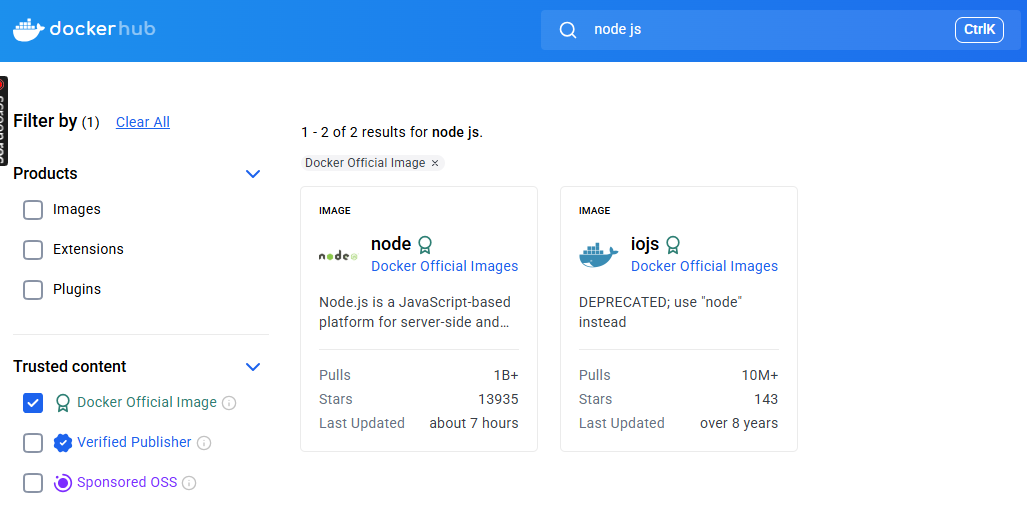


Requirement: - We have a todo application and their readme file so using this how we can create an customize image for the container?

1. Open <https://github.com/devopsinsiders/ReactTodoUIMonolith>
2. Open readme file >> install node.js >> version will be 16.x >> these are the prerequisites
3. Second required was shown in Readme file >> npm install >> npm run build
4. When the artefacts were created then we were moved the code from local to Linux vm using SCP -r command.
5. These things were done in manually

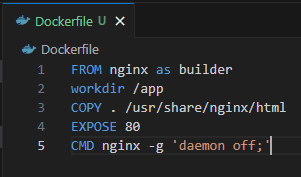
Requirement: If the same work we need to do by using docker and need to automate it then we will use the docker file.

1. Clone the repo in local computer git clone <https://github.com/devopsinsiders/ReactTodoUIMonolith>
2. Open with VS code
3. Create a file named as Dockerfile
4. Open Docker file the write the docker file
5. Open **README** file Open to side in VS code
6. Requirement is image required node.js V16.x
7. Visit to docker hub <https://hub.docker.com/>



Docker file for applications

Case 1:- We have a application Starbucks clone and we need to create an image for this so we need to write the docker file fist so with help of this image we can create the container where our Starbucks clone website will run.



Using docker build command hm image ko build kr lenge

Docker build -t starbucks:v1 .

Image build hone ke baad usko khi rkhna hoga to uske liye do tareeka hai

1. Docker hub (Public registry)
2. ACR (Azure container registry) (Private registry)

ACR creation on Azure: - create an Azure container registry on Azure cloud first.

Process to push the image on ACR is

1. Docker build
2. Docker tag
3. Docker login
4. Docker push

On Virtual machine where docker is installed

1. Az acr login -n <my acr registry name>
2. Sudo docker images
3. Sudo docker tag <source image>
4. Sudo docker tag todoui:v1 <registry server name> / todoui:v1 (source image name)
5. Sudo docker push <acr registry name> / <image name>

Now image pushed to ACR registry

Router ek IP allocate krta hai NIC card ko jo ki device pr lgi hoti hai NIC.

**Network interface card: -**

1. **Physical address (MAC address)**
2. **IP address (with the help of IP address we can able to connect with the internet)**

**Router: -**

1. **Privately connectivity provides krta Hai**
2. **IP allocate krta Hai**

Router ek IP allocate krta Hai NIC card ko same kaam docker bhi krta Hai

C-group feature ka use krke docker Network banata hai

1. Sudo docker Network –help
2. Sudo docker network ls

Bridge Network ek Private Network banata hai aur router jaisa kaam krta hai aur sare container default me bridge network me bnta hai.

We can check it by using docker inspect command

Agar khud se Network banana hai to (User define private Network banana padega)

1. Docker network create <custom network name> for example Sumantra
2. Docker run - -network Sumantra -d -p 3000:80 devopsinsiders/Netflix

Ab ek container ke andar jakar dusre container ki IP ping krte hai isse ye pta chalta hai ki do private network aapas me communicate nhi kar skte hai.

1. Sudo docker exec -it <container id > bash
2. Ping <ip>

Leking agar dono container same network me hai to dono aapas me communicate kar payenge.