Introduction to Docker

- Docker is an open-source platform designed to simplify the development, deployment, and management of applications by using **containerization**.
- It allows you to package an application and its dependencies into a lightweight, portable container that can run consistently across different environments, from a developer's local machine to a production server.

Key Concepts in Docker

1. **Containers**
    - Containers are lightweight and isolated units that run applications.
    - They include everything the application needs: libraries, dependencies, and configuration files.
    - Unlike virtual machines, containers share the host operating system's kernel, making them more efficient and faster.
2. **Images**
    - An image is a lightweight, standalone, and executable package that contains everything needed to run a piece of software.
    - Containers are instances of images.
    - Images are created using **Dockerfiles**, which define the container's environment.

3. **Docker Engine**
    - The core of Docker, responsible for building, running, and managing containers.
4. **Docker Hub**
    - A public repository where users can find and share Docker images.
    - It provides pre-built images for popular software like NGINX, MySQL, and Python.
5. **Docker Compose**
    - A tool for defining and managing multi-container Docker applications.
    - Uses a YAML file (`docker-compose.yml`) to configure services, networks, and volumes.

Advantages of Docker

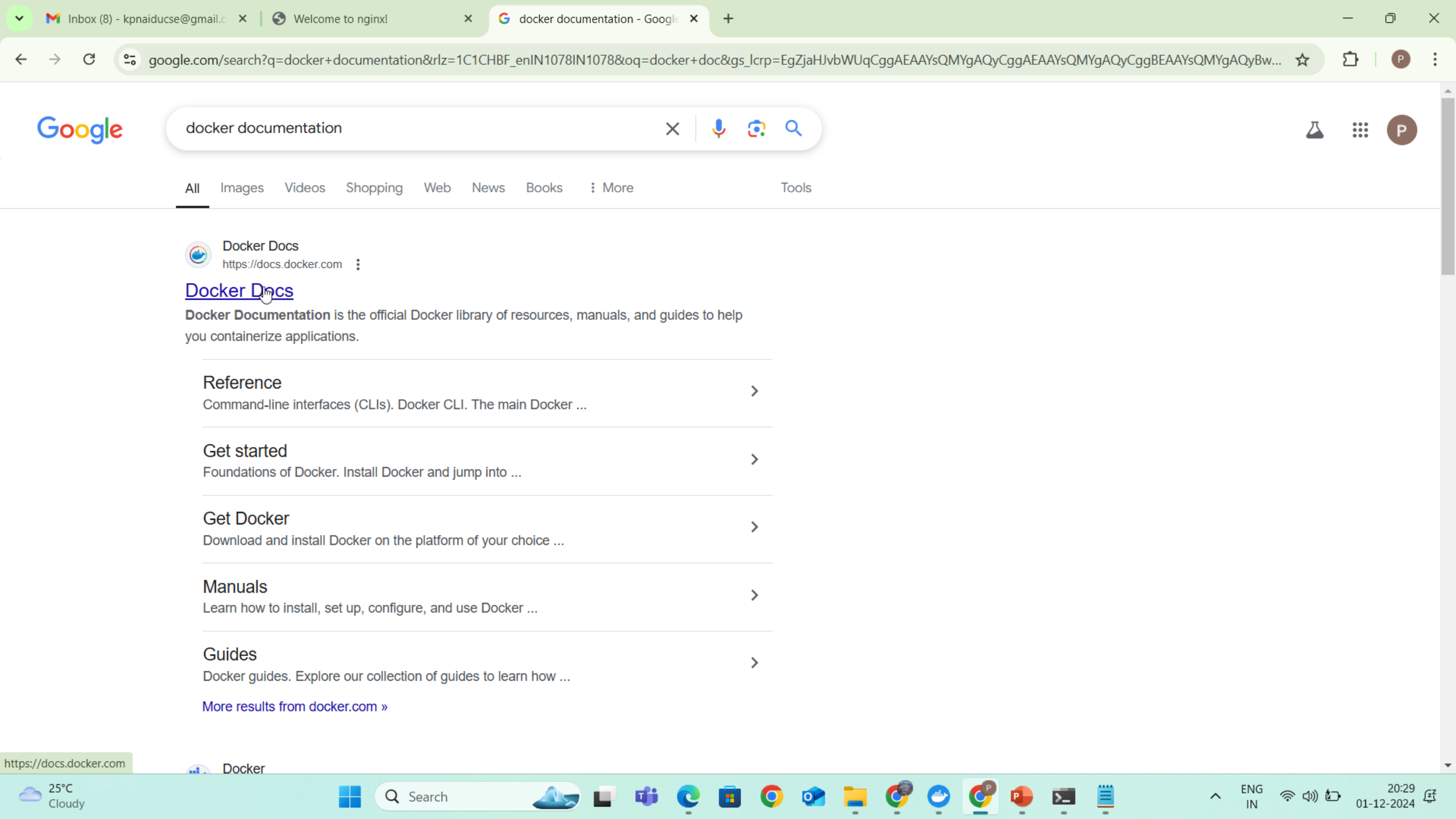
1. **Portability**
    - "Build once, run anywhere": Docker containers can run consistently on any platform, whether it's a developer's laptop, a test environment, or a cloud server.
2. **Efficiency**
    - Containers are lightweight compared to virtual machines because they share the host OS kernel.
3. **Isolation**
    - Each container operates independently, ensuring there’s no conflict between applications or their dependencies.
4. **Scalability**
    - Docker works seamlessly with orchestration tools like Kubernetes, enabling efficient scaling of containerized applications.
5. **Faster Deployment**
    - Containers start in seconds, significantly reducing the time to deploy and test applications.

Use Cases of Docker

1. **Development and Testing**
    - Developers can create consistent environments to eliminate "works on my machine" issues.
2. **Microservices**
    - Docker is ideal for microservices architectures, where applications are split into smaller, independently deployable services.
3. **Continuous Integration/Continuous Deployment (CI/CD)**
    - Automates application builds, tests, and deployments.
4. **Cloud Deployment**
    - Simplifies moving applications to and from the cloud.
5. **Big Data and Machine Learning**
    - Use Docker to package tools and dependencies for data processing and model training.

how to install docker on windows 11

https://www.youtube.com/watch?v=bw-bMhlhcpg


Inbox (8) - kpnaiducse@gmail.c
Welcome to nginx!
docker documentation - Google
google.com/search?q=docker+documentation&rlz=1C1CHBF_enIN1078IN1078&oq=docker+doc&gs_lcrp=EgZjaHJvbWUqCggAEAAYsQMYgAQyCggAEAAYsQMYgAQyCggBEAAYsQMYgAQyBw...
Google
docker documentation
All
Images
Videos
Shopping
Web
News
Books
More
Tools
Docker Docs
https://docs.docker.com
Docker Docs
Docker Documentation is the official Docker library of resources, manuals, and guides to help you containerize applications.
Reference
Command-line interfaces (CLIs). Docker CLI. The main Docker ...
Get started
Foundations of Docker. Install Docker and jump into ...
Get Docker
Download and install Docker on the platform of your choice ...
Manuals
Learn how to install, set up, configure, and use Docker ...
Guides
Docker guides. Explore our collection of guides to learn how ...
More results from docker.com »
Docker
https://docs.docker.com
25°C
Cloudy
Search
ENG
IN
20:29
01-12-2024

Inbox (8) - kpnaiducse@gmail.c | Welcome to nginx! | Docker Docs

docs.docker.com

More Docker. Easy Access. New Streamlined Plans. Learn more. →

dockerdocs Get started Guides Manuals Reference Search ^K Ask AI

Get Docker

Learn how to install Docker for Mac, Windows, or Linux and explore our developer tools.

Get Docker

Guides

Get started and learn how Docker can optimize your development workflows.

Manuals

Learn how to install, set up, configure, and use Docker products.

Reference

Browse the CLI and API documentation.

25°C Cloudy

Search

ENG IN

20:29 01-12-2024

Get started
Get Docker
What is Docker?
Introduction
Docker concepts
Docker workshop
Educational resources

best installation path for you.

> **Docker Desktop terms**
>
> Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription.

Docker Desktop for Mac

A native application using the macOS sandbox security model that delivers all Docker tools to your Mac.

Docker Desktop for Windows

A native Windows application that delivers all Docker tools to your Windows computer.

Docker Desktop for Linux

A native Linux application that delivers all Docker tools to your Linux computer.

> **Note**
>
> If you're looking for information on how to install Docker Engine, see Docker Engine installation

Edit this page
Request changes

Install Docker Desktop on Windows

> **Docker Desktop terms**
>
> Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription.

This page contains the download URL, information about system requirements, and instructions on how to install Docker Desktop for Windows.

Docker Desktop for Windows - x86_64

Docker Desktop for Windows - Arm (Beta)

For checksums, see Release notes

System requirements

> **Tip**
>
> **Should I use Hyper-V or WSL?**

> Docker Desktop's functionality remains consistent on both WSL and Hyper-V, without a preference for either architecture. Hyper-V and WSL have their own advantages and disadvantages, depending on your specific set up and your planned use case.

WSL 2 backend, x86_64 | Hyper-V backend, x86_64 | WSL 2 backend, Arm (Beta)

- WSL version 1.1.3.0 or later.
- Windows 11 64-bit: Home or Pro version 22H2 or higher, or Enterprise or Education version 22H2 or higher.
- Windows 10 64-bit: Minimum required is Home or Pro 22H2 (build 19045) or higher, or Enterprise or Education 22H2 (build 19045) or higher.
- Turn on the WSL 2 feature on Windows. For detailed instructions, refer to the Microsoft documentation.
- The following hardware prerequisites are required to successfully run WSL 2 on Windows 10 or Windows 11:
  - 64-bit processor with Second Level Address Translation (SLAT)
  - 4GB system RAM
  - Enable hardware virtualization in BIOS. For more information, see Virtualization.

Edit this page
Request changes

Table of contents

The full form of **WSL** is **Windows Subsystem for Linux**.

It is a compatibility layer developed by Microsoft that allows users to run a Linux operating system (like Ubuntu, Debian, or Kali) directly on a Windows machine without the need for a virtual machine or dual-boot setup.

> **NGINX** (pronounced "engine-x") is an open-source web server that also functions as a reverse proxy, load balancer, and HTTP cache. It is widely used for its high performance, scalability, and lightweight architecture. NGINX was initially created by Igor Sysoev in 2004 to address the **C10k problem**—handling 10,000 simultaneous connections efficiently.

NGINX vs. Apache

Feature	NGINX	Apache
Architecture	Event-driven, asynchronous	Process/thread-based
Performance	Better for high concurrency	Slower under heavy load
Configuration	Complex but flexible	Easier for beginners
Use Case	Modern, high-traffic apps	Legacy support

NGINX is a powerful tool for modern web applications, API management, and infrastructure optimization.

Step 5: Enable Hyper-V in Windows

If you're running Docker on Windows, you also need **Hyper-V** enabled:

1. **Open Windows Features**:
    - Search for **Turn Windows features on or off** in the Start menu.
2. **Enable Hyper-V**:
    - Check the box for **Hyper-V** and **Windows Hypervisor Platform**.
    - Click **OK** and restart your computer.

Step 1: Check if Virtualization is Enabled

1. **Open Task Manager**:
    - Right-click on the taskbar and select **Task Manager**, or press `Ctrl + Shift + Esc`.
2. **Go to the Performance Tab**:
    - Click the **Performance** tab.
    - Select **CPU** from the left pane.
3. **Check Virtualization Status**:
    - Look for "Virtualization" on the lower right. If it says **Enabled**, you can skip to using Docker.
    - If it says **Disabled**, you need to enable it in the BIOS.

Step 1: Install Docker Desktop

1. Download and install Docker Desktop.
2. During installation, enable the option to use Docker CLI.

Step 2: Start Docker Desktop

1. Launch Docker Desktop from the Start Menu.
2. Ensure that Docker Desktop is running and the Docker Engine is started.

Step 4: Test Docker Installation

Run the following command to verify Docker is installed and running:

```bash
docker --version
```

To check if Docker is functioning correctly:

```bash
docker run hello-world
```

1. Pull an Image from Docker Hub

```bash
docker pull nginx
```

This downloads the `nginx` image to your local machine.

---

2. List Downloaded Images

```bash
docker images
```

2. List Downloaded Images

```bash
docker images
```

Example output:

```
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
nginx               latest    abc12345xyz    2 days ago      23.2MB
```

3. Run a Container

```bash
docker run -d -p 8080:80 --name my-nginx nginx
```

- `-d`: Run in detached mode (in the background).
- `-p 8080:80`: Map port 8080 on your host to port 80 in the container.
- `--name my-nginx`: Name the container `my-nginx`.

Access the running container at `http://localhost:8080`.

Command Prompt

```
C:\Users\LENOVO>docker ps -a
CONTAINER ID   IMAGE                              COMMAND                  CREATED          STATUS                          PORTS                    NAMES
8f866a28118f   hello-world                        "/hello"                 52 seconds ago   Exited (0) 50 seconds ago                                elegant_wu
b511344af007   docker/welcome-to-docker:latest    "/docker-entrypoint.…"   23 hours ago     Exited (255) 20 minutes ago     0.0.0.0:8088->80/tcp     welcome-to-d
ocker

C:\Users\LENOVO>docker imahes
docker: 'imahes' is not a docker command.
See 'docker --help'

C:\Users\LENOVO>docker images
REPOSITORY                 TAG       IMAGE ID       CREATED         SIZE
docker/welcome-to-docker   latest    c1f619b6477e   12 months ago   18.6MB
hello-world                latest    d2c94e258dcb   19 months ago   13.3kB

C:\Users\LENOVO>docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
2d429b9e73a6: Pull complete
20c8b3871098: Pull complete
06da587a7970: Pull complete
f7895e95e2d4: Pull complete
7b25f3e99685: Pull complete
dffc1412b7c8: Pull complete
d550bb6d1800: Pull complete
Digest: sha256:0c86dddac19f2ce4fd716ac58c0fd87bf69bfd4edabfd6971fb885bafd12a00b
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest

C:\Users\LENOVO>docker images
REPOSITORY                 TAG       IMAGE ID       CREATED         SIZE
nginx                      latest    1ee494ebb83f   4 days ago      192MB
docker/welcome-to-docker   latest    c1f619b6477e   12 months ago   18.6MB
hello-world                latest    d2c94e258dcb   19 months ago   13.3kB

C:\Users\LENOVO>docker run -d -p 8080:80 --name my-nginx nginx
8bc9d0fef5c43c5389464045c78a0cf6ee813a540ea30acf178445b3ca98ae94

C:\Users\LENOVO>
```

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Command Prompt

```
C:\Users\LENOVO>docker --version
Docker version 27.3.1, build ce12230

C:\Users\LENOVO>docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:305243c734571da2d100c8c8b3c3167a098cab6049c9a5b066b6021a60fcb966
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/


C:\Users\LENOVO>docker ps -a
CONTAINER ID   IMAGE                              COMMAND                  CREATED          STATUS                          PORTS                    NAMES
8f866a28118f   hello-world                        "/hello"                 52 seconds ago   Exited (0) 50 seconds ago                                elegant_wu
b511344af007   docker/welcome-to-docker:latest    "/docker-entrypoint.…"   23 hours ago     Exited (255) 20 minutes ago     0.0.0.0:8088->80/tcp     welcome-to-d
ocker

C:\Users\LENOVO>
```

Step 5: Run Docker Commands

Here are examples of common Docker commands:

- **List Docker Containers**:

```bash
docker ps -a
```

- **Run a Container**:

```bash
docker run -d -p 80:80 nginx
```

This runs an NGINX web server in a detached mode and maps port 80 on the container to port 80 on the host.

4. List Running Containers

bash Copy code

```
docker ps
```

Example output:

bash Copy code

```
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
123abc456xyz   nginx     "nginx -g 'daemon of…"   10 seconds ago  Up 10 seconds  0.0.0.0:80
```

5. Stop a Running Container

```bash
docker stop my-nginx
```

---

6. Remove a Container

```bash
docker rm my-nginx
```

7. Remove an Image

bash Copy code

```
docker rmi nginx
```

Additional Tips

- Use `docker-compose` for managing multi-container applications. Create a `docker-compose.yml` file, then run:

  bash Copy code

  ```
  docker-compose up
  ```

Docker Basic Commands

- docker pull image_name : To pull image
- docker run image_name : To create container.
- docker ps: To check running contianers
- docker ps -a: To check all containers
- docker rm container_id: To remove container.
- docker images: To show list of images.
- docker rmi image_name: To remove image.

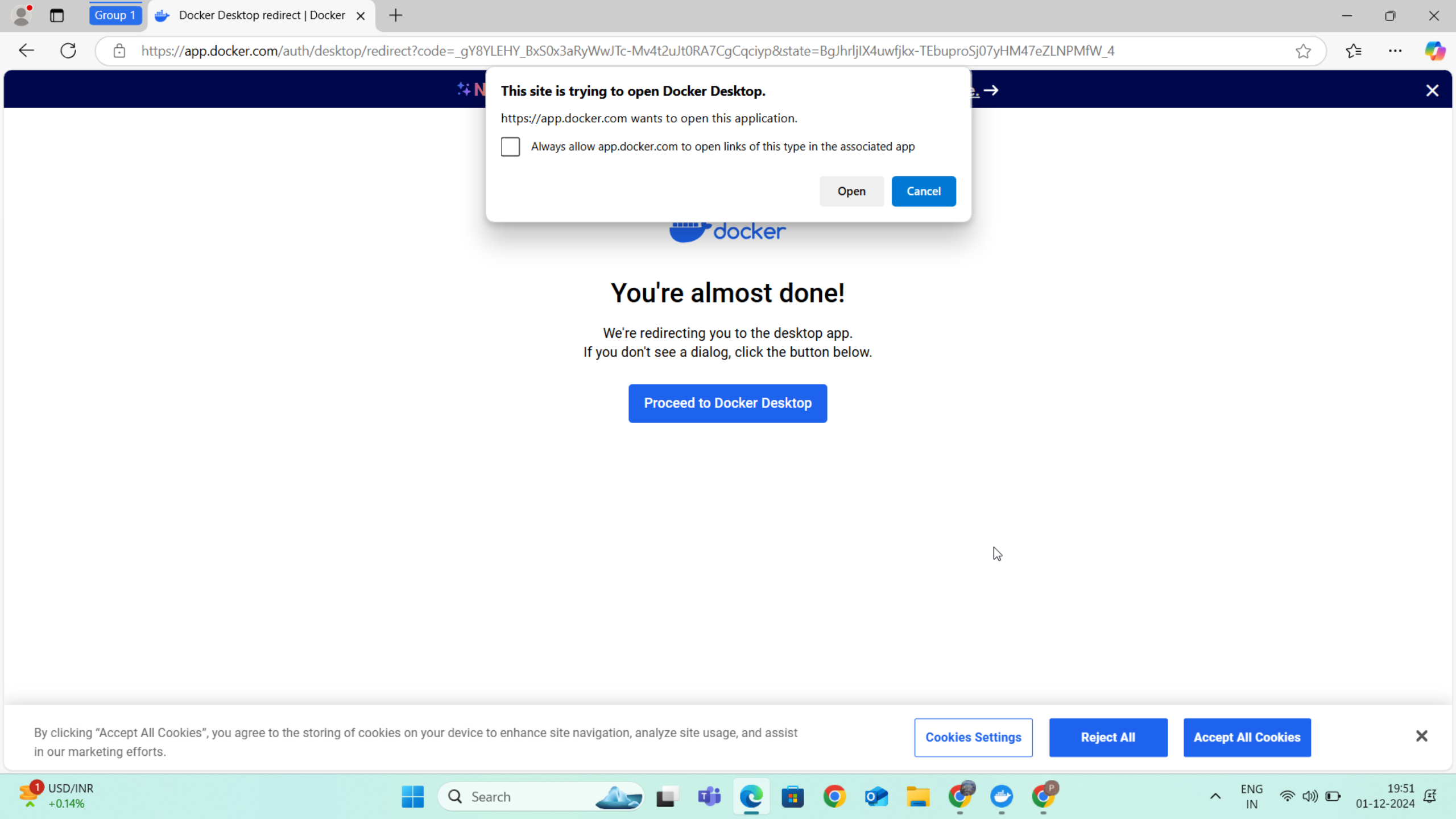
Step 5: Enable Hyper-V in Windows

If you're running Docker on Windows, you also need **Hyper-V** enabled:

1. **Open Windows Features**:
    - Search for **Turn Windows features on or off** in the Start menu.
2. **Enable Hyper-V**:
    - Check the box for **Hyper-V** and **Windows Hypervisor Platform**.
    - Click **OK** and restart your computer.

Step 1: Check if Virtualization is Enabled

1. **Open Task Manager**:
    - Right-click on the taskbar and select **Task Manager**, or press `Ctrl + Shift + Esc`.
2. **Go to the Performance Tab**:
    - Click the **Performance** tab.
    - Select **CPU** from the left pane.
3. **Check Virtualization Status**:
    - Look for "Virtualization" on the lower right. If it says **Enabled**, you can skip to using Docker.
    - If it says **Disabled**, you need to enable it in the BIOS.


Group 1
Docker Desktop redirect | Docker
https://app.docker.com/auth/desktop/redirect?code=_gY8YLEHY_BxS0x3aRyWwJTc-Mv4t2uJt0RA7CgCqciyp&state=BgJhrljIX4uwfjkx-TEbuproSj07yHM47eZLNPMfW_4
This site is trying to open Docker Desktop.
https://app.docker.com wants to open this application.
Always allow app.docker.com to open links of this type in the associated app
Open
Cancel
docker
You're almost done!
We're redirecting you to the desktop app.
If you don't see a dialog, click the button below.
Proceed to Docker Desktop
By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts.
Cookies Settings
Reject All
Accept All Cookies
USD/INR
+0.14%
Search
ENG
IN
19:51
01-12-2024

Here's a concise list of common Dockerfile commands:

- `FROM`: Sets the base image (e.g., `FROM python:3.9-slim`).
- `WORKDIR`: Sets the working directory inside the container (e.g., `WORKDIR /app`).
- `COPY`: Copies files from the host to the container (e.g., `COPY . /app`).
- `RUN`: Executes a command during the image build process (e.g., `RUN pip install -r requirements.txt`).
- `CMD`: Specifies the command to run the container (e.g., `CMD ["python", "app.py"]`).

2. Prepare Your Python Application

Ensure your Python project has:

- A **main script** (e.g., `app.py`).
- A **requirements file** (`requirements.txt`) listing dependencies.

Example `app.py`:

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, Docker!"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Dockerfile

```
# Use an official Python runtime as a parent image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 5000 available to the outside world
EXPOSE 5000

# Define the command to run the application
CMD ["python", "app.py"]
```

4. Build the Docker Image

Run the following command to build the Docker image:

```bash
docker build -t python-app .
```

Here:

- `-t python-app` tags the image as `python-app`.
- `.` refers to the current directory (containing the Dockerfile).

5. Run the Docker Container

Start a container from the image:

```bash
docker run -p 5000:5000 python-app
```

6. Verify the Application

Visit `http://localhost:5000` in your browser. You should see `Hello, Docker!`.

5. Run the Docker Container

Start a container from the image:

bash

```
docker run -p 5000:5000 python-app
```

What is Kubernetes?

- Kubernetes (often abbreviated as **K8s**) is an open-source platform designed to automate the deployment, scaling, and management of containerized applications.
- Developed by Google and now maintained by the **Cloud Native Computing Foundation (CNCF)**, Kubernetes is widely used for orchestrating applications in distributed environments.

Key Features of Kubernetes:

1. **Container Orchestration:** Automates the deployment, scaling, and management of containers.
2. **Load Balancing and Service Discovery:** Automatically distributes traffic across containers and manages service endpoints.
3. **Scaling:**Supports horizontal scaling (adding/removing container replicas) based on resource usage or custom metrics.

4. **Self-Healing:**
    - Restarts failed containers, replaces unresponsive nodes, and reschedule workloads to healthy nodes.
5. **Declarative Configuration:**
    - Allows users to define desired states for applications (e.g., number of replicas) using YAML/JSON files.
6. **Storage Management:**
    - Supports dynamic storage provisioning using local or cloud storage.

Core Components:

1. **Master Node:**
    - **API Server:** Exposes Kubernetes APIs for interaction.
    - **Scheduler:** Assigns workloads (pods) to nodes.
    - **Controller Manager:** Ensures the cluster's desired state is maintained.
    - **etcd:** A key-value store for cluster state.
2. **Worker Nodes:**
    - **Kubelet:** Agent running on nodes to execute containers and manage pods.
    - **Kube-proxy:** Handles networking and load balancing for pods.
3. **Pods:**
    - The smallest deployable unit in Kubernetes, consisting of one or more containers.

Advantages:

- **Portability:** Runs on-premises, cloud, or hybrid environments.
- **Scalability:** Easily handles applications of any size.
- **Resilience:** Ensures application availability and fault tolerance.
- **Community Support:** Backed by a large, active developer community.

Use Cases:

- Deploying microservices.
- Managing large-scale applications.
- Running machine learning workflows.
- Streamlining CI/CD pipelines.