Experiment – 7

**Aim:** To understand Docker architecture and container lifecycle with Docker installation.

**Theory:**

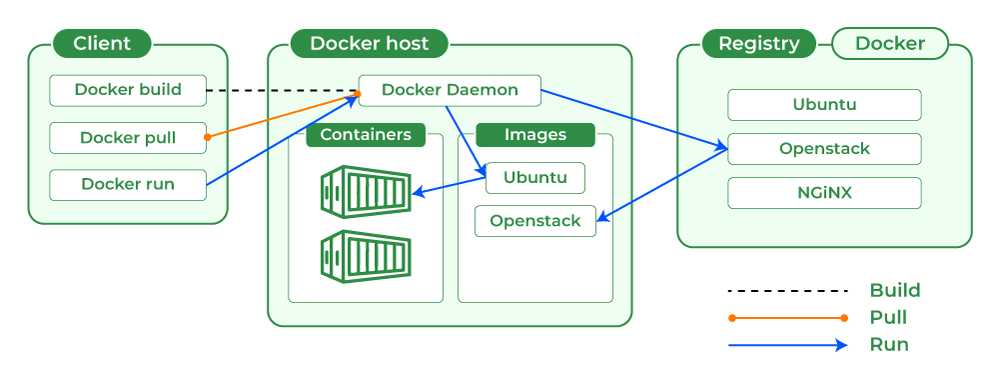
**What is Docker?**

Docker is an open-source platform that uses containerization technology to enable developers to package applications and their dependencies into a standardized unit called a **container**. Containers are lightweight, portable, and ensure that the application runs consistently across different environments.

**Key Concepts**

1. **Containers**: Containers are isolated environments that run applications. They include everything needed to run the application, such as the code, runtime, libraries, and system tools. This ensures that the application behaves the same, regardless of where it is deployed.
2. **Images**: Docker images are read-only templates used to create containers. They contain the application code, libraries, and dependencies. Images can be shared via Docker Hub or other container registries.
3. **Docker Engine**: This is the core component of Docker, responsible for creating and managing containers. It consists of a server (a long-running daemon process), a REST API for interacting with the daemon, and a command-line interface (CLI).
4. **Docker Hub**: A cloud-based registry service where Docker users can create, test, store, and distribute container images. It acts as a central repository for Docker images.

**Docker Architecture:**



**1. Docker Client**

The Docker client is the primary interface through which users interact with Docker. It can be a command-line interface (CLI) or a graphical user interface (GUI). The client sends commands to the Docker daemon using the Docker API. Common commands include docker build, docker pull, and docker run.

**2. Docker Daemon (dockerd)**

The Docker daemon is a background process that manages Docker objects such as images, containers, networks, and volumes. It listens for Docker API requests and processes them. The daemon can run on the same host as the client or on a remote host. It is responsible for building, running, and distributing Docker containers.

**3. Docker Host**

A Docker host is a machine that runs the Docker daemon and manages Docker containers. It includes the Docker daemon, images, containers, networks, and storage. The host is responsible for running one or more containers.

**4. Docker Registry**

Docker registries are repositories where Docker images are stored and distributed. The most common registry is Docker Hub, a public registry that allows users to share and download images. Private registries can also be set up for internal use. Images are pulled from the registry using the docker pull command and pushed using the docker push command.

**5. Docker Images**

Docker images are read-only templates used to create containers. They contain the application code, libraries, dependencies, and other necessary files. Images are built from a Dockerfile, which is a script containing a series of instructions on how to build the image.

**6. Docker Containers**

Containers are the runnable instances of Docker images. They are lightweight and isolated environments that run applications. Containers share the host system’s kernel but have their own filesystem, networking, and process space. They can be managed using Docker API or CLI commands to start, stop, delete, or move them.

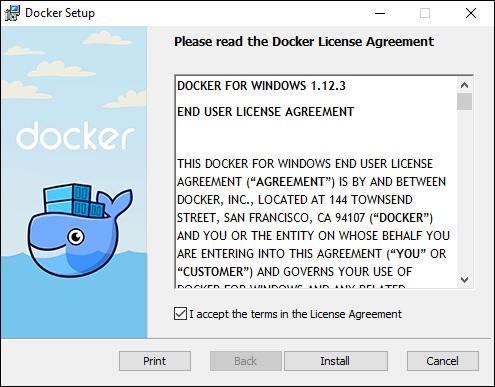
**7. Docker Storage**

Docker provides several storage options to persist data generated by and used by Docker containers. Storage can be managed using storage drivers, which control and manage images and containers on the Docker host. Types of Docker storage include:

* **Data Volumes**: These can be mounted directly into the filesystem of the container and are essentially directories or files on the Docker host filesystem.

**Docker Installation:**

**Step 1** − Click on the Agreement terms and then the Install button to proceed ahead with the installation.



**Step 2** − Once complete, click the Finish button to complete the installation.



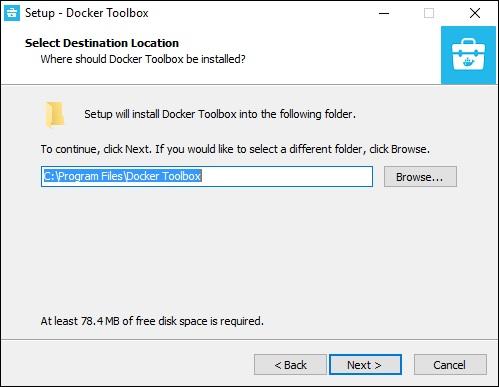
## **Docker ToolBox**

Once the installer has been downloaded, double-click it to start the installer and then follow the steps given below.

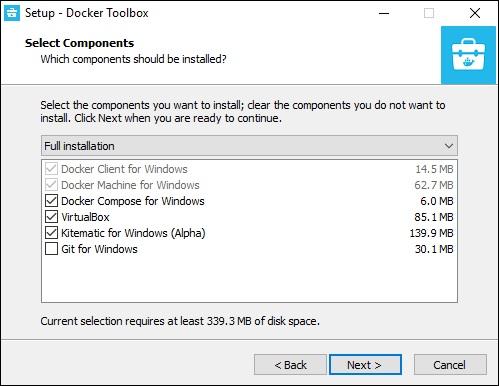
**Step 1** − Click the Next button on the start screen.



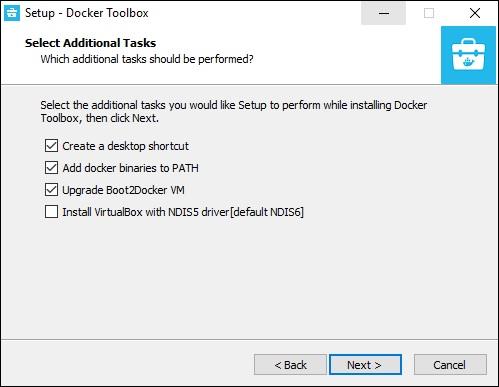
**Step 2** − Keep the default location on the next screen and click the Next button.



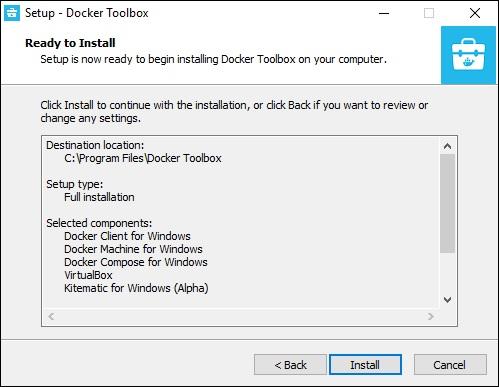
**Step 3** − Keep the default components and click the Next button to proceed.



**Step 4** − Keep the Additional Tasks as they are and then click the Next button.

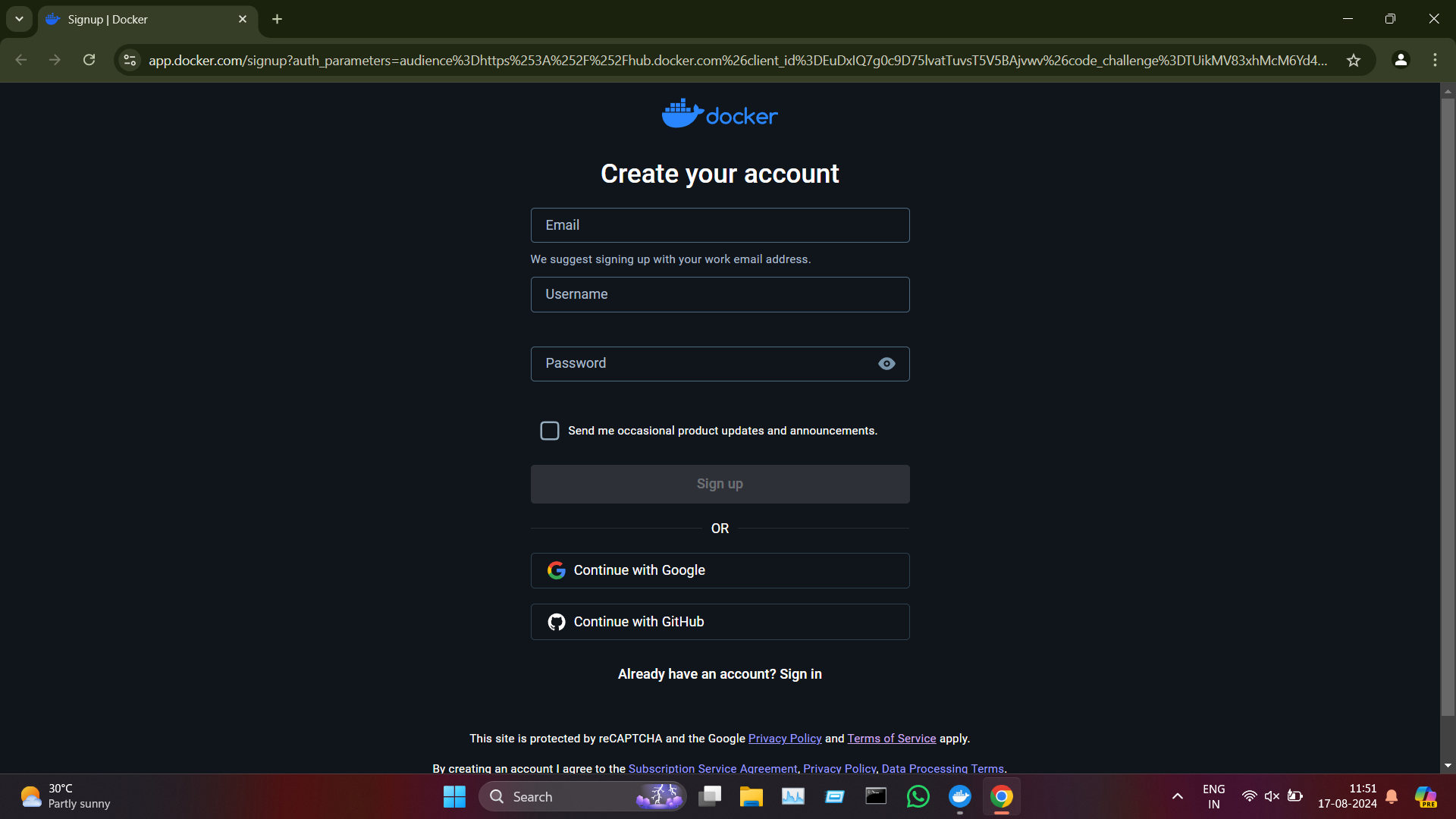


**Step 5** − On the final screen, click the Install button.

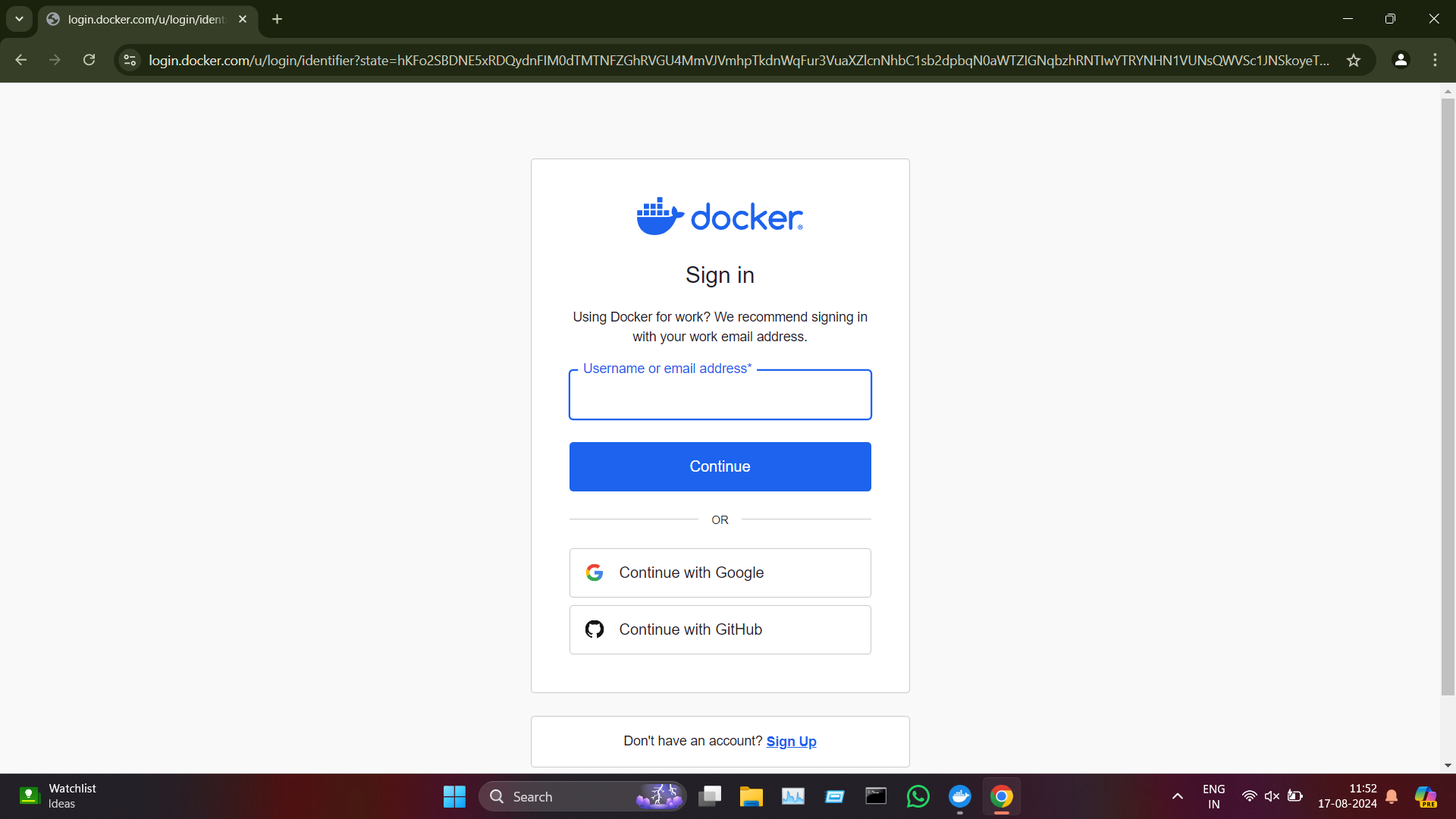


**Docker hub account creation:**

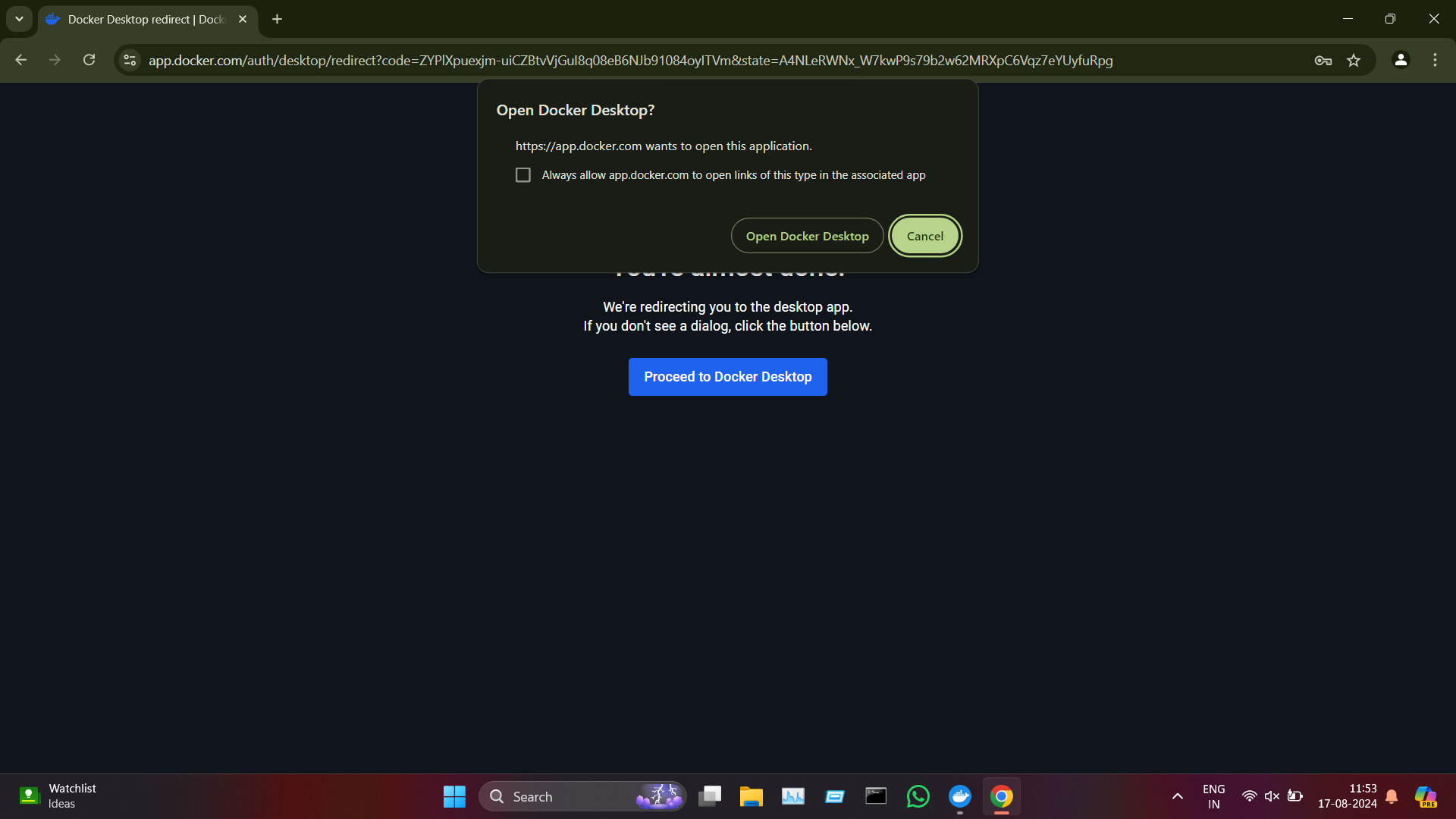
**Step 1:** Enter your Email, Username and Password and click Sign up or Continue with Google or Github.



**Step 2:** Sign in with Username and Password or Continue with Google or Github.



**Step 3:** Click ‘Open Docker Desktop’ to continue.



**Container Life-cycle:**

### 1. Create

In this stage, a container is created from a Docker image. The image contains all the necessary files and dependencies required to run the application. The docker create command is used to create a container, but it does not start it.

### 

### 2. Start

Once a container is created, it can be started using the docker start command. This command initializes the container and runs the application inside it.

### 

### 3. Run

The docker run command is a combination of create and start. It creates a new container from an image and starts it in a single step.

### 

### 

### 4. Stop

The docker stop command gracefully stops a running container by sending a SIGTERM signal, allowing the processes to terminate properly.

### 

### 5. Restart

The docker restart command stops and then starts a container. This is useful for applying configuration changes or recovering from errors.

### 

### 6. Remove

Finally, a container can be removed using the docker rm command. This deletes the container and frees up the resources it was using.

