HandOut.0: Docker Preliminaries and Installation

Sign-up with Docker:

- Make an account (with your IBA email) on www.docker.com and register

Installation of Docker Desktop

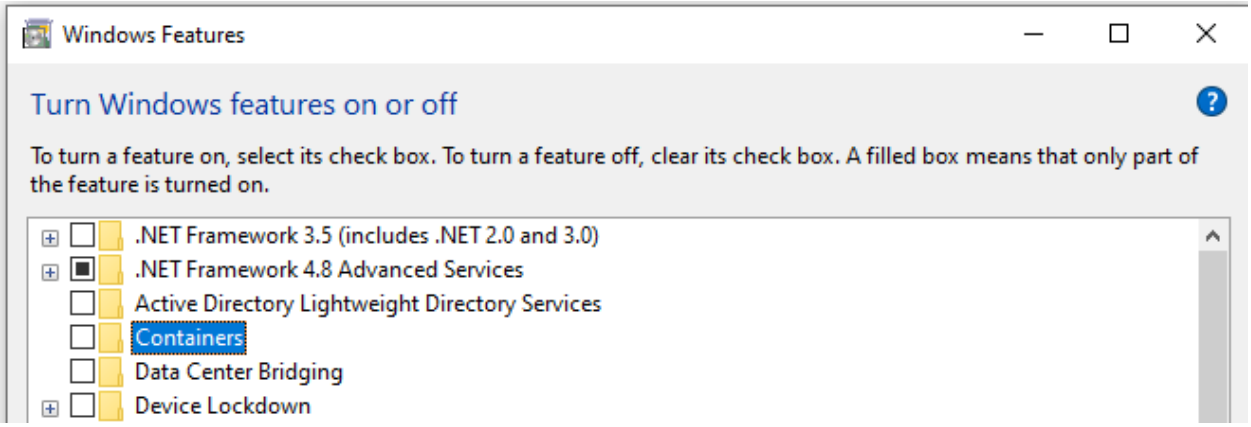
- Access https://docs.docker.com/desktop/install/windows-install/
- Download Docker Desktop for Windows – x86_64 (as of 23rd August)
- Use Recommended Settings during installation
- For installation, it is best to have at least 1TB HDD and at least 16GB of RAM (Note: if you have 8 GB, you can get your RAM updated from local markets; it will be useful in the long run)
- Sign in to Docker Desktop using your Docker account made above (in Step 1)

In case of any problems during installation:

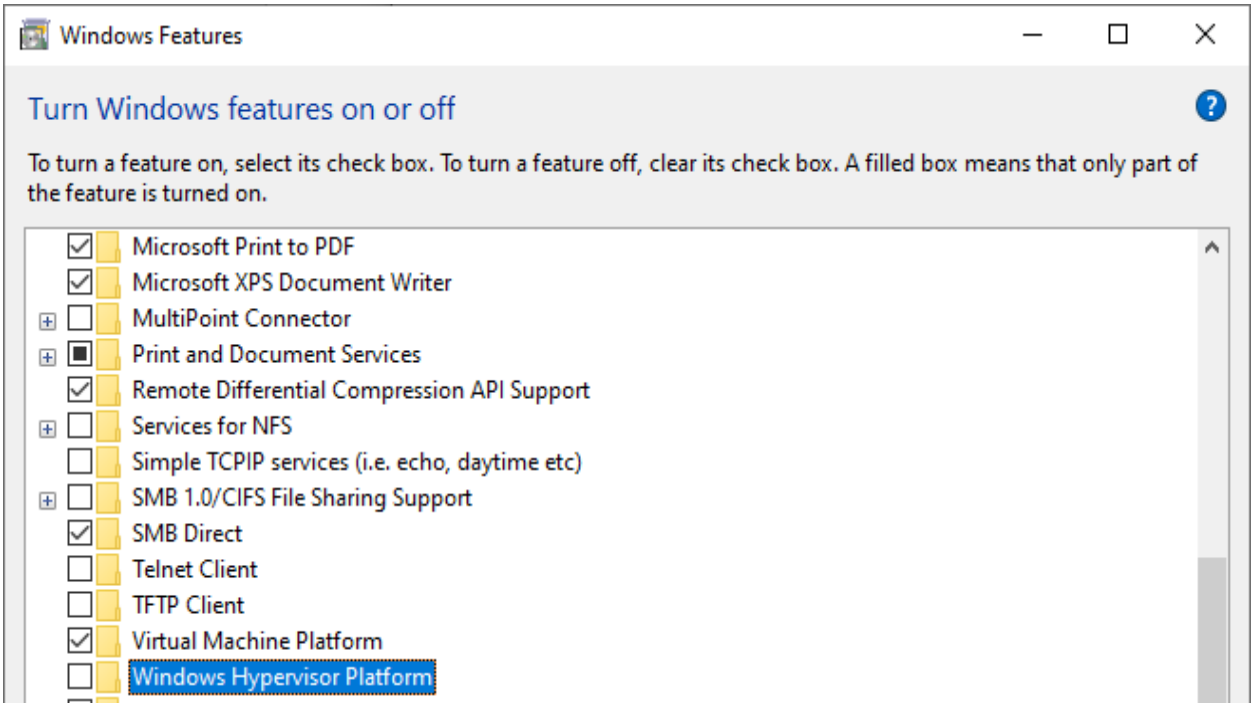
- Try disabling any antivirus software or firewall that may be blocking the installation process.
- Try running the Docker Desktop installer as an administrator.
- Make sure that the following settings are enabled in your computer's BIOS/UEFI boot menu (accessed through F2, F12, Esc etc.):
    - Intel Virtualization Technology (VT-x)/Intel VT or AMD Virtualization (AMD-V).
    - Intel VT-d or AMD IOMMU (enable if available – improves performance for virtualization)
    - Secure boot – this should generally remain enabled but in case of issues, it can be disabled
    - Update your BIOS/UEFI to the latest update

Miscellaneous:

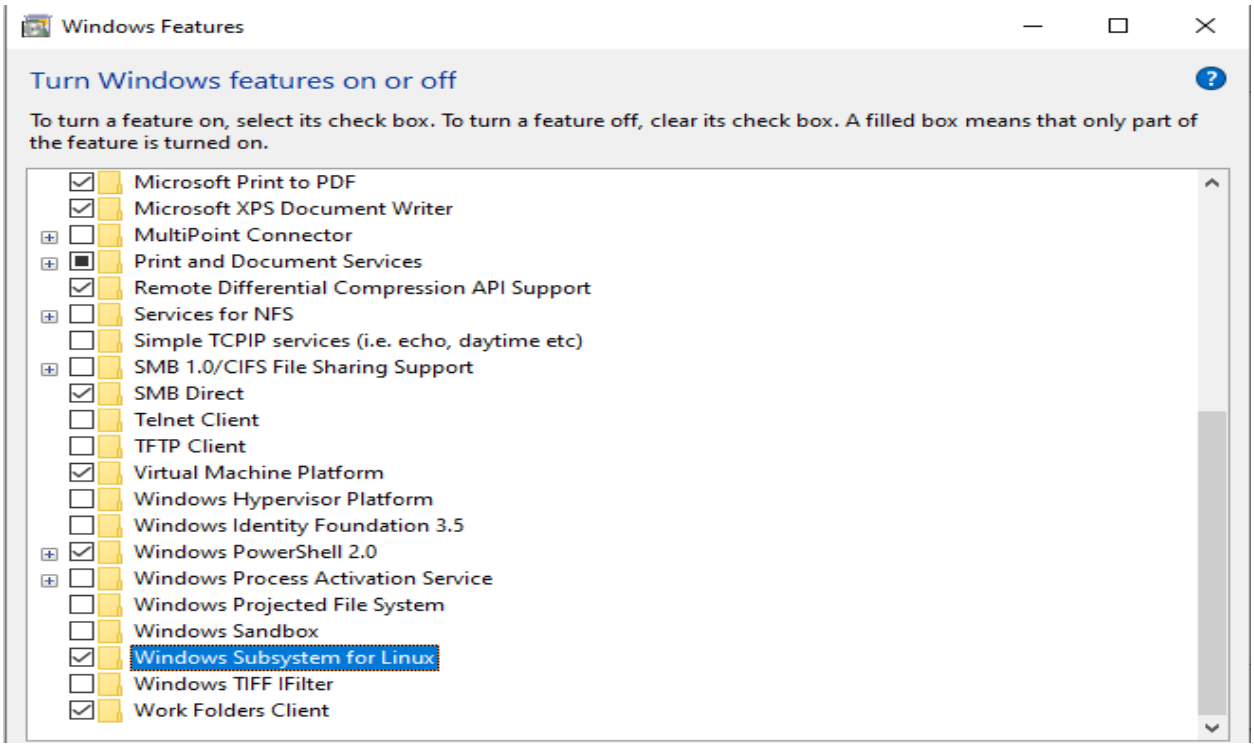
- Note that Docker is a containerization platform. It is stand-alone.
- Windows also offers its own Container features: Search for "Windows Features" from Start Menu and see "Containers": it should be disabled because Docker is stand-alone



- Similarly, Windows also offers its own hypervisor platform. However, Docker (or containerization) is independent of hypervisors, which are used to run one or more virtual machines (not containers). You can see Windows hypervisor in “Windows Features”, but it should also be disabled:



However, WSL (Windows Sub-System for Linux) is needed (as Docker cannot run without the Linux environment). It should be enabled in Windows Features:



WSL

WSL is a compatibility layer by Microsoft that allows users to run a Linux distribution natively on Windows without the overhead of traditional VMs.

It provides a Linux environment within Windows, including support for Linux command-line tools, utilities, and applications, directly from the Windows OS.

Key features:

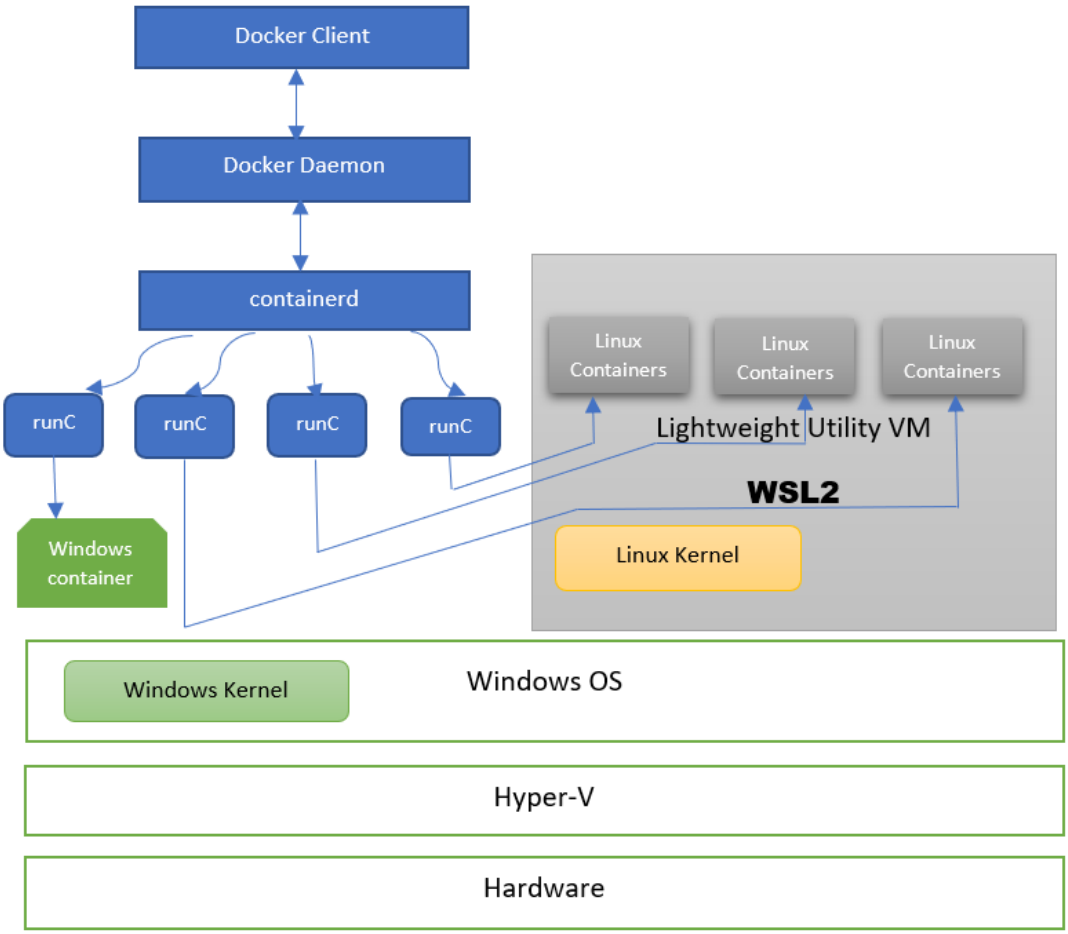
- Translates Linux system calls into Windows system calls.
- WSL1: A translation layer to emulate Linux kernel interfaces
- WSL2: A full Linux kernel running in a lightweight virtual machine (VM) using Windows Hyper-V virtualization.
- Access Windows files from Linux
- Running Windows executables from within the Linux shell.
- Use bash, git, or programming languages, without leaving Windows
- It has good performance and is nice and simple for Windows users

Key features wrt Docker:

- Docker runs natively on Linux – it directly uses Linux kernel, hence allowing containers to share the OS while remaining isolated
- WSL2 is backend for Docker Desktop – all containers run in the VM of WSL2

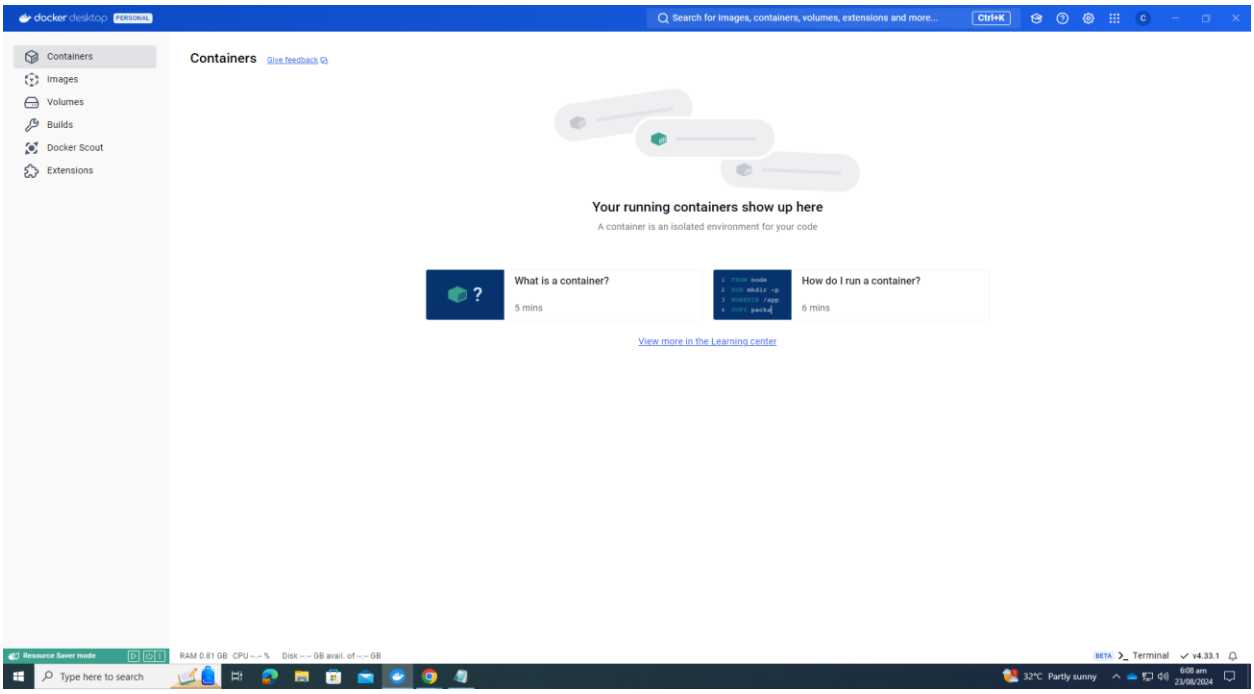
- You can share files between Docker containers and Windows host.
- Kubernetes can be used to orchestrate Docker containers.

Architecture:



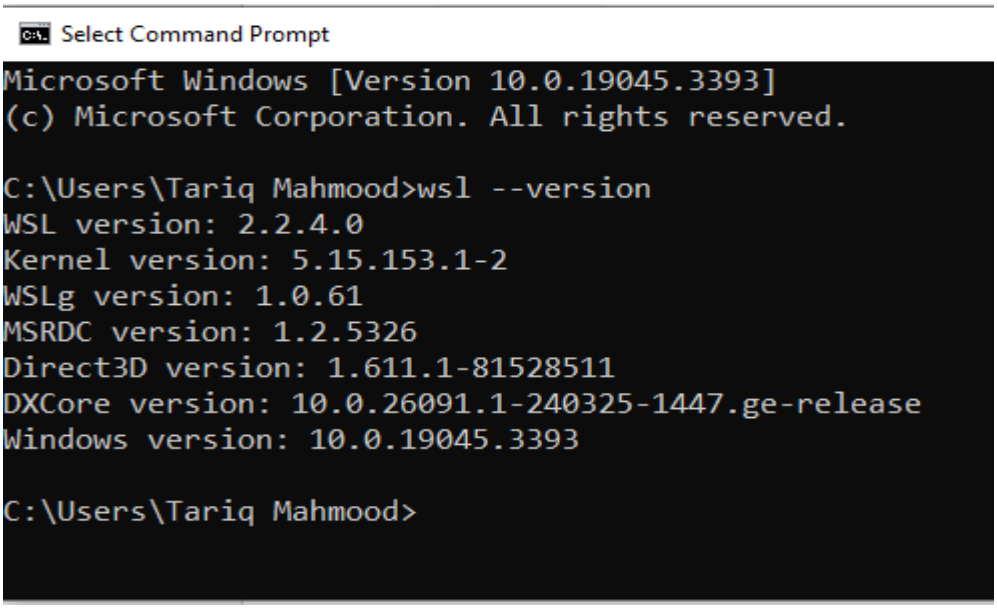
Docker Desktop Initial View:

You should get the following initial view:



Verify WSL:

- First, ensure that WSL is installed and running on your machine: type wsl --version on cmd. You should get the following:

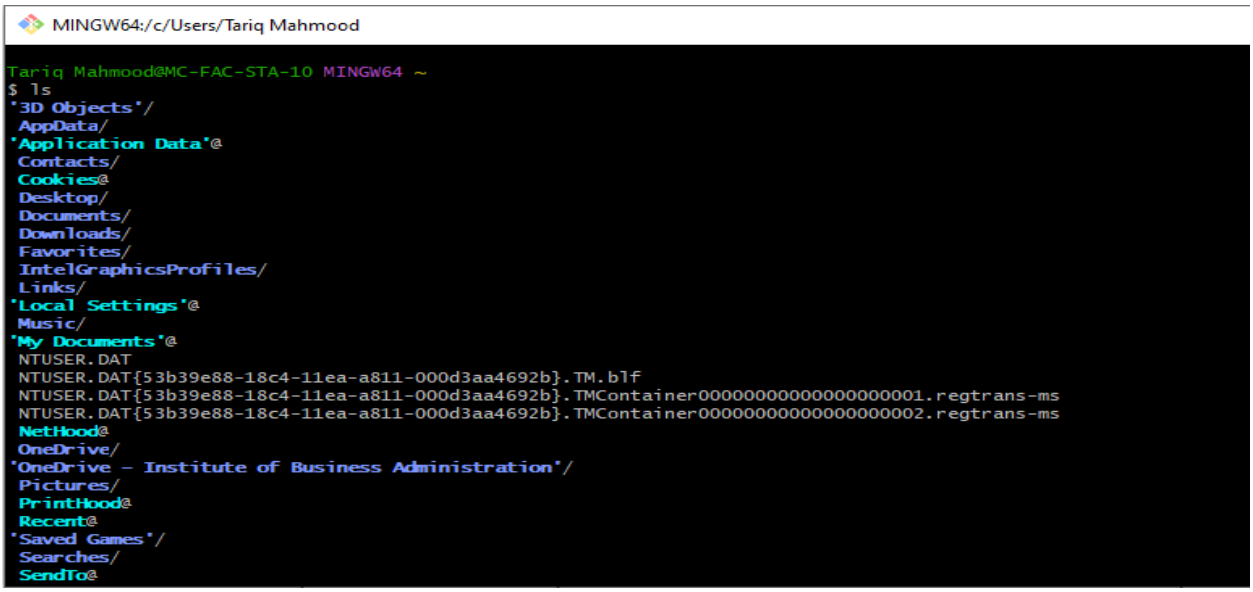


- WSLg stands for Windows Subsystem for Linux Gui. Its purpose is to facilitate the running of Linux GUI applications on Windows platform (X11 and Wayland). This is outside the scope of the course but for those interested: https://github.com/microsoft/wslg

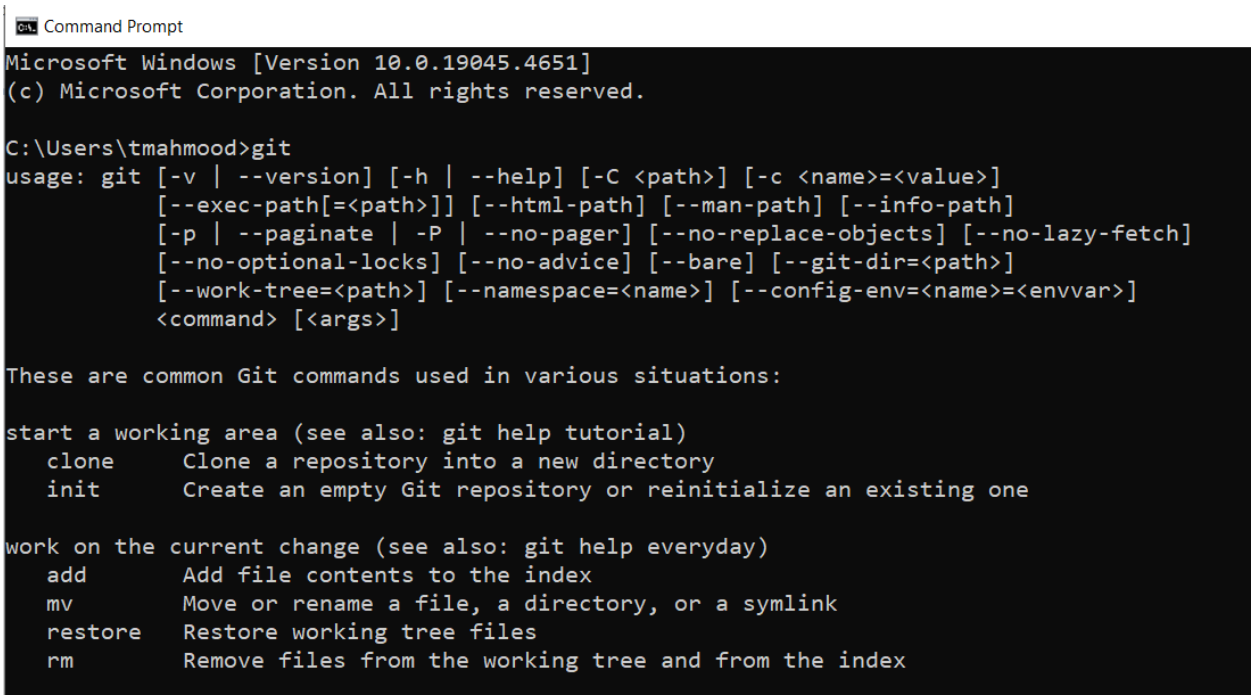
- MSRDC is Microsoft Remote Desktop Connection. It allows users to remotely connect to other computers or virtual machines, primarily to access their GUI. **It won't be used in Docker containers** – these use X11 forwarding and Virtual Network Computing (VNC) to run GUIs inside containers.
- Direct3D and DXCore provide information about the DirectX graphics library of Windows. **They are generally not used directly by Docker**.

Installing Git on Windows:

- It is better to install Git to manage your Github from Windows.
- Access https://gitforwindows.org/
- Press Download and run the installer. Total space required could be around 340 MB.
- Use default settings but you can specify the default editor of your own choice in the second or third step. There are many other options but it's best to use default setting unless you can control the non-defaults (e.g. SSH of your own choice etc.)
- You can launch Git bash when the installation is complete. This will open Mingw-W64, i.e., an environment that allows you to compile Windows applications using GCC – so it's GCC for Windows. If you know GCC, then you explore it to compile Windows apps.



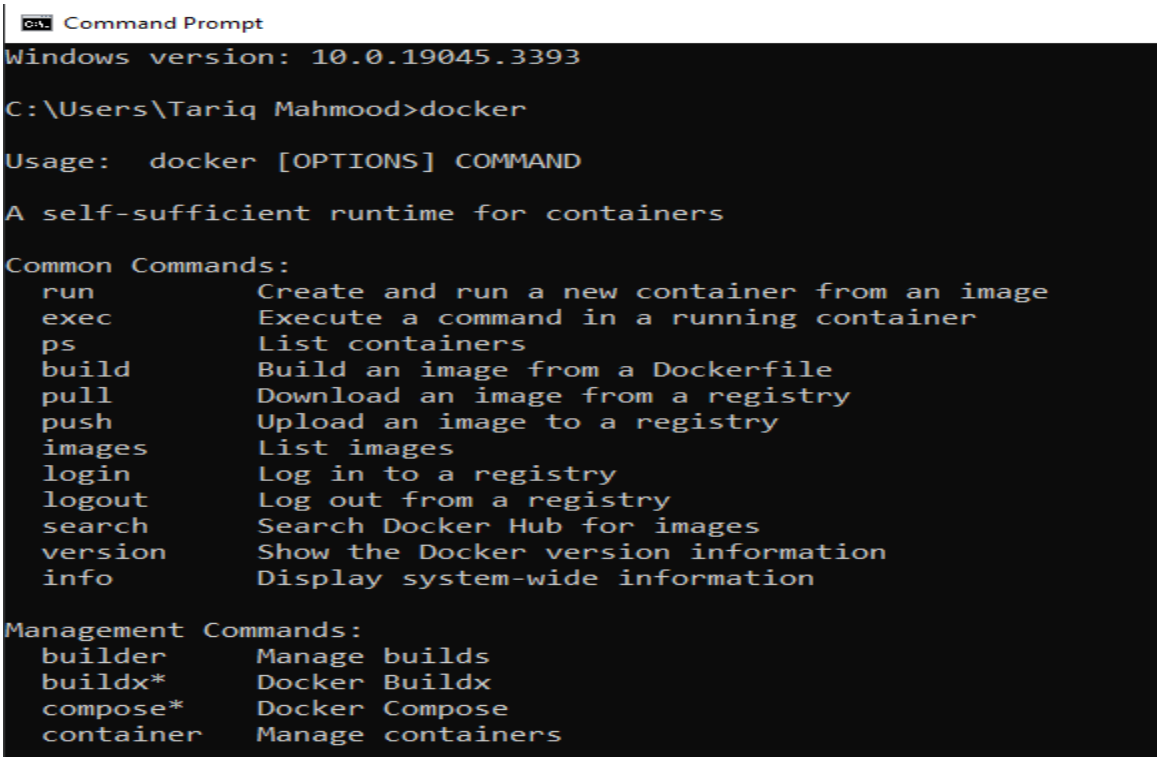
Verify that GIT is accessible from command line: open CMD and type git



Docker Desktop Containerization:

Verify that Docker is up and running.

Open CMD, type docker and press Enter.


Command Prompt
Windows version: 10.0.19045.3393
C:\Users\Tariq Mahmood>docker
Usage: docker [OPTIONS] COMMAND
A self-sufficient runtime for containers
Common Commands:
run Create and run a new container from an image
exec Execute a command in a running container
ps List containers
build Build an image from a Dockerfile
pull Download an image from a registry
push Upload an image to a registry
images List images
login Log in to a registry
logout Log out from a registry
search Search Docker Hub for images
version Show the Docker version information
info Display system-wide information
Management Commands:
builder Manage builds
buildx* Docker Buildx
compose* Docker Compose
container Manage containers