.NET Core and Docker

Advanced C# programming

Prepared By: Supun Kandaudahewa
supun.kandaudahewa@humber.ca

HUMBER

WE ARE HUMBER

Agenda

- Introduction to Docker : Why docker is needed
- Containers vs VM’s
- Benefits of using Containers
- Docker and DevOps
- Installing Docker: Various ways
- Docker and Images
- Demo (Installation, Configurations, Deploying a .NET Web API in a Docker Container)

Architecture of Enterprise Grade Applications

- Enterprise grade applications consists of application stacks including various technologies. Managing the compatibility of these with underlying OS is always a difficult task.

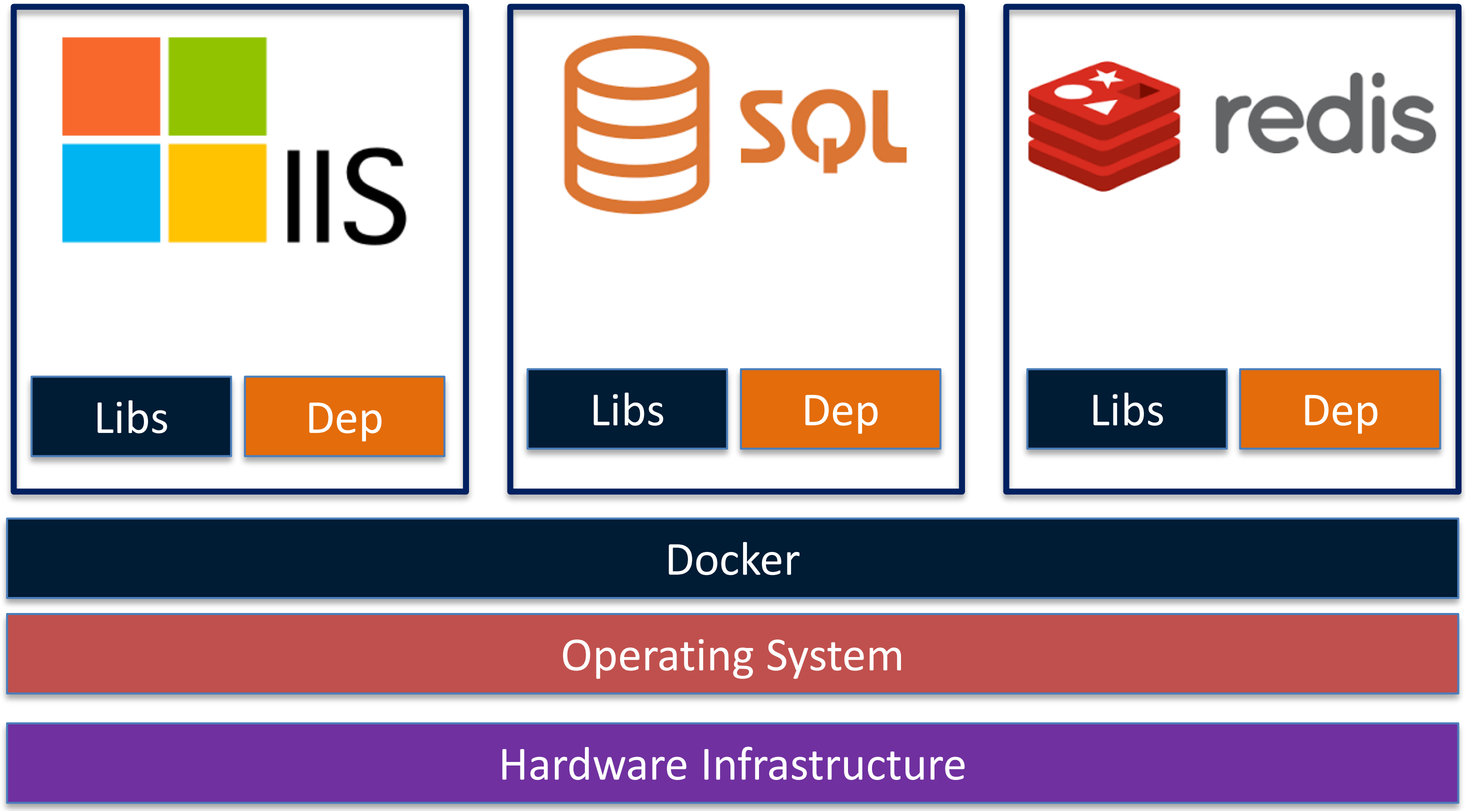
Issues Faced

- **Compatibility issues with the OS** - All the functionalities would not work in a single OS and different services would require different version
- **It works on my machine ??** – The dev and test environments were not the same.
- **Deployments were cumbersome** – Lengthy processes were to be followed to create exact same environment. Lot of back-and-forth communication between the Dev and the OPS team.
- **Architecture of the application can change** – When a any component was upgraded, needed to repeat the whole process to verify the compatibility.
- **Removing a component was risky** – As it might remove other dependencies.

The Matrix from Hell!!

How Docker Solves the Problem

Docker Solution→ Run each service with its own dependencies and libraries in a separate container.

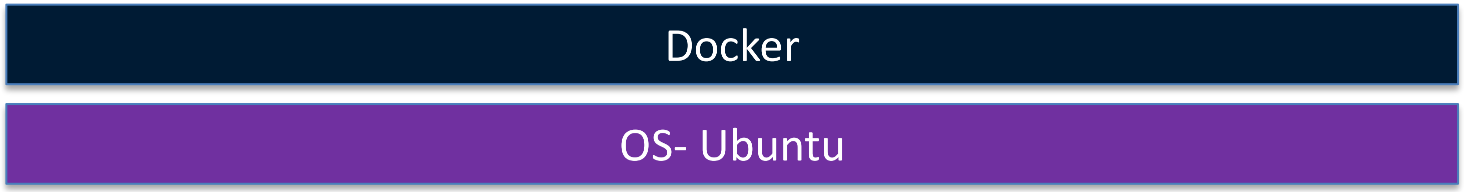


What are Containers

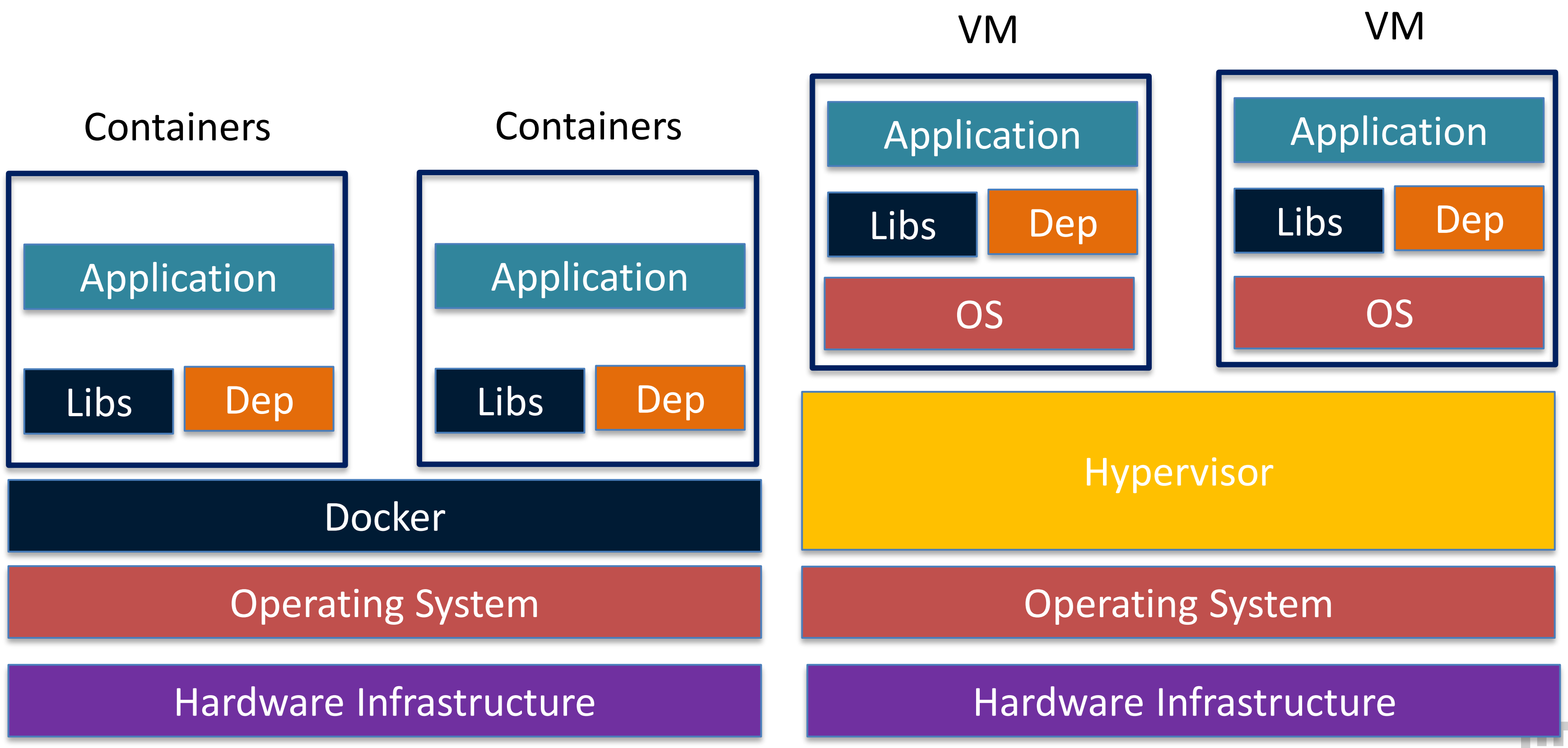
- Containers are completely isolated environments. They can have their own processes, services network interfaces etc.
- But they all share the same OS kernel, where VMs need own OS.
- Containers have been there for sometimes and docker utilizes LXC containers (.
- Docker is a high-level tool which can be used to manage these containers.
- He main purpose of docker is **not virtualization** but to package and containerize and an app for smooth shipping and execution.

OS Architecture

- The heart of any OS is its Kernel. Kernel is responsible for interacting with the hardware infrastructure and getting things done.
- On top of the kernel, there are different software applications, and these differentiate different OS's.
- As an example, Fedora, Ubuntu, CentOS etc. are based on the same Linux kernel. The software sits on top of it what makes the OSs different.
- So, if docker is installed on an Ubuntu OS (which has the Linux Kernel) any flavor based on Linux Kernel can be run on it. (Ex. Debian, CentOS, Fedora, SUSE)

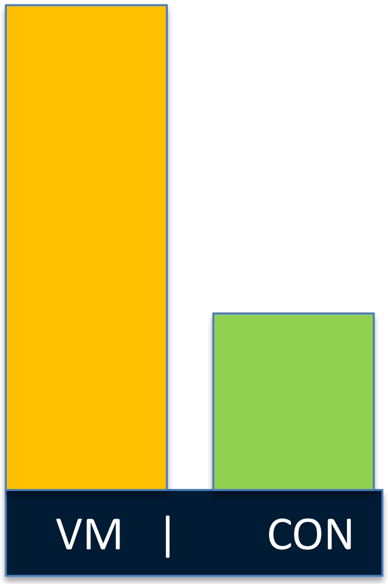


Containers Vs Virtual Machines



Containers Vs Virtual Machines Contd.

Resource Utilization



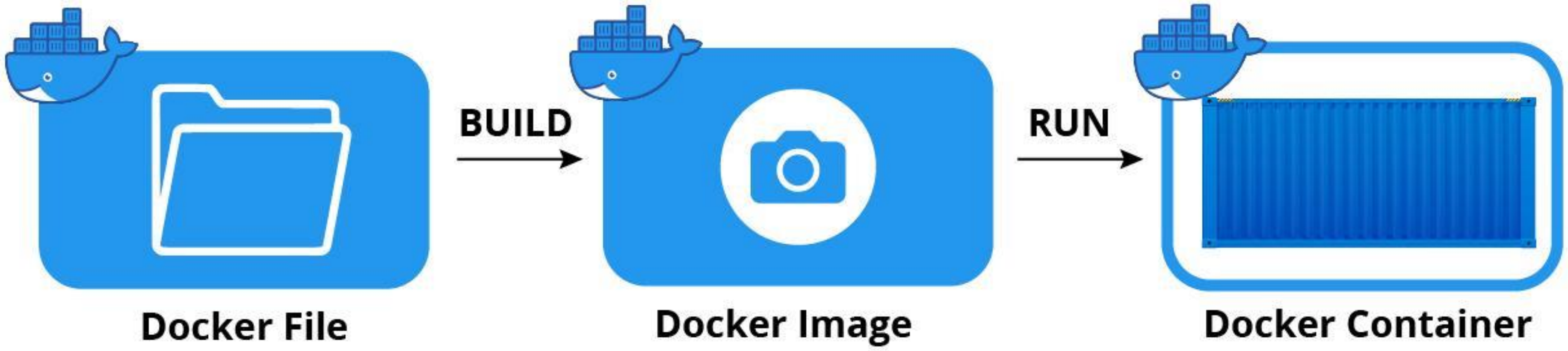
Boot time

Storage requirements

Licensing cost

Containers vs Images

- Image is a package or a template just like VM template. In docker images are used to create one or more containers.



https://jfrog.com/knowledge-base/a-beginners-guide-to-understanding-and-building-docker-images/

Docker and DevOps

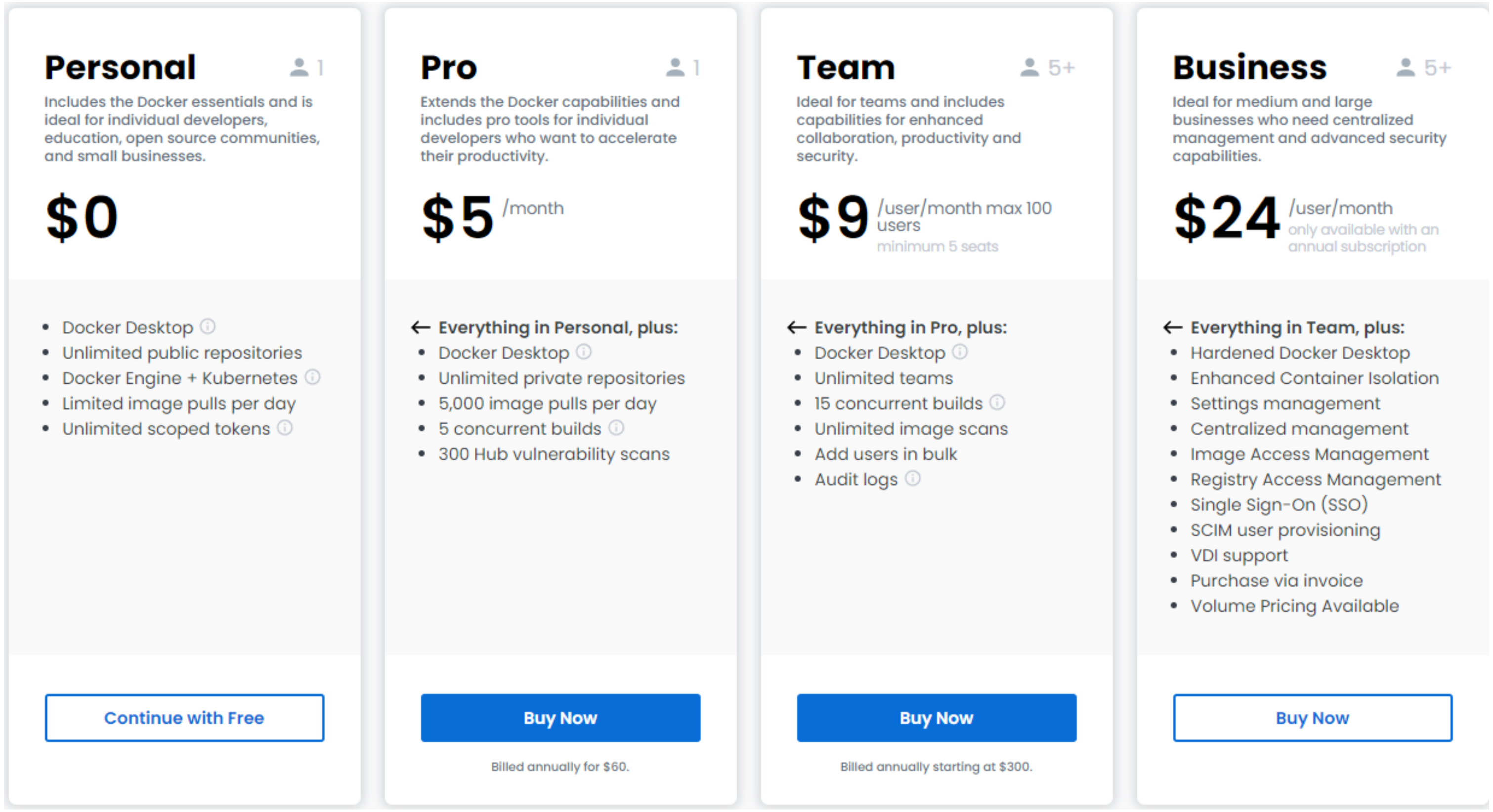
- Traditionally, developers develop application and hand it over to the operation team for deployment.
- This will contain the artifacts and set of very complex instructions. Usually, a app.zip file and a lengthy instruction manual.
- Since the Ops team is not very familiar with the app, they often struggle with deploying this. This often leads to lot of back-and-forth communication.
- With Docker the Dev team and Ops team can work hand in hand (DevOps).
- The deliverables are usually a app.zip file and a Dockerfile.
- Then a Docker image is created, and it can be used to create the containers.

Docker Editions and Docker Hub

- Docker has two versions, **Docker Community Edition** and **Docker Enterprise Edition**.

- Docker Hub - A hosted repository service provided by Docker for finding and sharing container images with your team and it supports:
  - Private Repositories.
  - Push and pull container images.
  - Automated Builds.

Docker Editions and Docker Hub



Personal 1

Includes the Docker essentials and is ideal for individual developers, education, open source communities, and small businesses.

$0

- Docker Desktop
- Unlimited public repositories
- Docker Engine + Kubernetes
- Limited image pulls per day
- Unlimited scoped tokens

Continue with Free

Pro 1

Extends the Docker capabilities and includes pro tools for individual developers who want to accelerate their productivity.

$5 /month

← **Everything in Personal, plus:**

- Docker Desktop
- Unlimited private repositories
- 5,000 image pulls per day
- 5 concurrent builds
- 300 Hub vulnerability scans

Buy Now

Billed annually for $60.

Team 5+

Ideal for teams and includes capabilities for enhanced collaboration, productivity and security.

$9 /user/month max 100 users
minimum 5 seats

← **Everything in Pro, plus:**

- Docker Desktop
- Unlimited teams
- 15 concurrent builds
- Unlimited image scans
- Add users in bulk
- Audit logs

Buy Now

Billed annually starting at $300.

Business 5+

Ideal for medium and large businesses who need centralized management and advanced security capabilities.

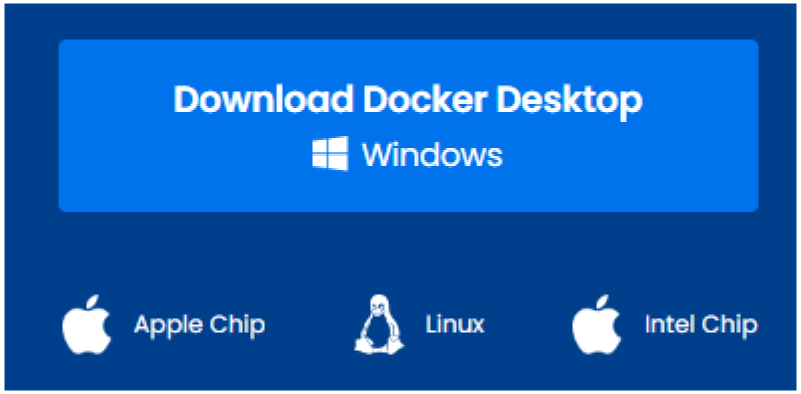
$24 /user/month
only available with an annual subscription

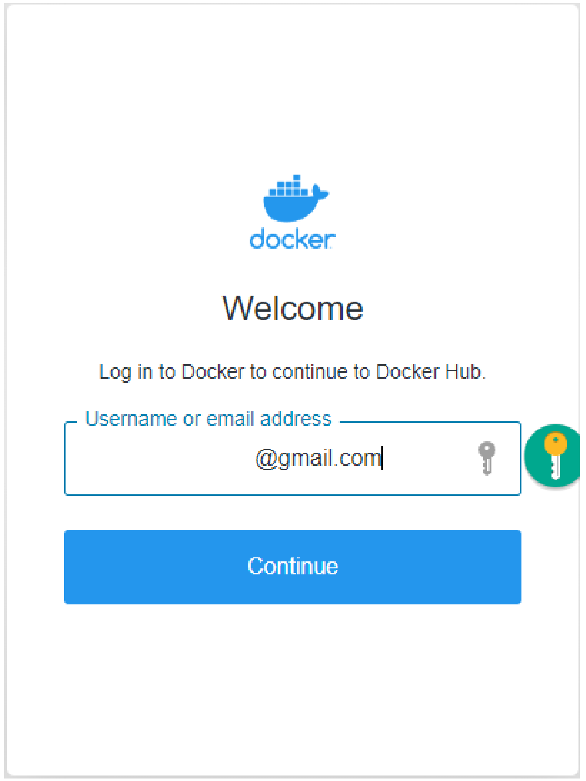
← **Everything in Team, plus:**

- Hardened Docker Desktop
- Enhanced Container Isolation
- Settings management
- Centralized management
- Image Access Management
- Registry Access Management
- Single Sign-On (SSO)
- SCIM user provisioning
- VDI support
- Purchase via invoice
- Volume Pricing Available

Buy Now

Getting Started




docker

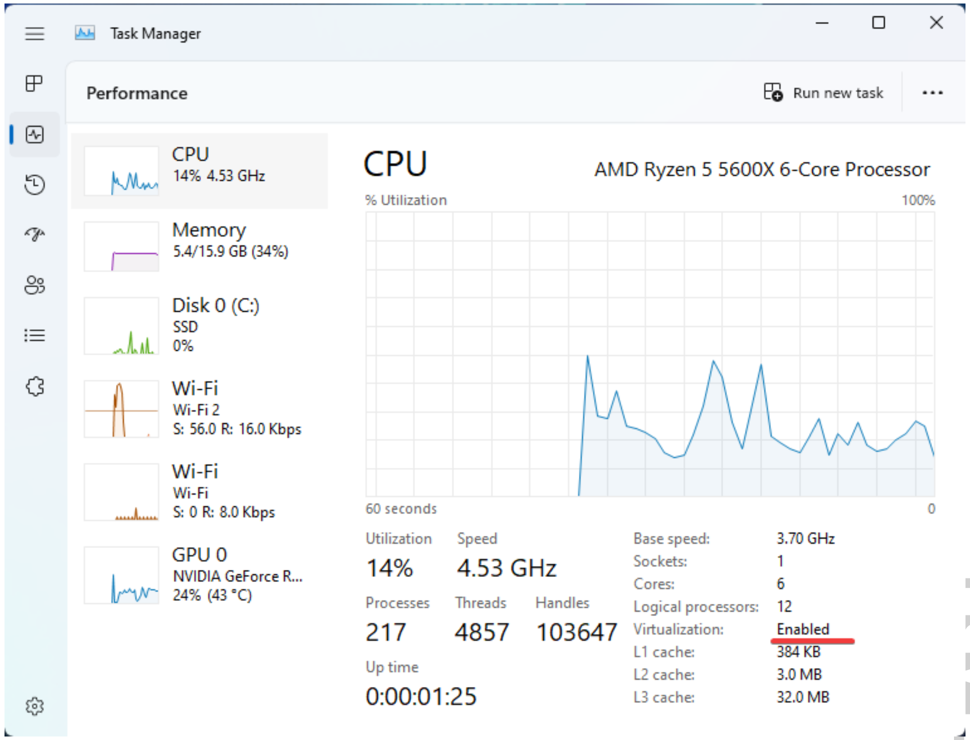
Welcome

Log in to Docker to continue to Docker Hub.

Username or email address

@gmail.com

Continue



CPU
14% 4.53 GHz

Memory
5.4/15.9 GB (34%)

Disk 0 (C:)
SSD
0%

Wi-Fi
Wi-Fi 2
S: 56.0 R: 16.0 Kbps

Wi-Fi
Wi-Fi
S: 0 R: 8.0 Kbps

GPU 0
NVIDIA GeForce R...
24% (43 °C)

CPU AMD Ryzen 5 5600X 6-Core Processor

% Utilization 100%

60 seconds 0

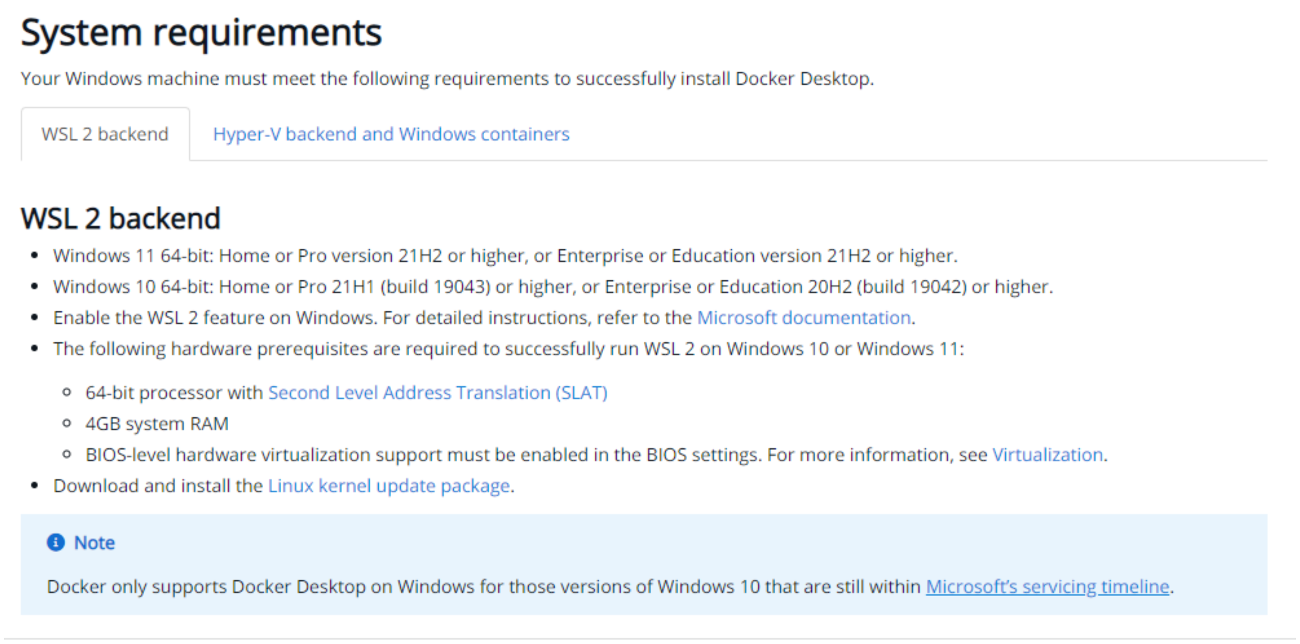
Utilization 14%
Speed 4.53 GHz
Processes 217
Threads 4857
Handles 103647
Up time 0:00:01:25

Base speed:	3.70 GHz
Sockets:	1
Cores:	6
Logical processors:	12
Virtualization:	Enabled
L1 cache:	384 KB
L2 cache:	3.0 MB
L3 cache:	32.0 MB

https://www.docker.com/

System Requirements

- It is really important to go through the requirements according to your OS version and architecture before moving forward with the installation.



System requirements

Your Windows machine must meet the following requirements to successfully install Docker Desktop.

WSL 2 backend | Hyper-V backend and Windows containers

WSL 2 backend

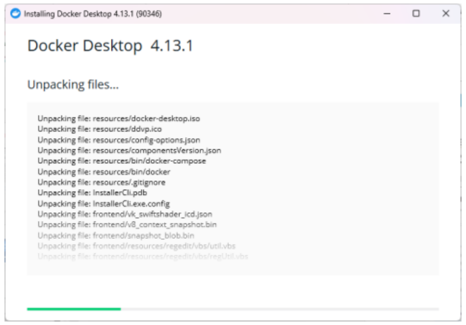
- Windows 11 64-bit: Home or Pro version 21H2 or higher, or Enterprise or Education version 21H2 or higher.
- Windows 10 64-bit: Home or Pro 21H1 (build 19043) or higher, or Enterprise or Education 20H2 (build 19042) or higher.
- Enable the WSL 2 feature on Windows. For detailed instructions, refer to 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
  - BIOS-level hardware virtualization support must be enabled in the BIOS settings. For more information, see Virtualization.
- Download and install the Linux kernel update package.

Note

Docker only supports Docker Desktop on Windows for those versions of Windows 10 that are still within Microsoft's servicing timeline.

https://docs.docker.com/desktop/install/windows-install/

Installation



Installing Docker Desktop 4.13.1 (90346)

Docker Desktop 4.13.1

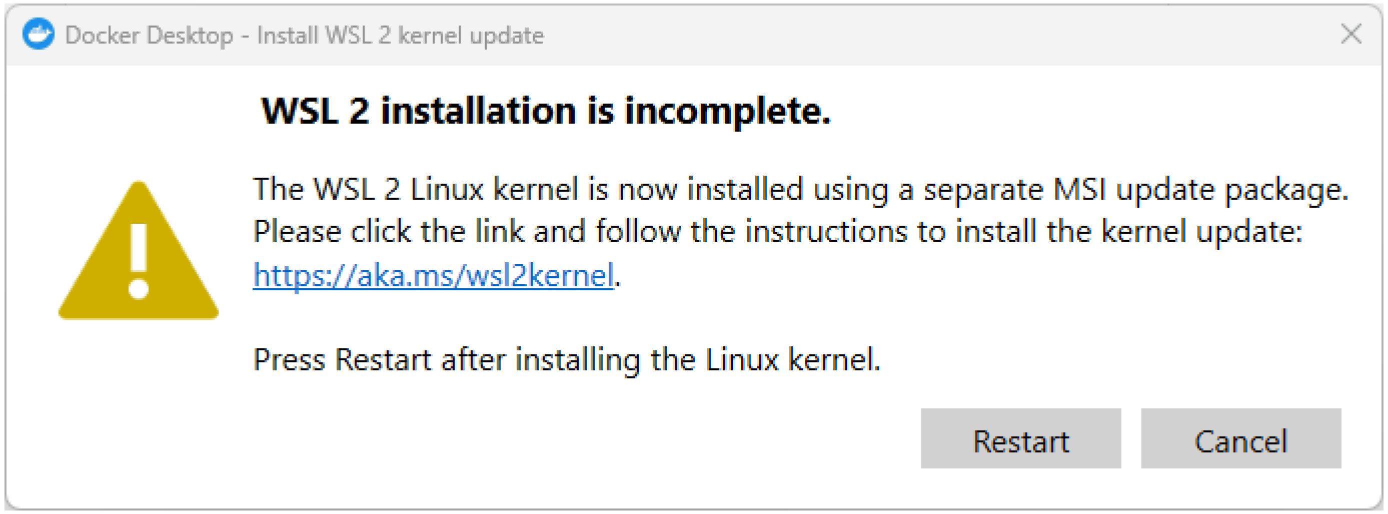
Unpacking files...

```
Unpacking file: resources/docker-desktop.iso
Unpacking file: resources/ddvp.ico
Unpacking file: resources/config-options.json
Unpacking file: resources/componentsVersion.json
Unpacking file: resources/bin/docker-compose
Unpacking file: resources/bin/docker
Unpacking file: resources/.gitignore
Unpacking file: InstallerCli.pdb
Unpacking file: InstallerCli.exe.config
Unpacking file: frontend/vk_swiftshader_icd.json
Unpacking file: frontend/v8_context_snapshot.bin
Unpacking file: frontend/snapshot_blob.bin
Unpacking file: frontend/resources/regedit/vbs/util.vbs
Unpacking file: frontend/resources/regedit/vbs/regUtil.vbs
```

Welcome to the Windows Subsystem for Linux Update Setup Wizard

The Setup Wizard will install Windows Subsystem for Linux Update on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

https://learn.microsoft.com/en-us/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package



Docker Desktop - Install WSL 2 kernel update

WSL 2 installation is incomplete.

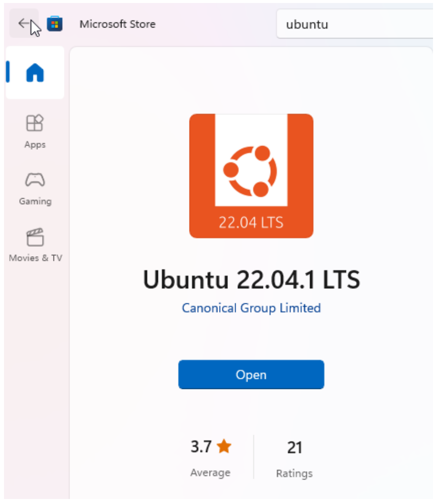
The WSL 2 Linux kernel is now installed using a separate MSI update package. Please click the link and follow the instructions to install the kernel update: https://aka.ms/wsl2kernel.

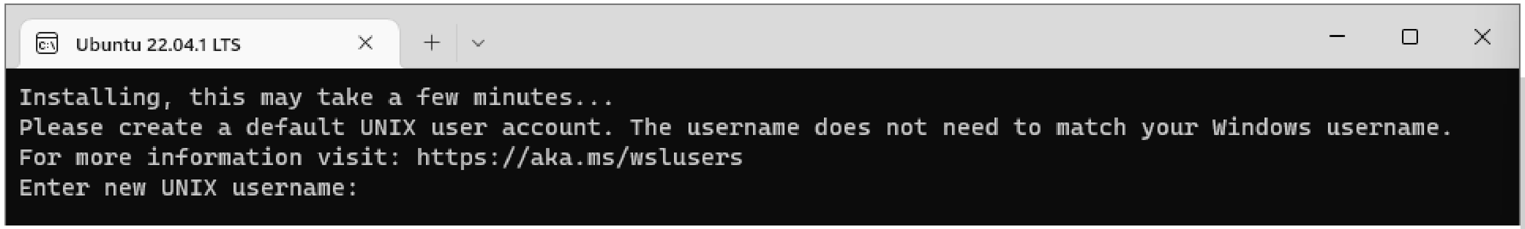
Press Restart after installing the Linux kernel.

Installation

Open powershell and type following command and then get the linux distro installed from Microsoft Store

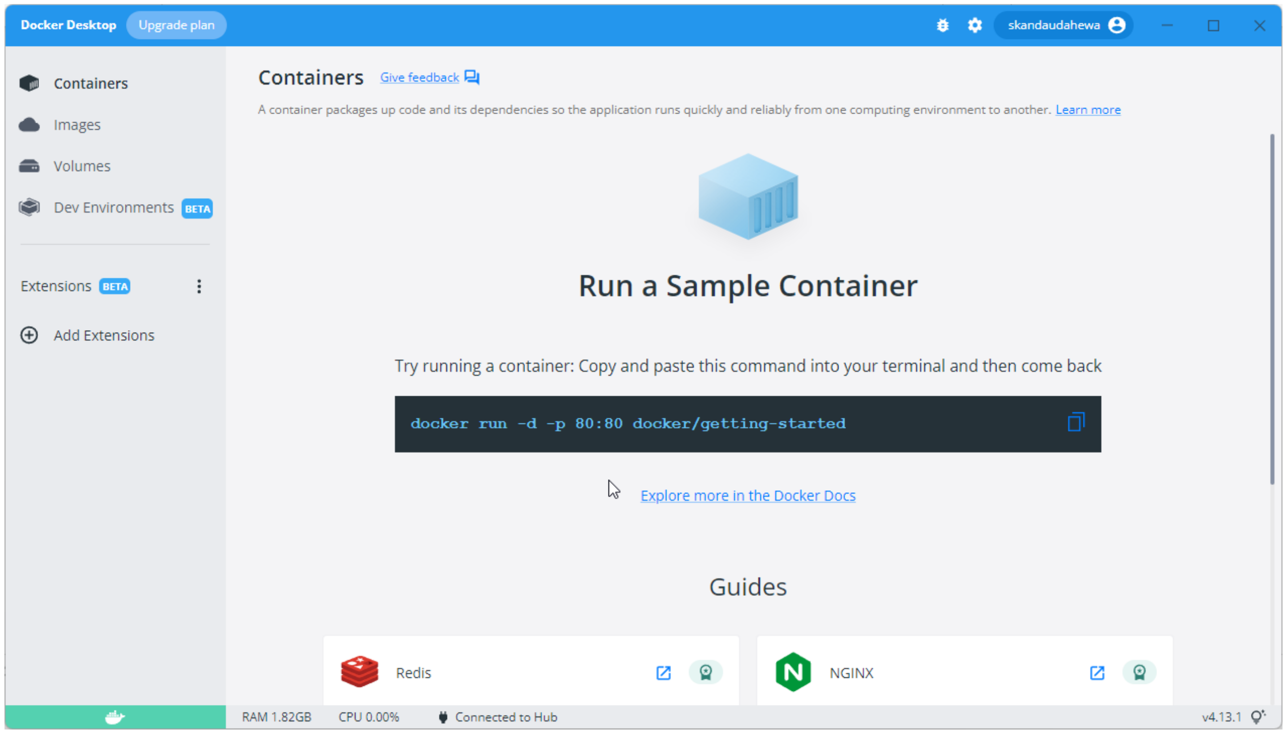
```
wsl --set-default-version 2
```





```
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username:
```

Starting Up Docker



Containers Give feedback

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

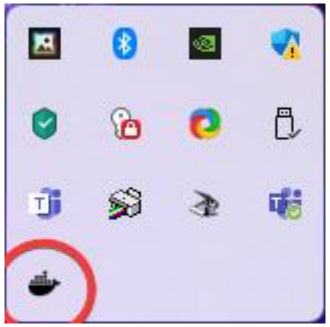
Run a Sample Container

Try running a container: Copy and paste this command into your terminal and then come back

```
docker run -d -p 80:80 docker/getting-started
```

Explore more in the Docker Docs

Guides



docker run -d -p 80:80 docker/getting-started

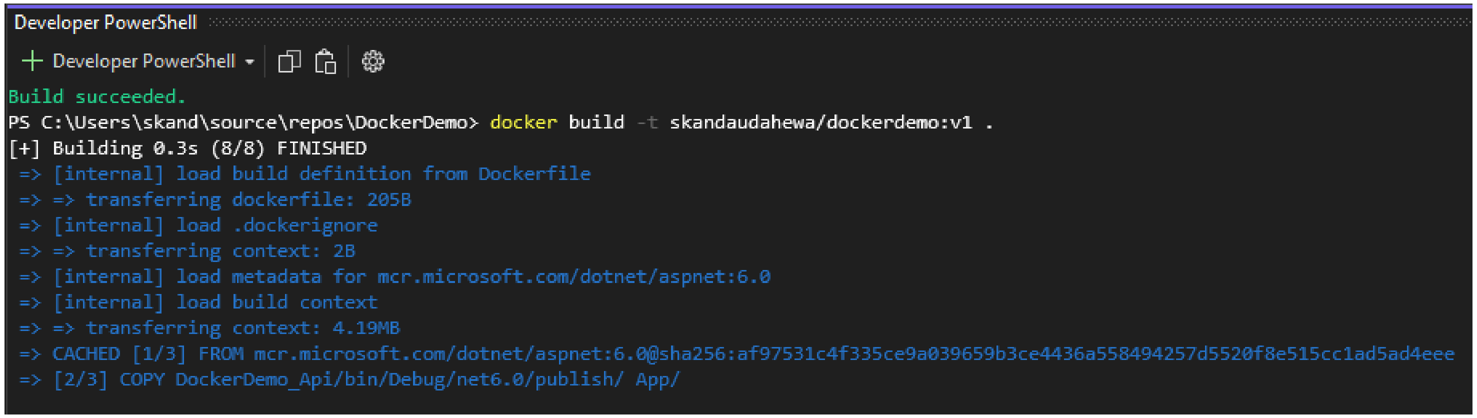
Creating the Dockerfile

- In order to containerize our app, the first step is to create a Docker file.
- The name should be **Dockerfile** without any extension and all the instruction regarding the creation of docker container should be specified here.
- Create this file in the root of the project.

https://learn.microsoft.com/en-us/dotnet/core/docker/build-container?tabs=windows

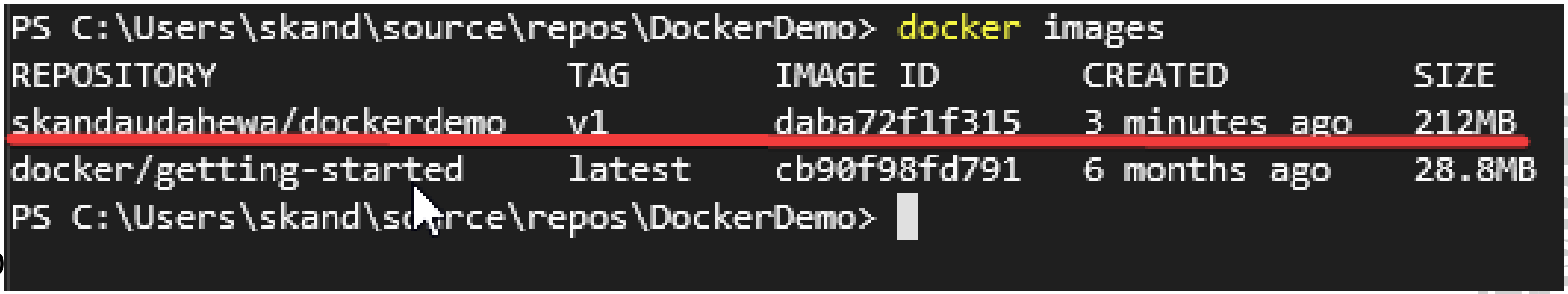
Creating the Image

- Go to View → Terminal and type below command:



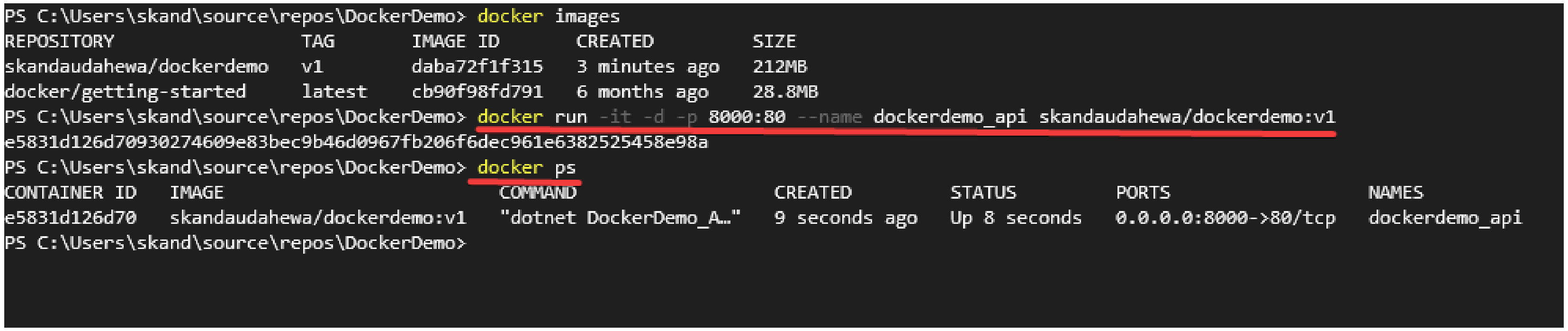
```
Build succeeded.
PS C:\Users\skand\source\repos\DockerDemo> docker build -t skandaudahewa/dockerdemo:v1 .
[+] Building 0.3s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 205B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:6.0
 => [internal] load build context
 => => transferring context: 4.19MB
 => CACHED [1/3] FROM mcr.microsoft.com/dotnet/aspnet:6.0@sha256:af97531c4f335ce9a039659b3ce4436a558494257d5520f8e515cc1ad5ad4eee
 => [2/3] COPY DockerDemo_Api/bin/Debug/net6.0/publish/ App/
```

- To check whether the docker image is created successfully:

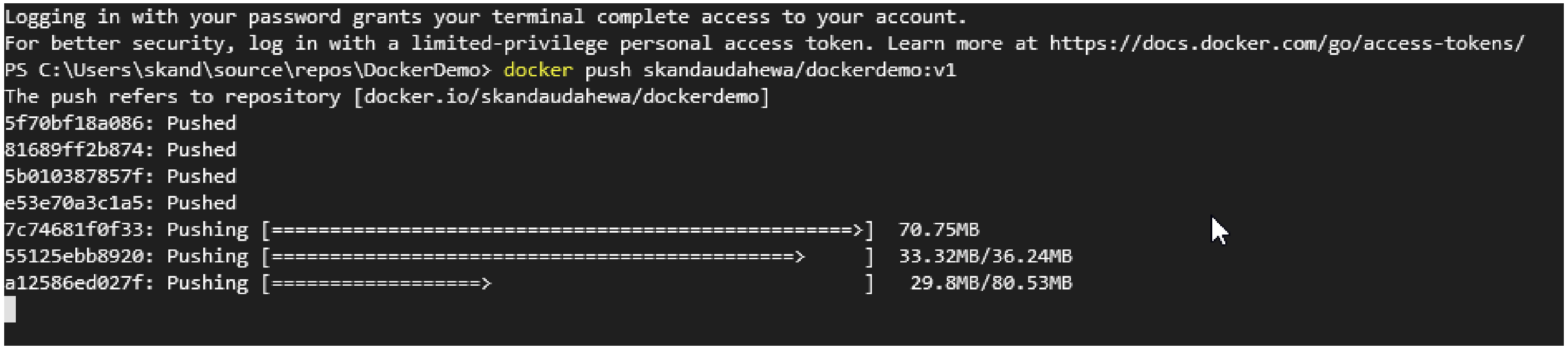


```
PS C:\Users\skand\source\repos\DockerDemo> docker images
REPOSITORY                 TAG       IMAGE ID       CREATED          SIZE
skandaudahewa/dockerdemo   v1        daba72f1f315   3 minutes ago    212MB
docker/getting-started     latest    cb90f98fd791   6 months ago     28.8MB
PS C:\Users\skand\source\repos\DockerDemo>
```

Creating, Running the Container and Verifying



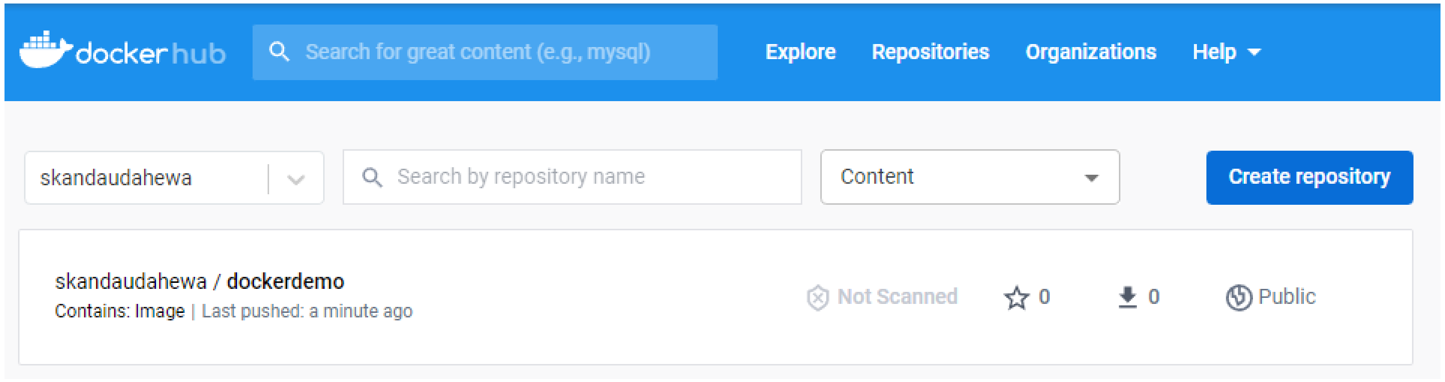
```
PS C:\Users\skand\source\repos\DockerDemo> docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
skandaudahewa/dockerdemo    v1        daba72f1f315   3 minutes ago    212MB
docker/getting-started      latest    cb90f98fd791   6 months ago     28.8MB
PS C:\Users\skand\source\repos\DockerDemo> docker run -it -d -p 8000:80 --name dockerdemo_api skandaudahewa/dockerdemo:v1
e5831d126d70930274609e83bec9b46d0967fb206f6dec961e6382525458e98a
PS C:\Users\skand\source\repos\DockerDemo> docker ps
CONTAINER ID   IMAGE                         COMMAND                  CREATED         STATUS         PORTS                  NAMES
e5831d126d70   skandaudahewa/dockerdemo:v1   "dotnet DockerDemo_A…"   9 seconds ago   Up 8 seconds   0.0.0.0:8000->80/tcp   dockerdemo_api
PS C:\Users\skand\source\repos\DockerDemo>
```

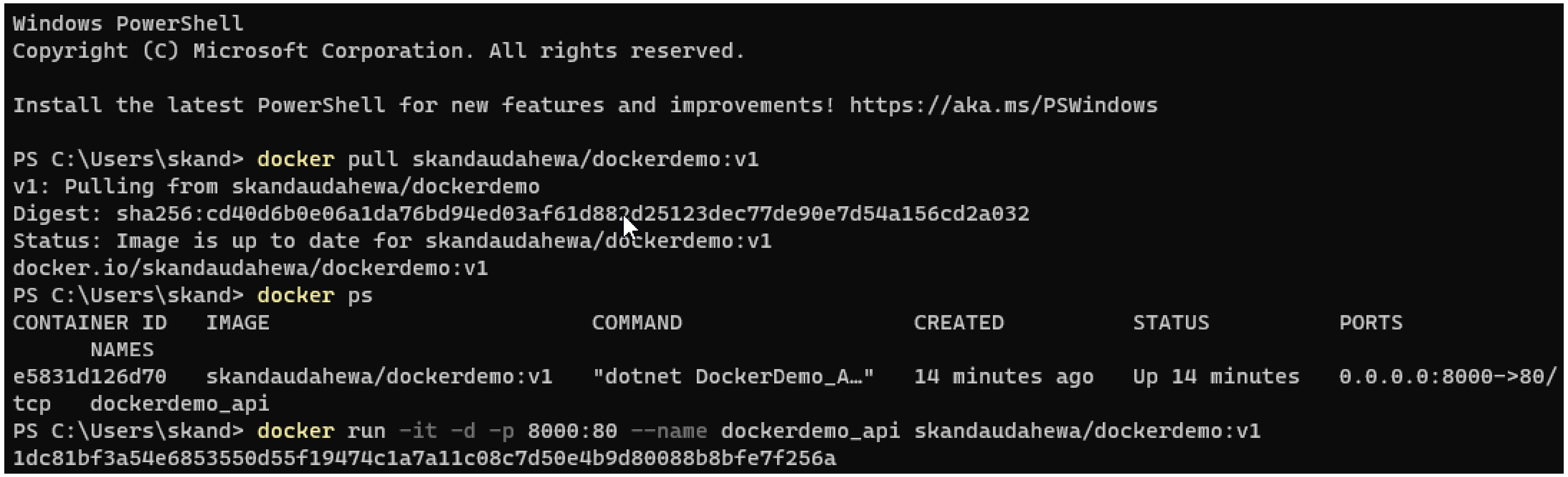


```
Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
PS C:\Users\skand\source\repos\DockerDemo> docker push skandaudahewa/dockerdemo:v1
The push refers to repository [docker.io/skandaudahewa/dockerdemo]
5f70bf18a086: Pushed
81689ff2b874: Pushed
5b010387857f: Pushed
e53e70a3c1a5: Pushed
7c74681f0f33: Pushing [==================================================>]  70.75MB
55125ebb8920: Pushing [=============================================>     ]  33.32MB/36.24MB
a12586ed027f: Pushing [==================>                                ]   29.8MB/80.53MB
```

Checking on DockerHub

- You can pull this image from the docker hub and quickly deploy it to a container. **docker pull skandaudahewa/dockerdemo:v1**


docker hub
Search for great content (e.g., mysql)
Explore
Repositories
Organizations
Help
skandaudahewa
Search by repository name
Content
Create repository
skandaudahewa / dockerdemo
Contains: Image | Last pushed: a minute ago
Not Scanned
0
0
Public



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\skand> docker pull skandaudahewa/dockerdemo:v1
v1: Pulling from skandaudahewa/dockerdemo
Digest: sha256:cd40d6b0e06a1da76bd94ed03af61d882d25123dec77de90e7d54a156cd2a032
Status: Image is up to date for skandaudahewa/dockerdemo:v1
docker.io/skandaudahewa/dockerdemo:v1
PS C:\Users\skand> docker ps
CONTAINER ID   IMAGE                         COMMAND                  CREATED          STATUS          PORTS
      NAMES
e5831d126d70   skandaudahewa/dockerdemo:v1   "dotnet DockerDemo_A…"   14 minutes ago   Up 14 minutes   0.0.0.0:8000->80/
tcp   dockerdemo_api
PS C:\Users\skand> docker run -it -d -p 8000:80 --name dockerdemo_api skandaudahewa/dockerdemo:v1
1dc81bf3a54e6853550d55f19474c1a7a11c08c7d50e4b9d80088b8bfe7f256a
```

THANK YOU.

HUMBER

WE ARE HUMBER