**Experiment-7: Working with Docker and DockerHub**

**Aim:** To install Docker, Minikube, and DockerHub account, execute Docker CLI commands, modify and push Docker images, create and push Dockerfile images, run multiple containers using Docker Compose, deploy and scale applications using Minikube, and deploy and manage monitoring systems using Nagios in Docker.

**Introduction:**

**Installation of Docker, Minikube, and Account in DockerHub:**

**Docker:** Docker is a platform for developing, shipping, and running applications in containers, which provide a lightweight, portable, and self-sufficient environment for applications.

**Minikube:** Minikube is a tool that enables running a single-node Kubernetes cluster locally, simplifying the process of learning and developing Kubernetes applications.

**DockerHub**: DockerHub is a cloud-based repository where Docker users can create, test, store, and distribute container images.

**Docker CLI Commands:**

**Docker CLI:** The Docker Command Line Interface (CLI) provides commands to interact with Docker, including building, running, managing, and inspecting containers.

**Modify and Push Docker Image:**

**Modifying Docker Image:** This involves making changes to an existing Docker image, such as updating software or configurations.

**Pushing Docker Image:** The docker push command uploads a Docker image to a registry like DockerHub.

**Create and Push Dockerfile Image:**

**Dockerfile**: A Dockerfile is a script that contains a series of instructions on how to build a Docker image.

**Building and Pushing Image:** Creating a Dockerfile, building the image with docker build, and pushing it to DockerHub using docker push.

**Running Multiple Containers Using Docker Compose:**

**Docker Compose:** Docker Compose is a tool for defining and running multi-container Docker applications using a docker-compose.yml file.

**Deploying and Scaling Applications Using Minikube:**

**Deploying Applications:** Using Minikube to deploy applications in a local Kubernetes cluster.

**Scaling Applications:** Adjusting the number of replicas for applications to handle varying loads.

**Deploying and Managing Monitoring Systems Using Nagios in Docker**:

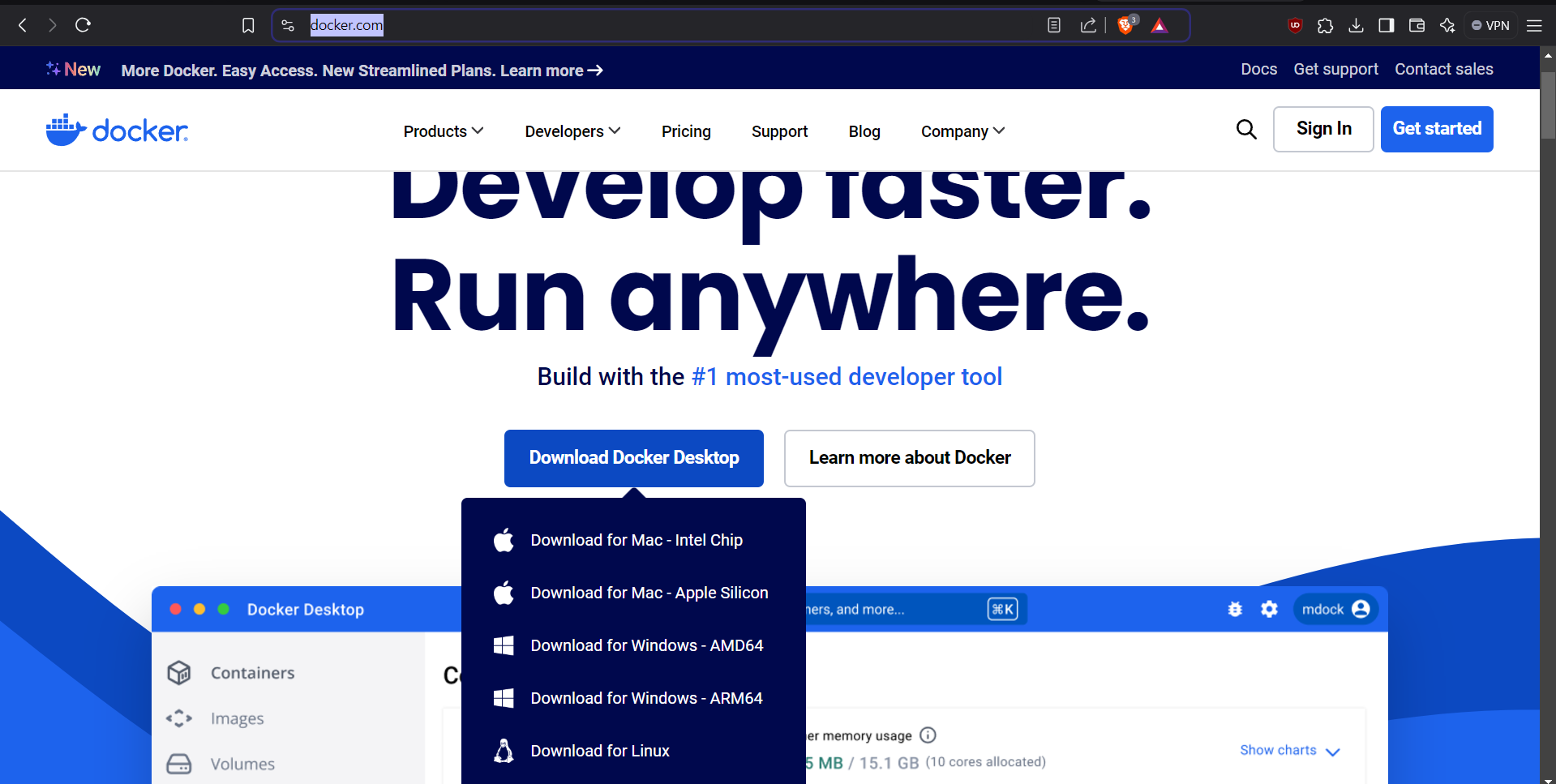
**Nagios:** Nagios is an open-source monitoring system that provides monitoring and alerting services for servers, switches, applications, and services.

**Nagios in Docker:** Deploying Nagios in Docker containers to monitor system metrics and applications.

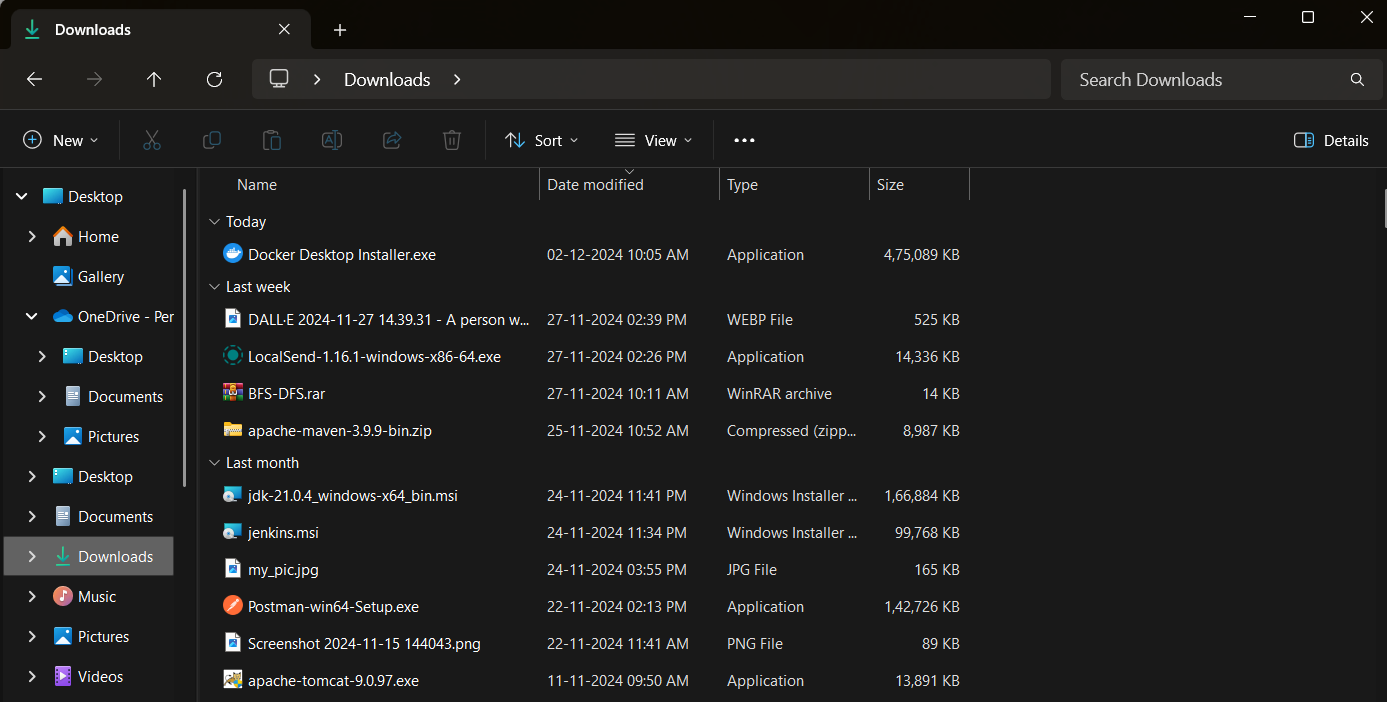
**Procedure:**

**Installation of Docker**

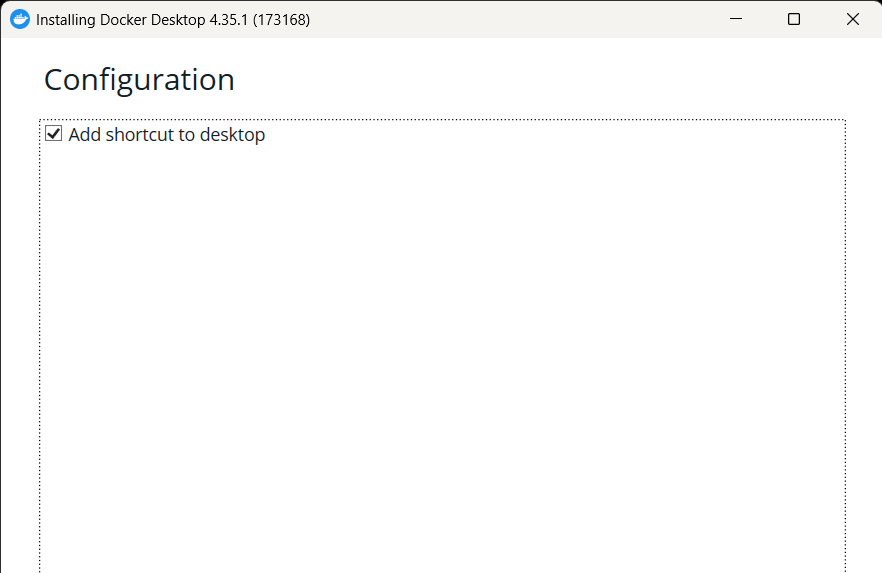
1. Go to <https://www.docker.com> and download Docker for your Operating System.



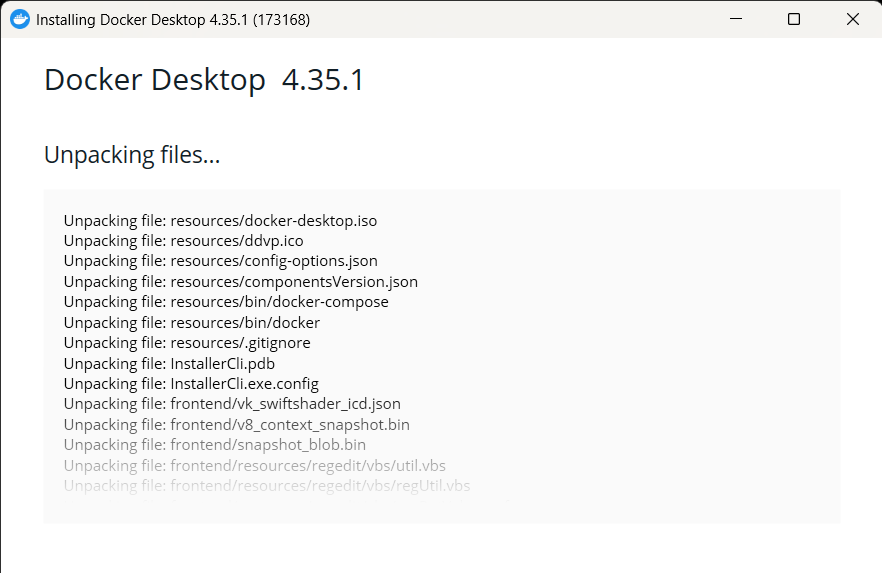
1. Run the downloaded docker installation executable.



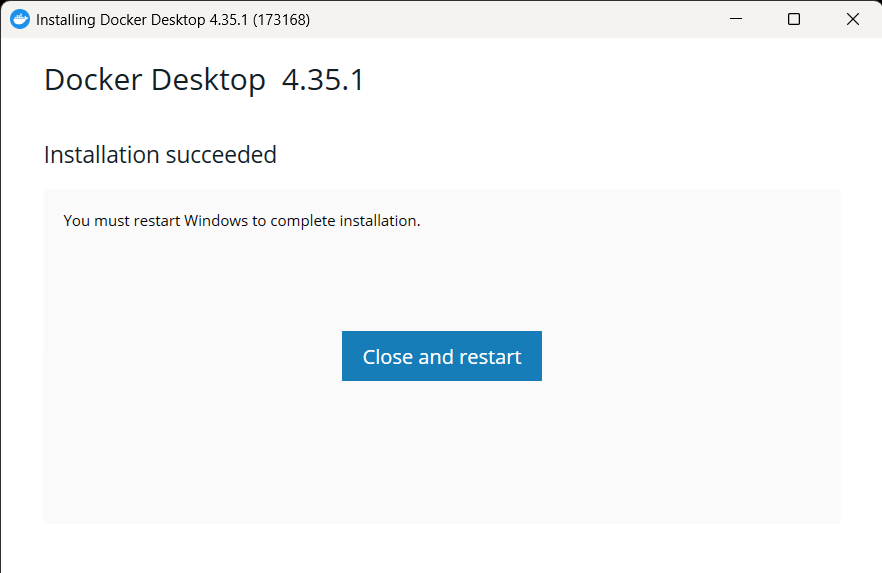
1. Click on “OK” in the configuration window.



1. Wait for Docker to finish installing.



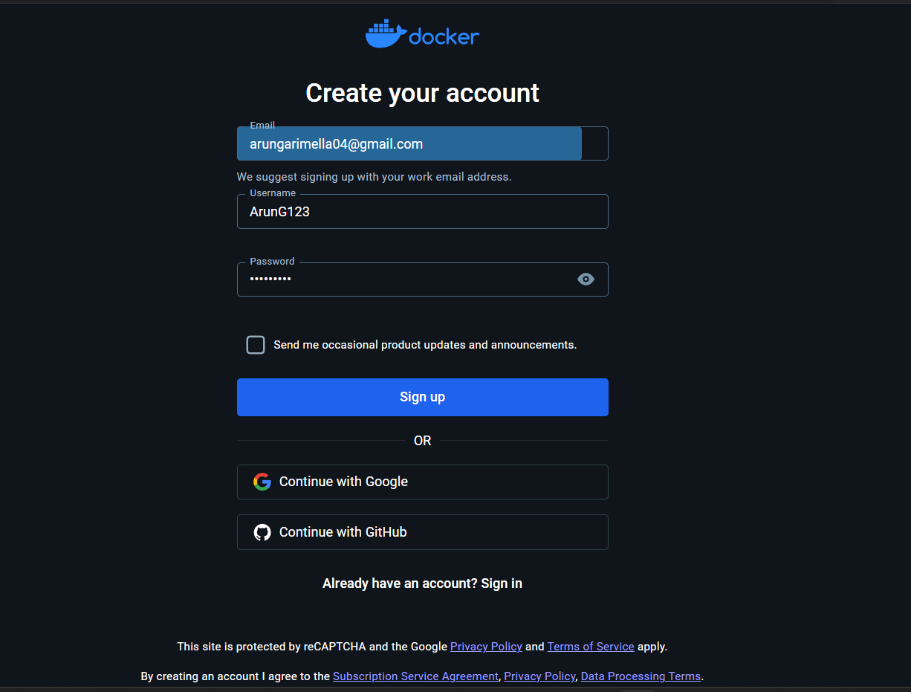
1. Click on “Close and Restart”.

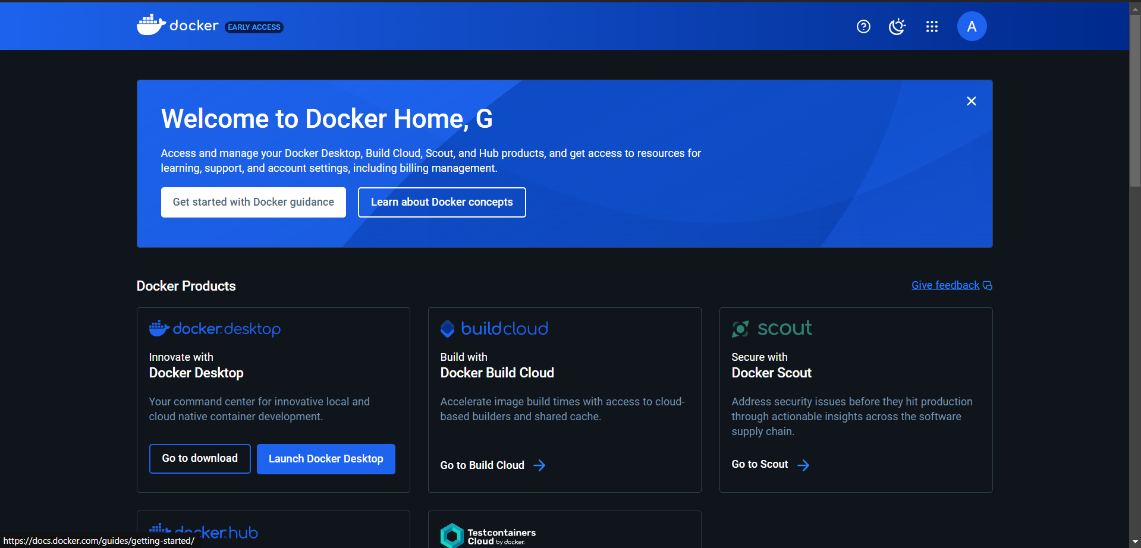


1. After restarting select “Accept”

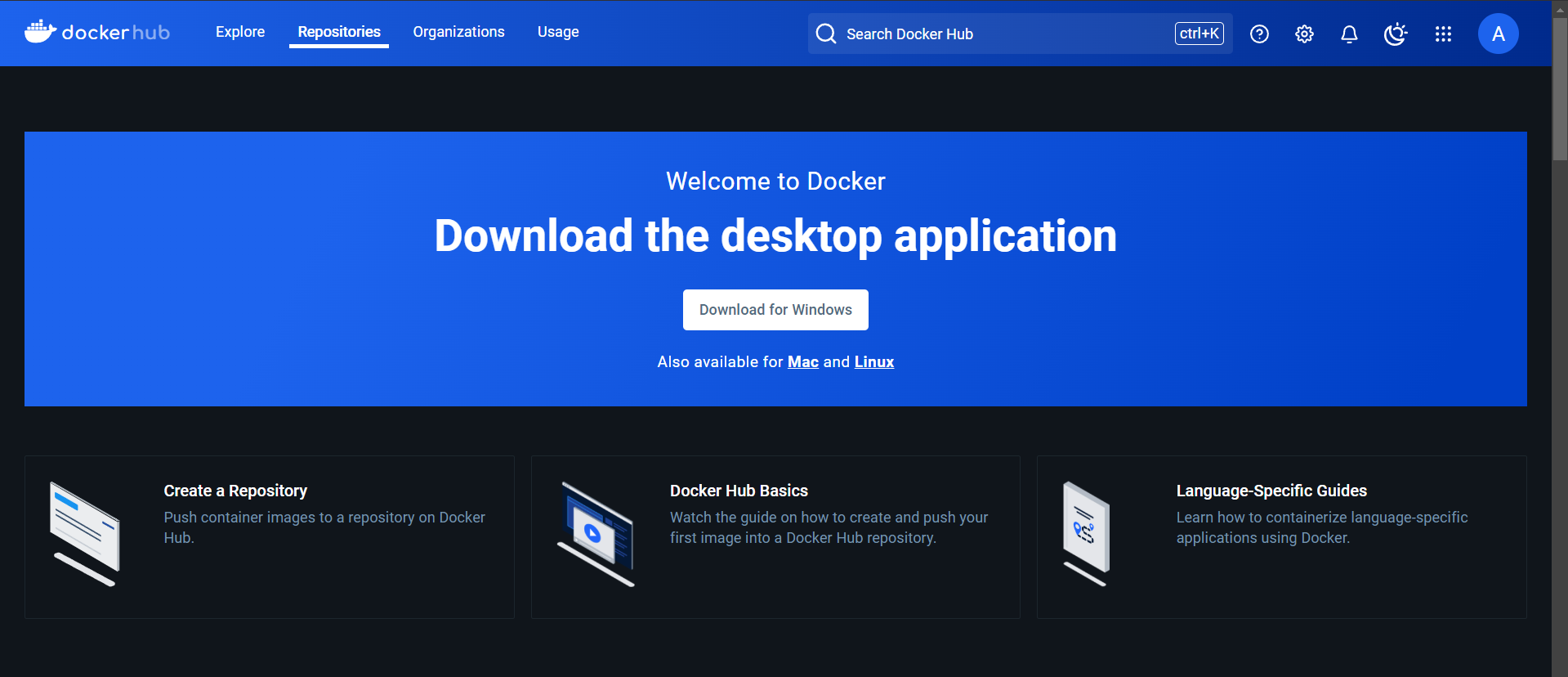


1. Create a Docker account and sign in.

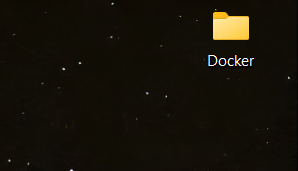




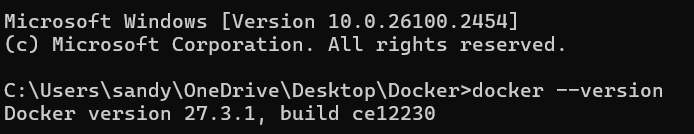
1. You have successfully created a docker hub account.



**Creation Of Images And Container And Pushing Them Into Dockerhub**

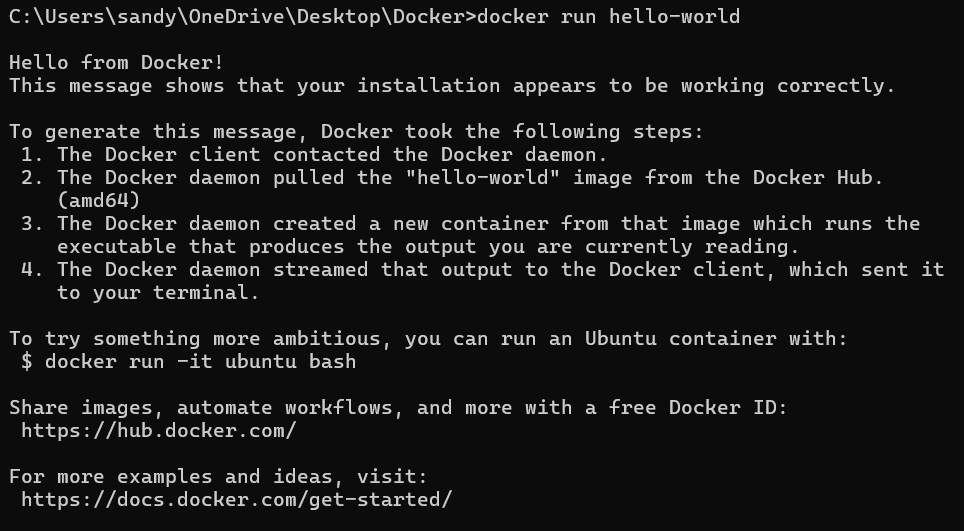
1. Sign In into Docker Hub
2. Click on personal-> continue with free
3. Go to gmail and verify your account
4. Once logged into the DockerHub,there are no containers and Images in Docker Hub
5. Create a folder on desktop as”Docker”
6. 
7. Check Docker Version

docker –version



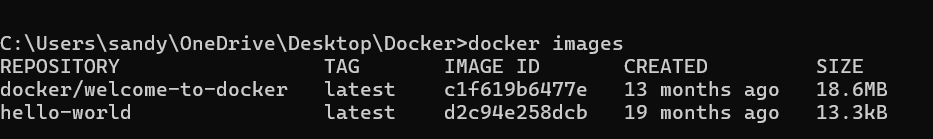
1. RUN image HELLO-WORLD IMAGE

docker run hello-world



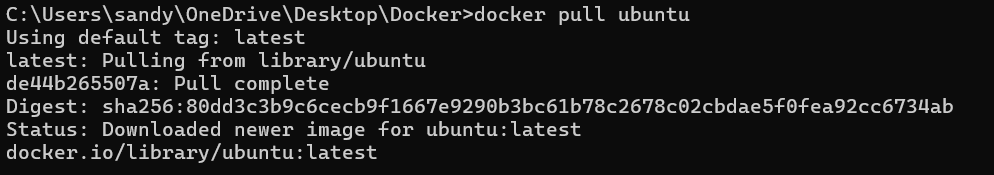
1. CHECK IMAGES

docker images



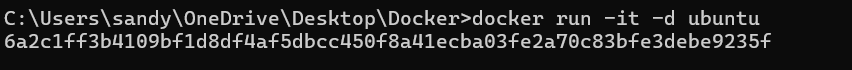
1. Pull Ubuntu Image from Docker Hub

docker pull ubuntu



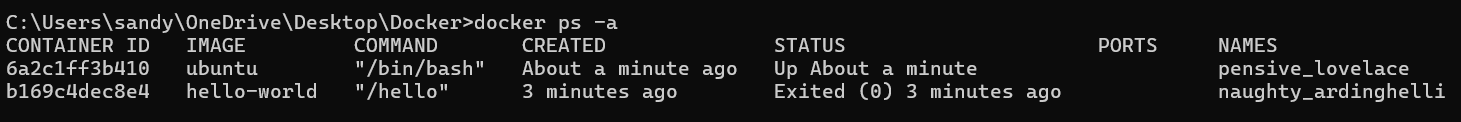
1. Run the image as docker container

docker run -it -d ubuntu



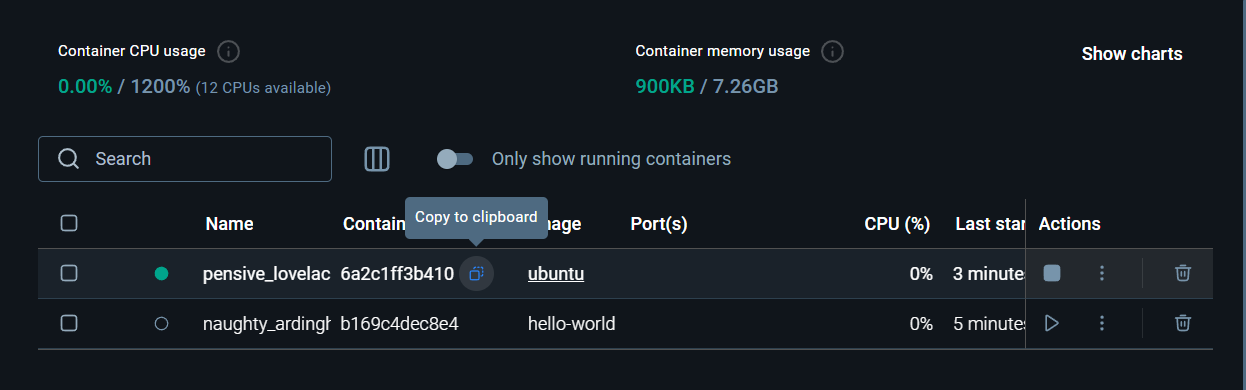
1. Check the all running and stopped containers

docker ps –a



1. During the execution of the container Ubuntu ->

copy the container ID



1. To accessing/executing a running container (ubuntu)

docker exec -it 1c6072756052 bash

1. Give echo message in the running container

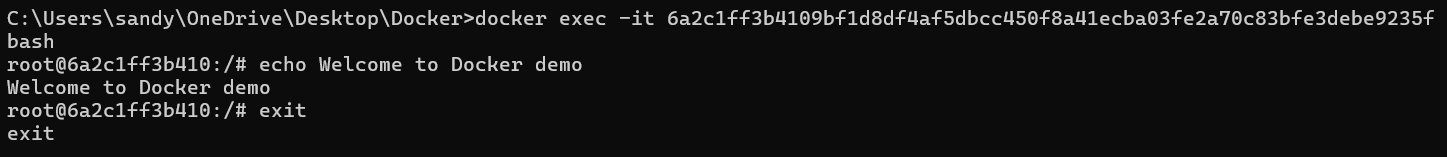
root@1c6072756052:/# echo Welcome to docker demo

Welcome to docker demo

1. Exit from the running container

root@1c6072756052:/# exit

exit



PS C:\Users>

1. To stop the running container

docker stop containerID

PS C:\Users > docker stop 1c6072756052

1c6072756052



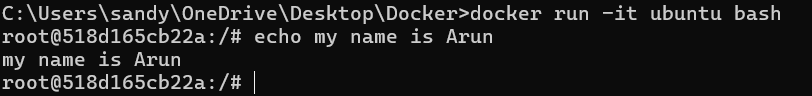
PS C:\Users >

1. run ubuntu with echo

docker run –it ubantu bash root@1c6072756052*:/# echo Bhargavi*

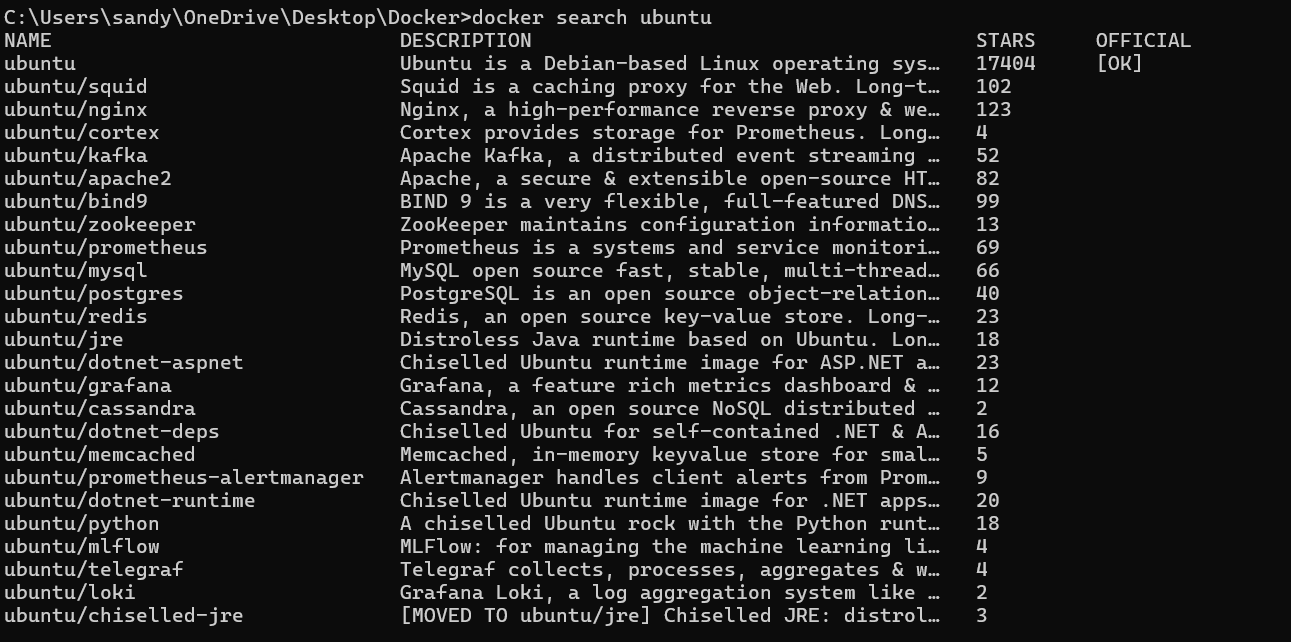
*Bhargavi*

root@1c6072756052:/#



1. Pull the ubuntu image from the docker repository and run it as a container that will provide all the features of ubuntu

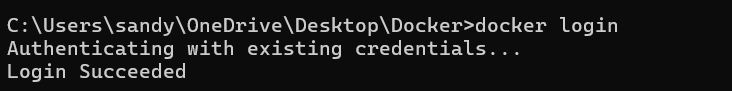
$ docker search ubuntu displays **public images in ubuntu**



1. Pushing the image from local repository to docker hub

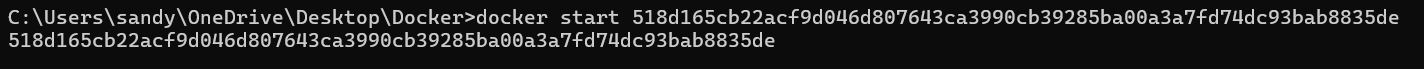
* Step1:create dockerhub account
* Step2:login using dochub
* *PS C:\Users > docker login*
* *Username:<< >>* ( here give user name)
* *Password:<< >>* ( here give user password)

*Login Succeeded*



1. Restart image

$docker start containerID used to restart the container if it has stopped due to application crash or if error occurred



**Creating Docker Image using Docker File**

1. Create two files, one file Dockerfile without extension and the other app.js (.js file contains Javascript code for execution on webpages)

**a)Creating a sample javascript file**

**Type the following commands**

$ Vi app.js

Console.log(“Docker Image using Docker File”)

Press Esc-> :wq-> to save and exit



**b)Creating Docker Image using Docker File**

1. Type the following commands
2. Node app.js will execute app.js file in node

$ Vi dockerfile

*#getting base image alpine*

*From node :alpine*

*WORKDIR / bhargavikumbham*

*COPY . /bhargavikumbham*

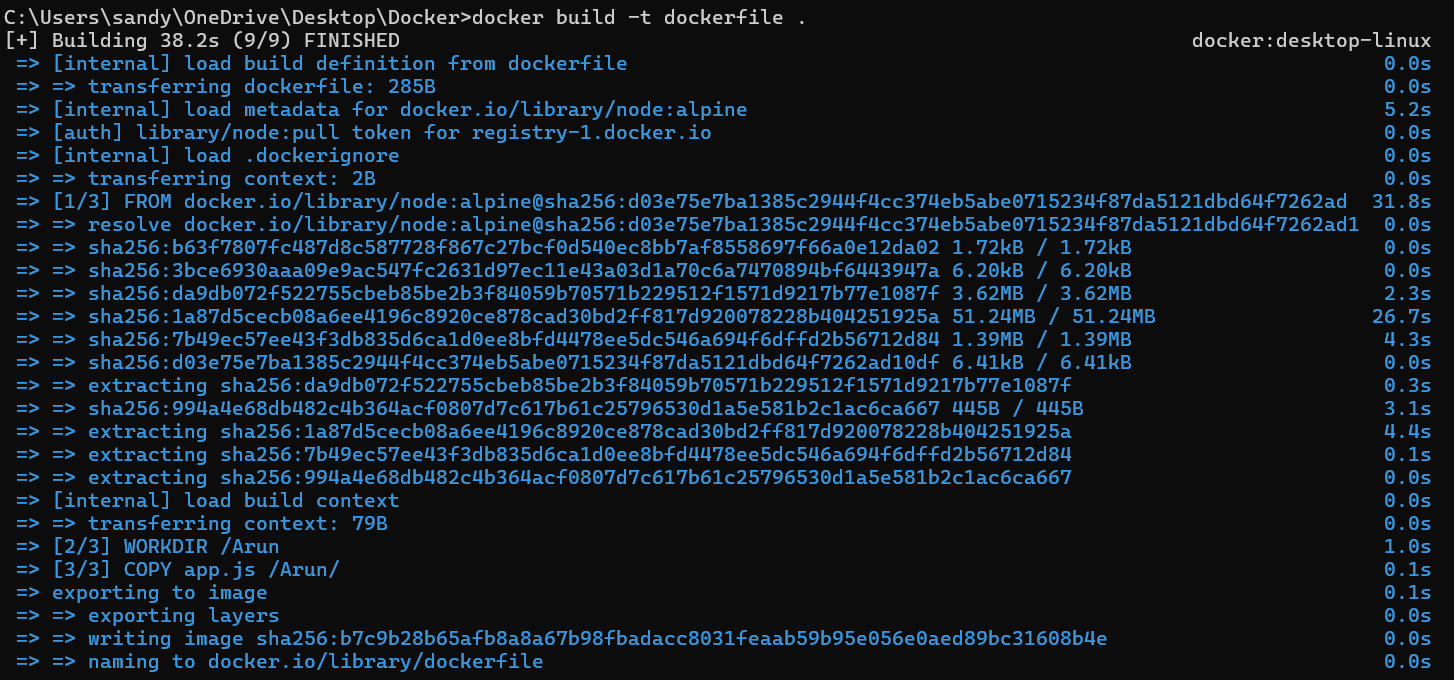
*CMD node app.js*



1. Press Esc-> :wq-> to save and exit



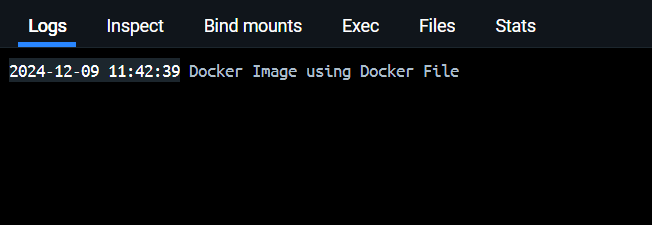
$ docker build –t myimage



$ docker images

myimage will display here

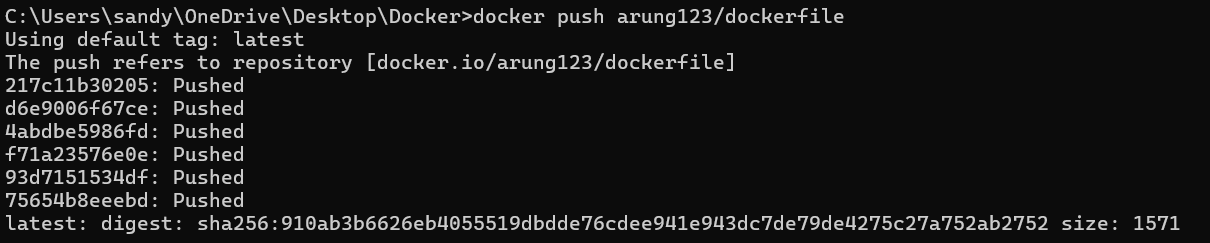
1. Go to docker desktop-> local-> check myimage will exist-> click on run
2. Diplays the console output as “Docker Image using Docker File”



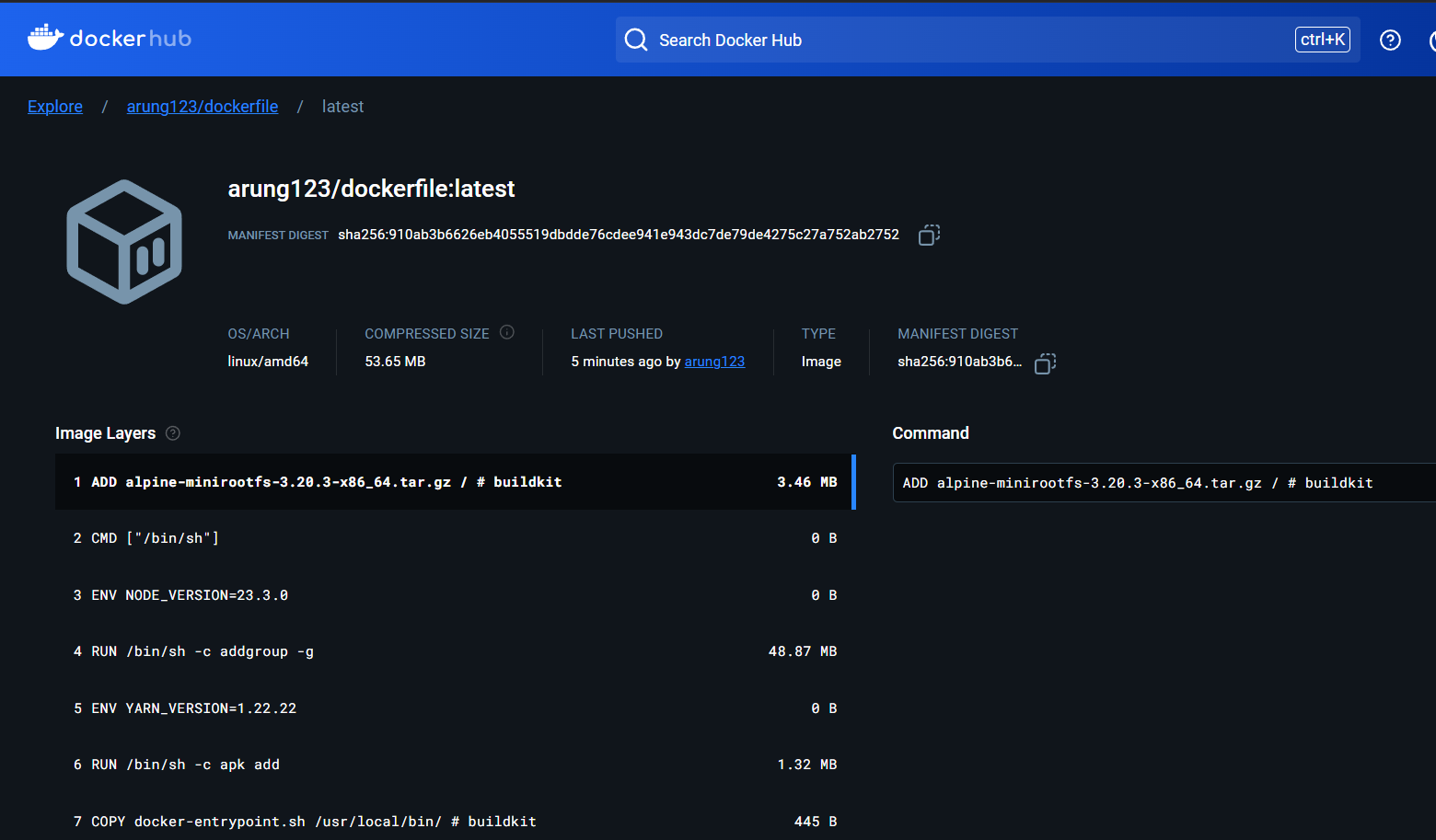
1. $ docker tag myimage bhargavikumbham/myimage



1. $ docker push bhargavikumbham/myimage



1. Go to desktop docker and check under the specified directory myimage will display

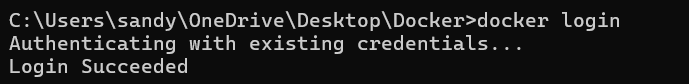


1. $ docker logout

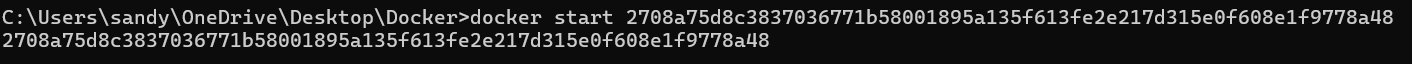


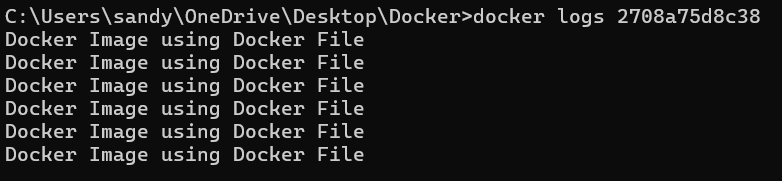
**How to push images from local rep to docker hub rep:**

1. Step1:create dockerhub account
2. Step2:login using dochub
3. *PS C:\Users > docker login*
4. *Username:<< >>* ( here give user name)
5. *Password:<< >>* ( here give user password)
6. *Login Succeeded*



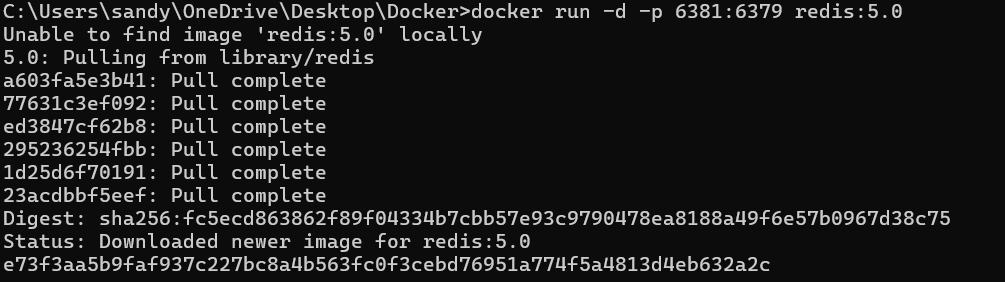
1. 13) To restart image
2. $docker start containerID used to restart the container if it has stopped due to application crash or if error occurred



1. 14) Working with container in interactive mode
2. docker container exec -it containerID 15) Troubleshooting containers using logs $docker logs 505174ea315 Helps to see what’s going on in container
3. 16) Creating same image with different version in one application 
4. docker run redis:latest



1. docker run redis:5.0



1. Here two redis image

**Docker Compose, Installation Of Minikube**

**And Working With It**

Here’s a basic example of a Docker Compose file that runs WordPress and MySQL together:

**version: '3.8' # Docker Compose file format version**

**services:**

**wordpress: # WordPress service**

**image: wordpress:latest**

**ports:**

**- "8080:80" # Map port 80 of the container to port 8080 of the host**

**environment:**

**WORDPRESS\_DB\_HOST: db:3306 # Database host**

**WORDPRESS\_DB\_USER: wordpress**

**WORDPRESS\_DB\_PASSWORD: wordpress**

**WORDPRESS\_DB\_NAME: wordpress**

**depends\_on:**

**- db # Ensures the db service starts first**

**db: # MySQL service**

**image: mysql:5.7**

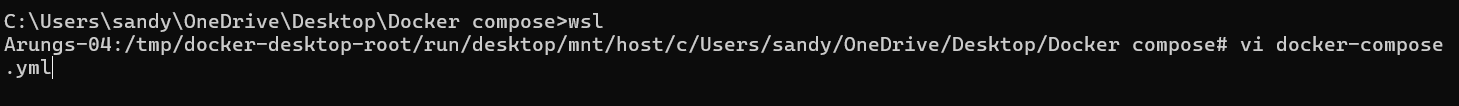
**environment:**

**MYSQL\_ROOT\_PASSWORD: rootpassword**

**MYSQL\_DATABASE: wordpress**

**MYSQL\_USER: wordpress**

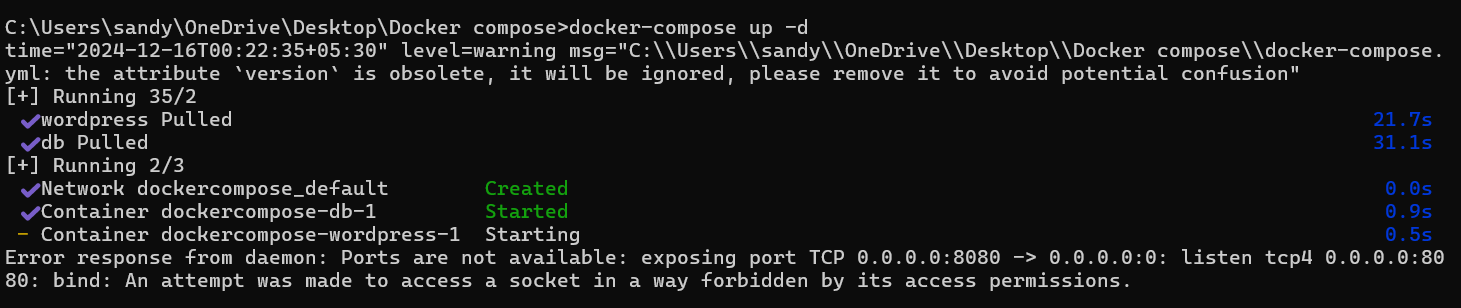
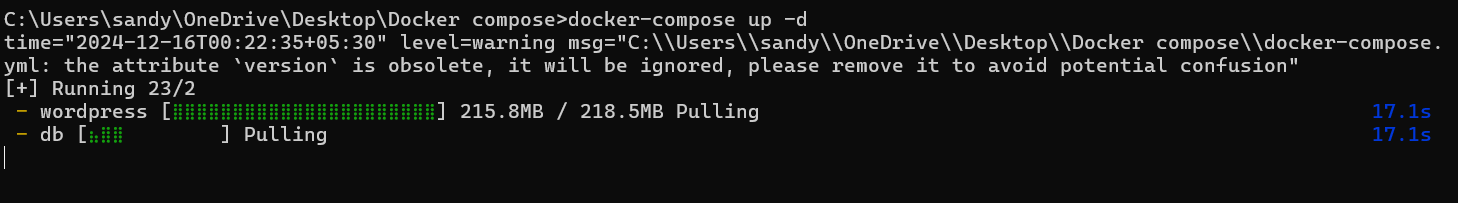
**MYSQL\_PASSWORD: wordpress**



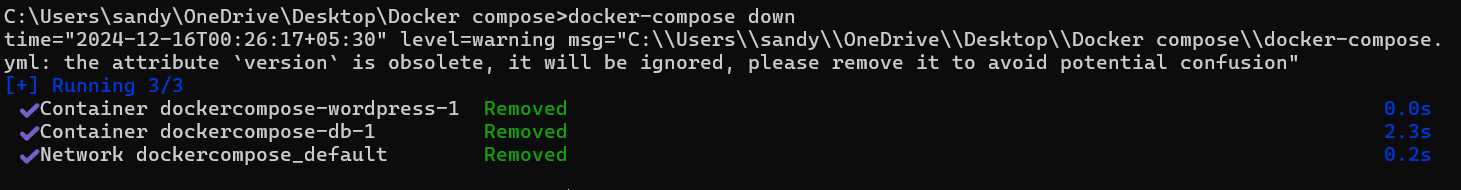
**How to Run It**

To run a multi-container setup like the one above:

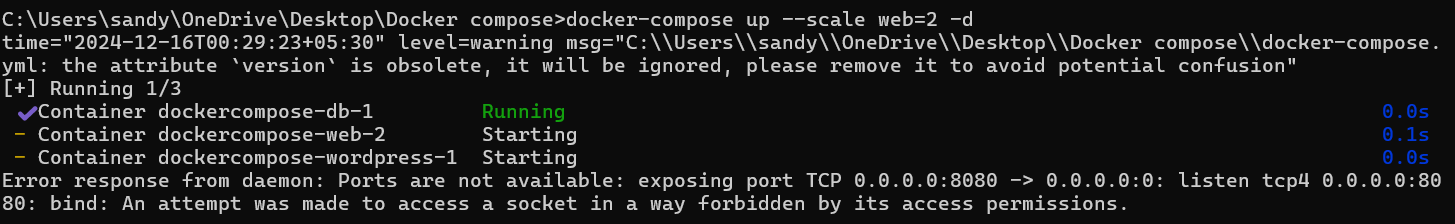
1. Save the file as docker-compose.yml.

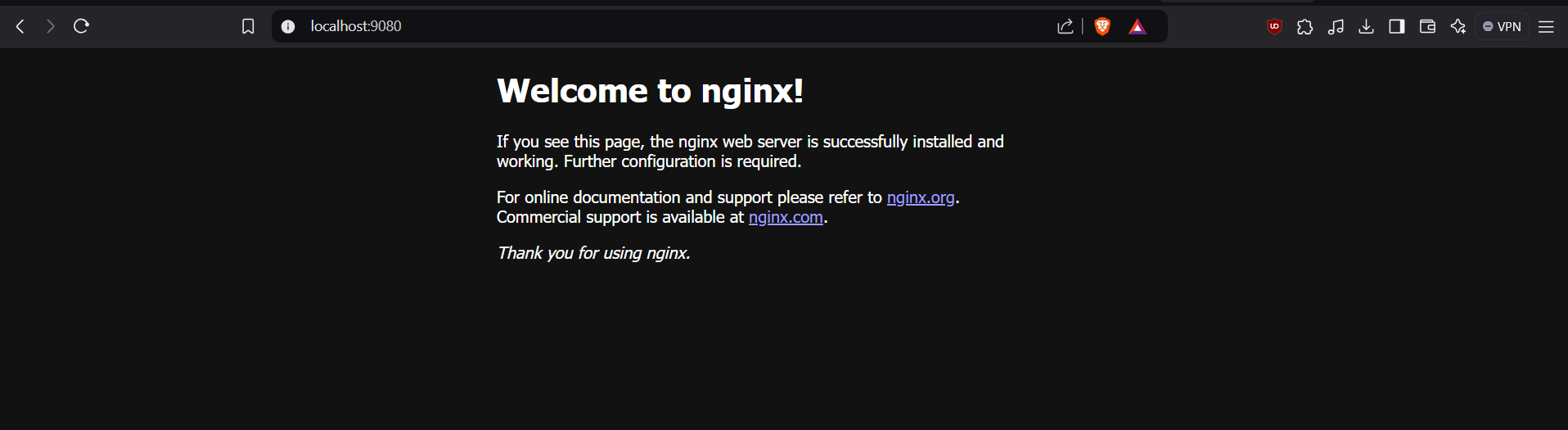
docker-compose up –d

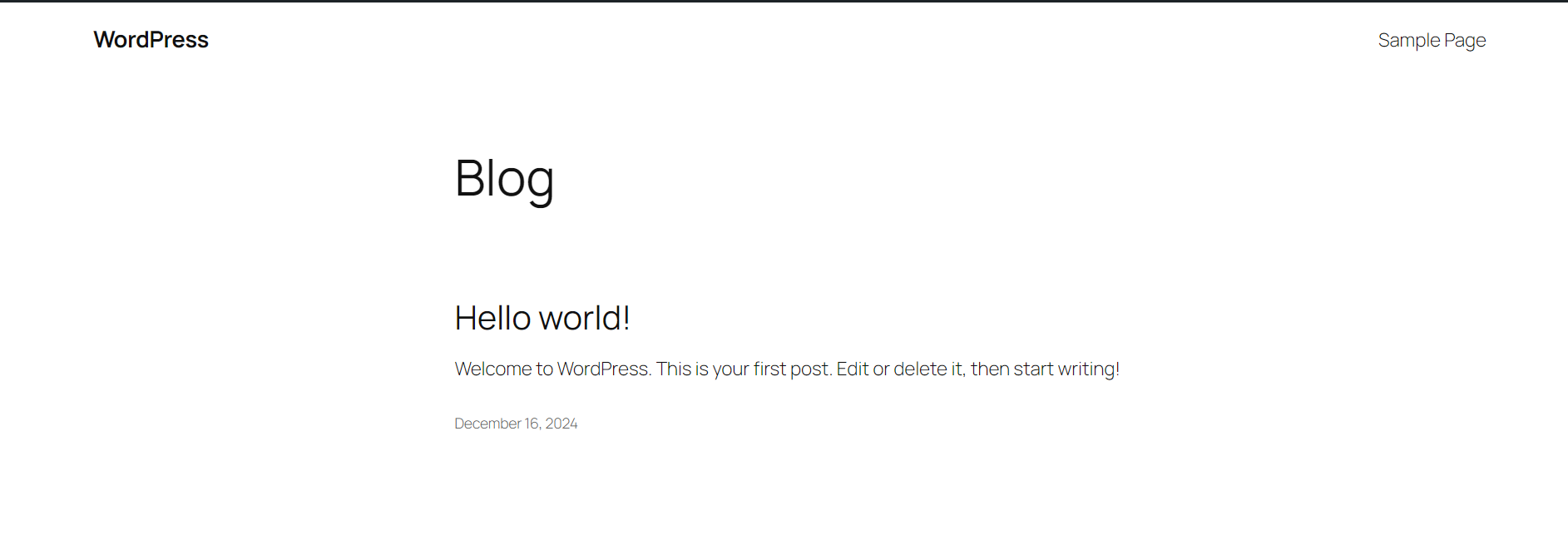
1. To stop the containers

 docker-compose down

1. To scale the container

docker-compose up --scale <service name>=2 -d





**MiniKube Installation on PC**

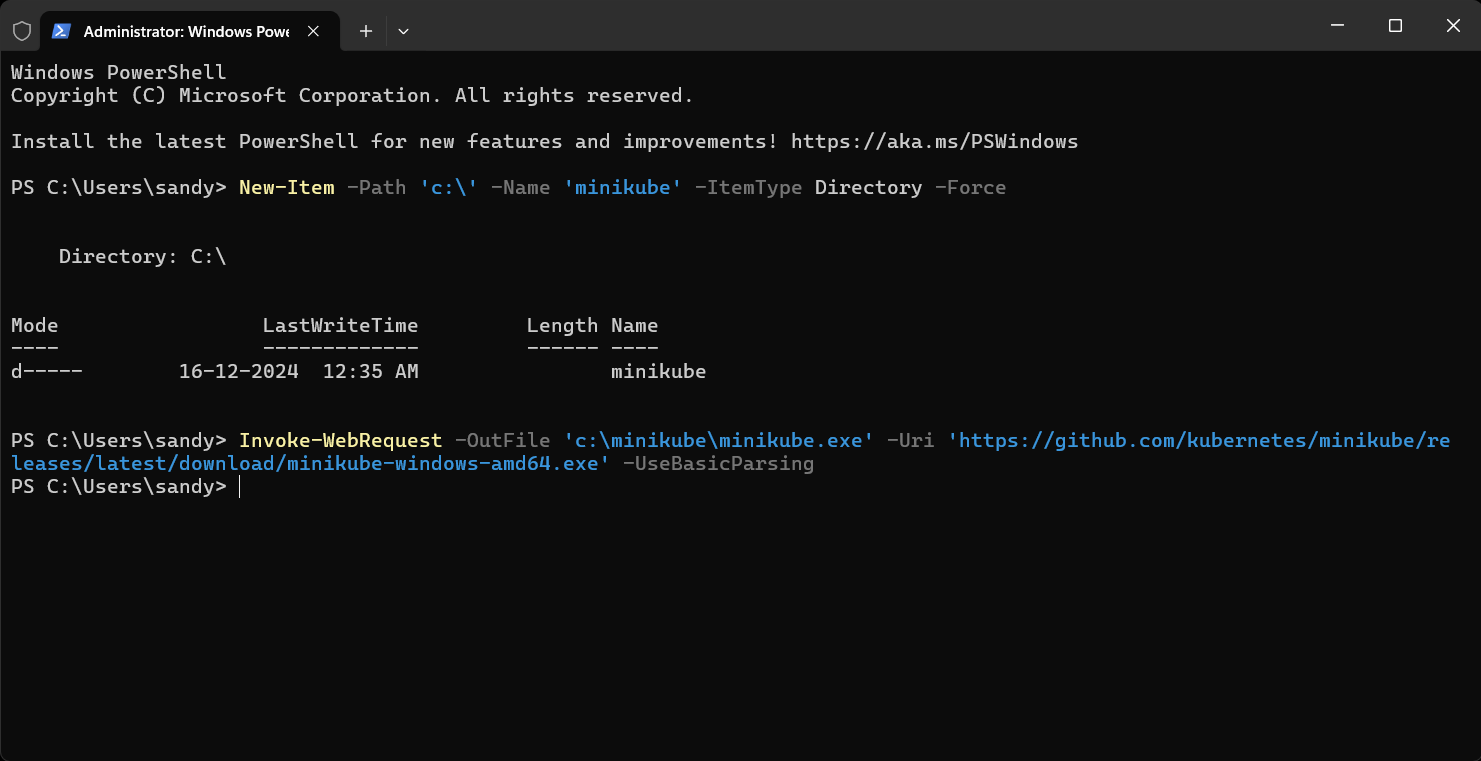
### ****Step 1:** Install Prerequisites,**

Before installing Minikube, ensure the following are installed:

1. **Virtualization Support**:
   * Verify virtualization is enabled:
2. **Hypervisor**:
   * Minikube supports multiple hypervisors (e.g., **Hyper-V**, **VirtualBox**, or **Docker** as a driver).
   * Install one of the following:
     + **Hyper-V** (pre-installed on Windows 10/11 Pro or Enterprise).
     + **Docker Desktop** (if you want to use Docker as the driver).

### ****Step 2:** Download Minikube**

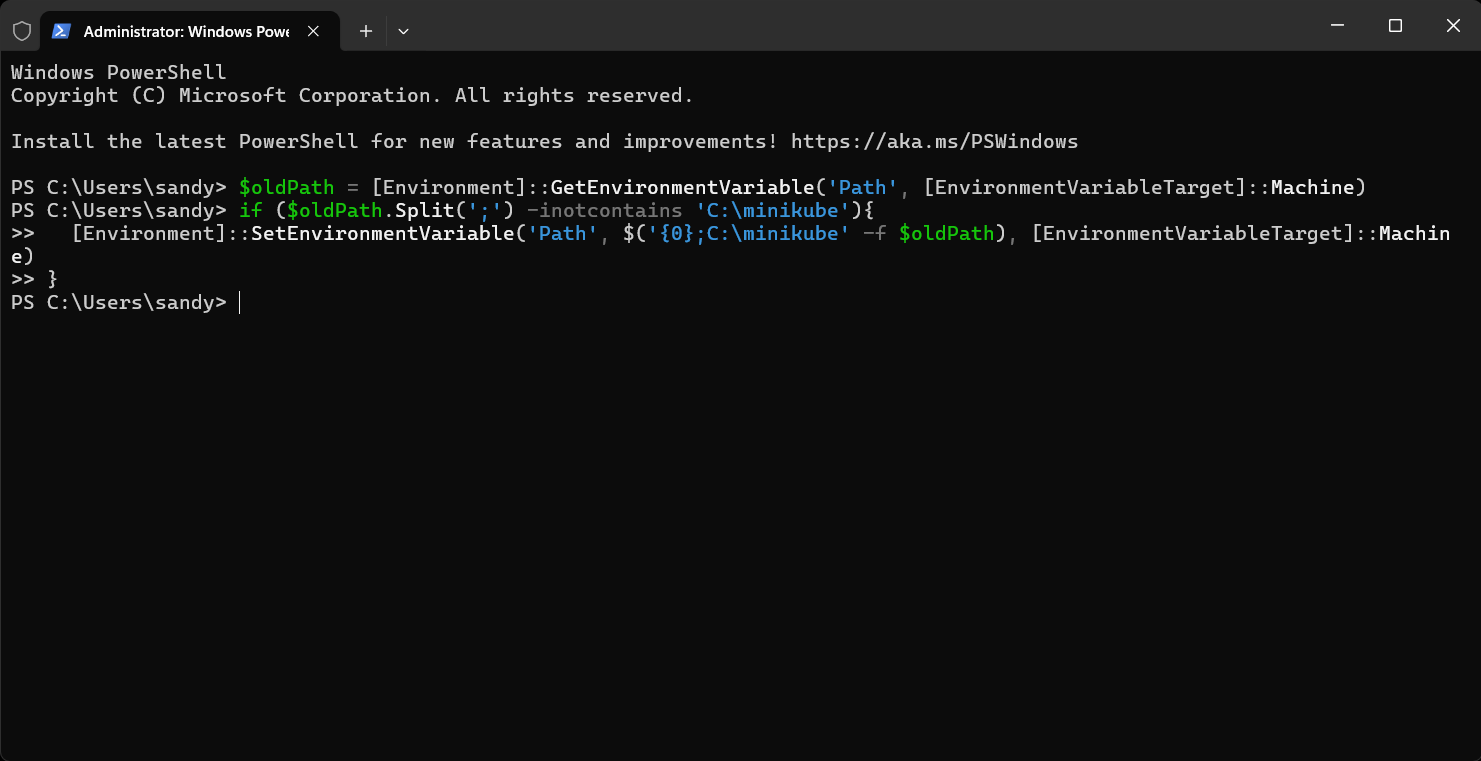
1. Open a PowerShell or Command Prompt with administrator privileges.
2. Download the latest Minikube executable using this command:
3. curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-installer.exe
4. Install Minikube by running the installer:
5. .\minikube-installer.exe



### ****Step 3:** Add Minikube to PATH**

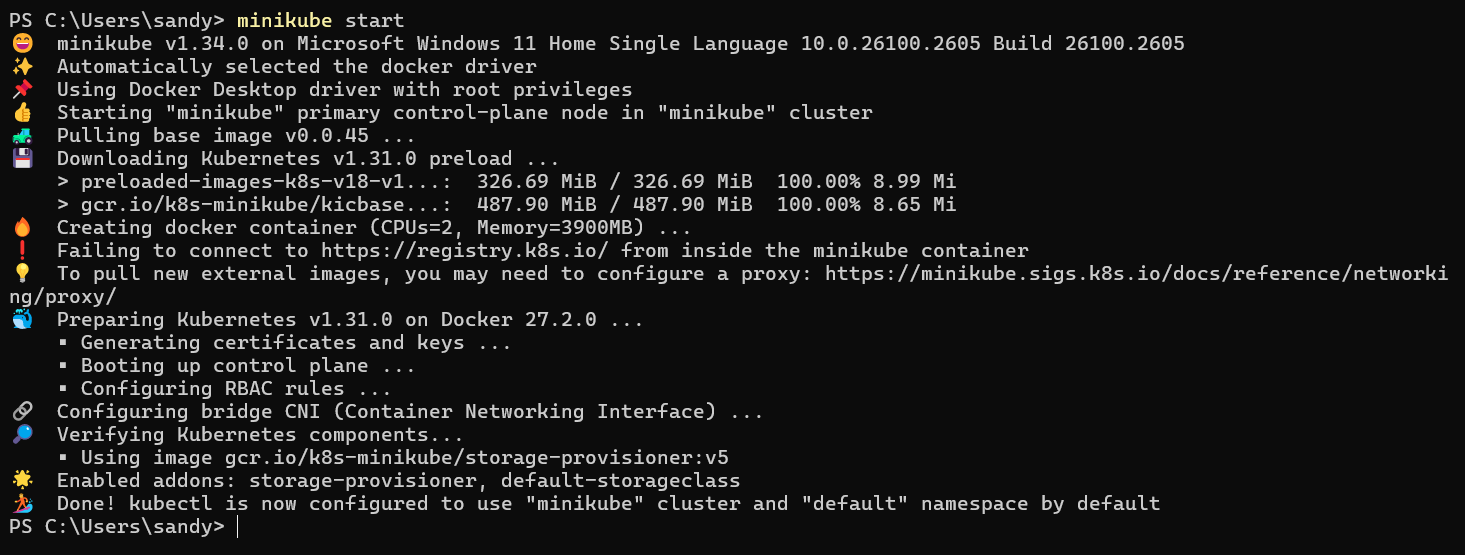
If Minikube is not automatically added to your PATH during installation:

1. Open **System Properties** → **Environment Variables**.
2. Add the directory where Minikube is installed (e.g., C:\Program Files\Minikube) to your PATH variable.

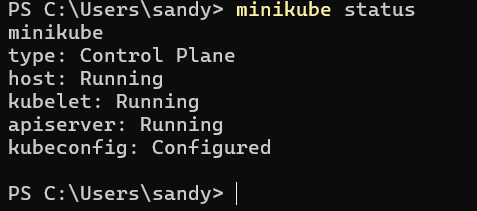


### **Step 4: Start Minikube**

1. Open a terminal (PowerShell or CMD).
2. Start Minikube with a specified driver (e.g., Hyper-V, Docker, or VirtualBox). For example:
   * **Hyper-V**:
   * minikube start --driver=hyperv
   * **Docker**:
   * minikube start --driver=docker

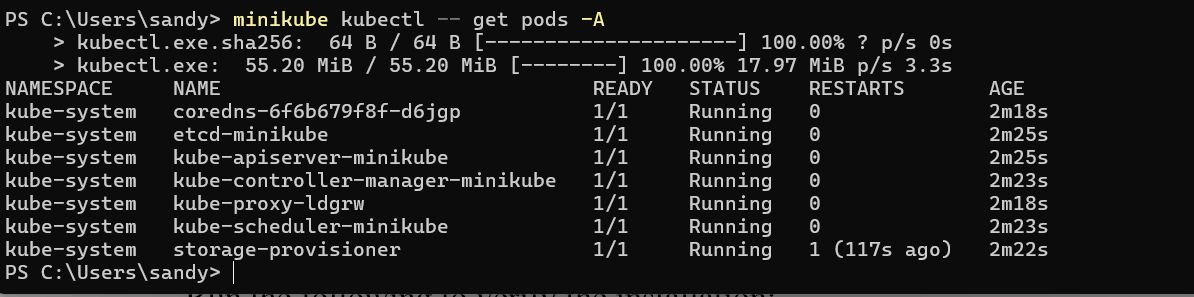


1. Verify Minikube is running:

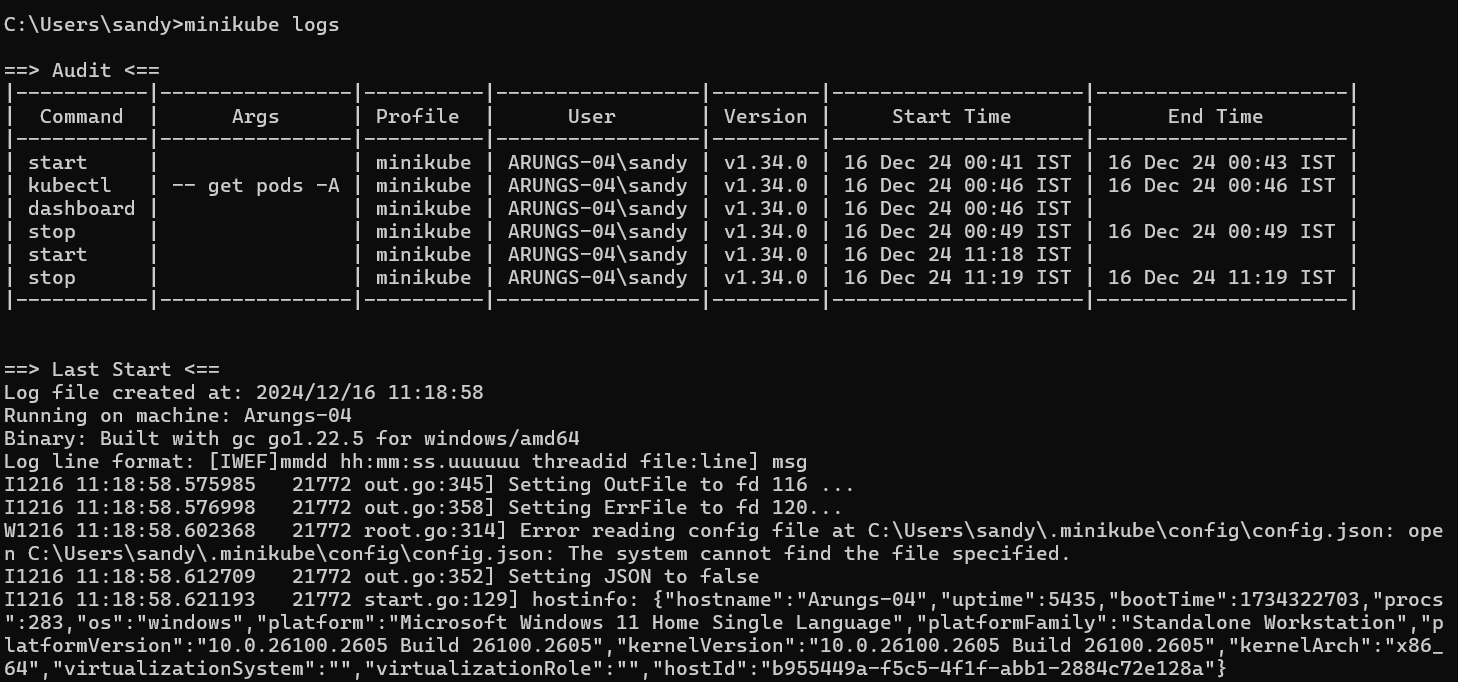
minikube status

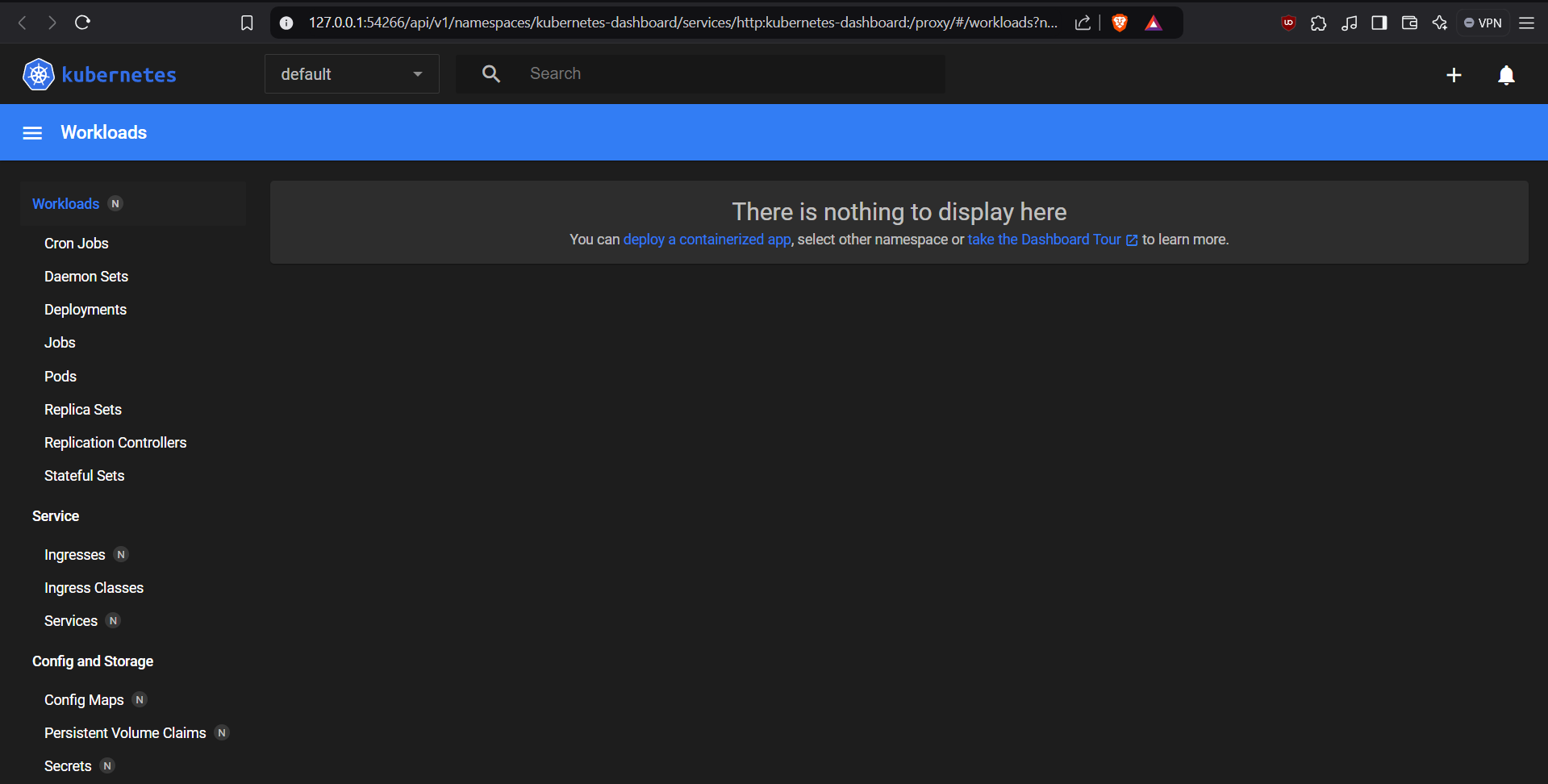
### **Step 5: Interact with Minikube**

Once Minikube is running:

1. Use kubectl to interact with the cluster.
   * Install kubectl if not already installed:
   * minikube kubectl -- get pods -A
   * Or download it separately from the [official Kubernetes site](https://kubernetes.io/docs/tasks/tools/install-kubectl/).
2. Open the Minikube dashboard (optional):

minikube dashboard

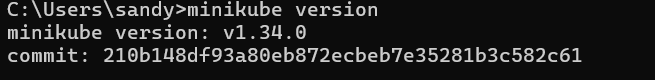




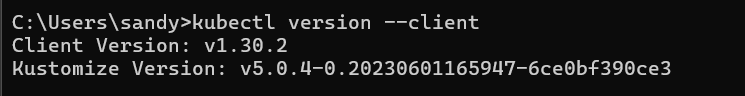
### **Optional: Check Your Installation**

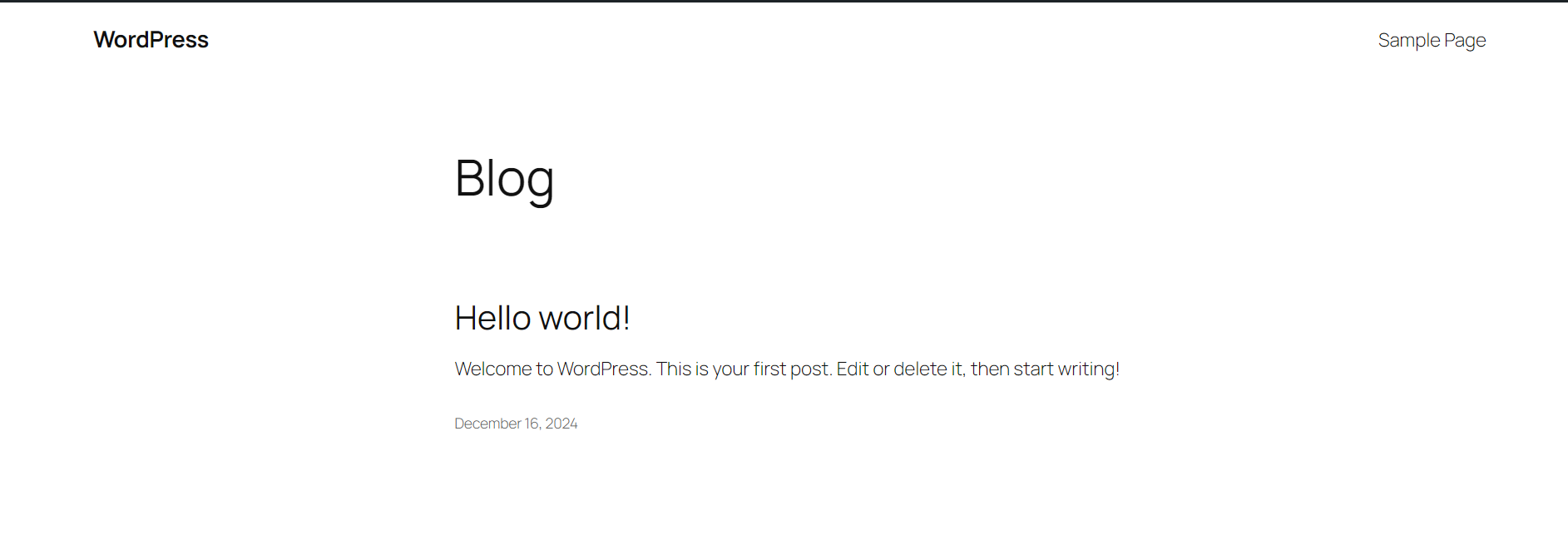
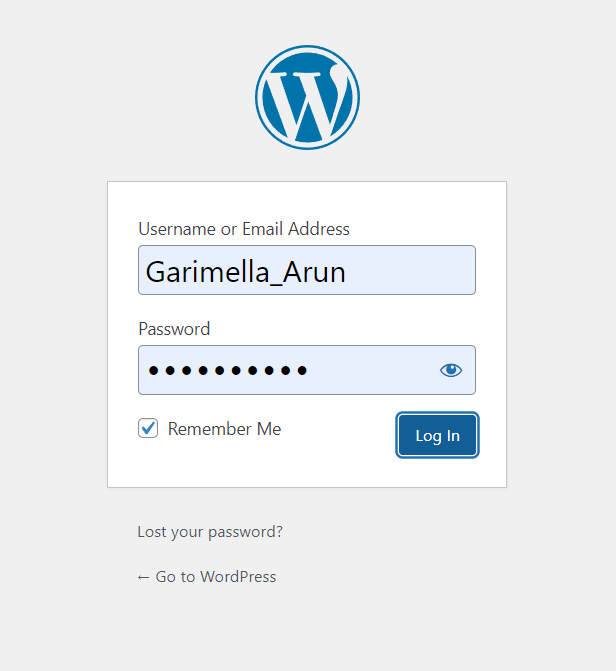
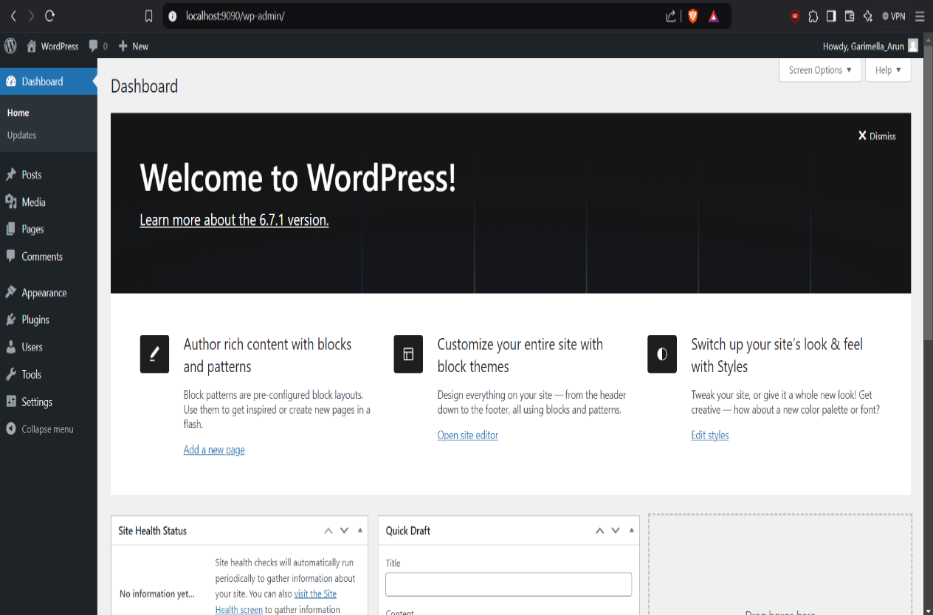
Run the following to verify the installation:

minikube version



kubectl version –client

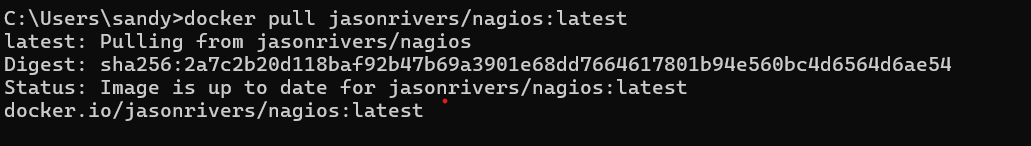


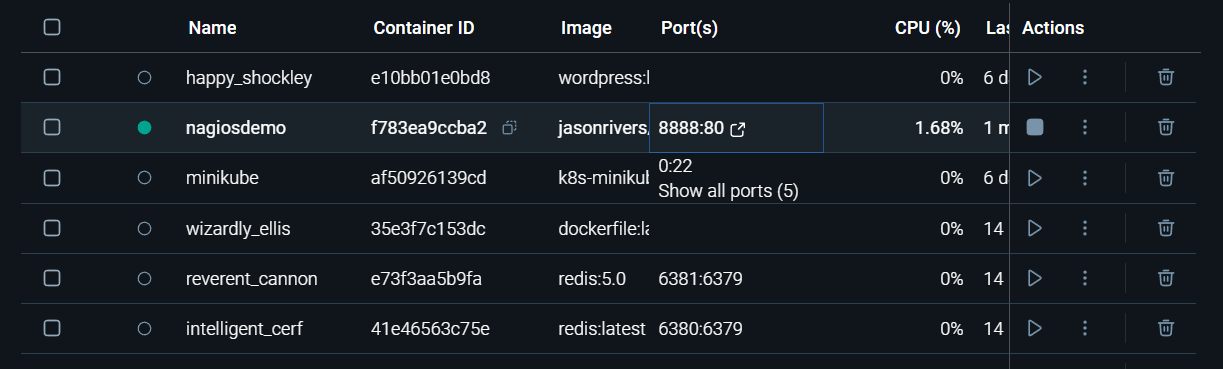
**WordPress installation:**

**NAGIOS INSTALLATION**

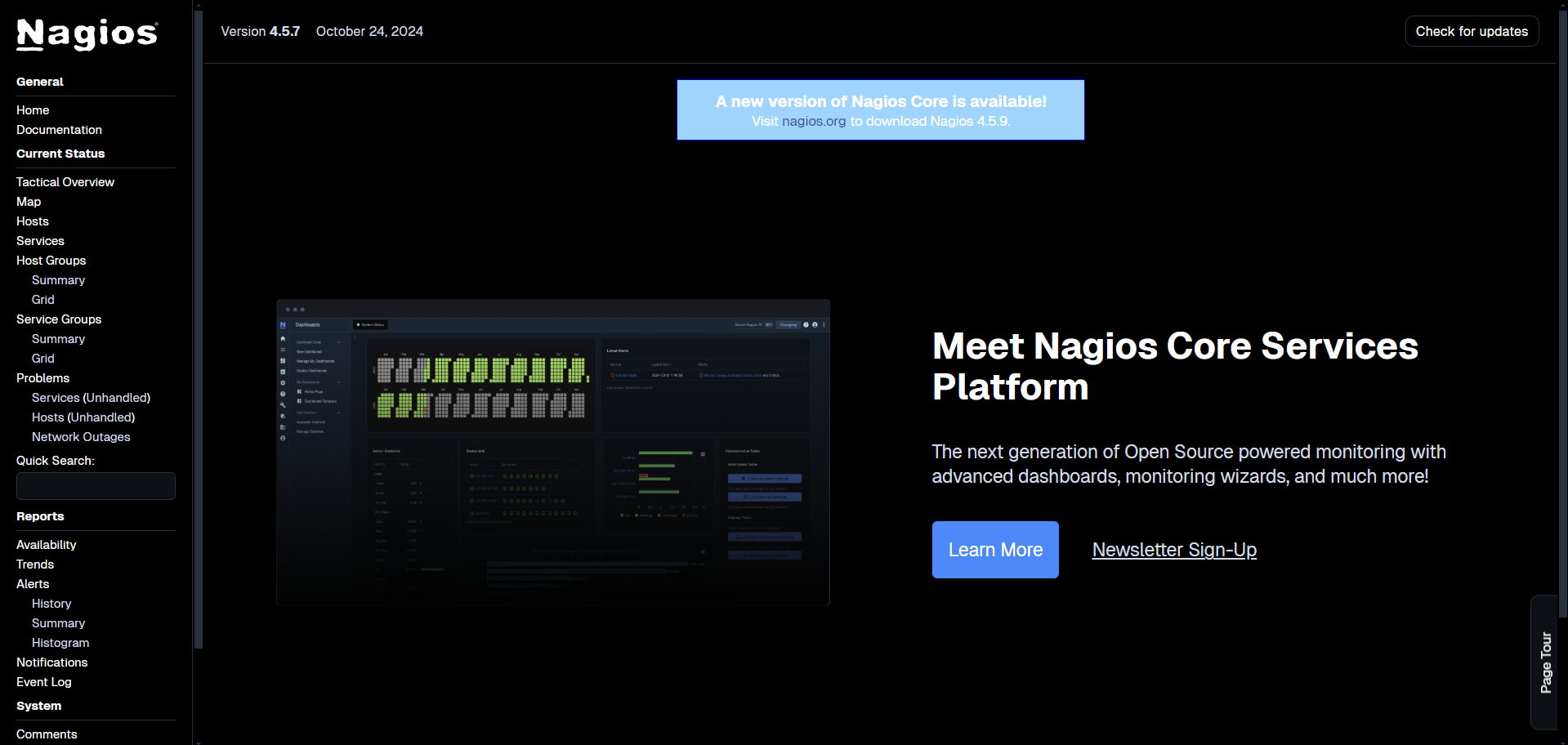
1. run the command in terminal

docker pull jasonrivers/nagios:latest



1. run the command
2. docker run -name nagiosdemo -p 8888:80 jasonrivers/nagios:latest

3. open<http://localhost:8888/> in your browser and type your login credentials.

4.Nagios dashboard will appear.

**Result:**

Gained hands-on experience with Docker, Minikube, and DockerHub, including installing and configuring these tools, executing Docker CLI commands, modifying and pushing Docker images, creating and pushing Dockerfile images, running multiple containers with Docker Compose, and deploying and scaling applications with Minikube. Learnt how to deploy and manage monitoring systems using Nagios in Docker, enhancing skills in containerization, orchestration, and system monitoring.