DOCKER

Long gone are the days of Linux container management, with the rapid increase in popularity of the cloud technology we know that we need more robust solutions for containerization of our apps along with all the tools needed or their management and docker is our best bet to achieve this.

Why do we need containerization??

Computer resources are the backbone or the cloud. On AWS (amazon web services) we have Amazon elastic Compute cloud (EC2) which allows us to launch virtual machines in the cloud and make scale computing easier for developers.

Similarly, in Google Cloud Platform we can create virtual machines with compute engine.

We choose an operating system and also other resources like RAM and number or CPU's for that virtual machine. Cloud services provides us with all the necessary tools to manage our virtual machines and also provides security to them.

The problem with virtual machines

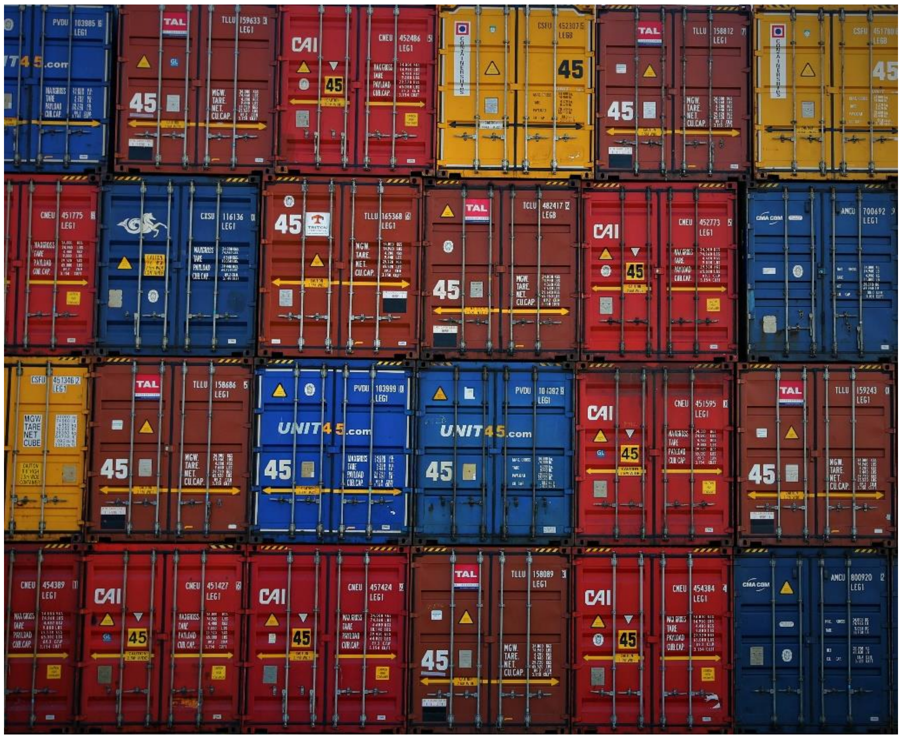
The biggest problem with running virtual machines is scalability. For e.g. Imagine that the app you built on the VM gains popularity and its users increases so much that they max out the CPU and memory of the VM.

One option is to make the VM more powerful by adding more CPU's and memory to the VM. This is called as *Vertical Scaling*.

The other option is to create more VM's and distribute the workload among them. This is known as *horizontal scaling*.

With horizontal scaling comes the problem is of load balancing i.e. one VM gets most of the traffic while others just sit idle.

Containerization lets us free of worrying about scaling our infrastructure at all.



Docker comes into play

Containerization is defined as a form of operating system virtualization, through which applications are run in isolated user spaces called containers, all using the same shared operating system (OS). Every container is isolated from each other.

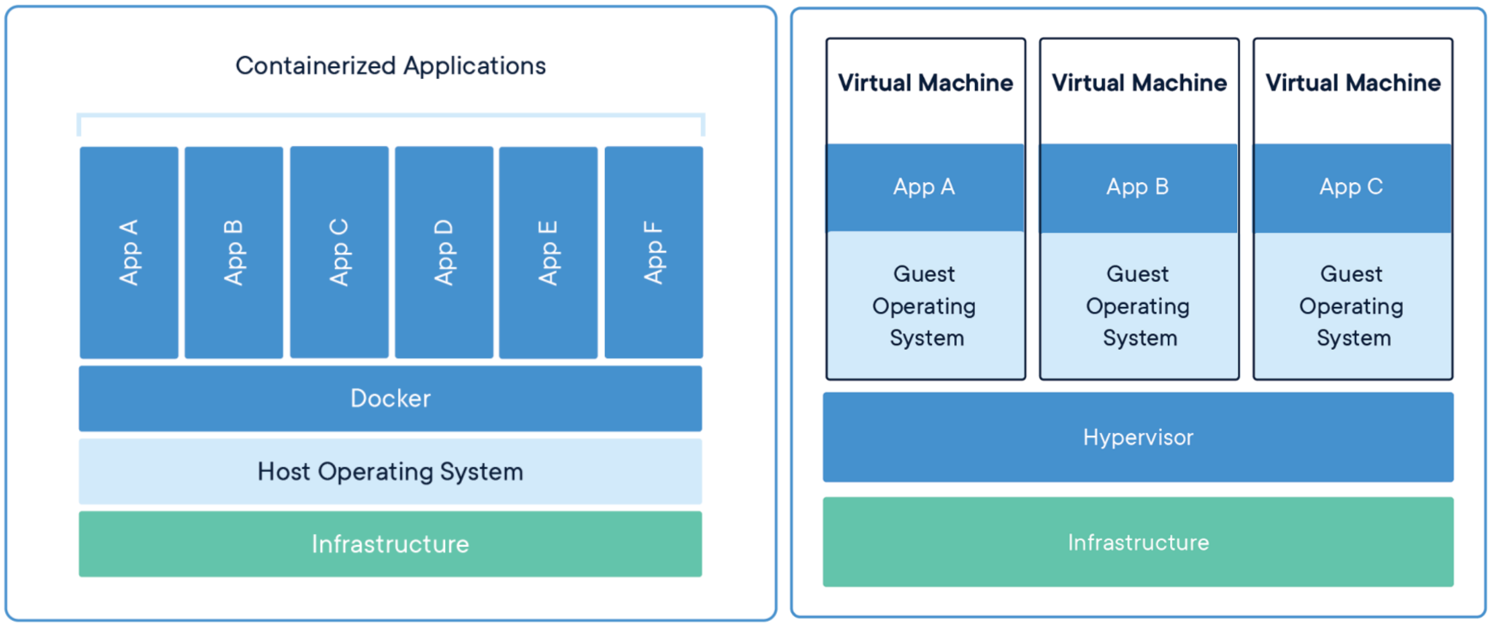
According to Docker:

> *'A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.'*

Yes, I know you might be thinking that they both sounds the same, but there are major differences under the hood:

- Every docker container share the same system resources. Therefore, every container has access to complete system resources.
- They do not have complete operating systems inside.
- They are very lightweight as compared to virtual machines because they only contain application level dependencies.

How the containers work is a more complicated concept, and here I'm not going into that.



Docker allows the developers to put their applications in a container and then deploy it to any cloud service. The clouds have container registries where we can upload our containers and use them across different services.

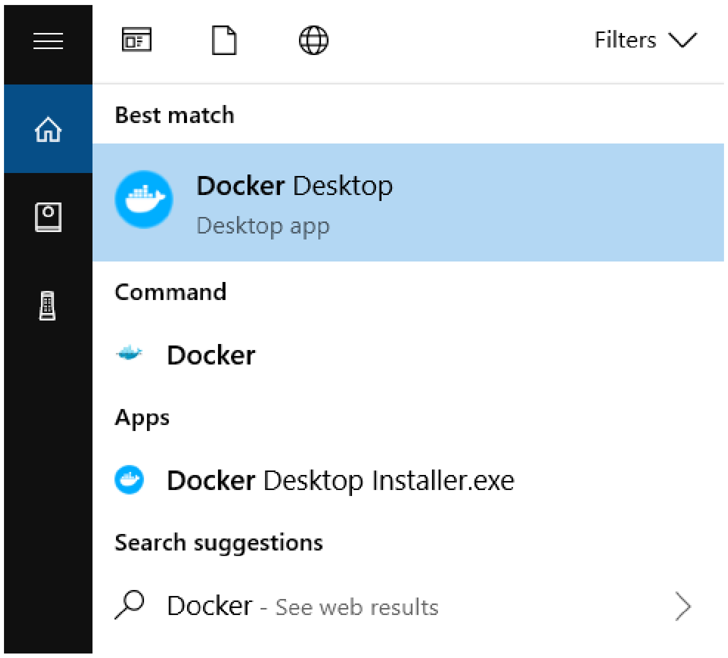
Installing Docker

Install Docker Desktop on windows

1. System Requirements
   - Windows 10 64-bit: Pro, Enterprise, or Education (Build 15063 or later).
   - Hyper-V and Containers Windows features must be enabled.
   - The following hardware prerequisites are required to successfully run Client Hyper-V on Windows 10:
     - 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.

2. Installation
- Double-click Docker Desktop Installer.exe to run the installer.

- If you haven't already downloaded the installer (Docker Desktop Installer.exe), you can get it from Docker Hub. It typically downloads to your Downloads folder, or you can run it from the recent downloads bar at the bottom of your web browser.
- Follow the instructions on the installation wizard to accept the license, authorize the installer, and proceed with the install.
- When prompted, authorize the Docker Desktop Installer with your system password during the install process. Privileged access is needed to install networking components, links to the Docker apps, and manage the Hyper-V VMs.
- Click Finish on the setup complete dialog and launch the Docker Desktop application.



- When the whale icon in the status bar stays steady, Docker Desktop is up-and-running, and is accessible from any terminal window.

Installing Docker in Ubuntu

You can follow the steps in the given link to install docker in Ubuntu.

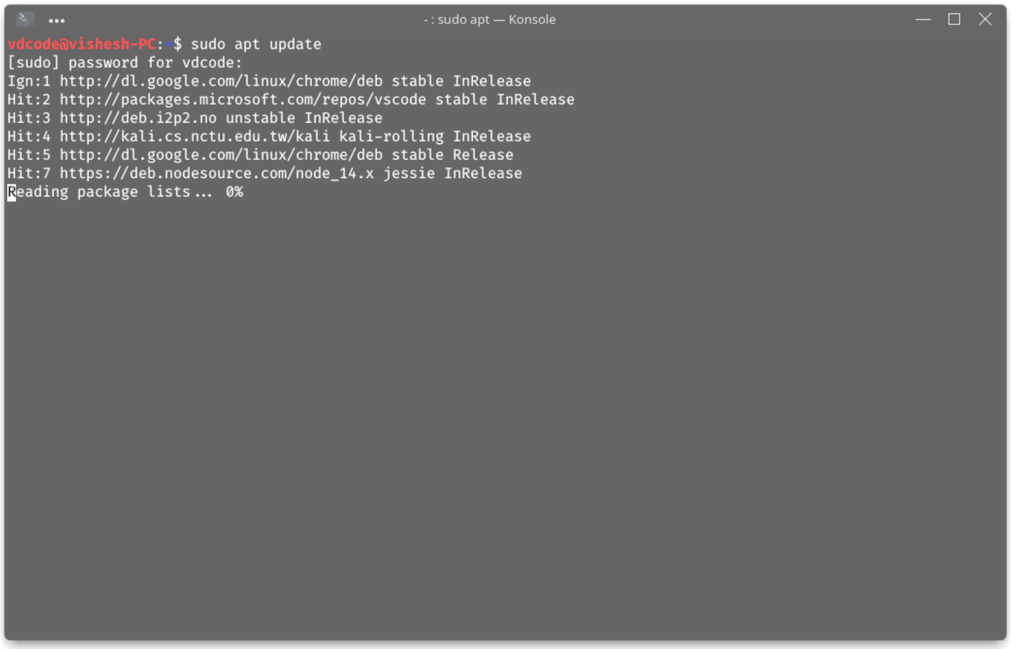
https://docs.docker.com/engine/install/ubuntu/

Installing Docker in Kali Linux

Unfortunately installing docker in every operating system is not as easy in every operating system. Docker is also a favorite tool among pen-testers as well Therefore I'm showing you steps to install docker in kali Linux as well.

Simply enter the following commands in the terminal listed below.

1. "sudo apt update"



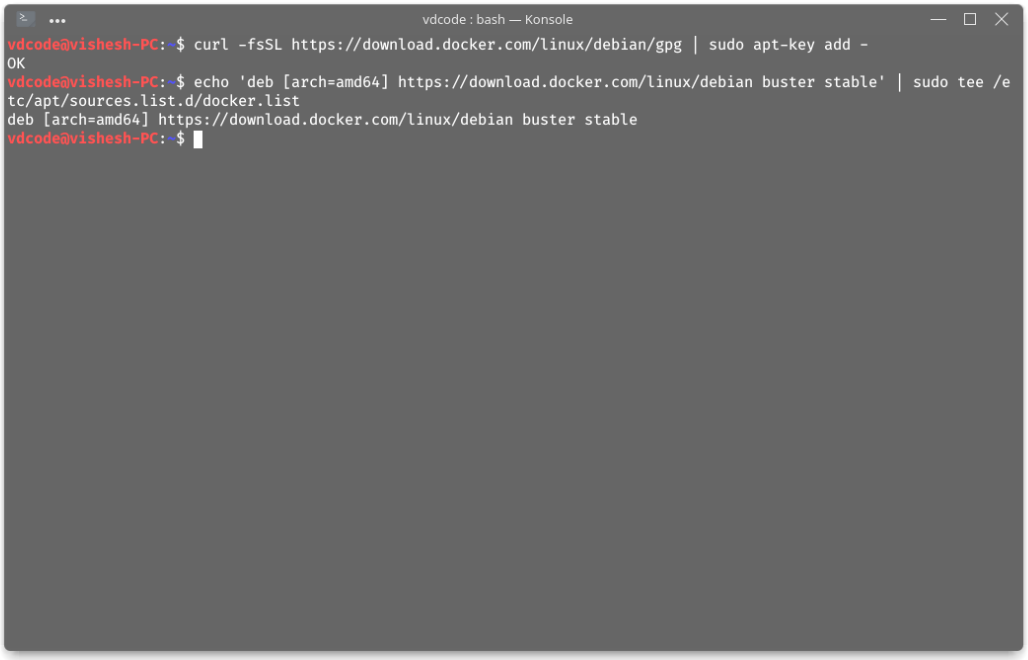
```
vdcode@vishesh-PC:~$ sudo apt update
[sudo] password for vdcode:
Ign:1 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 http://packages.microsoft.com/repos/vscode stable InRelease
Hit:3 http://deb.i2p2.no unstable InRelease
Hit:4 http://kali.cs.nctu.edu.tw/kali kali-rolling InRelease
Hit:5 http://dl.google.com/linux/chrome/deb stable Release
Hit:7 https://deb.nodesource.com/node_14.x jessie InRelease
Reading package lists... 0%
```

2. Then we need to add the official Docker PGP key like so:

   curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add –

3. Next, we configure APT so we will be able to download, install and update Docker.

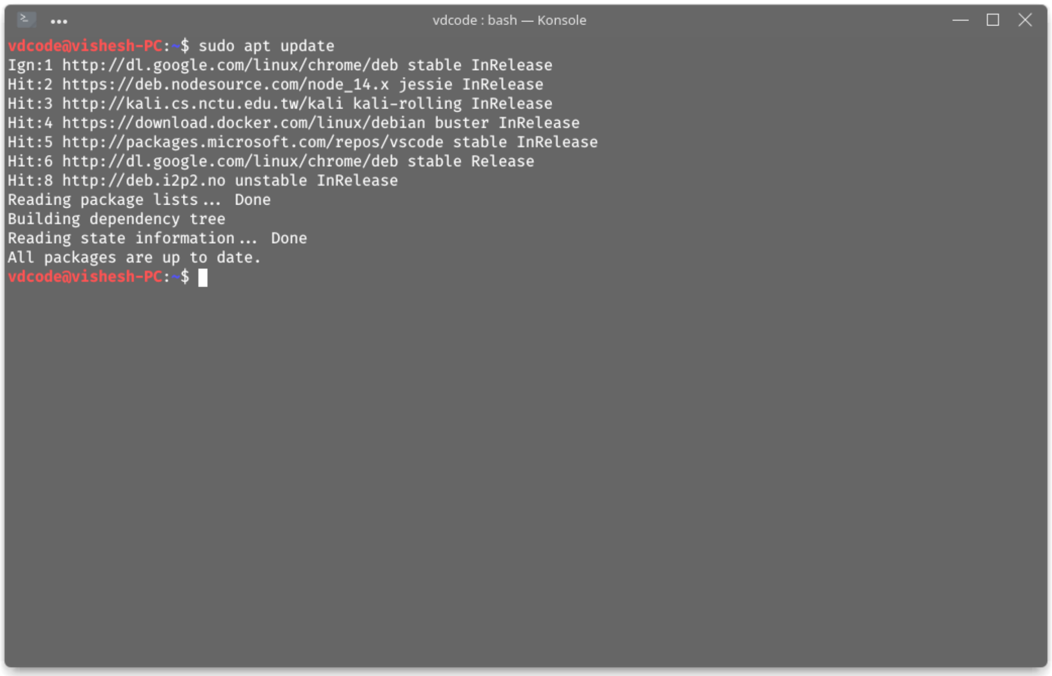
   "echo 'deb [arch=amd64] https://download.docker.com/linux/debianbuster stable' | sudo tee /etc/apt/sources.list.d/docker.list"



```
vdcode@vishesh-PC:~$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
OK
vdcode@vishesh-PC:~$ echo 'deb [arch=amd64] https://download.docker.com/linux/debian buster stable' | sudo tee /e
tc/apt/sources.list.d/docker.list
deb [arch=amd64] https://download.docker.com/linux/debian buster stable
vdcode@vishesh-PC:~$
```

4. Update your repositories again:

   "sudo apt update"



```
vdcode@vishesh-PC:~$ sudo apt update
Ign:1 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 https://deb.nodesource.com/node_14.x jessie InRelease
Hit:3 http://kali.cs.nctu.edu.tw/kali kali-rolling InRelease
Hit:4 https://download.docker.com/linux/debian buster InRelease
Hit:5 http://packages.microsoft.com/repos/vscode stable InRelease
Hit:6 http://dl.google.com/linux/chrome/deb stable Release
Hit:8 http://deb.i2p2.no unstable InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
vdcode@vishesh-PC:~$
```

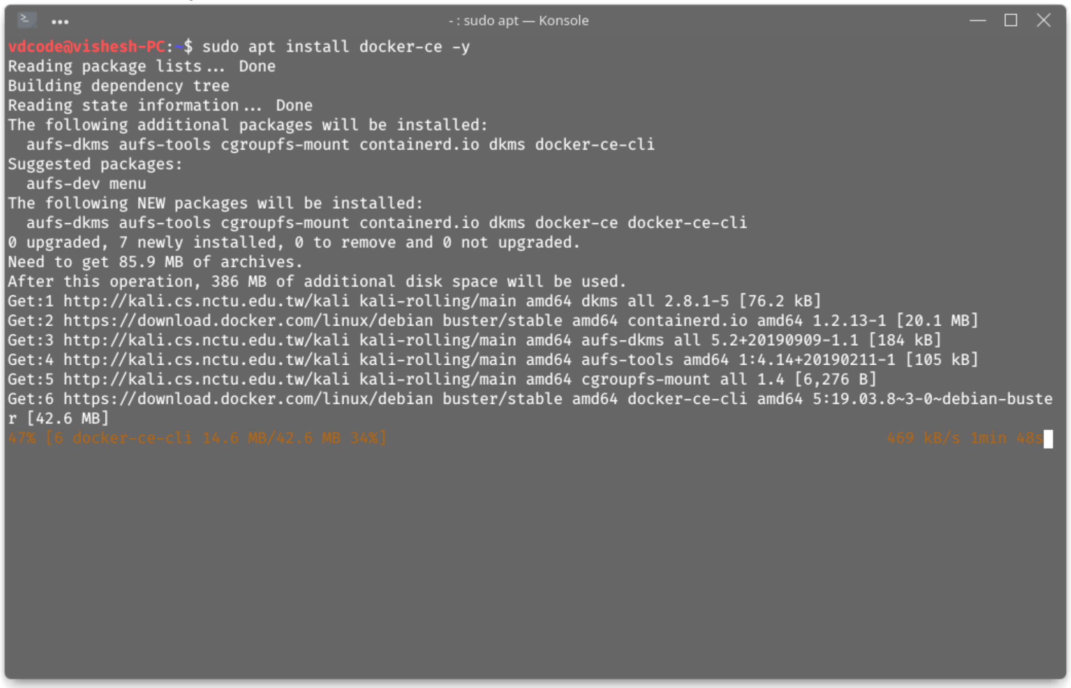
This time you can notice that in "Hit:4" message docker repo has been added which was not available in first update command.

5. Remove any older versions if installed:

   "sudo apt remove docker docker-engine docker.io"

6. Once this is done, we are ready to install Docker on Kali Linux:

   "sudo apt install docker-ce -y"



```
vdcode@vishesh-PC:~$ sudo apt install docker-ce -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  aufs-dkms aufs-tools cgroupfs-mount containerd.io dkms docker-ce-cli
Suggested packages:
  aufs-dev menu
The following NEW packages will be installed:
  aufs-dkms aufs-tools cgroupfs-mount containerd.io dkms docker-ce docker-ce-cli
0 upgraded, 7 newly installed, 0 to remove and 0 not upgraded.
Need to get 85.9 MB of archives.
After this operation, 386 MB of additional disk space will be used.
Get:1 http://kali.cs.nctu.edu.tw/kali kali-rolling/main amd64 dkms all 2.8.1-5 [76.2 kB]
Get:2 https://download.docker.com/linux/debian buster/stable amd64 containerd.io amd64 1.2.13-1 [20.1 MB]
Get:3 http://kali.cs.nctu.edu.tw/kali kali-rolling/main amd64 aufs-dkms all 5.2+20190909-1.1 [184 kB]
Get:4 http://kali.cs.nctu.edu.tw/kali kali-rolling/main amd64 aufs-tools amd64 1:4.14+20190211-1 [105 kB]
Get:5 http://kali.cs.nctu.edu.tw/kali kali-rolling/main amd64 cgroupfs-mount all 1.4 [6,276 B]
Get:6 https://download.docker.com/linux/debian buster/stable amd64 docker-ce-cli amd64 5:19.03.8~3-0~debian-buste
r [42.6 MB]
47% [6 docker-ce-cli 14.6 MB/42.6 MB 34%]                                              469 kB/s 1min 48s
```

It will take some time to complete. Be patient ;)

7. Finally starting docker.

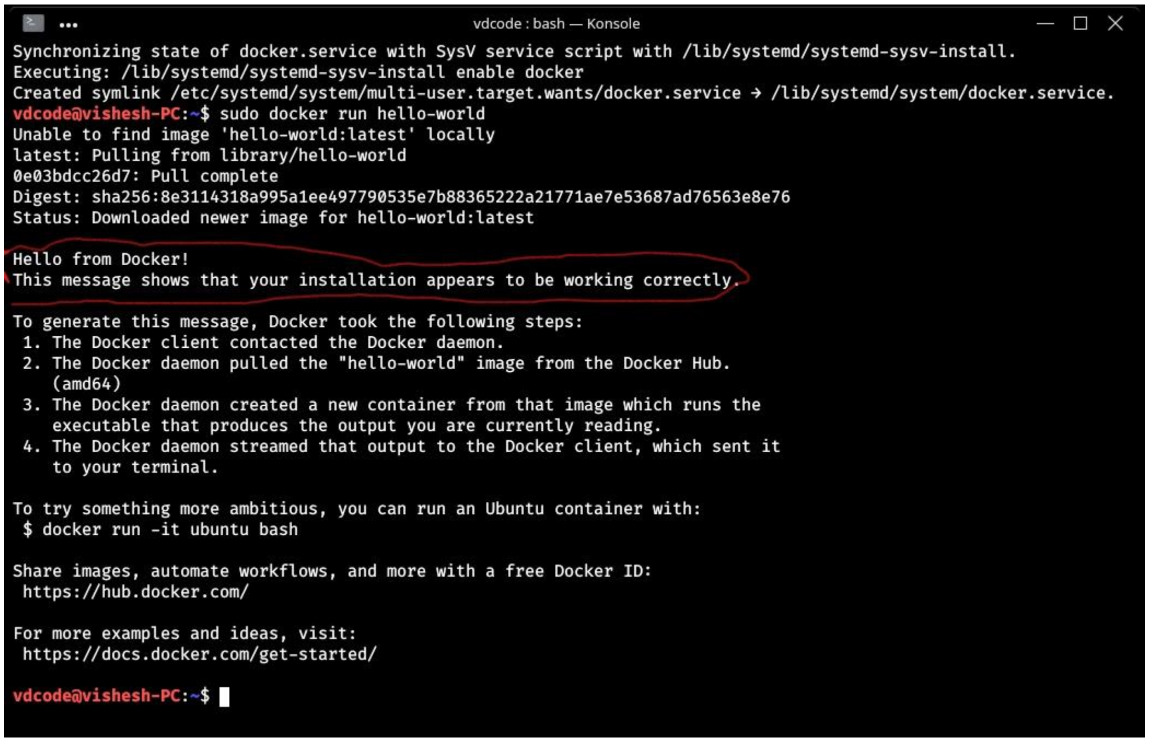
   "sudo systemctl start docker"

8. Run this command to start docker automatically on system restart

   "sudo systemctl enable docker"

9. Verifying the installation

   "sudo docker run hello-world"



```
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
vdcode@vishesh-PC:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:8e3114318a995a1ee497790535e7b88365222a21771ae7e53687ad76563e8e76
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/

vdcode@vishesh-PC:~$
```

Watching this message...Hurray docker is installed successfully :D

Conclusions

Overall Docker is an awesome tool which makes a developer's life easier and more productive. There are tools like Kubernetes which organizes our containers into groups of pods and can automatically scale those pods based upon the amount of traffic or utilization.

While the developers can sit back and have a cup of coffee.

I highly encourage you to use docker in your projects as well. 😊