# Week – 1
# Learning About Docker

Documented by-Darshit Jung KC

## Summary

In this project, Maven is employed as the build tool for managing and automating the build process of a Java application. The application's dependencies, configurations, and build lifecycle are specified in the project's Maven configuration file (pom.xml).

The application utilizes a MySQL database, and to streamline the deployment and management of the database, Docker is employed. The MySQL database is containerized, meaning it is encapsulated within a Docker container, allowing for consistent and reproducible deployments across different environments. Docker Compose is used to define and orchestrate the multi-container application stack, providing a simple way to manage the entire application environment, including both the Java application and the MySQL database.

Here's a breakdown of the key components and their roles in the project:

Maven:

Maven is utilized as the build automation tool to manage the project's build lifecycle, dependencies, and project structure.

The pom.xml file contains project configurations, dependencies, and build plugins.

Java Application:

The Java application source code is structured according to Maven conventions.

Maven is used to compile, test, and package the application into executable artifacts (e.g., JAR or WAR files).

MySQL Database:

The MySQL database is containerized using Docker.

Docker provides a lightweight, isolated environment for the MySQL database, ensuring consistent behavior across different environments.

Docker Compose:

Docker Compose is employed to define and manage the multi-container application stack.

The docker-compose.yml file specifies the services (containers) needed for the application, including the MySQL database and any other required services.

Deployment and Scaling:

The Docker Compose file simplifies the deployment process, allowing for easy scaling or replication of the application stack if needed.

Docker containers facilitate portability and consistency, enabling developers to run the application stack on different environments with minimal configuration.

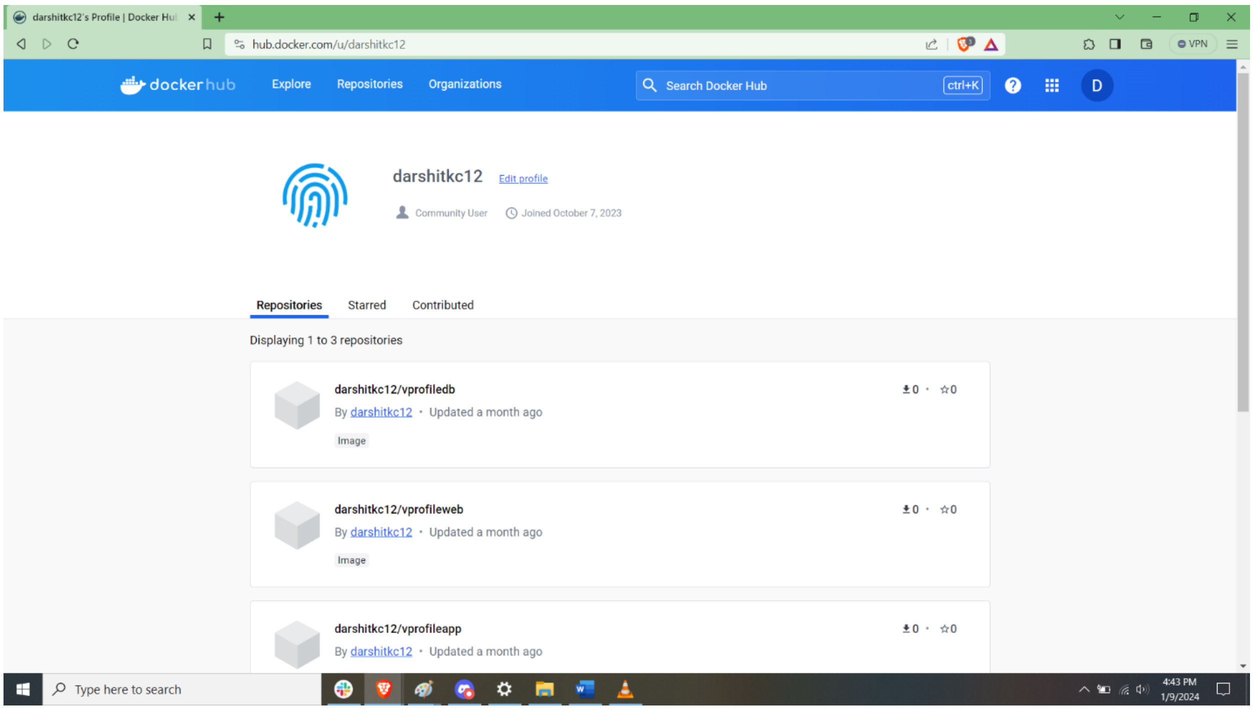
By combining Maven for build automation and Docker with Docker Compose for containerized deployment, the project achieves a streamlined and reproducible development, testing, and deployment process. Developers can easily manage dependencies, build the application, and deploy the entire stack, including the MySQL database, using standardized and version-controlled configurations

# Docker

Docker is a set of platforms as a service (PaaS) product that use OS-level virtualization to deliver software in packages called containers.

## Docker Setup

### Creating Profile & Setting up repository



darshitkc12 Edit profile

Community User · Joined October 7, 2023

Repositories · Starred · Contributed

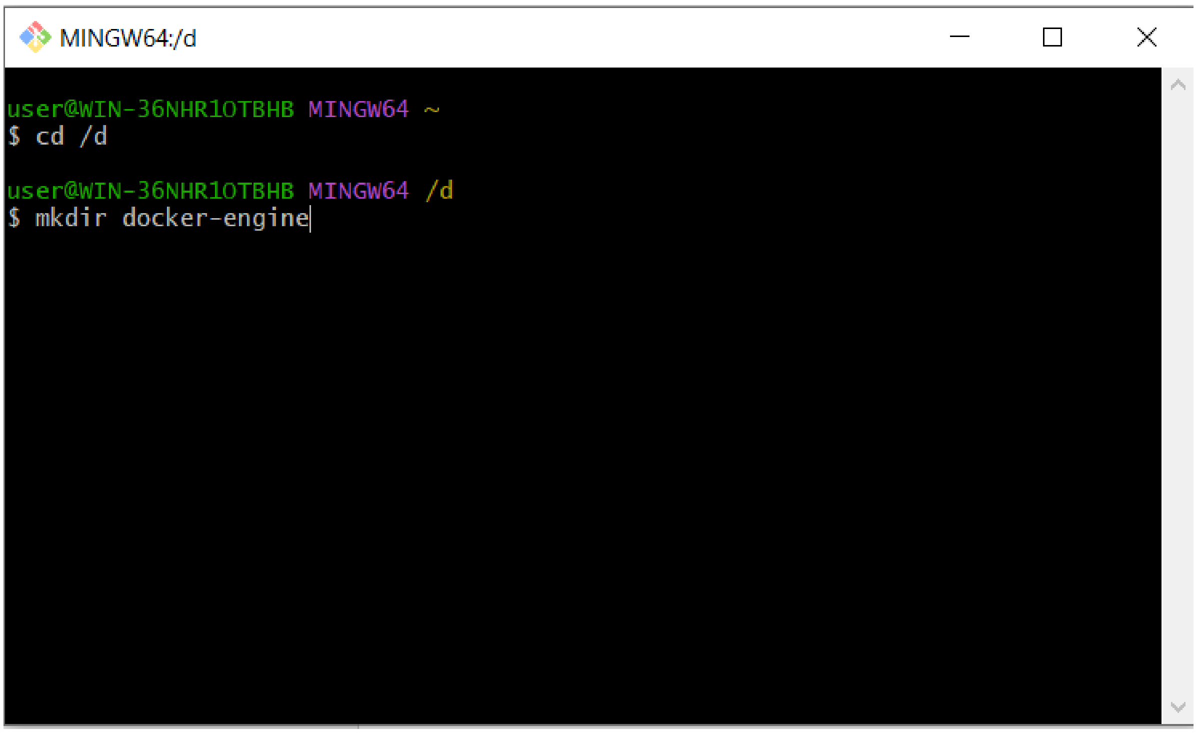
Displaying 1 to 3 repositories

darshitkc12/vprofiledb
By darshitkc12 · Updated a month ago
Image
0 · 0

darshitkc12/vprofileweb
By darshitkc12 · Updated a month ago
Image
0 · 0

darshitkc12/vprofileapp
By darshitkc12 · Updated a month ago
0 · 0

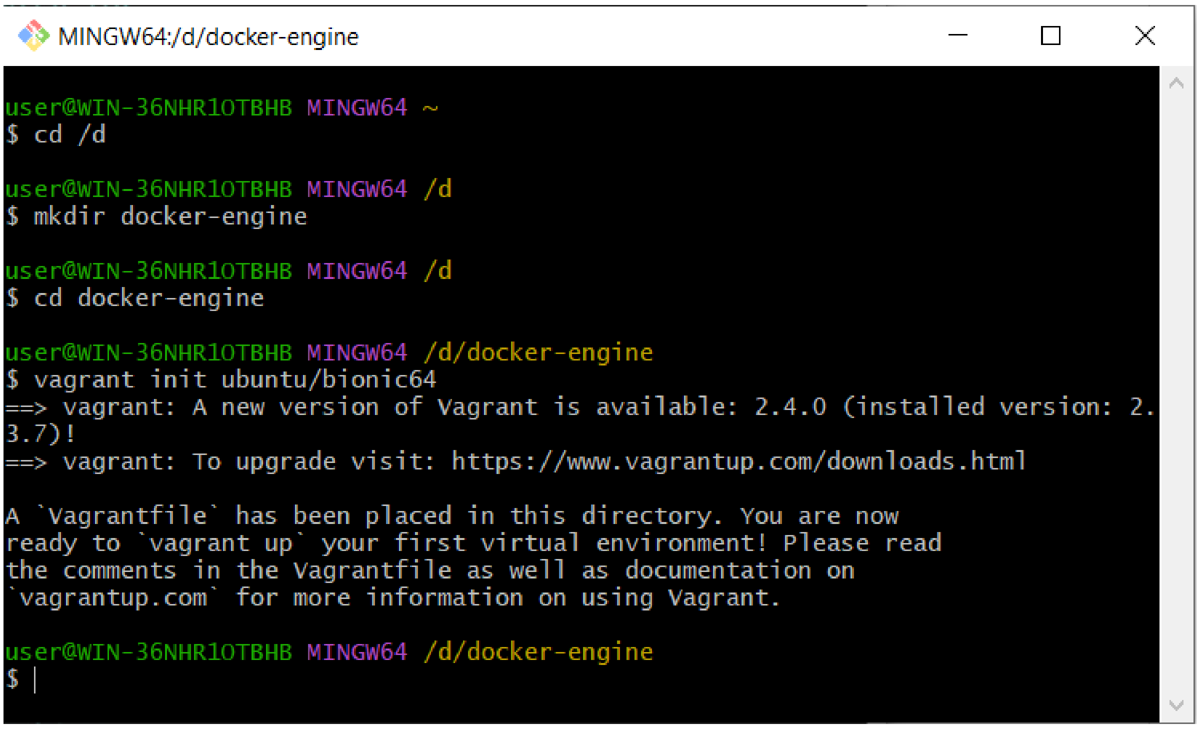
Creating directory to work on project



MINGW64:/d

```
user@WIN-36NHR1OTBHB MINGW64 ~
$ cd /d

user@WIN-36NHR1OTBHB MINGW64 /d
$ mkdir docker-engine
```

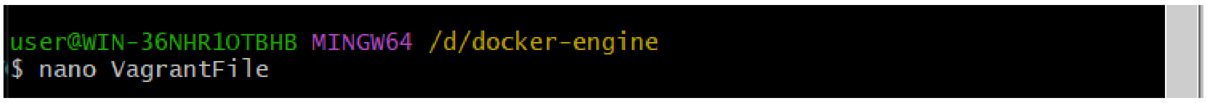
Initializing vagrant file for running VM



MINGW64:/d/docker-engine

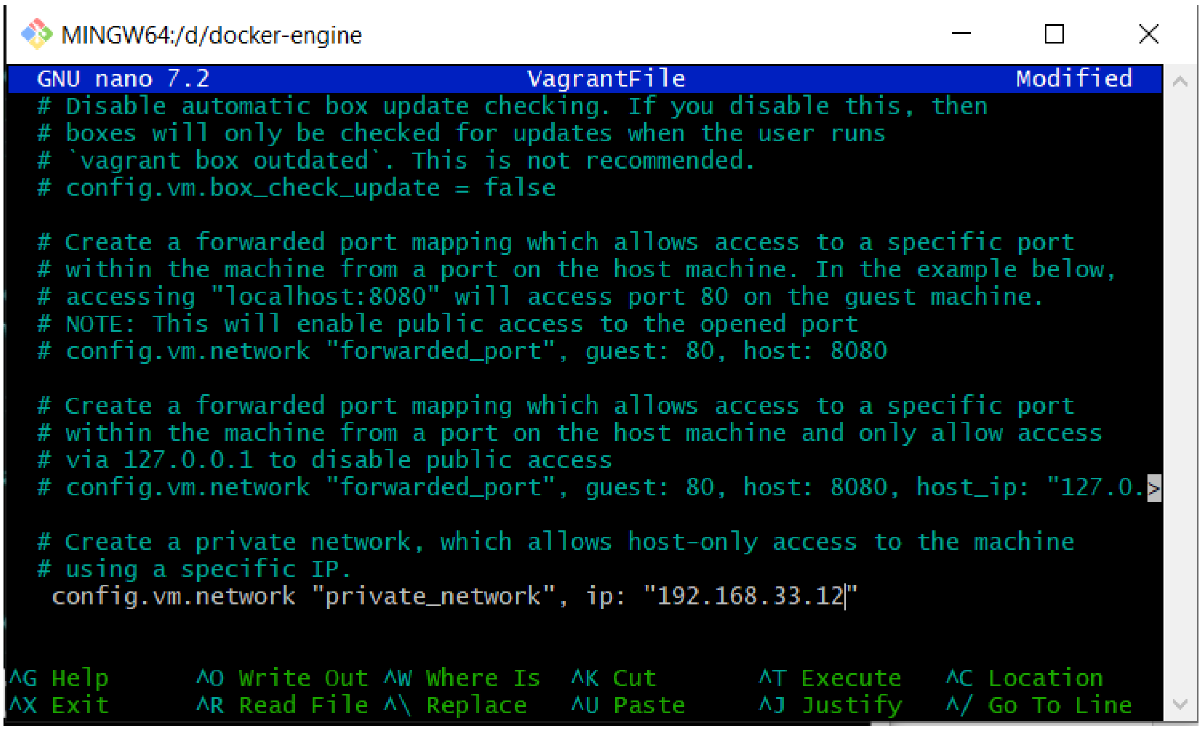
```
user@WIN-36NHR1OTBHB MINGW64 ~
$ cd /d

user@WIN-36NHR1OTBHB MINGW64 /d
$ mkdir docker-engine

user@WIN-36NHR1OTBHB MINGW64 /d
$ cd docker-engine

user@WIN-36NHR1OTBHB MINGW64 /d/docker-engine
$ vagrant init ubuntu/bionic64
==> vagrant: A new version of Vagrant is available: 2.4.0 (installed version: 2.
3.7)!
==> vagrant: To upgrade visit: https://www.vagrantup.com/downloads.html

A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.

user@WIN-36NHR1OTBHB MINGW64 /d/docker-engine
$ 
```

Edit VM configuration file



```
user@WIN-36NHR1OTBHB MINGW64 /d/docker-engine
$ nano VagrantFile
```

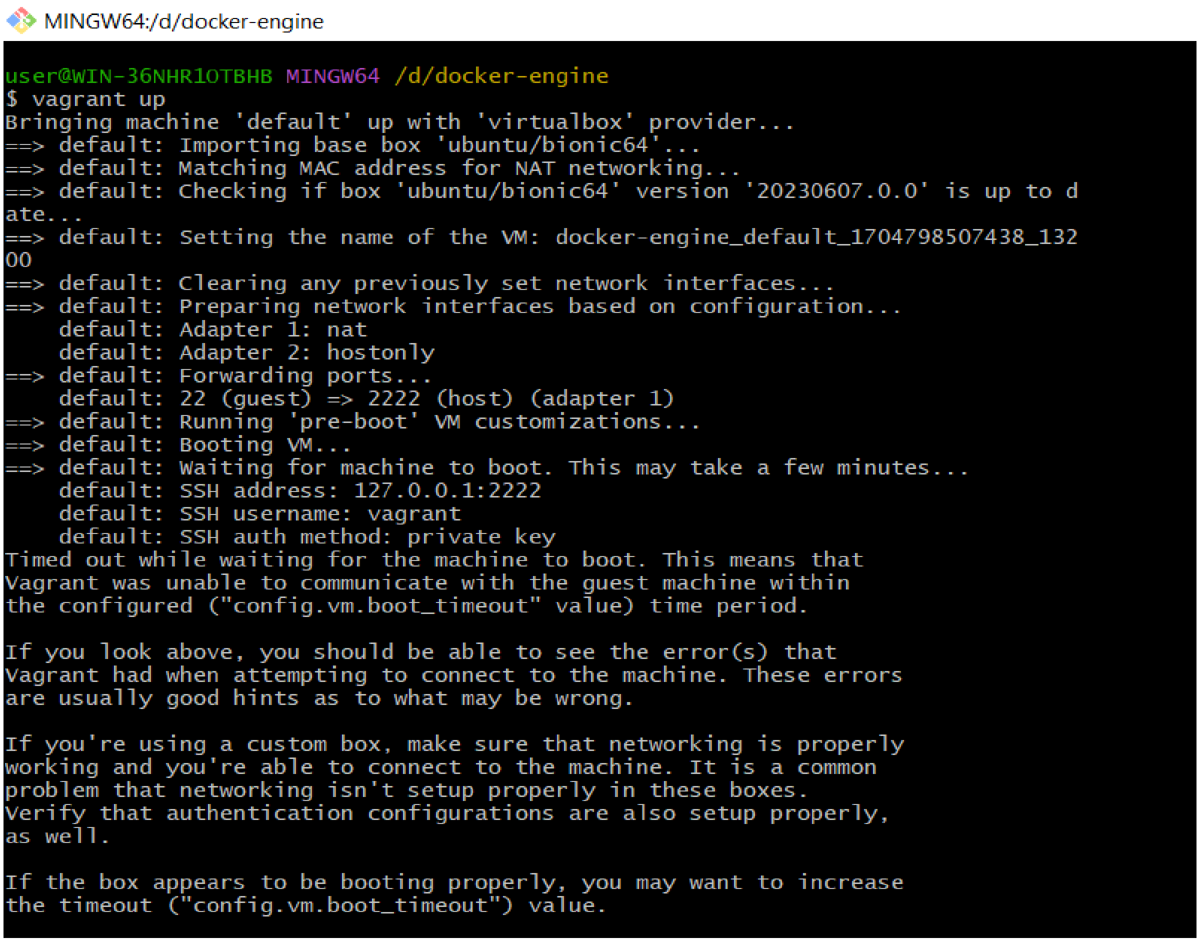
Allowing VM to be access through web



MINGW64:/d/docker-engine

```
 GNU nano 7.2                    VagrantFile                       Modified
 # Disable automatic box update checking. If you disable this, then
 # boxes will only be checked for updates when the user runs
 # `vagrant box outdated`. This is not recommended.
 # config.vm.box_check_update = false

 # Create a forwarded port mapping which allows access to a specific port
 # within the machine from a port on the host machine. In the example below,
 # accessing "localhost:8080" will access port 80 on the guest machine.
 # NOTE: This will enable public access to the opened port
 # config.vm.network "forwarded_port", guest: 80, host: 8080

 # Create a forwarded port mapping which allows access to a specific port
 # within the machine from a port on the host machine and only allow access
 # via 127.0.0.1 to disable public access
 # config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.>

 # Create a private network, which allows host-only access to the machine
 # using a specific IP.
  config.vm.network "private_network", ip: "192.168.33.12"


^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

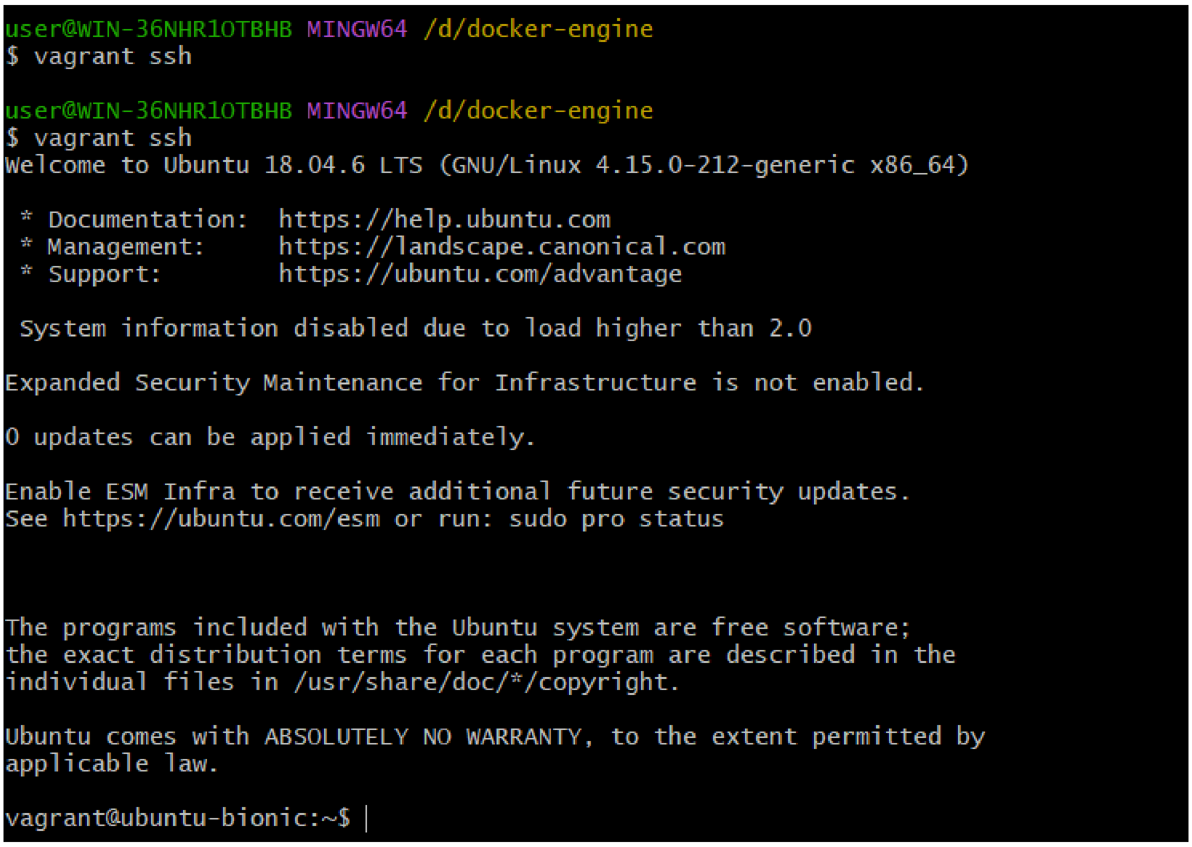
Running VM



MINGW64:/d/docker-engine

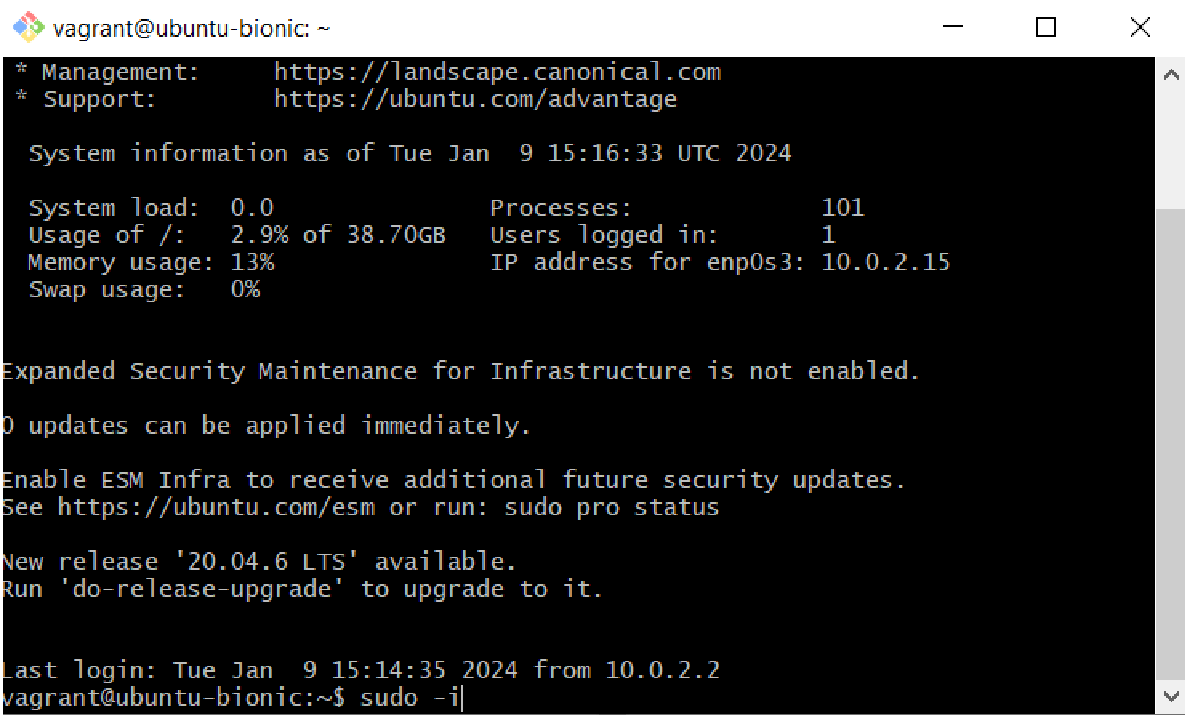
```
user@WIN-36NHR1OTBHB MINGW64 /d/docker-engine
$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'ubuntu/bionic64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'ubuntu/bionic64' version '20230607.0.0' is up to d
ate...
==> default: Setting the name of the VM: docker-engine_default_1704798507438_132
00
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
Timed out while waiting for the machine to boot. This means that
Vagrant was unable to communicate with the guest machine within
the configured ("config.vm.boot_timeout" value) time period.

If you look above, you should be able to see the error(s) that
Vagrant had when attempting to connect to the machine. These errors
are usually good hints as to what may be wrong.

If you're using a custom box, make sure that networking is properly
working and you're able to connect to the machine. It is a common
problem that networking isn't setup properly in these boxes.
Verify that authentication configurations are also setup properly,
as well.

If the box appears to be booting properly, you may want to increase
the timeout ("config.vm.boot_timeout") value.
```

Accessing VM through shell



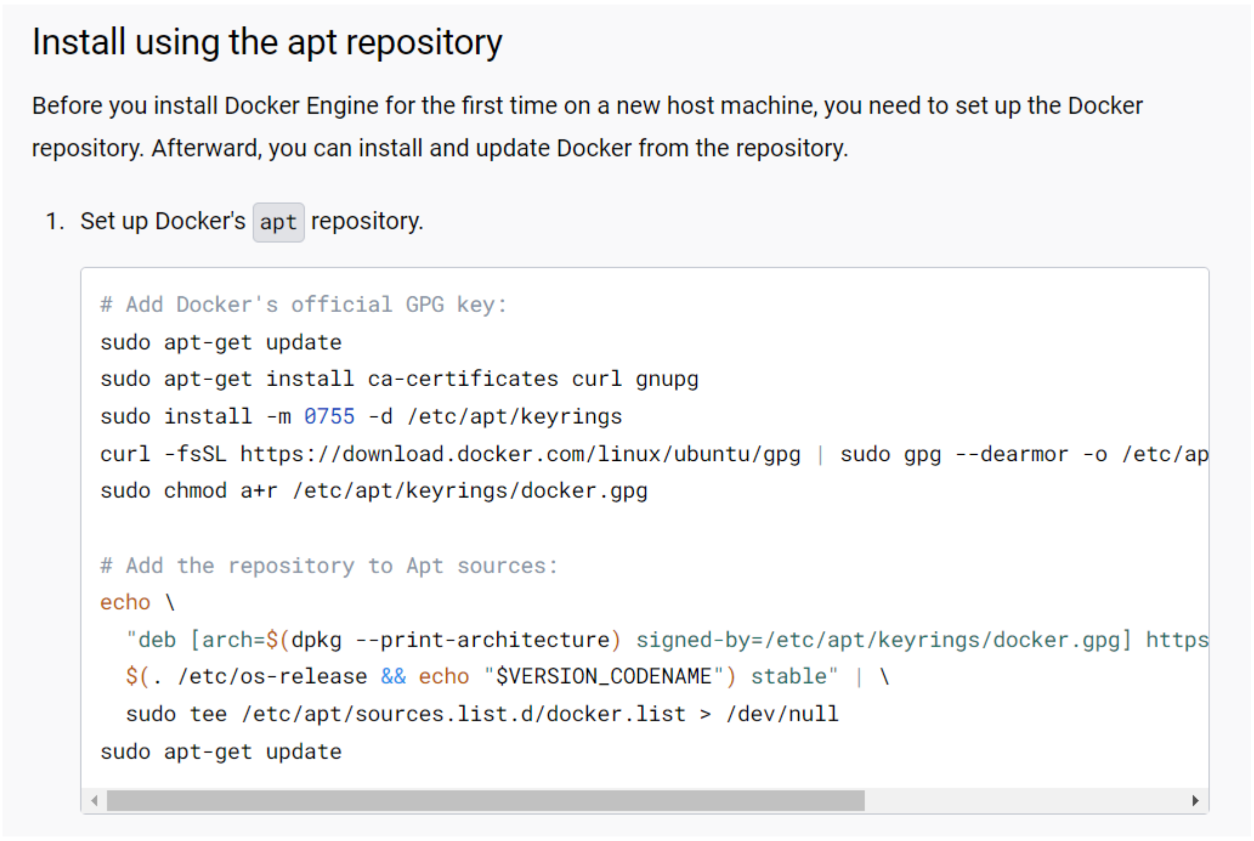
```
user@WIN-36NHR1OTBHB MINGW64 /d/docker-engine
$ vagrant ssh

user@WIN-36NHR1OTBHB MINGW64 /d/docker-engine
$ vagrant ssh
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-212-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information disabled due to load higher than 2.0

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

Enable ESM Infra to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

vagrant@ubuntu-bionic:~$ |
```

Giving super user do permission



```
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Jan  9 15:16:33 UTC 2024

  System load:  0.0               Processes:             101
  Usage of /:   2.9% of 38.70GB   Users logged in:       1
  Memory usage: 13%               IP address for enp0s3: 10.0.2.15
  Swap usage:   0%


Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

Enable ESM Infra to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Tue Jan  9 15:14:35 2024 from 10.0.2.2
vagrant@ubuntu-bionic:~$ sudo -i
```

Installing Docker using git repository

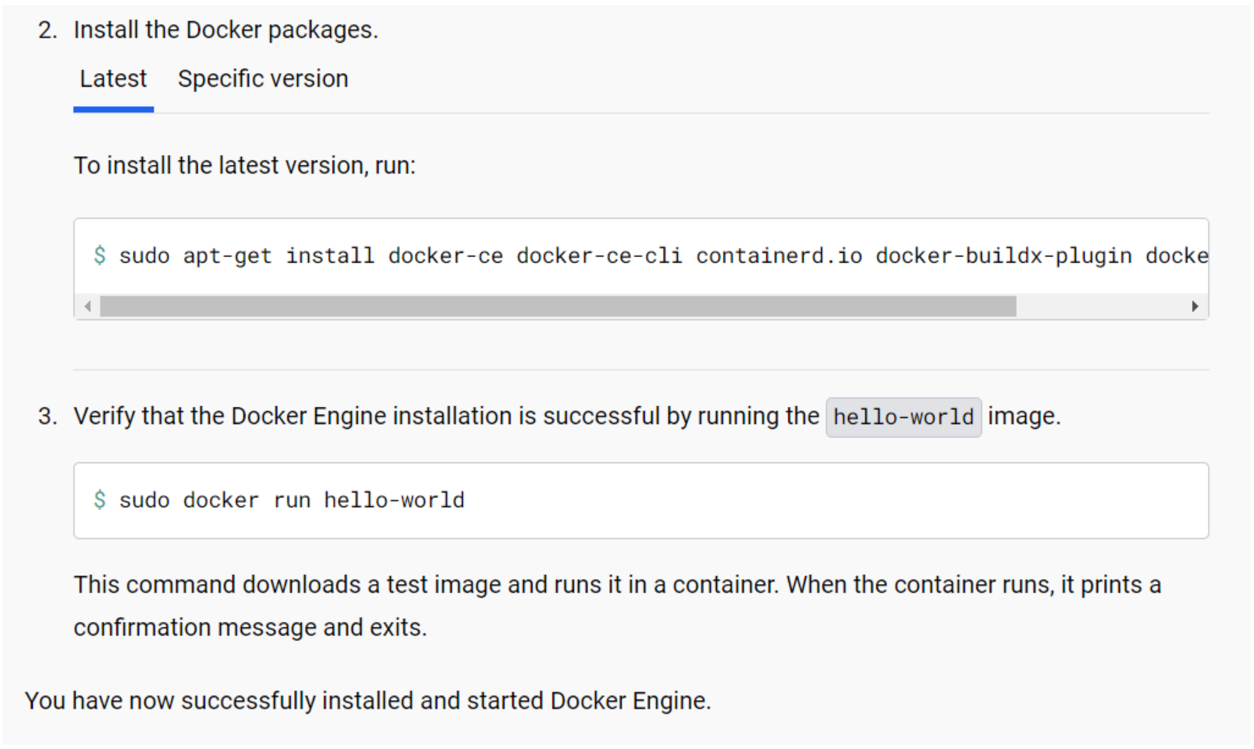


## Install using the apt repository

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

1. Set up Docker's `apt` repository.

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/ap
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```



2. Install the Docker packages.

To install the latest version, run:

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docke
```

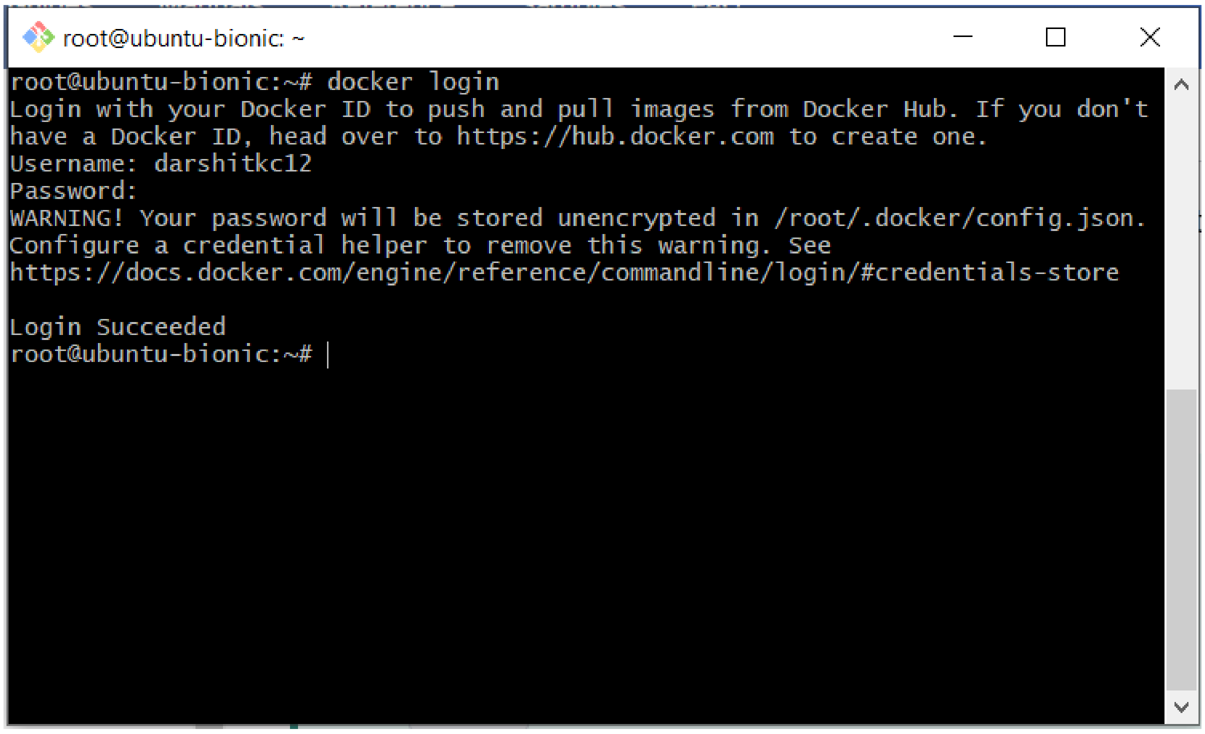
3. Verify that the Docker Engine installation is successful by running the `hello-world` image.

```
$ sudo docker run hello-world
```

This command downloads a test image and runs it in a container. When the container runs, it prints a confirmation message and exits.

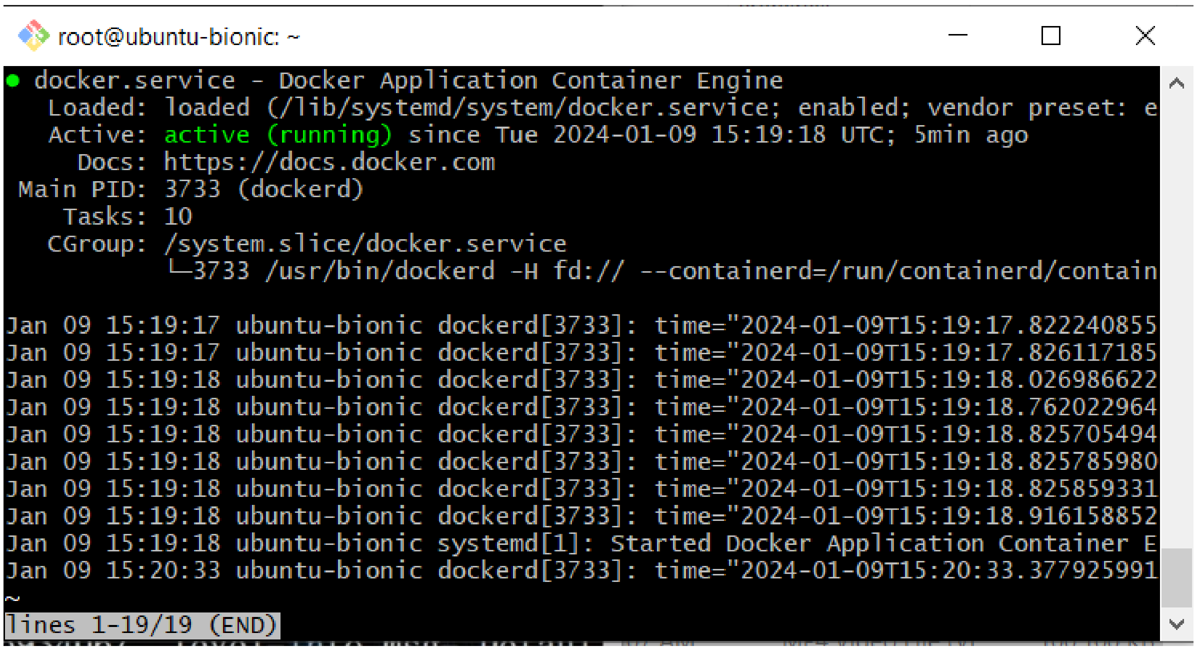
You have now successfully installed and started Docker Engine.

Logging in in docker through shell



```
root@ubuntu-bionic:~# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't
have a Docker ID, head over to https://hub.docker.com to create one.
Username: darshitkc12
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ubuntu-bionic:~# |
```

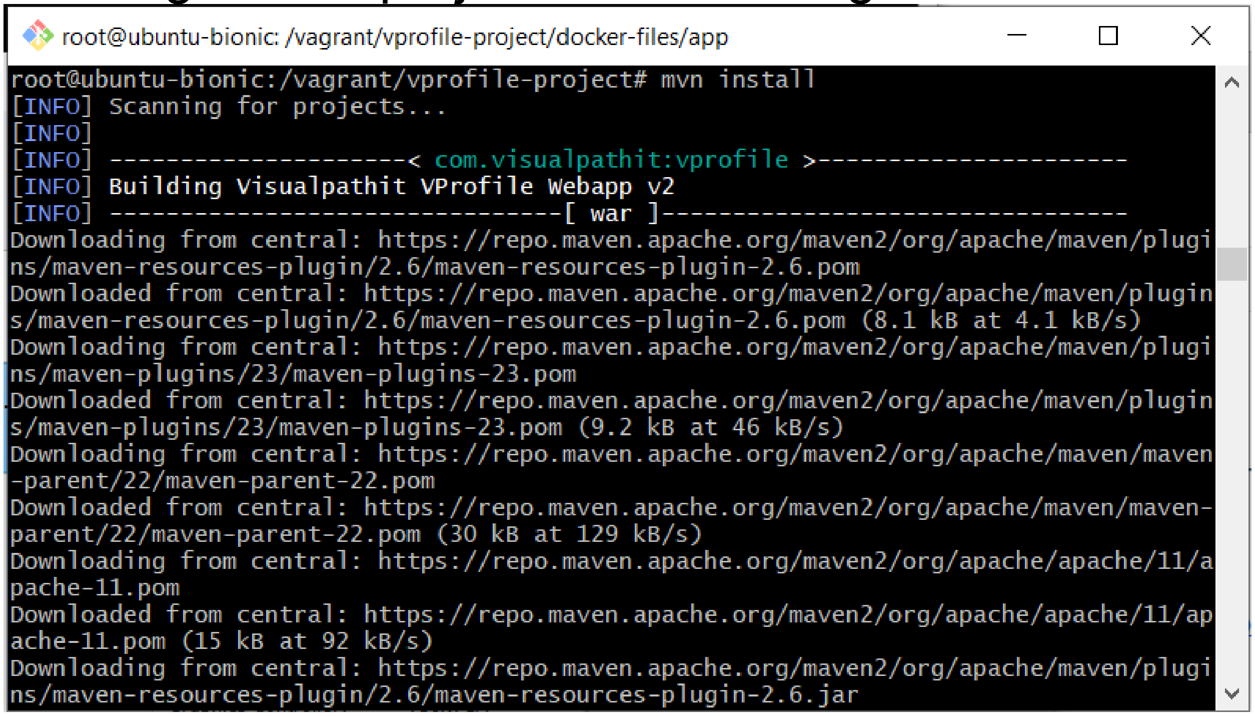
## Checking docker status



root@ubuntu-bionic: ~

```
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: e
   Active: active (running) since Tue 2024-01-09 15:19:18 UTC; 5min ago
     Docs: https://docs.docker.com
 Main PID: 3733 (dockerd)
    Tasks: 10
   CGroup: /system.slice/docker.service
           └─3733 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/contain

Jan 09 15:19:17 ubuntu-bionic dockerd[3733]: time="2024-01-09T15:19:17.822240855
Jan 09 15:19:17 ubuntu-bionic dockerd[3733]: time="2024-01-09T15:19:17.826117185
Jan 09 15:19:18 ubuntu-bionic dockerd[3733]: time="2024-01-09T15:19:18.026986622
Jan 09 15:19:18 ubuntu-bionic dockerd[3733]: time="2024-01-09T15:19:18.762022964
Jan 09 15:19:18 ubuntu-bionic dockerd[3733]: time="2024-01-09T15:19:18.825705494
Jan 09 15:19:18 ubuntu-bionic dockerd[3733]: time="2024-01-09T15:19:18.825785980
Jan 09 15:19:18 ubuntu-bionic dockerd[3733]: time="2024-01-09T15:19:18.825859331
Jan 09 15:19:18 ubuntu-bionic dockerd[3733]: time="2024-01-09T15:19:18.916158852
Jan 09 15:19:18 ubuntu-bionic systemd[1]: Started Docker Application Container E
Jan 09 15:20:33 ubuntu-bionic dockerd[3733]: time="2024-01-09T15:20:33.377925991
~
lines 1-19/19 (END)
```

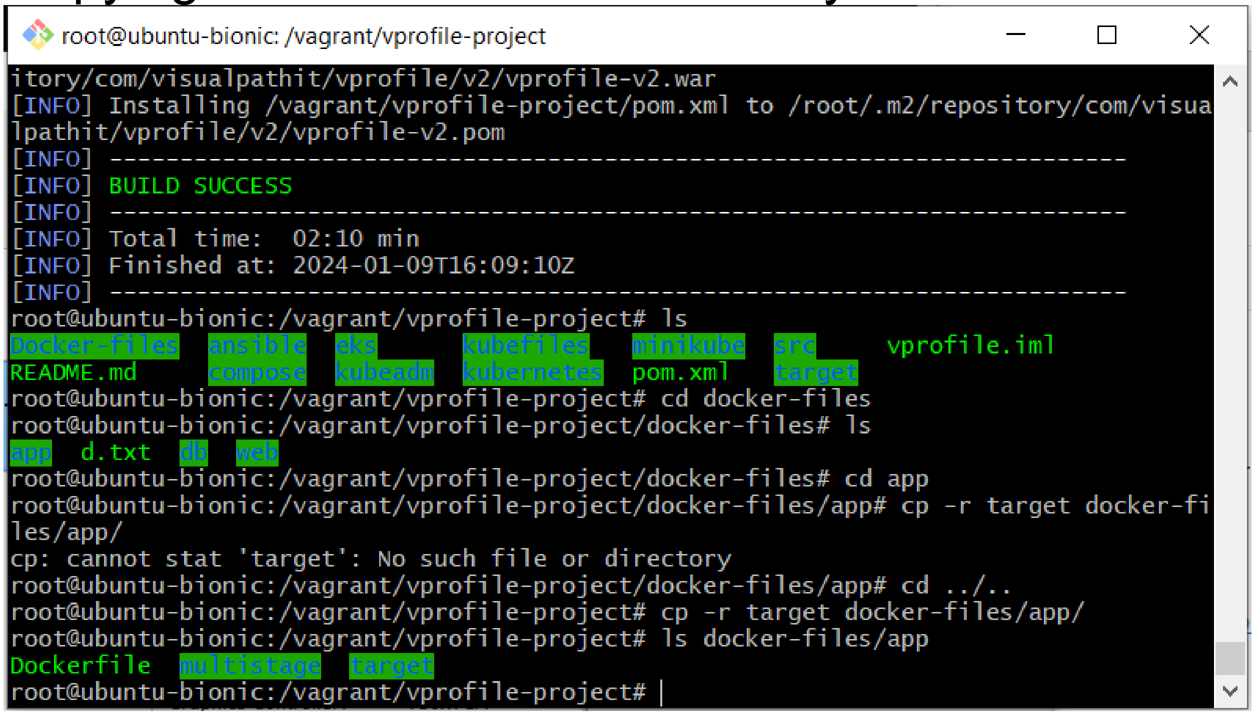
## Building project through maven



root@ubuntu-bionic: /vagrant/vprofile-project/docker-files/app

```
root@ubuntu-bionic:/vagrant/vprofile-project# mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.visualpathit:vprofile >----------------------
[INFO] Building Visualpathit VProfile Webapp v2
[INFO] --------------------------------[ war ]---------------------------------
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugi
ns/maven-resources-plugin/2.6/maven-resources-plugin-2.6.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugin
s/maven-resources-plugin/2.6/maven-resources-plugin-2.6.pom (8.1 kB at 4.1 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugi
ns/maven-plugins/23/maven-plugins-23.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugin
s/maven-plugins/23/maven-plugins-23.pom (9.2 kB at 46 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven
-parent/22/maven-parent-22.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-
parent/22/maven-parent-22.pom (30 kB at 129 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/apache/11/a
pache-11.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/11/ap
ache-11.pom (15 kB at 92 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugi
ns/maven-resources-plugin/2.6/maven-resources-plugin-2.6.jar
```

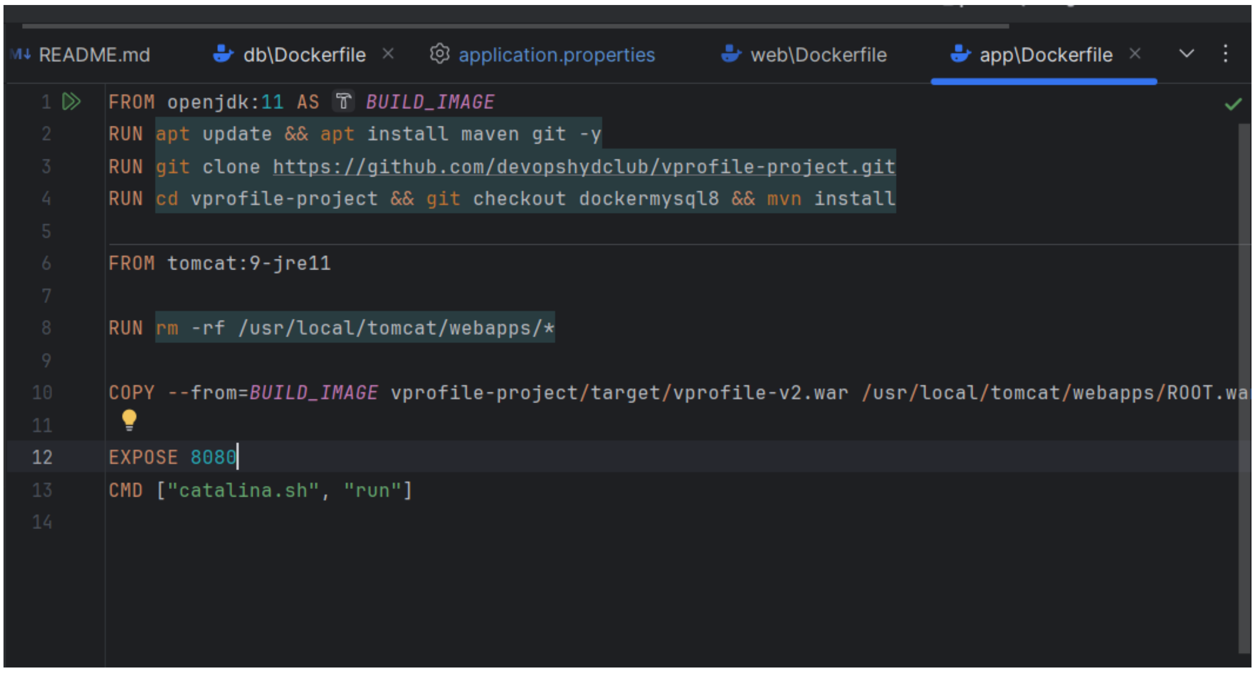
## Copying artifact to another directory



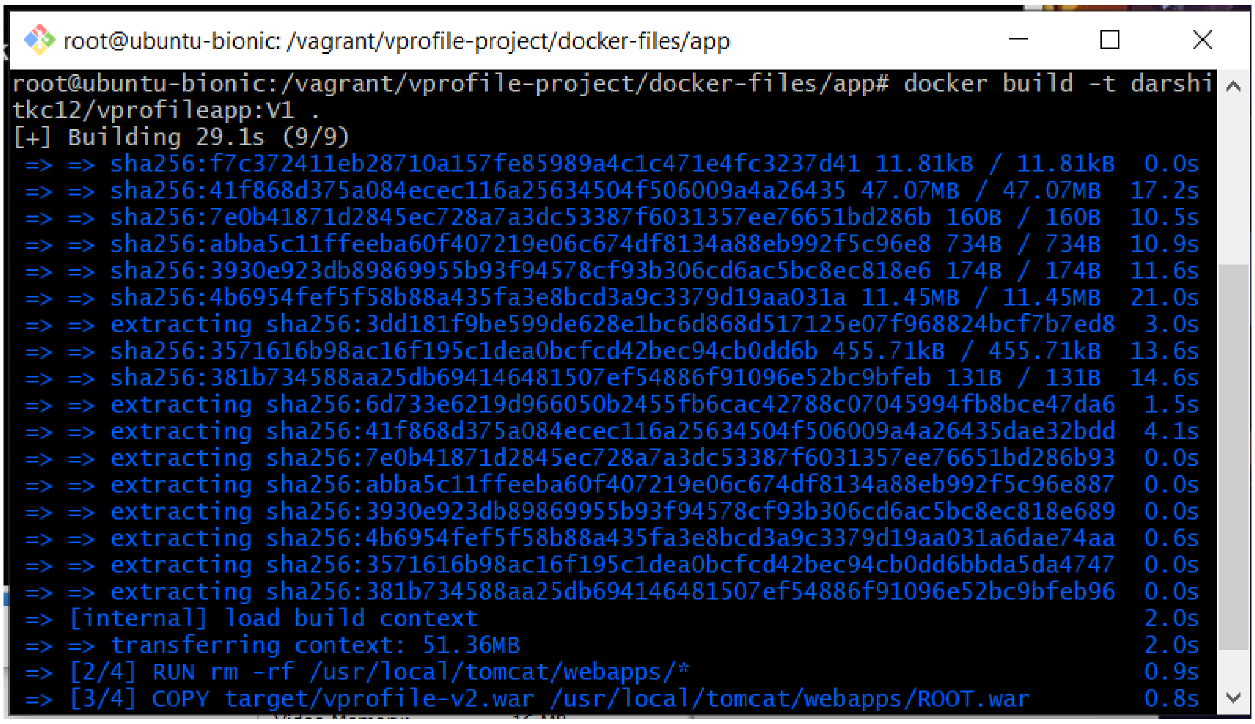
root@ubuntu-bionic: /vagrant/vprofile-project

```
itory/com/visualpathit/vprofile/v2/vprofile-v2.war
[INFO] Installing /vagrant/vprofile-project/pom.xml to /root/.m2/repository/com/visua
lpathit/vprofile/v2/vprofile-v2.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  02:10 min
[INFO] Finished at: 2024-01-09T16:09:10Z
[INFO] ------------------------------------------------------------------------
root@ubuntu-bionic:/vagrant/vprofile-project# ls
Docker-files  ansible  eks      kubefiles    minikube  src     vprofile.iml
README.md     compose  kubeadm  kubernetes   pom.xml   target
root@ubuntu-bionic:/vagrant/vprofile-project# cd docker-files
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files# ls
app  d.txt  db  web
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files# cd app
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/app# cp -r target docker-fi
les/app/
cp: cannot stat 'target': No such file or directory
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/app# cd ../..
root@ubuntu-bionic:/vagrant/vprofile-project# cp -r target docker-files/app/
root@ubuntu-bionic:/vagrant/vprofile-project# ls docker-files/app
Dockerfile  multistage  target
root@ubuntu-bionic:/vagrant/vprofile-project# |
```

## Application is being built as docker container

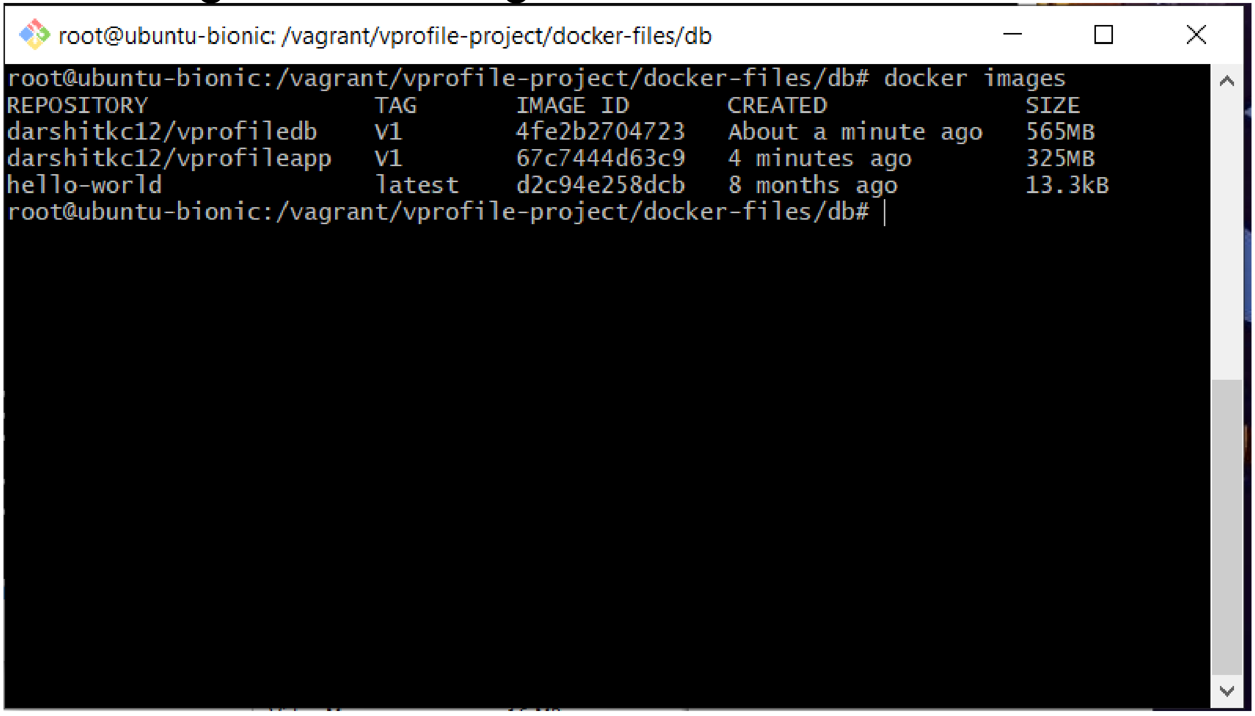


```
FROM openjdk:11 AS BUILD_IMAGE
RUN apt update && apt install maven git -y
RUN git clone https://github.com/devopshydclub/vprofile-project.git
RUN cd vprofile-project && git checkout dockermysql8 && mvn install

FROM tomcat:9-jre11

RUN rm -rf /usr/local/tomcat/webapps/*

COPY --from=BUILD_IMAGE vprofile-project/target/vprofile-v2.war /usr/local/tomcat/webapps/ROOT.wa

EXPOSE 8080
CMD ["catalina.sh", "run"]

```


root@ubuntu-bionic: /vagrant/vprofile-project/docker-files/app

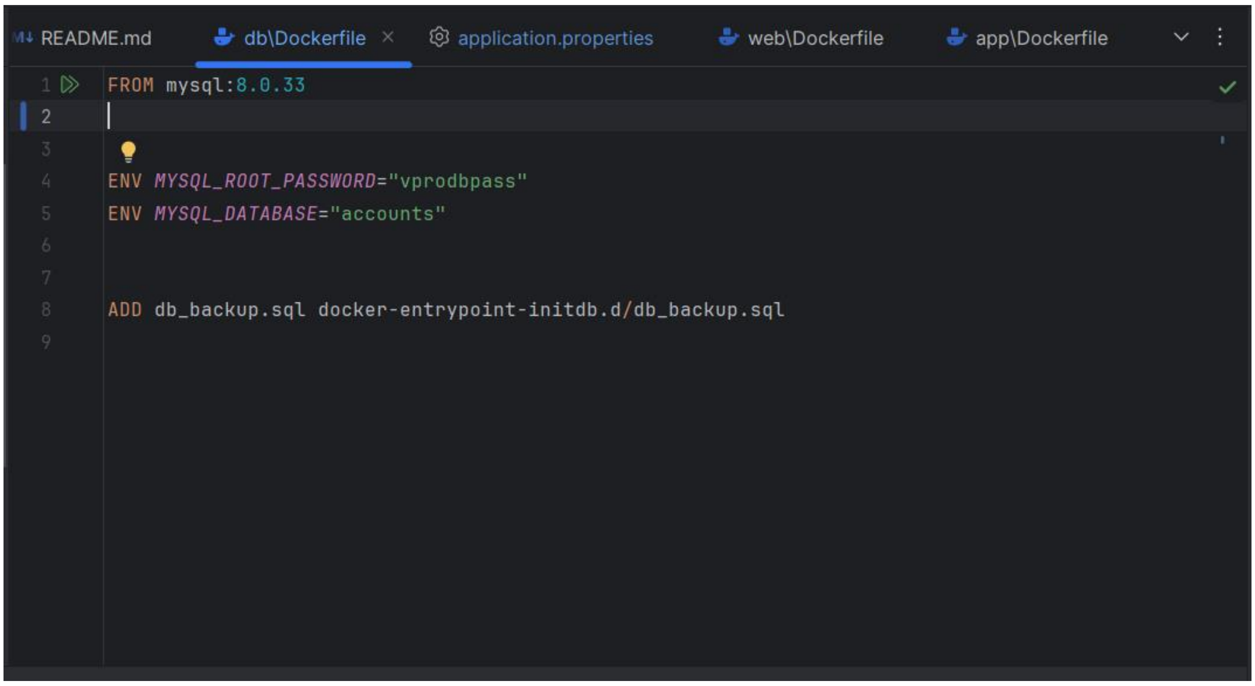
```
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/app# docker build -t darshi
tkc12/vprofileapp:V1 .
[+] Building 29.1s (9/9)
 => => sha256:f7c372411eb28710a157fe85989a4c1c471e4fc3237d41 11.81kB / 11.81kB  0.0s
 => => sha256:41f868d375a084ecec116a25634504f506009a4a26435 47.07MB / 47.07MB  17.2s
 => => sha256:7e0b41871d2845ec728a7a3dc53387f6031357ee76651bd286b 160B / 160B  10.5s
 => => sha256:abba5c11ffeeba60f407219e06c674df8134a88eb992f5c96e8 734B / 734B  10.9s
 => => sha256:3930e923db89869955b93f94578cf93b306cd6ac5bc8ec818e6 174B / 174B  11.6s
 => => sha256:4b6954fef5f58b88a435fa3e8bcd3a9c3379d19aa031a 11.45MB / 11.45MB  21.0s
 => => extracting sha256:3dd181f9be599de628e1bc6d868d517125e07f968824bcf7b7ed8  3.0s
 => => sha256:3571616b98ac16f195c1dea0bcfcd42bec94cb0dd6b 455.71kB / 455.71kB  13.6s
 => => sha256:381b734588aa25db694146481507ef54886f91096e52bc9bfeb 131B / 131B  14.6s
 => => extracting sha256:6d733e6219d966050b2455fb6cac42788c07045994fb8bce47da6  1.5s
 => => extracting sha256:41f868d375a084ecec116a25634504f506009a4a26435dae32bdd  4.1s
 => => extracting sha256:7e0b41871d2845ec728a7a3dc53387f6031357ee76651bd286b93  0.0s
 => => extracting sha256:abba5c11ffeeba60f407219e06c674df8134a88eb992f5c96e887  0.0s
 => => extracting sha256:3930e923db89869955b93f94578cf93b306cd6ac5bc8ec818e689  0.0s
 => => extracting sha256:4b6954fef5f58b88a435fa3e8bcd3a9c3379d19aa031a6dae74aa  0.6s
 => => extracting sha256:3571616b98ac16f195c1dea0bcfcd42bec94cb0dd6bbda5da4747  0.0s
 => => extracting sha256:381b734588aa25db694146481507ef54886f91096e52bc9bfeb96  0.0s
 => [internal] load build context                                              2.0s
 => => transferring context: 51.36MB                                           2.0s
 => [2/4] RUN rm -rf /usr/local/tomcat/webapps/*                               0.9s
 => [3/4] COPY target/vprofile-v2.war /usr/local/tomcat/webapps/ROOT.war       0.8s
```

## Checking docker images

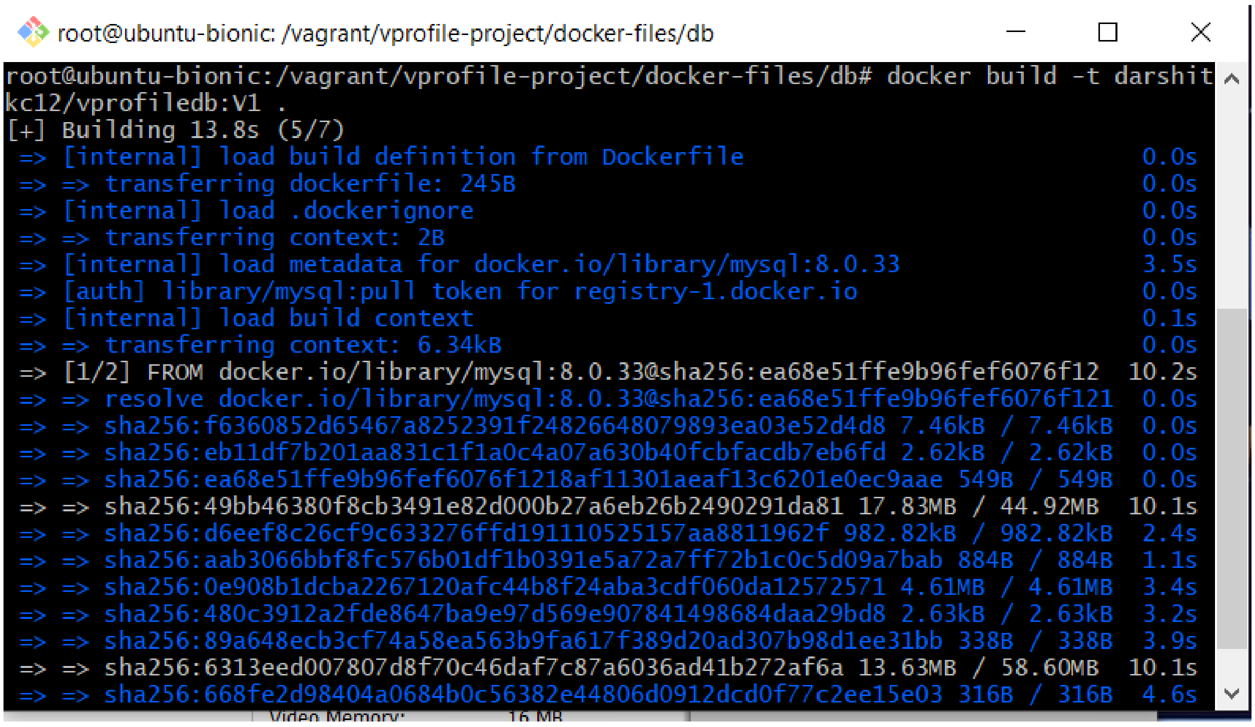

root@ubuntu-bionic: /vagrant/vprofile-project/docker-files/db

```
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/db# docker images
REPOSITORY                TAG       IMAGE ID       CREATED              SIZE
darshitkc12/vprofiledb    V1        4fe2b2704723   About a minute ago   565MB
darshitkc12/vprofileapp   V1        67c7444d63c9   4 minutes ago        325MB
hello-world               latest    d2c94e258dcb   8 months ago         13.3kB
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/db# |
```

## Building database as docker container

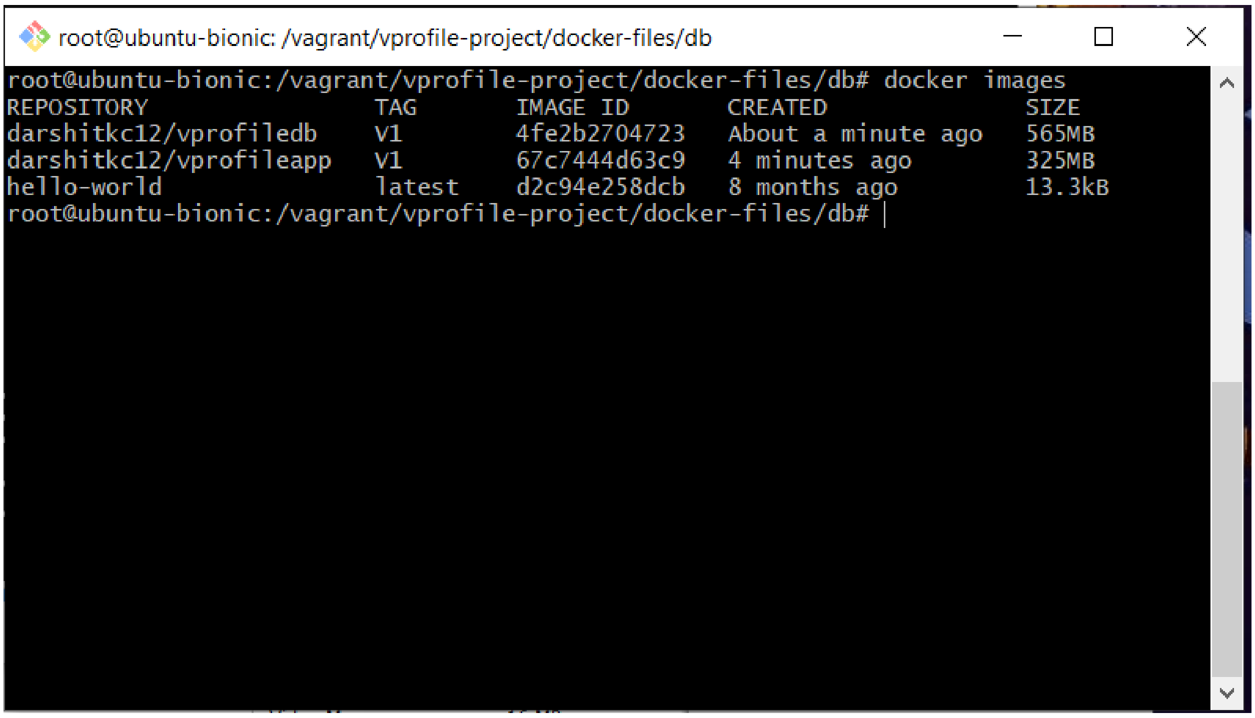


```
FROM mysql:8.0.33


ENV MYSQL_ROOT_PASSWORD="vprodbpass"
ENV MYSQL_DATABASE="accounts"


ADD db_backup.sql docker-entrypoint-initdb.d/db_backup.sql
```



root@ubuntu-bionic: /vagrant/vprofile-project/docker-files/db

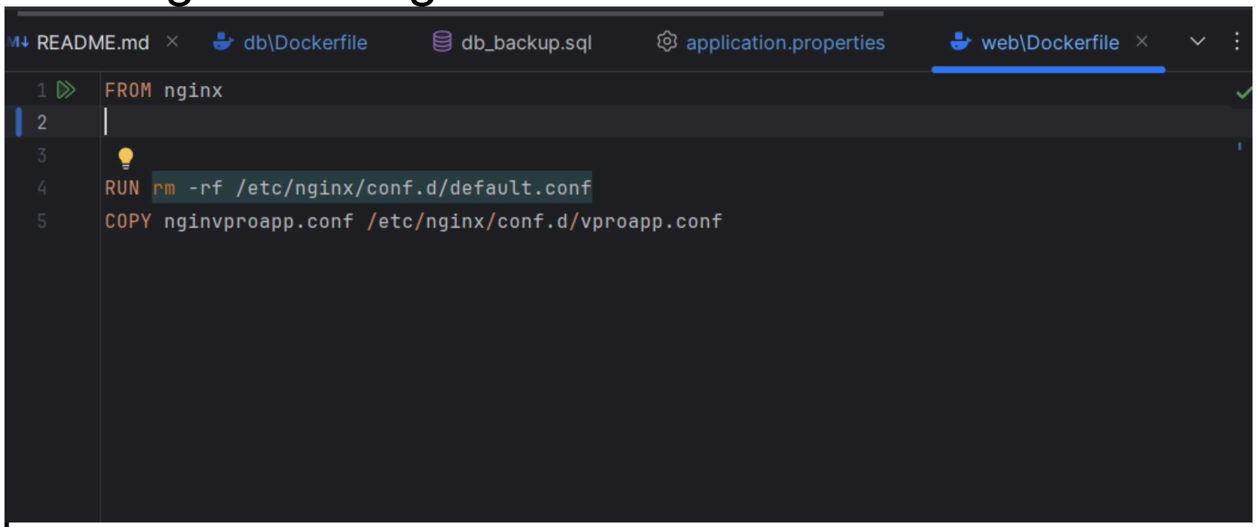
```
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/db# docker build -t darshit
kc12/vprofiledb:V1 .
[+] Building 13.8s (5/7)
 => [internal] load build definition from Dockerfile                          0.0s
 => => transferring dockerfile: 245B                                          0.0s
 => [internal] load .dockerignore                                             0.0s
 => => transferring context: 2B                                               0.0s
 => [internal] load metadata for docker.io/library/mysql:8.0.33               3.5s
 => [auth] library/mysql:pull token for registry-1.docker.io                  0.0s
 => [internal] load build context                                             0.1s
 => => transferring context: 6.34kB                                           0.0s
 => [1/2] FROM docker.io/library/mysql:8.0.33@sha256:ea68e51ffe9b96fef6076f12  10.2s
 => => resolve docker.io/library/mysql:8.0.33@sha256:ea68e51ffe9b96fef6076f121  0.0s
 => => sha256:f6360852d65467a8252391f24826648079893ea03e52d4d8 7.46kB / 7.46kB  0.0s
 => => sha256:eb11df7b201aa831c1f1a0c4a07a630b40fcbfacdb7eb6fd 2.62kB / 2.62kB  0.0s
 => => sha256:ea68e51ffe9b96fef6076f1218af11301aeaf13c6201e0ec9aae 549B / 549B  0.0s
 => => sha256:49bb46380f8cb3491e82d000b27a6eb26b2490291da81 17.83MB / 44.92MB  10.1s
 => => sha256:d6eef8c26cf9c633276ffd191110525157aa8811962f 982.82kB / 982.82kB  2.4s
 => => sha256:aab3066bbf8fc576b01df1b0391e5a72a7ff72b1c0c5d09a7bab 884B / 884B  1.1s
 => => sha256:0e908b1dcba2267120afc44b8f24aba3cdf060da12572571 4.61MB / 4.61MB  3.4s
 => => sha256:480c3912a2fde8647ba9e97d569e907841498684daa29bd8 2.63kB / 2.63kB  3.2s
 => => sha256:89a648ecb3cf74a58ea563b9fa617f389d20ad307b98d1ee31bb 338B / 338B  3.9s
 => => sha256:6313eed007807d8f70c46daf7c87a6036ad41b272af6a 13.63MB / 58.60MB  10.1s
 => => sha256:668fe2d98404a0684b0c56382e44806d0912dcd0f77c2ee15e03 316B / 316B  4.6s
```



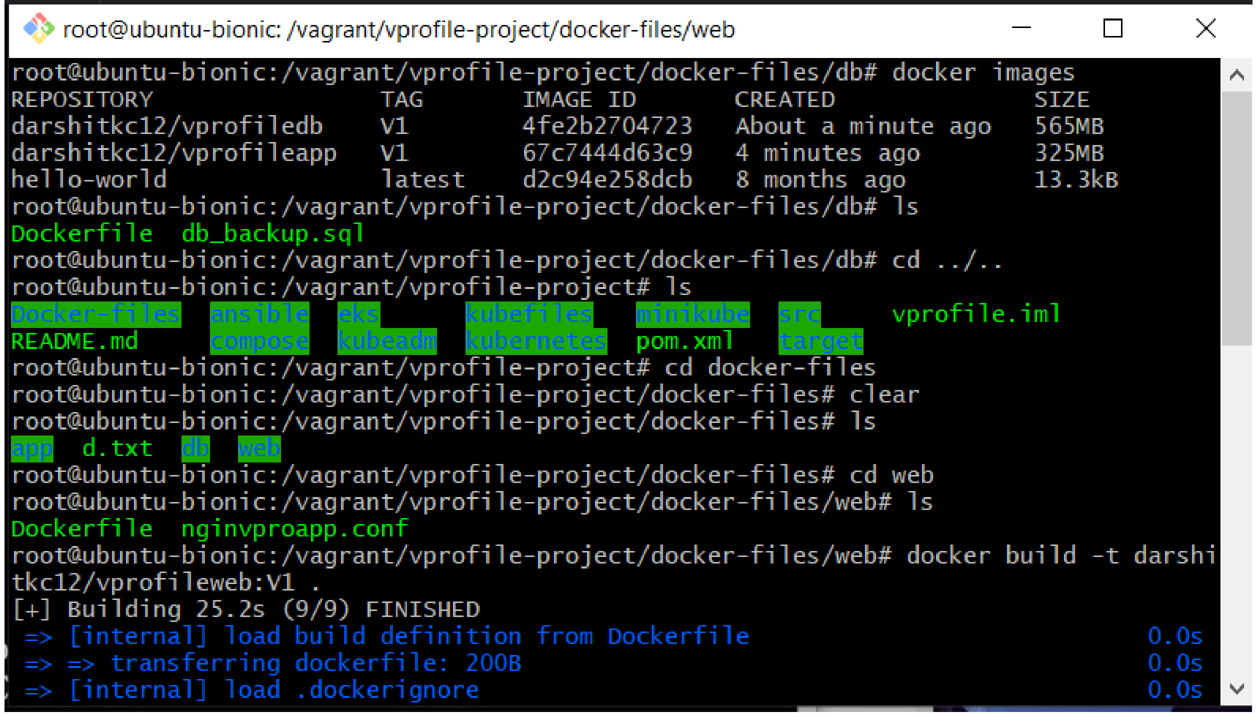
root@ubuntu-bionic: /vagrant/vprofile-project/docker-files/db

```
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/db# docker images
REPOSITORY                TAG       IMAGE ID       CREATED              SIZE
darshitkc12/vprofiledb    V1        4fe2b2704723   About a minute ago   565MB
darshitkc12/vprofileapp   V1        67c7444d63c9   4 minutes ago        325MB
hello-world               latest    d2c94e258dcb   8 months ago         13.3kB
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/db# 
```

## Building web images as docker container

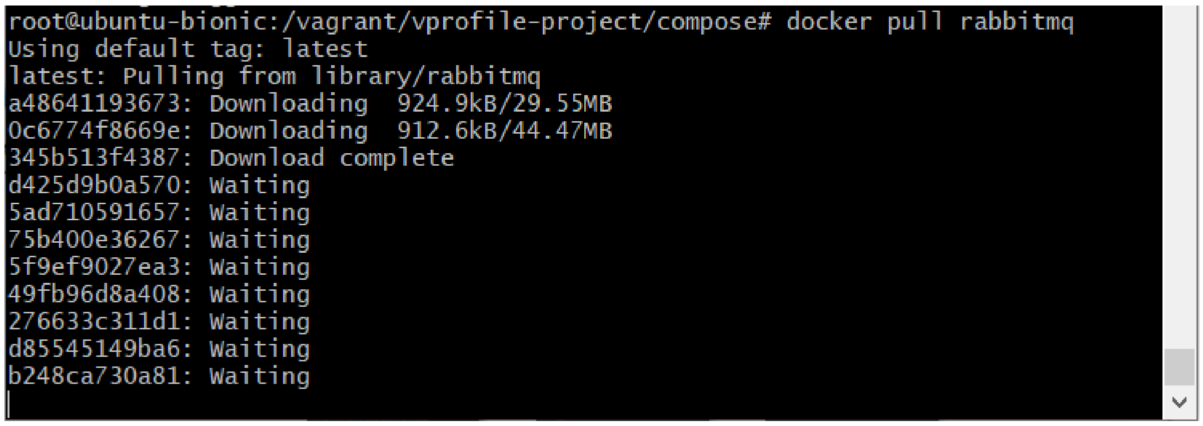


```
FROM nginx


RUN rm -rf /etc/nginx/conf.d/default.conf
COPY nginvproapp.conf /etc/nginx/conf.d/vproapp.conf
```


root@ubuntu-bionic: /vagrant/vprofile-project/docker-files/web

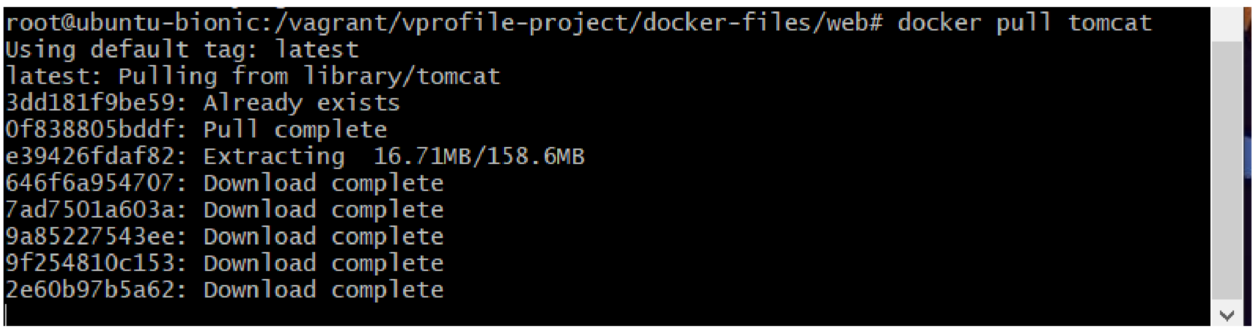
```
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/db# docker images
REPOSITORY                TAG       IMAGE ID       CREATED              SIZE
darshitkc12/vprofiledb    V1        4fe2b2704723   About a minute ago   565MB
darshitkc12/vprofileapp   V1        67c7444d63c9   4 minutes ago        325MB
hello-world               latest    d2c94e258dcb   8 months ago         13.3kB
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/db# ls
Dockerfile  db_backup.sql
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/db# cd ../..
root@ubuntu-bionic:/vagrant/vprofile-project# ls
Docker-files  ansible  eks      kubefiles   minikube  src     vprofile.iml
README.md     compose  kubeadm  kubernetes  pom.xml   target
root@ubuntu-bionic:/vagrant/vprofile-project# cd docker-files
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files# clear
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files# ls
app  d.txt  db  web
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files# cd web
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/web# ls
Dockerfile  nginvproapp.conf
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/web# docker build -t darshi
tkc12/vprofileweb:V1 .
[+] Building 25.2s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                     0.0s
 => => transferring dockerfile: 200B                                     0.0s
 => [internal] load .dockerignore                                        0.0s
```

## Pulling rabbitmq from docker repository



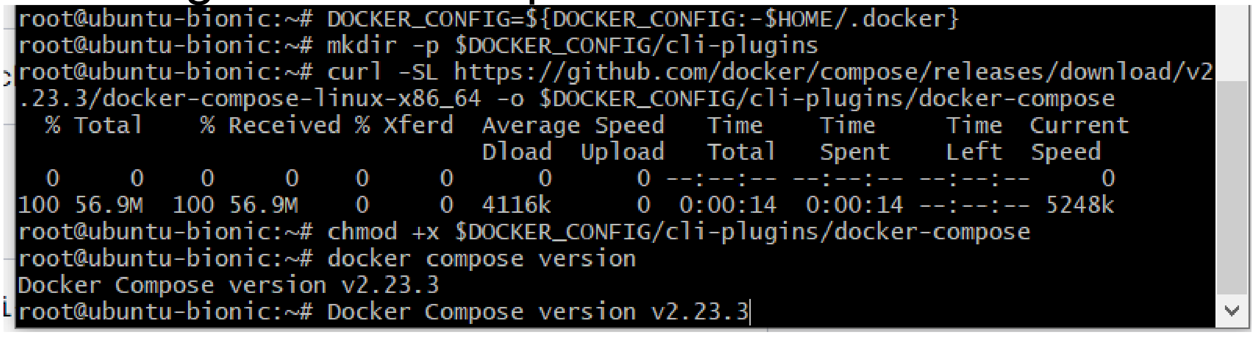
```
root@ubuntu-bionic:/vagrant/vprofile-project/compose# docker pull rabbitmq
Using default tag: latest
latest: Pulling from library/rabbitmq
a48641193673: Downloading  924.9kB/29.55MB
0c6774f8669e: Downloading  912.6kB/44.47MB
345b513f4387: Download complete
d425d9b0a570: Waiting
5ad710591657: Waiting
75b400e36267: Waiting
5f9ef9027ea3: Waiting
49fb96d8a408: Waiting
276633c311d1: Waiting
d85545149ba6: Waiting
b248ca730a81: Waiting
```

## Pulling tomcat from docker repository



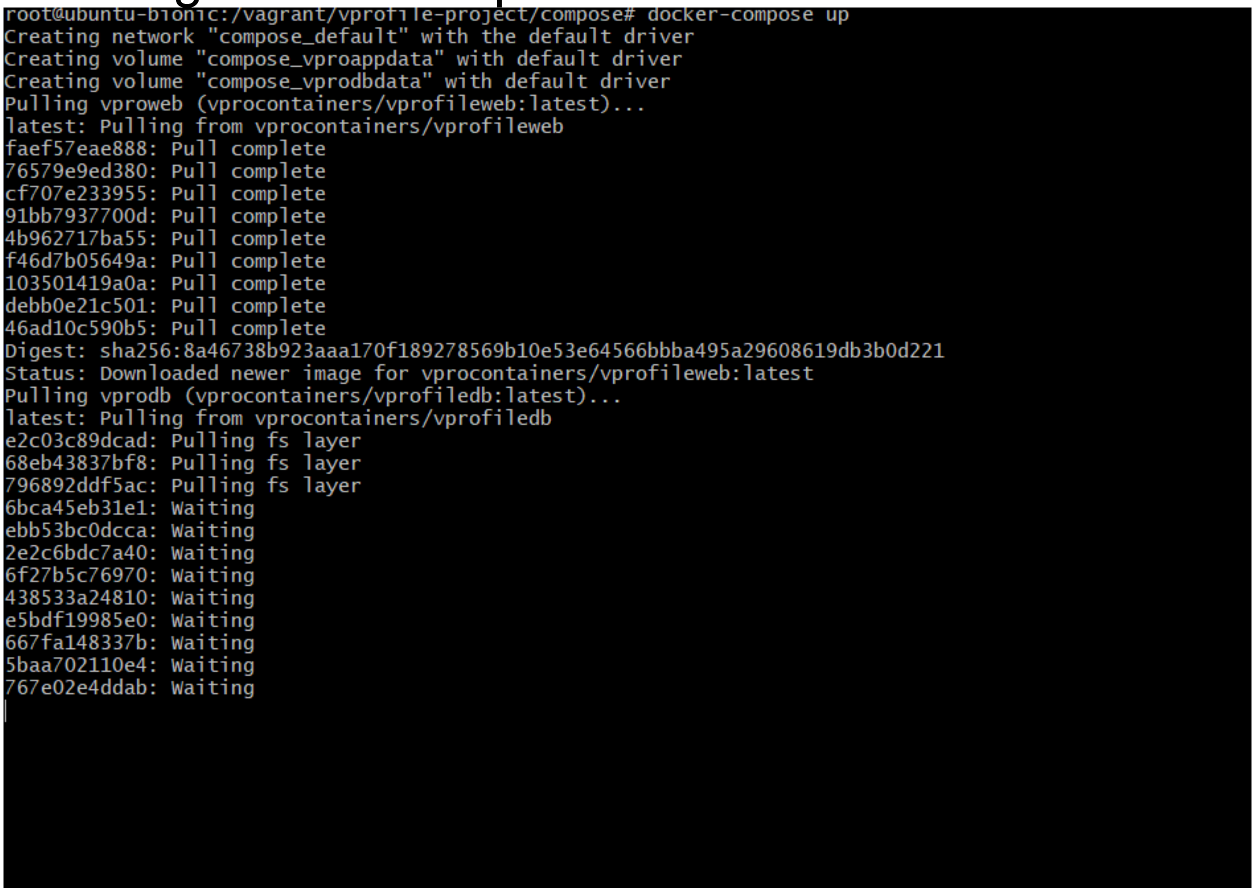
```
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/web# docker pull tomcat
Using default tag: latest
latest: Pulling from library/tomcat
3dd181f9be59: Already exists
0f838805bddf: Pull complete
e39426fdaf82: Extracting  16.71MB/158.6MB
646f6a954707: Download complete
7ad7501a603a: Download complete
9a85227543ee: Download complete
9f254810c153: Download complete
2e60b97b5a62: Download complete
```

## Installing Docker Compose



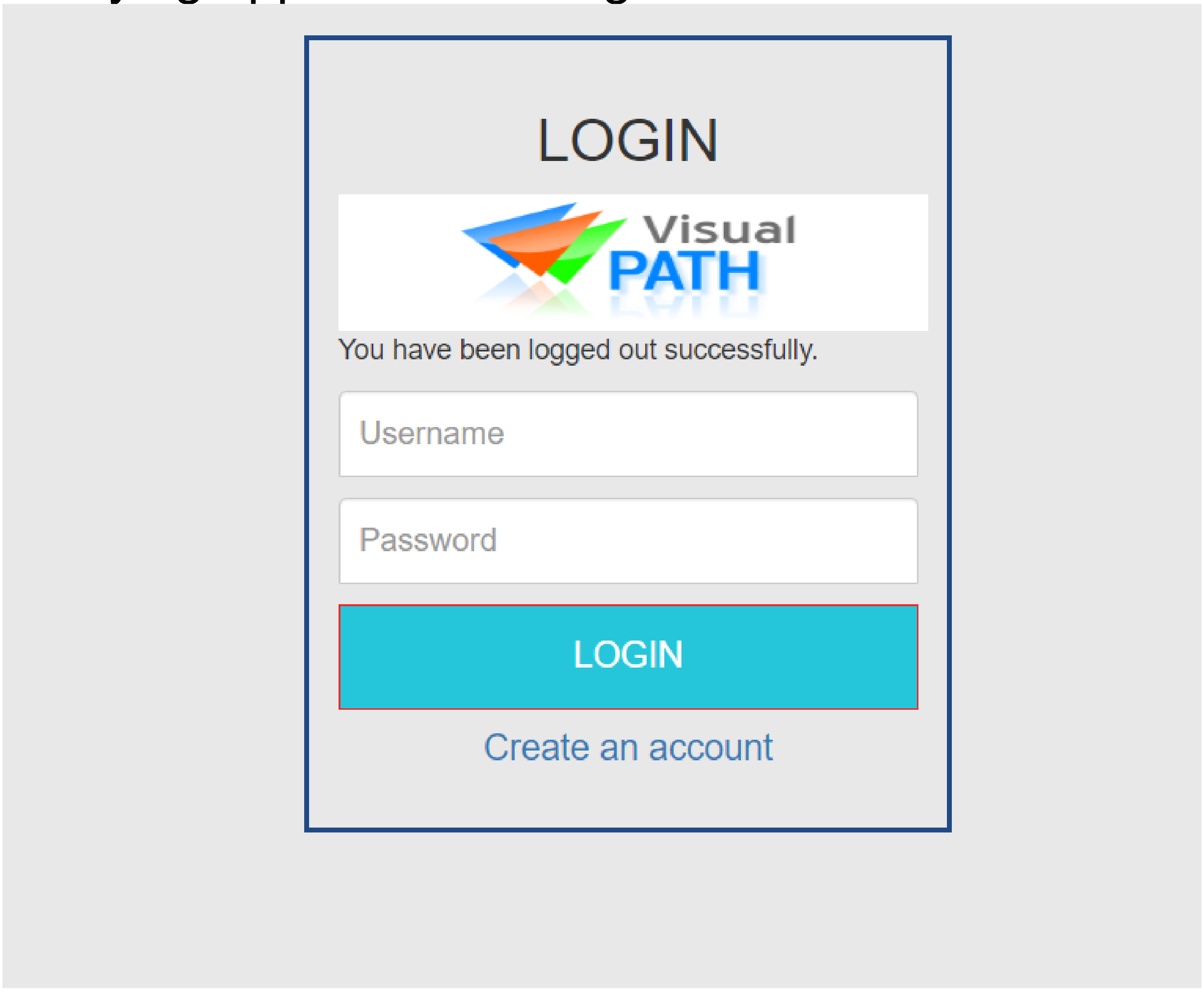
```
root@ubuntu-bionic:~# DOCKER_CONFIG=${DOCKER_CONFIG:-$HOME/.docker}
root@ubuntu-bionic:~# mkdir -p $DOCKER_CONFIG/cli-plugins
root@ubuntu-bionic:~# curl -SL https://github.com/docker/compose/releases/download/v2
.23.3/docker-compose-linux-x86_64 -o $DOCKER_CONFIG/cli-plugins/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 56.9M  100 56.9M    0     0  4116k      0  0:00:14  0:00:14 --:--:-- 5248k
root@ubuntu-bionic:~# chmod +x $DOCKER_CONFIG/cli-plugins/docker-compose
root@ubuntu-bionic:~# docker compose version
Docker Compose version v2.23.3
root@ubuntu-bionic:~# Docker Compose version v2.23.3
```

## Running docker compose



```
root@ubuntu-bionic:/vagrant/vprofile-project/compose# docker-compose up
Creating network "compose_default" with the default driver
Creating volume "compose_vproappdata" with default driver
Creating volume "compose_vprodbdata" with default driver
Pulling vproweb (vprocontainers/vprofileweb:latest)...
latest: Pulling from vprocontainers/vprofileweb
faef57eae888: Pull complete
76579e9ed380: Pull complete
cf707e233955: Pull complete
91bb7937700d: Pull complete
4b962717ba55: Pull complete
f46d7b05649a: Pull complete
103501419a0a: Pull complete
debb0e21c501: Pull complete
46ad10c590b5: Pull complete
Digest: sha256:8a46738b923aaa170f189278569b10e53e64566bbba495a29608619db3b0d221
Status: Downloaded newer image for vprocontainers/vprofileweb:latest
Pulling vprodb (vprocontainers/vprofiledb:latest)...
latest: Pulling from vprocontainers/vprofiledb
e2c03c89dcad: Pulling fs layer
68eb43837bf8: Pulling fs layer
796892ddf5ac: Pulling fs layer
6bca45eb31e1: Waiting
ebb53bc0dcca: Waiting
2e2c6bdc7a40: Waiting
6f27b5c76970: Waiting
438533a24810: Waiting
e5bdf19985e0: Waiting
667fa148337b: Waiting
5baa702110e4: Waiting
767e02e4ddab: Waiting
```

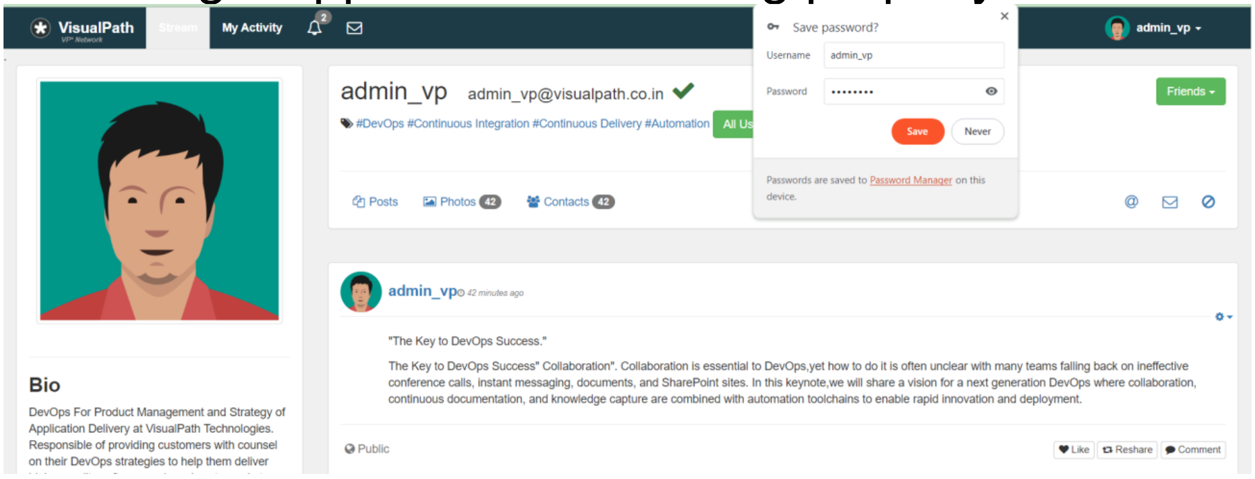
## Verifying application through browser



LOGIN

You have been logged out successfully.

## Checking if application is working properly



admin_vp admin_vp@visualpath.co.in

#DevOps #Continuous Integration #Continuous Delivery #Automation

Username admin_vp

Password ••••••••

Passwords are saved to Password Manager on this device.

Bio

DevOps For Product Management and Strategy of Application Delivery at VisualPath Technologies. Responsible of providing customers with counsel on their DevOps strategies to help them deliver

admin_vp 42 minutes ago

"The Key to DevOps Success."

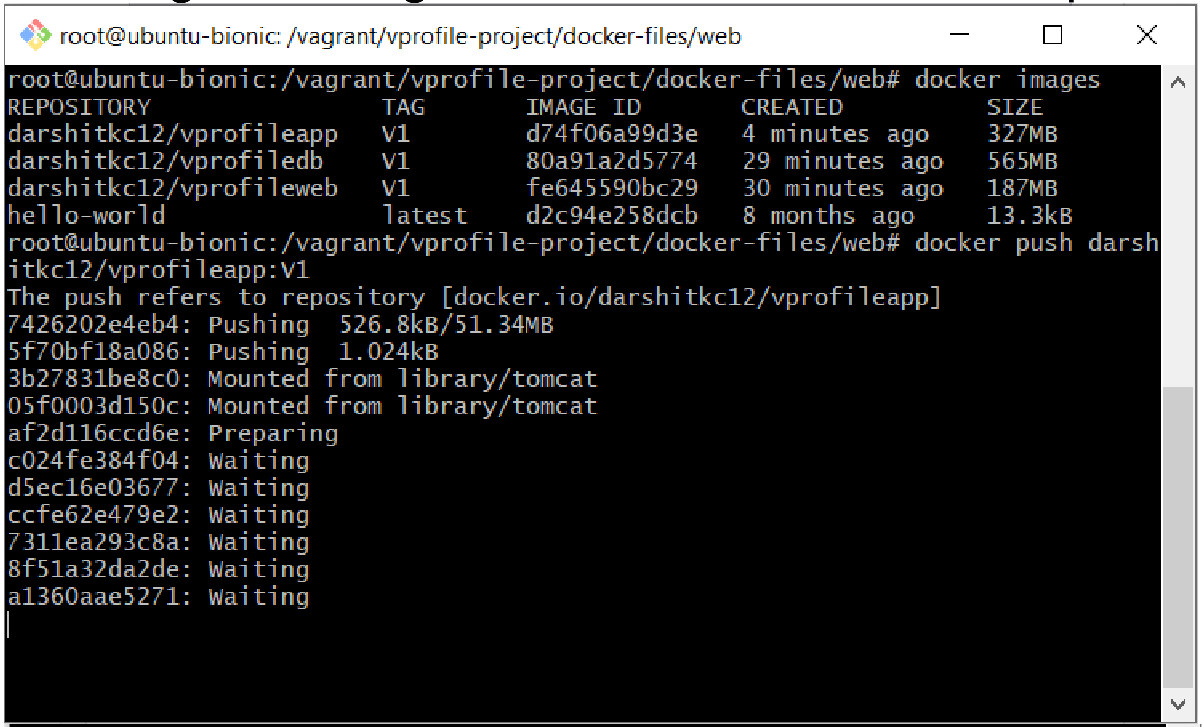
The Key to DevOps Success" Collaboration". Collaboration is essential to DevOps,yet how to do it is often unclear with many teams falling back on ineffective conference calls, instant messaging, documents, and SharePoint sites. In this keynote,we will share a vision for a next generation DevOps where collaboration, continuous documentation, and knowledge capture are combined with automation toolchains to enable rapid innovation and deployment.

## Rabbitmq initiated

### Generated 2 Connections

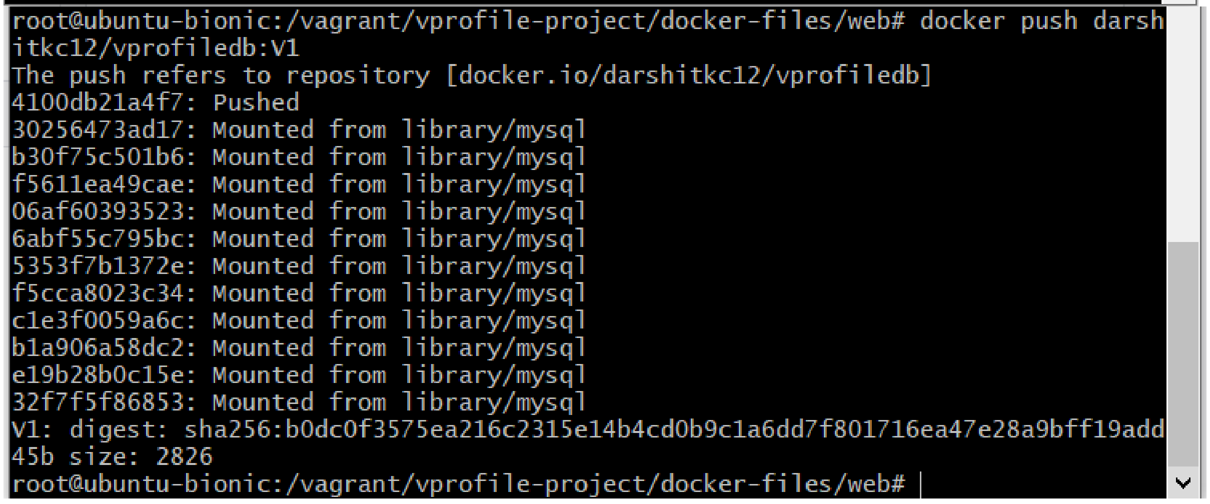
### 6 Chanels 1 Exchage and 2 Que

Pushing image to docker repository

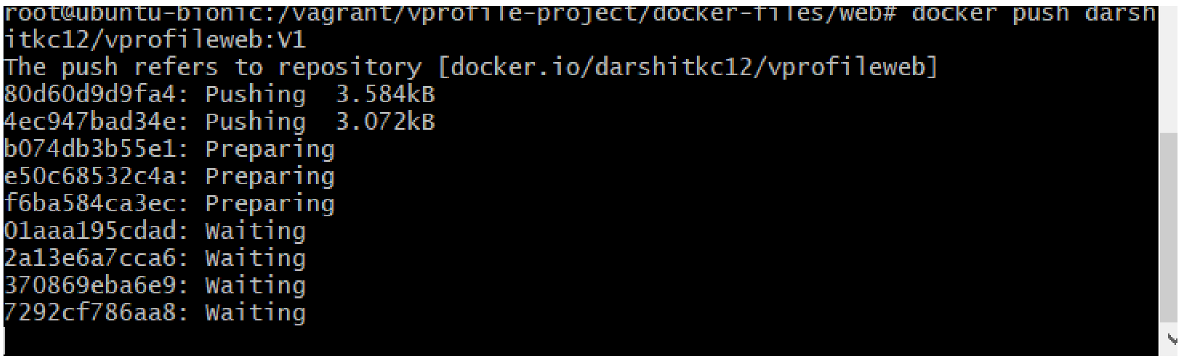


root@ubuntu-bionic: /vagrant/vprofile-project/docker-files/web

```
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/web# docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
darshitkc12/vprofileapp   V1        d74f06a99d3e   4 minutes ago    327MB
darshitkc12/vprofiledb    V1        80a91a2d5774   29 minutes ago   565MB
darshitkc12/vprofileweb   V1        fe645590bc29   30 minutes ago   187MB
hello-world               latest    d2c94e258dcb   8 months ago     13.3kB
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/web# docker push darsh
itkc12/vprofileapp:V1
The push refers to repository [docker.io/darshitkc12/vprofileapp]
7426202e4eb4: Pushing  526.8kB/51.34MB
5f70bf18a086: Pushing  1.024kB
3b27831be8c0: Mounted from library/tomcat
05f0003d150c: Mounted from library/tomcat
af2d116ccd6e: Preparing
c024fe384f04: Waiting
d5ec16e03677: Waiting
ccfe62e479e2: Waiting
7311ea293c8a: Waiting
8f51a32da2de: Waiting
a1360aae5271: Waiting
```

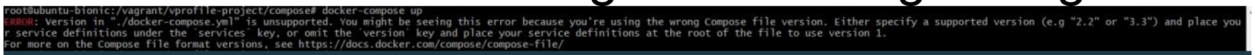


```
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/web# docker push darsh
itkc12/vprofiledb:V1
The push refers to repository [docker.io/darshitkc12/vprofiledb]
4100db21a4f7: Pushed
30256473ad17: Mounted from library/mysql
b30f75c501b6: Mounted from library/mysql
f5611ea49cae: Mounted from library/mysql
06af60393523: Mounted from library/mysql
6abf55c795bc: Mounted from library/mysql
5353f7b1372e: Mounted from library/mysql
f5cca8023c34: Mounted from library/mysql
c1e3f0059a6c: Mounted from library/mysql
b1a906a58dc2: Mounted from library/mysql
e19b28b0c15e: Mounted from library/mysql
32f7f5f86853: Mounted from library/mysql
V1: digest: sha256:b0dc0f3575ea216c2315e14b4cd0b9c1a6dd7f801716ea47e28a9bff19add
45b size: 2826
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/web# |
```

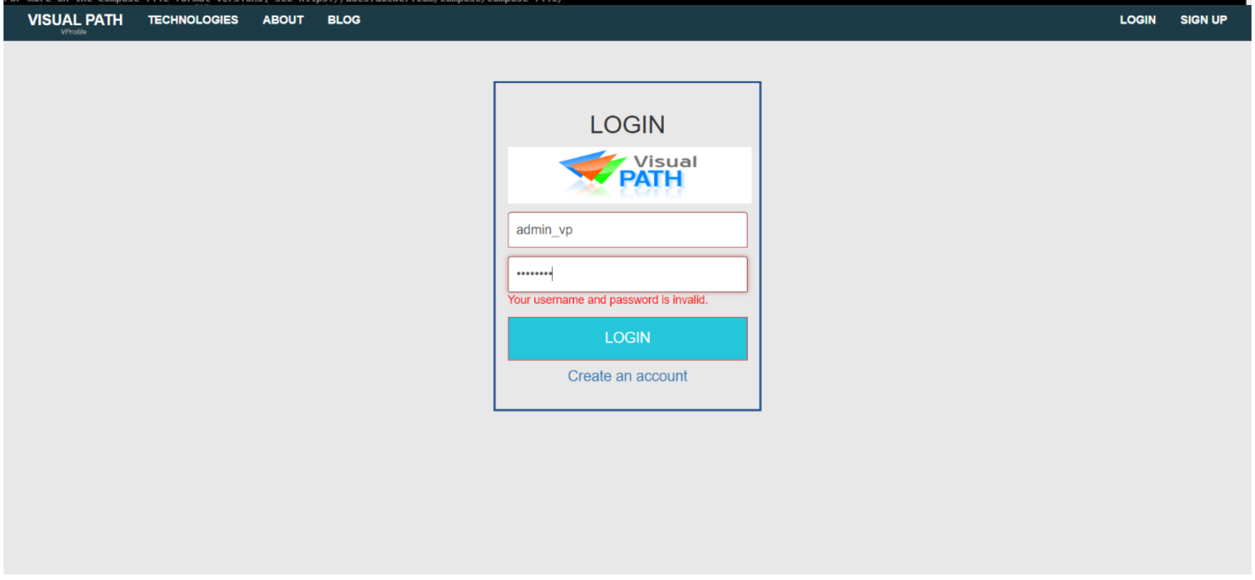


```
root@ubuntu-bionic:/vagrant/vprofile-project/docker-files/web# docker push darsh
itkc12/vprofileweb:V1
The push refers to repository [docker.io/darshitkc12/vprofileweb]
80d60d9d9fa4: Pushing  3.584kB
4ec947bad34e: Pushing  3.072kB
b074db3b55e1: Preparing
e50c68532c4a: Preparing
f6ba584ca3ec: Preparing
01aaa195cdad: Waiting
2a13e6a7cca6: Waiting
370869eba6e9: Waiting
7292cf786aa8: Waiting
```

Error
Error due database configuration being wrong

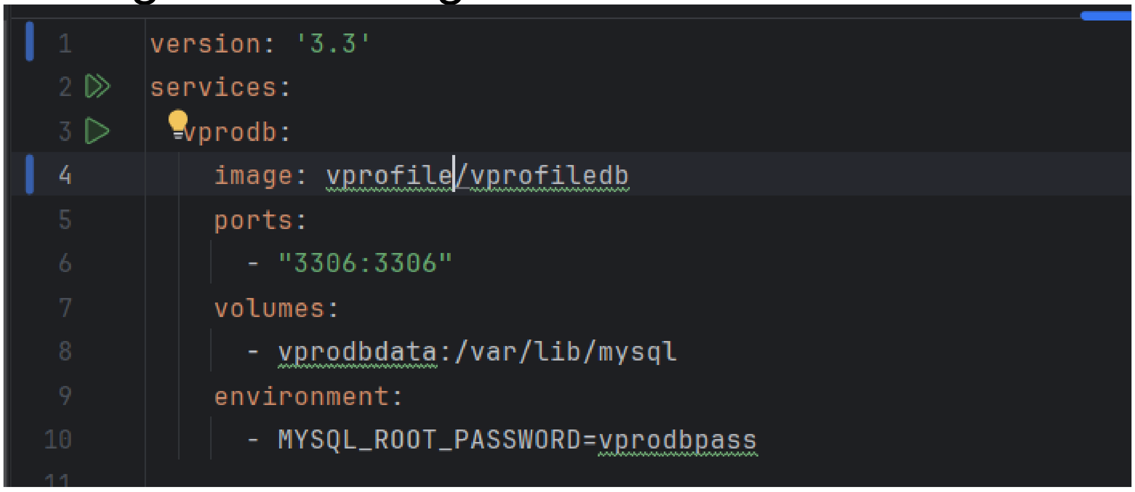


```
root@ubuntu-bionic:/vagrant/vprofile-project/compose# docker-compose up
ERROR: Version in "./docker-compose.yml" is unsupported. You might be seeing this error because you're using the wrong Compose file version. Either specify a supported version (e.g "2.2" or "3.3") and place you
r service definitions under the `services` key, or omit the `version` key and place your service definitions at the root of the file to use version 1.
For more on the Compose file format versions, see https://docs.docker.com/compose/compose-file/
```



admin_vp
••••••••
Your username and password is invalid.

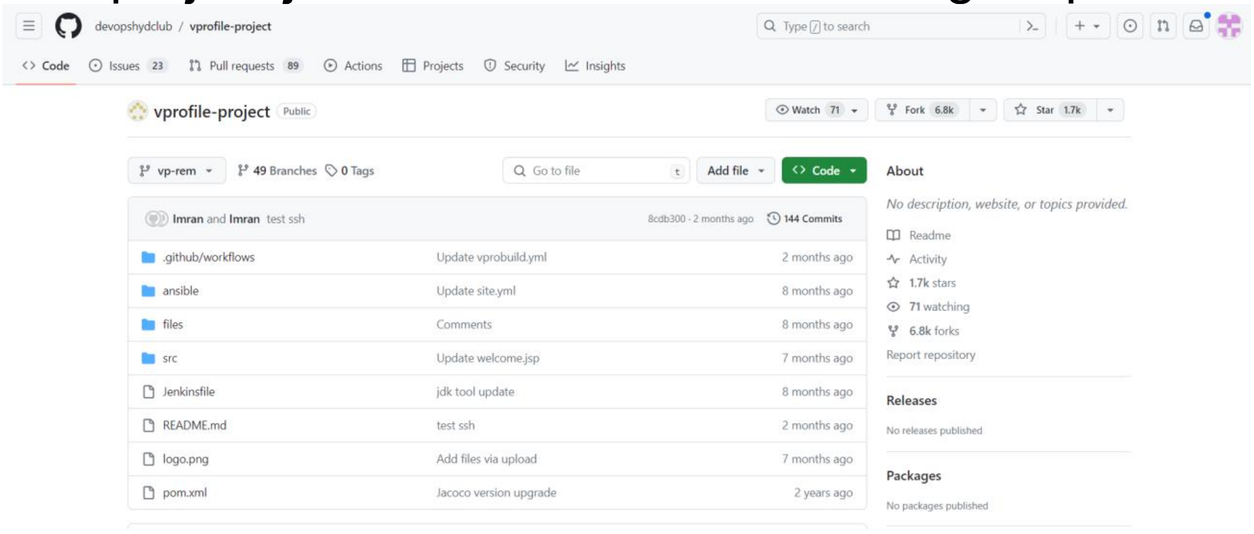
Wrong docker image name



```
version: '3.3'
services:
  vprodb:
    image: vprofile/vprofiledb
    ports:
      - "3306:3306"
    volumes:
      - vprodbdata:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=vprodbpass
```

Study Material

The project java file is taken from these git repo


devopshydclub / vprofile-project

vprofile-project Public

vp-rem · 49 Branches · 0 Tags

Imran and Imran test ssh · 8cdb300 · 2 months ago · 144 Commits

| | | |
|---|---|---|
| .github/workflows | Update vprobuild.yml | 2 months ago |
| ansible | Update site.yml | 8 months ago |
| files | Comments | 8 months ago |
| src | Update welcome.jsp | 7 months ago |
| Jenkinsfile | jdk tool update | 8 months ago |
| README.md | test ssh | 2 months ago |
| logo.png | Add files via upload | 7 months ago |
| pom.xml | Jacoco version upgrade | 2 years ago |

About

No description, website, or topics provided.

Readme
Activity
1.7k stars
71 watching
6.8k forks

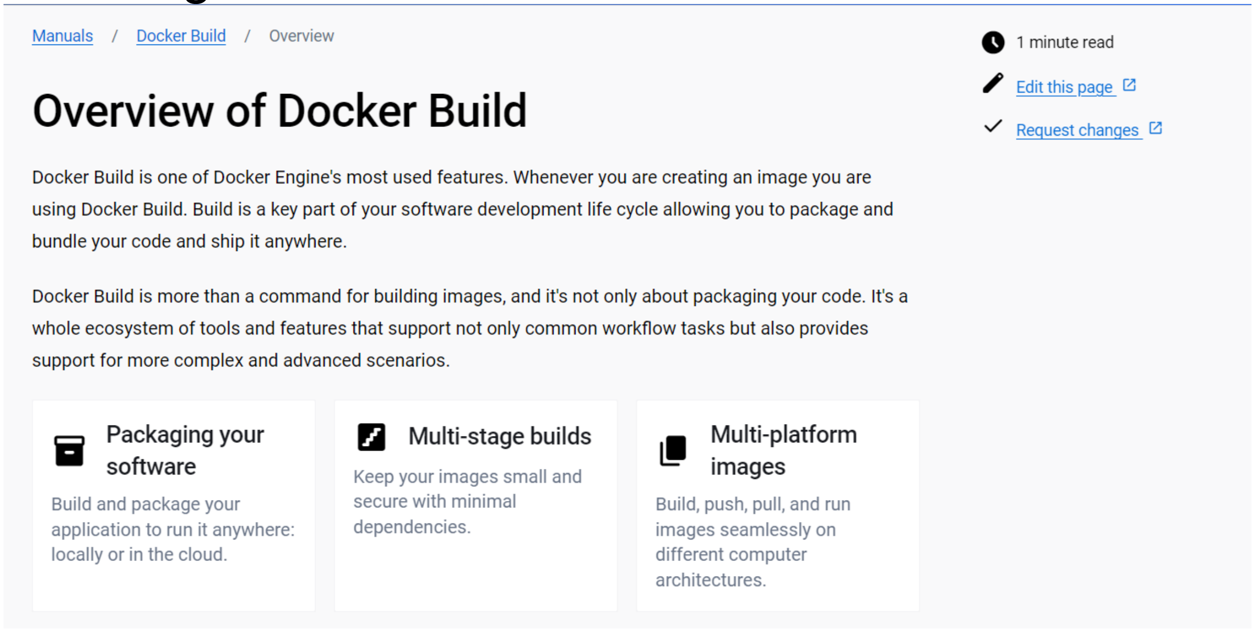
Releases

No releases published

Packages

No packages published

## Learning about docker



Manuals / Docker Build / Overview

1 minute read
Edit this page
Request changes

### Overview of Docker Build

Docker Build is one of Docker Engine's most used features. Whenever you are creating an image you are using Docker Build. Build is a key part of your software development life cycle allowing you to package and bundle your code and ship it anywhere.

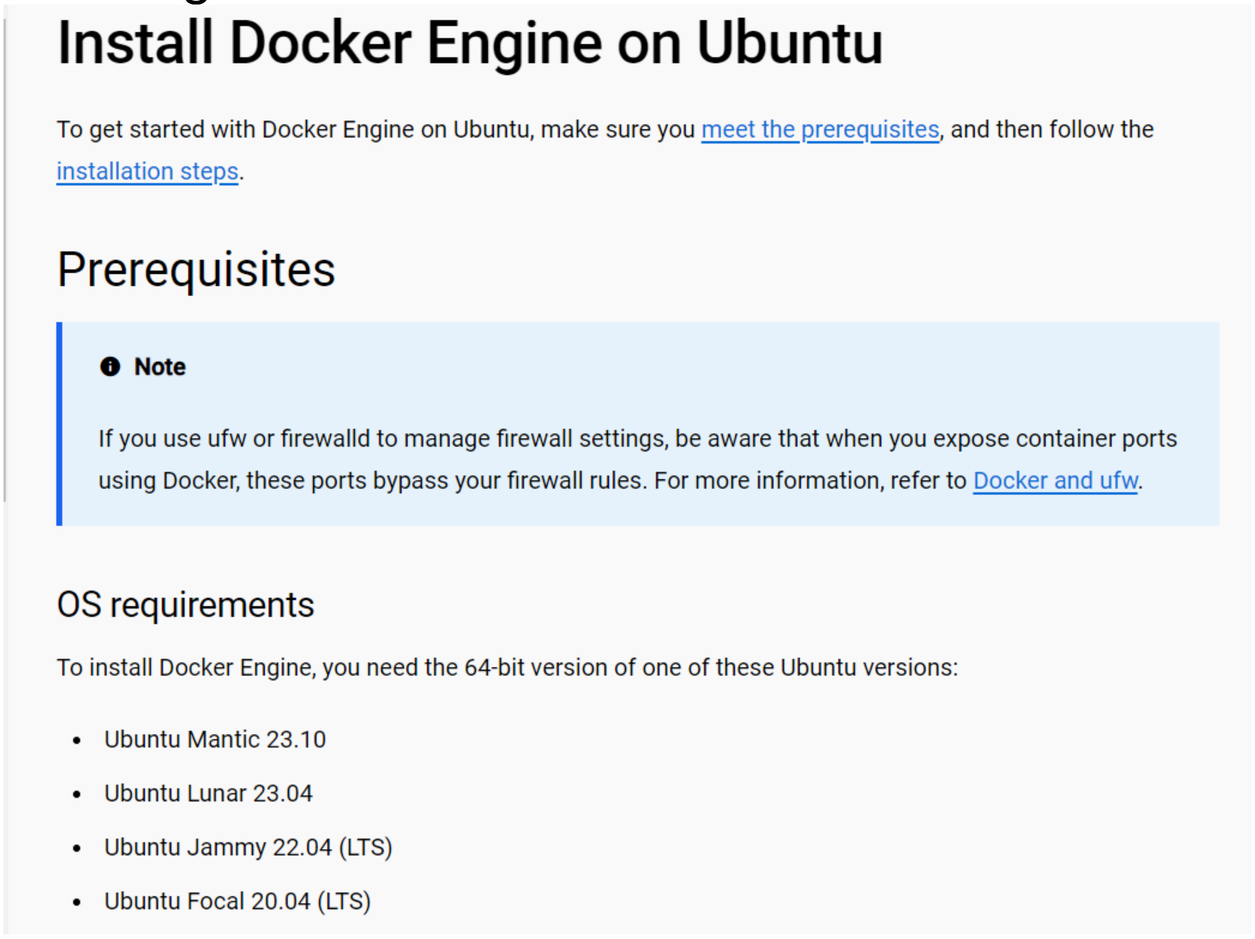
Docker Build is more than a command for building images, and it's not only about packaging your code. It's a whole ecosystem of tools and features that support not only common workflow tasks but also provides support for more complex and advanced scenarios.

**Packaging your software**
Build and package your application to run it anywhere: locally or in the cloud.

**Multi-stage builds**
Keep your images small and secure with minimal dependencies.

**Multi-platform images**
Build, push, pull, and run images seamlessly on different computer architectures.

## Learning about installation about docker



### Install Docker Engine on Ubuntu

To get started with Docker Engine on Ubuntu, make sure you meet the prerequisites, and then follow the installation steps.

### Prerequisites

**Note**

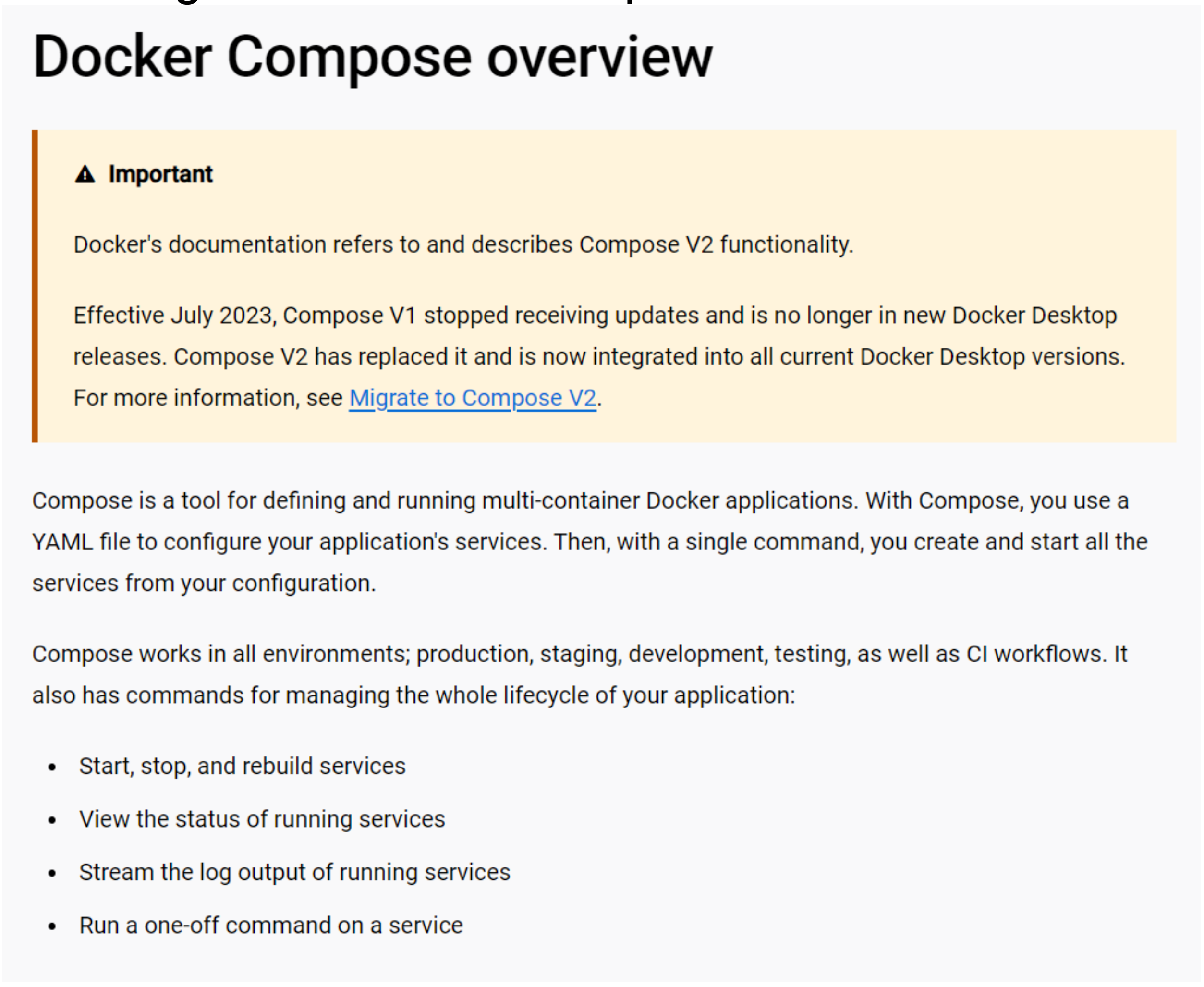
If you use ufw or firewalld to manage firewall settings, be aware that when you expose container ports using Docker, these ports bypass your firewall rules. For more information, refer to Docker and ufw.

#### OS requirements

To install Docker Engine, you need the 64-bit version of one of these Ubuntu versions:

- Ubuntu Mantic 23.10
- Ubuntu Lunar 23.04
- Ubuntu Jammy 22.04 (LTS)
- Ubuntu Focal 20.04 (LTS)

## Learning about docker compose



## Docker Compose overview

**Important**

Docker's documentation refers to and describes Compose V2 functionality.

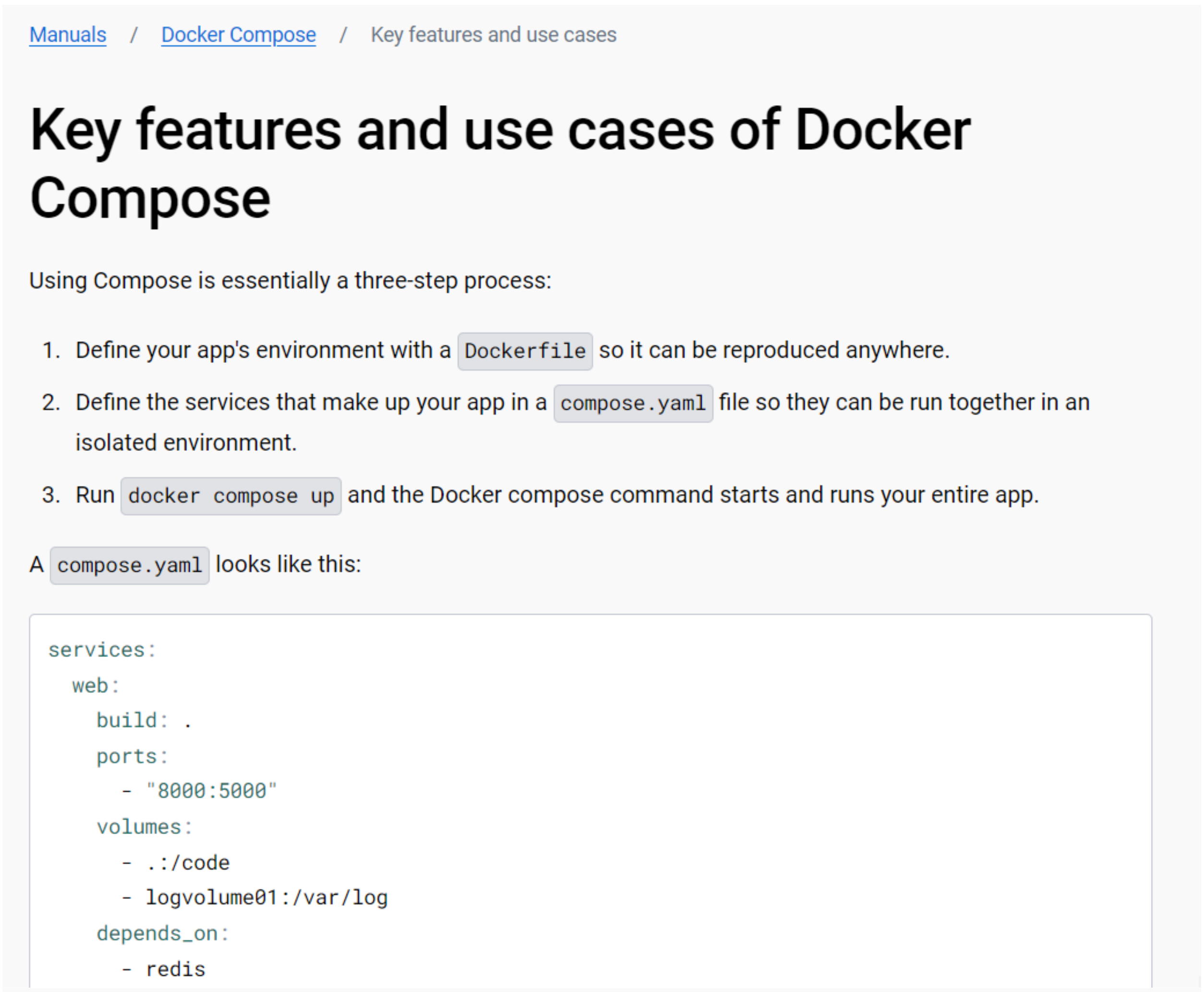
Effective July 2023, Compose V1 stopped receiving updates and is no longer in new Docker Desktop releases. Compose V2 has replaced it and is now integrated into all current Docker Desktop versions. For more information, see Migrate to Compose V2.

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

Compose works in all environments; production, staging, development, testing, as well as CI workflows. It also has commands for managing the whole lifecycle of your application:

- Start, stop, and rebuild services
- View the status of running services
- Stream the log output of running services
- Run a one-off command on a service

Learning about key and feature about docker compose



Manuals / Docker Compose / Key features and use cases

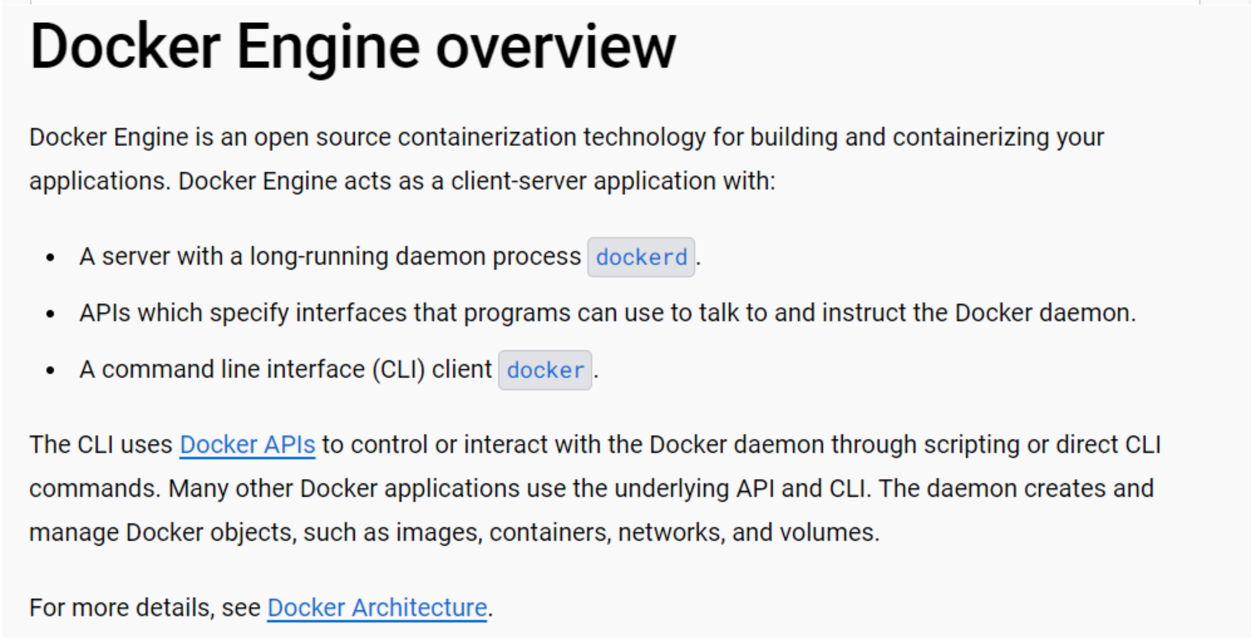
## Key features and use cases of Docker Compose

Using Compose is essentially a three-step process:

1. Define your app's environment with a `Dockerfile` so it can be reproduced anywhere.
2. Define the services that make up your app in a `compose.yaml` file so they can be run together in an isolated environment.
3. Run `docker compose up` and the Docker compose command starts and runs your entire app.

A `compose.yaml` looks like this:

```
services:
  web:
    build: .
    ports:
      - "8000:5000"
    volumes:
      - .:/code
      - logvolume01:/var/log
    depends_on:
      - redis
```



## Docker Engine overview

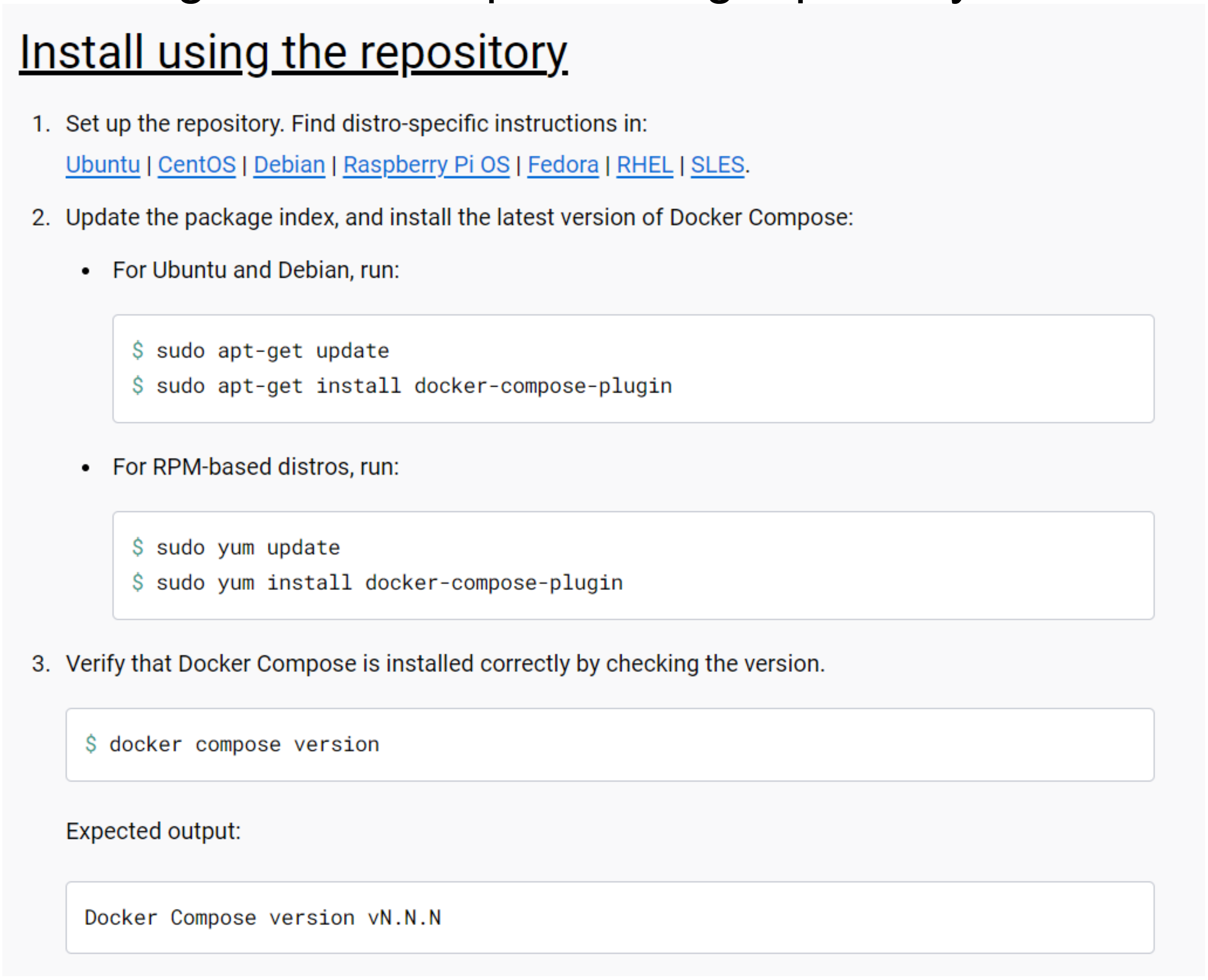
Docker Engine is an open source containerization technology for building and containerizing your applications. Docker Engine acts as a client-server application with:

- A server with a long-running daemon process `dockerd`.
- APIs which specify interfaces that programs can use to talk to and instruct the Docker daemon.
- A command line interface (CLI) client `docker`.

The CLI uses Docker APIs to control or interact with the Docker daemon through scripting or direct CLI commands. Many other Docker applications use the underlying API and CLI. The daemon creates and manage Docker objects, such as images, containers, networks, and volumes.

For more details, see Docker Architecture.

Installing docker compose using repository



## Install using the repository

1. Set up the repository. Find distro-specific instructions in:
   Ubuntu | CentOS | Debian | Raspberry Pi OS | Fedora | RHEL | SLES.
2. Update the package index, and install the latest version of Docker Compose:
   - For Ubuntu and Debian, run:

     ```
     $ sudo apt-get update
     $ sudo apt-get install docker-compose-plugin
     ```

   - For RPM-based distros, run:

     ```
     $ sudo yum update
     $ sudo yum install docker-compose-plugin
     ```

3. Verify that Docker Compose is installed correctly by checking the version.

   ```
   $ docker compose version
   ```

   Expected output:

   ```
   Docker Compose version vN.N.N
   ```

The following project was completed using study material as well as video