

Assignment 03



Abir Ahammed Bhuiyan

ID: 20101197

Dept. of Computer
Science and Engineering

Section : 01

Subject : CSE484

Semester : Spring-2024

Date : 22.02.2024

1. Install docker

We will install docker in Arch Linux so we will be using `pacman` to install docker.

```
sudo pacman -S docker docker-compose
```

```
[abir@ahammed-20101197] [~/docker]
└─o sudo pacman -S docker docker-compose
[sudo] password for abir:
warning: docker-1:25.0.2-1 is up to date -- reinstalling
resolving dependencies...
looking for conflicting packages...

Packages (2) docker-1:25.0.2-1  docker-compose-2.24.5-1

Total Download Size:    12.60 MiB
Total Installed Size:   165.47 MiB
Net Upgrade Size:       56.20 MiB

:: Proceed with installation? [Y/n] y
```

Install has been completed, let's `start` and `enable` docker service and check the status if the service is running.

```
sudo systemctl start docker
sudo systemctl enable docker
sudo systemctl status docker
```

```
[abir@ahammed-20101197] [~/docker]
└─o sudo systemctl start docker
[abir@ahammed-20101197] [~/docker]
└─o sudo systemctl enable docker
created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[abir@ahammed-20101197] [~/docker]
└─o sudo systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: disabled)
  Active: active (running) since Fri 2024-02-16 05:23:27 +06; 12s ago
TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
  Main PID: 14289 (dockerd)
    Tasks: 11
   Memory: 51.7M (peak: 63.5M)
      CPU: 473ms
     CGroup: /system.slice/docker.service
             └─14289 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Feb 16 05:23:27 ahammed-20101197 systemd[1]: Starting Docker Application Container Engine...
Feb 16 05:23:27 ahammed-20101197 dockerd[14289]: time="2024-02-16T05:23:27.189364255+06:00" level=info>
Feb 16 05:23:27 ahammed-20101197 dockerd[14289]: time="2024-02-16T05:23:27.318990492+06:00" level=info>
Feb 16 05:23:27 ahammed-20101197 dockerd[14289]: time="2024-02-16T05:23:27.643802257+06:00" level=info>
Feb 16 05:23:27 ahammed-20101197 dockerd[14289]: time="2024-02-16T05:23:27.707499227+06:00" level=warn>
Feb 16 05:23:27 ahammed-20101197 dockerd[14289]: time="2024-02-16T05:23:27.707892885+06:00" level=info>
Feb 16 05:23:27 ahammed-20101197 dockerd[14289]: time="2024-02-16T05:23:27.708203455+06:00" level=info>
Feb 16 05:23:27 ahammed-20101197 dockerd[14289]: time="2024-02-16T05:23:27.839681934+06:00" level=info>
Feb 16 05:23:27 ahammed-20101197 systemd[1]: Started Docker Application Container Engine.
[lines 1-21/21 (END)]
```

To avoid using `sudo` with each docker command, we can add our account (in this case `abir`) to the docker group like this:

```
sudo usermod -aG docker abir
```

After changing user group setting we must `logout` for the above change to take effect.

Now, verify if the docker is working or not by using the below command.

```
docker run hello-world
```

```
[abir@ahammed-20101197] [~/docker]
└─$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
clec31eb5944: Pull complete
Digest: sha256:d000bc569937abbe195e20322a0bde6b2922d805332fd6d8a68b19f524b7d21d
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/
```

Docker has been successfully installed and working in our machine.

2. Show outputs of basic Docker commands (i.e pull, search, run, build, commit, rm, rmi, etc.. find more from Google)

pull

This command uses to pull docker images from docker hub via CLI.

```
docker pull archlinux
```

```
[abir@ahammed-20101197] [~/docker]
└─o docker pull archlinux
Using default tag: latest
latest: Pulling from library/archlinux
9a82a64c3a84: Pull complete
45f82ee8a39c: Pull complete
Digest: sha256:fed6efe803e79d94544f93607e4afec1056cf9bee5744c965eec0944624d81f
Status: Downloaded newer image for archlinux:latest
docker.io/library/archlinux:latest
```

images

By using `images` command we can see what images have been stored in our local machine.

```
docker images
```

```
[abir@ahammed-20101197] [~/docker]
└─o docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
archlinux        latest    69f38d8f6347  6 weeks ago   436MB
hello-world     latest    d2c94e258dcf  9 months ago  13.3kB
```

run

To run a docker image we can use `run` command. We can specify a flag `-it`, which just stands for `interactive terminal`.

```
docker run -it <image_name>
```

```
[abir@ahammed-20101197] [~/docker]
└─o docker run -it archlinux:latest /bin/bash
[root@c4fdaed69e9f /]# pacman -Syyu
:: Synchronizing package databases...
 core downloading...
 extra downloading...
 :: Starting full system upgrade...
 resolving dependencies...
 :: There are 2 providers available for dbus-units:
 :: Repository core
 1) dbus-broker-units 2) dbus-daemon-units
```

version

Using `--version` or `version` command we can see all the version information of docker.

```
[abir@ahammed-20101197] [~/docker]
└─$ docker --version
Docker version 25.0.2, build 29cf629222
[abir@ahammed-20101197] [~/docker]
└─$ docker version
Client:
Version:          25.0.2
API version:      1.44
Go version:       go1.21.6
Git commit:       29cf629222
Built:            Thu Feb  1 10:50:44 2024
OS/Arch:          linux/amd64
Context:          default

Server:
Engine:
  Version:          25.0.2
  API version:      1.44 (minimum version 1.24)
  Go version:       go1.21.6
  Git commit:       fce6e0ca9b
  Built:            Thu Feb  1 10:50:44 2024
  OS/Arch:          linux/amd64
  Experimental:    false
containerd:
  Version:          v1.7.13
  GitCommit:        7c3aca7a610df76212171d200ca3811ff6096eb8.m
runc:
  Version:          1.1.12
  GitCommit:
docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

rmi

Using `rmi` command we can remove an image. In the picture, we have tried to remove an image name `myarch`.

```
[abir@ahammed-20101197] [~]
└ o docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
image-serve     latest   d8b5452dfa33  33 hours ago  42.6MB
eniac00/myarch  latest   4991cc415eaf  41 hours ago  436MB
myarch          latest   4991cc415eaf  41 hours ago  436MB
nginx           alpine   6913ed9ec8d0  7 days ago   42.6MB
mongo           latest   b8df2163f9aa  2 weeks ago   755MB
archlinux        latest   69f38d8f6347  7 weeks ago   436MB
hello-world     latest   d2c94e258dcb  9 months ago  13.3kB
[abir@ahammed-20101197] [~]
└ o docker rmi myarch:latest
Untagged: myarch:latest
[abir@ahammed-20101197] [~]
└ o docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
image-serve     latest   d8b5452dfa33  33 hours ago  42.6MB
eniac00/myarch  latest   4991cc415eaf  41 hours ago  436MB
nginx           alpine   6913ed9ec8d0  7 days ago   42.6MB
mongo           latest   b8df2163f9aa  2 weeks ago   755MB
archlinux        latest   69f38d8f6347  7 weeks ago   436MB
hello-world     latest   d2c94e258dcb  9 months ago  13.3kB
[abir@ahammed-20101197] [~]
└ o
```

ps

Using `ps -a` we can view process status of all the containers.

```
[abir@ahammed-20101197] [~]
└ o docker ps -a
CONTAINER ID  IMAGE          COMMAND          CREATED      STATUS          PORTS      NAMES
6c0db7707365  469ae7bf5468  "/docker-entrypoint..."  21 hours ago  Exited (0) 21 hours ago
27bdf39819f6  image-serve    "/docker-entrypoint..."  33 hours ago  Exited (0) 32 hours ago
[abir@ahammed-20101197] [~]
└ o
```

3. Create a Docker image using Dockerfile.

We are going to create our own version of Arch Linux that will print `Hello World` from my ArchLinux.

First we have to create a directory and open a new file with any name in our case we have created a directory called `Docker` and inside that directory we have created a file named `myArch`.

```
[abir@ahammed-20101197] [~]
└ o mkdir Docker
[abir@ahammed-20101197] [~]
└ o cd Docker/
[abir@ahammed-20101197] [~/Docker]
└ o touch myArch
[abir@ahammed-20101197] [~/Docker]
└ o
```

Then write the following in the file and save it.

```
1
1 FROM archlinux
2
3 MAINTAINER eniac00 <eniac00@gmail.com>
4
5 CMD ["echo", "Hello World from my ArchLinux"]
6
```

NORMAL myArch utf-8 14% N:1/7

After that we have to build the docker image from the docker file. To do that simply run the below command.

```
docker build -t my-arch -f myArch .
```

```
[abir@ahammed-20101197] [~/Docker]
└─$ docker build -t my-arch -f myArch .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 2.048kB
Step 1/3 : FROM archlinux
--> 69f38d8f6347
Step 2/3 : MAINTAINER eniac00 <eniac00@gmail.com>
--> Running in 3f0223016025
--> Removed intermediate container 3f0223016025
--> 7d5ae21792bf
Step 3/3 : CMD ["echo", "Hello World from my ArchLinux"]
--> Running in b4bab7c4344b
--> Removed intermediate container b4bab7c4344b
--> 4991cc415eaf
Successfully built 4991cc415eaf
Successfully tagged my-arch:latest
└─$
```

Our docker image is ready to go in our case its name is `my-arch`. Let's run it.

```
docker run my-arch
```

```
[abir@ahammed-20101197] [~/Docker]
└ o docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
my-arch         latest   4991cc415eaf  54 seconds ago  436MB
archlinux        latest   69f38d8f6347  7 weeks ago   436MB
hello-world     latest   d2c94e258dcf  9 months ago   13.3kB
[abir@ahammed-20101197] [~/Docker]
└ o docker run my-arch
Hello World from my ArchLinux
[abir@ahammed-20101197] [~/Docker]
```

After running it we can see our desired output like the picture above.

4. Run a container as a single task, show outputs, and show the status of all containers (using docker ps -a)

To run a docker image (later container) as a single task we can use the `run` command like below.

```
docker run hello-world
```

Here `hello-world` is the image name.

```
[abir@ahammed-20101197] [~/Docker]
└ o docker run hello-world

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/
[abir@ahammed-20101197] [~/Docker]
└ o docker run my-arch
Hello World from my ArchLinux
[abir@ahammed-20101197] [~/Docker]
```

To see the all the containers status run a command like below.

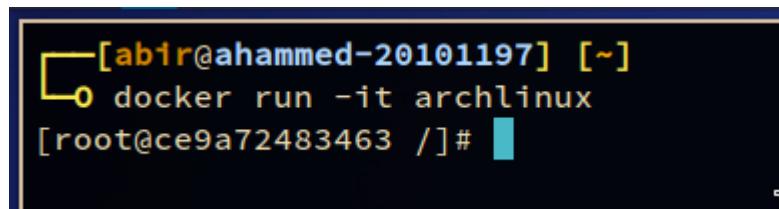
```
docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
6464f5d9a32d	my-arch	"echo 'Hello World f..."	46 seconds ago	Exited (0) 45 seconds ago		recurring_galileo
960992a6999b	hello-world	"/hello"	54 seconds ago	Exited (0) 53 seconds ago		bold_mclean
886ba1dfe398	hello-world	"/hello"	2 minutes ago	Exited (0) 2 minutes ago		beautiful_cori
0c52e25752d5	archlinux	"/usr/bin/bash"	3 minutes ago	Exited (130) 2 minutes ago		awesome_bose
b30cc860d71	my-arch	"echo 'Hello World f..."	4 minutes ago	Exited (0) 3 minutes ago		thirsty_feistel

5. Run a container in iterative mode and install different packages in the container. Show each step.

Using the below command we can run a docker image in iterative mode (here `archlinux` is the image name).

```
docker run -it archlinux
```



After running we can see that our prompt has been changed, that indicates we are currently in the shell of our newly created container. we can install different packages but first we have to update the package manager in our case that is `pacman`.

```
pacman -Syu
```

```
pacman -S neofetch  
pacman -S vim
```

```
[root@ce9a72483463 /]# pacman -Syu  
:: Synchronizing package databases...  
core downloading...  
extra downloading...  
:: Starting full system upgrade...  
resolving dependencies...  
:: There are 2 providers available for dbus-units:  
:: Repository core  
  1) dbus-broker-units  2) dbus-daemon-units  
  
Enter a number (default=1):  
looking for conflicting packages...
```

We have tried to install two packages `neofetch` and `vim`. After installing we have tried to run them and they ran successfully.

```
[root@ce9a72483463 /]# pacman -S neofetch
resolving dependencies...
looking for conflicting packages...

Package (1)      New Version  Net Change  Download Size

extra/neofetch  7.1.0-2           0.33 MiB     0.08 MiB

Total Download Size:  0.08 MiB
Total Installed Size:  0.33 MiB

:: Proceed with installation? [Y/n] y
:: Retrieving packages...
neofetch-7.1.0-2-any downloading...
checking keyring...
checking package integrity...
loading package files...
checking for file conflicts...
:: Processing package changes...
installing neofetch...
optional dependencies for neofetch
  catimg: Display Images
  chafa: Image to text support
  feh: Wallpaper Display
  imagemagick: Image cropping / Thumbnail creation / Take a screenshot
  jp2a: Display Images
  libcaca: Display Images
  nitrogen: Wallpaper Display
  w3m: Display Images
  xdotool: See https://github.com/dylanaraps/neofetch/wiki/Images-in-the-terminal
  xorg-xdpyinfo: Resolution detection (Single Monitor)
  xorg-xprop: Desktop Environment and Window Manager
  xorg-xrandr: Resolution detection (Multi Monitor + Refresh rates)
  xorg-xwininfo: See https://github.com/dylanaraps/neofetch/wiki/Images-in-the-terminal
:: Running post-transaction hooks...
(1/1) Arming ConditionNeedsUpdate...
```

```
[root@ce9a72483463 /]# neofetch
 _`_
 .o+'
 `ooo/
 `+oooo:
 `+oooooo:
 --+oooooo+:
 `/:-:++oooo+:
 `/++++/++++++:
 `/+++++++/+++++:
 `/+++++oooooooooo/`_
 ./ooooooooooooo+`_
 .oooooooo-````/oooooooo+
 -oooooooo. :oooooooo.
 :oooooooo/   oooooo+..
 /oooooooo/   +oooooo/-.
 `/oooooooo+/-  -:/+oooooooo+-.
 `+so+:-`       `.-/+oso:
 `++:.          `-/+/
 .`              `/

root@ce9a72483463
-----
OS: Arch Linux x86_64
Host: B365M GAMING HD
Kernel: 6.7.5-arch1-1
Uptime: 1 hour, 52 mins
Packages: 118 (pacman)
Shell: bash 5.2.26
Resolution: 1920x1080
CPU: Intel i5-9400F (6) @ 4.100GHz
GPU: NVIDIA GeForce GT 1030
Memory: 3166MiB / 7889MiB


```

```
[root@ce9a72483463 /]# pacman -S vim
resolving dependencies...
looking for conflicting packages...

Package (3)          New Version           Net Change  Download Size

core/gpm              1.20.7.r38.ge82d1a6-5   0.40 MiB    0.14 MiB
extra/vim-runtime     9.1.0080-1                  36.44 MiB   7.01 MiB
extra/vim              9.1.0080-1                  4.91 MiB   2.18 MiB

Total Download Size:  9.33 MiB
Total Installed Size: 41.75 MiB

:: Proceed with installation? [Y/n] y
:: Retrieving packages...
  vim-runtime-9.1.0080-1-x86_64 downloading...
  vim-9.1.0080-1-x86_64 downloading...
  gpm-1.20.7.r38.ge82d1a6-5-x86_64 downloading...
checking keyring...
checking package integrity...
loading package files...
checking for file conflicts...
:: Processing package changes...
installing vim-runtime...
Optional dependencies for vim-runtime
  sh: support for some tools and macros [installed]
  python: demoserver example tool
  gawk: mve tools upport [installed]
installing gpm...
installing vim...
Optional dependencies for vim
  python: Python language support
  ruby: Ruby language support
  lua: Lua language support
  perl: Perl language support
  tcl: Tcl language support
:: Running post-transaction hooks...
(1/2) Reloading system manager configuration...
  Skipped: Current root is not booted.
(2/2) Arming ConditionNeedsUpdate...
[root@ce9a72483463 /]#
```

```
[root@ce9a72483463 /]# vim --version
VIM - Vi IMproved 9.1 (2024 Jan 02, compiled Feb 06 2024 15:30:35)
Included patches: 1-80
Compiled by Arch Linux
Huge version without GUI. Features included (+) or not (-):
+acl          +file_in_path      +mouse_urxvt      -tag_any_white
+arabic       +find_in_path     +mouse_xterm      +tcl/dyn
+autocmd      +float           +multi_byte      +termguiccolors
+autochdir    +folding          +multi_lang      +terminal
-autoservername -footer          -mzscheme        +terminfo
-balloon_eval  +fork()          +netbeans_intg   +termresponse
+balloon eval term +gettext      +num64           +textobjects
```

6. Run a database container in the background. Then show logs of the running container. After, access the container in interactive mode. show some SQL queries inside the container.

For this we will be using a popular `NoSQL` database known as `MongoDB`. To pull the docker image from the remote to our local machine we can try this command,

```
docker pull mongo
```

```
[abir@ahammed-20101197] [~]
└ o docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
01007420e9b0: Pull complete
bc3bec6a423e: Pull complete
c5db81b694a8: Pull complete
427a1a117df0: Pull complete
dfb180c9e7b5: Pull complete
92e6f08e133c: Pull complete
374f042f3159: Pull complete
73549bb43006: Pull complete
Digest: sha256:5a54d0323fe207d15dc48773a7b9e7e519f83ad94a19c2ddac201d7aae109eb1
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest
[abir@ahammed-20101197] [~]
└ o
```

After that we can run the container in the background.

```
docker run --name some-mongo -d mongo
```

```
[abir@ahammed-20101197] [~]
└ o docker run --name some-mongo -d mongo
6b1d67424d0122910fd1999e81f7733aff440bca96eea475c943bc89ccc4c635
[abir@ahammed-20101197] [~]
└ o
```

To see the logs this command can be use,

```
docker logs some-mongo
```

```
[abir@ahammed-20101197] [-]
└─$ docker logs some-mongo
{"t":{"$date":"2024-02-20T22:40:22.848+00:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"main","msg":"Automatica
ocols 'none'"}
{"t":{"$date":"2024-02-20T22:40:22.849+00:00"},"s":"I", "c":"NETWORK", "id":4915701, "ctx":"main","msg":"Initialize
Version"0,"maxWireVersion":21}, "incomingInternalClient": {"minWireVersion":0,"maxWireVersion":21}, "outgoing": {"minW
{"t":{"$date":"2024-02-20T22:40:22.849+00:00"},"s":"I", "c":"NETWORK", "id":4648601, "ctx":"main","msg":"Implicit T
ver, tcpFastOpenClient, and tcpFastOpenQueueSize."}
{"t":{"$date":"2024-02-20T22:40:22.850+00:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"main","msg":"Successful
rService","namespace":"config.tenantMigrationDonors"}
{"t":{"$date":"2024-02-20T22:40:22.850+00:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"main","msg":"Successful
lientsService","namespace":"config.tenantMigrationRecipients"}
{"t":{"$date":"2024-02-20T22:40:22.850+00:00"},"s":"I", "c":"CONTROL", "id":5945603, "ctx":"main","msg":"Multi thre
{"t":{"$date":"2024-02-20T22:40:22.850+00:00"},"s":"I", "c":"TENANT_M", "id":7091600, "ctx":"main","msg":"Starting T
{"t":{"$date":"2024-02-20T22:40:22.851+00:00"},"s":"I", "c":"CONTROL", "id":4615611, "ctx":"initandlisten","msg":"M
ecture": "64-bit", "host": "91b22e9d17ab"}
{"t":{"$date":"2024-02-20T22:40:22.851+00:00"},"s":"I", "c":"CONTROL", "id":23403, "ctx":"initandlisten","msg":"B
84e314b497f282ea8aa06d7ded3eb205", "openSSLVersion": "OpenSSL 3.0.2 15 Mar 2022", "modules": [], "allocator": "tcmalloc", "e
86_64"}]}
{"t":{"$date":"2024-02-20T22:40:22.851+00:00"},"s":"I", "c":"CONTROL", "id":51765, "ctx":"initandlisten","msg": "O
{"t":{"$date":"2024-02-20T22:40:22.851+00:00"},"s":"I", "c":"CONTROL", "id":21951, "ctx":"initandlisten","msg": "O
{"t":{"$date":"2024-02-20T22:40:22.852+00:00"},"s":"I", "c":"STORAGE", "id":22297, "ctx":"initandlisten","msg": "U
age engine. See http://dochub.mongodb.org/core/prodnotes-filesystem". "tags": ["startUpWarnings"]}
```

To access the running database container in interactive mode we can use `exec` command.

```
docker exec -it some-mongo bash
```

Also, after accessing the `bash` shell inside the container we can use `mongosh` command to run the MongoDB Shell .

```
[abir@ahammed-20101197] [-]
└─$ docker exec -it some-mongo bash
root@6b1d67424d01:/# mongosh
Current Mongosh Log ID: 65d536a74660987d013941a7
Connecting to:      mongodbs://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.4
Using MongoDB:     7.0.5
Using Mongosh:    2.1.4

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy)
You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting
2024-02-20T23:31:50.163+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/wiredtiger-xfs
2024-02-20T23:31:50.735+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted.
2024-02-20T23:31:50.735+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never'
2024-02-20T23:31:50.735+00:00: vm.max_map_count is too low
-----

test>
```

Some queries inside the container

To see the version information of the current MongoDB use `db.version()` inside `mongosh` shell. Moreover, we can see all the databases list with `show dbs` and to create a new database we can use the `use <database_name>` command.

```
db.version()
show dbs
use docker_test
```

```
test> db.version()
7.0.5
test> show dbs
admin   40.00 KiB
config  12.00 KiB
local   40.00 KiB
test> use docker_test
switched to db docker_test
docker_test>
```

To create a new collection and insert some data inside the database we can use `insert` command.

```
db.products.insert( { item: "card", qty: 15 } )
```

Above command will create a collection named `products` and insert data `{ item: "card", qty: 15 }`.

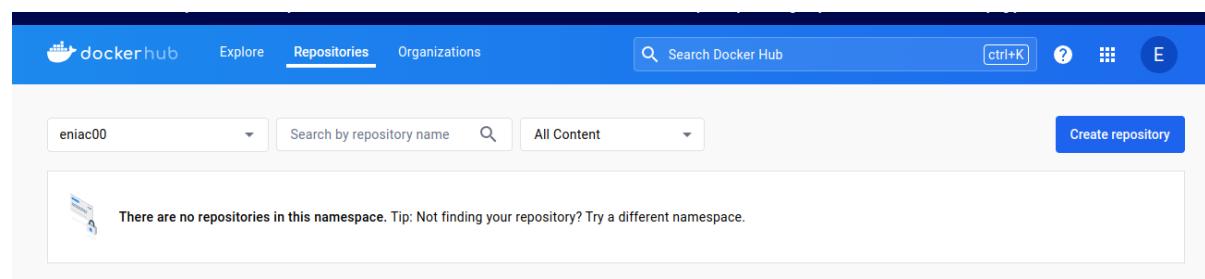
Moreover, we can use `find()` to list all the data present in a collection.

```
db.products.find().pretty()
```

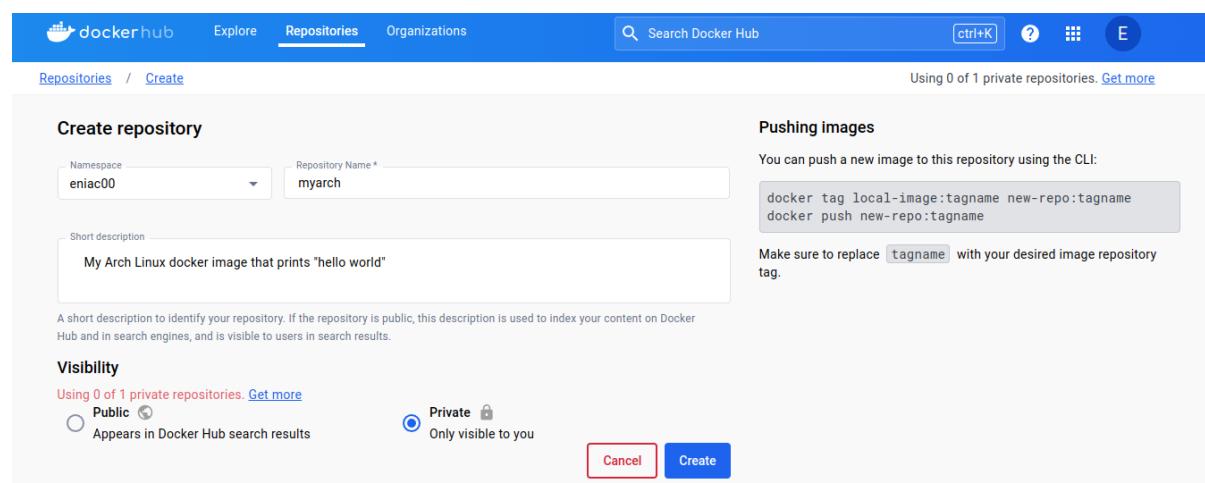
```
docker_test> db.products.insert( { item: "card", qty: 15 } )
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '_id': ObjectId('65d536fb4660987d013941a8') }
}
docker_test> db.products.find().pretty()
[
  { _id: ObjectId('65d536fb4660987d013941a8'), item: 'card', qty: 15 }
]
docker_test>
```

8. How to make your own private registry? Show steps.

We can create a new account in `Docker Hub` after login into our account we can select `Repositories` tab.



From there we can tap on the `Create repository` button to create a new repository. Here, we are creating a new repository `myarch`. Then click on `Create`.



This is how our new repository name `myarch` has been created. Note that, we have created a **Private repository**.

The screenshot shows the Docker Hub interface for a private repository. At the top, there are tabs for 'Explore', 'Repositories' (which is selected), 'Organizations', and search bar with 'ctrl+K'. Below the header, the repository path 'eniac00 / Repositories / myarch / General' is shown. A message indicates 'Using 1 of 1 private repositories. [Get more](#)'. The main content area shows the repository details: 'eniac00/myarch' (private), created less than a minute ago, and a description 'My Arch Linux docker image that prints "hello world"'. To the right, under 'Docker commands', it says 'To push a new tag to this repository:' followed by a button to run 'docker push eniac00/myarch:tagname'. Below this, the 'Tags' section is empty with a note: 'This repository is empty. Push some images to it to see them appear here.' The 'Automated Builds' section explains how to connect GitHub or Bitbucket for automatic builds, noting it's available with Pro, Team and Business subscriptions. A blue 'Upgrade' button is also present.

7. Push your own image into the Docker public registry/Hub.

For pushing our own image to our public/private repository we can use,

```
docker login -u <username>
```

Use password or token to authenticate your login.

A terminal window showing the execution of the `docker login` command. The user is prompted for a password, warned about storing credentials unencrypted, and then informed that the login succeeded. The terminal prompt is '[abir@ahammed-20101197] ~'.

```
[abir@ahammed-20101197] ~
└ o docker login -u eniac00
Password:
WARNING! Your password will be stored unencrypted in /home/abir/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[abir@ahammed-20101197] ~
```

We will push our newly created docker image `myarch`.

```
docker tag myarch:latest eniac00/myarch:latest
```

```
docker push eniac00/myarch:latest
```

```
[abir@ahammed-20101197] [-]
└ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
myarch          latest   4991cc415eaf  2 hours ago  436MB
mongo           latest   b8df2163f9aa  13 days ago  755MB
archlinux        latest   69f38d8f6347  7 weeks ago  436MB
hello-world     latest   d2c94e258dc9  9 months ago  13.3kB
[abir@ahammed-20101197] [-]
└ docker tag myarch:latest eniac00/myarch:latest
[abir@ahammed-20101197] [-]
└ docker push eniac00/myarch:latest
The push refers to repository [docker.io/eniac00/myarch]
d822bba4f032: Pushed
6de0fb710365: Pushed
latest: digest: sha256:9c049bad6e06f49c8d73274f294190260af7c3016550d8db35f30003c00219e1 size: 738
[abir@ahammed-20101197] [-]
└
```

In the browser we can see that our image has been successfully pushed to [Docker Hub](#).

The screenshot shows the Docker Hub interface for the repository `eniac00/myarch`. The repository was updated 1 minute ago and contains one tag, `latest`, which is an Arch Linux image type. In the Docker commands section, there is a button to push a new tag. The Automated Builds section indicates that manually pushing images to Hub connects to GitHub or Bitbucket to automatically build and tag new images whenever code is updated.

9. Create a small website or app with minimal functionality (Could be a simple HTML website that has a button that opens a static image/file) inside a Docker container. Then run the application (inside the container) in the background of your HOST machine in any port. Browse the website from your host machine.

First, create a new directory and create a webpage along with a docker file there.

```
[abir@ahammed-20101197] [~/webpage_docker]
└ ll
drwxr-xr-x  abir abir 4.0 KB Wed Feb 21 12:01:24 2024 .
drwxr-xr-x+ abir abir 4.0 KB Wed Feb 21 13:28:09 2024 ..
.rw-r--r--  abir abir 48 B  Wed Feb 21 12:01:17 2024 image-serve
.rw-r--r--  abir abir 1.8 KB Wed Feb 21 12:00:16 2024 index.html
[abir@ahammed-20101197] [~/webpage_docker]
```

Insert necessary codes in the `*.html` file and update docker file (i.e. `image-serve`) like below image and save.

The screenshot shows a terminal window with a black background and white text. At the top, it says "NORMAL > image-serve". Below that, the text "image-serve" is followed by "3L, 48B". The main content of the terminal is a Dockerfile:

```
1 FROM nginx:alpine
1
2 COPY . /usr/share/nginx/html
```

There are many blank lines between the first two lines of the Dockerfile, indicating that the file was copied from another location.

Next build the image from the docker file.

```
docker build -t image-serve -f image-serve .
```

The screenshot shows a terminal window with a black background and white text. It displays the output of the `docker build` command:

```
[abir@ahammed-20101197] [~/webpage_docker]
└─o docker build -t image-serve -f image-serve .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 4.608kB
Step 1/2 : FROM nginx:alpine
--> 6913ed9ec8d0
Step 2/2 : COPY . /usr/share/nginx/html
--> d8b5452dfa33
Successfully built d8b5452dfa33
Successfully tagged image-serve:latest
[abir@ahammed-20101197] [~/webpage_docker]
```

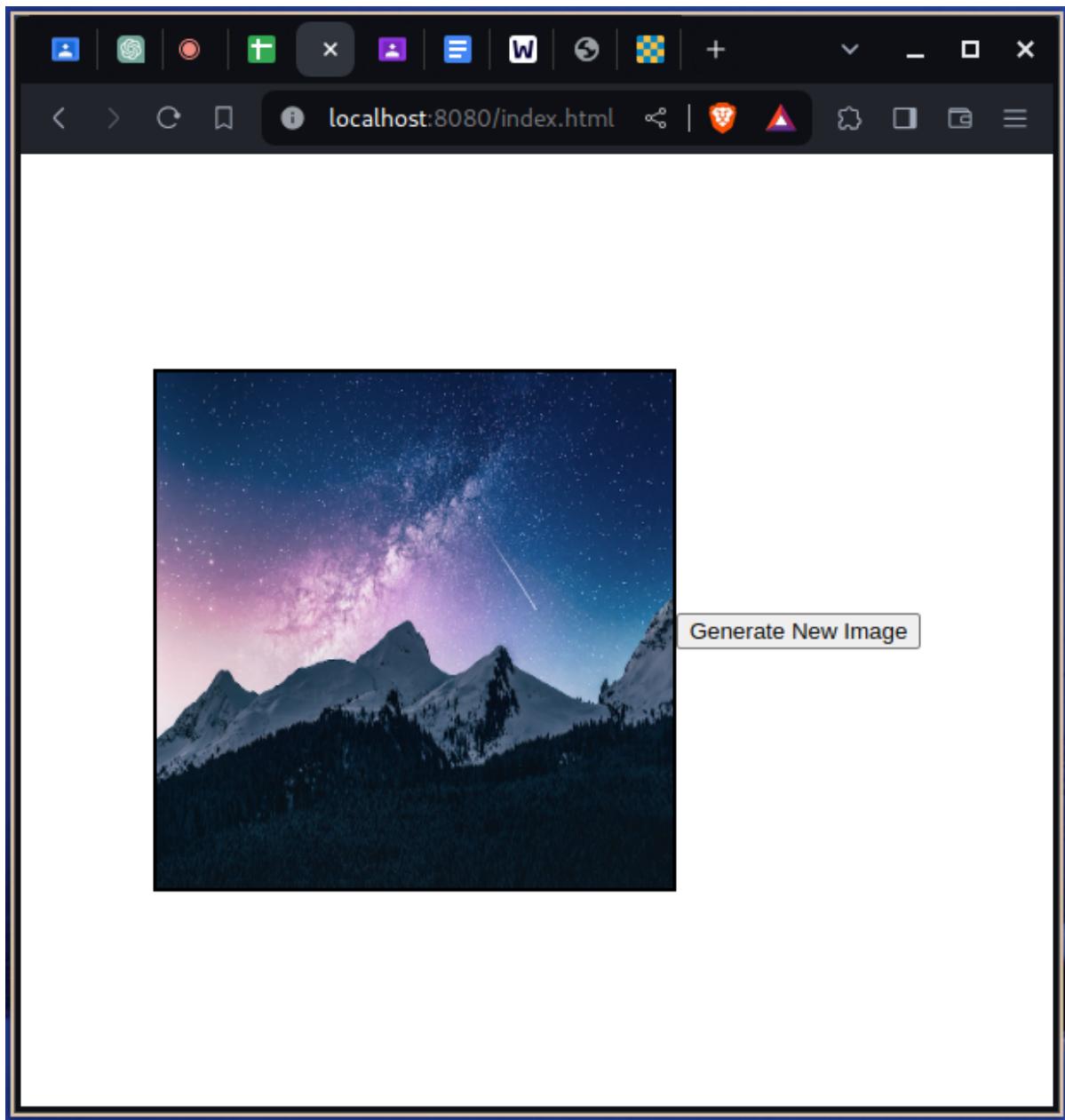
Now, run the docker container in the background and map containers `port 80` to host machine's `port 8080` with the below command.

```
docker run -d -p 8080:80 --name myserver image-serve
```

The screenshot shows a terminal window with a black background and white text. It displays the output of the `docker run` command and a table of running containers:

```
[abir@ahammed-20101197] [~/webpage_docker]
└─o docker run -d -p 8080:80 --name myserver image-serve
27bdf39819fce5452e678431ad3e52afdd90960be84798bc94e2b394eedd015c
[abir@ahammed-20101197] [~/webpage_docker]
└─o docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
27bdf39819fc image-serve "/docker-entrypoint...." 6 seconds ago Up 6 seconds 0.0.0.0:8080->80/tcp, :::8080->80/tcp myserver
[abir@ahammed-20101197] [~/webpage_docker]
```

We can visit the `localhost:8080/index.html` in any browser and our containerized webpage will be visible to us like the below image.



10. Migrate the new container having the application into another machine. Again run the container and browse the URL. It should work.

In my machine

To migrate a docker image we first we need to archive our image in a `*.tar` file. In this case we will be migrating our image `image-serve`, so we need to archive it.

```
docker save image-serve:latest > image_serve.tar
```

After that we can share that archived image in any machine. We will be using `scp` command to securely share the archived image over the secure shell (`ssh`).

```
scp image_serve.tar abir@192.168.0.111:~/docker_test/
```

```
[abir@ahammed-20101197] [-]
└─ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
image-serve     latest   db85452dfa33  About an hour ago  42.6MB
eniac00/myarch  latest   4991cc415eaf  9 hours ago   436MB
myarch          latest   4991cc415eaf  9 hours ago   436MB
nginx           alpine   6913ed9e98d0  6 days ago    42.6MB
monit           latest   b8df2163f9aa  2 weeks ago   755MB
archlinux        latest   69ff38d8f63a47 7 weeks ago   436MB
hello-world     latest   d2c94e25ddcb  9 months ago  13.3KB

[abir@ahammed-20101197] [-]
└─ docker save image-serve:latest > image_serve.tar
[abir@ahammed-20101197] [-]
└─ scp image_serve.tar abir@192.168.0.111:/~/docker_test/
abir@192.168.0.111's password:
image_serve.tar
[abir@ahammed-20101197] [-]

100% 42MB 9.8MB/s 00:04
```

In another machine

In another machine we can see that archived image. Now we just have to load the archived image using `load` command make sure that `docker` is installed in that machine.

```
docker load < image_serve.tar
```

We can run the newly created image (i.e. `image-serve`) using the above command we have seen in No. 9.

```
docker run -d -p 8080:80 --name myserver image-serve
```

```
[abir@nusaiba] [~/docker_test]
└─ ls
⑥ image_serve.tar
[abir@nusaiba] [~/docker_test]
└─ docker load < image_serve.tar
aedc3bda2944: Loading layer [=====] 7.63MB/7.63MB
718db50a47c0: Loading layer [=====] 5.426MB/5.426MB
3137f8f0c641: Loading layer [=====] 3.584kB/3.584kB
c5140fc719dd: Loading layer [=====] 4.608kB/4.608kB
9909978d630d: Loading layer [=====] 2.56kB/2.56kB
2593b08e5428: Loading layer [=====] 5.12kB/5.12kB
d8527026595f: Loading layer [=====] 7.168kB/7.168kB
667a247707f0: Loading layer [=====] 31.29MB/31.29MB
abc7a5ebdaaf: Loading layer [=====] 6.656kB/6.656kB
Loaded image: image-serve:latest
[abir@nusaiba] [~/docker_test]
└─ docker run -d -p 8080:80 --name myserver image-serve
0f6644b0f019d6de28ed5a9f2ad5db1fb40fb4e4c859b32eb23301920d455c55
[abir@nusaiba] [~/docker_test]
└─ docker ps -a
CONTAINER ID      IMAGE      COMMAND      CREATED      STATUS      PORTS
 NAMES
0f6644b0f019      image-serve      "/dock...er-entrypoint...."  5 seconds ago  Up 3 seconds      0.0.0.0:80->
myserver
6fd063b56c9f      hello-world      "/hello"      2 minutes ago  Exited (0) 2 minutes ago
sweet_beaver
[abir@nusaiba] [~/docker_test]
└─
```

Visit `localhost:8080/index.html` in any browser and we can see our webpage.

