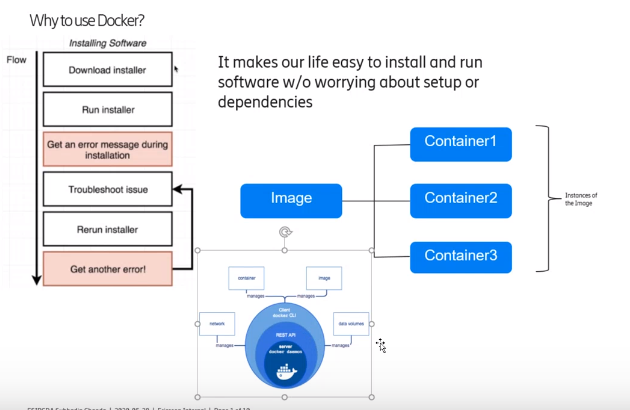
# Concept(s) to remember

1. Hypervisor is a kind of software by which we can run several virtual machines. Some of the hypervisor are
   * Virtual Box
   * VM Ware
   * Hyper-V (windows only)
2. Virtual machine vs Containers (all containers share same underlying kernel of OS, doesn’t need specific CPU, memory, very fast to start)
3. Docker architecture
   * Docker client
   * Docker Server (docker engine/docker demon)
   * Image (files + commands)
   * Container (process + ram + kernel …)

# Docker client server architecture



# Docker Workflow

## Create image

* + 1. Write docker file in location say "C:\repository\git\Docker"
    2. Build docker image from file

*C:\Users\echypal> docker build -t <hub-user>/<repo-name>[:<tag>] <docker file   
 directory location>*

If tag is not specified it is taken as latest

*C:\Users\echypal> docker build -t akash007ganga/hello-docker C:\repository\git\Docker*

* + 1. check image is available

*C:\Users\echypal>docker images*

*REPOSITORY TAG IMAGE ID CREATED SIZE*

*akash007ganga /hello-docker latest c3bd764e3654 24 seconds ago 113MB*

*nginx 1.14-alpine 8a2fb25a19f5 2 years ago 16MB*

*docker/whalesay latest 6b362a9f73eb 6 years ago 247MB*

* + 1. Create and start Containers from image.(it is a process with own file system provided by image)

*C:\Users\echypal>docker run akash007ganga/hello-docker*

*Hello Docker !*

*Or*

*C:\Users\echypal>docker create akash007ganga/hello-docker*

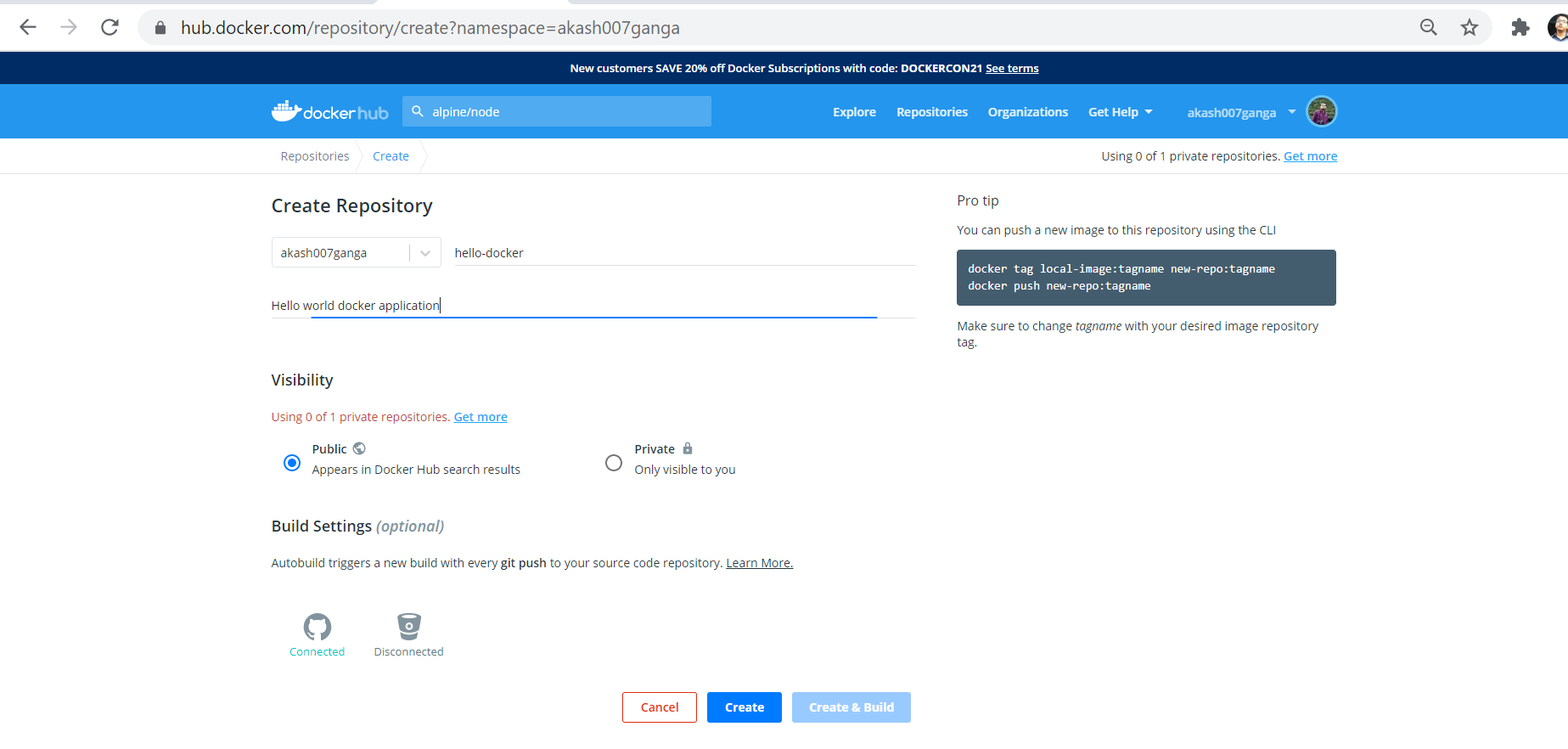
*<container id>*

*C:\Users\echypal>docker start <container id>*

*Hello Docker !*

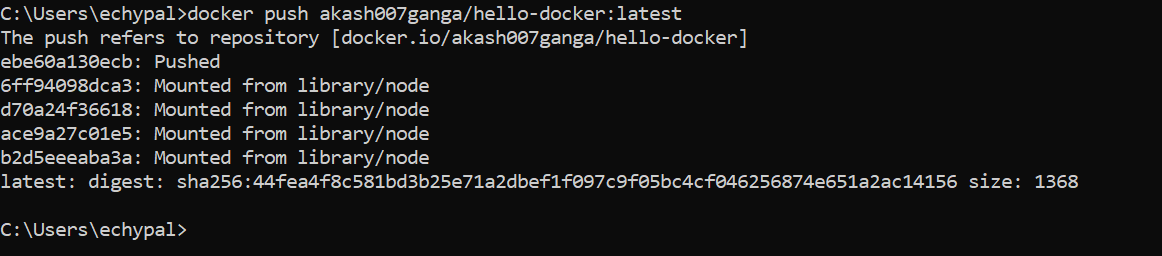
## Push the image into docker registry(like docker hub)

1. Create the repository in docker hub (hello-docker)

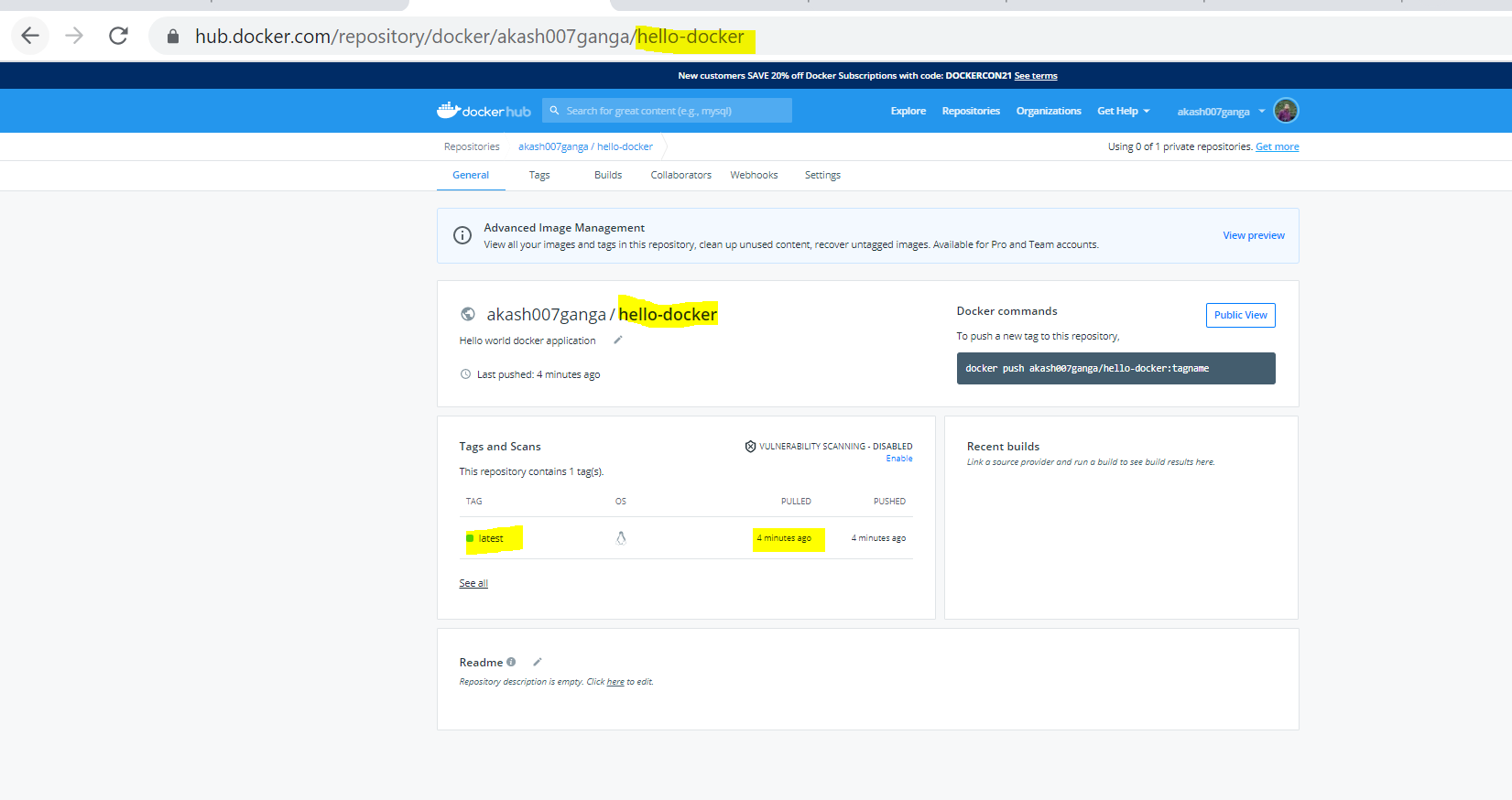


1. Push the image

*C:\Users\echypal>docker push akash007ganga/hello-docker:latest*

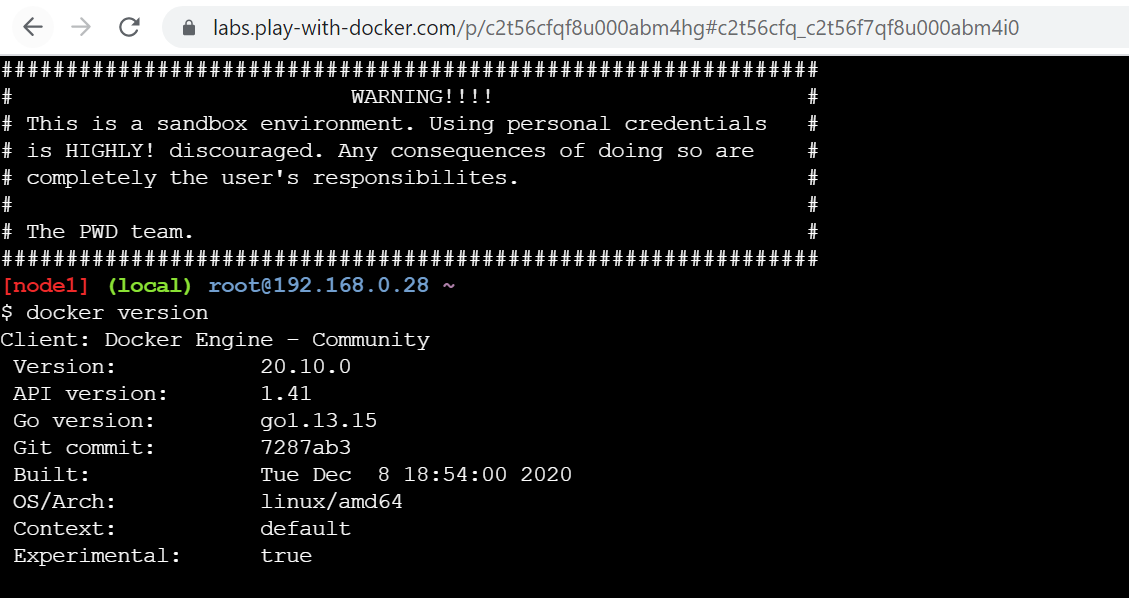


1. Check from docker hub

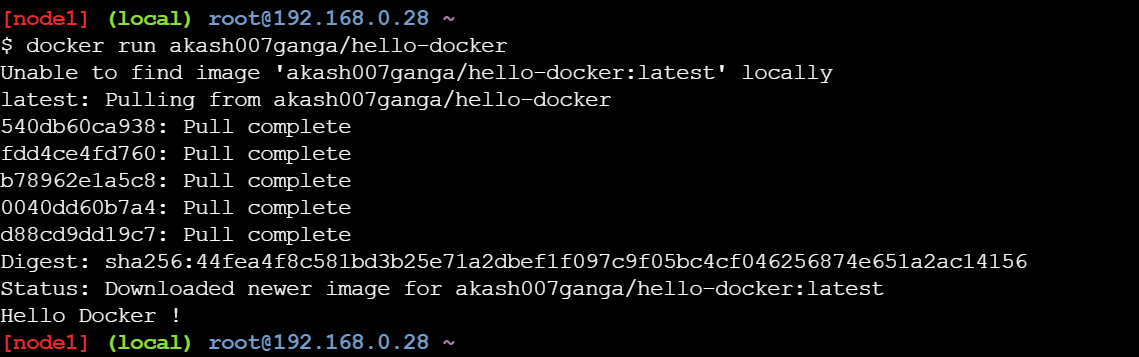


## Run the image into another machine

1. Go to a sandbox environment <https://labs.play-with-docker.com/>
2. Login with docker ID (if docker hub is logged in then it will automatically login)
3. Click on start
4. Click on add new instance to start a new VM
5. Maximize the console with alt + enter
6. Check docker version



1. Run our image



1. Check images



# Docker Commands

1. **Version**
2. C:\Users\echypal>docker --version

Docker version 19.03.13, build 4484c46d9d

1. C:\Users\echypal>docker version

Client: Docker Engine - Community

Cloud integration: 1.0.1

Version: 19.03.13

API version: 1.40

Go version: go1.13.15

Git commit: 4484c46d9d

Built: Wed Sep 16 17:00:27 2020

OS/Arch: windows/amd64

Experimental: false

Server: Docker Engine - Community

Engine:

Version: 19.03.13

API version: 1.40 (minimum version 1.12)

Go version: go1.13.15

Git commit: 4484c46d9d

Built: Wed Sep 16 17:07:04 2020

OS/Arch: linux/amd64

Experimental: false

containerd:

Version: v1.3.7

GitCommit: 8fba4e9a7d01810a393d5d25a3621dc101981175

runc:

Version: 1.0.0-rc10

GitCommit: dc9208a3303feef5b3839f4323d9beb36df0a9dd

docker-init:

Version: 0.18.0

GitCommit: fec3683

C:\Users\echypal>

1. **Run a docker image**

docker run docker/whalesay cowsay hello world (executing a command while running the container)

* Pull: docker pull docker/whalesay
* Create container from image :

C:\Users\echypal>**docker create docker**/whalesay

ff02a844536571325e501fcfb2f1151a499ca5eb4c6329e715651ebb60574a97

C:\Users\echypal>

* start it : docker start ff02a844536571325e501fcfb2f1151a499ca5eb4c6329e715651ebb60574a97
* and run startup command

Unable to find image 'docker/whalesay:latest' locally

latest: Pulling from docker/whalesay

Image docker.io/docker/whalesay:latest uses outdated schema1 manifest format. Please upgrade to a schema2 image for better future compatibility. More information at https://docs.docker.com/registry/spec/deprecated-schema-v1/

e190868d63f8: Pull complete

909cd34c6fd7: Pull complete

0b9bfabab7c1: Pull complete

a3ed95caeb02: Pull complete

00bf65475aba: Pull complete

c57b6bcc83e3: Pull complete

8978f6879e2f: Pull complete

8eed3712d2cf: Pull complete

Digest: sha256:178598e51a26abbc958b8a2e48825c90bc22e641de3d31e18aaf55f3258ba93b

Status: Downloaded newer image for docker/whalesay:latest

\_\_\_\_\_\_\_\_\_\_\_\_\_

< hello world >

-------------

\

\

\

## .

## ## ## ==

## ## ## ## ===

/""""""""""""""""\_\_\_/ ===

~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ / ===- ~~~

\\_\_\_\_\_\_ o \_\_/

\ \ \_\_/

\\_\_\_\_\\_\_\_\_\_\_/

C:\Users\echypal>

1. **Start/run an existing container**

* docker start <container-id>
* docker start -a <container-id> (show logs)

example: docker container for database cannot be deleted every time because data will also be lost

1. **list all running container**

C:\Users\echypal>docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

1. **See all the containers**

C:\Users\echypal>docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

26061ad10348 docker/whalesay "cowsay hello world" 5 minutes ago Exited (0) 5 minutes ago sleepy\_varahamihira

C:\Users\echypal>

1. **Stop a container**

* C:\Users\echypal>docker stop 26061ad10348 (container id/names)

26061ad10348

* Docker kill <container id>

1. **remove a container**

* C:\Users\echypal>docker rm sleepy\_varahamihira (container id/names)

sleepy\_varahamihira

* Check after remove

C:\Users\echypal>docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

C:\Users\echypal>

* all container

docker rm $(docker ps -a -q )

note: q gives unique id's

1. **List available images**

* C:\Users\echypal>docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

hello-world latest bf756fb1ae65 10 months ago 13.3kB

docker/whalesay latest 6b362a9f73eb 5 years ago 247MB

* C:\Users\echypal> docker image ls

* unique images

docker images -q

* All images including intermediate images: docker images -a

1. **Remove doker images**

* docker rmi docker/whalesay

* forced

docker rmi mysql -f (forced. needed when conflict: unable to delete 6e447ce4863d (must be forced) - image is referenced in multiple repositories)

* remove all images (with forced)

docker rmi -f $(docker images -q)

p.s.: all the dependent containers should be deleted before deleting the docker image

1. **Run command inside a running container**

C:\Users\echypal>docker exec -it 28f13a42fb72 sh

/opt # ls

Dockerfile app.py requirements.txt templates

/opt #

1. **Run a container with the nginx:1.14-alpine image and name it webapp**

docker run --name webapp nginx:1.14-alpine

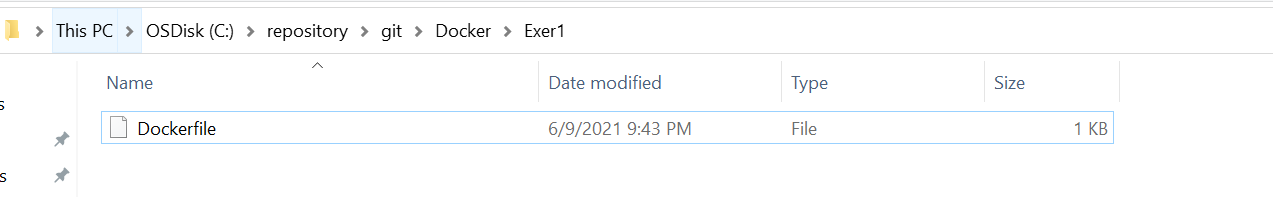
# Docker Exercise -1

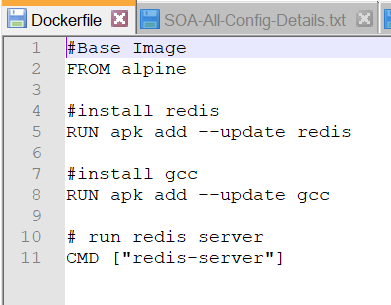
## Problem Statement

* + Create a docker image using docker file which has the following

1. Take a base linux image like alpine/busybox
2. Install redis on top of it
3. Install gcc on it
4. Set your startup command to run redis server
   * Build an image from above docker file and name it akash007ganga/image-exer1
   * Check the steps docker performed to do the above job
   * Check intermediate images created from container.
   * Create a container, container-1 from that image.
   * Create a folder, folder-1 inside the container container-1.
   * Create another container, container-2 from that image.
   * Check that folder-1 is not inside container-2
   * Find difference of container-2 and container-1 from their base container(created from image)
   * Change some steps in the docker file and prove that only steps beyond the change point will be reworked.
   * Remove all the intermediate images

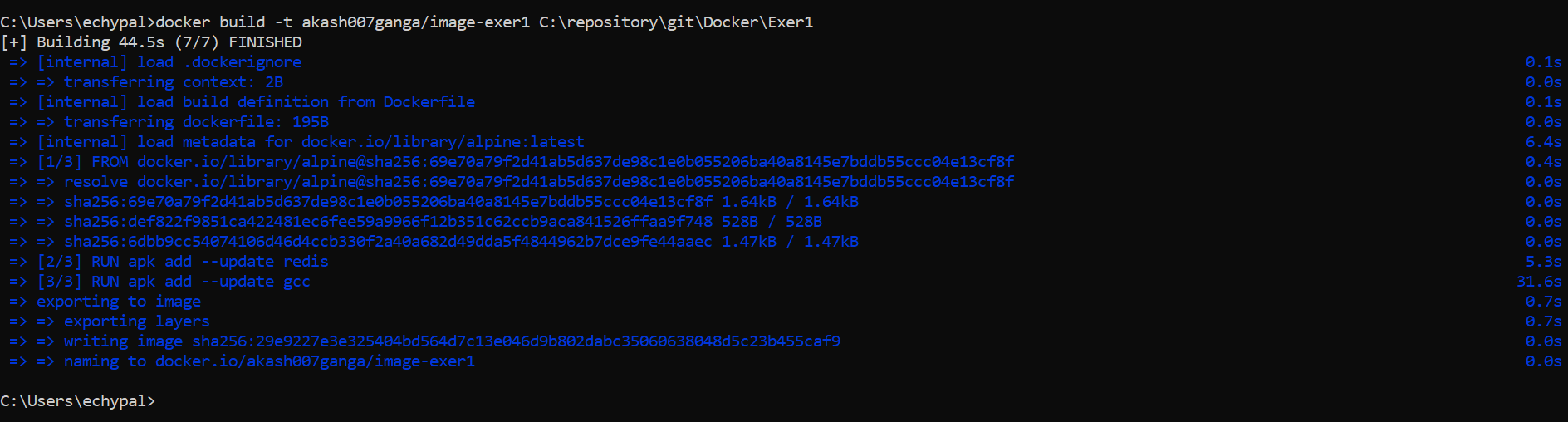
## Prepare Docker File

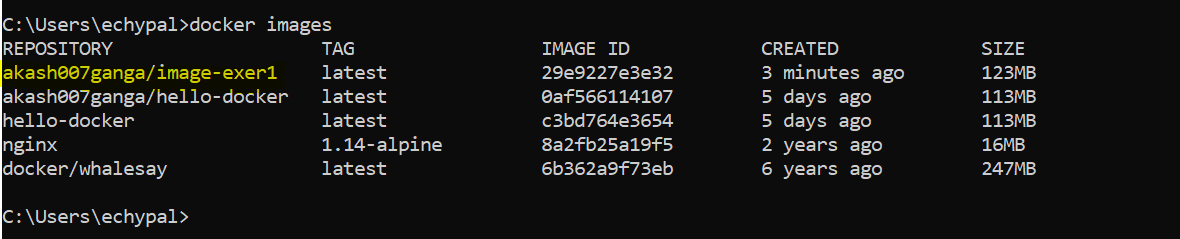




## Create the image

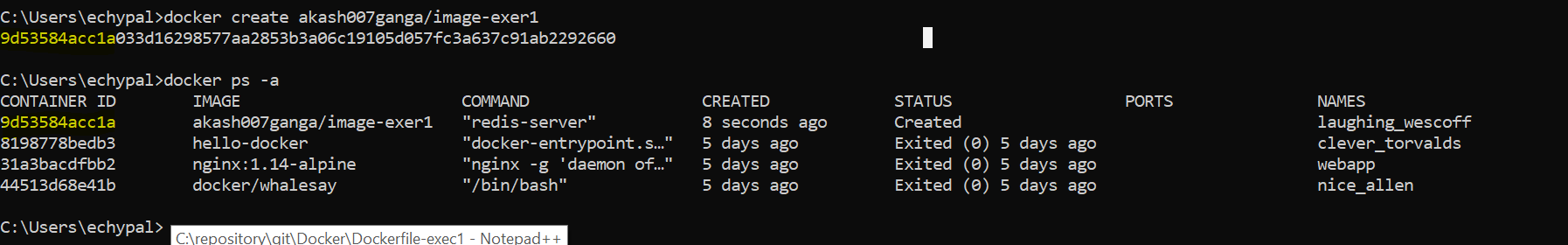
*C:\Users\echypal> docker build -t <hub-user>/<repo-name>[:<tag>] <docker file   
 directory location>*

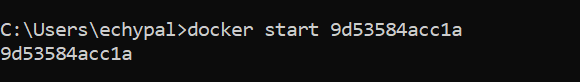


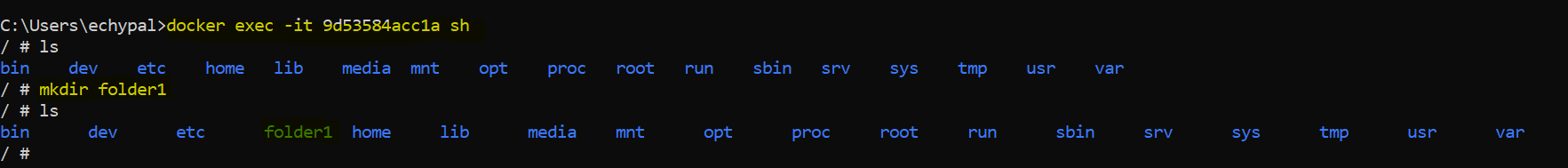


In current docker version only final image is created. If we change the docker file, then earlier image will be tagged as none and new image will be given name as akash007ganga/image-exer1.

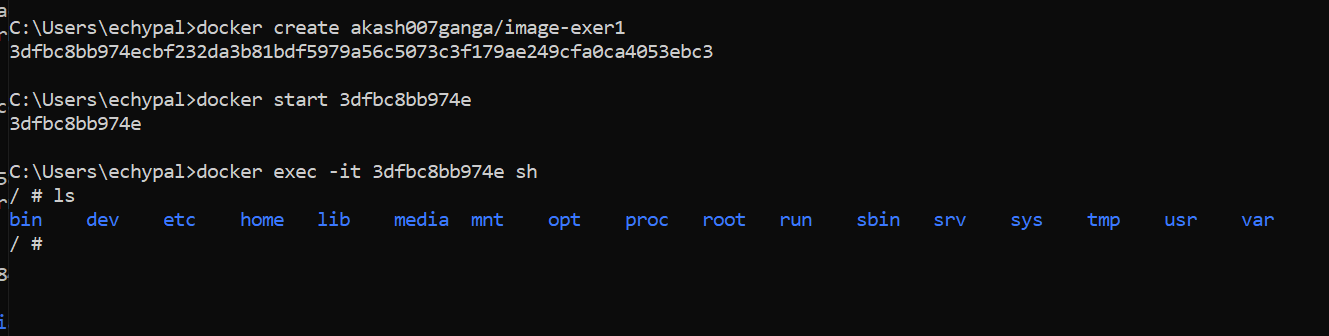
## Create container (container-1) from image, start it and create a folder





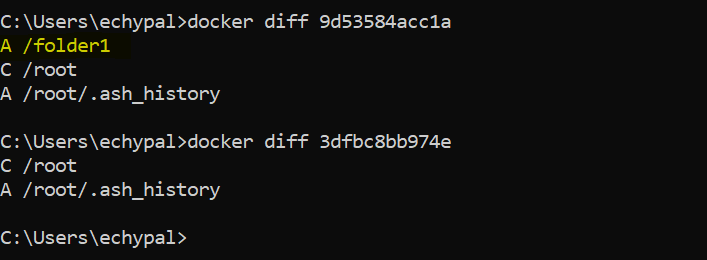


## Create container (container-2) from image, start it



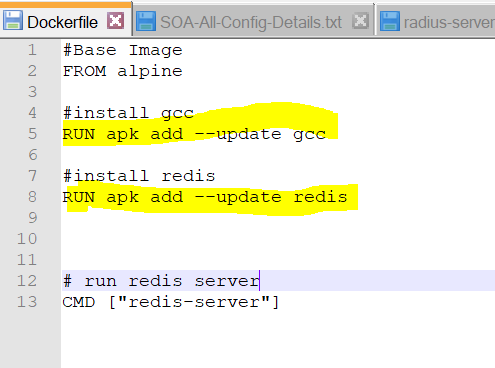
folder-1 is not present in container-2

## Difference of Containers from their base version

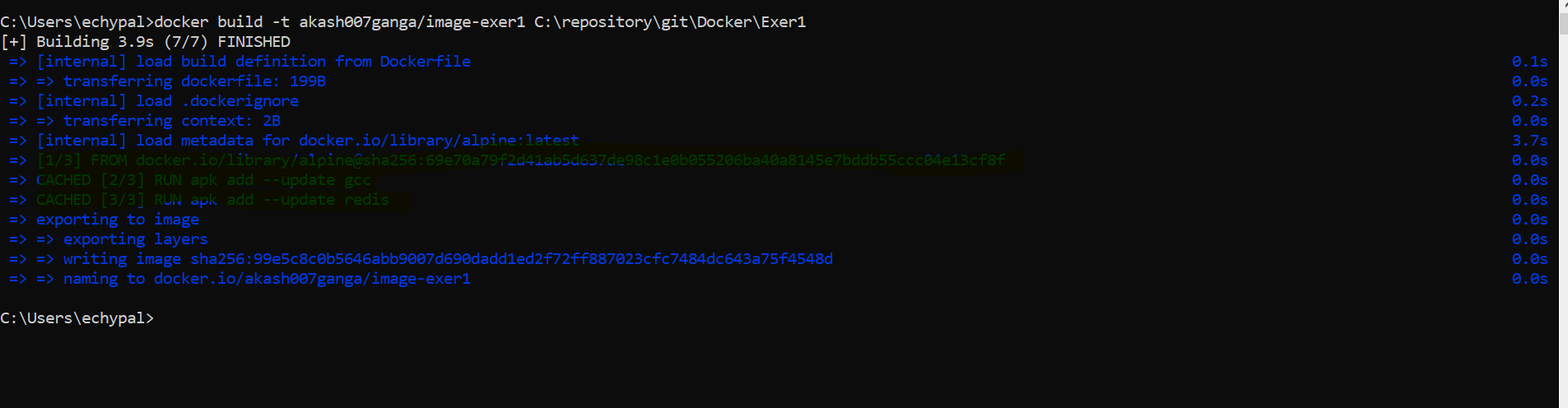


In Container-1, folder-1 was created

## Interchange some steps in Dockerfile and build



(Installed gcc before radis)



It used the cache. Nevertheless, we should write the docker file in such a way that big steps are done first which are not changed frequently.

# Docker Exercise -2

## Problem Statement

* + Create one mysql container from corresponding image
  + Run the container. Forward the port while creating the container to local machine
  + Change the root password of thee mysql running instance.
  + Create one table.
  + Insert some data in it.
  + Check from local machine if data can be seen or not.
  + Create one normal Spring Boot application which can use the above database and fetch record.
  + Containerize the Spring Boot REST Application and make docker image
  + Create container from image, forward port to local machine
  + Do REST call from local machine to application(query some data to application which in turn will query the database and show).