Image Creation and Sharing

Agenda

- Base Image
- Docker File
- Working with containers
- Optimization of Docker File
- Publishing Image on Docker Hub
- Private Registry

BASE IMAGE

What is Base Image?

- Docker base image is the basic image on which you add layers and create a final image containing your App.
- Docker keeps track of the difference between the base image and the new image by creating a new image layer using the union filesystem being used.
- For example, In order to run a LAMP stack as a Docker container any of the Linux OS(Ubuntu 14.0, CentOS 7, etc) is used as a base image. Apache, MySQL and PHP are installed over the base image, which results in the final LAMP docker image which can be executed as a container

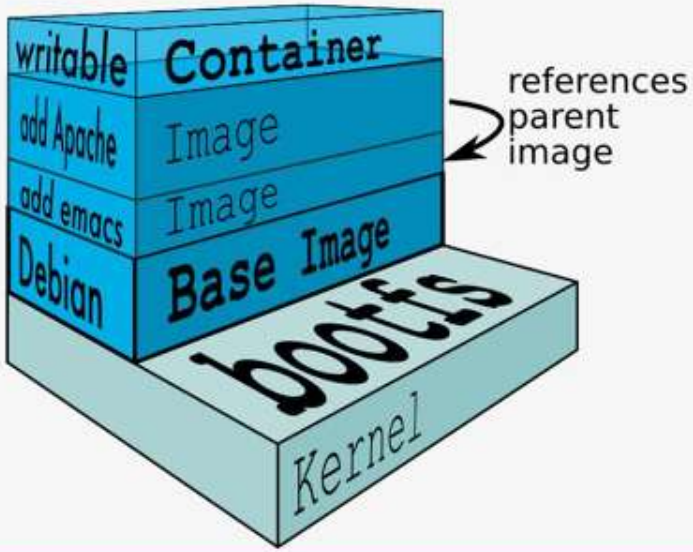


Image Layer

- Images are comprised of multiple layers
- Every image contains a base layer
- Docker uses a copy on write system
- Layers are just read only image

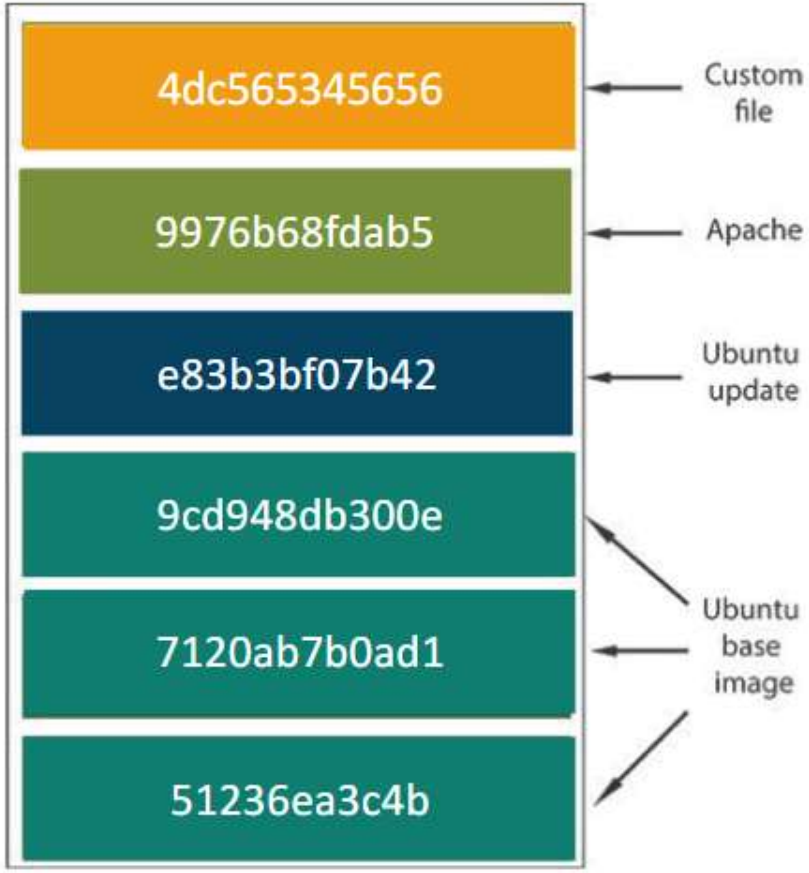


Image Selection – Base Image

- When creating your own images, you will need to decide which base image to start from
- The best-case scenario is that you don't need to create an image at all; you can just use an existing one and mount your configuration files and/or data into it
- This is likely to be the case for common application software, such as databases and web servers, where there are official images available
- In general, you are much better off using an official image than rolling your own

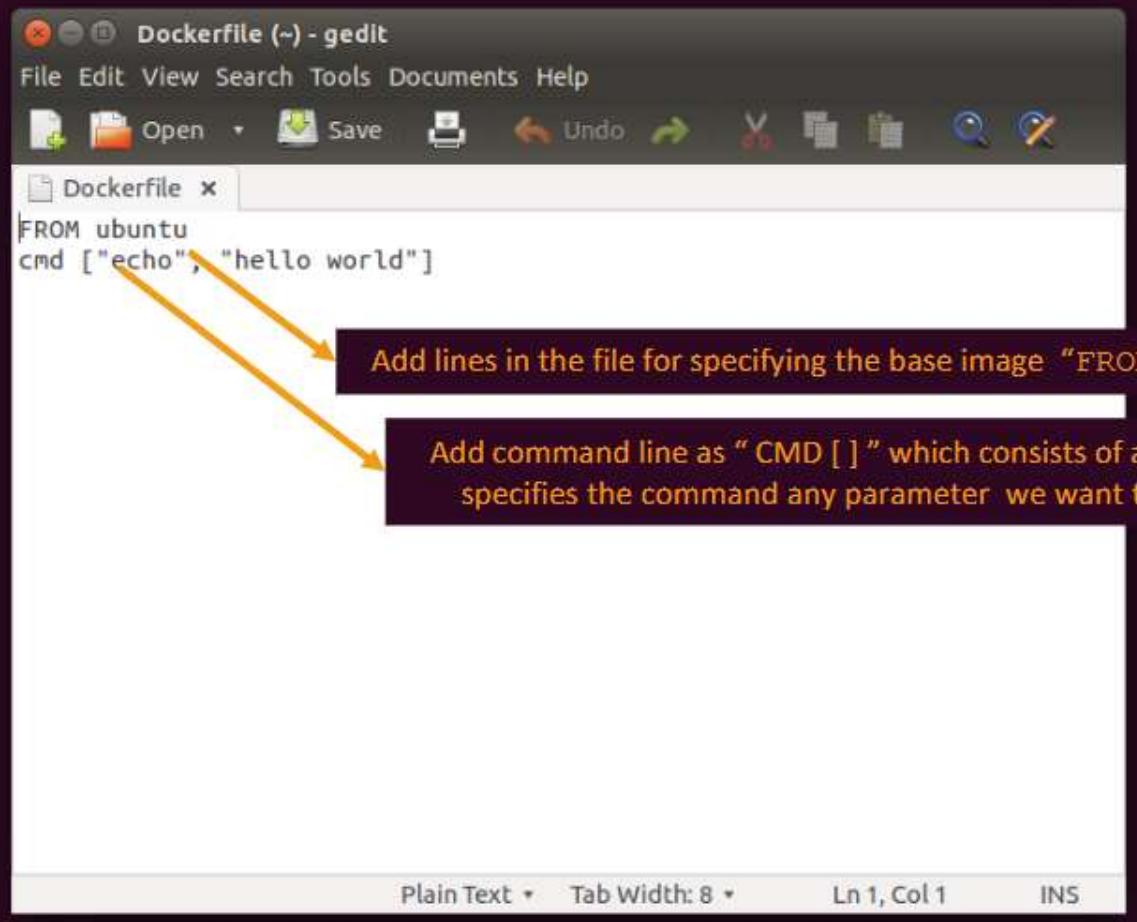
Building a Docker Image

- Step 1: Create an empty file with any Filename "`$ touch <FILE_NAME>`" and open it in editing mode using "`$ gedit <FILE_NAME>`"
- Step 2: Editing the File
- Step 3: Build the file using the command < `docker build [OPTIONS] PATH | URL |` "
- Step 4: Finally run the file using "`docker run`"

Step 1: Creating an empty file

```
root@test01:~# touch Dockerfile
root@test01:~# gedit Dockerfile
```

Step 2: Editing the File



Add lines in the file for specifying the base image "FROM <base_image>"

Add command line as " CMD [ ] " which consists of an array that specifies the command any parameter we want to execute

Step 3: Building Docker File

```
root@test01:~# gedit Dockerfile
root@test01:~# docker build .
Sending build context to Docker daemon   513 kB
Step 1 : FROM ubuntu
 ---> 4ca3a192ff2a
Step 2 : CMD echo hello world
 ---> Running in 876177664b4f
 ---> 7c226dc91bb2
Removing intermediate container 876177664b4f
Successfully built 7c226dc91bb2
root@test01:~#
```

Running hello-world in intermediate container

Final image ID

Step 3: Building Docker File contd…

```
root@test01:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
<none>              <none>              7c226dc91bb2        8 minutes ago       128.2 MB
hello-world         latest              48b5124b2768        3 months ago        1.84 kB
ubuntu              latest              4ca3a192ff2a        4 months ago        128.2 MB
root@test01:~#
```

Step 4: Run the File

```
root@test01:~# docker run --name test 7c226dc91bb2
hello world
root@test01:~#
```

Running docker file with name “test”

DOCKER FILE

Docker File

- Docker file is the basic building block of Docker containers
- Docker file is a file with a set of instructions and forms the basis for any Docker Image.
- Every time, base image is going to be based upon another image. You are going to pick up a base image and build up on that image.

Docker File: Main Section

1. FROM & MAINTAINER
2. UPDATE / UPGRADE
3. Environment Variable
4. EXPOSE port
5. CMD

FROM:

- Every Docker file starts with this command
- It shows where is the base image coming from
- Will pick up an image from Docker hub or some other repository and make some changes for ex: environmental changes, or expose your ports etc. and then save the file.
- Example:

```
FROM ubuntu:latest
```

Docker File: Main Section

1. FROM & MAINTAINER
2. UPDATE / UPGRADE
3. Environment Variable
4. EXPOSE port
5. CMD

MAINTAINER:

- The section of the Docker file shows the maintainer or the owner of the Docker file
- It requires certain format – It requires the name and the email id
- Following is the format:
  - `MAINTAINER  name <email id >`
- Example:

```
FROM ubuntu:latest
MAINTAINER edurekaDocker atul.harsha@edureka.co
```

Docker File: Main Section

1. FROM & MAINTAINER
2. UPDATE / UPGRADE
3. Environment Variable
4. EXPOSE port
5. CMD

- Set of actions you want to perform on the base image, where the modification of the base image starts.
- These actions have to be performed with root images

```
FROM ubuntu:latest
MAINTAINER edurekaDocker atul.harsha@edureka.co
RUN apt -get update
RUN apt -get upgrade
```

Docker File: Main Section

1. FROM & MAINTAINER
2. UPDATE / UPGRADE
3. Environment Variable
4. EXPOSE port
5. CMD

- Environment Variables in docker are declared with 'ENV' statement
- Environment variables are notated in Dockerfile as $variable_name or ${variable_name}
- Set up the environment variable and pass a variable that we need to pass inside the container that runs on base image eg: `ENV MYVALUE Edureka-Test`
- When you run the container this value will have to be passed using "echo $MYVALUE"

```
FROM ubuntu:latest
MAINTAINER edurekaDocker atul.harsha@edureka.co
RUN apt -get update
RUN apt -get upgrade
ENV MYVALUE Edureka-Test
```

Docker File: Main Section

1. FROM & MAINTAINER
2. UPDATE / UPGRADE
3. Environment Variable
4. EXPOSE port
5. CMD

- EXPOSE: Let the service in the container is not accessible from outside Docker, but from inside other Docker containers.
- It is good for inter-container communication.
- Ports are set up in the Docker file to be exposed.
- When you run docker ps for this container you will see the information for the ports which are exposed

```
FROM ubuntu:latest
MAINTAINER edurekaDocker atul.harsha@edureka.co
RUN apt -get update
RUN apt -get upgrade
ENV MYVALUE Edureka-Test
EXPOSE 80
EXPOSE 24
```

Docker File: Main Section

1. FROM & MAINTAINER
2. UPDATE / UPGRADE
3. Environment Variable
4. EXPOSE port
5. CMD

- Command for starting up of a service of some kind
- Anything that is after a command is a list of things to run within any container that is initiated on a base image
- All the actions to run when the containers are initiated is described in this section
- Following format is required to set up this directive: CMD ["/bin/bash"]

```
FROM ubuntu:latest
MAINTAINER edurekaDocker atul.harsha@edureka.co
RUN apt -get update
RUN apt -get upgrade
ENV MYVALUE Edureka-Test
EXPOSE 80
EXPOSE 24
CMD  ["/bin/bash"]
```

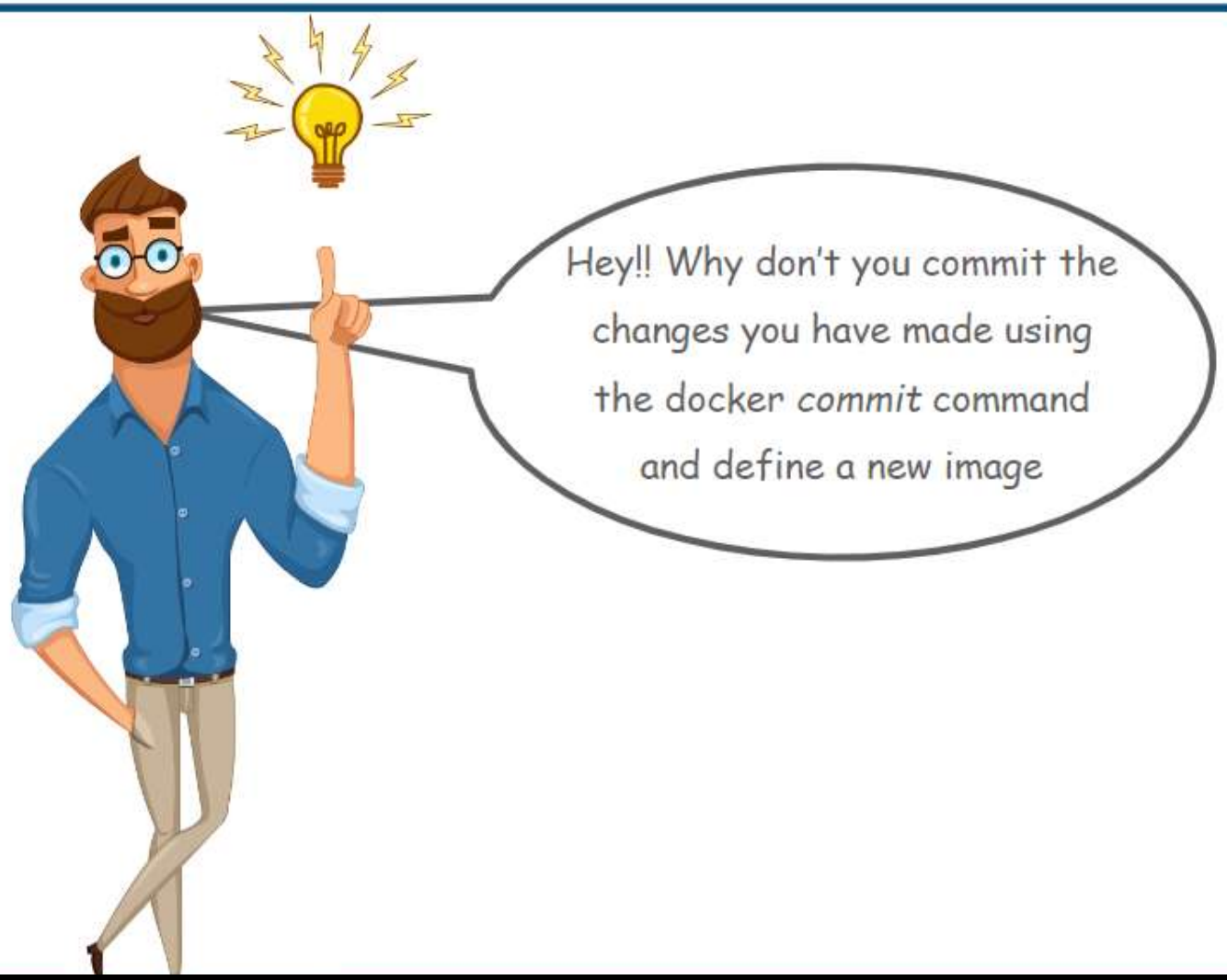
Docker RUN vs CMD

- RUN executes command(s) in a new layer and creates a new image. It is often used for installing software packages.
- CMD sets default command and/or parameters, which can be overwritten from command line when docker container runs.

WORKING WITH CONTAINER

Docker Problem I

Docker Solution I



Solution: docker-commit

```
root@edureka:~# docker run -it ubuntu:14.04 /bin/bash
root@2fa250ffb93b:/# apt update
Ign http://archive.ubuntu.com trusty InRelease
Get:1 http://archive.ubuntu.com trusty-updates InRelease [65.9 kB]
Get:2 http://archive.ubuntu.com trusty-security InRelease [65.9 kB]
Get:3 http://archive.ubuntu.com trusty Release.gpg [933 B]
Get:4 http://archive.ubuntu.com trusty-updates/main Sources [491 kB]
Get:5 http://archive.ubuntu.com trusty-updates/restricted Sources [6467 B]
Get:6 http://archive.ubuntu.com trusty-updates/universe Sources [226 kB]
Get:7 http://archive.ubuntu.com trusty-updates/main amd64 Packages [1226 kB]
Get:8 http://archive.ubuntu.com trusty-updates/restricted amd64 Packages [21.2
kB]
```

Its safe to remove the stopped container and start new ones based on the ubuntu:update

Solution: docker-commit contd…

```
root@edureka:~# docker commit 2fa250ffb93b ubuntu:update
sha256:e41846db43d7f367505c68810db2965b355570a20c839df5b47e8ac85d1514d7
root@edureka:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
 STATUS              PORTS               NAMES
root@edureka:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
ubuntu              update              e41846db43d7        2 minutes ago
 210.5 MB
<none>              <none>              fb74b3578bd3        20 hours ago
 188 MB
ubuntu              14.04               302fa07d8117        3 weeks ago
 188 MB
hello-world         latest              48b5124b2768        3 months ago
 1.84 kB
root@edureka:~#
```

Its safe to remove the stopped container and start new ones based on the ubuntu:update image

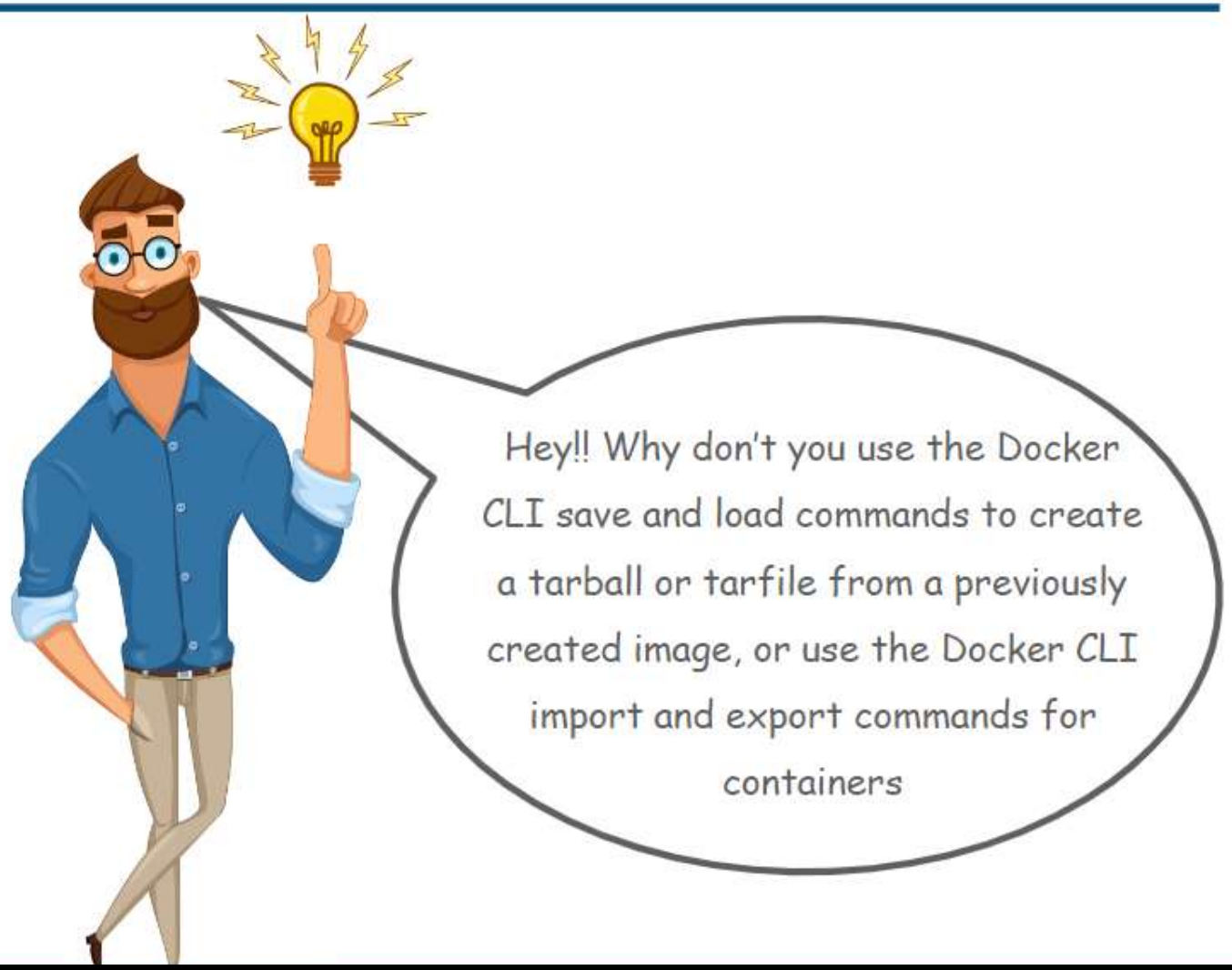
View Changes using docker-diff

- To view the changes made in the image use the command: `docker diff <container-id>`

```
root@edureka:~# docker diff 2fa250ffb93b
C /root
A /root/.bash_history
C /tmp
C /var
C /var/cache
C /var/cache/apt
D /var/cache/apt/pkgcache.bin
D /var/cache/apt/srcpkgcache.bin
C /var/lib
C /var/lib/apt
C /var/lib/apt/lists
A /var/lib/apt/lists/archive.ubuntu.com_ubuntu_dists_trusty-security_InRelease
A /var/lib/apt/lists/archive.ubuntu.com_ubuntu_dists_trusty-security_main_binar
y-amd64_Packages.gz
A /var/lib/apt/lists/archive.ubuntu.com_ubuntu_dists_trusty-security_main_sourc
```

Docker Problem II

Docker Solution II



Exporting a Container

```
root@edureka:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED
   STATUS                          PORTS                   NAMES
2fa250ffb93b        ubuntu:14.04        "/bin/bash"              About an hour ago
   Exited (0) About an hour ago                                thirsty_mirzakhani
c1803ff982ca        ubuntu:14.04        "/bin/bash"              22 hours ago
   Exited (127) 22 hours ago                                   thirsty_perlman
f76c9b9f4a41        hello-world         "/hello"                 8 days ago
   Exited (0) 8 days ago                                       cocky_morse
15cde3dab21c        hello-world         "tail -f /dev/null"      8 days ago
   Created                                                     happy_hawking
6206c23a6b92        hello-world         "/hello"                 8 days ago
   Exited (0) 8 days ago                                       thirsty_bartik
root@edureka:~# docker export 2fa250ffb93b > update.tar
root@edureka:~# ls
update.tar  WordPress
root@edureka:~#
```

Export a stopped container into a tarball/ tarfile

Importing a Container

```
root@edureka: ~
root@edureka:~# docker import - update < update.tar
sha256:6f174220363d05aae6ba0c5baa63d18ea1a3c891014fc4d1
root@edureka:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREA
 SIZE
update              latest              6f174220363d        24 seconds ago
 187.3 MB
ubuntu              update              e41846db43d7        About an hour ago
 210.5 MB
<none>              <none>              fb74b3578bd3        22 hours ago
 188 MB
ubuntu              14.04               302fa07d8117        3 weeks ago
 188 MB
hello-world         latest              48b5124b2768        3 months ago
 1.84 kB
root@edureka:~#
```

Upload the tar file on a web server and let your collaborator download it and use the import command on his Docker host

Save Command

```
root@edureka:~# docker save -o update1.tar update
root@edureka:~# ls -l
total 383564
-rw------- 1 root root 196383744 May  5 17:26 update1.tar
-rw-r--r-- 1 root root 196375552 May  5 16:07 update.ta
drwxr-xr-x 2 root root      4096 May  3 19:52 WordPr
root@edureka:~# docker rmi update
Untagged: update:latest
Deleted: sha256:6f174220363d05aae66a0c5baa63d18eafa
e
Deleted: sha256:dea4ebd431871a1550ce3f9a593ea2c9e378622
0
root@edureka:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
ubuntu              update              e41846db43d7        2 hours ago
 210.5 MB
<none>              <none>              fb74b3578bd3        23 hours ago
 188 MB
ubuntu              14.04               302fa07d8117        3 weeks ago
 188 MB
hello-world         latest              48b5124b2768        3 months ago
 1.84 kB
```

If you would rather deal with images that you have already committed, you can use the load and save commands

Load Command

```
root@edureka:~# docker load < update1.tar
dea4ebd43187: Loading layer 196.4 MB/196.4 MB
Loaded image: update:latest
root@edureka:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
update              latest              6f174220363d        About an hour ago
 187.3 MB
ubuntu              update              e41846db43d7        2 hours ago
 210.5 MB
<none>              <none>              fb74b3578bd3        23 hours ago
 188 MB
ubuntu              14.04               302fa07d8117        3 weeks ago
 188 MB
hello-world         latest              48b5124b2768        3 months ago
 1.84 kB
```

Optimising a Docker File

Best Practices

- Run a single process per container
- Containers should be ephemeral, that is
- Use a *.dockerignore* file
- Use official images from Docker Hub instead of writing your own from scratch
- Finally, minimize the number of layers of your images

Versioning an Image with Tags

- In order to keep track while creating multiple image and multiple version of same image Tags can be used instead of image ID
- Tag command allows to rename an existing image, or create a new tag for the same name.

Versioning an Image with Tags contd…

```
root@test01:~# docker images
REPOSITORY          TAG                 IMAGE ID            CRE
ATED             SIZE
flask               latest              7c226dc91bb2        12
days ago         128.2 MB
busybox             latest              00f017a8c2a6        8 w
eeks ago         1.11 MB
<none>              <none>              48b5124b2768        3 m
onths ago        1.84 kB
ubuntu              latest              4ca3a192ff2a        5 m
onths ago        128.2 MB
root@test01:~# docker tag ubuntu foobar
root@test01:~# docker images
REPOSITORY          TAG                 IMAGE ID            CRE
ATED             SIZE
flask               latest              7c226dc91bb2        12
days ago         128.2 MB
busybox             latest              00f017a8c2a6        8 w
eeks ago         1.11 MB
<none>              <none>              48b5124b2768        3 m
onths ago        1.84 kB
foobar              latest              4ca3a192ff2a        5 m
onths ago        128.2 MB
ubuntu              latest              4ca3a192ff2a        5 m
onths ago        128.2 MB
```

Versioning an Image with Tags contd…

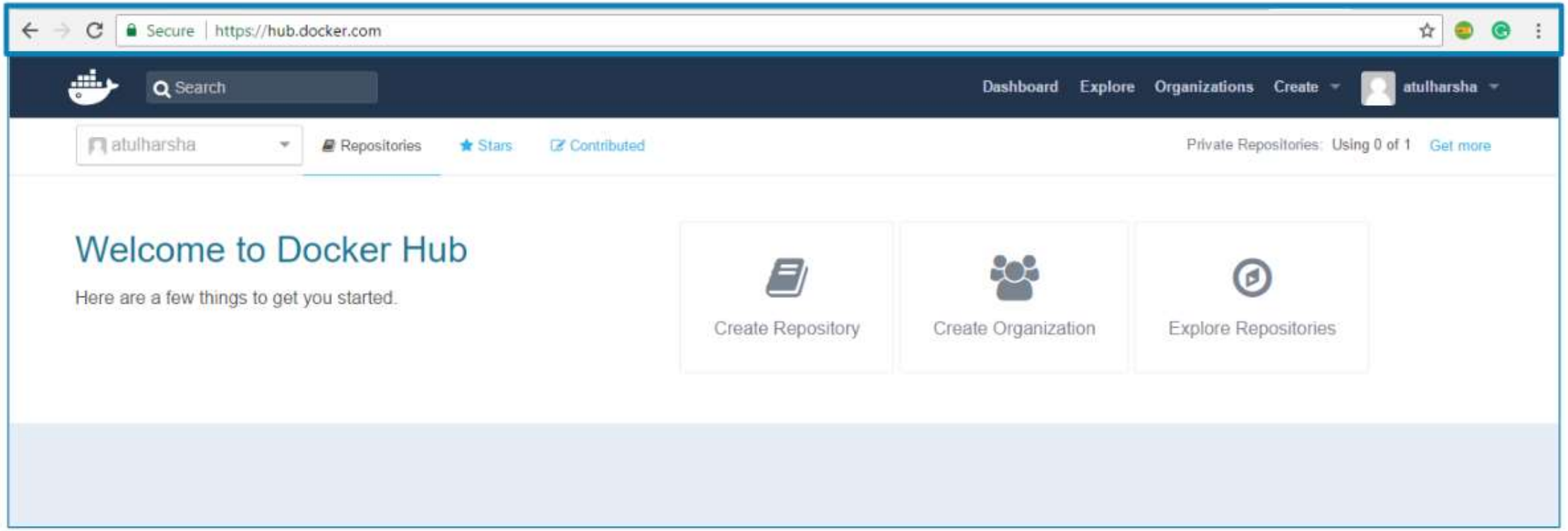
```
root@test01:~# docker tag ubuntu foobar:sample
root@test01:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
flask               latest              7c226dc91bb2        12 days ago
128.2 MB
busybox             latest              00f017a8c2a6        8 weeks ago
1.11 MB
<none>              <none>              48b5124b2768        3 months ago
1.84 kB
foobar              latest              4ca3a192ff2a        5 months ago
128.2 MB
foobar              sample              4ca3a192ff2a        5 months ago
128.2 MB
ubuntu              latest              4ca3a192ff2a        5 months ago
128.2 MB
root@test01:~#
```

Publishing on Docker Hub

Docker Hub

- Docker Hub is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker Cloud so you can deploy images to your hosts.
- Publish/Share an image on the docker hub using the following steps:
- Create an account on the Docker Hub
- Login into the hub from the Docker Host
- Push your image

Docker Hub Web Page



Docker Web Page

Docker Hub Login

```
root@docker:~# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, he
ad over to https://hub.docker.com to create one.
Username: atulharsha
Password:
Login Succeeded
root@docker:~#
```

Docker Hub Push – docker commit

Create a new image from a container's changes

```
root@docker:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
0adacd5c5583        ubuntu              "/bin/bash"         14 seconds ago      Up 14 seconds
                    wizardly_hawking
root@docker:~# docker commit -m "ubuntu" -a "Atul Harsha" 0adacd5c5583 atulharsha/ubuntu
sha256:838d9eb4b836f41213528f80490a8ea7c73ce647072e53149c72550beaaffc2d
root@docker:~# docker push atulharsha/ubuntu
The push refers to a repository [docker.io/atulharsha/ubuntu]
73e5d2de6e3e: Pushed
08f405d988e4: Pushed
511ddc11cf68: Pushed
a1a54d352248: Pushed
9d3227c1793b: Mounted from library/ubuntu
latest: digest: sha256:2b7b54fd34f2490f0891ee0295d3599ed1b4c88b563cdc04210cbf2d2d958fa5 size: 1357
root@docker:~# _
```

Docker will attempt to push the various layers that make the image If the layer is pre-existing on the Docker Hub, it will skip it

Before pushing the image make sure to add the image tag as "<docker_hub_username>/image_name"
This can be done using either "docker commit" as shown above or by using "docker tag"
docker commit [OPTIONS] CONTAINER [USERNAME/][REPOSITORY[:TAG]]

Docker Hub Push – docker tag

```
root@docker:~# docker tag hello-world atulharsha/hello-world
root@docker:~# docker push atulharsha/hello-world
The push refers to a repository [docker.io/atulharsha/hello-world]
98c944e98de8: Layer already exists
latest: digest: sha256:c5515758d4c5e1e838e9cd307f6c6a0d620b5e07e6f927b07d05f6d12a1ac8d7 size: 524
root@docker:~#
```

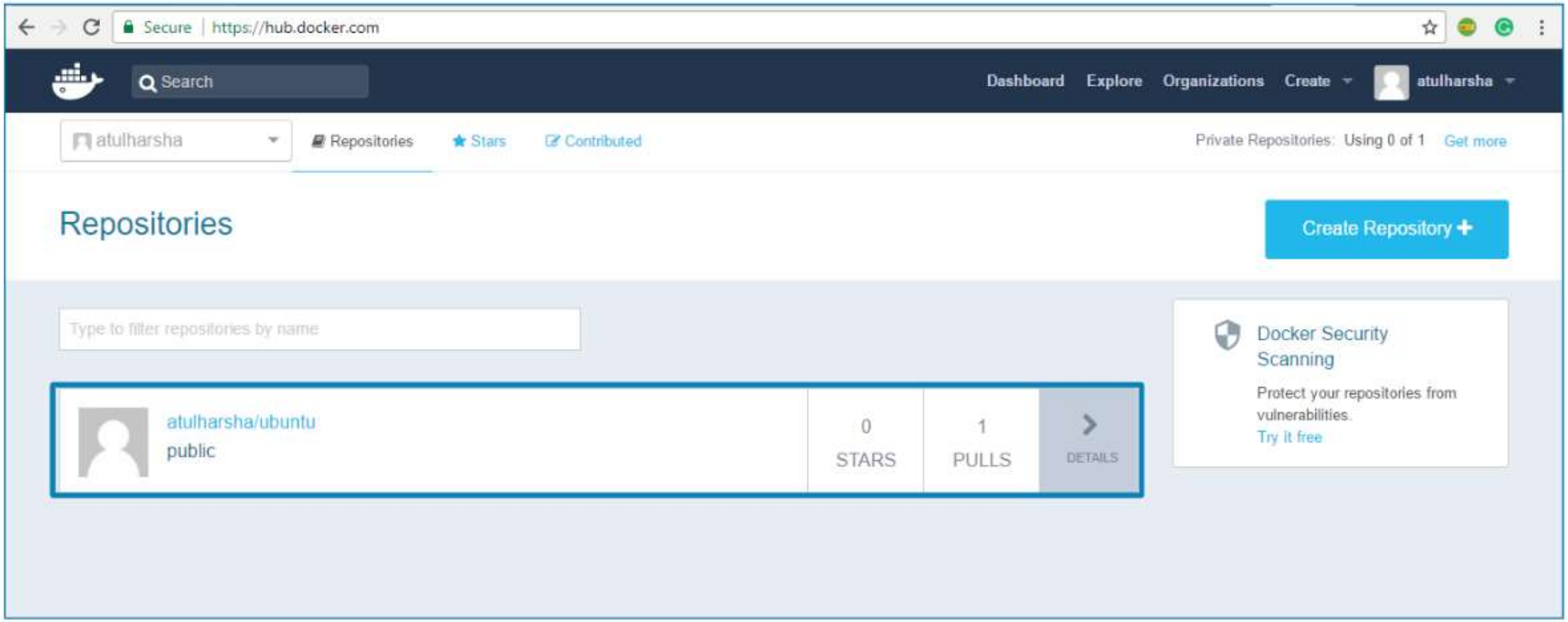
```
docker tag [OPTIONS] IMAGE[:TAG] [REGISTRYHOST/][USERNAME/]NAME[:TAG]
```

Docker Hub Pull

```
root@docker:~# docker pull atulharsha/ubuntu
Using default tag: latest
latest: Pulling from atulharsha/ubuntu
Digest: sha256:2b7b54fd34f2490f0891ee0295d3599ed1b4c88b563cdc04210cbf2d2d958fa5
Status: Image is up to date for atulharsha/ubuntu:latest
root@docker:~#
```

Once the image is pushed to docker hub, it can be downloaded via pull command

Docker Hub: Image published



Running a Private Registry

- Pull the official *registry* image and run it as a detached container

```
root@docker:~# docker pull registry:2
2: Pulling from library/registry
709515475419: Pull complete
df6e278d8f96: Pull complete
4b0b08c1b8f7: Pull complete
80119f43a01e: Pull complete
acf34ba23c50: Pull complete
Digest: sha256:412e3b6494f623a9f03f7f9f8b8118844deaecfea19e3a5f1ce54eed4f400296
Status: Downloaded newer image for registry:2
root@docker:~# docker run -d -p 5000:5000 registry:2
b6c4fdc2086d5d59f3f93adad7fdfc6d185d4f669b12646c67662e33cb6d272f
root@docker:~# curl -i http://localhost:5000/v2/
HTTP/1.1 200 OK
Content-Length: 2
Content-Type: application/json; charset=utf-8
Docker-Distribution-Api-Version: registry/2.0
X-Content-Type-Options: nosniff
Date: Thu, 11 May 2017 11:07:38 GMT
```

running the Docker registry with API version v2.

Quick test to check that the registry is running

Running a Private Registry

- Tag the image with the proper naming convention for use with a private registry
- In this case the registry is running at *http://localhost:5000*, so we will prefix our tag with localhost:5000 and then push this image to the private registry

```
root@docker:~# docker tag ubuntu localhost:5000/ubuntu
root@docker:~# docker push localhost:5000/ubuntu
The push refers to a repository [localhost:5000/ubuntu]
73e5d2de6e3e: Pushed
08f405d988e4: Pushed
511ddc11cf68: Pushed
a1a54d352248: Pushed
9d3227c1793b: Pushed
latest: digest: sha256:f3a61450ae43896c4332bda5e78b453f4a93179045f20c8181043b26b5e79028 size: 1357
root@docker:~#
```