USING DOCKER IN SOFTWARE INSTALLATIONS

Finding the correct image

- Previously, we used images to download and install software on the machine using containers.
- An image can be identified by the name and the version of the software it contains. For example, ubuntu:14.04
- Every Docker image contains an *identifier*, which is a long set of numbers and characters. It gets changed whenever a change is made to the underlying files within.
- Because image identifiers are not human-readable, Docker users work with repositories instead.

Docker repositories

- A repository can be thought of as a large vessel containing a number of images.
- It has a name that is a lot similar to URLs. For example, `quay.io/replicatedcom/nagios`. It is made up of the following segments:
  - *The registry host:* *`quay.io`*
  - *The username:* *`replicatedcom`*
  - *The image:* *`nagios`*
- An image can be further identified by using tags. A tag often reflects the image version. A tag is added at the end of the image name, preceded by a colon like `quay.io/replicatedcom/nagios:latest`
- But that is not the only possible usage. In fact, an image can have more than one tag.
- They can be used to specify the minor version of your software. For example, an image might have the tag 1, 1.1

Docker hub

- You may've identified your required image using the name, the tags, and the repository. But how do you find that repository? That is the role of Docker indexes.
- An index contains a catalog for the available Docker repositories. Think of it like a search engine. A search engine does not host web pages, but it contains an easy-to-use way to search and find the page that you want.
- There are more than one Docker index available, but the default one is Docker Hub. You can use the command line tools to pull images from Docker hub or from other image indexes.
- When you search Docker hub for an image, the following information is returned:
  - *Number of pulls: how many times that image was pulled*
  - *The number of stars: can act as a rough indicator to the quality of the image depending on the community's opinion.*
  - *Whether or not the image is trusted*
  - *Whether or not the image is "official"*

Working with Docker Hub

- You can use Docker command line tools to create and publish images to Docker Hub. You have two options for this:
  - *Create the image locally then push it to the repository (untrusted)*
  - *Create a Dockerfile (discussed later) and use Docker's continuous build system to build the image. This is preferred by most users than the first approach; as the Dockerfile contains the commands and the activities that the image is going to make when a container is created. Images pushed this way are marked as TRUSTED.*
- Before you can push images to Docker Hub, you'll have to create an account on Docker (the basic type is free). After which you have to issue `docker login` command to authenticate yourself. Then you can push images to any repository that you own. You can logout after you are done by using `docker logout`.

Other registries than Docker Hub

- Although Docker Hub is very popular, and would contain most of what you need to test and learn Docker, it is not the only source of images.
- You can use other registries provided by other companies. You can also host your own repository, build your own image using a Dockerfile or manually download and run an image from a file.
- If you want to download an image from a registry other than Docker's default one, just specify the full path instead of just the image name. For example, `docker pull quay.io/nitrous/wordpress`, which will download a WordPress image from the quay.io repository.
- Notice the way the URL is formed: it starts with the provider (quay.io), then the username of the image owner (nitrous), then the image name (wordpress). Optionally there could be tags specifying the version or other details.
- Docker knows how to communicate with the registry and download the image. But sometimes registries require that you authenticate yourself before downloading any images. If that's the case you'll have to consult the appropriate image documentation.

Obtaining an image from a file

- Pulling images from repositories may be the easiest option, but it is not always available. Let’s say you have a server with no Internet connection, or behind a firewall that block access to repositories, or in any situation where you don’t have access to an online repository, but you do have access to an image file. Using Docker, you can load an image from this file easily.
- In the following example, I already have an image file busybox.tar available. It contains a busybox image. First lets remove the image already available on the system:
  ```
  docker rmi busybox
  ```
- Then let's load the image that we have locally using:
  ```
  docker load –i busybox.tar
  ```
- You can also save an image after it's been pulled and use it elsewhere. For example, the following command will save the busybox image pulled from Docker Hub repository to a local .tar file:
  ```
  docker save –o busybox.tar busybox:latest
  ```

Building your own image

- You can build an image using a Dockerfile. It is just a script that describes how the image is going to be built, what commands to run, what files to be included and so on.
- In the following example, I am using a Dockerfile to build the server container that we've worked with in the previous section:
  - Create an empty directory somewhere on your filesystem: `mkdir mydockerimage`
  - Using your favorite editor, create a script file called runner.sh: `vim runner.sh`
  - Write the following inside this file:

```
#!/bin/sh
echo "Server started at " $(date)
while true; do
     # The loop is doing nothing. It is just to keep the container running in the
background
     sleep 5
done
```

  - Using your favorite editor, create a new file called Dockerfile: `vim Dockerfile` and write the following inside it:

```
FROM busybox:latest
RUN mkdir /runner
COPY ./runner.sh /runner
RUN adduser -DHs /bin/sh worker
WORKDIR /runner
RUN chown worker runner.sh
RUN chmod a+x runner.sh
USER worker
CMD ["/runner/runner.sh"]
```

- Now you are ready to build your image:

```
docker build –t masteringdocker/server:latest .
```