**DESCRIPTION**

**Building an Image**

**References**

* [Docker CLI reference](https://docs.docker.com/reference/)
* [docker build](https://docs.docker.com/engine/reference/commandline/build/)
* [docker commit](https://docs.docker.com/engine/reference/commandline/commit/)
* [docker file](https://docs.docker.com/engine/reference/builder/)

An image will typically be built from a dockerfile, though it can be built from a container that's already running and has been modified. The former is preferred, but the latter can be helpful in a development environment.

**Create image with build**

> docker build anyflags PATH

* docker build indicates to the CLI that you are going to create a new image. You have the option of adding flags for additional information and configuration. A couple flags to note:
  + -f path indicates that the dockerfile to be used is in the location indicated from the subsequent path (it will be used when that path is different from the build context.)
  + -t name:version allows you to name the image being created and indicate the version. If you were to just have -t name it would default to having the version marked as latest.
* The path specified at the end of the docker build command will be understood as the build context. It will often just be . to indicate the current directory. Though it could be a git url or even a path to another part of your filesystem.

**Example**

Create and save the following docker file in a new empty folder. Perhaps you create a folder docker-examples in your Documents folder.

# Define the parent image

FROM ubuntu

# Install needed programs

RUN apt-get update

RUN apt-get -y dist-upgrade

RUN apt-get -y install default-jdk

# Within the image and thus container, set the working directory to the new directory example

WORKDIR /example

# Create Hello World Java program and save it in the appropriate file

RUN echo 'public class HiWorld{ public static void main(String[] args){System.out.println("Hi world");}}'> HiWorld.java

# Compile the Java program, creating the file that the JVM can actually run

RUN javac HiWorld.java

# Run the HelloWorld program in the container

CMD ["java", "HiWorld"]

From the same directory in which the dockerfile has been saved run the subsequent command. (If you saved it in the directory as described above then you would need to run cd Documents\docker-examples)

> docker build -t java-hello-world .

Do not forget the . at the end.

There are a couple of things to take notice of here. First, the dockerfile need be the only thing in its directory from which we run the build command. That being said, typically you will also have other files in that directory that are needed for the image.

If we wanted to build the same docker image, perhaps from another folder entirely, then we would have the following command:

>docker build -f "C:\Users\UserName\Documents\Dockerfile" -t java-hello-world .

**Note**: Don't forget the . at the end- indicating that the current context is your build context.

**Create image with commit**

Assume that you have a container that is already running and you have made some changes to it. Maybe you have installed needed software or changed some configuration.

Then you can commit these changes to a new image that can be used to spin up even more containers.

Note the new image won't include any data saved in the *volumes* of your container.

> docker commit flags CONTAINER imagename

* Essentially you are *commiting* the changes from the container specified to the image specified.
* It is typically preferrable to use a dockerfile, but there are some use cases like debugging where this is helpful.
* There are a number of flags you can use. They include:
  + -c will allow you to execute/apply Dockerfile instructions
  + -m message to include when committing changes
  + -p pauses the container whose state you are committing to the new image

For example imagine you installed some new software and you wanted to create a new image from your current running container:

>docker commit -c "WORKDIR /new-example" -m "added new technology" -p 1f7e0cf664ca my-fabulous-image

**Imgage management**

To list all the images.

* docker images
* docker images -a
  + -a allows you to list even hidden images.

**Helpful Reading/Examples**

* [Web app for beginners in Python](https://github.com/docker/labs/blob/master/beginner/chapters/webapps.md)
* [Basic Spring App Example](https://stackify.com/guide-docker-java/)