**Docker Stuff**

# Docker Basic Commands

* Install Docker: **yum install docker**
* Start Docker Service: **systemctl start docker** or **service docker start**
* See all Docker Containers on system: **docker ps –a**
* See all Docker images: **docker images**
* Pull a Docker image: **docker pull location/image**

**Example:** docker pull docker.elastic.co/elasticsearch/elasticsearch:5.3

* Grab and run docker container from docker hub: **docker run container-name** (hello-world)
* Run an OS command inside docker image: **docker run apache\_base cat /etc/passwd**

An image is a filesystem and parameters to use at runtime. It doesn’t have state and never changes. A container is a running instance of an image.

So when you type ***docker run***, you load the image into the container.

You can get images from the Docker Store: <https://store.docker.com/>

EXAMPLE of ElasticSearch Docker Container for dev testing:

1. docker pull docker.elastic.co/elasticsearch/elasticsearch:5.3.0
2. docker run -p 9200:9200 -e "http.host=0.0.0.0" –e\ "transport.host=127.0.0.1"\ docker.elastic.co/elasticsearch/elasticsearch:5.3.0
3. To use a custom elasticsearch.yml file add it to the docker run command:
4. docker run -v full\_path\_to/custom\_elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml -p 9200:9200 -e "http.host=0.0.0.0" –e\ "transport.host=127.0.0.1"\ docker.elastic.co/elasticsearch/elasticsearch:5.3.0

# Build Your Own Image

## Using a pre-done base OS image

You can create your own custom images, but it’s easiest to start with a image base image from the Docker hub. Creating a new, from scratch base image is not trivial.

1. To start you create a directory for your image (keep them separate like /opt/docker/apachec\_docker
2. Then Sample DockerFile:

FROM docker/whalesay:latest

RUN apt-get -y update && apt-get install -y fortunes

CMD /usr/games/fortune -a | cowsay

or a Dockerfile for a basic webserver might look like:

# Barebones Apache installation on Ubuntu  
  
FROM ubuntu  
  
MAINTAINER DockerFan version 1.0  
  
ENV DEBIAN\_FRONTEND noninteractive  
  
ENV APACHE\_RUN\_USER www-data  
ENV APACHE\_RUN\_GROUP www-data  
ENV APACHE\_LOG\_DIR /var/log/apache2  
ENV APACHE\_LOCK\_DIR /var/lock/apache2  
ENV APACHE\_PID\_FILE /var/run/apache2.pid  
  
RUN apt-get update && apt-get install -y apache2  
  
EXPOSE 8080  
  
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]

Here is what these commands do:

* FROM is always your first instruction, because it names the base image you're building your new image from.
* MAINTAINER is the creator of the Dockerfile.
* ENV sets environment variables, in the form ENV [key] [value]. This assigns a single value to the key, as in our example Dockerfile. The value can be any string, including spaces and punctuation, so you can configure values like IP addresses, URLs, and passphrases. Note that when you set DEBIAN\_FRONTEND noninteractive (for an unattended installation) in your Dockerfile there is no equals sign, as there is when you script a standard Debian or Ubuntu installation with export DEBIAN\_FRONTEND=noninteractive.
* There are multiple instructions for setting environment variables: ADD, COPY, ENV, EXPOSE, LABEL, USER, WORKDIR, VOLUME, STOPSIGNAL, and ONBUILD.
* RUN executes commands. The example above, RUN apt-get update && apt-get install -y apache2, demonstrates two important steps. It is a good practice to use apt-get update && apt-get install foo together to ensure that an updated packaged will be installed. Docker makes generous use of caching, so this prevents a cached packaged from being installed. If you're sure your cached packages are fresh enough then it's not necessary, and will save you some download time. apt-get install -y must be used together with ENV DEBIAN\_FRONTEND noninteractive; it means answer Yes to all prompts and run non-interactively.
* EXPOSE defines which ports you want open at runtime, in a space-delimited list.
* CMD **can be used only once** in your Dockerfile. If you have more than one, only the last one will run. The preferred syntax is CMD ["executable","param1","param2"]. The parameters are optional and comma-separated if you have more than one.

1. Now, from within the same directory as your DockerFile, you can build an image using:

**docker build -t apache\_base . (-t is the image tag or name)**

# Build Your Own Ubuntu Base Docker Image

1. You need install debootstrap if not already installed on your system. This grabs the files you need.
2. xcsx
3. xcsxd
4. ZXsx

# Build Your Own Base Docker Image from a Tarball

1. First, you need to create the Tarball of your OS

**tar -cvf os.tar --exclude=/os.tar --exclude=/proc**

**--exclude=/tmp --exclude=/mnt --exclude=/dev --exclude=/sys /**

2. Then import the tar file into docker as an image

**cat os.tar | docker import - my\_os**

# Offline Docker Hub

You can build an offline docker hub.

1. Download the images you want
   1. **sudo docker pull ubuntu** (or whatever)
2. Save the image to a file
   1. **sudo docker save –o ubuntu\_image.docker ubuntu**
3. Transfer the file
4. load the image on offline server
   1. **sudo docker load ubuntu\_image.docker**