Educational Material

Included is background information on how to install and use Docker. Also an example for creating and running a simple Docker container are included.

Install Docker on your machine

Mac OS

Windows

Docker's website provides instructions for installation on may distributions of Linux. Provided below is the link for installing Docker on Ubuntu.

Ubuntu

Working with Docker from the command line once installed

With Docker one can build containers by using Docker's library of open source Dockerfiles. Examples one can use from Docker Hub are Ubuntu, Hadoop, and MySQl. However, one can build their own container by writing their own Dockerfile. Docker uses a Dockerfile to build images (what are needed to run a container). A Dockerfile is just text document that contains commands Docker will use to build an image for a container. There are four commands we used to build our Docker container image.

FROM

> FROM

This instruction sets a base image for the container image one is building with their Dockerfile. It must be the first non-comment instruction in the Dockerfile.

Docker documentation on FROM

COPY

> COPY

The COPY instruction copies new files or directories from and adds them to the filesystem of the container from this Dockerfile built image.

Docker documentation on COPY

RUN

> RUN

The RUN instruction will execute any commands in a new layer on top of the current image and commit the results. The resulting committed image will be used for the next step in the Dockerfile.

Docker documentation on RUN

CMD

CMD command param1 param2 CMD [“command”, “param1”, “param2”]

The CMD instruction is used to provide defaults for an executing container. These defaults can include an executable or they can omit an executable.

Docker documentation on CMD

Dockerfile Example

```
FROM ubuntu
RUN apt-get update
COPY . /my_files
CMD ["sleep", "infinity"]
```

The Dockerfile above builds from the Docker Hub ubuntu image, runs the `apt-get update` command, copies all the contents of the current directory into a directory called **my_files** in the container, and sets the default command to `sleep infinity`.

This section describes how to build your container image from the Dockerfile above, run the container, and work inside the container. There are only a few Docker commands one needs to be familiar with to use their first Docker container.

docker build

docker build -t my-first-container-image .

The above command builds an image from a Dockerfile in the current directory you are in (run the command while in the directory where the Dockerfile is OR give the path to the directory where your Dockerfile is located). The **-t** flag allows you to name this image, and in this case we named it **my-first-container**.

Docker documentation on docker build

docker run

docker run -d –name my-first-container-name

The above command starts a container from the image built from the previous `docker build` command. It names it **my-first-container-name**.

Docker documentation on docker run

docker exec

docker exec -it my-first-container-name bash

The above command starts an interactive session within the container using the bash application (shell).

Docker documentation on docker exec

Other Helpful Docker Links

Docker Hub

Dockerfile Best Practices

Dockerfile Reference

Working with Hadoop and Spark in Docker containers

Hadoop

Once your container is up, you can perform Hadoop commands by using the environment variable `$HADOOP_PREFIX`. For example, if you want to download the results to your local output folder you would simply run the following command. `$HADOOP_PREFIX/bin/hdfs/ dfs -get output output`

Really you could write multiple scripts, inputs into container, that run all the Hadoop commands using the `$HADOOP_PREFIX` environment variable. Thus opening up your time for developing your MapReduce application rather than spending time on setup and going through redundant steps.

Spark

The Spark container has the same story, there is a `$SPARK_HOME` environment which leads you down to where all the Spark files are stored. So from `$SPARK_HOME` you would be able to perform any operation you would like.

For example, Spark provides a shell for a user to learn Spark (you can find their tutorial here). If you would like to run that shell you would just execute `$SPARK_HOME/bin/spark-shell`.

You could also create scripts, input them into your container, and run the scripts that do all of the work with `$SPARK_HOME`.