Docker - RethinkDB recipe

Purpose:

The purpose of this recipe is to explain how to build a docker image with RethinkDB pre-installed and ready to run.

We will do this by creating a Dockerfile that downloads the base image, updates and adds an apt source and then installs the database software.

Create the Dockerfile

The dockerfile example used below comes from @github: crosbymichael/Dockerfiles/rethinkdb/

In order to build an image that contains a running instance of RethinkDB we need to specify a number of things. First is the base image to use. Once we've defined the base image we then need to setup our sources repo for ***apt*** to use.

Then we update the base system with calls to **apt-get update** and **apt-get upgrade**.

The next step is to install all the pre requisites for RethinkDB,and finally RethinkDB itself.

Once we've got RethinkDB installed we next need to setup some environmental parameters for it. These include ports and file system locations to use.

Finally, the entrypoint is set and a default command. In this case we start the app passing in the help parameter.

```
# Dockerfile for Rethinkdb
# http://www.rethinkdb.com/

FROM ubuntu

RUN echo "deb http://archive.ubuntu.com/ubuntu precise main universe" >
/etc/apt/sources.list
RUN apt-get update
RUN apt-get upgrade -y

RUN apt-get install -y python-software-properties
RUN add-apt-repository ppa:rethinkdb/ppa
RUN apt-get update
RUN apt-get install -y rethinkdb

# Rethinkdb process, cluster, and webui
EXPOSE 28015 29015 8080

VOLUME ["/rethinkdb"]
WORKDIR /rethinkdb
ENTRYPOINT ["/usr/bin/rethinkdb"]
CMD ["--help"]
```

A note on ports in Docker and containers.

Two of the core concepts of docker are repeatability and portability. Images should able to run on any host and as many times as needed. With Dockerfiles you have the ability to map the private and public ports, however, you should never map the public port in a Dockerfile. By mapping to the public port on your host you will only be able to have one instance of your dockerized app running.

If you care what public port the container maps to, you must pass the -p option when running the image, otherwise, docker will automatically assign a port for the container.

Never map the public port in a Dockerfile

Create the image

Copy the contents of the Dockerfile listed above and save to a file named Dockerfile

Next use the ***build*** command to create the image. Pass in a name and optionally a tag with the -t flag.

Note the '.' at the end of the line, this signifies to Docker that the Dockerfile is in the ***current*** directory.

```
docker build -t johnzan/rethink .
```

Tagging the image

Unless you are experimenting with creating an image with docker, it's always a good idea to pass the -t option to the docker build command. This ensures that resulting image is tagged. Remember, a simple human readable tag will always be easier to remember and will also help you remember what each image was created for.

Always pass the -t parameter to tag the resulting image

Save / Commit the image

After you have your image setup and configured the way you want, you can save the resulting container image to the docker index to share with others. The command to do this is as follows:

```
docker push NAME
```

In this example I'll use johnzan/rethink, but this can be anything you want. Try to keep it simple and give the container image a meaningful name so others can recoginise what it is for.

```
docker push johnzan/rethink
```

You need to register on the site to be able to push your images.

Push an image or a repository to the registry

Running as a new container

If you have more than one container running on a server using the port 8080 then you'll need to map the port 8080 that RethinkDB wants to listen on to something else. Below are two different commands to start the container. The first is for cases where there are no other services listening on port 8080 and the second example is where there is.

Example 1. When there is nothing using port 8080

```
docker run johnzan/rethink --bind all
```

Example 2. In the case where you already have a service using port 8080

You can map these requests from a different port.

```
docker run -p 8091:8080 johnzan/rethink --bind all
```

In this example, all calls to the server at port 8091 will be passed/mapped back to port 8080 on the container running the RethinkDB service.

Final thoughts & credits

Docker has the potential to be a game changer on so many different levels and in so many different ways that its hard to imagine where this technology will go. But one thing is certain, for a lot of people the power, flexibility and ease of use of this technology will make creating amazing applications so very much quicker and easier than ever before. So thanks to everyone involved in creating Docker and making available to everyone.

Here are some links to check out if you're interested in more details on creating

containers.

Docker - The home of Docker and all things containers

Docker command line help - A list of the docker commands currently available

Michael Crosby's collection of Dockerfiles on Github - An excellent collection of examples

Dockerfile Best Practices - Another gem from Michale Crosby

Any comments/feedback please feel free to give me a shout.

Thanks.

johnzan at gmail dot com

You can also find me on the web at the following:

Blog - My blog

Linkedin - My linkedin profile

Twitter @john_zanchetta

About.me

Github Github - source for this as well as a PDF version for download