How to make a Dockerfile for Elasticsearch

Docker containers gives you a way to ship and run applications with their environment in an isolated and repeatable way. While there are a myriad of Docker images out there [**1**](https://www.elastic.co/blog/how-to-make-a-dockerfile-for-elasticsearch#fn1), creating your own Dockerfile allows you to customize it, for instance by installing plugins, changing the base image, strip out what you don't need, etc. Dockerfiles also act as a way to document how an application gets installed and deployed.

In this introductory post we will go through how to create a Dockerfile from scratch for running Elasticsearch, and discuss a few things that you need to consider when creating your own.

Building the image

A Dockerfile is a recipe with steps describing how to build your Docker image. You start from a base image, which gives you the basics needed for running applications, then run steps on top of that, which results in a new image. If you want you can also use the resulting image as a base image for another image.

The simplest Dockerfile you can create is something like this:

FROM ubuntu:14.04

Put this in a file called Dockerfile.   
You can now build it by running:

docker build -t my-es-image .

The image has been built and can be run with:

docker run --rm -it my-es-image /bin/bash

While this created an image that is not very useful, we have now learned how to build and test an image. Now let’s create a more useful one. Since Elasticsearch requires Java to run, let’s install it first.

FROM ubuntu:14.04

ENV DEBIAN\_FRONTEND=noninteractive

RUN apt-get install -y --no-install-recommends software-properties-common && add-apt-repository -y ppa:webupd8team/java && \

apt-get update && \

(echo oracle-java8-installer shared/accepted-oracle-license-v1-1 select true | sudo /usr/bin/debconf-set-selections) && \

apt-get install --no-install-recommends -y oracle-java8-installer && \

rm -rf /var/cache/oracle-jdk8-installer && \

echo "networkaddress.cache.ttl=60" >> /usr/lib/jvm/java-8-oracle/jre/lib/security/java.security && \

apt-get clean && rm -rf /var/lib/apt/lists/\*

ENV JAVA\_HOME /usr/lib/jvm/java-8-oracle

This one installs Oracle JDK 8. If you build and run it with the docker run command above then you can test that is works by running java -version.

At this point we’re ready to install Elasticsearch. Let’s use the apt package.

RUN groupadd -g 1000 elasticsearch && useradd elasticsearch -u 1000 -g 1000

RUN apt-key adv --keyserver pgp.mit.edu --recv-keys 46095ACC8548582C1A2699A9D27D666CD88E42B4 && \

add-apt-repository -y "deb http://packages.elastic.co/elasticsearch/2.x/debian stable main" --keyserver https://pgp.mit.edu/ && \

apt-get update && \

apt-get install -y --no-install-recommends elasticsearch

WORKDIR /usr/share/elasticsearch

RUN set -ex && for path in data logs config config/scripts; do \

mkdir -p "$path"; \

chown -R elasticsearch:elasticsearch "$path"; \

done

Before we run it, we should add an elasticsearch.yml file. Create a file named elasticsearch.yml in the same directory as the Dockerfile, with this content:

cluster.name: "docker-cluster"

network.host: 0.0.0.0

Also, to get logging to work with docker we should add a simple logging.yml file:

rootLogger: INFO,console

appender:

console:

type: console

layout:

type: consolePattern

conversionPattern: "[%d{ISO8601}][%-5p][%-25c] %m%n"

Note: When running with logging to stdout/stderr Docker stores the log in a json file, and it is recommended to specify a max size for the log file to rotate, and a max number of files to keep. E.g. --log-opt max-size=50m --log-opt max-file=10 [**2**](https://www.elastic.co/blog/how-to-make-a-dockerfile-for-elasticsearch#fn2)

To add these files to the container we add the following to the Dockerfile:

COPY logging.yml /usr/share/elasticsearch/config/

COPY elasticsearch.yml /usr/share/elasticsearch/config/

This will bake the files into the image when running docker build. [**3**](https://www.elastic.co/blog/how-to-make-a-dockerfile-for-elasticsearch#fn3)   
What’s left now is to actually make the container run Elasticsearch at startup.

USER elasticsearch

ENV PATH=$PATH:/usr/share/elasticsearch/bin

CMD ["elasticsearch"]

EXPOSE 9200 9300

To run this, first build it as before, and then run with docker run --rm -p 9200:9200 my-es-image. This opens port 9200 from the container to the host and runs CMD. You should now see the Elasticsearch instance starting.

Note that currently this Elasticsearch instance does not persist the data between runs, since it is ephemeral and we didn’t specify any data volumes.

Persistent storage volumes

Storing data persistently in Docker requires the use of volumes. Volumes are a bit tricky because of the way it works with permissions. By default volumes are using bind-mount, which means that a file belonging to a user with ID 1000 inside the container will be owned by user 1000 on the host, which may or may not be the same actual user.   
Docker are working on a way of fixing this, and it has been partially implemented, but we need “phase 2” of user namespaces to solve this fully.

In the meantime, we need some workaround for storing the data. One way to get around it is to hard code the User ID and make sure that it is the same on all machines running that container.   
If we only care that the data is persisted between restarts, another way is to let the container create the volume so the permissions are correct for the container, since we don’t need to access the container contents from the host machine.

Let’s do the latter approach that with a feature called “named volumes”. Create a named volume:

docker volume create --name esdata

You can mount that volume using the -v option:

docker run --rm -ti -p 9200:9200 -v esdata:/usr/share/elasticsearch/data my-es-image

Even if you restart your Elasticsearch container now it should preserve all data.

Note: This requires at least version 1.10 of Docker

Memory and heap size

By default memory for a container is unbounded. If you want to limit for the max memory the container uses you can specify e.g. --memory="4g" with docker run.  
You should also set the heap size for Elasticsearch. The normal recommendation with allocating half of the memory to the heap also applies. For example:

docker run --rm -ti -p 9200:9200 -v esdata:/usr/share/elasticsearch/data --memory="4g" -e ES\_HEAP\_SIZE=2g my-es-image

Conclusion

We have created a basic docker image which runs Elasticsearch and stores the data persistently. It is basic, but it's a starting point. Next steps could be to [**set up networking**](https://www.elastic.co/blog/docker-networking) and [**install plugins**](https://www.elastic.co/blog/elasticsearch-docker-plugin-management), which we will look at in follow-up blog posts.

EXAMPLE : ----------

# Start from Ubuntu base image

FROM ubuntu:14.04

# Set environment variable to avoid interactive prompts during package installation

ENV DEBIAN\_FRONTEND=noninteractive

# Install Java

RUN apt-get update && \

apt-get install -y --no-install-recommends software-properties-common && \

add-apt-repository -y ppa:webupd8team/java && \

apt-get update && \

echo oracle-java8-installer shared/accepted-oracle-license-v1-1 select true | /usr/bin/debconf-set-selections && \

apt-get install -y --no-install-recommends oracle-java8-installer && \

rm -rf /var/cache/oracle-jdk8-installer && \

echo "networkaddress.cache.ttl=60" >> /usr/lib/jvm/java-8-oracle/jre/lib/security/java.security && \

apt-get clean && rm -rf /var/lib/apt/lists/\*

# Set JAVA\_HOME environment variable

ENV JAVA\_HOME /usr/lib/jvm/java-8-oracle

# Add Elasticsearch repository and install Elasticsearch

RUN groupadd -g 1000 elasticsearch && useradd elasticsearch -u 1000 -g 1000 && \

apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys D88E42B4 && \

add-apt-repository -y "deb http://packages.elastic.co/elasticsearch/2.x/debian stable main" && \

apt-get update && \

apt-get install -y --no-install-recommends elasticsearch

# Set up Elasticsearch directories and permissions

WORKDIR /usr/share/elasticsearch

RUN set -ex && for path in data logs config config/scripts; do \

mkdir -p "$path"; \

chown -R elasticsearch:elasticsearch "$path"; \

done

# Copy configuration files

COPY logging.yml /usr/share/elasticsearch/config/

COPY elasticsearch.yml /usr/share/elasticsearch/config/

# Switch to elasticsearch user

USER elasticsearch

# Set PATH environment variable

ENV PATH=$PATH:/usr/share/elasticsearch/bin

# Define default command to run Elasticsearch

CMD ["elasticsearch"]

# Expose necessary ports

EXPOSE 9200 9300