# An exercise in Discovery, Building Docker Images, using Makefiles & Docker Compose. – Part 1

## Docker images, Multi stage builds, Makefiles Projects with sub directories and their own Makefiles.

**Overview**

This all started with another [blog](https://medium.com/@georgelza/an-exercise-in-discovery-streaming-data-in-the-analytical-world-part-1-e7c17d61b9d2) I was writing, where in I was exploring Kafka, kSqlDB, Apache Flink, Apache Hive, Apache Iceberg & Apache Paimon and various other supporting bits and bobs.

During that “rabbit hole” I ended needing to build “some…” docker images, leading to allot of learning along the way and redoing things over and over to make it easier, faster, simpler.

Figured others might be interested, find value out of it.

I’m by no means an expert, this is purely what I learned along the way to make it easier for myself.

We will cover a basic docker image build (Ubuntu 20.04), installing some basic OS packages, installing application server (based OpenJDK 11) and Apache Hadoop DFS cluster, adding required configuration files (directly into the container during build or mounting them at run time) & dealing with environment variables.

During this phase, I will show 2 Dockerfile’s. One where I stage some files locally first and a 2nd where I pull the source files directly from the internet during the build.

I will additionally discuss the importance of the build order in your Dockerfile.

As a separate mini project, I will also show a docker image build using a multistage image build to reduce the image size and attack vectors.

As a sidetrack we will also have a look at using Makefiles, they are a great way to “package all your commands together.

To demonstrate all the above processes, we will build an Apache Hadoop DFS cluster, based on Ubuntu 20.04 / OpenJDK 11 / Apache Hadoop 3.3.8, comprised out of:

* namenode
* nodemanager
* resourcemanager
* datanodes &
* historyserver

As it stands this will be a X-part posting, but it’s by no means complete.

NOTE: I work on an Apple MacBook based on their ARM64 aka AARC64 architecture. Where needed I will point out which lines can be changed to make everything Intel/AMD64 compatible.

Good luck, as always, this is all fraught with rabbit holes, so many and you can disappear so easily… But it’s all fun and you will always discover something new or validate a previous learned skill.

See [Building Docker Images](ttps://github.com/georgelza/dockerimagebuilding.git) for all the code.

**About Me**

I’m a techie, a technologist, always curious, love data, have for as long as I can remember always worked with data in one form or the other, Database admin, Database product lead, data platforms architect, infrastructure architect hosting databases, backing it up, optimizing performance, accessing it. Data data data… it makes the world go round.

In recent years, pivoted into a more generic Technology Architect role, capable of full stack architecture.

[George Leonard](https://www.linkedin.com/in/george-leonard-945b502/)

[georgelza@gmail.com](mailto:georgelza@gmail.com)

------------------------------------------------------------------------------------------------------------------

# An exercise in Discovery, Building Docker Images, using Makefiles & Docker Compose. – Part 2

## Docker images, Multi stage builds, Makefiles Projects with sub directories and their own Makefiles.

Ok we will start with our base OS image. See the build-ubuntu-os-20.04 sub directory.

NOTE: this is one of our ARM64 specific images, to revert to standard AMD64 (Intel), simply remove the arm64v8/bit from the FROM clause.

FROM arm64v8/ubuntu:20.04

# original https://github.com/YanYunNN/hadoop-cluster-docker-m1/blob/main/Dockerfile

WORKDIR /root

RUN echo "--> Install OS dependencies openssh-server & misc tools" && \

build\_deps="openssh-server wget neovim curl unzip net-tools" && \

apt-get update && \

apt-get install -y $build\_deps

RUN echo "--> Purge apt artifacts" && \

apt-get purge -y --auto-remove $build\_deps && \

apt-get clean && \

rm -rf /var/lib/apt/lists/\*

We start our build by specifying our source image, in this case arm64v8/ubuntu:20.04

Next we define a WORKDIR /root this accomplishes 2 outcomes; it creates the directory if it does not exist, and it changes into the directory.

Following this we execute a docker primitive/command called RUN. I start the command with echo “some text” to output a description of what’s being done, followed by “&& \” which creates a line continue onto the next line.

RUN echo "--> Install OS dependencies openssh-server & misc tools" && \

build\_deps="openssh-server wget neovim curl unzip net-tools" && \

apt-get update && \

apt-get install -y $build\_deps

The second line we define a variable “build\_deps” with a list of packages assigned to it that we want to install using the apt-get install command:

build\_deps=”openssh-server wget neovim curl unzip net-tools”

The value of using a variable here is seen in the next RUN command where we clean up after the install and now instruct apt-get to clean up, using: apt-get purge – auto-remove &build\_deps variable/list define previously.

RUN echo "--> Purge apt artifacts" && \

apt-get purge -y --auto-remove $build\_deps && \

apt-get clean && \

rm -rf /var/lib/apt/lists/\*

NOTE: Lesson learned, the package vim during installation request’s physical location which can’t be by passed, neovim on the other hand not.

The build is executed by calling make build in the same directory, this in return issue:

sudo docker build -t ubuntu20.04:$(VERSION) .

To see the output image execute docker images

Ok, that’s all for Part 2, In the next part we will move onto building our base Open JDK 11 application server using our base image.

See [Building Docker Images](ttps://github.com/georgelza/dockerimagebuilding.git) for all the code.

**About Me**

I’m a techie, a technologist, always curious, love data, have for as long as I can remember always worked with data in one form or the other, Database admin, Database product lead, data platforms architect, infrastructure architect hosting databases, backing it up, optimizing performance, accessing it. Data data data… it makes the world go round.

In recent years, pivoted into a more generic Technology Architect role, capable of full stack architecture.

[George Leonard](https://www.linkedin.com/in/george-leonard-945b502/)

[georgelza@gmail.com](mailto:georgelza@gmail.com)

------------------------------------------------------------------------------------------------------------------

# An exercise in Discovery, Building Docker Images, using Makefiles & Docker Compose. – Part 3

## Docker images, Multi stage builds, Makefiles Projects with sub directories and their own Makefiles.

This is going to be a rather short issue, below is the Dockerfile used to build our OpenJDK11 image.

We start of by using the FROM primitive, but this time we specify our previously build base OS image (georgelza/ubuntu20.04:1.0.1).

As can be seen, we use pretty much have the same logic as discussed in part 2, except we changed the value assigned to build\_deps.

The reason we do this as a separate build (vs including the OpenJDK11 as a package in the previous builds build\_deps) is to allow us to use our base OS image from part 2 in other builds, i.e. we might need to build a OpenJDK8 application server also, which then means we can simply re-use the OS image for both, with simply the relevant application server version changed.

The only addition here is we add an environment variable JAVA\_HOME. As this is added to this image all images that will use this image will inherit the variable.

Note: by including the repo owner (georgelza) in this case it allows me to upload mu images to hub.docker.com from where I can pull them when needed…

A Todo that’s now available, multi-platform builds. That’s where a single image name is assigned, but docker builds both AMD64 and ARM64 images… There are more Architectures that can be build. This then allows anyone that issues a pull command to pull the right image for the right architecture, but now that’s a total new Rabbit hole for another day/blog.

FROM georgelza/ubuntu20.04:1.0.1

WORKDIR /root

ENV JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-arm64

# Install some useful tools

RUN echo "--> install Open JDK 11 ARM64" && \

build\_deps="openjdk-11-jdk" && \

apt-get update && \

apt-get install -y $build\_deps

RUN echo "--> Purge apt artifacts" && \

apt-get purge -y --auto-remove $build\_deps && \

apt-get clean && \

rm -rf /var/lib/apt/lists/\*

To execute the build as per previous, execute “make build” in the “build-ubuntu-os-openjdk11” subdirectory. See the Makefile for the command executed, You will notice it is exactly the same as the OS build except for the value assigned to image\_name variable that has changed.

We now have a base Ubuntu 20.04 OS, with some useful tools and OpenJDK 11 installed. This combination can be used for a large amount of the Apache projects available.

As per the README.md, we could consider using a smaller Ubuntu image like jammy… but that’s an exploration for another day. -> TODO

See you in Part 4 where we will start with the Apache Hadoop build.

**My Repo’s**

All the code used during this article will be available on the below GIT repo.

[Building Docker Images](ttps://github.com/georgelza/dockerimagebuilding.git)

**About Me**

I’m a techie, a technologist, always curious, love data, have for as long as I can remember always worked with data in one form or the other, Database admin, Database product lead, data platforms architect, infrastructure architect hosting databases, backing it up, optimizing performance, accessing it. Data data data… it makes the world go round.

In recent years, pivoted into a more generic Technology Architect role, capable of full stack architecture.

[George Leonard](https://www.linkedin.com/in/george-leonard-945b502/)

[georgelza@gmail.com](mailto:georgelza@gmail.com)

------------------------------------------------------------------------------------------------------------------

# An exercise in Discovery, Building Docker Images, using Makefiles & Docker Compose. – Part 4

## Docker images, Multi stage builds, Makefiles Projects with sub directories and their own Makefiles.

And this is where it gets real…

We will first build a base image, using the previous Open JDK 11 image we build in Part 3. The base HDFS image is then utilized to build the various HDFS servers. Ye and here I am sure someone will tell me I can rather simply start up the base with different commands/input parameter to make it act as the different servers, but that’s for another day. For now, I did It like this, as it also shows some form of inheritance.

For building the base image I will show 2 versions… one where the install software is staged locally, and a 2nd where it is downloaded during the docker build processing using wget in the Dockerfile

For this part we will also be looking at passing in environment variables (I will show a docker-compose.yml snippet and a docker run snippet) and a neat way that those are pushed into configuration values stored in files at startup time by the entrypoint.sh we copied into the image at build time.

Ok, let’s start.

Note the Hadoop binary that is utilized here is the ARM64/AARCH64 version. The 2 descriptions ARM64 and AARCH64 are generally used interchangeably by the industry.

The Intel/AMD64 version can be sourced from “ [hadoop-3.3.5.tar.gz](https://archive.apache.org/dist/hadoop/common/hadoop-3.3.5/hadoop-3.3.5.tar.gz) . Pay attention to the JAVA\_HOME… setting. I should probably rather make this more generic and not include -arm64, but it might have value down the line, if you ever echo’d the environment variable, you will immediately know it’s an ARM64/AARCH64 version/platform build.

Our Dockerfile

FROM georgelza/ubuntu20.04-openjdk11:1.0.1

USER root

RUN echo "--> set environment variable"

ENV JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-arm64

ENV HADOOP\_VERSION=3.3.5

ENV HADOOP\_HOME=/opt/hadoop-3.3.5

ENV HADOOP\_CONF\_DIR=/etc/hadoop

ENV MULTIHOMED\_NETWORK=1

ENV PATH=$HADOOP\_HOME/bin/:$PATH

WORKDIR /tmp/

COPY stage/hadoop-3.3.5-aarch64.tar.gz /tmp/

RUN echo "--> install Apache Hadoop 3.3.5" && \

tar -xzvf /tmp/hadoop-3.3.5-aarch64.tar.gz && \

mv /tmp/hadoop-3.3.5 /opt/ && \

rm /tmp/hadoop-\*

# Intel/AMD64 version

# https://archive.apache.org/dist/hadoop/common/hadoop-3.3.5/hadoop-3.3.5.tar.gz

RUN ln -s $HADOOP\_HOME/etc/hadoop $HADOOP\_CONF\_DIR

RUN echo "--> make Apache Hadoop directories" && \

mkdir -p $HADOOP\_HOME/logs && \

mkdir /hadoop-data

ADD bin/entrypoint.sh /entrypoint.sh

RUN chmod a+x /entrypoint.sh

ENTRYPOINT ["/entrypoint.sh"]

Up next is our Modified Dockerfile where we use wget to download the install media during build time.

I travel allot, and at times during the travelling I’m building. While travelling I don’t always have access to good internet connectivity so what I do is download the install media (\*.tar.gz and \*.jar files) once, stage it into a local stage directory close to my Dockerfile, which then allows me to build and rebuild and rebuild without downloading the files every time… That’s one reason, another could simply be it’s faster, a 3rd and this is for the enterprise guys and their security departments…

They might have a rule that only allows them to build from installation media that the security department have first vetted.

I know a different way to accomplish the same might be to run a private docker image repo which is provisioned with security tooling to scan all images for vulnerabilities and there are loads of those, or it simply might be by staging the installation media on a common shared location anyone needing to build a similar images or rebuild this image can use the exact same install media.

FROM georgelza/ubuntu20.04-openjdk11:1.0.1

USER root

RUN echo "--> set environment variable"

ENV JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-arm64

ENV HADOOP\_VERSION=3.3.5

ENV HADOOP\_HOME=/opt/hadoop-3.3.5

ENV HADOOP\_CONF\_DIR=/etc/hadoop

ENV MULTIHOMED\_NETWORK=1

ENV PATH=$HADOOP\_HOME/bin/:$PATH

WORKDIR /tmp/

RUN echo "--> install Apache Hadoop 3.3.5" && \

wget https://archive.apache.org/dist/hadoop/common/hadoop-3.3.5/hadoop-3.3.5-aarch64.tar.gz && \

tar -xzvf /tmp/hadoop-3.3.5-aarch64.tar.gz && \

mv /tmp/hadoop-3.3.5 /opt/ && \

rm /tmp/hadoop-\*

# Intel/AMD64 version

# https://archive.apache.org/dist/hadoop/common/hadoop-3.3.5/hadoop-3.3.5.tar.gz

RUN ln -s $HADOOP\_HOME/etc/hadoop $HADOOP\_CONF\_DIR

RUN echo "--> make Apache Hadoop directories" && \

mkdir -p $HADOOP\_HOME/logs && \

mkdir /hadoop-data

ADD bin/entrypoint.sh /entrypoint.sh

RUN chmod a+x /entrypoint.sh

ENTRYPOINT ["/entrypoint.sh"]

Both are built by executing the respective make <command> from our Makefile located in our build-hadoop-openjdk11 directory. For the locally staged software execute make buildbase.

For the wget based version I’ve added “make buildwgbase” to instruct the make to use the alternate Dockerfile called Dockerfile.wget. Notice how the docker command is modified to instruct it to use a specified input file and not the default “Dockerfile”.

I tend to find I prefer wget when the download is a single tar.gz file/bundle. If the file to be downloaded is a single jar file, or multiple jars as per the below then the curl command is more useful.

The below command does a couple of things for us. We start with a description of what are doing via the echo “description text”.

Next up we make a directory via (mkdir -p) under our current directory where we will down the file into and changing into the directory. Next is the curl command with -O telling curl to download the file to the current directory and lastly popd, this basically exits the current directory to where we were before the pushd command.

All the below is executed as a single command via the one RUN primitive and the usage of && \ at the end of each line that create all of this as a single line via the line continuation.

RUN echo "-> Install JARs: Flink's Kafka connector" && \

mkdir -p ./lib/kafka && pushd $\_ && \

curl [https://repo1.maven.org/maven2/org/apache/flink/flink-sql-connector-kafka/3.2.0-1.18/flink-sql-connector-kafka-3.2.0-1.18.jar -O && \](https://repo1.maven.org/maven2/org/apache/flink/flink-sql-connector-kafka/3.2.0-1.18/flink-sql-connector-kafka-3.2.0-1.18.jar%20-O%20&&%20\)

curl <https://repo.maven.apache.org/maven2/org/apache/flink/flink-sql-avro-confluent-registry/1.18.1/flink-sql-avro-confluent-registry-1.18.1.jar> -O && \

popd

I will discuss the entrypoint.sh in a later Part. Generally, we want to do the steps that very static as high up in the Dockerfile as possible to take advantage of docker cache’ing.

Think that will be it for part 4… In the next part we will create the various HDFS servers and discuss some of the scripts used to customize each server’s configuration.

**My Repo’s**

All the code used during this article will be available on the below GIT repo.

[Building Docker Images](ttps://github.com/georgelza/dockerimagebuilding.git)

**About Me**

I’m a techie, a technologist, always curious, love data, have for as long as I can remember always worked with data in one form or the other, Database admin, Database product lead, data platforms architect, infrastructure architect hosting databases, backing it up, optimizing performance, accessing it. Data data data… it makes the world go round.

In recent years, pivoted into a more generic Technology Architect role, capable of full stack architecture.

[George Leonard](https://www.linkedin.com/in/george-leonard-945b502/)

[georgelza@gmail.com](mailto:georgelza@gmail.com)

------------------------------------------------------------------------------------------------------------------

# An exercise in Discovery, Building Docker Images, using Makefiles & Docker Compose. – Part 5

## Docker images, Multi stage builds, Makefiles Projects with sub directories and their own Makefiles.

<Build 5 HDFS servers>

Discuss entrypoint.sh and run.sh’s commands per directory.

**My Repo’s**

All the code used during this article will be available on the below GIT repo.

[Building Docker Images](ttps://github.com/georgelza/dockerimagebuilding.git)

------------------------------------------------------------------------------------------------------------------

# An exercise in Discovery, Building Docker Images, using Makefiles & Docker Compose. – Part 6

## Docker images, Multi stage builds, Makefiles Projects with sub directories and their own Makefiles.

<section>

**My Repo’s**

All the code used during this article will be available on the below GIT repo.

[Building Docker Images](ttps://github.com/georgelza/dockerimagebuilding.git)

**About Me**

I’m a techie, a technologist, always curious, love data, have for as long as I can remember always worked with data in one form or the other, Database admin, Database product lead, data platforms architect, infrastructure architect hosting databases, backing it up, optimizing performance, accessing it. Data data data… it makes the world go round.

In recent years, pivoted into a more generic Technology Architect role, capable of full stack architecture.

[George Leonard](https://www.linkedin.com/in/george-leonard-945b502/)

[georgelza@gmail.com](mailto:georgelza@gmail.com)

------------------------------------------------------------------------------------------------------------------

# An exercise in Discovery, Building Docker Images, using Makefiles & Docker Compose. – Part 7

## Docker images, Multi stage builds, Makefiles Projects with sub directories and their own Makefiles.

<section>

**My Repo’s**

All the code used during this article will be available on the below GIT repo.

[Building Docker Images](ttps://github.com/georgelza/dockerimagebuilding.git)

**About Me**

I’m a techie, a technologist, always curious, love data, have for as long as I can remember always worked with data in one form or the other, Database admin, Database product lead, data platforms architect, infrastructure architect hosting databases, backing it up, optimizing performance, accessing it. Data data data… it makes the world go round.

In recent years, pivoted into a more generic Technology Architect role, capable of full stack architecture.

[George Leonard](https://www.linkedin.com/in/george-leonard-945b502/)

[georgelza@gmail.com](mailto:georgelza@gmail.com)

------------------------------------------------------------------------------------------------------------------

# An exercise in Discovery, Building Docker Images, using Makefiles & Docker Compose. – Part 8

## Docker images, Multi stage builds, Makefiles Projects with sub directories and their own Makefiles.

<section>

**My Repo’s**

All the code used during this article will be available on the below GIT repo.

[Building Docker Images](ttps://github.com/georgelza/dockerimagebuilding.git)

**About Me**

I’m a techie, a technologist, always curious, love data, have for as long as I can remember always worked with data in one form or the other, Database admin, Database product lead, data platforms architect, infrastructure architect hosting databases, backing it up, optimizing performance, accessing it. Data data data… it makes the world go round.

In recent years, pivoted into a more generic Technology Architect role, capable of full stack architecture.

[George Leonard](https://www.linkedin.com/in/george-leonard-945b502/)

[georgelza@gmail.com](mailto:georgelza@gmail.com)

------------------------------------------------------------------------------------------------------------------

# An exercise in Discovery, Building Docker Images, using Makefiles & Docker Compose. – Part 9

## Docker images, Multi stage builds, Makefiles Projects with sub directories and their own Makefiles.

<Makefile>

**My Repo’s**

All the code used during this article will be available on the below GIT repo.

[Building Docker Images](ttps://github.com/georgelza/dockerimagebuilding.git)

**About Me**

I’m a techie, a technologist, always curious, love data, have for as long as I can remember always worked with data in one form or the other, Database admin, Database product lead, data platforms architect, infrastructure architect hosting databases, backing it up, optimizing performance, accessing it. Data data data… it makes the world go round.

In recent years, pivoted into a more generic Technology Architect role, capable of full stack architecture.

[George Leonard](https://www.linkedin.com/in/george-leonard-945b502/)

[georgelza@gmail.com](mailto:georgelza@gmail.com)

------------------------------------------------------------------------------------------------------------------

# An exercise in Discovery, Building Docker Images, using Makefiles & Docker Compose. – Part 10

## Docker images, Multi stage builds, Makefiles Projects with sub directories and their own Makefiles.

<Multistage builds>

**My Repo’s**

All the code used during this article will be available on the below GIT repo.

[Building Docker Images](ttps://github.com/georgelza/dockerimagebuilding.git)

**About Me**

I’m a techie, a technologist, always curious, love data, have for as long as I can remember always worked with data in one form or the other, Database admin, Database product lead, data platforms architect, infrastructure architect hosting databases, backing it up, optimizing performance, accessing it. Data data data… it makes the world go round.

In recent years, pivoted into a more generic Technology Architect role, capable of full stack architecture.

[George Leonard](https://www.linkedin.com/in/george-leonard-945b502/)

[georgelza@gmail.com](mailto:georgelza@gmail.com)

And that’s it for now… Thank you for sticking with me through this exploration. All it did was create a small little to do list that is growing as I am typing this, of things/subjects I’d like to explore more and blog . Till next time.



**About Me**

I’m a techie, a technologist, always curious, love data, have for as long as I can remember always worked with data in one form or the other, Database admin, Database product lead, data platforms architect, infrastructure architect hosting databases, backing it up, optimizing performance, accessing it. Data data data… it makes the world go round.

In recent years, pivoted into a more generic Technology Architect role, capable of full stack architecture.

[George Leonard](https://www.linkedin.com/in/george-leonard-945b502/)

[georgelza@gmail.com](mailto:georgelza@gmail.com)

Some more References:

[Apache Flink](https://flink.apache.org/) originally by [Ververica](https://docs.ververica.com/)

* [Get Started - Installation](https://docs.ververica.com/vvp/getting-started/installation?_gl=1*a1ub31*_gcl_au*MjI3NTE0OTU0LjE3MjA2OTY4NDY).