­­SOFT8026 – Data-driven Microservices

# Lab 1 – Installing Docker and Running some Containers

This is likely to be a short lab, just to get you warmed up and ready to go for next week.

We will explore Docker and containerization shortly in the lectures. Containers are almost like little lightweight mini virtual machines that are designed primarily to run single services (such as a database, web service or small web application). That’s as much as we need to know for the moment.

However, in this tutorial, you will install Docker and get it up and running with a simple dummy application.

Note: you have options here, including using Windows, installing Linux on a VM (e.g. using VirtualBox), and so on. This document describes using Ubuntu on a VM.

Begin by visiting the following webpage:

<https://docs.docker.com/install/>

You will be installing Docker Engine on Linux using a repository (this is usually more reliable than the package approach with .deb files). While there is a desktop version for Windows and Mac, I am assuming you are using Linux, probably using a VM, and probably an Ubuntu-based one like Ubuntu, Mint, Xubuntu or Kubuntu.

Assuming you are using Ubuntu-based Linux, proceed to the installation guide at <https://docs.docker.com/engine/install/ubuntu/>

When you run the hello world application, you should see:

larkin@larkin-kubuntu:~/Downloads$ sudo docker run hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
1b930d010525: Pull complete   
Digest: sha256:2557e3c07ed1e38f26e389462d03ed943586f744621577a99efb77324b0fe535  
Status: Downloaded newer image for hello-world:latest  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
 1. The Docker client contacted the Docker daemon.  
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
    (amd64)  
 3. The Docker daemon created a new container from that image which runs the  
    executable that produces the output you are currently reading.  
 4. The Docker daemon streamed that output to the Docker client, which sent it  
    to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
 $ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
 <https://hub.docker.com/>  
  
For more examples and ideas, visit:  
 <https://docs.docker.com/get-started/>  
  
larkin@larkin-kubuntu:~/Downloads$

Take a moment to examine the highlighted text. It first checks locally (in our repository) for the latest version of hello-world (we didn’t specify a version, so latest is defaulted); when that isn’t found, it downloads from Docker Hub (see <https://hub.docker.com/_/hello-world>). Try running the command again – you won’t see the highlighted text, because the image is now local. The rest of the text is just the output from the program.

We will delve deeper into Docker during the lectures and it will feature a lot in the labs. However, if you are curious, you can examine the GitHub repository of the hello-world application at <https://github.com/docker-library/hello-world>. The text is generated by running a binary compiled from the C program: <https://github.com/docker-library/hello-world/blob/master/hello.c>

# Post-installation Steps

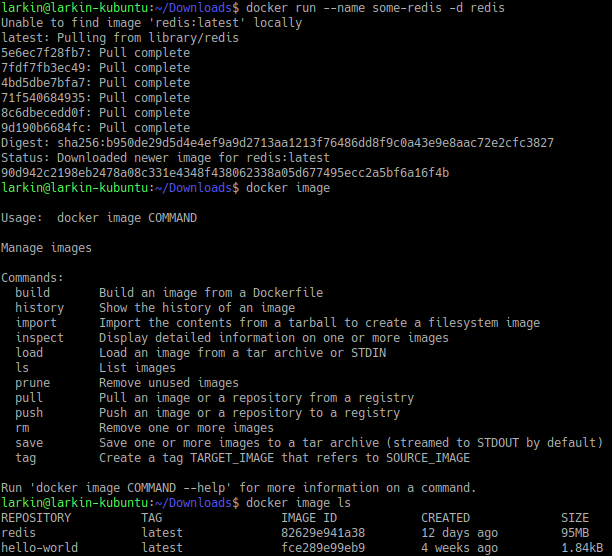
Continue to the post-installation steps at <https://docs.docker.com/install/linux/linux-postinstall/>. For now, only follow the steps under “Manage Docker as a non-root user”. This will remove the need to always use sudo when issuing docker commands.

Don’t worry if the docker group already exists.

# Redis Container

The hello-world example shows us a simple application running in a container on the local machine. We can run just about anything in its own container, including a database. Go to the following webpage: <https://docs.docker.com/samples/library/redis/>

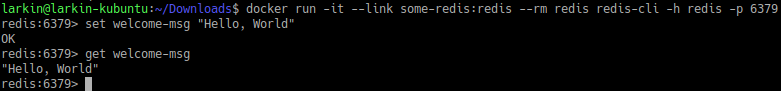
Ignore the “start with persistent storage” option for now (this means that when the container is removed, so will all the data we create). Run the commands as per the screenshot below.



We now have a running container that was created from a local image (which was built 12 days ago – or however many it says for you).

Don’t try to “connect to it from an application“ – we are not at the point of container-to-container communication yet. Instead, use redis via a local redis-cli client:

docker run -it --link some-redis:redis --rm redis redis-cli -h redis -p 6379



Don’t worry about all the parameters above. The main one for now is -it which give us an interactive terminal so that we can enter commands and get standard output.

# Suggested Exercise Work

With any time remaining, you can go and explore the images available to you. Go to <https://hub.docker.com/> and create an account (if you don’t already have one). Think of services you’ve used in previous projects, e.g. mysql, postgres, mongodb, and search for them. Read the instructions and see if there is an example with -it that allows you to test it out.

You could investigate the “busybox” image and learn how to ping other containers as a way to verify they are responsive.