FIT5202 - Data Processing For Big Data

Lab 01 Activity: Setting up your environment

In this activity, we will learn how to set up the environment and make it ready for processing big data. Big data processing employs many open-source software/libraries to form a software stack. It's a challenge to ensure the compatibility of different libraries/versions from various sources. In this unit, we will use container technology (Docker) to manage the environment, and a pre-built and optimised image has been provided with the following software:

1. **Python (3.10) as a programming language**
   Python 3.11 and above version is not compatible with some dependencies as of the time of this document (Jul/2024)
2. **Jupyter Lab/Jupyter Notebook as an IDE for Python development.**
   The docker image has both the Jupyter lab and notebook pre-installed. Generally, Jupyter Lab is recommended since it's a newer version of Jupyter Notebook and provides many useful features like in-built file management, CSV viewer, terminal emulator, etc. However, due to backward compatibility issues, a few known bugs exist in Jupyter Lab when you try to use real-time visualisation (e.g., plotting steam data in real-time).
3. The latest Apache Spark (3.5.0) as a big data processing and analysis tool
4. Apache Kafka as the tool for streaming
5. (Optional commonly used libraries) Scikit Learn/Numpy/MatplotLib(latest version as of Nov/2023)

We will install Docker Desktop first, then follow the steps to use Jupyter Notebooks with it. As this is a big data unit, we will only learn basic Docker commands to manage our environment.

Week/Session 1: Docker Installation and Environment Setup

Step 0: System Requirements

Big data processing and machine learning are very demanding for computing resources. Having a desktop/laptop with a faster CPU and larger RAM will speed up some lab activities and machine learning training.

Minimum Requirement:
A Windows 10/11 or Linux or MacOS laptop/desktop that is less than 5 years old, with a minimum of 8GB RAM and a solid-state drive (SSD).
(Note: While the assignments are doable with 8GB RAM, some students may frequently encounter out-of-memory (OOM) kill issues.)

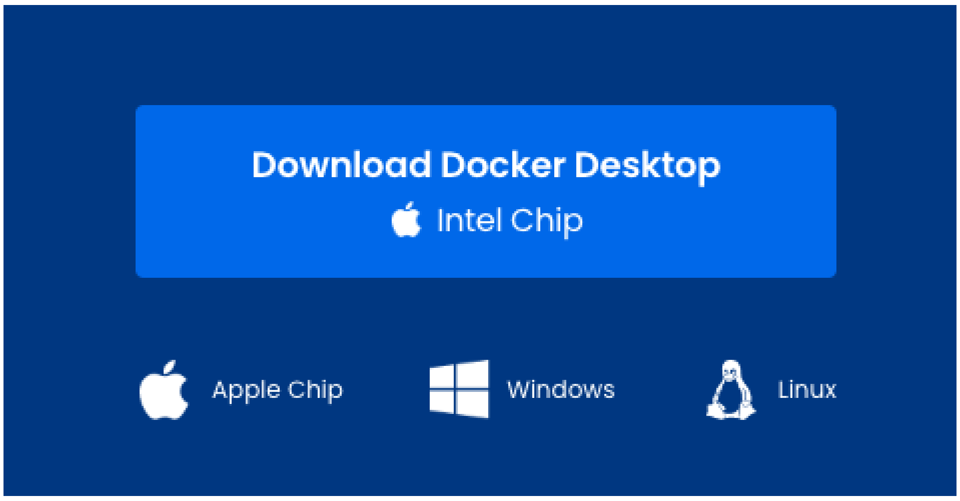
Recommendations (optional, not a hard requirement):
Less than 3 years old computer with 16GB RAM, 256GB NVME SSD.

For Windows/Linux laptops: Avoid "U" series CPUs, get standard or "H"/"HK" version if possible.
For Mac, M1/M2/M3 is recommended (instead of old Intel ones).

Step 1: Installing Docker Desktop

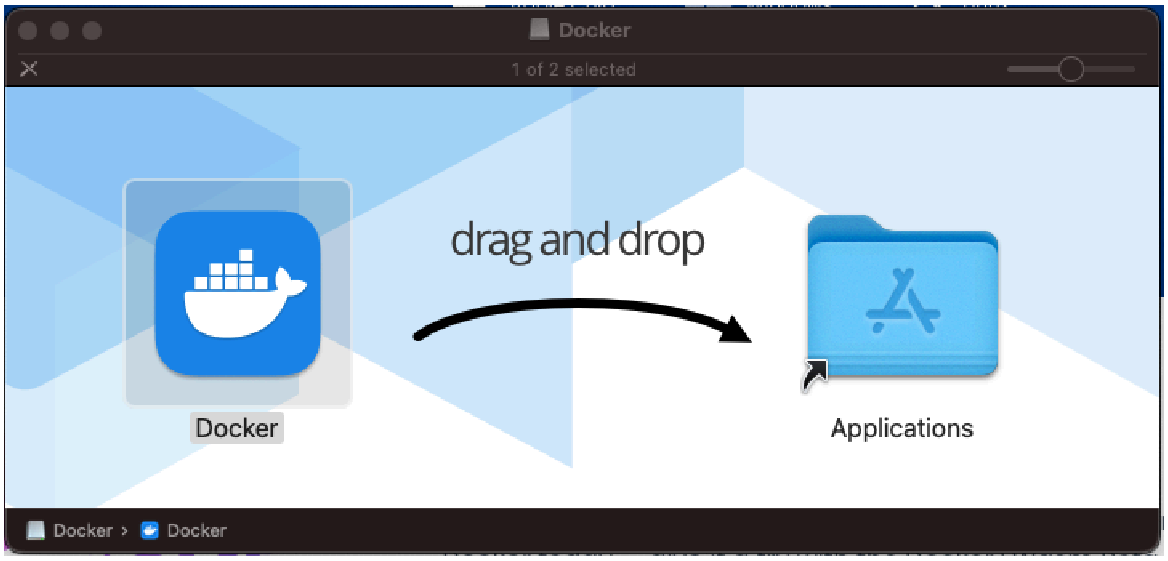
a) Download Docker Desktop

Go to https://www.docker.com/ and download the version appropriate for your operating system.
Note: Mac has two different versions, one for Intel CPU and the other for M1/M2/M3, please download accordingly.

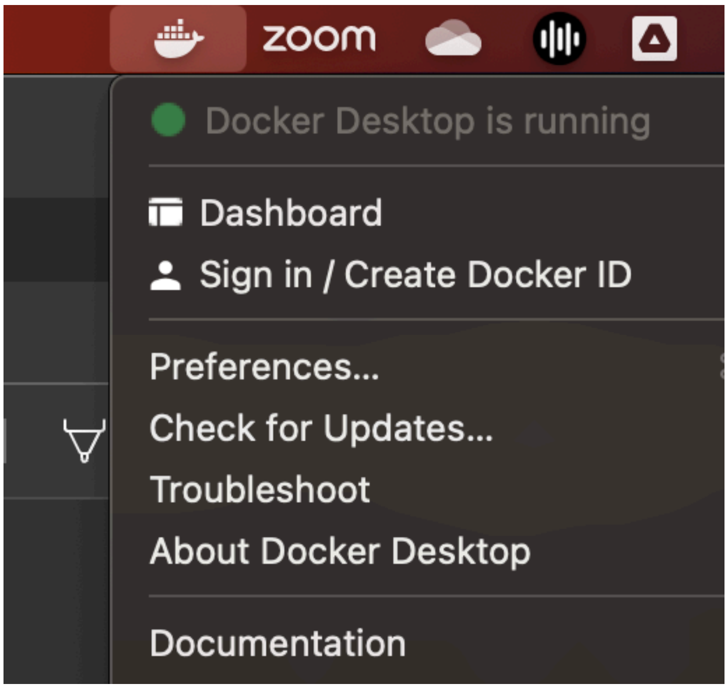


b) Installation and Tuning for MacOS

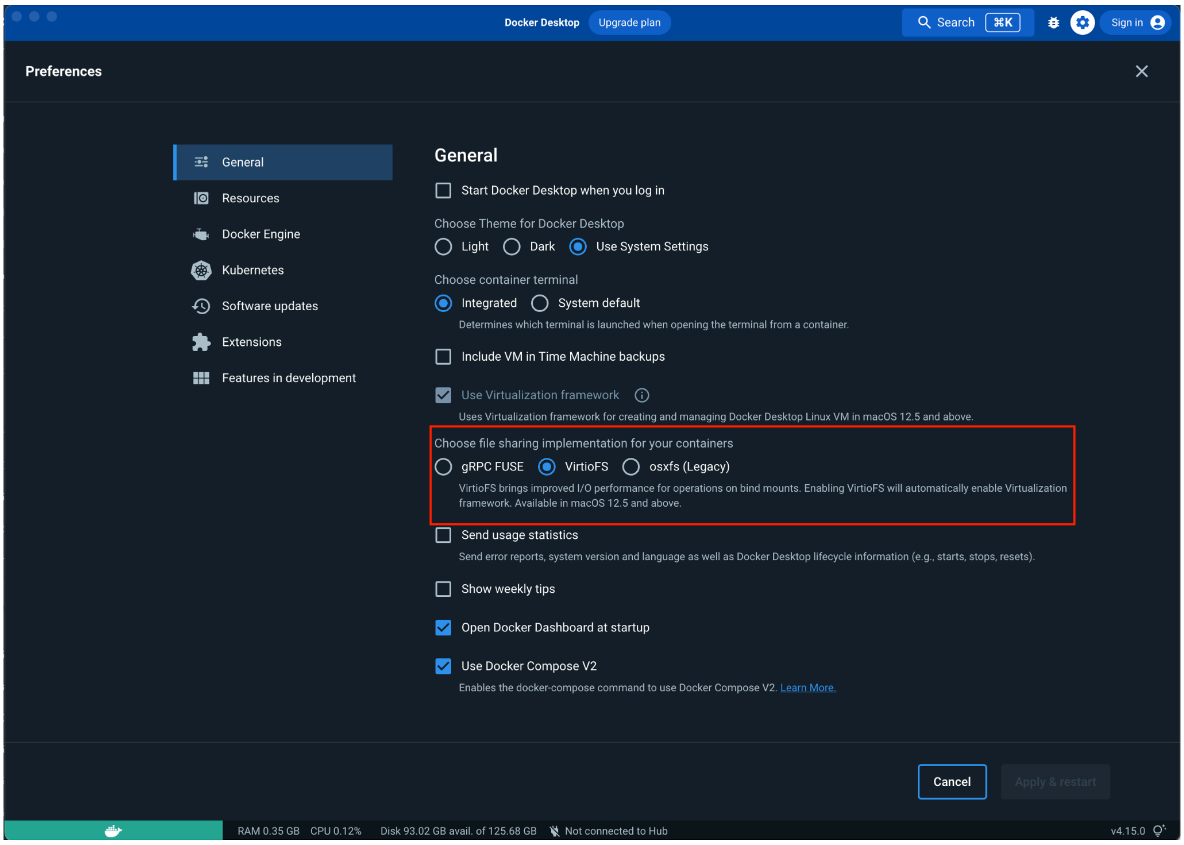
1) Double-click on the downloaded .dmg file and drag Docker to your "Applications".



2) Find "Docker" from your application and start it, then click on the Docker icon and select "Preferences".



3) In the "General" tab, change file sharing implementation to "VirtioFS". This option will improve IO performance on bind mounts.


Preferences

General

- [ ] Start Docker Desktop when you log in

Choose Theme for Docker Desktop
Light Dark [✓ Use System Settings]

Choose container terminal
[✓ Integrated] System default
Determines which terminal is launched when opening the terminal from a container.

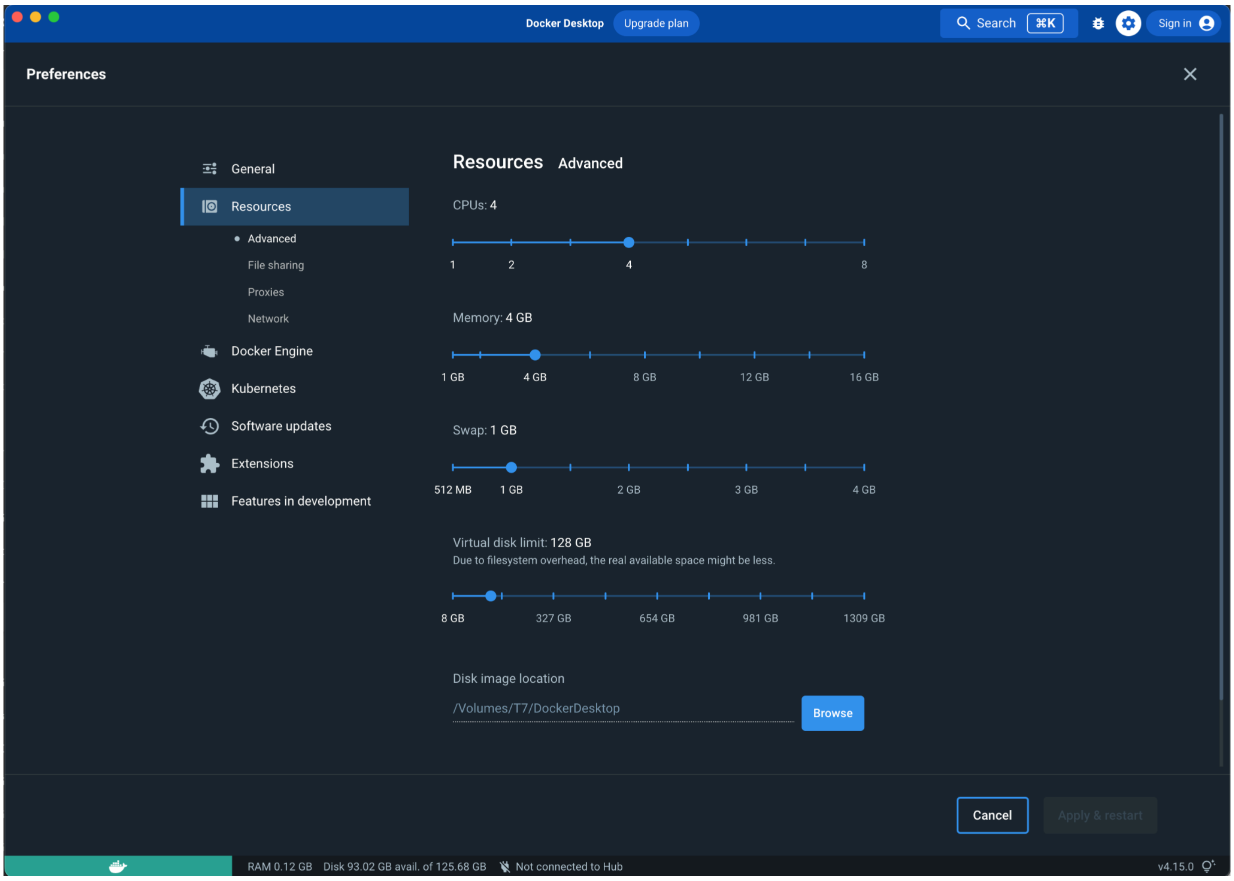
- [ ] Include VM in Time Machine backups
- [x] Use Virtualization framework
Uses Virtualization framework for creating and managing Docker Desktop Linux VM in macOS 12.5 and above.

Choose file sharing implementation for your containers
gRPC FUSE [✓ VirtioFS] osxfs (Legacy)
VirtioFS brings improved I/O performance for operations on bind mounts. Enabling VirtioFS will automatically enable Virtualization framework. Available in macOS 12.5 and above.

- [ ] Send usage statistics
Send error reports, system version and language as well as Docker Desktop lifecycle information (e.g., starts, stops, resets).
- [ ] Show weekly tips
- [x] Open Docker Dashboard at startup
- [x] Use Docker Compose V2
Enables the docker-compose command to use Docker Compose V2. Learn More.

RAM 0.35 GB CPU 0.12% Disk 93.02 GB avail. of 125.68 GB Not connected to Hub v4.15.0

4) In the "Resource" tab, please change CPUs, Memory and Virtual disk limits depending on your laptop specification. We recommend using 8GB RAM for Docker if possible.


Preferences

Resources Advanced

CPUs: 4

Memory: 4 GB

Swap: 1 GB

Virtual disk limit: 128 GB
Due to filesystem overhead, the real available space might be less.

Disk image location
/Volumes/T7/DockerDesktop

RAM 0.12 GB Disk 93.02 GB avail. of 125.68 GB Not connected to Hub v4.15.0

c) Windows 10/11

1) Prerequisites: Window Subsystem for Linux (WSL 2) is required. Please follow the instructions from Microsoft: [Install WSL | Microsoft Learn]
2) (Optional) Click on "Start", find "Microsoft Store" and search for Ubuntu 22.04(LTS). This will install a base Linux distribution for Docker.
3) Double-click on the .exe file and follow the instructions, all default settings work fine.
4) (Optional) Please refer to steps 3 and 4 for performance tuning.

d) Linux

Docker engine is recommended instead of Docker Desktop. Please follow the instructions depending on your distribution.

Ubuntu: [Install Docker Engine on Ubuntu | Docker Documentation]
Debian: [Install Docker Engine on Debian | Docker Documentation]
CentOS: [Install Docker Engine on CentOS | Docker Documentation]

Step 2: Running Docker Image

After installing Docker, start "Docker Desktop" and open a **terminal/Window command line** and run the following command (works for all platforms).

1) Create a docker network
   docker network create fit5202

2) Run the docker container.
   Change the **red** part to your local folder containing notebook files.

docker run --network fit5202 -v /home/jay/fit5202/labs:/home/student -p 5202:5202 -p 4040:4040 monashfit/fit5202-pyspark jupyter notebook

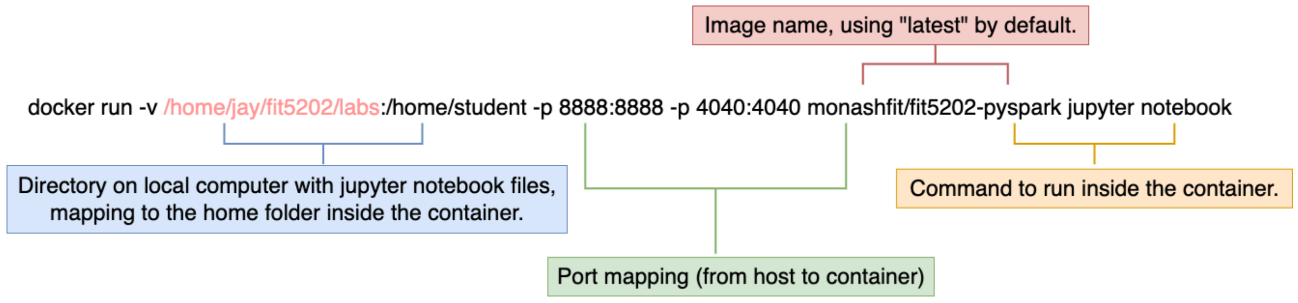


Image name, using "latest" by default.
docker run -v /home/jay/fit5202/labs:/home/student -p 8888:8888 -p 4040:4040 monashfit/fit5202-pyspark jupyter notebook
Directory on local computer with jupyter notebook files, mapping to the home folder inside the container.
Command to run inside the container.
Port mapping (from host to container)

The above diagram explains different parameters (note: port 8888 in the figure is the default port, we changed it to 5202 to avoid conflict.)

Parameters:

1) -v(blue colour): Your local folder to be binded to the home directory inside the container. Please note Windows path needs to include the drive letter and use \. For example:
   docker run **--network fit5202** ***-v D:\5202_docker\labs:/home/student -p 5202:5202 -p 4040:4040 monashfit/fit5202-pyspark:latest jupyter notebook***

You can change the red part to your own folder. a) Make sure this folder exists before running the command, all your jupyter notebook files will be stored in this folder. b) Avoid spaces, and special or non-English characters in your folder name.

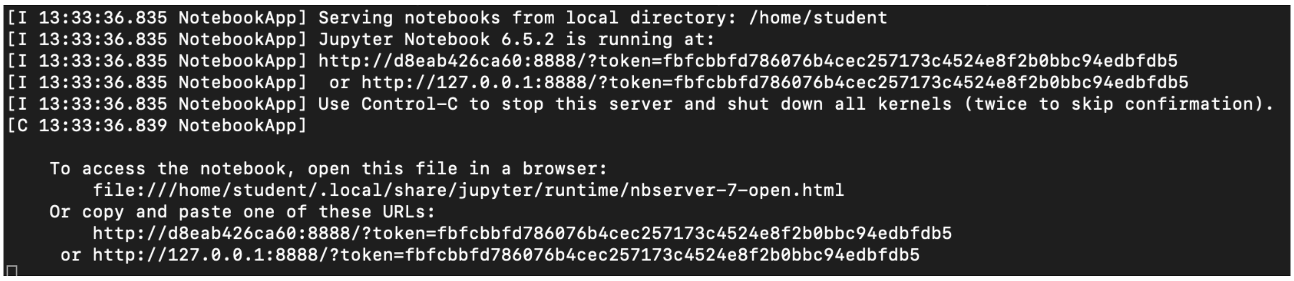
2) -p (5202 and 4040): Port mapping from host to container, 5202 is the default port we use for Jupyter notebook and 4040 is the default port for Spark UI. The left port number before: is host port, the right side is the container port.
3) “jupyter notebook”: execute jupyter notebook inside the container.

The “docker run” command is only required for the first time starting a container. When you finish using it, you can press **Ctrl+C** or use “docker stop [container id]” to stop it and “docker start [container]” to start it again later on.

To remove a container, use “**docker rm [container id]**”.

Step 3: Access Jupyter Notebook in Browser

If the above steps are successful, you will see the outputs similar to the following picture on your terminal.



```
[I 13:33:36.835 NotebookApp] Serving notebooks from local directory: /home/student
[I 13:33:36.835 NotebookApp] Jupyter Notebook 6.5.2 is running at:
[I 13:33:36.835 NotebookApp] http://d8eab426ca60:8888/?token=fbfcbbfd786076b4cec257173c4524e8f2b0bbc94edbfdb5
[I 13:33:36.835 NotebookApp]  or http://127.0.0.1:8888/?token=fbfcbbfd786076b4cec257173c4524e8f2b0bbc94edbfdb5
[I 13:33:36.835 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 13:33:36.839 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/student/.local/share/jupyter/runtime/nbserver-7-open.html
    Or copy and paste one of these URLs:
        http://d8eab426ca60:8888/?token=fbfcbbfd786076b4cec257173c4524e8f2b0bbc94edbfdb5
     or http://127.0.0.1:8888/?token=fbfcbbfd786076b4cec257173c4524e8f2b0bbc94edbfdb5
```

Please copy and paste the URL including the token to your preferred web browser.

Step 4: Using Jupyter Notebook

Download the Jupyter notebook files from Moodle, and either:
1) Upload the files to Jupyter notebook
Or 2) Copy the files to the binded local folder on your laptop.

Now that you have a working environment, please continue exploring the Jupyter notebooks from Moodle.

Howtos and FAQs

- ### How to view running/stopped containers?

  To view running containers, you can use “**docker ps**”.
  To view all containers including stopped ones, the command is “**docker ps -a**”.

- ### What is a shorthand?

  When you execute the “docker ps” command, you will see a long container id. It is required to perform some commands. (e.g. docker stop [container id]). Since we don’t want to type the long ID every time, you can use the initial 1-2 characters instead of the whole ID, as long as Docker can identify the unique container with it.

- ### Do we have to run the “docker run …” command every time?

  No. To stop a container, the command is “docker stop [container id]”; or, it will be automatically stopped when you shut down your computer. The next time you need your container, run “docker start [container id]”.

- ### How do I get the token after stop/start?

  For security reasons, jupyter will generate a new token if the session is timed out. (e.g. stop a container and start it again after more than 30 minutes). To view the new token, the command is “docker logs [container id]”; or you can “attach” to the running container.

- ### How to delete containers and clean up?

  You can use “docker rm [container id]” to remove containers that are no longer required. It won’t delete your volume/data in case you still need them. To perform a cleanup, the command is “**docker system prune**”.

- ### My token doesn’t work.
  1. Make sure you copy the whole token only. Sometimes you may copy the whitespace/line break character without noticing.
  2. Check for port conflict. If you have another local Jupyter Notebook/pyspark instance running, it may be using port 5202 and 4040. Solution: Change the listening port to another free port.

Week/Session 9: Zookeeper and Kafka in Docker

Based on the separated responsibility principle, we will need two more containers for Spark data streaming.

1. Zookeeper
   Command: **docker run --network fit5202 --name zookeeper -d -p 2181:2181 monashfit/fit5202-zookeeper**
   According to the Zookeeper document: “ZooKeeper is a centralised service for maintaining configuration information, naming, providing distributed synchronisation, and providing group services. All of these kinds of services are used in some form or another by distributed applications. “ Kafka uses zookeeper to store some of its critical data, hence, Zookeeper needs to be started first.
2. Kafka

```
docker run --network fit5202 --name kafka -d -e
KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181 -e
KAFKA_ADVERTISED_HOST_NAME=kafka -p 9092:9092 monashfit/fit5202-kafka
```

Both containers will use the UTC timezone by default. To specific a timezone, an additional environment variable can be used, for example: **`-e TZ=Australia/Melbourne`**