



# CS 6220 Data Mining — Lecture 2

## January 18, 2023 (Week 2)

---

### 1 Docker

Different companies use different tools for development and different work environments. For future assignments, we won't be prescriptive, but in this homework, we're going to familiarize ourselves with some of the most useful and common delivery and development environment tools in industry today.

Docker <http://www.Docker.com> is a useful mechanism for delivering software or scaling it up. For example, say we want to run a multi-computer job, passing *Docker containers* to each of the nodes in the cluster is one way to have repetitive and predictable behavior when doing large scale compute.

There are two essential Docker units: a **container** and a **container image**.

1. A **container** is a sandboxed process on your machine that is isolated from all other processes on the host machine. That isolation leverages kernel namespaces and cgroups, features that have been in Linux for a long time. Docker has worked to make these capabilities approachable and easy to use. To summarize, a container:
  - a) is a runnable instance of an image. You can create, start, stop, move, or delete a container using the DockerAPI or CLI.
  - b) can be run on local machines, virtual machines or deployed to the cloud.
  - c) is portable (can be run on any OS).
  - d) is isolated from other containers and runs its own software, binaries, and configurations.
2. When running a container, it uses an isolated filesystem. This custom filesystem is provided by a **container image**. Since the image contains the container's filesystem, it must contain everything needed to run an application - all dependencies, configurations, scripts, binaries, etc. The image also contains other configuration for the container, such as environment variables, a default command to run, and other metadata.

Go ahead and download and install Docker. The getting started guide on Docker has detailed instructions for setting up Docker on

- Mac <https://docs.docker.com/desktop/install/mac-install/>,
- Linux <https://docs.docker.com/install/linux/docker-ce/ubuntu>
- Windows <https://docs.docker.com/docker-for-windows/install>.

## 2 Executing Your “Hello World”

For this lab, we’ll start with creating a Dockerfile in your submission folder. Specify the operating system and version of Python in the Dockerfile. You will subsequently need to install Python and libraries that you anticipate importing. Do not add the data into the image; you will need to pass that into the container with the `-v` Docker option.

For example, here’s the most basic Dockerfile:

```
FROM ubuntu:20.04

RUN apt update && apt install -y sbcl
WORKDIR /usr/src
```

For this assignment, you’ll set up your Docker environment and the appropriate versions of Python. Specifically,

1. Download and install Docker
2. Create your Dockerfile
3. Compile your Docker image
4. Screenshot a list of the Docker images available
5. Screenshot a list of the running Docker containers that include one with the image you created
6. Include both screenshots and the command you used in your write up

## 3 Specifying Ports and Storage

At times, you may need several resources from your host machine. For example, there may be some files that you’d like to grab or whole folders that you would like to access. Or, there may be instances where you’d like for the container to be exposed to network traffic.

### 3.1 Volumes

Let’s say there are some files that you wish to be made available to the container that you’re running. There are a couple of ways to do so. You can either make them available:

- When you’re specifying the container
- When you’re starting a new container

In this section, we’ll make a folder available with the `-v` flag.