Cloud Computing Spring 2024
Homework 1: Getting familiar with virtual machines and containers

Deadline: Monday, March 11, 2024

1 Aim and Scope

The aim of this homework is for you to become familiar with the usage of virtual machines and containers. You will install Linux in a virtual machine and will use Podman as a container engine. As the first assignment will be based on the use of containers, you can consider this homework as an initial practice and preparation for the first assignment. Note that this homework is to be completed and submitted *individually*. The learning goals are:

- To set up a fully-fledged basic (Linux) virtual machine. (It is *not* allowed to use WSL or WSL2).
- To build a container image provided source and Containerfile.
- To clone a container image from the web.

2 Tasks

To complete this homework you are asked to complete the tasks listed below and to write up a (very) short report (max. 1 – 2 pages A4). If you need additional information or help for some of these steps, refer to the section *Tips & Tricks* at the end of the document, look for the information yourself on the Internet, or ask the Teaching Assistant (TA). If you contact the TA, we expect that you have done an investigation of the problem yourself first.

1. Create a virtual machine and install a Linux distribution of choice. As hypervisor you can consider Linux KVM/Qemu, VirtualBox, VMWare Desktop/Player, etc., but do note that WSL and WSL2 are excluded from this list. We strongly suggest to *not* install a graphical desktop environment (GUI) within the virtual machine as this will save a lot of disk space. Instead, connect to the created virtual machine using `ssh` to continue working in the virtualized Linux environment. In this case, a disk image of 5 GiB and 2 GiB of RAM (main memory) should be sufficient.
   For a Linux distribution we suggest to pick from Rocky Linux 8, Ubuntu 22.04 (minimal), Debian 11 or Fedora.

2. Within the virtual machine install Podman using the distribution's package manager (this would likely either be `dnf` or `apt-get`). **Important:** also install `podman-plugins` on Rocky or Fedora based systems.

3. As `root`, create a Podman network with a name of choice.

4. Download the source and Container file of the *server* from BrightSpace. Use this to build a container image and to run this as a container (again as `root`!). Give the running container the name `ccoservertest` (important!). Do not forget to run this container within the Podman network created in the previous step.

5. Using Podman (and as `root`), pull the client container from Docker hub at location: `docker.io/liacsrietveld/ccoclient:2024`.

6. Run the client container using Podman, as `root`, and attached to the Podman network created in a previous step. The client container will connect to the server container you have created.

7. If the client container runs successfully and prints "`obtained hash:`" together with a hash of 80 characters in length, you have completed all the tasks and can write up the report (see next Section). Note that if the client container prints "`connection to webserver failed`", your setup is *not* working as expected.

3 Report

Your report can be short (max. 1 – 2 pages A4) and should contain the following:

- Your name and student ID.
- The name and version of the Linux distribution that you have installed.
- A *screenshot* of the output of the command `hostnamectl` when run on your Linux virtual machine.
  (In case that command does not exist, please include the output of `lshw -class system` and `cat /etc/os-release` in your report).
- The Podman command that you used to *run* the **server**.
- A *screenshot* of the output of: `podman image tree ccoclient`.
- A *screenshot* of the output of running the client container, including the hash that was printed. This is the hash printed after "`obtained hash:`" of 80 characters in length, *not* a hash as given by the container engine. Please also include the hash in your report as verbatim text that can be cut and pasted by us.
  (*Privacy notice: the hash does not contain any identifying information. It does contain a timestamp when it was created.*)

4 Submission

The homework must be worked on and a report submitted *individually*. The report must be submitted through BrightSpace. Make sure your report includes name and student ID. Submission of the homework is mandatory for completion of the course. The submitted homeworks will be assessed with a sufficient/non-sufficient mark.

Deadline: Monday, March 11, 2024

A Tips & Tricks

This section contains a number of tips and tricks to help you get started with Podman. The assignment should work with any of the Linux distributions we have mentioned in the section *Tasks*. We have verified it works on Rocky Linux 8 and Debian 11. Note that we strongly suggest to create and run the containers as the `root` user, as this is required for the container networking to work smoothly.

A.1 Installing *podman*

The command required to install Podman depends on the Linux distribution you have chosen:

- **Rocky or Fedora Linux**: `dnf install podman podman-plugins`
- **Debian or Ubuntu Linux**:
  `apt-get install podman golang-github-containernetworking-plugin-dnsname`

The `dnsname` plugin is required for the container interworking component of the assignment to work. *Debian 11 note:* in order for this plugin to work out of the box, the following command needs to be executed (once) before creating the first Podman network:
`dpkg-divert --divert /usr/lib/cni/dnsname --rename /usr/lib/dnsname`.

A.2 Commands to create container images

podman can create container images using Dockerfiles or Containerfiles. This can be achieved using the `podman build` command. Make sure to use the `-t` option to give the container image a name (or tag).

A.3 Creating Podman networks

Refer to the `podman network create` command.

A.4 Starting container instances

From images, container instances can be started. Multiple container instances can be started from a single image. Container instances are started using the *podman run* command. Note that this will not create full copies of the container image, rather, the image remains read-only and an overlay file system is placed on top of it to catch any writes.

The *podman run* command has a variety of useful options: `--name` to give the container instances a recognizable name, `--rm` to delete the container instances after shutdown (the source container image will remain!), `--net` to attach the container to an existing Podman network. You can also control whether the container process should remain in the foreground (for interactive programs) or whether this process should be run in the background (for servers/daemons). The option `-d` will run the container in the background, `-it` will run it in interactive terminal mode.

Refer to the Podman documentation and manual pages for more detailed information about these command line options.

Other useful commands are `podman ps` to inspect the currently active containers and `podman stop` to stop a container (which is running in the background).