Installing CURATE via Docker

Introducing Docker – Why does CURATE use it?

CURATE, and many other bioinformatic tools alike, are intended to be running on Linux-based systems or distributions. However, not all systems are Linux-based, and the code may not run properly unless all the libraries, system files, etc., are installed. Moreover, the code is not guaranteed to work flawlessly without system-based errors. This poses a major setback when it comes to ensuring that the pipeline runs on different machines. **Docker** is a software that sets the industry standard for ensuring that applications can run perfectly on different systems – by setting an "isolated environment" where pipelines and code scripts can be executed without having to set up your entire system from scratch.

As of v1.0, CURATE supports **Containerization** – a process that packages altogether the dependencies (code, runtime, system tools, filesystem, and settings) into a single unit called a **Image**, which is necessary for it to run.

This short guide will provide you a walkthrough for setting up Docker and an environment Image for the first time and show some important nuts and bolts in the process.

1. Quick installation

This guide assumes that your system has Docker and Git installed! If not – please refer to the official Docker's website and download Docker, and to Git's official website and download it too.

The installation process is simple and requires 2 steps:

1. Clone the latest repository on CURATE's official GitHub documentation:
   ```
   git clone https://github.com/user/ILYAKAZHDAN/CURATE.git
   ```
   - After cloning, place the contents in a directory of your choosing
   - Then, CD to that directory
2. Build a Docker Image:
   ```
   docker build -t CURATE_env [directory path]
   ```

Once installed, start **Docker Desktop**, and then run CURATE in your Terminal (Linux):

```
docker run --rm -it -v $(pwd):/work CURATE_env bash CURATE_v1.0.sh
```

If you are on Windows PowerShell, use this command to run CURATE:

```
docker run --rm -it -v "${PWD}:/work" CURATE_env bash CURATE_v1.0.sh
```

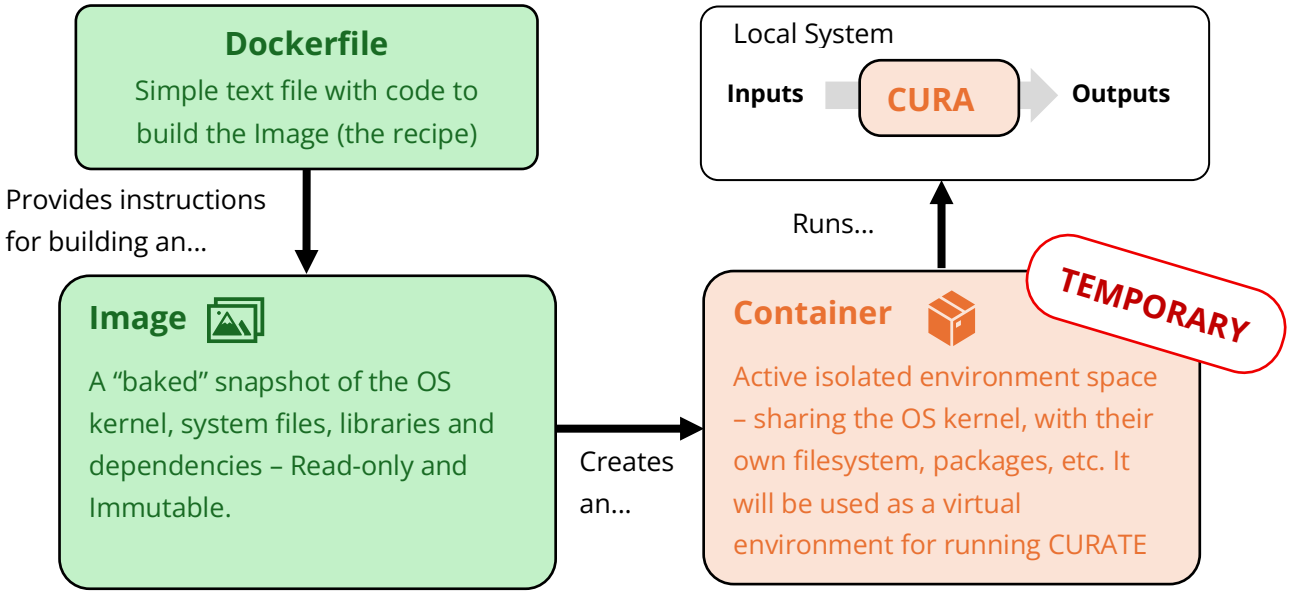
That's about it! Enjoy ☺! To learn more, dive into **2. How does Docker work?**

2. How does Docker work?

Docker provides the foundations for CURATE (the application) to work. It does this laying down three essential compartments – **Dockerfile**, an **Image** and **Container** that are in interplay with one another:

- **Dockerfile** – a text file with code – which instructs Docker to build an **Image**. It contains commands that build up the OS kernel, environment variables, system files, and install the dependencies that the CURATE needs to run.
- **Image** – The outcome of the instructions given by **Dockerfile**, containing layers that are stacked on top of one another – corresponding to the order of subsequent instructions. It essentially holds a snapshot of all the required system files, libraries, binaries, and default filesystem – all of which are Immutable (unchangeable.)
- **Container** – Mirrors all the files from the **Image**, and uses them as temporary files to create an active state – the working ground for the application. After the application is done running, the associated **Container should be removed** as a clean-up measure (to avoid accumulation of temporary files.)

We will get started by introducing two central core concepts: Images and Containers



Optional for advanced users – If you would like to create your own Docker image from scratch and then run it, you can do so – by referring to 3. Nuts and Bolts of Docker.

3. Nuts and Bolts of Docker

3.1. Build an Image from Scratch

Building an Image is preceded by making a **Dockerfile** – the instructions (recipe) "booklet" in order to create the essential environment that CURATE can run on.

1. Start by creating a **Dockerfile** with the following code:

```
## CURATE Dockerfile ##
# This Dockerfile contains the instructions code for building a Docker image that
# carries out the installation of the required packages for running CURATE.

# 1) Selects a base image from mambaorg/micromamba:1.5.10
# This creates a minimal Linux-based system (Debian distribution) with micromamba
# as the package manager for installing the required packages.
FROM mambaorg/micromamba:1.5.10

# 2) Creates a new conda environment named 'pipeline' and installs the required packages.
# The packages are installed from the conda-forge and bioconda channels using micromamba.
# Then, it cleans up the cache to save space.
RUN micromamba create -y -n CURATE_env \
    -c conda-forge -c bioconda \
    fastp \
    trimmomatic \
    bowtie2 \
    fastqc \
    blast \
    entrez-direct \
    star \
    samtools \
    seqkit \
    && micromamba clean --all --yes

# 3) Sets the PATH environment variable to make all tools available for execution
ENV PATH=/opt/conda/envs/CURATE_env/bin:$PATH

# 4) Sets the default working directory inside the container
# This is where the pipeline will be executed
WORKDIR /work

# 5) Sets the default command to run if the user doesn't specify it
# Defaults to an interactive shell in the Terminal, allowing the user
# to decide what to do inside the container
CMD ["bash"]
```

Keep in mind:

- Each line of code (not commented-out) is an instruction layer – and together, they are stacked on top of one another, resulting in a snapshot state of the OS kernel, system files, dependencies and environment variables.
- The container essentially will be executed in a virtual environment-based working path – "/work" – set as a default. No other directory is hardcoded.
- **Neither the script (CURATE_vX.X.sh), nor the inputs or outputs are integrated into Dockerfile** because the Image that stems out from the latter would not allow the user to easily manage or modify the script or the inputs or outputs at hand, as well as it would prohibit them from debugging the outputs or the script.

Installing CURATE via Docker

2. Download **CURATE_vX.X.sh** from GitHub.
3. Create a directory called **`CURATE_env`**, and place **`Dockerfile`** and **CURATE_vX.X.sh** inside.
4. Create a Docker Image called **`CURATE_env`** place inside that :

```
docker build -t CURATE_env [directory path]
```

It means: Based on the **`Dockerfile`** at [directory path], create a Docker-based Image, named **`CURATE_env`**, at [directory path]. Note: The name of the environment must be the same.

After building the image, it is stored in a "Virtual Machine" which is practically inaccessible to the user in Finder/File Explorer.

After building the image, we can browse inside of it:

```
docker run --rm -it -v $(pwd):/work CURATE_env bash

ls /work
```

- Once you entered the Docker-based environment, ensure that all packages were installed properly by typing:
  `ls /opt/conda/envs/CURATE_env/bin`

Or – execute the following command:

```
docker run --rm -it CURATE_env micromamba list -n CURATE_env
```

3.2. Run CURATE by using a created or existing Image:

5. Go to the script's working directory on your computer:

```
cd /your/path/to/CURATE_env/
```

6. Then, run it by executing:

```
docker run --rm -it -v $(pwd):/work CURATE_env bash CURATE_v1.0.sh
```

- `docker run` – The subcommand to create and start a container from an image
- `--rm` – **Removes the container automatically** when the job finishes/stops
  - This step ensures that the container – which has temporary files, is cleaned up.
  - However, the Image will stay intact – as it should.
- `-it` – "-i" (interactive), "-t" (tty, pseudo-terminal), allowing the user to use the Terminal
- `-v $(pwd):/work` – Mounts the volume – the script's working directory `$(pwd)` (on the host computer) into `/work` (on the Container)
- `CURATE_env` is the name of the image to run
- The command to run inside the container – `bash` opens a Terminal inside the container.

4. Setting up Docker's Resources on Windows

Docker is now **tightly integrated to Windows Subsystem for Linux**, making it harder to access its resource footprint on Windows. Therefore, to manage the RAM resources that the image can allocate in order to run, you must set up the resource limit on WSL first.

Before proceeding, this guide assumes that you have WSL installed on your system. If not, please install it by opening a PowerShell Terminal, and execute the following:

```
wsl --install
```

Once installed, create a new configuration file, where you will write down the cap amount of RAM that CURATE will be able to use. You can do so as follows:

```
notepad "$HOME\.wslconfig"
```

When the file opens, write down the following contents to specify the maximum number of RAM that you allocate to Docker. Replace "15GB" with a number of memory available on your system (You must use at least 15 GB for CURATE.)

```
[wsl2]

memory=15GB
```

Then, shut down WSL by running:

```
wsl --shutdown
```

Thereafter, close Docker Desktop, and reopen it. Lastly, run the following on your terminal:

```
docker info | Select-String "Total Memory"
```

This will return the Total Memory available for disposal by the Docker image. For 15 GB for example, it should show something like this:

```
 Total Memory: 14.63GiB
```

Once set up, you can run CURATE as shown on Page 1 – **1. Quick installation**.