Advanced Docker Course

Reproducibility

To reproduce an analysis we have to fix the version of all programs used.

All commands to install a package prefer the latest version.

New version can have a different interface or give different results

Plan for reproducibility

When possible, fix the version

Plan for upgrades
Make easy to change the versions

Best practice: allow to change only the initial portion of the `Dockerfile`

- Use `ENV` to store versions
- Store an identification of program, data, environment used (`git rev-parse HEAD`, `sha1sum genome.fasta`, `samtools --version`)

Pipelines

Two different strategies:

1. Run the pipeline *inside* the container.
   - Contradicts the mantra *1 container = 1 small purpose*
   - Each container runs one program. Need to run the pipeline as a single program
2. Run the pipeline *outside* the container.
   - Harder to reproduce

Exercise 1

Build an image to run cutadapt on this sample fastq file with the command

```
cutadapt -a TATCCTTG -o output.fastq input.fastq
```

On the output file, you must run the `fastq-uniq` program, taken from the git repo.

Store the information needed to reproduce the analysis.

The final file **must** be stored **outside** the container.

We will discuss the differences between the development and the production version.

Permissions

Docker is run as *root* (but it depends on how it is installed).

Problem in HPC (or if you are not root)

Solution: run the program *inside* the container as a user

```
docker run -it -u `id -u`:`id -g` ubuntu:18.04 /bin/bash
```

can be used to run the container with a given user/group, only **if they already exist** inside the container

Permission 2

Strategies:

1. add a new user/group inside the image (in the Dockerfile)
    - What if the container is run by a different user?
2. add a new user/group inside the container
    - Which user to add? Find it at runtime
        - the user that has run `docker run`
        - the owner of the data directory

Solution

- gosu
- entrypoint.sh

Entrypoint.sh

```
#!/bin/bash
# Add local user
# with the same owner as /data
USER_ID=$(stat -c %u /data)
GROUP_ID=$(stat -c %g /data)

echo "Starting with UID:GID $USER_ID:$GROUP_ID"
groupadd -g "$GROUP_ID" group
useradd --shell /bin/bash -u "$USER_ID" -g group -o\
   -c "" -m user
export HOME=/
chown --recursive "$USER_ID":"$GROUP_ID" /data

exec gosu user "$@"
```

Exercise 2

Build an image to run cutadapt on this sample fastq file with the command

```
cutadapt -a TATCCTTG -o output.fastq input.fastq
```

On the output file, you must run the `fastq-uniq` program, taken from the git repo.

Store the information needed to reproduce the analysis.

The final file must be stored outside the container, and **be owned by you***