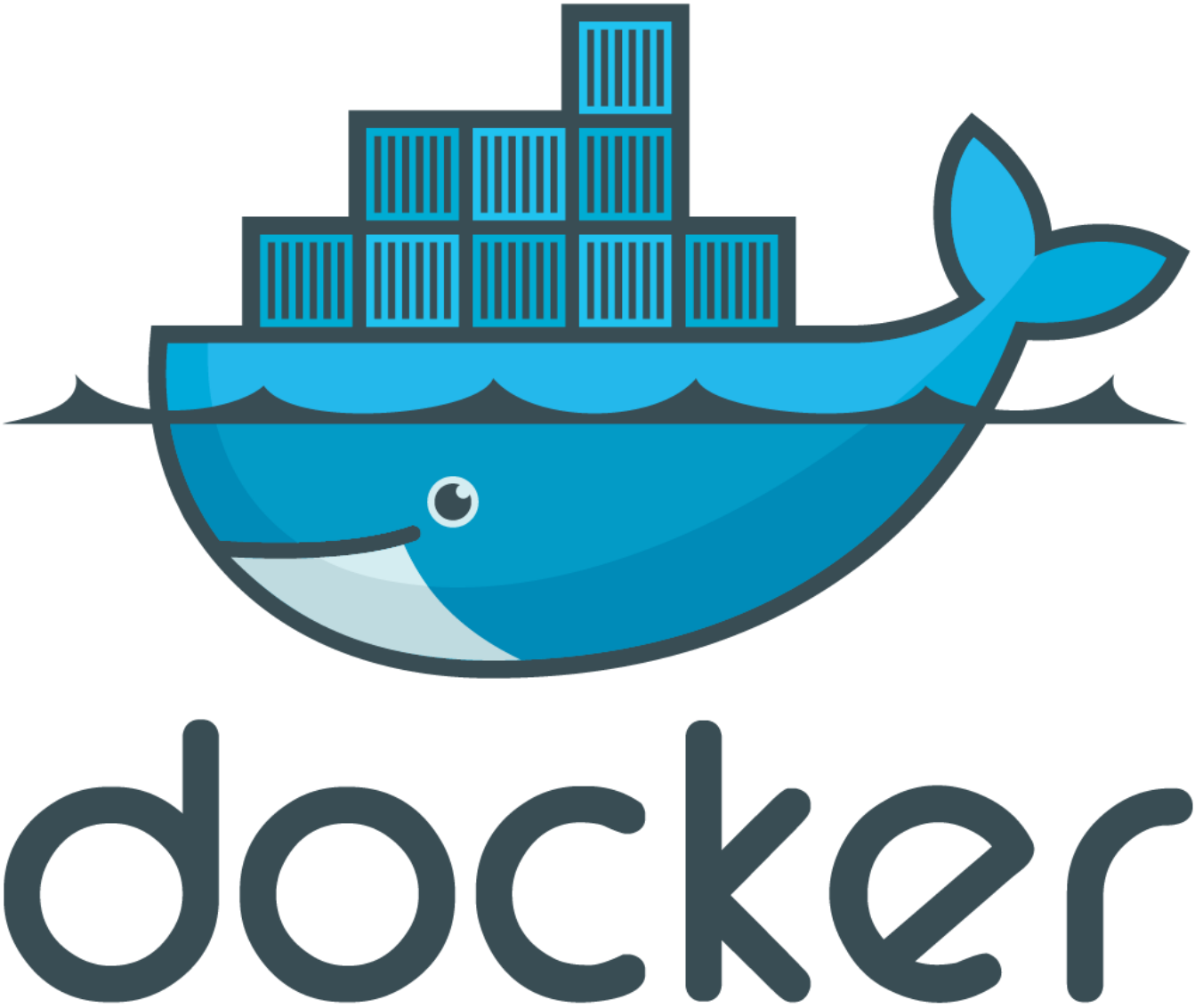

docker

Part 1

Introduction to Docker

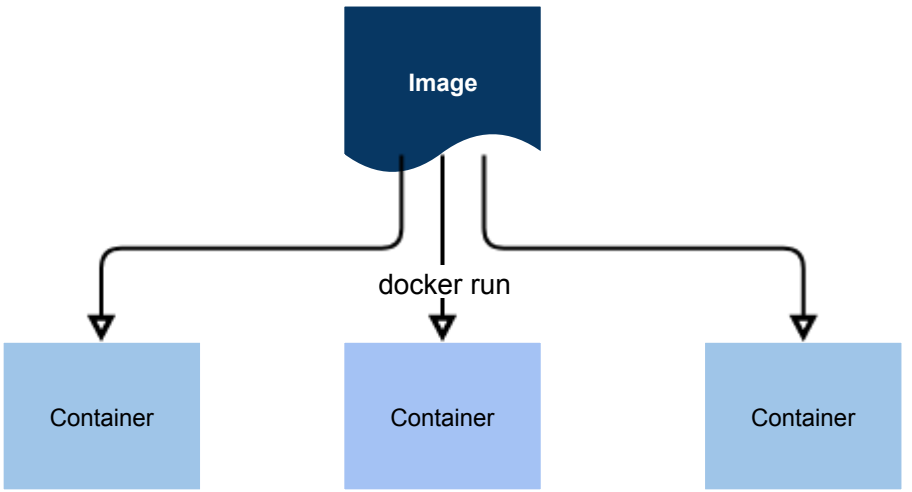
What is Docker?

> *Docker containers wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries – anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment - https://www.docker.com/what-docker*

- This is the ideal way of deploying applications such as:
  - Web applications with a proxy server, database and application code
  - Analysis pipelines that require many runtimes (Python, perl, Java etc.) and many tools (Samtools, GATK, VEP)

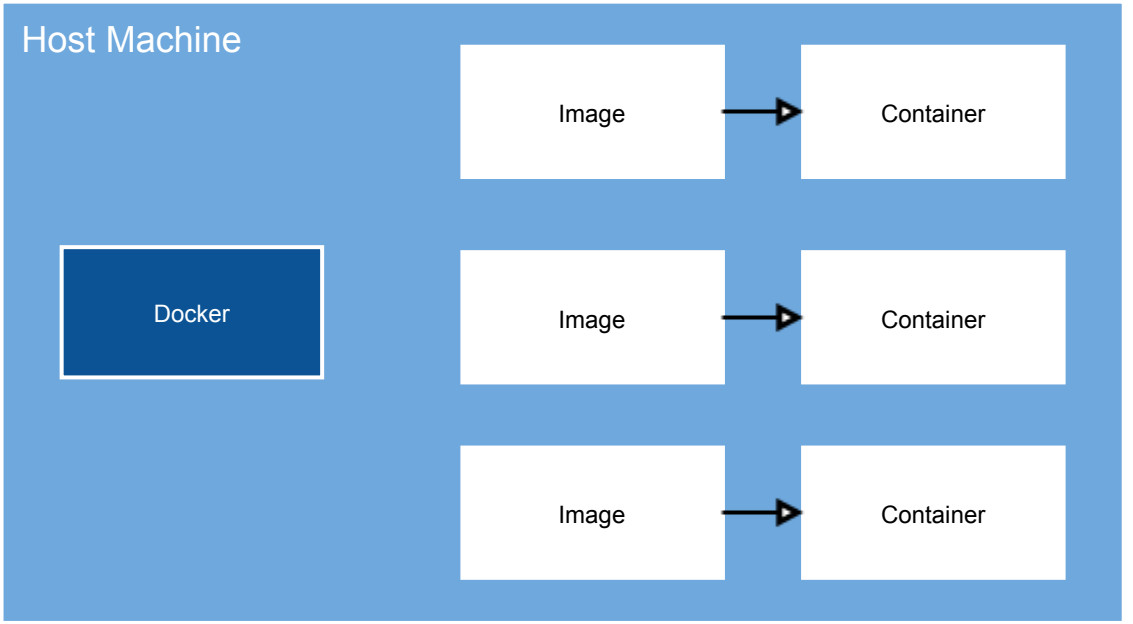
Terminology - Images and Containers

- A docker **image** an archive containing all the data needed to run the application
- When you run an **image**, it creates a **container**, which you can start and stop and delete without it affecting the image
- You can have many containers running the same image
- You can think of a Docker image as like a class in Object-Oriented Programming, and a docker container as like an object



Terminology - Host

- The **host** machine is the machine running Docker, on which images and containers are stored



Advantages

- **No Dependencies** – Docker images contain all their own dependencies, which means you don't have to do any installation yourself (compared to an application that is just source code or a .deb installer)
- **Cross platform** – Docker containers contain their own operating system, so they will run on any platform (even Windows!)
- **Easily distributable** - Can be distributed as a single .tar image file, or put on docker hub so they can docker pull your-image
- **Safe** – Files in a container can't access files on the host machine, so you can trust dockerized applications
- **Easy To Use** – Docker containers can always be run using one single docker run command

Docker vs VMs

- Virtual machines solve the same problem as docker, but are much more lightweight
- Virtual machines package the entire guest OS, while Docker uses the host kernel and a minimal OS that can be shared between containers

Part 2

Running Containers

Task 1.1 - Running your First Container

To run a container, all you need to do is specify the image name, and docker will pull the image from Docker Hub, and begin running it

```
docker run hello-world
```

Task 1.2 - Viewing the Image

Now that you've downloaded a Docker image, you can see it using:

```
docker images
```

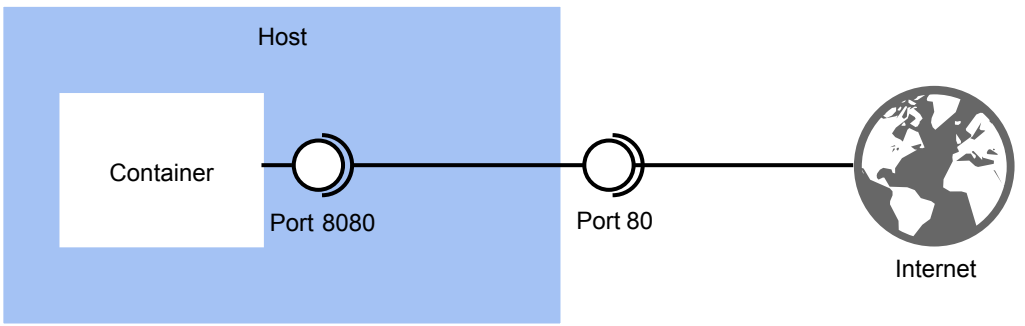
Port Mapping

- Docker containers are free to listen on whatever ports to want to, for example port 80/443 for web requests
- However, these ports are not available on the host machine unless you use

```
docker run -p host:container
```

- This command means "map port 8080 inside this container to port 80 on the host"

```
docker run -p 80:8080 nginx
```



Task 2.1 - Galaxy

- You want to run a Galaxy server on your NeCTAR instance
- The image is called `bgruening/galaxy-rna-seq`
- The galaxy image listens on port 8080 inside the container
- What is the correct `docker run` command to use?

Task 2.1 - Galaxy

- You want to run a Galaxy server on your NeCTAR instance
- The image is called `bgruening/galaxy-rna-seq`
- The galaxy image listens on port 8080 inside the container
- What is the correct `docker run` command to use?

```
docker run -p 8080:80 bgruening/galaxy-rna-seq
```

Volumes and Bind Mounts

- By default, Docker containers cannot access data on the host system. This means
  - You can't use host data in your containers
  - All data stored in the container will be lost when the container exits
- You can solve this in two ways:
  - `-v /path/in/host:/path/in/container`: This **bind mounts** a host file or directory into the container. Writes to one will affect the other.
  - `-v volume_name:/path/in/container`. This mounts a **named volume** into the container, which will live separately from the rest of your files. This is preferred, unless you need to access or edit the files from the host
- Useful resources:
  - Volumes
  - Bind Mounts
  - When to use volumes vs bind mounts

Task 2.2 - Galaxy Logs

- You need to store the logs for your Galaxy image on your host system
- The Galaxy container stores its logs in `/home/galaxy/logs`

Task 2.2 - Galaxy Logs

- You need to store the logs for your Galaxy image on your host system
- The Galaxy container stores its logs in `/home/galaxy/logs`

```
docker run \
-p 8080:80 \
-v `pwd`/galaxy_logs:/home/galaxy/logs bgruening/galaxy-rna-seq
```

or

```
docker run \
-p 8080:80 \
-v galaxy_logs:/home/galaxy/logs bgruening/galaxy-rna-seq
```

Listing running containers

- `docker ps` lists all currently running containers
- Can also show all terminated containers with the `-a` flag
- The IDs that are shown can be useful for other docker commands...
- Try it now:

```
CONTAINER ID        IMAGE                          COMMAND
4a594d121fa9        bgruening/galaxy-rna-seq       "/usr/bin/startup"
3a519be38d87        bwa_mem                        "./align.py --refe..."
64ad604535fd        bwa_mem                        "bash"
6cdc73c5699a        bwa_mem                        "./align.py --refe..."
```

Running commands inside a container

- You can run a command inside a running container using:

```
docker exec <CONTAINER_ID> <COMMAND>
```

- For example:

```
docker exec bd2ac6cce96f ls
```

- You can also run an interactive bash session inside the container with:

```
docker exec -it bd2ac6cce96f bash
```

Task 1.3 - Running a Container Interactively

- You started the Galaxy container using:

```
docker run -p -d 8080:80 bgruening/galaxy-rna-seq
```

- You want to make a quick edit to the Galaxy homepage
- The homepage is located at `/etc/galaxy/web/welcome.html`

Task 1.3 - Running a Container Interactively

- You started the Galaxy container using:

```
docker run -p -d 8080:80 bgruening/galaxy-rna-seq
```

- You want to make a quick edit to the Galaxy homepage
- The homepage is located at `/etc/galaxy/web/welcome.html`
- First, `docker ps` to find the container ID

Task 1.3 - Running a Container Interactively

- You started the Galaxy container using:

```
docker run -p -d 8080:80 bgruening/galaxy-rna-seq
```

- You want to make a quick edit to the Galaxy homepage
- The homepage is located at `/etc/galaxy/web/welcome.html`
- First, `docker ps` to find the container ID
- Next, `docker exec -it <CONTAINER_ID> bash`

Task 1.3 - Running a Container Interactively

- You started the Galaxy container using:

```
docker run -p -d 8080:80 bgruening/galaxy-rna-seq
```

- You want to make a quick edit to the Galaxy homepage
- The homepage is located at `/etc/galaxy/web/welcome.html`
- First, `docker ps` to find the container ID
- Next, `docker exec -it <CONTAINER_ID> bash`
- Now, run `nano /etc/galaxy/web/welcome.html` (or vim!) and save the file

Part 3

Making your Own Image

Dockerfiles

- The core of making a Docker image is a Dockerfile
- A Dockerfile is a list of commands, a lot like a shell script, that progressively builds the image:
  - `FROM` lists the image to "inherit" from
  - `RUN` executes a shell command
  - `COPY` copies some data from the host to the image
  - `ENTRYPOINT` sets the command that will be run when a container is created

Dockerfiles - Example

```
# Start with an empty Ubuntu image
FROM ubuntu

# Install apt dependencies
RUN apt-get update && apt-get install -y curl make build-essential libssl-d

# Copy in the repository
COPY . /opt/cpipe

# Move into the cpipe dir
WORKDIR /opt/cpipe

# Run the install script
RUN ./install.sh --noninteractive

# Run the main script
ENTRYPOINT ["./cpipe"]
```

Dockerfile Tips

- You should try to separate the Dockerfile into as many stages as possible, because this will allow for better caching
- `apt-get`:
  - You must run `apt-get update` and `apt-get install` in the same command, otherwise you will encounter caching issues
  - Remember to use `apt-get install -y`, because you will have no control over the process while it's building
- Useful resources:
  - Dockerfile reference
  - Best practices

Docker Build

- To go from a Dockerfile to a Docker image, use the `docker build` command
- You normally have to specify an image tag/name using `-t`, and a directory containing the Dockerfile. For example:

```
docker build -t my_samtools .
```

Task 2.1 - Dockerizing Samtools

- Samtools is a common utility for working with SAM and BAM alignment files
- Samtools can be installed using `apt-get install samtools`
- Make sure you tag this as `my_samtools` - we'll use this later!

Task 2.1 - Dockerizing Samtools

- Samtools is a common utility for working with SAM and BAM alignment files
- Samtools can be installed using `apt-get install samtools`
- Make sure you tag this as `my_samtools` - we'll use this later!

```
FROM ubuntu
RUN apt-get update && apt-get install -y samtools
ENTRYPOINT ["samtools"]
```

Now you've got a working samtools image, try testing it using the SAM file provided:

```
docker run my_samtools sort < alignment.bam
```

Task 2.2 - Dockerizing BWA

- bwa can be installed in much the same way as samtools
- Try `apt-get install bwa`
- Make sure you tag this as `my_bwa`

Task 2.2 - Dockerizing BWA

- bwa can be installed in much the same way as samtools
- Try `apt-get install bwa`
- Make sure you tag this as `my_bwa`

```
FROM ubuntu
RUN apt-get update && apt-get install -y bwa
ENTRYPOINT ["bwa"]
```

Task 2.3 - Dockerizing Freebayes

- Freebayes is a little harder to install - you'll need to build it from source
- Have a look at the git repo for some help: https://github.com/ekg/freebayes
- As a tip, the apt-get repositories you need for this will be: `git build-essential zlib1g-dev libbz2-dev liblzma-dev`
- Make sure you tag this as `my_freebayes`

Task 2.3 - Dockerizing Freebayes

- Freebayes is a little harder to install - you'll need to build it from source
- Have a look at the git repo for some help: https://github.com/ekg/freebayes
- As a tip, the apt-get repositories you need for this will be: `git build-essential zlib1g-dev libbz2-dev liblzma-dev`
- Make sure you tag this as `my_freebayes`

```
FROM ubuntu
RUN apt-get update && apt-get install -y git build-essential zlib1g-dev lib
WORKDIR /tmp
RUN git clone --recursive git://github.com/ekg/freebayes.git
WORKDIR freebayes
RUN make
RUN make install
ENTRYPOINT ["freebayes"]
```

Dockerized Pipelines

- Docker containers are often used to provide the tools and runtime environment for each stage in a bioinformatics pipeline
- Conveniently, the images you just make are just the right ones for running a variant calling pipeline
- Try running

```
cwltool workflow.cwl \
--read_1 data/NA12878_CARDIACM_MUTATED_L001_R1.fastq.gz \
--read_2 data/NA12878_CARDIACM_MUTATED_L001_R2.fastq.gz \
--reference data/ucsc.hg19.fasta
```

Part 4

Docker on HPC

Task 3.1 - Running on Singularity