Terra Training Session May15th, 2023

Welcome to a Terra Training Session hosted by Theiagen Genomics & the Massachusetts Department of Public Health

We appreciate your punctuality! Please give others a few minutes to arrive (and adjust their audio equipment). We will get started at **2:33 PM Eastern Time**. Thanks!

Docker for Public Health Bioinformatics

Week 1 - Introduction to Docker and Containerization

Monday May 15th, 2023

Curtis Kapsak, MS & Frank Ambrosio, MS | Theiagen Genomics

Course Introduction

Training Workshop Instructors

Frank Ambrosio, M.S.

Curtis Kapsak, M.S.

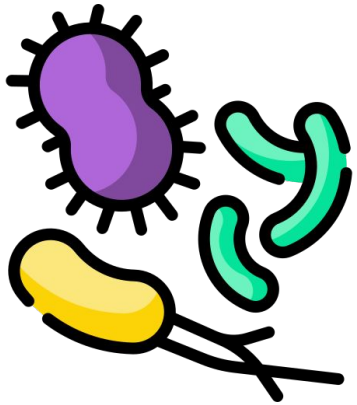
Training Workshop Overview

Training Information, Communication, and Support

- **Training Notion Page** created to host training resources and information
- **Support Contacts**:
  - support@terrapublichealth.zendesk.com

Main Course Objective

Learn about the concepts of Docker & containerization and their applications in public health bioinformatics



docker

Training Workshop Overview

This workshop is an Intermediate/Advanced course

Great resources for more information regarding containers and pathogen genomics

For more **technical content**, get connected with various **pathogen genomics communities** such as **PHA4GE, StaPH-B, & micro-binfie**

- **StaPH-B Docker User Guide**
- **Ten Recommendations for supporting open pathogen genomic analysis in public health**
  - Highlights containers and workflow management systems in context of public health
- **A Primer on Infectious Disease Bacterial Genomics**
  - Introduction to analyzing pathogen genomics data

Course Structure

4-Week Virtual Training Workshop

- **All training sessions** will begin at 2:30pm Eastern Time
  - Live Lectures (90m) on Mondays
  - Office Hours (60m) on Wednesdays
  - Exceptions:
    - No sessions the week of APHL Annual conf. (May 22-25)
    - Week 2 lecture will occur Tue May 30th 2:30-4pm EST due to Memorial Day
- **Live lectures** will include **hands-on exercises**
  - To participate, please ensure that you have registered for a GitHub account

Course Content

Week One - Intro to Docker and Containerization

- **Lecture Content**: Introduction to Docker containers
- **Hands-on Exercises**: Utilize a docker container to download a *Klebsiella pneumoniae* genome and to run Kleborate

Week Two - Container Repositories and Writing Dockerfiles

- **Lecture Content**: Intro to various repositories for Docker containers e.g. StaPH-B docker-builds & biocontainers
- **Hands-On Exercise**: Build docker images using pre-existing dockerfile

Course Content

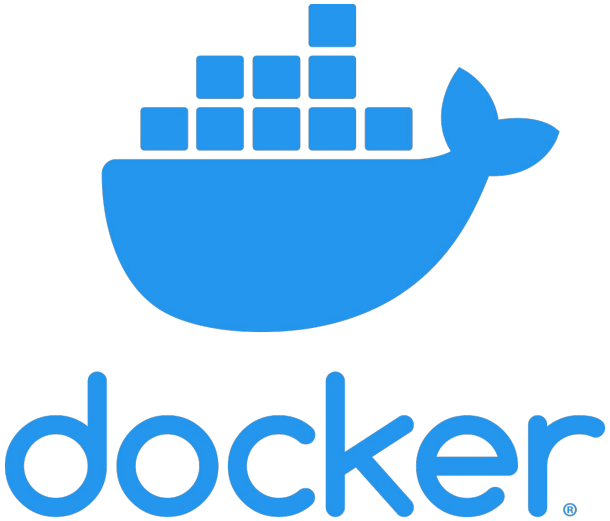
Week Three - Developing custom Docker Images

- **Lecture Content**: Intro to development and testing practices for writing dockerfiles
- **Hands-on Exercise**: TBD

Week Four - StaPH-B docker-builds project

- **Lecture Content**: Review of the StaPH-B docker-builds project and code repository
- **Hands-On Exercise**: Develop a dockerfile and create a pull request

Week 1
Intro to Docker and Containerization

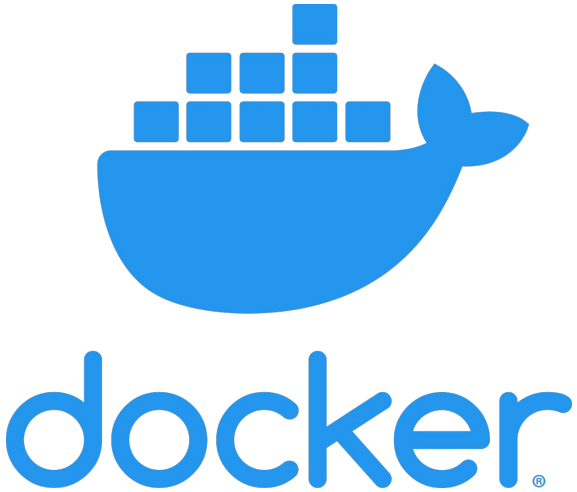


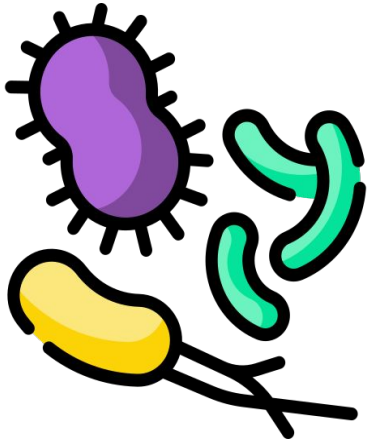
Intro to Docker and containerization

Lecture Content: Intro to Docker and containerization

Hands-on Exercises:

- **Exercise 1**: Use NCBI **`datasets`** to download a genome FASTA file - *Klebsiella pneumoniae*
- **Exercise 2**: Run **`kleborate`** on FASTA file for subtyping, serotyping, virulence and AMR prediction

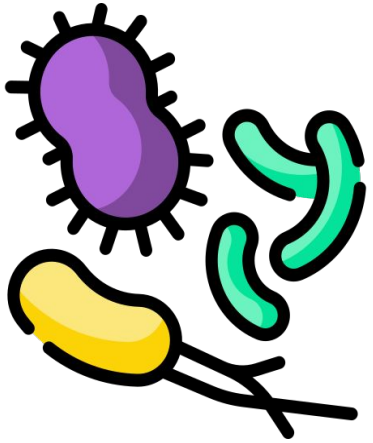




Intro to Docker and containerization

Goals by End of Week One

- Understand the concept of software containerization
- Understand the differences between Virtual Machines (VMs), virtual environments, and containers
- Understand the advantages and challenges of containers
- Learn the applications of containers in public health
- Learn how to utilize a container on the command line



Why use containers?

It's worth it!

"It changed my life" - Curtis

Why use Containers?

Ease of software install

- Avoid this →

How to get a bioinformatics headache

1. See tweet about new published tool
2. Read abstract - sounds awesome!
3. Fail to find link to source code - eventually Google it
4. Attempt to compile and install it
5. Google for 30 min for fixes
6. Finally get it built
7. Run it on tiny data set
8. Get a vague error
9. Delete and never revisit it again

Slide from: T. Seemann, ASMNGS18
https://www.slideshare.net/torstenseemann/how-to-write-bioinformatics-software-no-one-will-use/11

Why use Containers?

- One command to download and “install” a program onto your computer:

```
$ docker pull staphb/spades
Using default tag: latest
latest: Pulling from staphb/spades
b549f31133a9: Already exists
bf4358dc43e4: Pull complete
74ff0d5990f9: Pull complete
4f4fb700ef54: Pull complete
Digest:
sha256:b33f57d65cb63d631c6e3ba9b2a1c5a11ff4351475f38a1108ec61a5bf430077
Status: Downloaded newer image for staphb/spades:latest
docker.io/staphb/spades:latest
$ docker run staphb/spades spades.py --help
SPAdes genome assembler v3.15.5

Usage: spades.py [options] -o <output_dir>
```

It’s that easy!

Why use Containers?

Reproducibility

- A program may behave differently depending on how it and it's dependencies are installed
  - example: SPAdes – requires python
    - Which python version is installed? 2.7, 3.4, 3.5 ?
- **Docker images are static**, minimizing the chance that the program will run differently when installed on different computers
  - (latest version) StaPH-B SPAdes docker image has python 3.8.10
- **Goal = have the software run the same exact way, every time**

Why use Containers?

Reproducibility

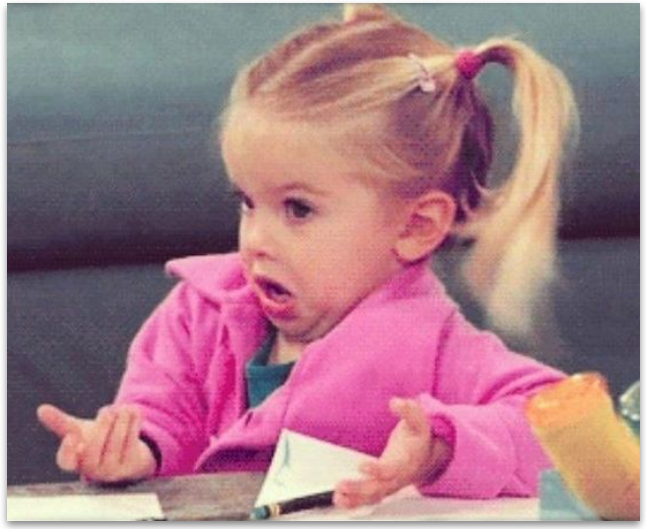
- Containers allow for pinning installations of specific:
  - database versions
  - software versions
- Can help with validating assays
- **Make your CLIA personnel happy!**
- example: E. coli/STEC serotyping via SerotypeFinder run via a docker container

Why use Containers?

Portability

- Run software on almost ANY computer/cluster/server/HPC
- Containers are easy to share
- Solves the issue of the all to familiar phrase:

"Well it works on my computer…"



Why use Containers?

Build complex & modular workflows

- Spend less time installing software, spend more time doing science
- Nearly all workflow managers utilize & prefer containers:
  - Nextflow
  - WDL
  - Snakemake
  - CWL
- Example WDL task on right, utilizes docker image for genome assembly

```
version 1.0

task shovill_pe {
  input {
    File read1_cleaned
    File read2_cleaned
    String samplename
    String docker = "quay.io/staphb/shovill:1.1.0"
    Int min_contig_length = 200
  }
  command <<<
    shovill --version | head -1 | tee VERSION
    shovill \
    --outdir out \
    --R1 ~{read1_cleaned} \
    --R2 ~{read2_cleaned} \
    --minlen ~{min_contig_length}
    mv out/contigs.fa out/~{samplename}_contigs.fasta
    mv out/contigs.gfa out/~{samplename}_contigs.gfa
  >>>
  output {
    File assembly_fasta = "out/~{samplename}_contigs.fasta"
    File contigs_gfa = "out/~{samplename}_contigs.gfa"
    String shovill_version = read_string("VERSION")
  }
  runtime {
      docker: "~{docker}"
      memory: "16 GB"
      cpu: 4
      disks: "local-disk 100 SSD"
      preemptible: 0
  }
}
```

Why use Containers?

Build complex & modular workflows

- Example workflows you may be familiar with:
  - WDL
    - TheiaCov, TheiaProk, TheiaEuk, any WDL workflows available on Terra.bio
  - Nextflow
    - Bactopia, Cecret, Donut Falls, MycoSNP-nf, PHoeNIx, StaPH-B toolkit, nf-core/viralrecon
- All of these use containers

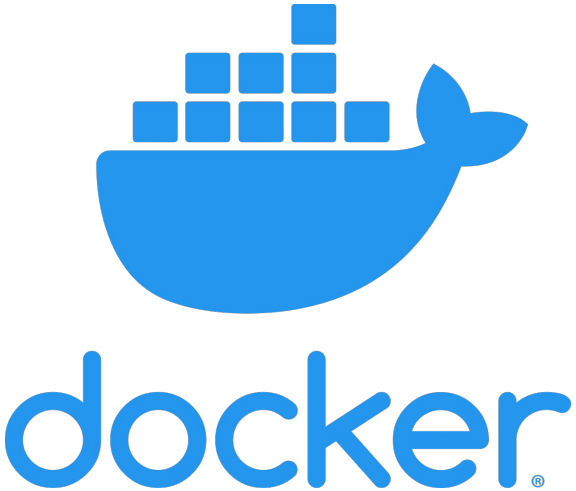
```
version 1.0

task shovill_pe {
  input {
    File read1_cleaned
    File read2_cleaned
    String samplename
    String docker = "quay.io/staphb/shovill:1.1.0"
    Int min_contig_length = 200
  }
  command <<<
    shovill --version | head -1 | tee VERSION
    shovill \
    --outdir out \
    --R1 ~{read1_cleaned} \
    --R2 ~{read2_cleaned} \
    --minlen ~{min_contig_length}
    mv out/contigs.fa out/~{samplename}_contigs.fasta
    mv out/contigs.gfa out/~{samplename}_contigs.gfa
  >>>
  output {
    File assembly_fasta = "out/~{samplename}_contigs.fasta"
    File contigs_gfa = "out/~{samplename}_contigs.gfa"
    String shovill_version = read_string("VERSION")
  }
  runtime {
      docker: "~{docker}"
      memory: "16 GB"
      cpu: 4
      disks: "local-disk 100 SSD"
      preemptible: 0
  }
}
```

Why use Containers?

Summary

- Ease of installing bioinformatics software
- Reproducibility
- Portability
- Build complex and modular bioinformatics workflows

What is a container?



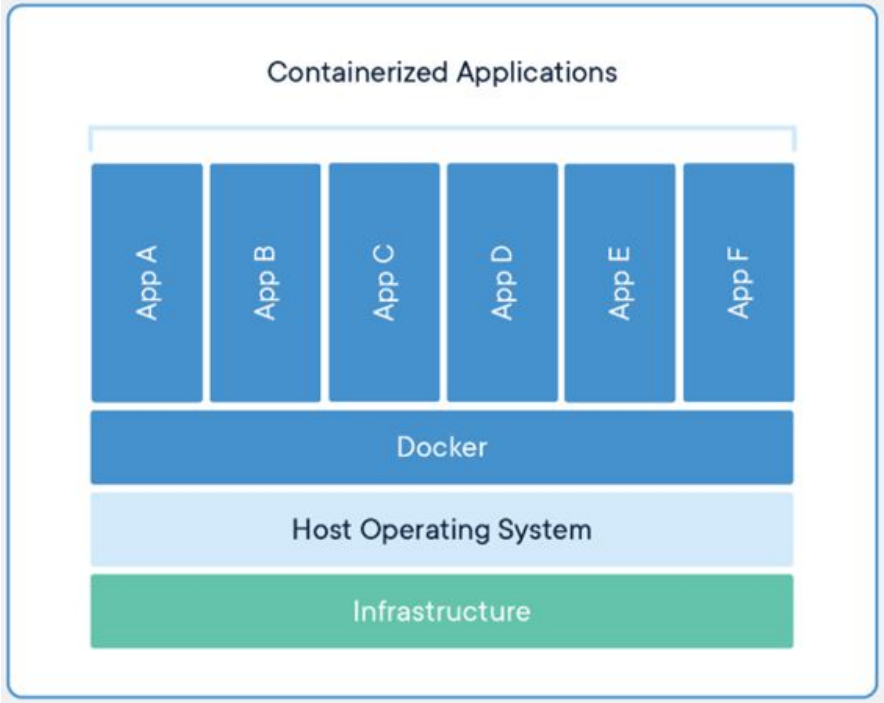
Containers

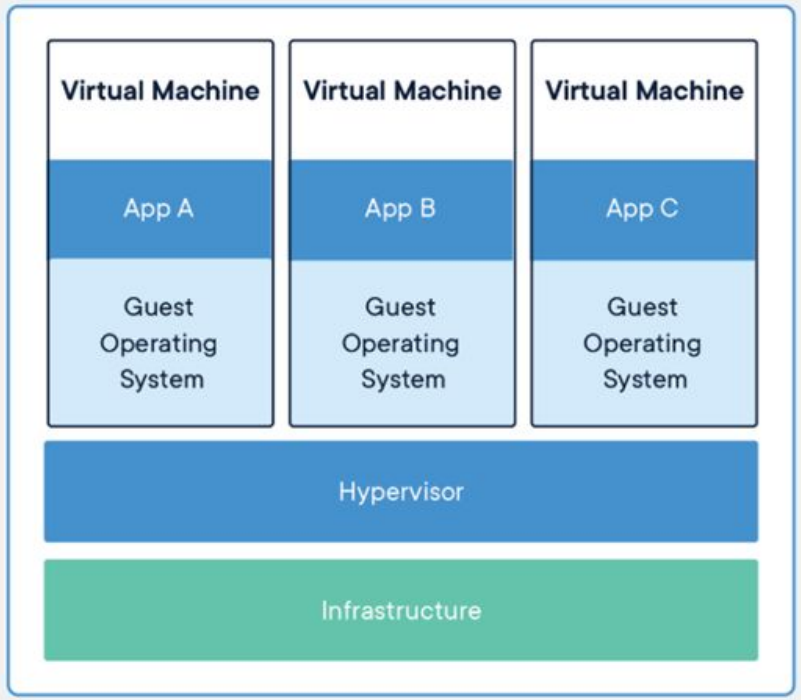
- **Software container**
  - A standard unit of software that **packages up code and all dependencies** so the application runs quickly and **reliably from one computing environment to another**
    - https://www.docker.com/resources/what-container/
- Leading platforms for containers running containers
  - Docker
  - Singularity
  - podman

Containers vs virtual machines (VM)

- **Aren't containers just fancy VMs? yes and no**

Containers →





VMs ←

extra overhead

- Run on existing operating system (OS) & hardware
- Can include an OS and dependencies, but in a smaller format (<1GB usually)
- Boot very fast

- VMs use more compute resources to run than containers
- Have OS, dependencies, but in a much larger format (10+ GB)
- Slow to boot

Containers vs virtual environments

	Containers	**Virtual environments** (like `conda` or `venv`)
architecture	includes complete filesystem, and dependencies required to run software	creates directories only containing required dependencies and python executable

Containers vs virtual environments

	Containers	**Virtual environments** (like `conda` or `venv`)
architecture	includes complete filesystem, and dependencies required to run software	creates directories only containing required dependencies and python executable
portability	highly portable to any compute environment that can run docker	usually specific to host operating system, but offers some portability with conda recipe files

Containers vs virtual environments

	Containers	**Virtual environments** (like `conda` or `venv`)
architecture	includes complete filesystem, and dependencies required to run software	creates directories only containing required dependencies and python executable
portability	highly portable to any compute environment that can run docker	usually specific to host operating system, but offers some portability with conda recipe files
isolation	complete isolation from host system	limited isolation from host system, host system dependencies can affect environment

Containers vs virtual environments

	Containers	**Virtual environments** (like `conda` or `venv`)
architecture	includes complete filesystem, and dependencies required to run software	creates directories only containing required dependencies and python executable
portability	highly portable to any compute environment that can run docker	usually specific to host operating system, but offers some portability with conda recipe files
isolation	complete isolation from host system	limited isolation from host system, host system dependencies can affect environment
resource requirements	requires significant resources to operate (cpu, disk space, and memory). Still less than a traditional VM	requires fewer resources and relies on host

Containers vs virtual environments

	Containers	**Virtual environments** (like `conda` or `venv`)
architecture	includes complete filesystem, and dependencies required to run software	creates directories only containing required dependencies and python executable
portability	highly portable to any compute environment that can run docker	usually specific to host operating system, but offers some portability with conda recipe files
isolation	complete isolation from host system	limited isolation from host system, host system dependencies can affect environment
resource requirements	requires significant resources to operate (cpu, disk space, and memory). Still less than a traditional VM	requires fewer resources and relies on host
deployment	often used in complex, production environments	often used for development, testing, and analysis

Break

5 min break!

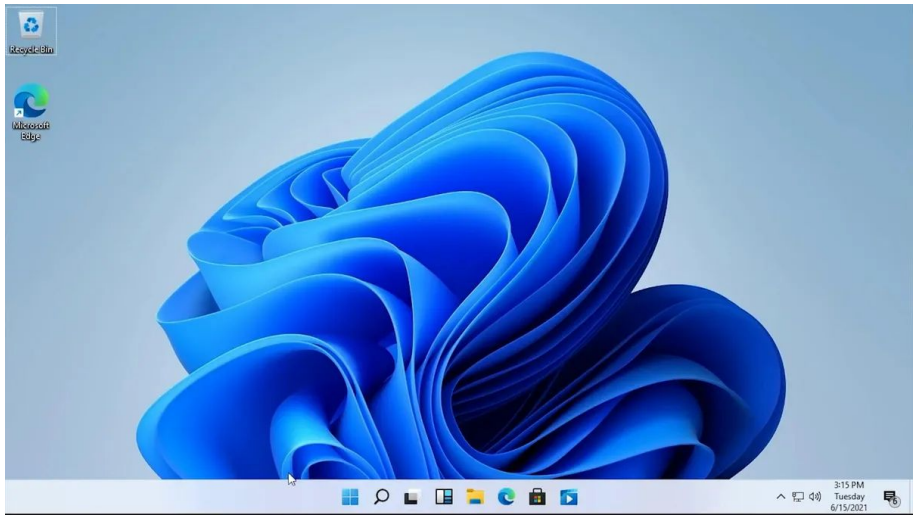
Docker architecture

How does all of this work?

What is a docker **image** and docker **container**?

- **A docker image is a read-only template with instructions for creating a Docker container**
- Similar to the file type used to install an operating system - ISO

- A **docker container is the runnable instance of an image**
- **Containers are ephemeral** (i.e. are temporary and usually deleted after use)



docker **image** is used to create the docker **container**

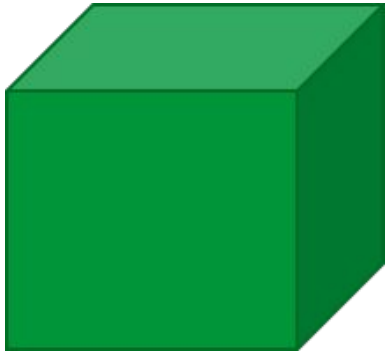
Central dogma of containers

Dockerfile

```
FROM ubuntu:xenial

# metadata
LABEL base.image="ubuntu:xenial"
LABEL version="1"
LABEL software="SPAdes"
LABEL software.version="3.13.0"
LABEL description="de novo DBG genome assembler"
LABEL website="http://cab.spbu.ru/files/release3.13.0/manual.html"

# Maintainer
MAINTAINER Curtis Kapsak <curtis.kapsak@state.co.us>

RUN apt-get update && apt-get install -y python \
  wget

RUN wget http://cab.spbu.ru/files/release3.13.0/SPAdes-3.13.0-Linux.tar.gz && \
  tar -xzf SPAdes-3.13.0-Linux.tar.gz && \
  rm -r SPAdes-3.13.0-Linux.tar.gz && \
  mkdir /data

ENV PATH="${PATH}:/SPAdes-3.13.0-Linux/bin"
WORKDIR /data
```

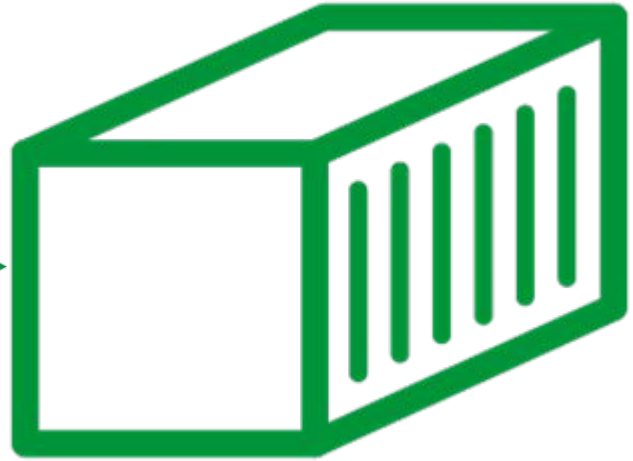
docker build →

Docker Image



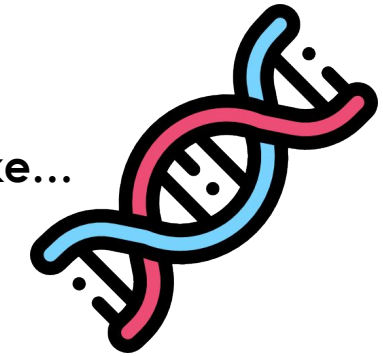
docker run →

Docker container



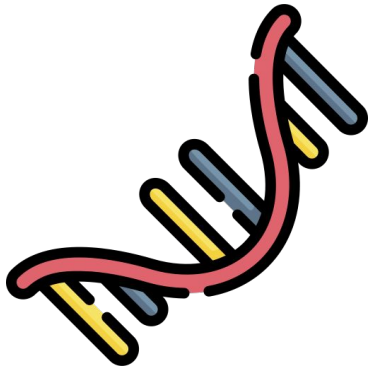
kind of like…

DNA



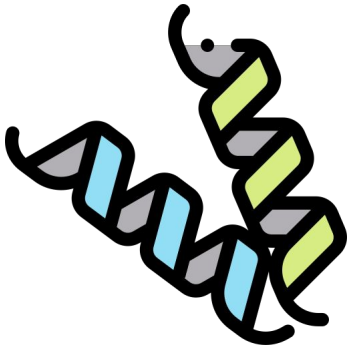
→

RNA



→

Protein



images: Flaticon.com, freepik.com

The Dockerfile

- In order to build a docker image, you need at a minimum one file: the **`Dockerfile`**
- **`Dockerfile`** = set of instructions used to build a docker image
- Similar to an installation script or a **.yml** file used for making/sharing conda environments
- **If you use the StaPH-B docker images, they are pre-built and you will not need the dockerfile**
  - But, if you want to create your own docker image and build it on your machine locally, the dockerfile is required

The Dockerfile

- Dockerfile instructions (`FROM`, `RUN`, `COPY`, `ENV`, etc.) will add a "layer" to the docker image
- Images are multi-layered and different images may share layers like the base image
  - `FROM ubuntu:focal`

Spades Dockerfile

- https://github.com/StaPH-B/docker-builds/blob/master/spades/3.15.5/Dockerfile

Dockerfile Cheat Sheet

- https://kapeli.com/cheat_sheets/Dockerfile.docset/Contents/Resources/Documents/index

```
FROM ubuntu:focal as app

# to make it easier to upgrade for new versions; ARG variables only persist during docker image build time
ARG SPADES_VER="3.15.5"

LABEL base.image="ubuntu:focal"
LABEL dockerfile.version="2"
LABEL software="SPAdes"
LABEL software.version="${SPADES_VER}"
LABEL description="de novo DBG genome assembler"
LABEL website="https://github.com/ablab/spades"
LABEL license="https://github.com/ablab/spades/blob/v3.15.3/assembler/LICENSE"
LABEL maintainer="Curtis Kapsak"
LABEL maintainer.email="kapsakcj@gmail.com"

# install dependencies; cleanup apt garbage
# python v3.8.10 is installed here; point 'python' to python3
RUN apt-get update && apt-get install --no-install-recommends -y python3 \
 python3-distutils \
 wget \
 pigz \
 ca-certificates && \
 apt-get autoclean && rm -rf /var/lib/apt/lists/* && \
 update-alternatives --install /usr/bin/python python /usr/bin/python3 10

# install SPAdes binary; make /data
RUN wget http://cab.spbu.ru/files/release${SPADES_VER}/SPAdes-${SPADES_VER}-Linux.tar.gz && \
  tar -xzf SPAdes-${SPADES_VER}-Linux.tar.gz && \
  rm -r SPAdes-${SPADES_VER}-Linux.tar.gz && \
  mkdir /data

# set PATH and locale settings for singularity
ENV LC_ALL=C.UTF-8 \
    PATH="${PATH}:/SPAdes-${SPADES_VER}-Linux/bin"

WORKDIR /data

# test layer
FROM app as test

# print version and run the supplied test flag
RUN spades.py --version && spades.py --test && spades.py --help
```

Containers

Docker Images can be built locally **or** pre-built images can be downloaded from public repositories like:

- Docker hub https://hub.docker.com/
- Quay.io https://quay.io/
- GitHub container registry (GHCR) https://ghcr.io
- Cloud provider container registries
  - GCP Artifact Registry
  - Amazon Elastic Container Registry
  - Microsoft Azure Container Registry
- Private registries are an (paid) option

How to share/obtain containers: container registries

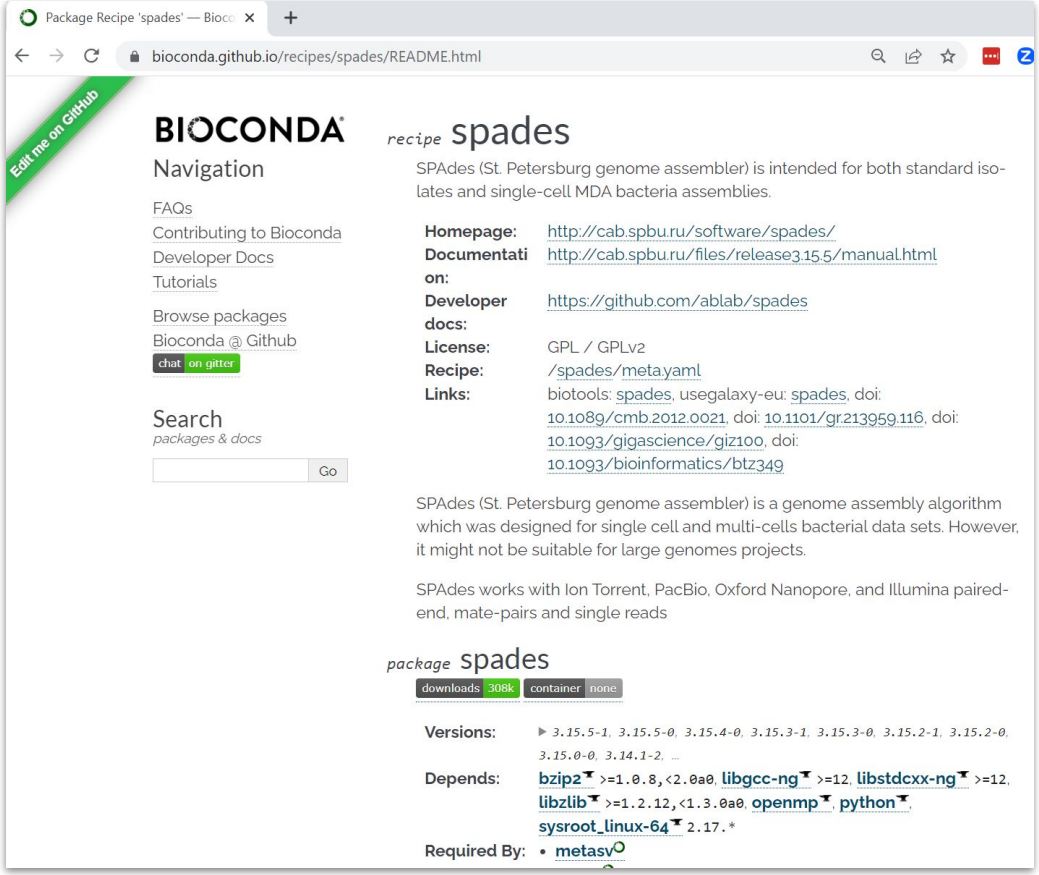
- Docker Hub https://hub.docker.com/
  - https://hub.docker.com/u/staphb
  - **millions of public docker images**
- Quay.io
  - https://quay.io/organization/staphb
    - (StaPH-B mirrors docker images between docker hub and quay)

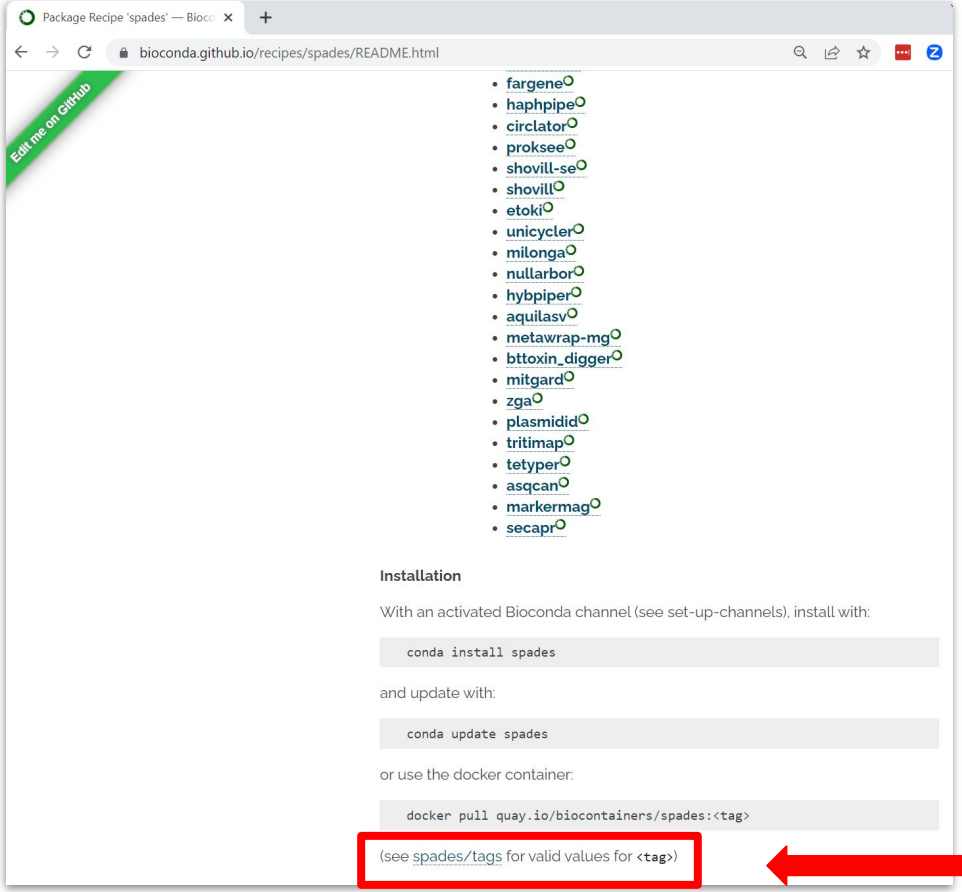
docker hub

Red Hat Quay.io

How to share/obtain containers: container registries

- All bioconda packages are available as docker images on quay.io
  - https://bioconda.github.io/recipes/spades/README.html





link to quay.io

How to share/obtain containers: container registries

- Sometimes bioinfo tool developers publish their own docker images
  - Bakta - https://github.com/oschwengers/bakta
  - NCBI AMRFinderPlus - https://github.com/ncbi/amr/wiki/Installing-AMRFinder

Installation

Bakta can be installed via BioConda, Docker, Singularity and Pip. However, we encourage to use Conda or Docker/Singularity to automatically install all required 3rd party dependencies.

In all cases a mandatory database must be downloaded.

BioConda

```
conda install -c conda-forge -c bioconda bakta
```

Docker

```
sudo docker pull oschwengers/bakta
sudo docker run oschwengers/bakta --help
```

Installation instructions and get-started guides: Docker docs

For further convenience, we provide a shell script (`bakta-docker.sh`) handling Docker related parameters (volume mounting, user IDs, etc):

```
bakta-docker.sh --db <db-path> --output <output-path> <input>
```

Installing AMRFinder

Arjun Prasad edited this page on Jan 12 · 21 revisions

Installation

AMRFinderPlus requires HMMER, BLAST+, Linux, and perl. We provide a bioconda package, Linux binaries, and the source code is available to compile AMRFinderPlus yourself though we haven't extensively tested compiling AMRFinderPlus on other systems and aren't supporting non-Linux systems at this time.

How to install

We distribute AMRFinderPlus, through **Bioconda**, as a **linux x86 binary**, as a **docker image**, and as **source code** through GitHub.

Bioconda

Here are links to instructions for three methods of installation. The simplest, and recommended method is to install AMRFinderPlus and all of the prerequisites with bioconda. See Install with bioconda.

Docker

A DockerHub image NCBI/amr is provided, and should be downloadable using `docker pull ncbi/amr`. The build instructions for the docker image are available on GitHub. See the docker README for details and some examples of how to use it.

The whole system

Docker architecture

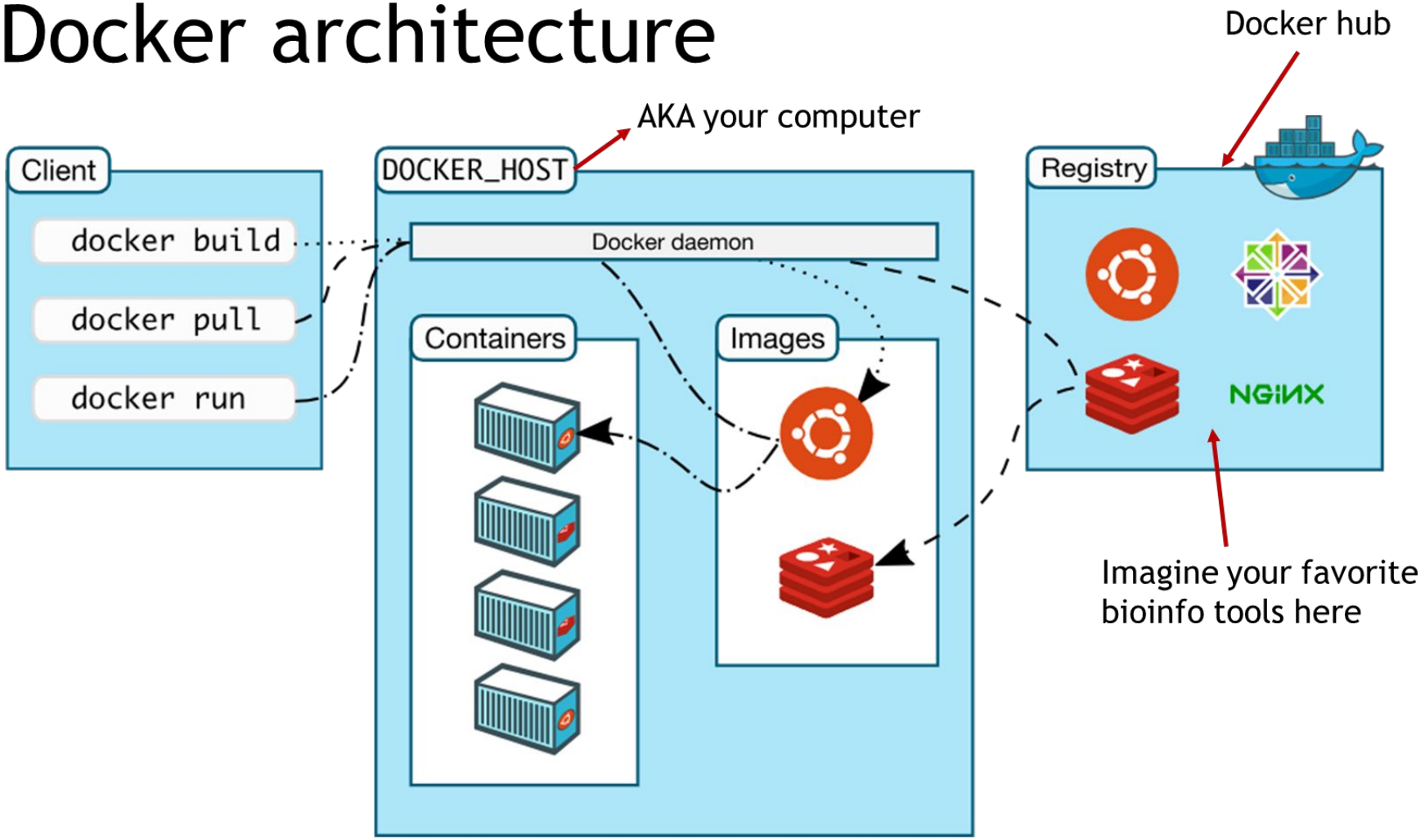


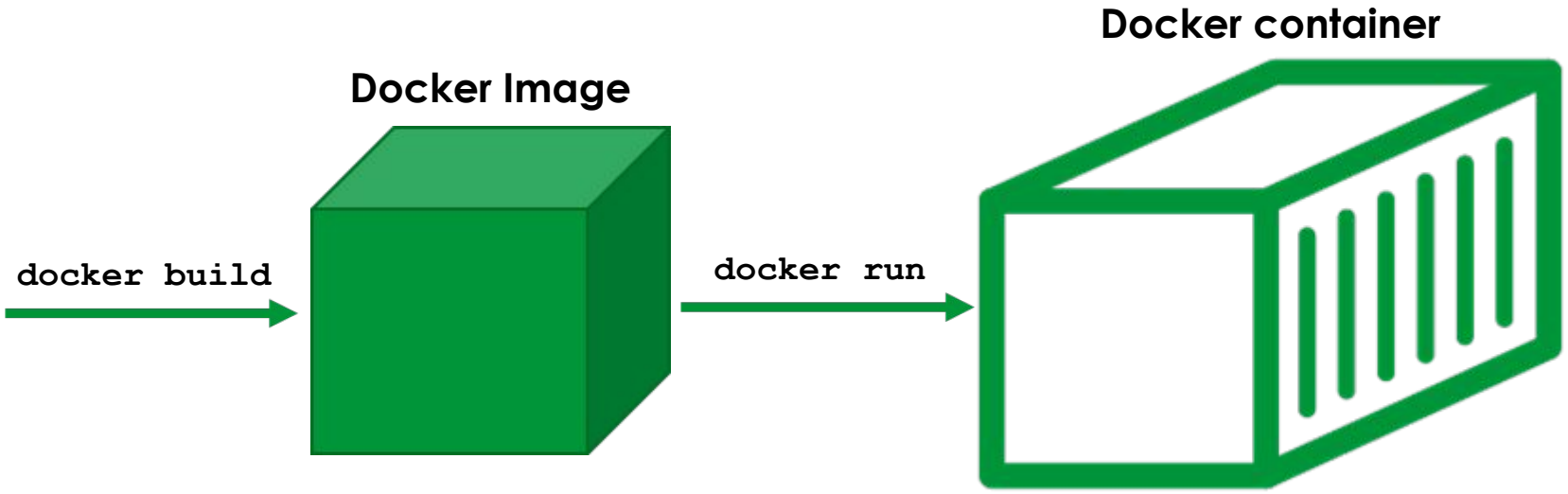
Image from: https://docs.docker.com/engine/docker-overview/

Take homes

- **Dockerfile** is used to create the docker **image**
- Docker **image** is used to create the docker **container**
- Container **is the runnable instance of an image**

Dockerfile

```
FROM ubuntu:xenial

# metadata
LABEL base.image="ubuntu:xenial"
LABEL version="1"
LABEL software="SPAdes"
LABEL software.version="3.13.0"
LABEL description="de novo DBG genome assembler"
LABEL website="http://cab.spbu.ru/files/release3.13.0/manual.html"

# Maintainer
MAINTAINER Curtis Kapsak <curtis.kapsak@state.co.us>

RUN apt-get update && apt-get install -y python \
  wget

RUN wget http://cab.spbu.ru/files/release3.13.0/SPAdes-3.13.0-Linux.tar.gz && \
  tar -xzf SPAdes-3.13.0-Linux.tar.gz && \
  rm -r SPAdes-3.13.0-Linux.tar.gz && \
  mkdir /data

ENV PATH="${PATH}:/SPAdes-3.13.0-Linux/bin"
WORKDIR /data
```



Take homes

- Where can I get docker **images**?
  - **Container registries - docker hub**, **quay.io**, **github container registry**, **cloud provider registries (GCP**, **AWS**, **Azure)**
- Container = A standard unit of software that **packages up code and all dependencies** so the application runs quickly and **reliably from one computing environment to another**

Break

5 min break!

Docker on the command line

- Common docker commands:
- `docker pull` – **downloads an image from a repository**
- `docker run` – **run a command in a container**
- `docker build` – **build an image from a dockerfile**
- `docker images` – **list all available images**
- `docker system prune` – **delete unused images & containers**
- `docker rmi` – **remove an image**
- `docker inspect` – **get information about an image**

Docker CLI Cheat Sheet:
https://docs.docker.com/get-started/docker_cheatsheet.pdf

Docker on the command line

Demo & hands on time!

- Navigate to here:
  - https://github.com/theiagen/docker-builds/tree/master/training/NE-BRR-docker-for-PH-bioinformatics-May2023
- Sign into Gitpod:
  - https://gitpod.io/workspaces

Further reading & resources

- StaPH-B Github repo and docker hub account
  - https://github.com/StaPH-B/docker-builds
  - https://hub.docker.com/u/staphb
- Docker Documentation - a wealth of info here. Note that we use Docker Community Edition, as you have to pay for the Enterprise Edition
  - https://docs.docker.com/
- An awesome tutorial/workshop on docker for bioinformatics
  - https://github.com/PawseySC/bio-workshop-18
- Template for your Dockerfile
  - https://github.com/StaPH-B/docker-builds/blob/master/dockerfile-template/Dockerfile
- Some best practices
  - https://staphb.org/docker-builds/make_containers/
- Search for docker images and (sometimes) Dockerfiles here:
  - http://hub.docker.com/
  - https://quay.io/
- "What is Docker?"(~11 min)
  - https://www.youtube.com/watch?time_continue=1&v=aLipr7tTuA4

Acknowledgements

- MA DPH
- Members of StaPH-B & the docker-builds contributors & maintainers
  - Erin Young, UT PHL
  - Kelsey Florek, WI PHL
  - Kevin Libuit, Theiagen Genomics
  - Frank Ambrosio, Theiagen Genomics
  - many more awesome people!
    - StaPH-B docker-builds contributors: https://github.com/StaPH-B/docker-builds#authorsmaintainers
- APHL
- CDC

StaPH-B

State Public Health Bioinformatics