NE BRR Training Session June 5th, 2023

Welcome to a NE BRR 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 3 - Developing Custom Docker Images

Monday June 5th, 2023

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

Course Introduction

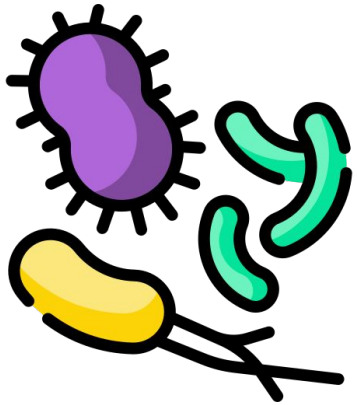
Training Workshop Overview

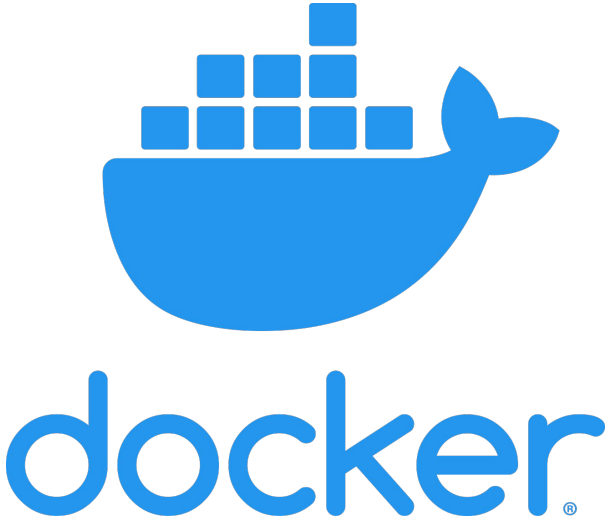
Training Information, Communication, and Support

- **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





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 dockerfiles

Course Content

Week Three - Developing custom Docker Images

- **Lecture Content**: Intro to development and testing practices for writing dockerfiles
- **Hands-on Exercise**: Create a new dockerfile NCBI datasets; assign homework: contribute dockerfile to StaPH-B docker-builds

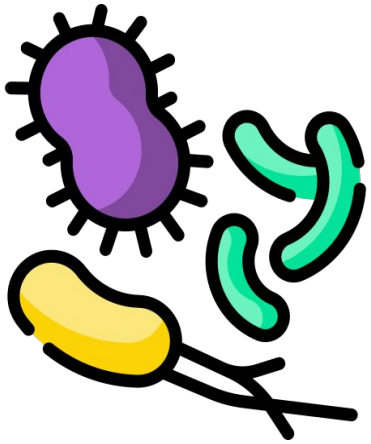
Week Four - StaPH-B docker-builds project

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

Container Repositories and Writing Dockerfiles

Goals by End of Week Three

- Learn best-practices for developing and testing dockerfiles
- Learn strategies for creating new dockerfiles for bioinformatics software
- Gain experience developing and testing a dockerfile



Outline

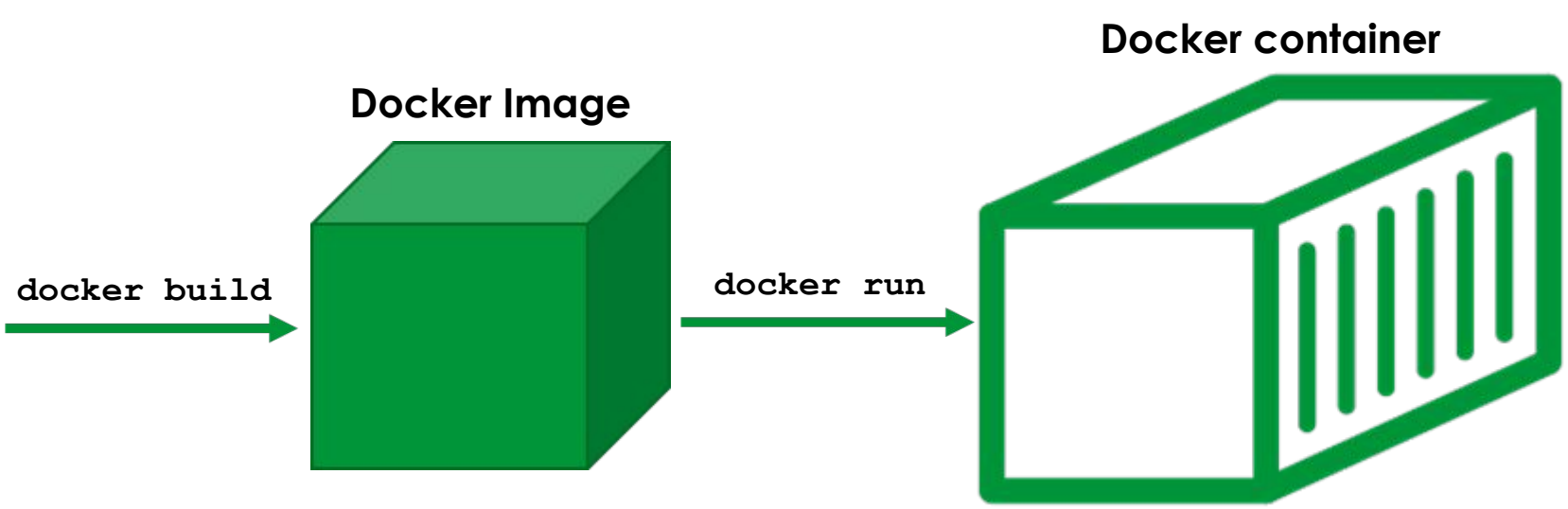
- Review Weeks 1 & 2
  - Dockerfiles & **`docker build`**
  - Best practices for writing Dockerfiles
- Strategies for creating and testing dockerfiles
- Homework - update or create your own dockerfile

Week 1 Review

- **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
```



Week 2 Review

Dockerfile instructions

- **`FROM`** defines the base docker image
- **`ARG`** set environmental variables ONLY available during build time
- **`ENV`** set environmental variables that persist during and after build time
- **`RUN`** executes a command in a new layer
- **`WORKDIR`** sets the working directory for executing commands
- **`COPY`** (and **`ADD`**) copy files into the docker image
- **`LABEL`** adds metadata to your docker image
- *There are a few other instructions, but these are the main ones

Week 2 Review

`docker build`

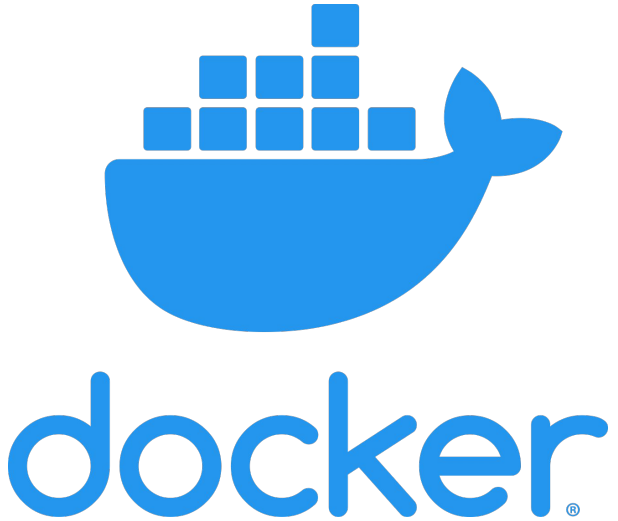
- Builds an image from a dockerfile
- At a minimum, requires a Dockerfile. Some dockerfiles require other files for building (scripts, databases, etc.)
- Official docs: https://docs.docker.com/engine/reference/commandline/build/
- General command structure:

```
docker build --tag <name>:<tag> <directory-with-dockerfile>
```

- example using SPAdes dockerfile:

```
docker build --tag spades:3.15.5 spades/3.15.5/
```

Week 3
Developing Custom Docker Images



Best practices for writing dockerfiles

- One docker container should be used for one purpose - one bioinfo tool*
  - * There are some exceptions!

Best practices for writing dockerfiles

- One docker container should be used for one purpose - one bioinfo tool*
  - * There are some exceptions!
- Only install what is necessary. Avoid installing extra programs to keep disk usage low

Best practices for writing dockerfiles

- One docker container should be used for one purpose - one bioinfo tool*
  - * There are some exceptions!
- Only install what is necessary. Avoid installing extra programs to keep disk usage low
- Fewer layers = better. **`RUN`**, **`COPY`**, and **`ADD`** instructions add layers

Best practices for writing dockerfiles

- One docker container should be used for one purpose - one bioinfo tool*
  - * There are some exceptions!
- Only install what is necessary. Avoid installing extra programs to keep disk usage low
- Fewer layers = better. `RUN`, `COPY`, and `ADD` instructions add layers
- Readability of dockerfile is helpful. Usually use one command per line

Best practices for writing dockerfiles

- One docker container should be used for one purpose - one bioinfo tool*
  - * There are some exceptions!
- Only install what is necessary. Avoid installing extra programs to keep disk usage low
- Fewer layers = better. **`RUN`**, **`COPY`**, and **`ADD`** instructions add layers
- Readability of dockerfile is helpful. Usually use one command per line
- No “large” databases or files. Large means >1GB
  - There are exceptions, but usually it’s better practice to bring large databases into the container at runtime instead of keeping in container
- Docker documentation: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

Strategies for creating dockerfiles

General tips

- **`docker build`** often while writing dockerfile. Trial and error as much as necessary!

Strategies for creating dockerfiles

General tips

- **`docker build`** often while writing dockerfile. Trial and error as much as necessary!
- If looking for the location of files, launch interactive container to see where files are located: **`docker run -it <image>`**
  - alternatively - add **`ls`**, **`find,`** or other commands in your dockerfile

Strategies for creating dockerfiles

General tips

- **`docker build`** often while writing dockerfile. Trial and error as much as necessary!
- If looking for the location of files, launch interactive container to see where files are located: **`docker run -it <image>`**
  - alternatively - add **`ls`**, **`find,`** or other commands in your dockerfile
- Helpful to have the dockerfile open in front of you when building an image. VSCode makes it easy for us

Strategies for creating dockerfiles

General tips

- **`docker build`** often while writing dockerfile. Trial and error as much as necessary!
- If looking for the location of files, launch interactive container to see where files are located: **`docker run -it <image>`**
  - alternatively - add **`ls`**, **`find,`** or other commands in your dockerfile
- Helpful to have the dockerfile open in front of you when building an image. VSCode makes it easy for us
- Use a Dockerfile linter (such as the Docker VSCode extension) to catch errors before you **`docker build`**

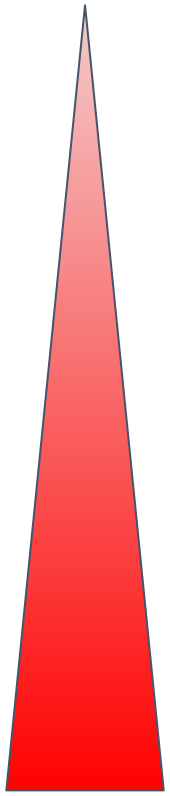
Strategies for creating dockerfiles

General tips

- **`docker build`** often while writing dockerfile. Trial and error as much as necessary!
- If looking for the location of files, launch interactive container to see where files are located: **`docker run -it <image>`**
  - alternatively - add **`ls`**, **`find,`** or other commands in your dockerfile
- Helpful to have the dockerfile open in front of you when building an image. VSCode makes it easy for us
- Use a Dockerfile linter (such as the Docker VSCode extension) to catch errors before you **`docker build`**
- Use **`docker build --progress=plain`** so that all STDOUT/STDERR is printed to screen - can see every command being executed

Strategies for creating dockerfiles

Easy



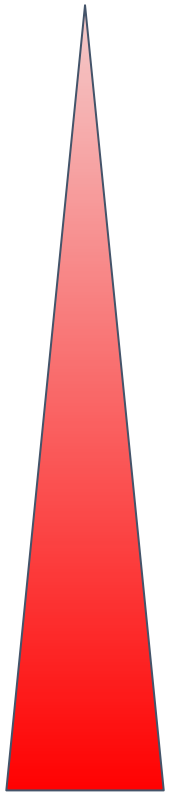
Difficult

I want to create a dockerfile, where do I start?

- Easiest - Use & modify an existing dockerfile
  - StaPH-B provides many dockerfiles
  - Tool developers (or tool users) may provide their own dockerfiles
    - If the dockerfile code is open source (and licensed as such) it's fair to use with proper attribution!

Strategies for creating dockerfiles

Easy



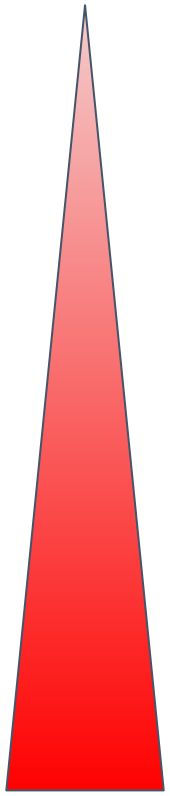
Difficult

I want to create a dockerfile, where do I start?

- Easiest - Use & modify an existing dockerfile
  - StaPH-B provides many dockerfiles
  - Tool developers (or tool users) may provide their own dockerfiles
    - If the dockerfile code is open source (and licensed as such) it's fair to use with proper attribution!
- A bit more challenging - start from a template dockerfile
  - StaPH-B provides a template dockerfile here: https://github.com/StaPH-B/docker-builds/blob/master/dockerfile-template/Dockerfile

Strategies for creating dockerfiles

Easy



Difficult

I want to create a dockerfile, where do I start?

- Easiest - Use & modify an existing dockerfile
  - StaPH-B provides many dockerfiles
  - Tool developers (or tool users) may provide their own dockerfiles
    - If the dockerfile code is open source (and licensed as such) it's fair to use with proper attribution!
- A bit more challenging - start from a template dockerfile
  - StaPH-B provides a template dockerfile here: https://github.com/StaPH-B/docker-builds/blob/master/dockerfile-template/Dockerfile
- Most challenging - writing a dockerfile from scratch

3 minute break: resume 3:03pm

Week 3 Exercise

Navigate to:

https://gitpod.io/workspaces

Strategies for creating dockerfiles

I'm starting from a template or pre-existing dockerfile, where do I start?

1. Read the tool's documentation. Familiarize yourself with the installation procedure.

   How does the tool author recommend to install the tool?

Strategies for creating dockerfiles

I'm starting from a template or pre-existing dockerfile, where do I start?

1. Read the tool's documentation. Familiarize yourself with the installation procedure. How does the tool author recommend to install the tool?
2. See what programming language the tool is written in - that will dictate how things are installed. Python, Perl, Rust, R, C/C++, something else?
    a. Does the installation require code compilation?
    b. If so, does the tool author provide pre-compiled binaries (executables)?
        i. pre-compiled binaries are usually easier to download and use than compiling code as part of Dockerfile

Strategies for creating dockerfiles

I'm starting from a template or pre-existing dockerfile, where do I start?

1. Remember that not every user will be running the container as the `**root**` linux user. Singularity and other container engines may run containers as non-root users
   a. Make sure that required files (scripts, databases, etc. files) are readable and executable to all users. You may have to use **`chmod`** command to change permissions on files

Strategies for creating dockerfiles

I'm starting from a template or pre-existing dockerfile, where do I start?

1. Remember that not every user will be running the container as the `root` linux user. Singularity and other container engines may run containers as non-root users
   a. Make sure that required files (scripts, databases, etc. files) are readable and executable to all users. You may have to use **`chmod`** command to change permissions on files
2. Place files in an expected location & document where important files are located.
   a. example: Mummer docker image used for ANI for enteric pathogens
      i. https://github.com/theiagen/docker-builds/tree/master/mummer/4.0.0-RGDv2
      ii. "*The FASTA files for RGDv2 can be found within the directory* ***`/RGDv2/`*** *inside the docker image.*"

Strategies for creating dockerfiles

Programming language specific tips

Python

- first - install python and try installing python dependencies (e.g. **`numpy`**) via **`apt-get`**
- second - install **`pip`** using **`apt-get`**, then use **`pip`** to install specific python packages
  - advantage: easy to pin versions
- example: NanoPlot

```
16  # install dependencies via apt; cleanup apt garbage; set locale to en_US.UTF-8
17  RUN apt-get update && apt-get install -y zlib1g-dev \
18   bzip2 \
19   libbz2-dev \
20   liblzma-dev \
21   libcurl4-gnutls-dev \
22   libncurses5-dev \
23   libssl-dev \
24   python3 \
25   python3-pip \
26   python3-setuptools \
27   locales && \
28   locale-gen en_US.UTF-8 && \
29   apt-get autoclean && rm -rf /var/lib/apt/lists/*
30
31  # for singularity compatibility
32  ENV LC_ALL=C
33
34  # install NanoPlot via pypi using pip3; make /data directory
35  RUN pip3 install matplotlib psutil requests NanoPlot==${NANOPLOT_VER} && \
36   mkdir /data
```

Strategies for creating dockerfiles

Programming language specific tips

Perl

- first - try installing perl dependencies (e.g. DateTime) via **`apt-get`**
- second - install **`cpanm`** using **`apt-get`**, then use **`cpanm`** to install specific perl dependencies
- example: Prokka

```
28    # install dependencies
29    RUN apt-get update && apt-get -y --no-install-recommends install \
30     bzip2 \
31     gzip \
32     wget \
33     perl \
34     less \
35     libdatetime-perl \
36     libxml-simple-perl \
37     libdigest-md5-perl \
38     default-jre \
39     bioperl \
40     hmmer \
41     zlib1g-dev \
42     python \
43     liblzma-dev \
44     libbz2-dev \
45     xz-utils \
46     curl \
47     g++ \
48     cpanminus \
```

```
73    RUN cpanm List::Util
```

Strategies for creating dockerfiles

Programming language specific tips

Compiled languages (C, C++, Rust)

- Pre-compiled binaries
  - Usually are operating system or CPU architecture specific
  - You usually want the **64-bit Linux binaries. AKA x86_64**
- example: Mash
- When binaries are not available, you may have to compile the code yourself
  - May require **`gcc`** (C code) or **`g++`** (C++ code) for compiling the code
  - Other dependencies might also be required for compilation, usually tool authors will list those. Example: **`zlib1g-dev`**, **`make`**, etc.
- example: Samtools

If time allows:
Demo: **`assembly-scan`**

Homework!

- Now that we've learned some of the tips and tricks for writing dockerfiles, let's put our knowledge to the test and write a new dockerfile.
- Let's share our dockerfiles & images with the community & contribute to the StaPH-B docker-builds project.
  - https://github.com/StaPH-B/docker-builds
- Please see the separate slide deck with instructions on how to contribute.

Homework!

- bioinfo tools & versions where dockerfiles are needed!
  - beginner/easy
    - update dragonflye v1.1.1 - current dockerfile needs version update. No GitHub issue yet - **Jessie**
    - update fastp v0.23.4 - current dockerfile needs version update. No GitHub issue yet
    - update minimap2 v2.26 - current dockerfile needs version update. No GitHub issue yet
    - update seqkit v2.4.0 - current dockerfile needs version update. No GitHub issue yet - **Luc**
    - update snp-sites v2.5.1 - current dockerfile missing app and test layers. GitHub issue - **Sean**
    - update kSNP3 v3.1 - current dockerfile missing app and test layers. GitHub issue - **Kari**
    - update colorid 0.1.4.3 - current dockerfile missing app and test layers. GitHub issue
    - update hmmer v3.3 - current dockerfile missing app and test layers. GitHub issue - **Neranjan**
    - update clustalo v1.2.4 - current dockerfile missing app and test layers. GitHub issue
  - intermediate
    - seqtk v1.4 - have dockerfile for v1.3, needs updating
    - sra-tools (AKA sra-toolkit) v3.0.5 - have dockerfile for 2.9.2, needs updating
    - krakenuniq 1.0.4 - have dockerfile in progress
  - advanced
    - Krocus v1.0.3 - start w/ dockerfile template
    - Meningotype v0.8.2-beta - start w dockerfile template
    - MIDAS v1.3.2 - start with other example dockerfiles (but tweak to StaPH-B requirements)
    - Samtools - start with existing dockerfile, update with new build stage - **Kutluhan**
  - Have ideas for tools not listed here?

Homework!

- Once you have been assigned a tool, it is your homework to follow the instructions for creating & testing a dockerfile, and submitting a Pull Request via GitHub to contribute your code to the StaPH-B docker-builds project
- Feel free to work on this task at your own pace in the GitPod environment
  - NOTE: you will need to create a new GitPod workspace for this, see slide deck for instructions
- Please use time during office hours this week and next week to ask questions, seek help & advice as you develop.
- Curtis (and potentially other StaPH-B maintainers) will review your code, make suggestions for improvements, help troubleshoot, etc. to guide you through the process

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