Terra Training Session May 30th, 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 2 - Container Repositories and Writing Dockerfiles

Tuesday May 30th, 2023

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

Course Introduction

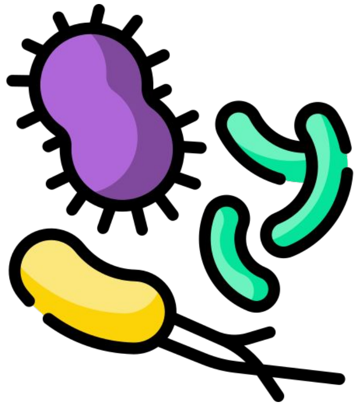
Training Workshop Overview

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



docker

Training Workshop Overview

This workshop is an Intermediate/Advanced course

Great resources for more information regarding containers and pathogen genomics

- [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

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

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**: 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

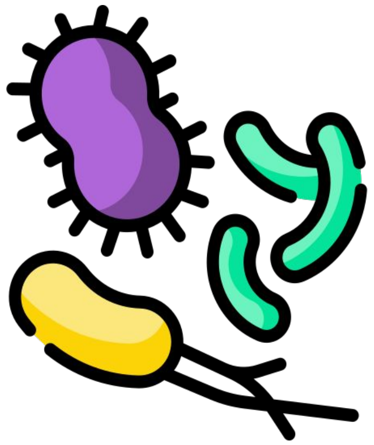
Week 2
Container Repositories and Writing Dockerfiles

Container Repositories and Writing Dockerfiles

Lecture Content: Intro to various repositories for public health related Docker images and writing Dockerfiles

Hands-on Exercises:

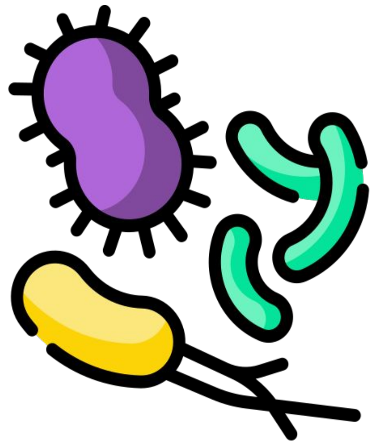
- **Exercise 1**: Use NCBI `datasets` to download a genome FASTA file - *Klebsiella pneumoniae*
- **Exercise 2**: Run `kleborate` on FASTA file for subtyping, serotyping, virulence and AMR prediction



Container Repositories and Writing Dockerfiles

Goals by End of Week Two

- Learn about publicly available container registries & resources
- Understand the Dockerfile and how it is used for building docker images
- Learn best-practices for writing dockerfiles
- Learn how to build a docker image on the command line using pre-defined dockerfiles



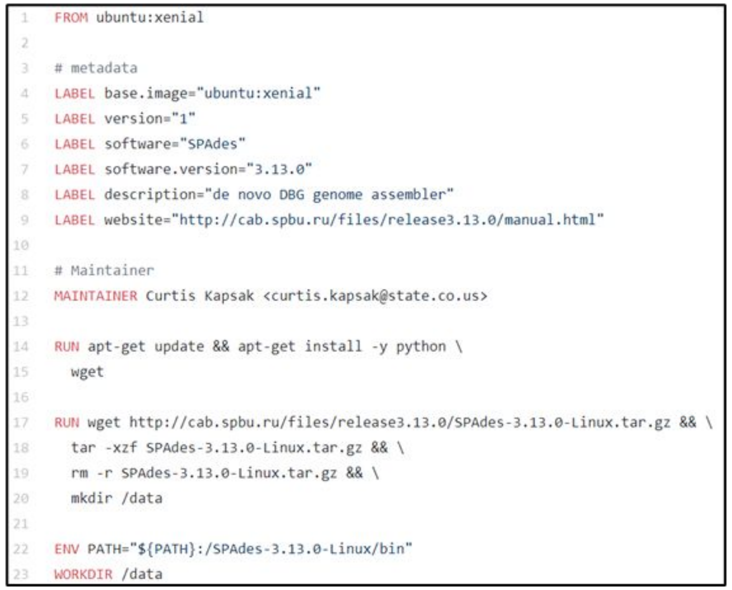
Outline

- Container registries & resources
- Getting started with Dockerfiles
  - Best practices for writing Dockerfiles
- Building docker images with **`docker build`**
- Hands-on exercises with **`docker build`**

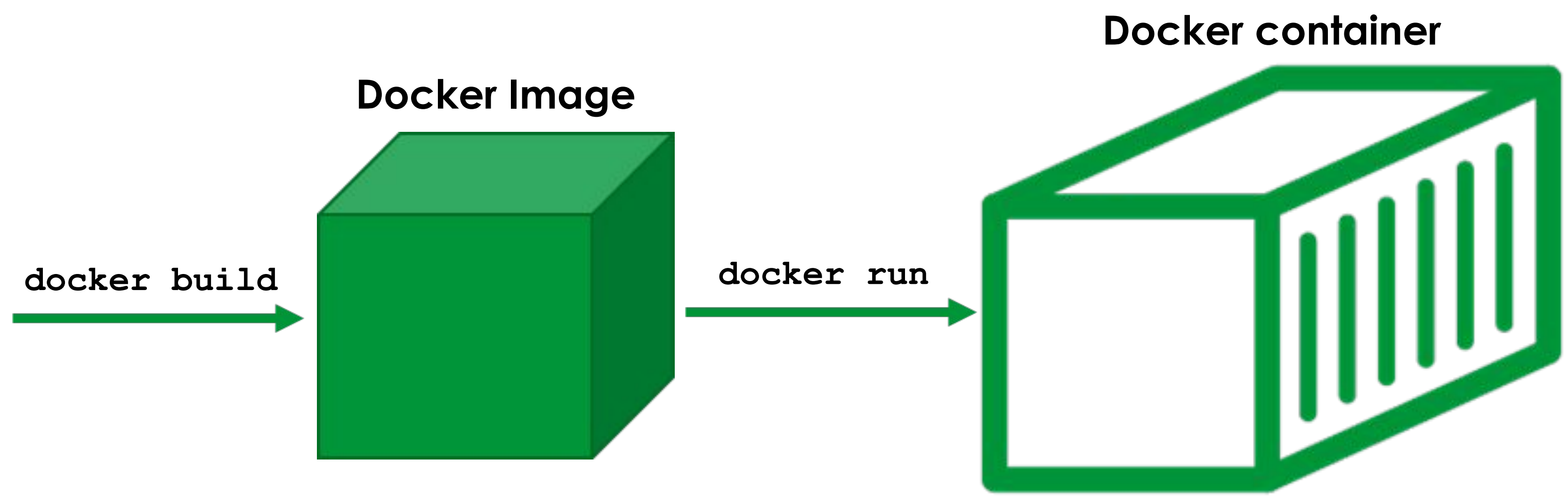
Week 1 Review

- **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
```



Review - Container registries

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

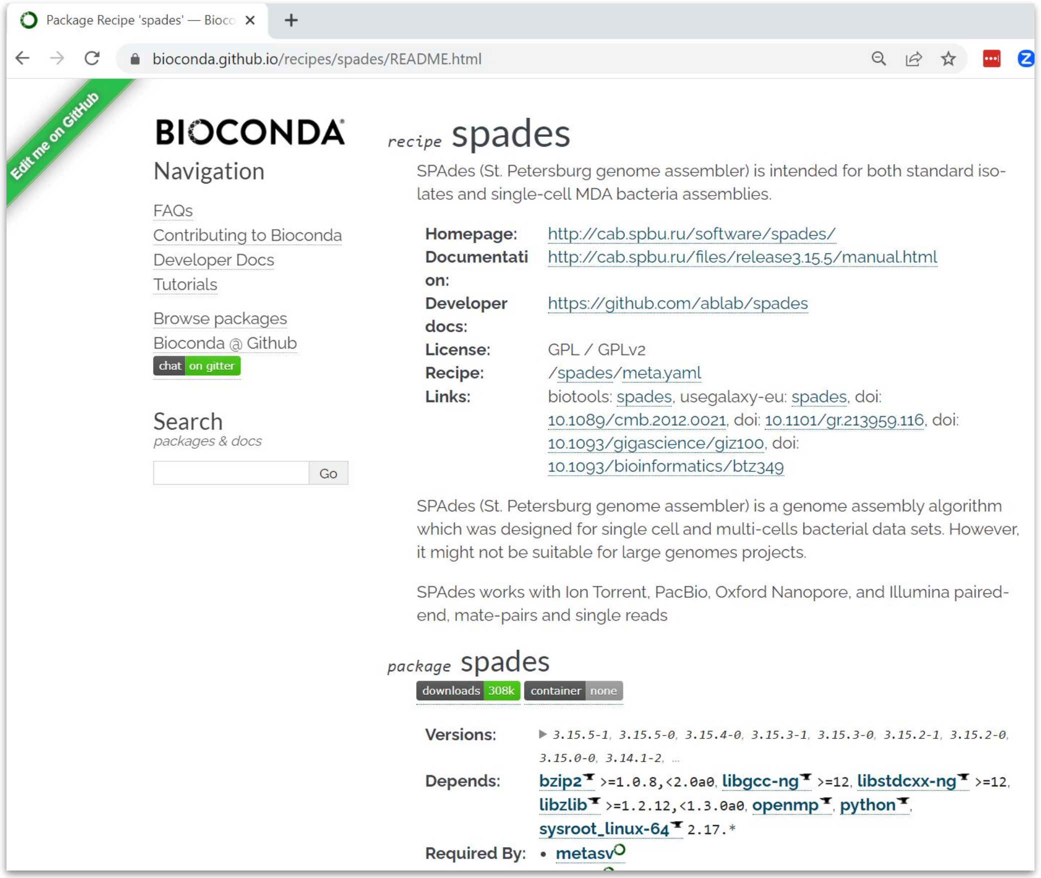
- Docker hub https://hub.docker.com/
- Quay.io https://quay.io/
- GitHub container registry (GHCR) https://ghcr.io
- Cloud provider container registries
  - GCP Artifact Registry
  - Amazon Elastic Container Registry
  - Microsoft Azure Container Registry
- Private registries are an (paid) option

Review - 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 dockerhub and quay)

Review - 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


Package Recipe 'spades' — Bioc
bioconda.github.io/recipes/spades/README.html
Edit me on GitHub
BIOCONDA
chat on gitter
Go

Navigation

FAQs
Contributing to Bioconda
Developer Docs
Tutorials

Browse packages
Bioconda @ Github

Search
packages & docs

recipe spades

SPAdes (St. Petersburg genome assembler) is intended for both standard isolates and single-cell MDA bacteria assemblies.

Homepage:	http://cab.spbu.ru/software/spades/
Documentation:	http://cab.spbu.ru/files/release3.15.5/manual.html
Developer docs:	https://github.com/ablab/spades
License:	GPL / GPLv2
Recipe:	/spades/meta.yaml
Links:	biotools: spades, usegalaxy-eu: spades, doi: 10.1089/cmb.2012.0021, doi: 10.1101/gr.213959.116, doi: 10.1093/gigascience/giz100, doi: 10.1093/bioinformatics/btz349

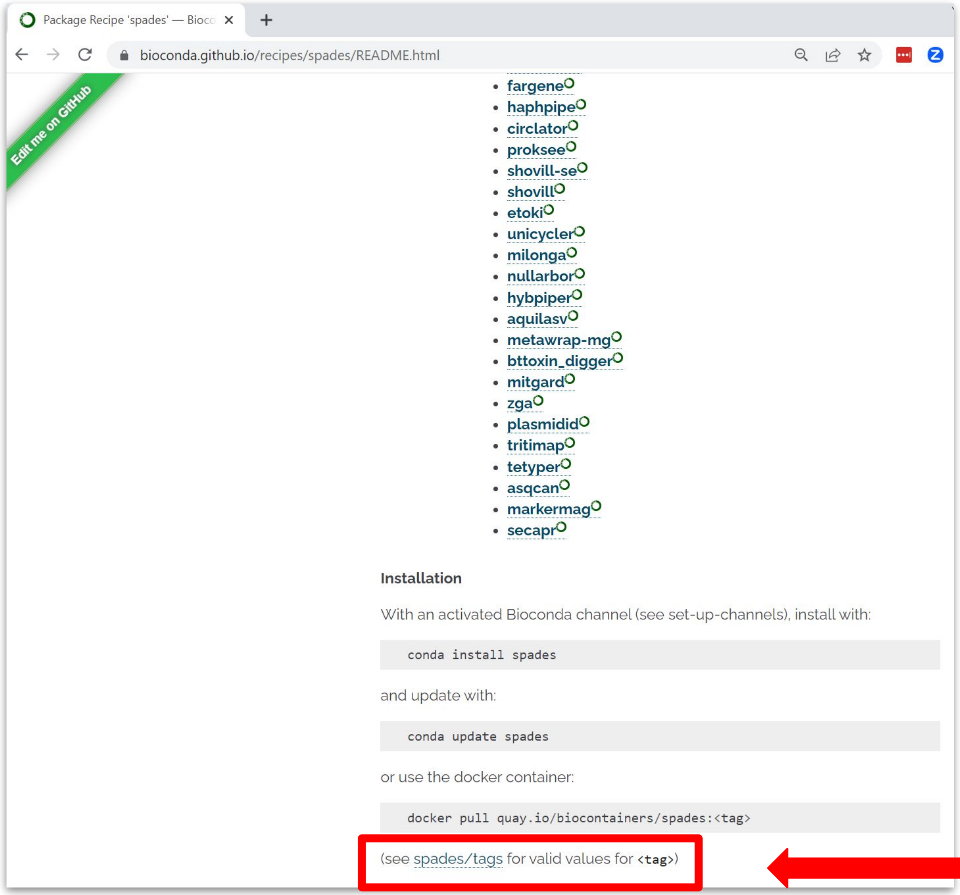
SPAdes (St. Petersburg genome assembler) is a genome assembly algorithm which was designed for single cell and multi-cells bacterial data sets. However, it might not be suitable for large genomes projects.

SPAdes works with Ion Torrent, PacBio, Oxford Nanopore, and Illumina paired-end, mate-pairs and single reads

package spades

downloads 308k container none

Versions:	▶ 3.15.5-1, 3.15.5-0, 3.15.4-0, 3.15.3-1, 3.15.3-0, 3.15.2-1, 3.15.2-0, 3.15.0-0, 3.14.1-2, ...
Depends:	bzip2 >=1.0.8,<2.0a0, libgcc-ng >=12, libstdcxx-ng >=12, libzlib >=1.2.12,<1.3.0a0, openmp, python, sysroot_linux-64 2.17.*
Required By:	• metasv


Package Recipe 'spades' — Bioc
bioconda.github.io/recipes/spades/README.html
Edit me on GitHub

- fargene
- haphpipe
- circlator
- proksee
- shovill-se
- shovill
- etoki
- unicycler
- milonga
- nullarbor
- hybpiper
- aquilasv
- metawrap-mg
- bttoxin_digger
- mitgard
- zga
- plasmidid
- tritimap
- tetyper
- asqcan
- markermag
- secapr

Installation

With an activated Bioconda channel (see set-up-channels), install with:

```
conda install spades
```

and update with:

```
conda update spades
```

or use the docker container:

```
docker pull quay.io/biocontainers/spades:<tag>
```

(see spades/tags for valid values for `<tag>`)

link to quay.io

Why write my own dockerfile?

- Not all docker images were created equally
  - Some (like biocontainers) are made by robots! 🤖
- Not all docker images work “out-of-the-box”
  - Limited-to-no testing performed with docker image

Why write my own dockerfile?

- Not all docker images were created equally
    - Some (like biocontainers) are made by robots! 🤖
- Not all docker images work “out-of-the-box”
    - Limited-to-no testing performed with docker image
- “I cannot find a docker image for the software I want to use”

Why write my own dockerfile?

- Not all docker images were created equally
  - Some (like biocontainers) are made by robots! 🤖
- Not all docker images work “out-of-the-box”
  - Limited-to-no testing performed with docker image
- “I cannot find a docker image for the software I want to use”
- “I want to know EXACTLY how X software was installed”
  - “Were versions pinned?”

Why write my own dockerfile?

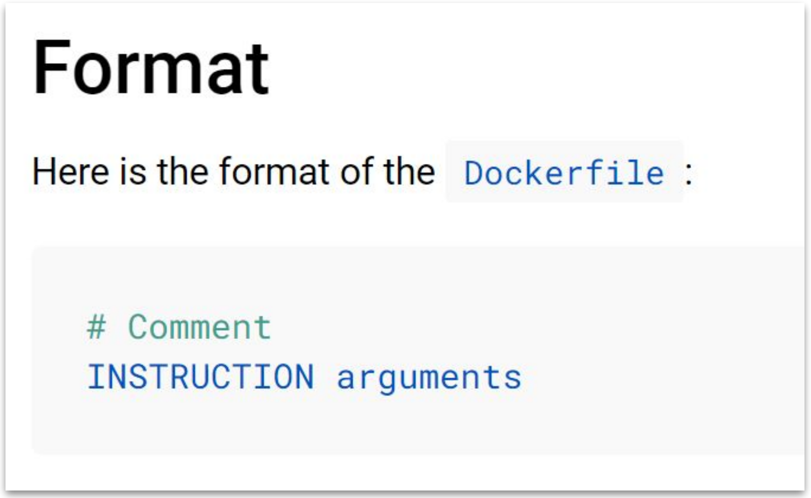
- Not all docker images were created equally
    - Some (like biocontainers) are made by robots! 🤖
- Not all docker images work “out-of-the-box”
    - Limited-to-no testing performed with docker image
- “I cannot find a docker image for the software I want to use”
- “I want to know EXACTLY how X software was installed”
    - “Were versions pinned?”
- “I want to include multiple tools in a single docker image”
    - minimap2 + samtools commands piped together:

```
minimap2 -x map-ont -a ref.fasta reads.fastq.gz | samtools sort -o out.bam
```

Getting started with Dockerfiles

Review - 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

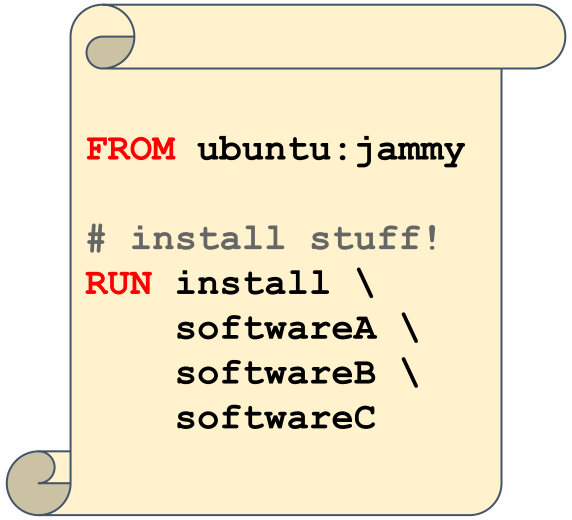


Format

Here is the format of the `Dockerfile`:

```
# Comment
INSTRUCTION arguments
```

https://docs.docker.com/engine/reference/builder/



```
FROM ubuntu:jammy

# install stuff!
RUN install \
    softwareA \
    softwareB \
    softwareC
```

The Dockerfile

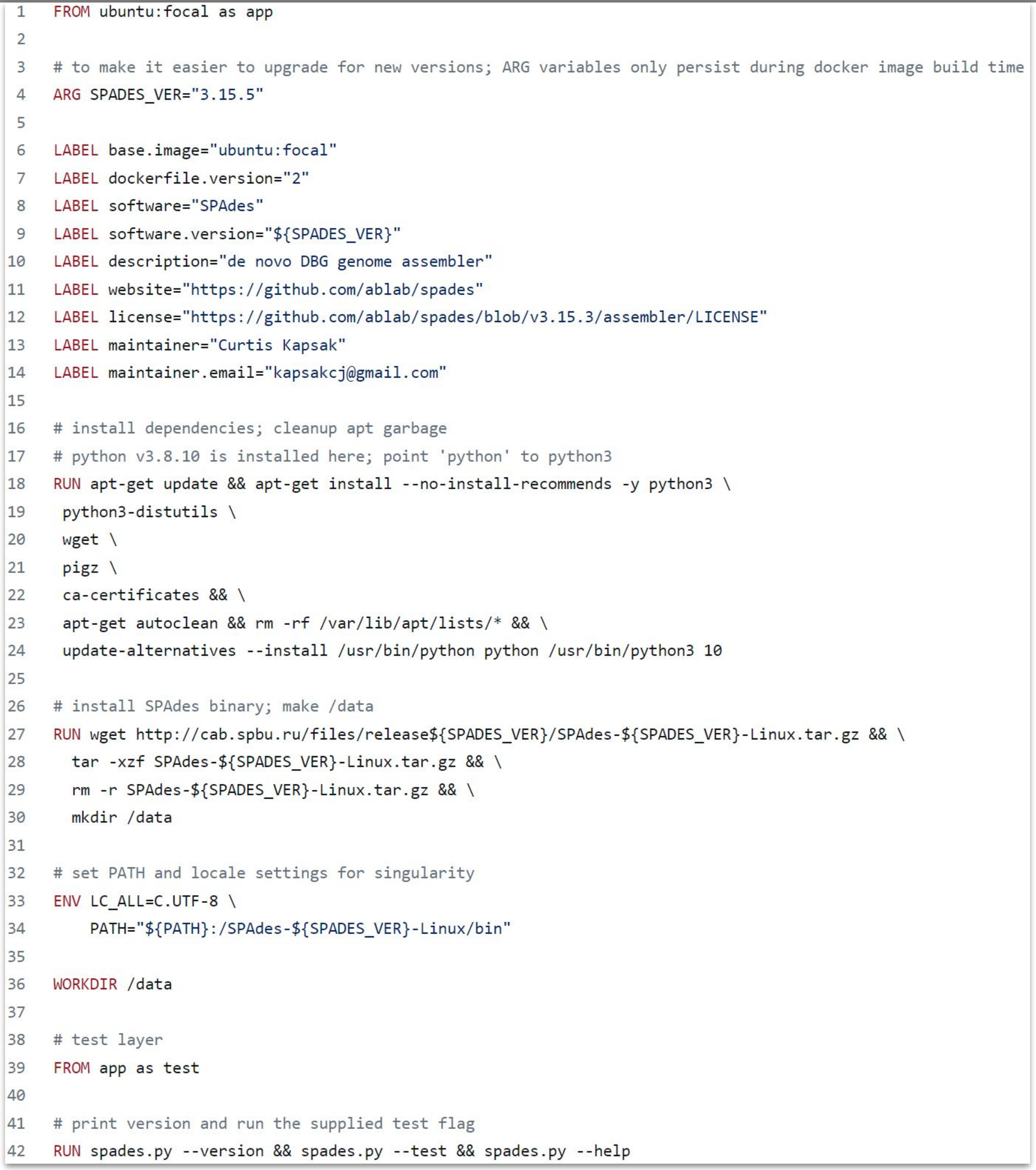
- Dockerfile instructions (`FROM`, `RUN`, `COPY`, `ENV`, etc.) will add a “layer” to the docker image
- Images are multi-layered and different images may share layers like the base image
  - `FROM ubuntu:focal`

Spades Dockerfile

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

Dockerfile Cheat Sheet

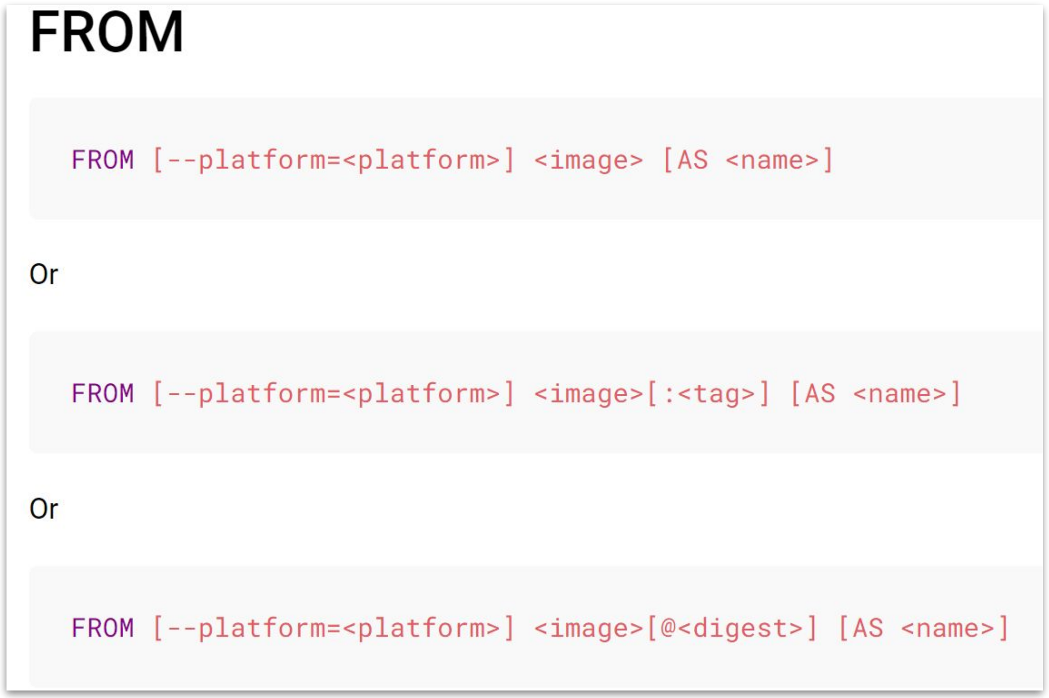
- https://kapeli.com/cheat_sheets/Dockerfile.docset/Contents/Resources/Documents/index



```
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
```

Dockerfile - **FROM**

- Initializes a new build stage
- Required - A valid Dockerfile must start with a **FROM** instruction
    - The only instruction that can precede **FROM** is an **ARG** variable (more on this later)
- **FROM** defines the base image
    - Recommendation - choose a base image and stick with it
- Official docs: https://docs.docker.com/engine/reference/builder/#from



FROM

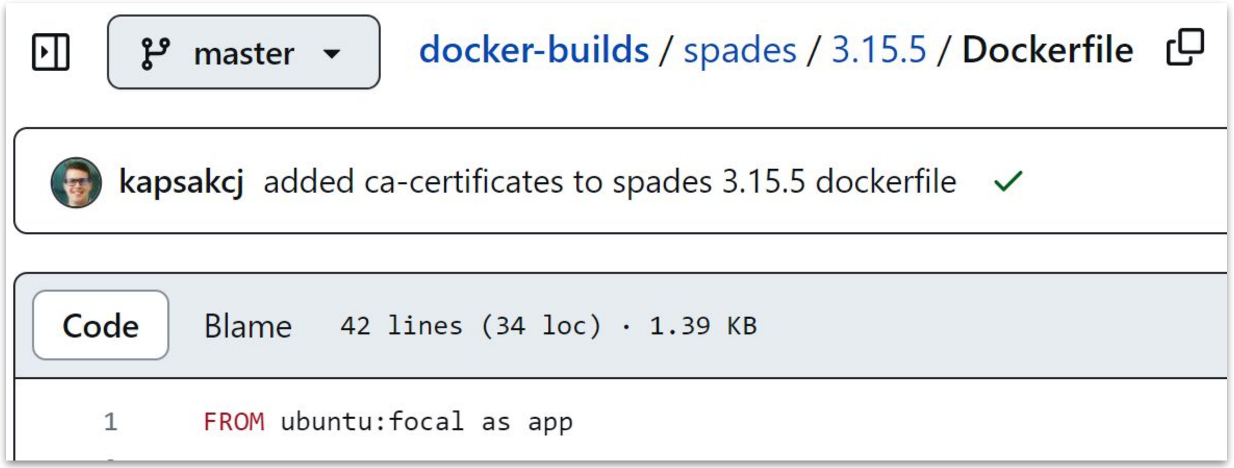
```
FROM [--platform=<platform>] <image> [AS <name>]
```

Or

```
FROM [--platform=<platform>] <image>[:<tag>] [AS <name>]
```

Or

```
FROM [--platform=<platform>] <image>[@<digest>] [AS <name>]
```



master docker-builds / spades / 3.15.5 / Dockerfile

kapsakcj added ca-certificates to spades 3.15.5 dockerfile ✓

Code Blame 42 lines (34 loc) · 1.39 KB

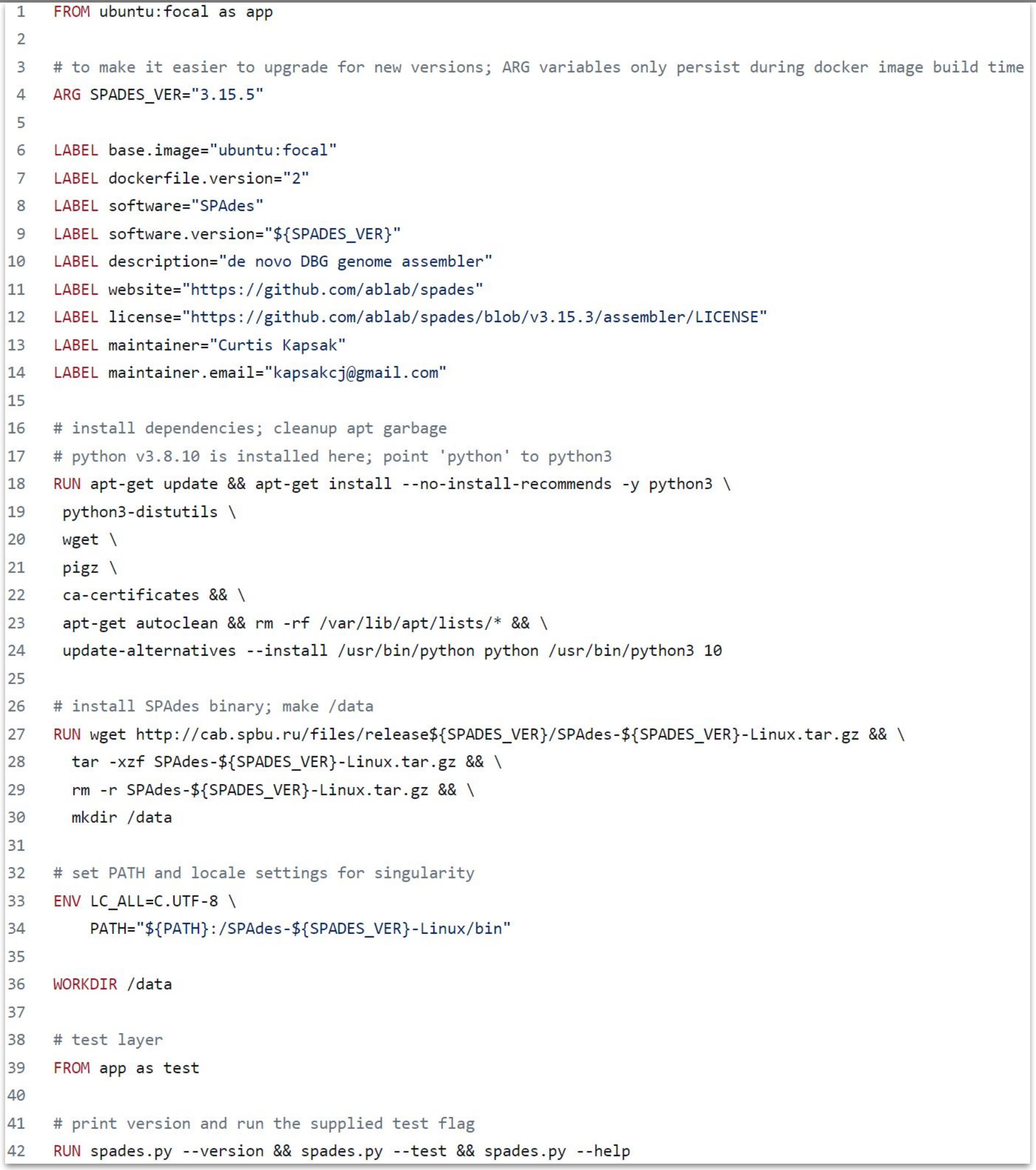
```
FROM ubuntu:focal as app
```

Dockerfile - **FROM**

My favorite base images

Ubuntu - hub.docker.com/_/ubuntu

- Familiar linux OS - has many basic linux commands installed (**`ls`**, **`cd`**, **`cp`**, **`mv`**, **`ps`**, etc.)
- Relatively easy to install dependencies via **`apt-get`**
    - packages.ubuntu.com for looking up what is available via **`apt-get`**
- Can use **`pip`** for python packages & **`cpan/cpanm`** for perl packages



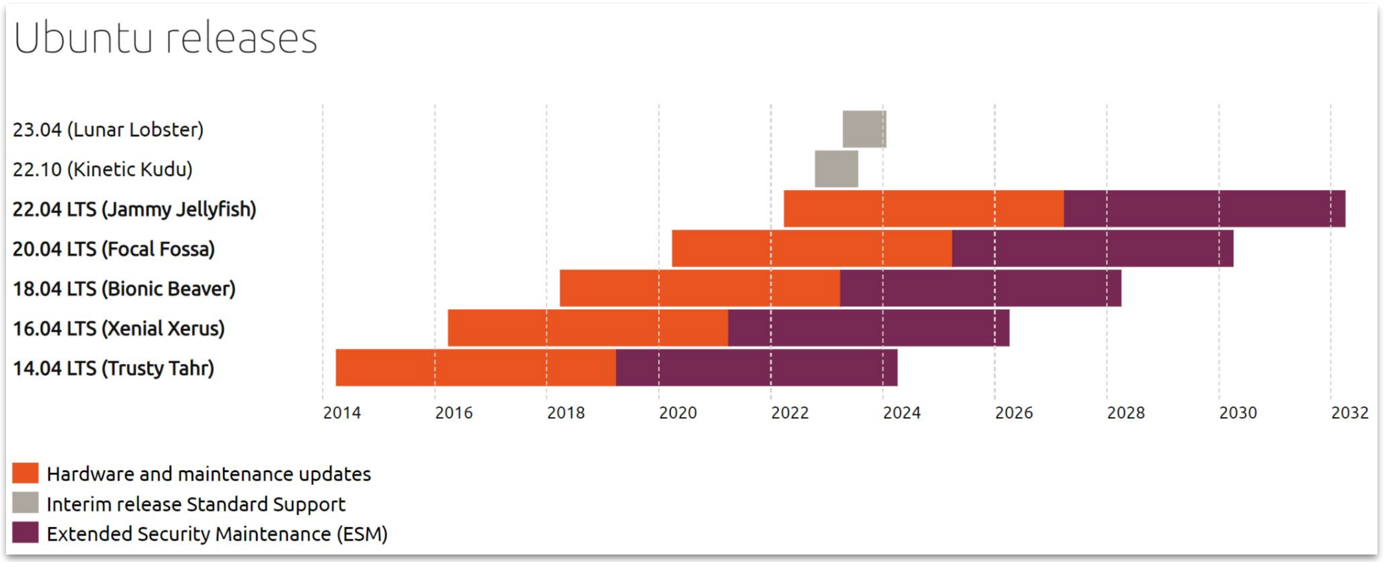
```
FROM ubuntu:focal as app

# 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
```

Dockerfile - FROM

My favorite base images

Ubuntu - hub.docker.com/_/ubuntu

- Recommendation
  - Ubuntu 20.04 LTS (focal)
    - `FROM ubuntu:focal`
  - Ubuntu 22.04 LTS (jammy)
    - `FROM ubuntu:jammy`
  - Or the next LTS (Long Term Support) release
- Older Ubuntu releases are nearing the end of support, use something new that will be supported long term!



https://ubuntu.com/about/release-cycle#ubuntu

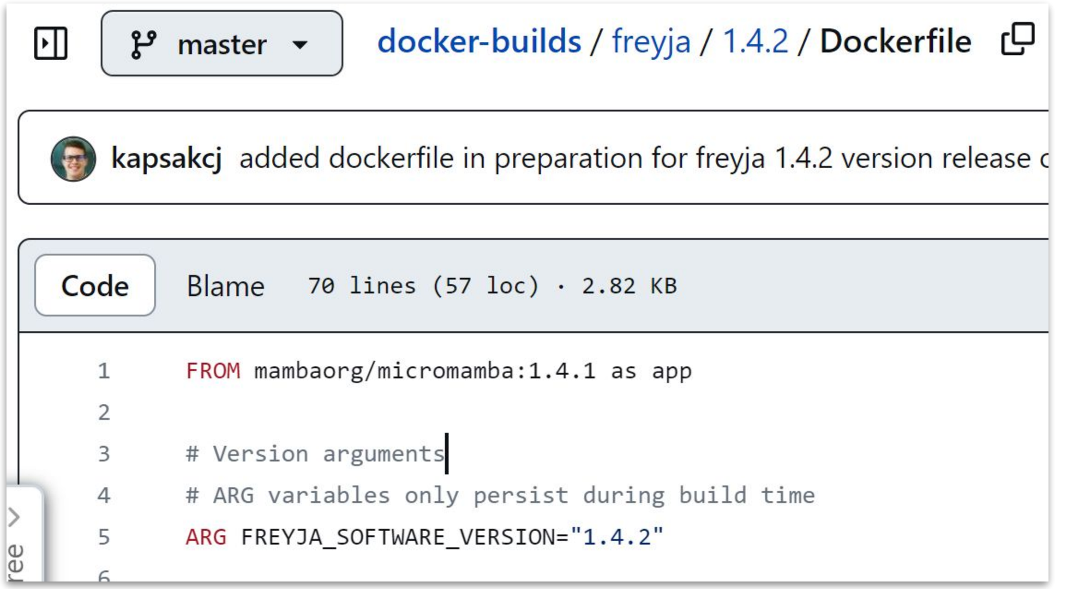
Dockerfile - FROM

My favorite base images:

micromamba -

hub.docker.com/r/mambaorg/micromamba

- Familiar linux OS (Debian)
- **`micromamba`** is preinstalled
  - **`micromamba`** is even more lightweight than conda or miniconda
- I use for complicated installations where I rely upon the conda package in bioconda ← not best practice!
- Also use for scenarios where I want to use a conda recipe file (.yml) for installation


master

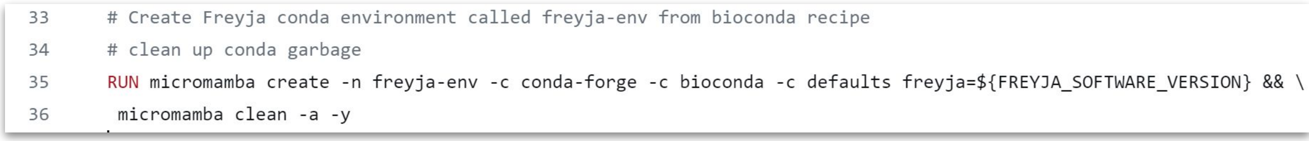
docker-builds / freyja / 1.4.2 / Dockerfile

kapsakcj added dockerfile in preparation for freyja 1.4.2 version release c

Code Blame 70 lines (57 loc) · 2.82 KB

```
FROM mambaorg/micromamba:1.4.1 as app

# Version arguments
# ARG variables only persist during build time
ARG FREYJA_SOFTWARE_VERSION="1.4.2"
```

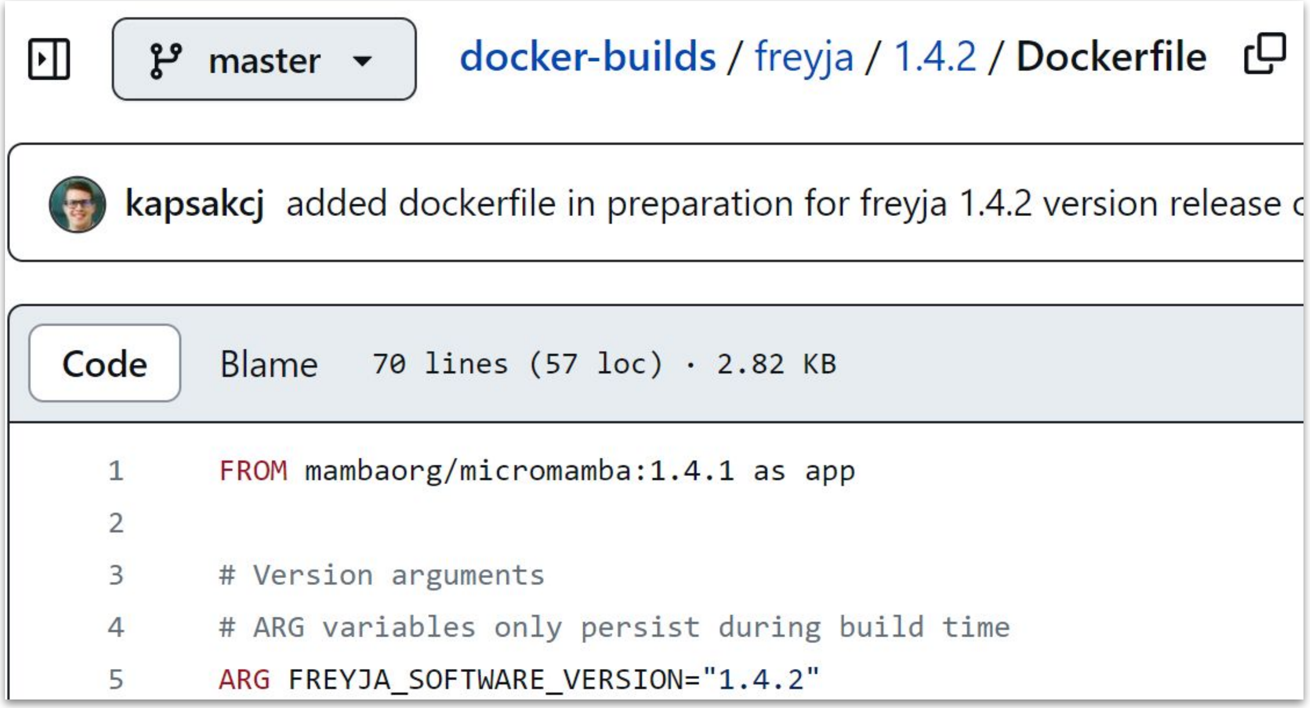
later in Freyja dockerfile ⬇️


```
# Create Freyja conda environment called freyja-env from bioconda recipe
# clean up conda garbage
RUN micromamba create -n freyja-env -c conda-forge -c bioconda -c defaults freyja=${FREYJA_SOFTWARE_VERSION} && \
 micromamba clean -a -y
```

Dockerfile - ARG

- Sets environmental variables that are ONLY available during docker image build time
- Once image is built, all **ARG** variables are unset/removed
- Useful for specifying versions of tools to install; can make it easy to upgrade versions by only changing one line of code

beginning of Freyja dockerfile ⬇️

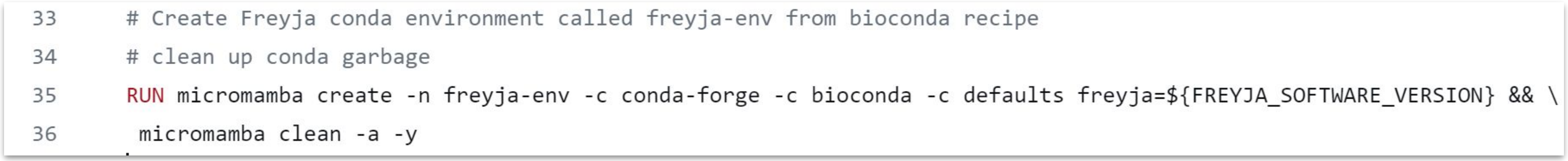


kapsakcj added dockerfile in preparation for freyja 1.4.2 version release c

70 lines (57 loc) · 2.82 KB

```
FROM mambaorg/micromamba:1.4.1 as app

# Version arguments
# ARG variables only persist during build time
ARG FREYJA_SOFTWARE_VERSION="1.4.2"
```

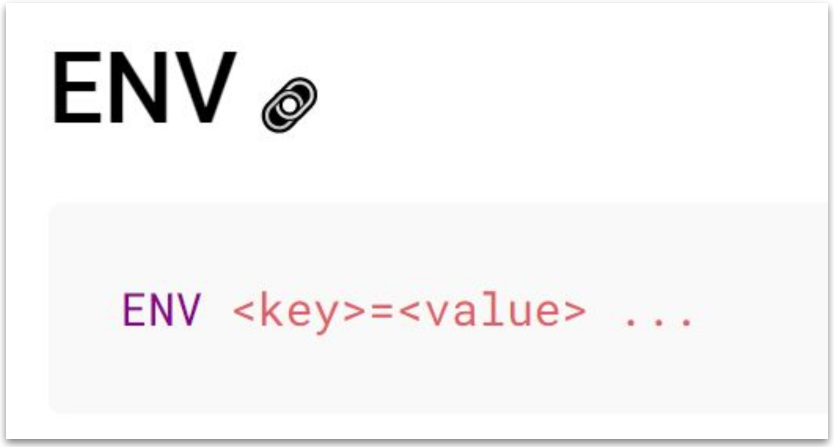
later in Freyja dockerfile ⬇️



```
# Create Freyja conda environment called freyja-env from bioconda recipe
# clean up conda garbage
RUN micromamba create -n freyja-env -c conda-forge -c bioconda -c defaults freyja=${FREYJA_SOFTWARE_VERSION} && \
 micromamba clean -a -y
```

Dockerfile - ENV

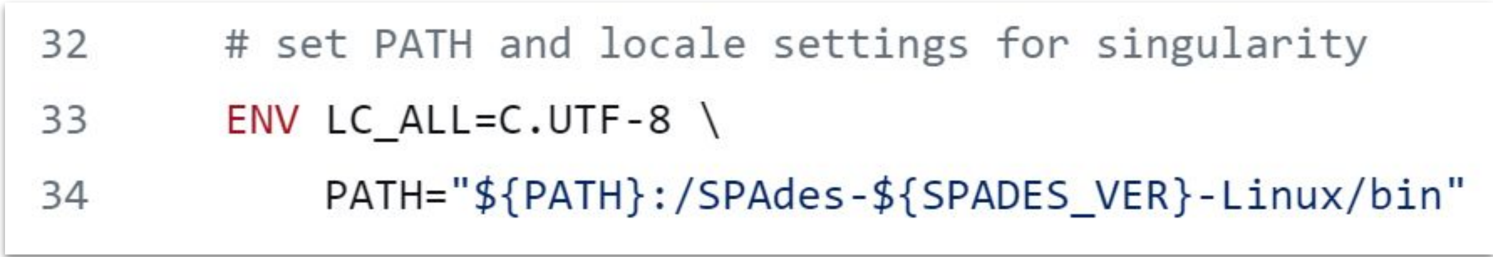
- Sets permanent environmental variables that are available during image build and afterwards for all users
- Useful for setting the **$PATH** variable
- Format:



ENV

```
ENV <key>=<value> ...
```

SPAdes dockerfile example:

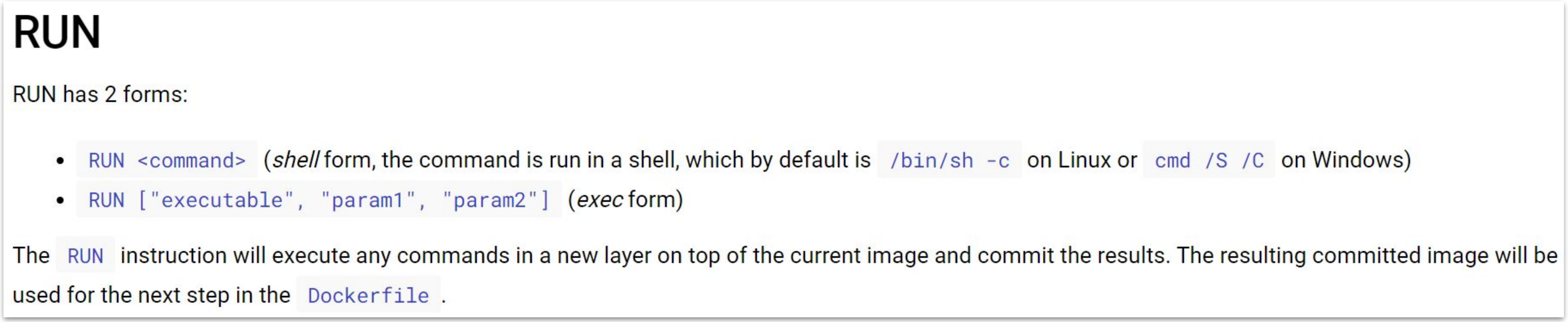


```
    # set PATH and locale settings for singularity
    ENV LC_ALL=C.UTF-8 \
        PATH="${PATH}:/SPAdes-${SPADES_VER}-Linux/bin"
```

- Can set multiple variables in one **ENV** layer using line breaks with a backslash \
- Useful for tools that do not automatically get added to your **$PATH** variable

Dockerfile - **RUN**

- Executes a command in a new layer
- Each **`RUN`** layer builds upon the previous **`FROM`** and **`RUN`** layers
- Changes made in **`RUN`** commands are saved in the final docker image
- Assume **`/bin/sh`** shell for running commands
- Docker official docs: https://docs.docker.com/engine/reference/builder/#run



RUN

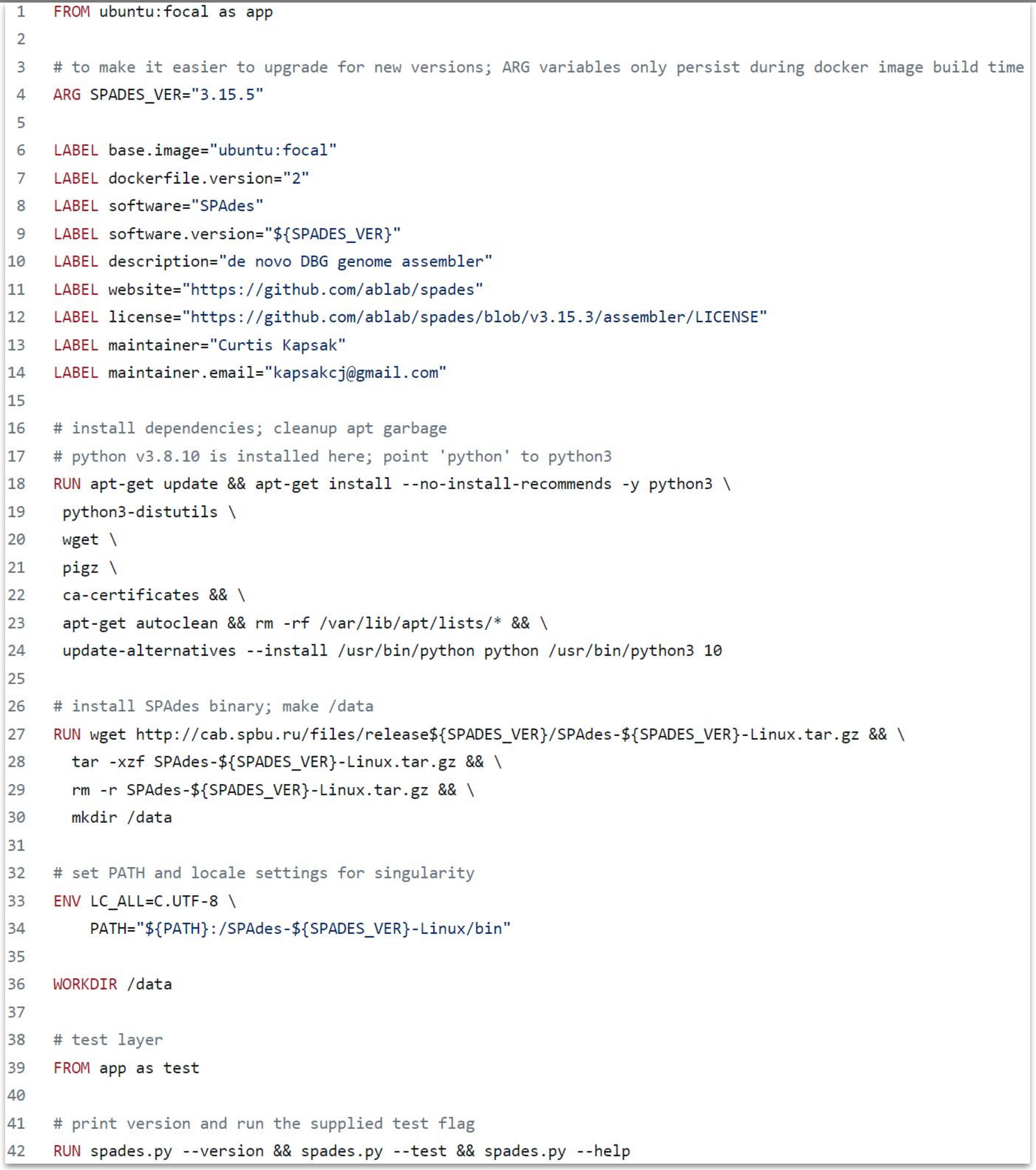
RUN has 2 forms:

- `RUN <command>` (*shell* form, the command is run in a shell, which by default is `/bin/sh -c` on Linux or `cmd /S /C` on Windows)
- `RUN ["executable", "param1", "param2"]` (*exec* form)

The `RUN` instruction will execute any commands in a new layer on top of the current image and commit the results. The resulting committed image will be used for the next step in the `Dockerfile`.

Dockerfile - RUN

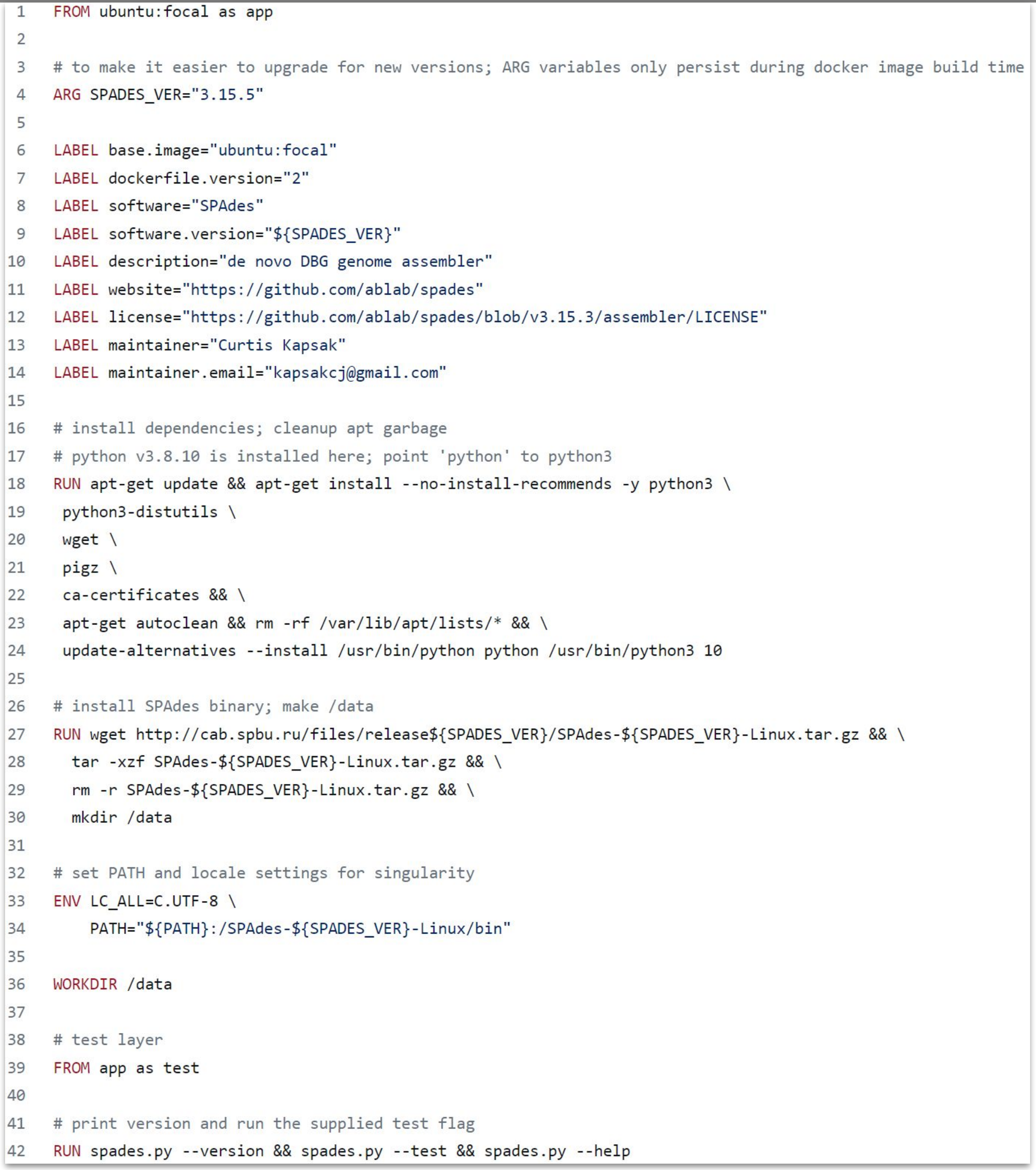
- Look at a Dockerfile in-depth: SPAdes
- Tricks-of-the-trade:
    - **`apt-get update && apt-get install`** in a single **RUN** statement, done early in dockerfile



```
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
```

Dockerfile - **RUN**

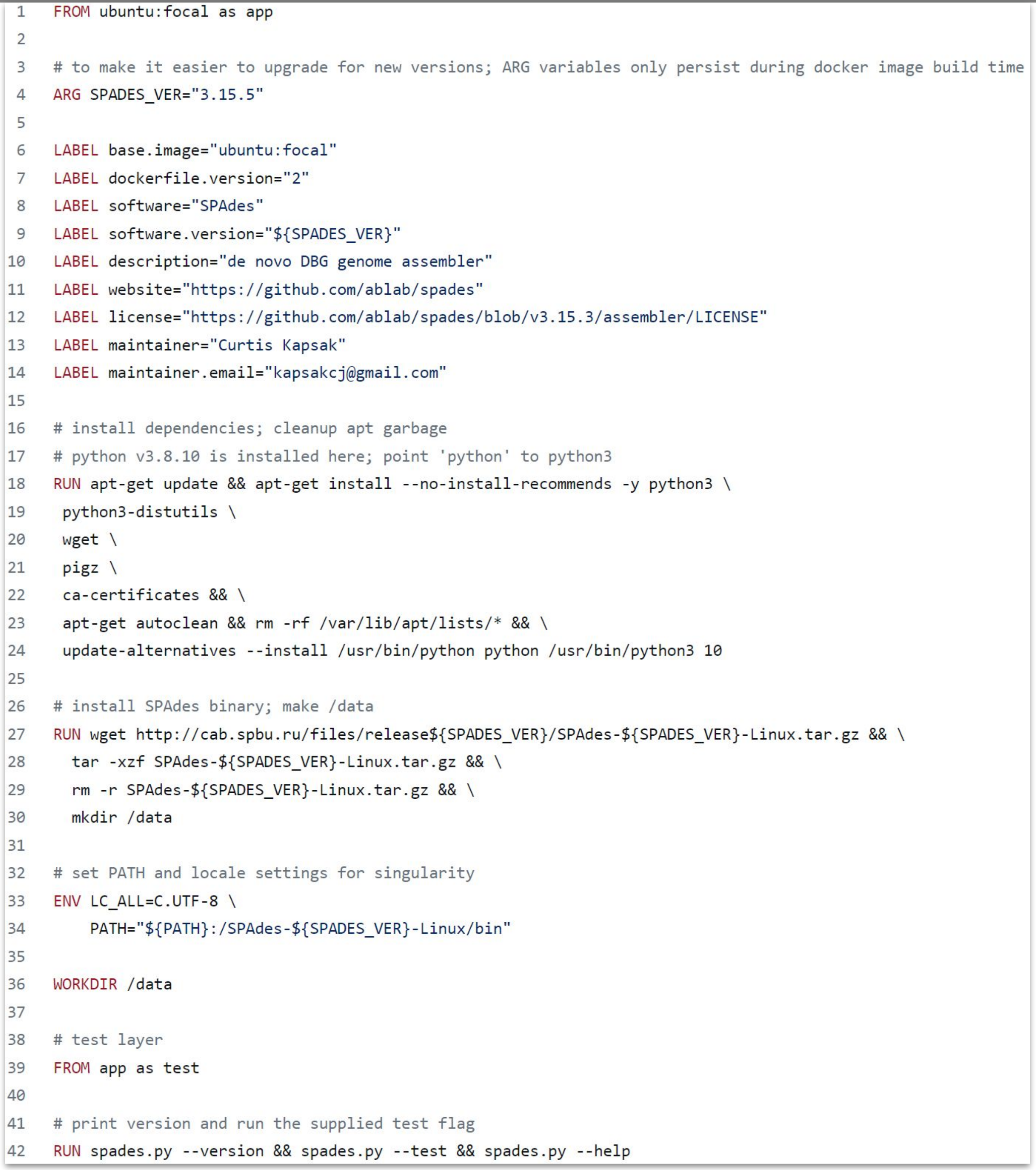
- Look at a Dockerfile in-depth: SPAdes
- Tricks-of-the-trade:
  - **`apt-get update && apt-get install`** in a single **`RUN`** statement, done early in dockerfile
  - Use syntax appropriate for your build stage (shell in ubuntu). Imagine you are using the command line



```
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
```

Dockerfile - **RUN**

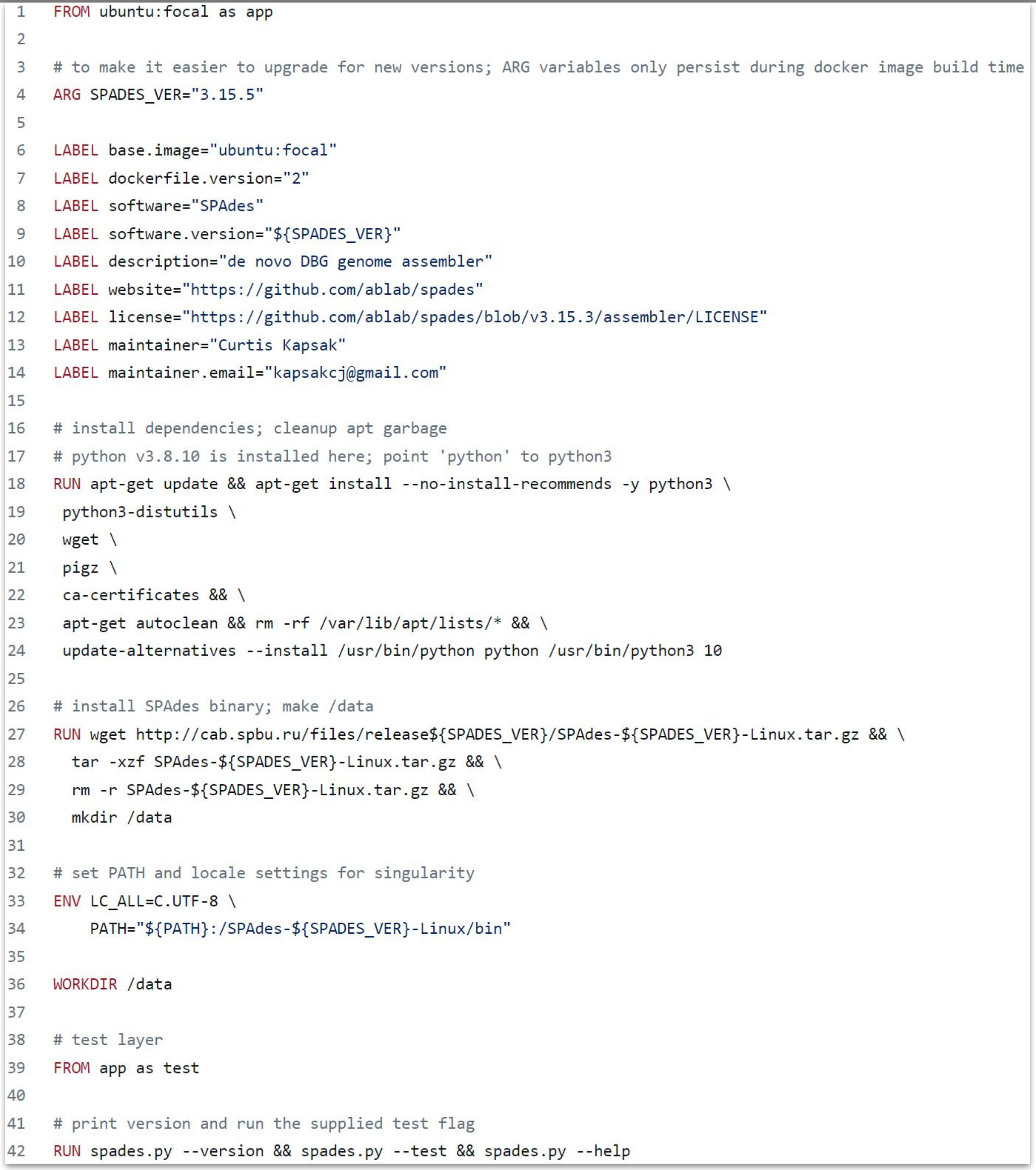
- Look at a Dockerfile in-depth: SPAdes
- Tricks-of-the-trade:
  - **`apt-get update && apt-get install`** in a single **`RUN`** statement, done early in dockerfile
  - Use syntax appropriate for your build stage (shell in ubuntu). Imagine you are using the command line
  - \ (backslashes) are for line breaks (readability)



```
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
```

Dockerfile - **RUN**

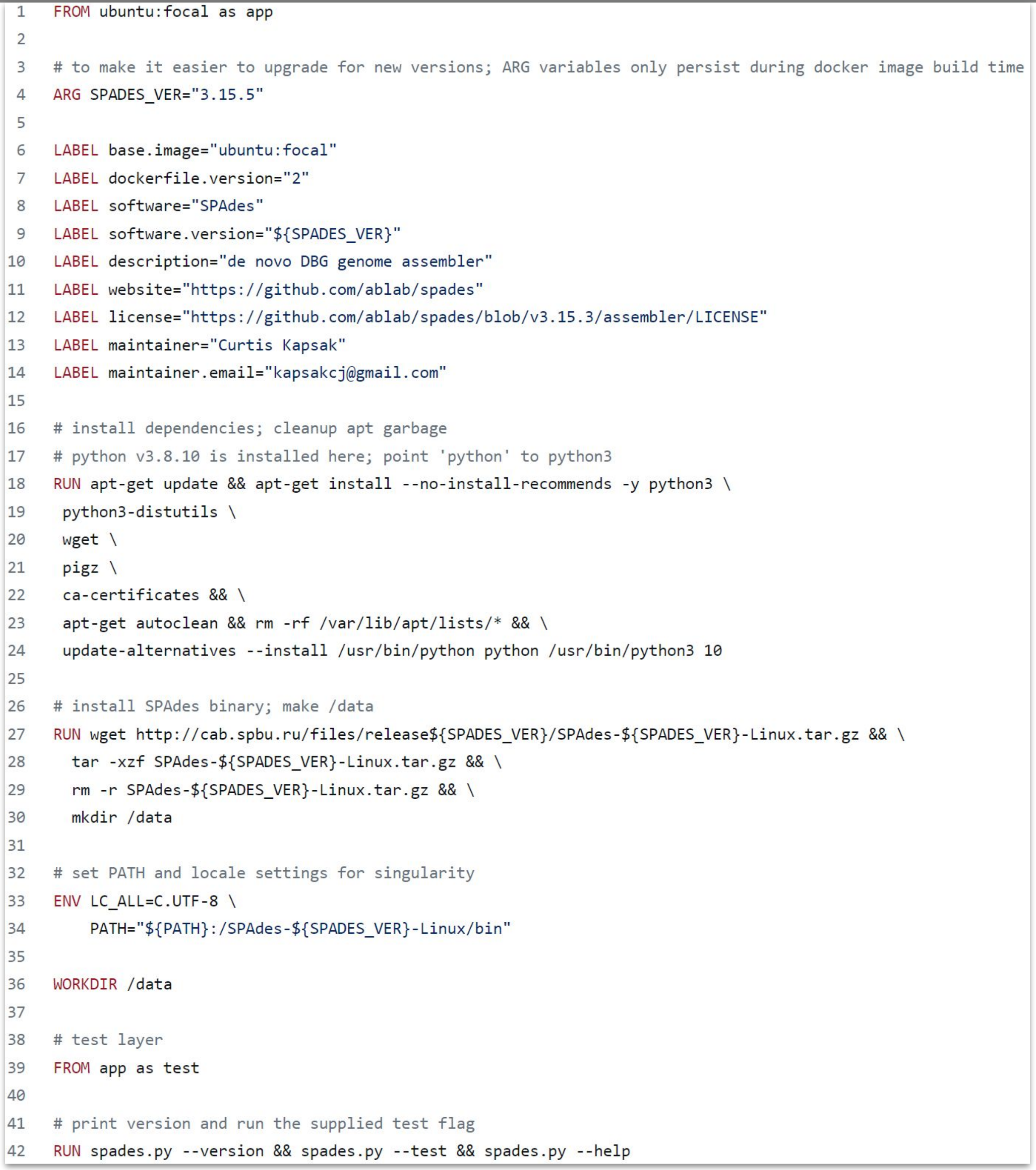
- Look at a Dockerfile in-depth: SPAdes
- Tricks-of-the-trade:
  - **`apt-get update && apt-get install`** in a single **`RUN`** statement, done early in dockerfile
  - Use syntax appropriate for your build stage (shell in ubuntu). Imagine you are using the command line
  - \ (backslashes) are for line breaks (readability)
  - **`&&`** bash operator is used to create long one-liners so that multiple commands are run sequentially and are dependent on each command running successfully



```dockerfile
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
```

Dockerfile - **RUN**

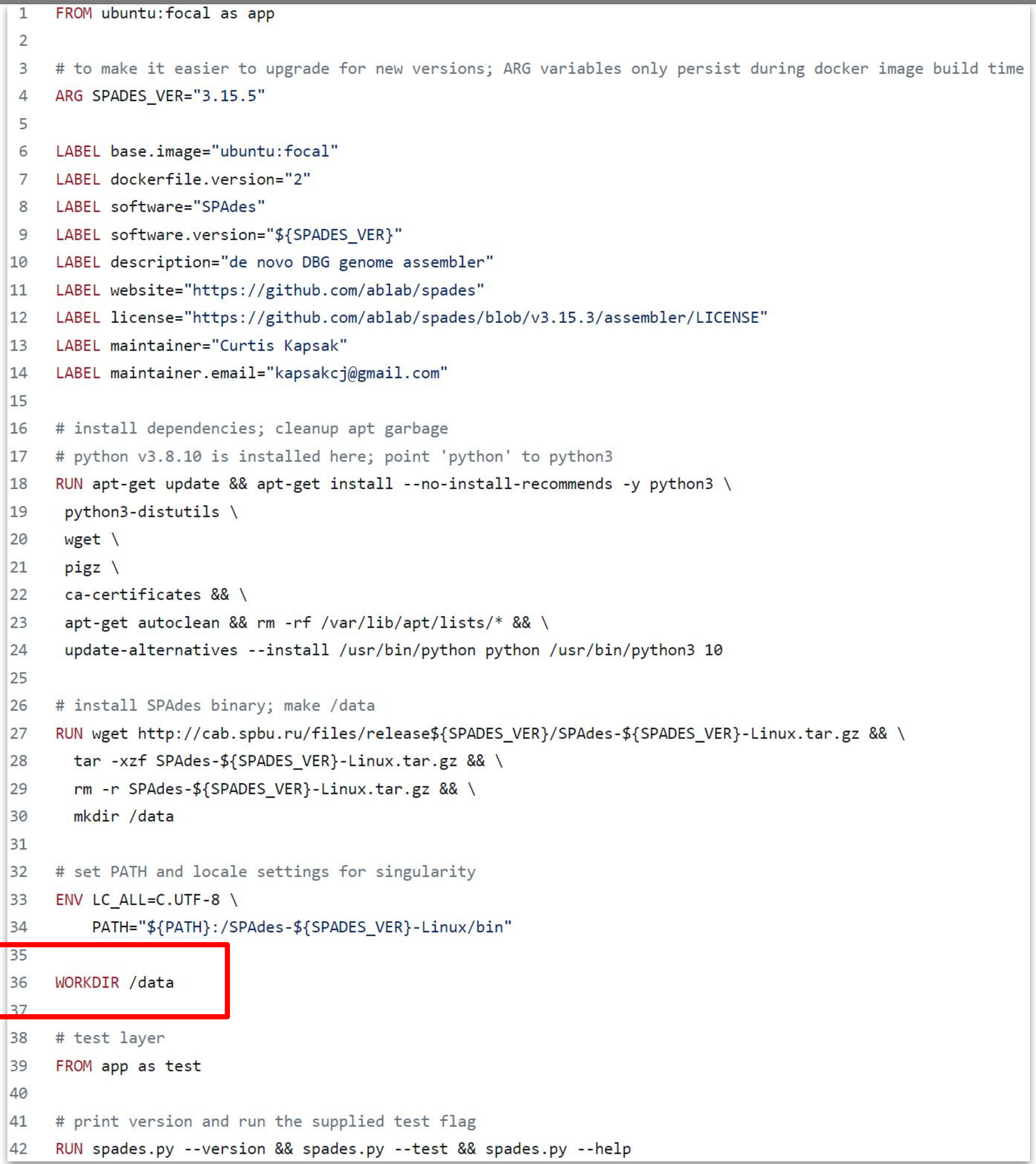
- Look at a Dockerfile in-depth: SPAdes
- Tricks-of-the-trade:
    - **`apt-get update && apt-get install`** in a single **`RUN`** statement, done early in dockerfile
    - Use syntax appropriate for your build stage (shell in ubuntu). Imagine you are using the command line
    - **`\`** (backslashes) are for line breaks (readability)
    - **`&&`** bash operator is used to create long one-liners so that multiple commands are run sequentially and are dependent on each command running successfully
    - Assume **`/bin/sh`** shell for running commands, but there is a way to set `**`/bin/bash`**` as normal shell. Not necessary unless using bash-specific cmdline tricks



```
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
```

Dockerfile - **WORKDIR**

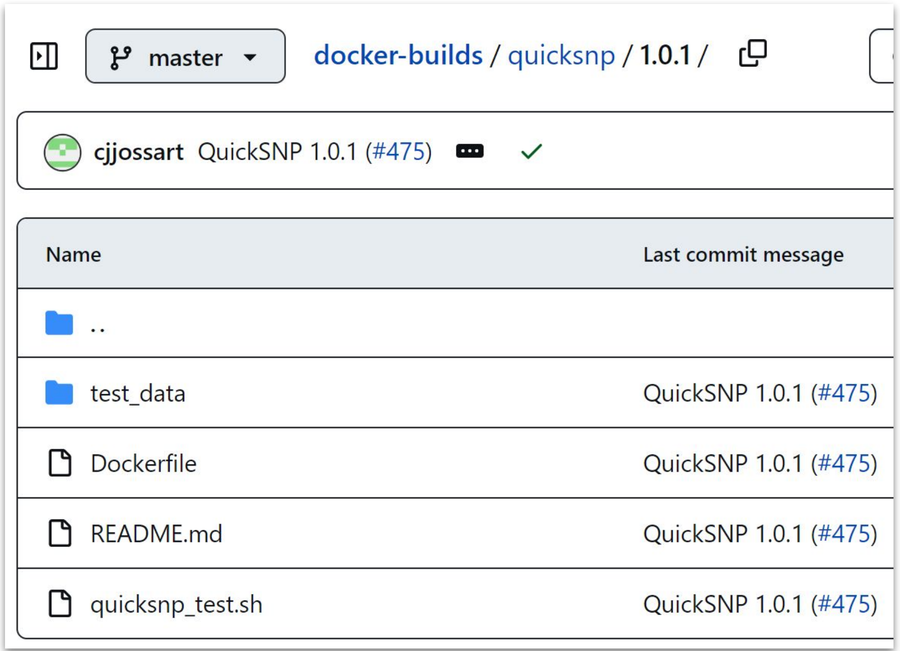
- Sets the working directory for any **RUN**, **CMD**, **ENTRYPOINT**, **COPY** and **ADD** instructions that follow it in the Dockerfile.
- **WORKDIR** will also persist after the image is build
- Can use multiple **WORKDIR**'s in one dockerfile, but only last **WORKDIR** will be saved in final image
- All StaPH-B/docker-builds images have the final **WORKDIR** set to **/data**
  - used for passing data in and out of containers
  - does not overlap with existing directories in container filesystem



```
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
```

Dockerfile - COPY

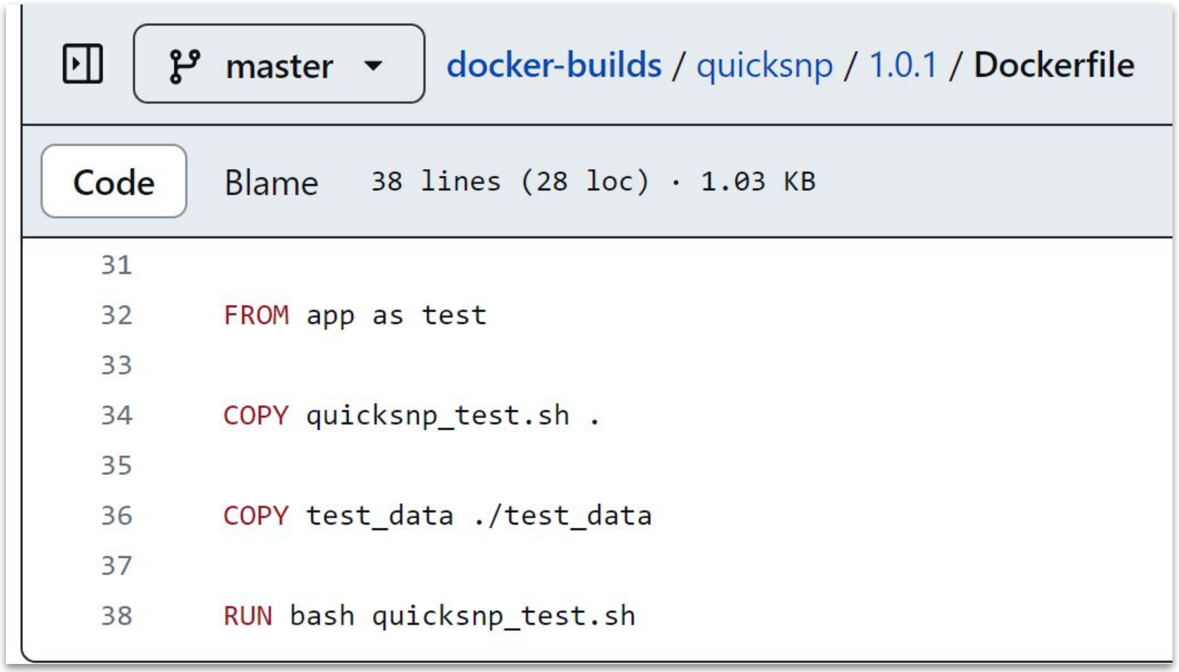
- Copies files into the image
- Files must be either:
  - located in same directory as Dockerfile (AKA the "build context")
  - From another stage of the build process
- Useful for copying in test data, (small) databases, additional code:



master ▾ docker-builds / quicksnp / 1.0.1 /

cjjossart QuickSNP 1.0.1 (#475) ✓

Name	Last commit message
..	
test_data	QuickSNP 1.0.1 (#475)
Dockerfile	QuickSNP 1.0.1 (#475)
README.md	QuickSNP 1.0.1 (#475)
quicksnp_test.sh	QuickSNP 1.0.1 (#475)



master ▾ docker-builds / quicksnp / 1.0.1 / Dockerfile

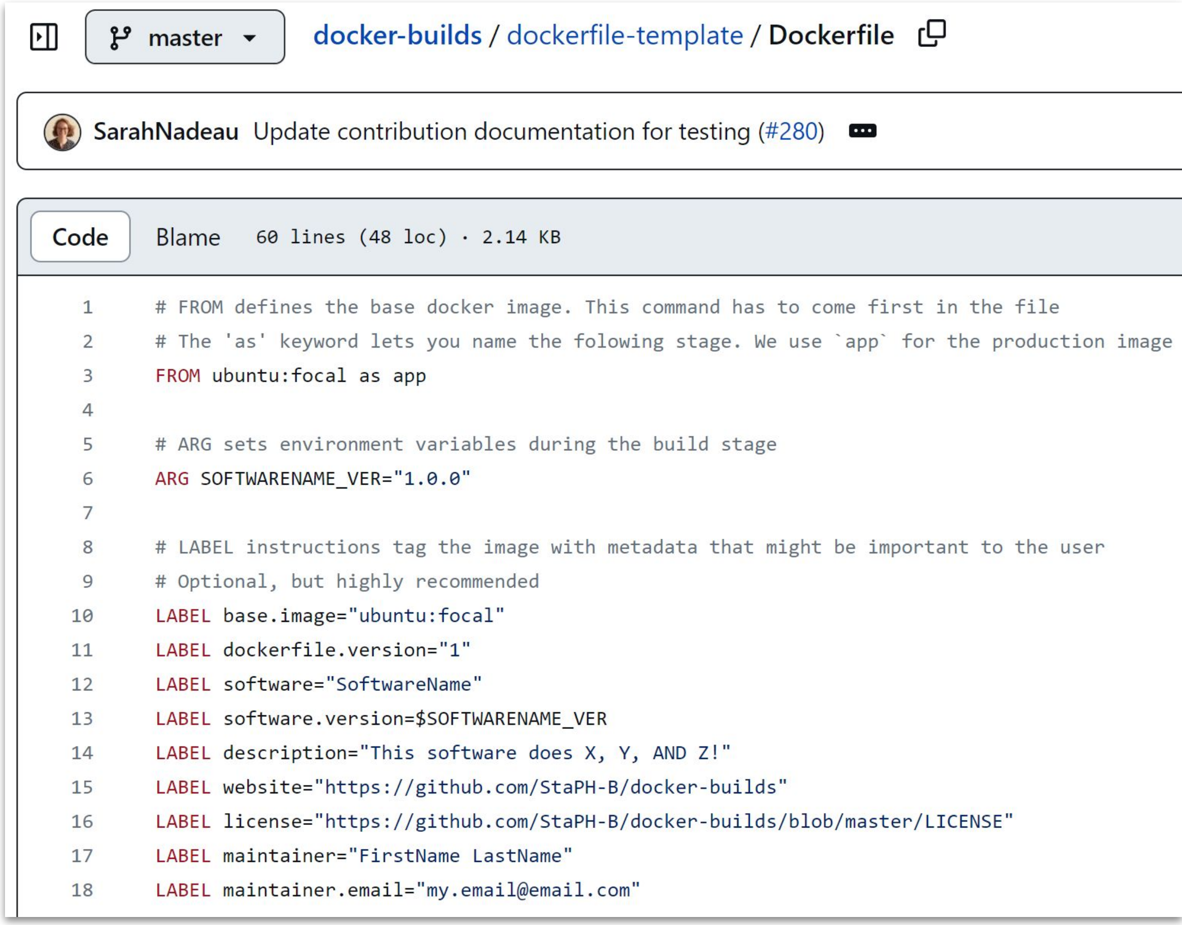
Code Blame 38 lines (28 loc) · 1.03 KB

```

FROM app as test

COPY quicksnp_test.sh .

COPY test_data ./test_data

RUN bash quicksnp_test.sh
```

Dockerfile - LABEL

- optional but highly recommended!
- allows addition of metadata to your docker image
- generally located near top of dockerfile, after **FROM**
- StaPH-B has some required labels for dockerfiles contributed to their repo
  - See the template Dockerfile for examples:

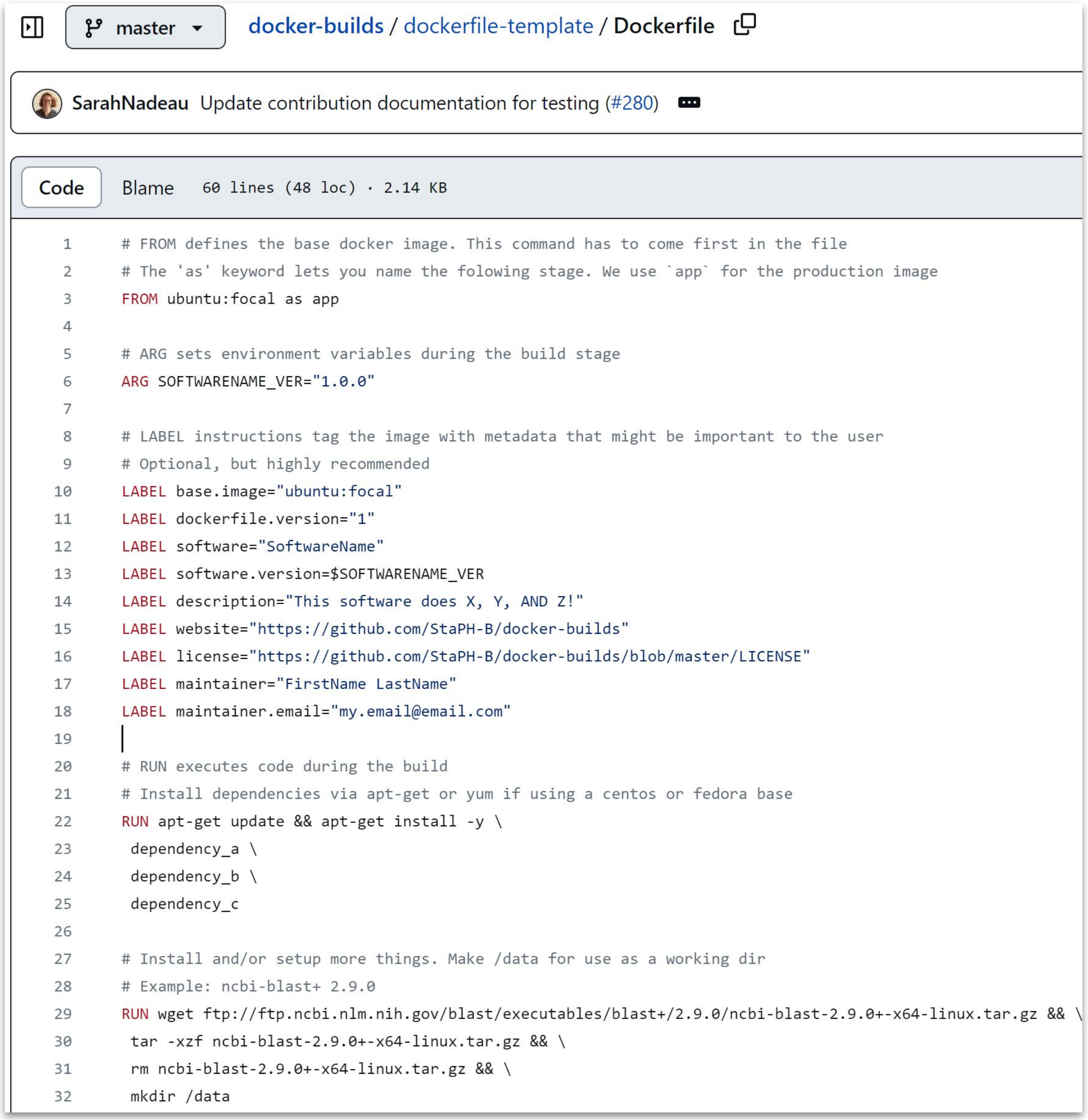
https://github.com/StaPH-B/docker-builds/blob/master/dockerfile-template/Dockerfile


master
docker-builds / dockerfile-template / Dockerfile
SarahNadeau Update contribution documentation for testing (#280)
Code
Blame
60 lines (48 loc) · 2.14 KB

```
# FROM defines the base docker image. This command has to come first in the file
# The 'as' keyword lets you name the folowing stage. We use `app` for the production image
FROM ubuntu:focal as app

# ARG sets environment variables during the build stage
ARG SOFTWARENAME_VER="1.0.0"

# LABEL instructions tag the image with metadata that might be important to the user
# Optional, but highly recommended
LABEL base.image="ubuntu:focal"
LABEL dockerfile.version="1"
LABEL software="SoftwareName"
LABEL software.version=$SOFTWARENAME_VER
LABEL description="This software does X, Y, AND Z!"
LABEL website="https://github.com/StaPH-B/docker-builds"
LABEL license="https://github.com/StaPH-B/docker-builds/blob/master/LICENSE"
LABEL maintainer="FirstName LastName"
LABEL maintainer.email="my.email@email.com"
```

Dockerfile - `comments`

- also optional but highly recommended!
- comment lines begin with **#**



master

docker-builds / dockerfile-template / Dockerfile

SarahNadeau Update contribution documentation for testing (#280)

Code Blame 60 lines (48 loc) · 2.14 KB

```dockerfile
# FROM defines the base docker image. This command has to come first in the file
# The 'as' keyword lets you name the folowing stage. We use `app` for the production image
FROM ubuntu:focal as app

# ARG sets environment variables during the build stage
ARG SOFTWARENAME_VER="1.0.0"

# LABEL instructions tag the image with metadata that might be important to the user
# Optional, but highly recommended
LABEL base.image="ubuntu:focal"
LABEL dockerfile.version="1"
LABEL software="SoftwareName"
LABEL software.version=$SOFTWARENAME_VER
LABEL description="This software does X, Y, AND Z!"
LABEL website="https://github.com/StaPH-B/docker-builds"
LABEL license="https://github.com/StaPH-B/docker-builds/blob/master/LICENSE"
LABEL maintainer="FirstName LastName"
LABEL maintainer.email="my.email@email.com"

# RUN executes code during the build
# Install dependencies via apt-get or yum if using a centos or fedora base
RUN apt-get update && apt-get install -y \
 dependency_a \
 dependency_b \
 dependency_c

# Install and/or setup more things. Make /data for use as a working dir
# Example: ncbi-blast+ 2.9.0
RUN wget ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.9.0/ncbi-blast-2.9.0+-x64-linux.tar.gz && \
 tar -xzf ncbi-blast-2.9.0+-x64-linux.tar.gz && \
 rm ncbi-blast-2.9.0+-x64-linux.tar.gz && \
 mkdir /data
```

Break

5 min break!

resume at 3:38pm

Quiz time:

- What is the base image used in iVar 1.4.1 dockerfile?
    - https://github.com/StaPH-B/docker-builds/blob/master/ivar/1.4.2/Dockerfile
- What **ARG** variables are used and what are they used for?
- How is `ivar` installed? What steps are taken?

Dockerfile → Docker image

`docker build`

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/
```

Multi-stage docker builds

- Docker images that use multiple **FROM** instructions
  - can use a new base image **OR**
  - can use previous stage
- Useful for "optimizing" Dockerfiles. Optimizing =
  - reducing number of layers
  - removing unnecessary software & files, etc. to reduce final size of docker image
- Not required in general, but are required for StaPH-B/docker-builds project

General format in Dockerfile (using previous stage as new **FROM** layer):

```
FROM baseimage as name
RUN stuff

FROM name as name2
RUN stuff

FROM name2 as final
RUN stuff
```

Multi-stage docker builds

General format in Dockerfile (<u>using previous stage as new FROM layer</u>):

```
FROM ubuntu:focal as stage1
RUN stuff


FROM stage1 as stage2
RUN stuff


FROM stage2 as finalstage
RUN stuff
```

Multi-stage docker builds

General format in Dockerfile (using new base images as new FROM layer):

```
FROM ubuntu:focal as stage1
RUN stuff

FROM python:slim as stage2
RUN stuff

FROM ubuntu:jammy as finalstage
RUN stuff
```

Multi-stage docker builds

- StaPH-B exclusive
- Require use of 2 stages, **`app`** and **`test`**
- **`app`** stage
  - stage for installing software and dependencies
  - only stage that remains in final docker image
- **`test`** stage
  - based on **`app`** stage
  - runs tests to ensure functionality and correct version of installed software
  - stage is NOT included in final docker image

General format in StaPH-B Dockerfiles:

~~~
```

FROM ubuntu:focal as app
                                   app stage
RUN install software


FROM app as test

RUN my-tests.sh                    test stage

RUN software --version
```
~~~

Multi-stage docker builds

- Let’s switch to GitPod and see this in practice
- Navigate to the training exercise document:
    - 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