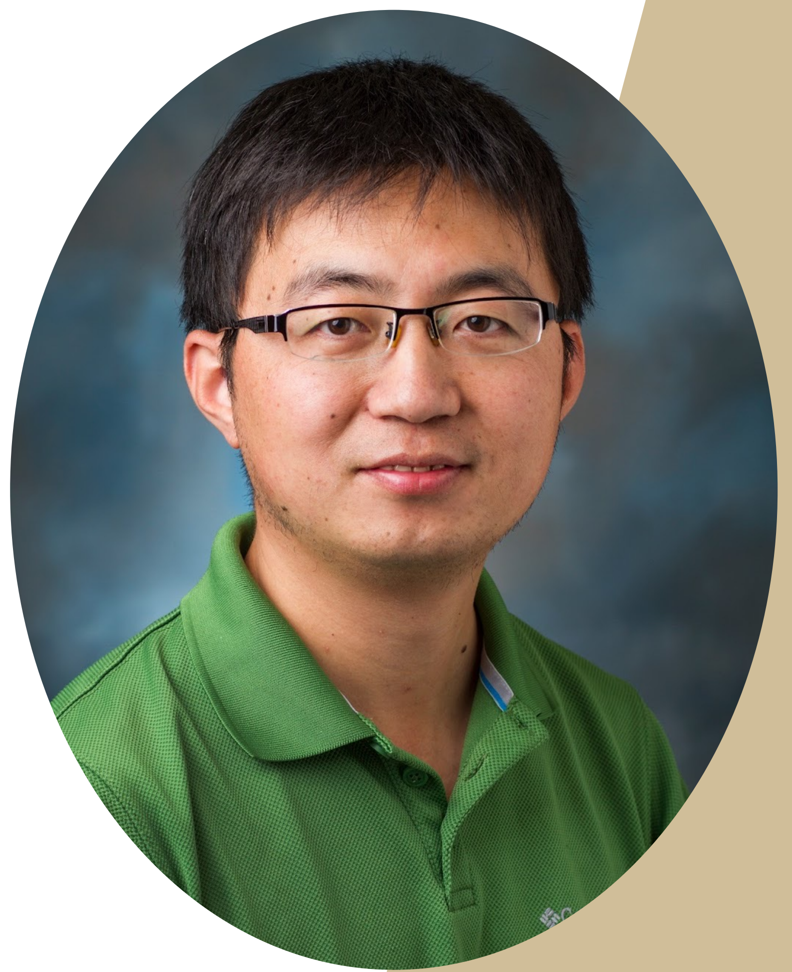
PURDUE UNIVERSITY | Information Technology RESEARCH COMPUTING

Container 101

ITaP Research Computing Virtual Workshop Series

Yucheng Zhang, Lev Gorenstein

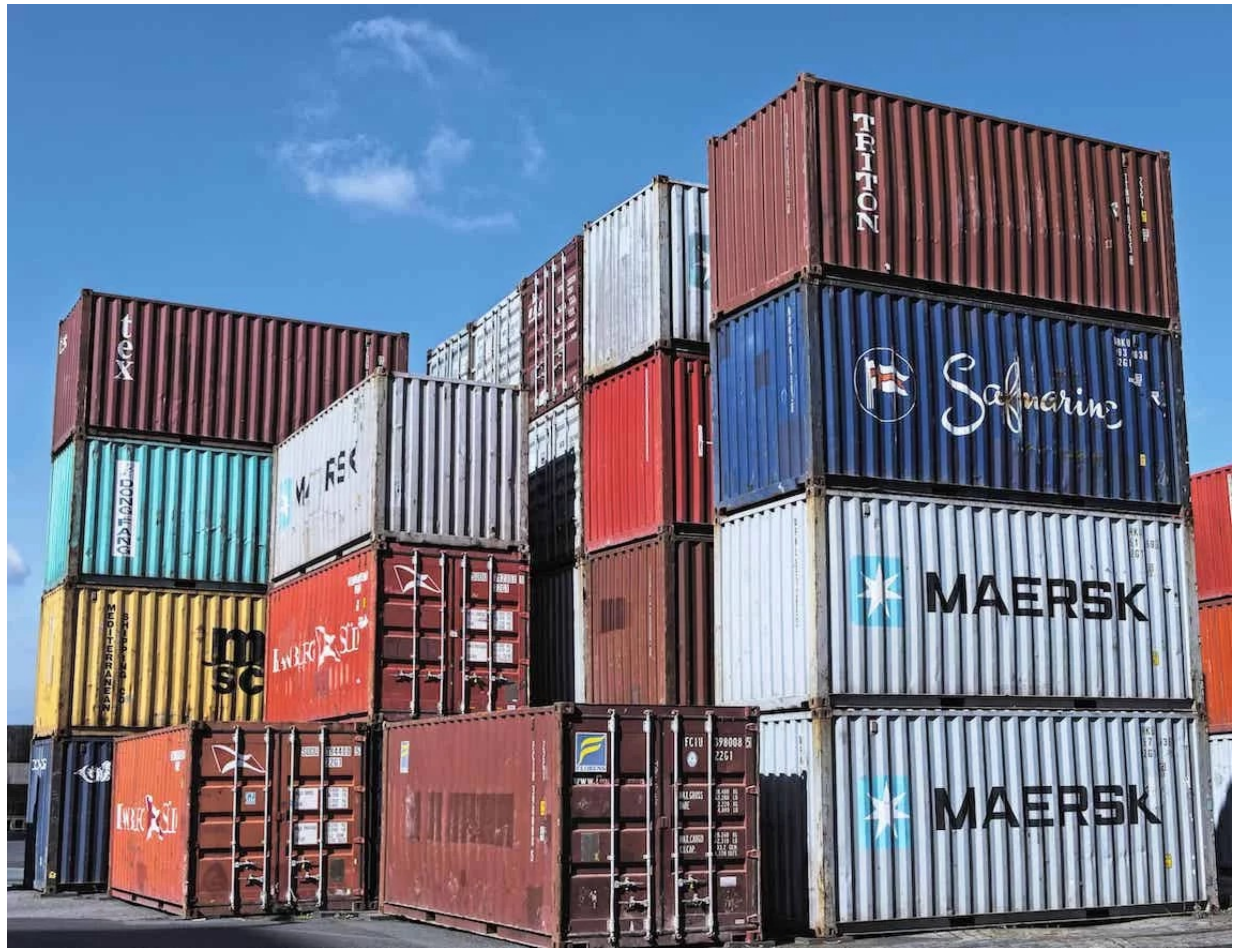


Yucheng Zhang
Senior Life Science Scientist
ITaP Research Computing

Outline

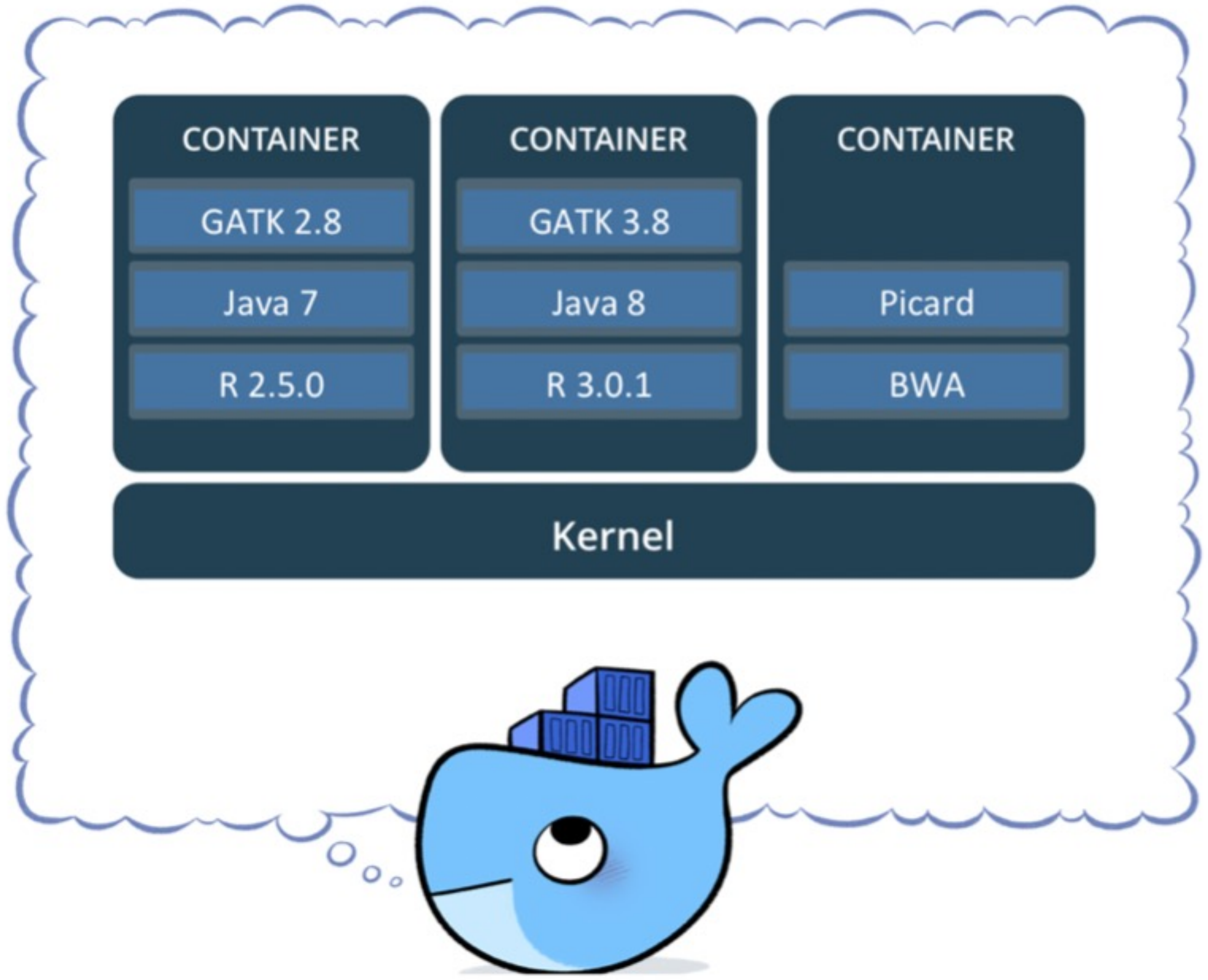
- What are containers and why should we use them?
- Docker and Singularity
- Singularity basics
- Using containers on RCAC clusters
- Deployed containers on RCAC clusters

What are containers?



- The arrival of modern shipping containers changed our transportation industry.
- Container is a standardized way to package items together into one shipment.
  1. Standard packaging
  2. Isolation and efficiency
  3. Portable
  4. Separation of concerns

What are containers?



A **container** is an abstraction for a set of technologies that aim to solve the problem of how to get software to run reliably when moved from one computing environment to another.

A container **image** is simply a file (or collection of files) saved on disk that stores everything you need to run a target application or applications.

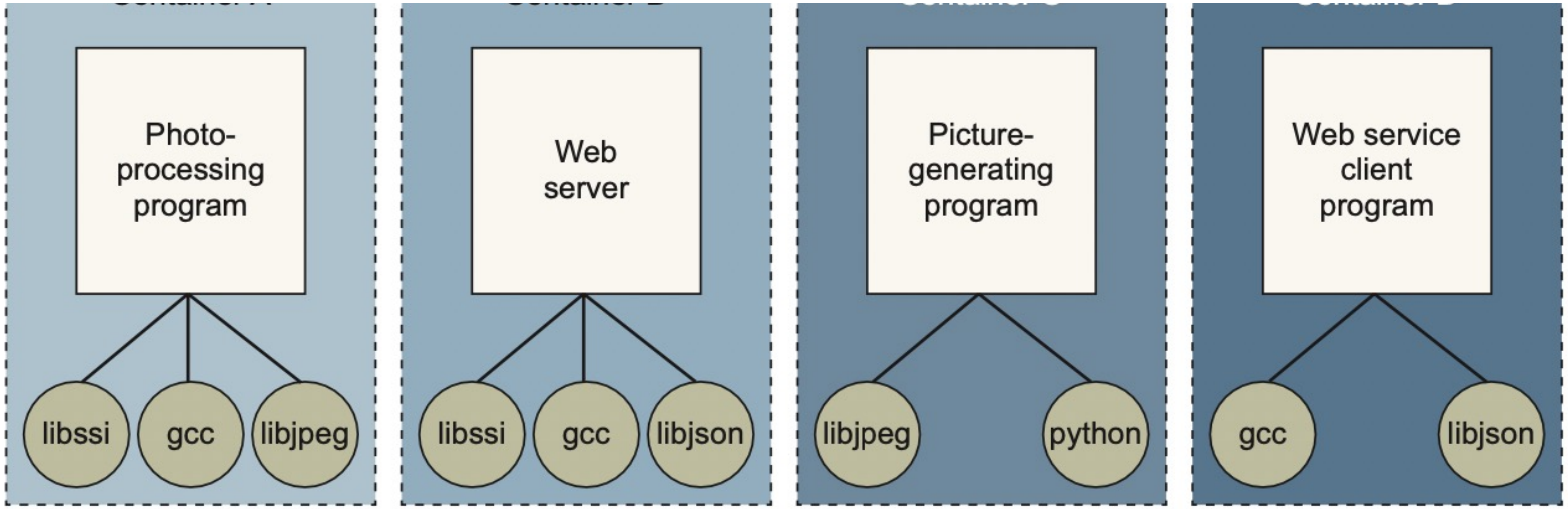
Registry: a place to store (and share) container images.

Source: https://support.terra.bio/hc/en-us/articles/360037340472-Docker-container-overview

Why should we use containers?

Getting organized

- ❖ Without containers, a computer can end up looking like a junk drawer. Applications have all sorts of dependencies.
- ❖ Containers keep things organized by isolating programs and their dependencies inside containers.

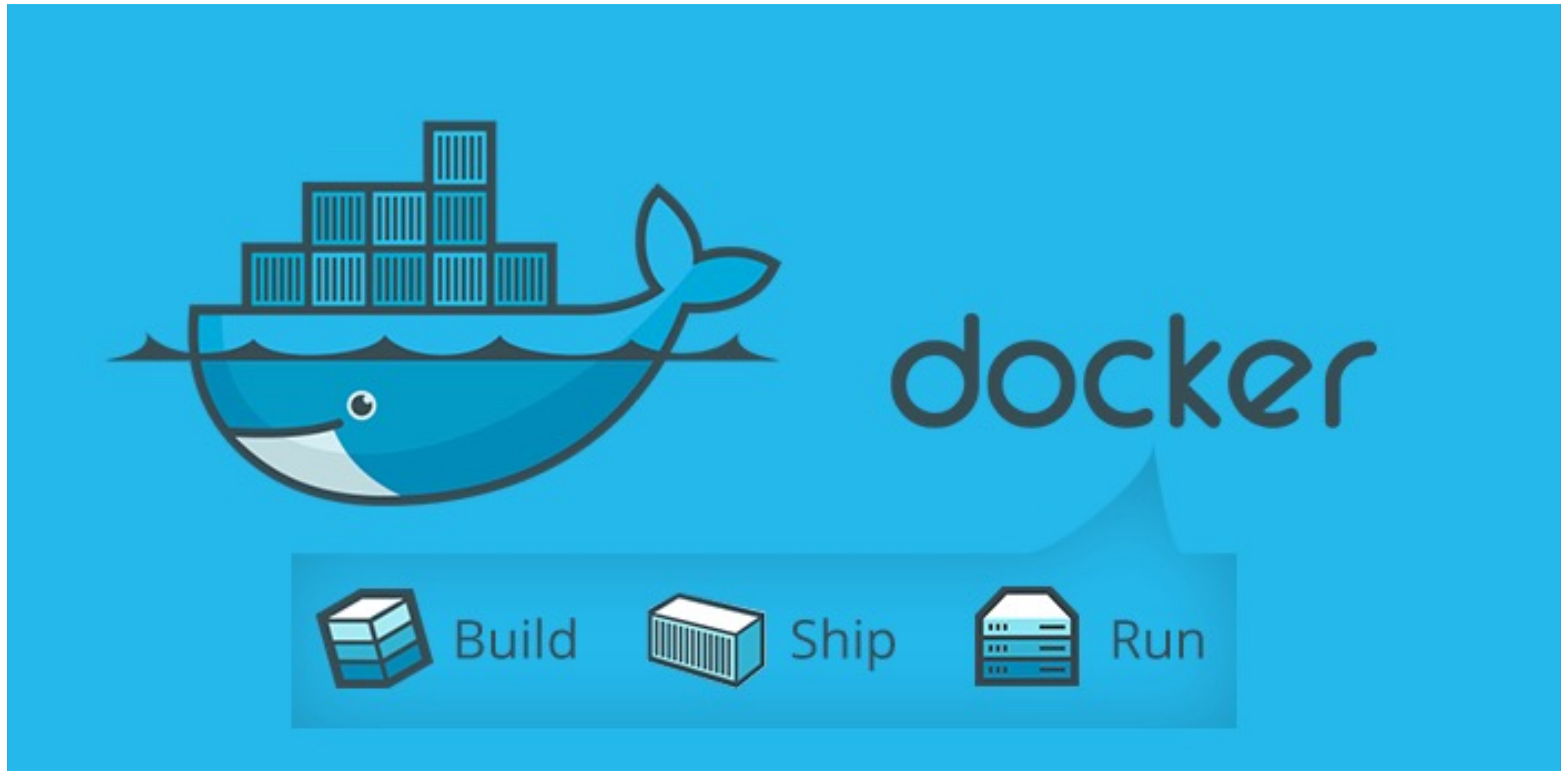


Source: Docker in Action by jeff nickoloff (2nd edition)

Why should we use containers?

Build once, run almost anywhere

An application’s dependencies typically include a specific operating system. Portability between operating systems is a major problem for software users. Although it’s possible to have compatibility between Linux software and macOS, using that same software on Windows can be more difficult.



Containers allow us to package up our complete software environment and ship it to numerous operating systems.

Source: https://medium.com/mindful-engineering/docker-setup-once-run-anywhere-3bb536fd7157

Why should we use containers?

Reproducibility

- ❖ Rising concerns about lack of repeatability, replicability and reproducibility in science and engineering.
- ❖ Not only “wet-lab” papers, but also “dry lab” papers, even published in high impact journals can seldom be repeated, due to variation in data collection methodologies, experimental environments, computational configuration, etc.
- ❖ **Containers can ensure identical versions of apps, libraries, compliers, etc.**

Docker

The concept of containers emerged in 1970s, but they were not well known until the emergence of Docker containers in 2013.

Docker is an open source platform for building, deploying, and managing containerized applications.

Some concerns about the security of Docker containers on HPC: Docker gives superuser privileges, but we do not want users to have full, unrestricted admin/ root access.

Singularity

- Singularity was developed in 2015 as an open-source project by researchers at Lawrence Berkeley National Laboratory led by Gregory Kurtzer.

- Singularity is emerging as the containerization framework of choice in HPC environments.
    1. Enable researchers to package entire scientific workflows, libraries, and even data.
    2. Users do not need to ask their system admin (e.g., RCAC) to install software for them.
    3. **Can use docker images.**
    4. Secure!
    5. **Does not require root privileges.**

Singularity basics

Detailed singularity user guide is available at: **sylabs.io/guides/3.8/user-guide**

The main singularity command

```
singularity [options] <subcommand> [subcommand options ...]
```

- ❖ Build
- ❖ Pull
- ❖ Shell
- ❖ Run
- ❖ Exec

Singularity workflow on HPC

1 (Optional). **Build** singularity containers on a computer system where you have root or sudo privilege, e.g., your personal computer with singularity installed.

2. **Pull** the public containers or **transfer** your own containers to HPC.

3. **Run** singularity containers on the HPC system.

Build your own containers by singularity

The first step is to install singularity on your personal computer.

We have singularity version **3.8.0** on the cluster. To guarantee compatibility, please be sure to follow the installation guide for version 3.8 on your system (https://sylabs.io/guides/3.8/user-guide/quick_start.html).

singularity build

1. Build from URI

```
singularity build myown_gatk.sif docker://broadinstitute/gatk:latest
```

2. Build using a singularity definition file

```
singularity build myown_container.sif definition.def
```

These two ways allow us to use containers that someone else created, or to create our own containers if we need to.

- Need to build using a computer with elevated privileges, then copy to cluster.
- If no access to such a computer, can also build in the cloud.

Remote builder

If you need to build an image from a system where you don't have admin privileges, we can build remotely using the Sylabs Remote Builder.

To remotely build an image using singularity, go through the following steps:

1. Go to: https://cloud.sylabs.io/, and generate a Sylabs account.
2. Create a new "Access Tokens", and copy it to clipboard.
3. SSH login to our clusters, and run `singularity remote login` in terminal and paste the access token at the prompt.
4. Then you can remotely build your own singularity image in the cluster.

```
singularity build -r myimage.sif myimage.def
```

or

```
singularity build --remote myimage.sif myimage.def
```

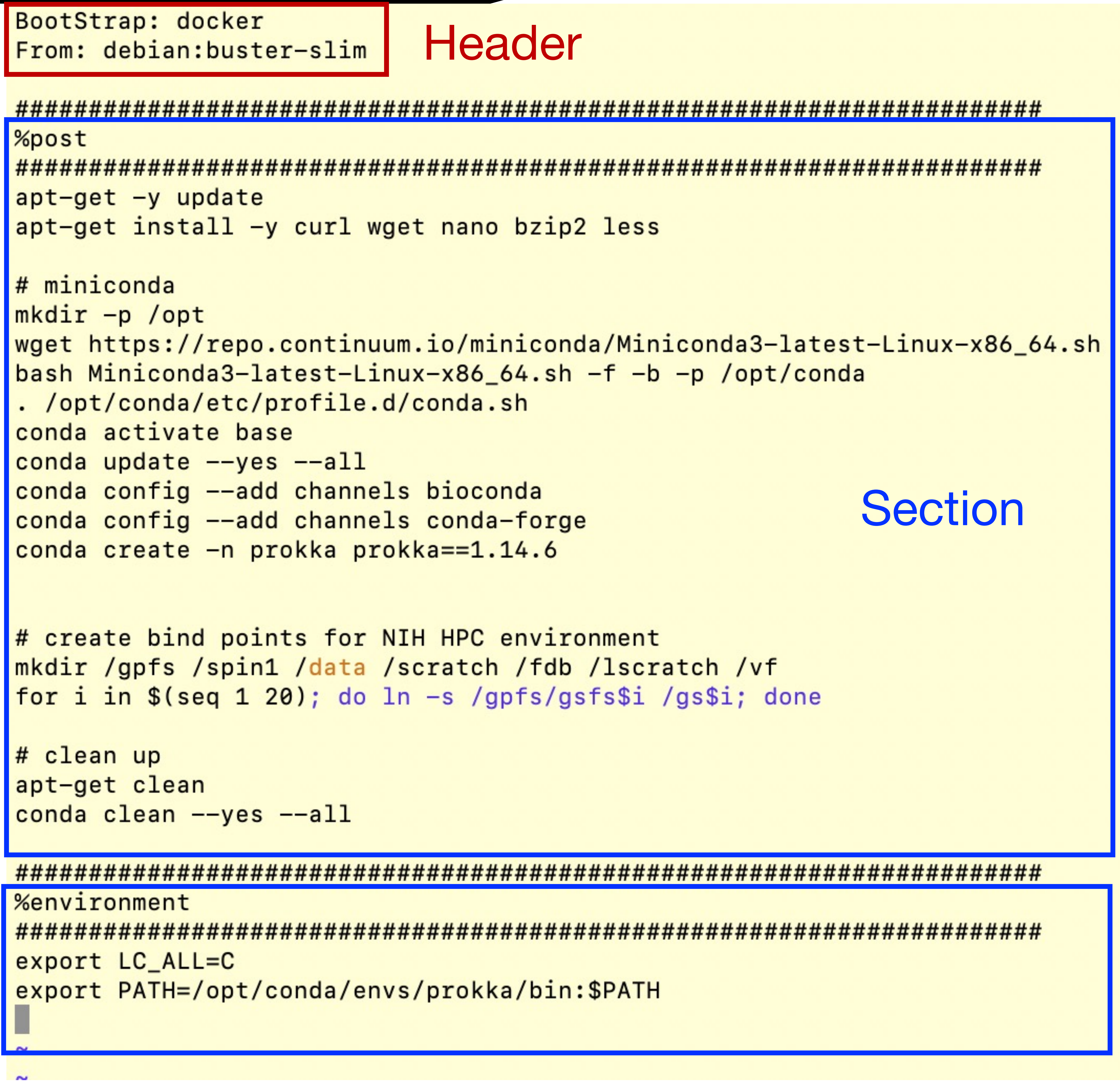
Once finished, the image will be downloaded automatically so that it's ready to use.

Singularity definition file

A **definition** file, or **def** file, is a recipe to build a container image with singularity. It is divided into two parts:

1. **Header**: the Header describes the core operating system to build within the container.
2. **Section**: each section is defined by a **%** character followed by the name of the particular section. Different sections add different content or execute commands at different times during the build process.

Detailed instruction on how to prepare a definition file is available at https://sylabs.io/guides/latest/user-guide/definition_files.html.



```
BootStrap: docker
From: debian:buster-slim

######################################################################
%post
######################################################################
apt-get -y update
apt-get install -y curl wget nano bzip2 less

# miniconda
mkdir -p /opt
wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh -f -b -p /opt/conda
. /opt/conda/etc/profile.d/conda.sh
conda activate base
conda update --yes --all
conda config --add channels bioconda
conda config --add channels conda-forge
conda create -n prokka prokka==1.14.6


# create bind points for NIH HPC environment
mkdir /gpfs /spin1 /data /scratch /fdb /lscratch /vf
for i in $(seq 1 20); do ln -s /gpfs/gsfs$i /gs$i; done

# clean up
apt-get clean
conda clean --yes --all

######################################################################
%environment
######################################################################
export LC_ALL=C
export PATH=/opt/conda/envs/prokka/bin:$PATH

~
~
```

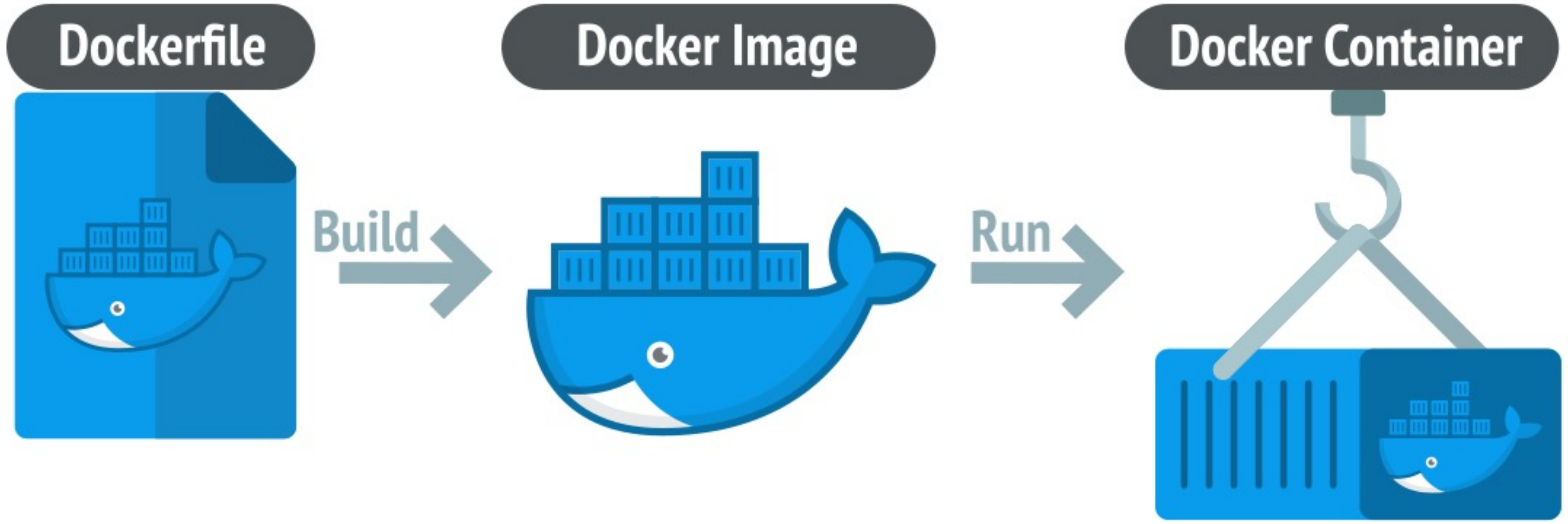
def file for prokka 1.14.6 prepared by NIH HPC staff

Build your own containers by docker

Using docker to build containers is another option:

1. Docker has a large, active, and stable ecosystems of container images.
2. **Singularity can use docker images.**

You can follow the installation guide for install Docker in your personal computer (https://docs.docker.com/get-docker/).



docker build

Build an image from the **Dockerfile** in the current directory and tag the image.

docker build -t NAME[:TAG] .

```
FROM ubuntu:bionic

RUN apt update

# need these packages to download and build samtools:
# https://github.com/samtools/samtools/blob/1.9/INSTALL
RUN apt install -y wget gcc libz-dev ncurses-dev libbz2-dev liblzma-dev \
    libcurl3-dev libcrypto++-dev make
RUN wget https://github.com/samtools/samtools/releases/download/1.9/samtools-1.9.tar.bz2 && \
    tar jxf samtools-1.9.tar.bz2 && \
    cd samtools-1.9 && ./configure && make install

CMD ["samtools"]
```

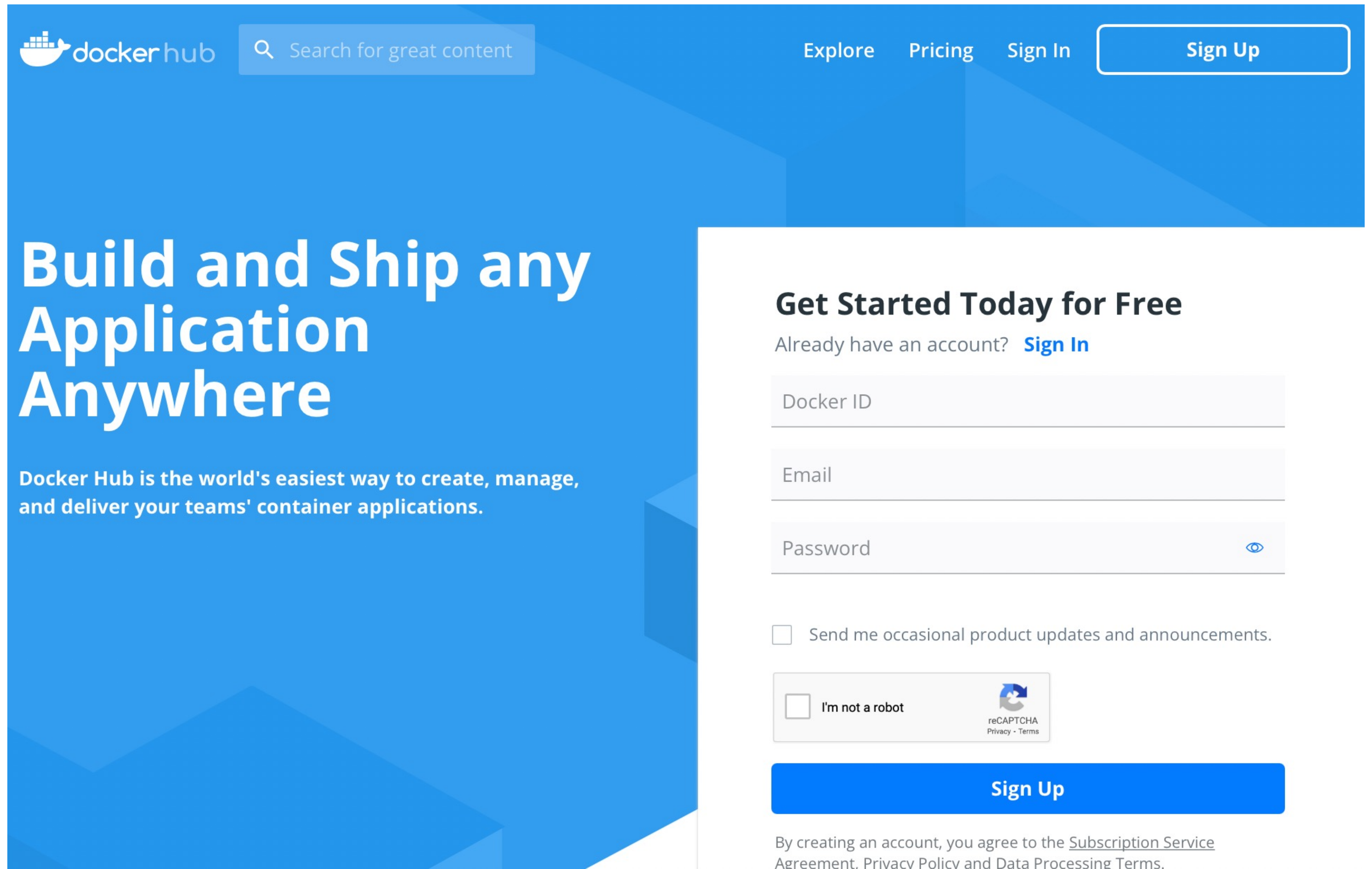
An example Dockerfile for building the container for the bioinformatics tool **samtools**.

Docker Hub

Docker Hub is a cloud-based public registry service to host both public and private images

Users get access to free public repositories for storing and sharing images or can choose a subscription plan for private repositories.

https://hub.docker.com



Build and Ship any Application Anywhere

Docker Hub is the world's easiest way to create, manage, and deliver your teams' container applications.

Get Started Today for Free

Already have an account? Sign In

☐ Send me occasional product updates and announcements.

☐ I'm not a robot

By creating an account, you agree to the Subscription Service Agreement, Privacy Policy and Data Processing Terms.

docker push

docker

To push an image to a Docker registry, go through the following steps:

1. Begin by tagging the image using the ***docker image tag*** command with the appropriate user or organization in the Docker Hub, as shown in the following code:

```
docker image tag myimage:tag username/myimage:tag
```

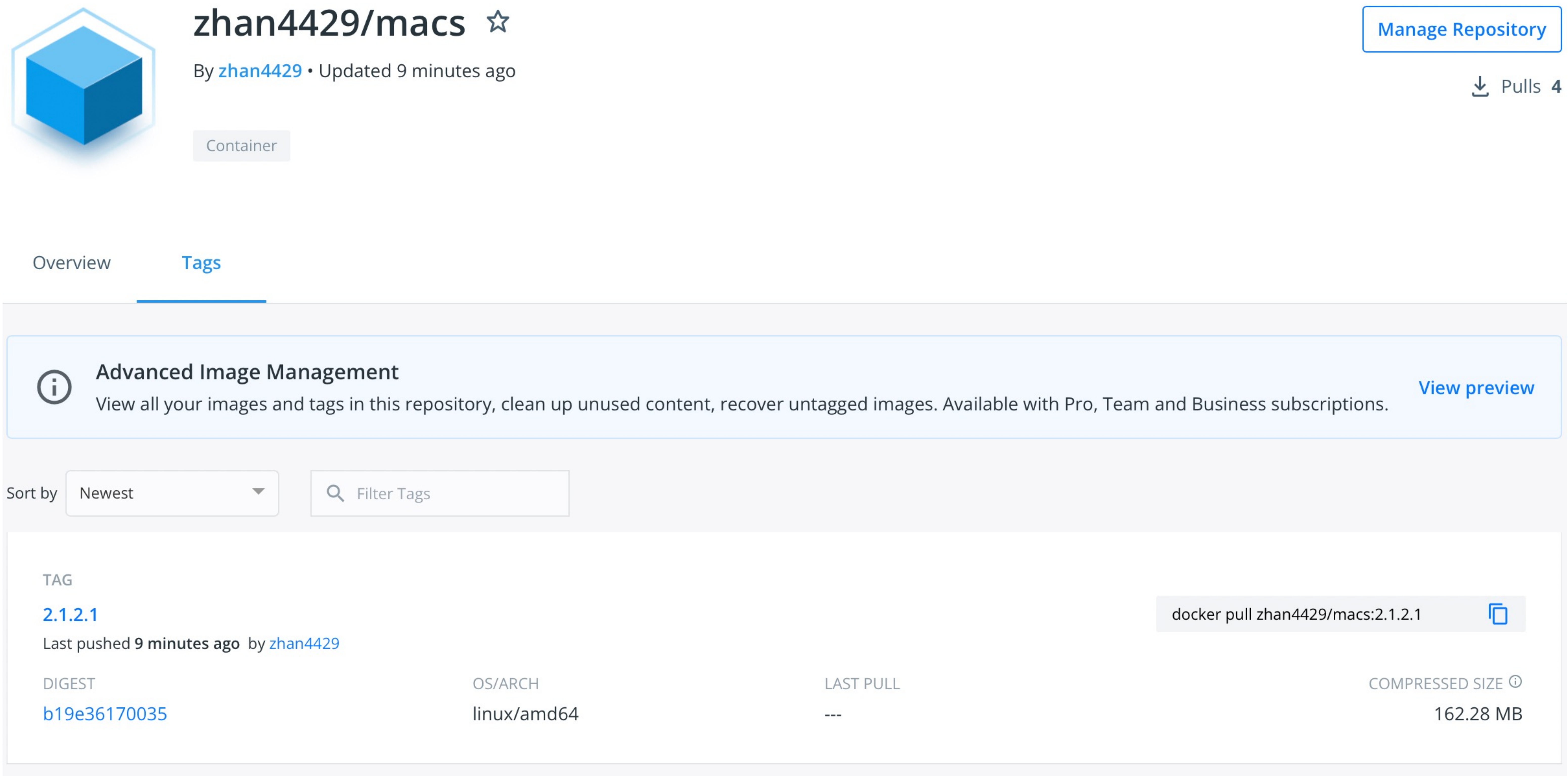
2. Logging into the Docker Hub registry

```
docker login
```

3. Push the image to Docker Hub

```
docker push username/myimage:tag
```

```
$ docker login
$ docker image tag macs:2.1.2.1  zhan4429/macs:2.1.2.1
$ docker push zhan4429/macs:2.1.2.1
```



zhan4429/macs ☆

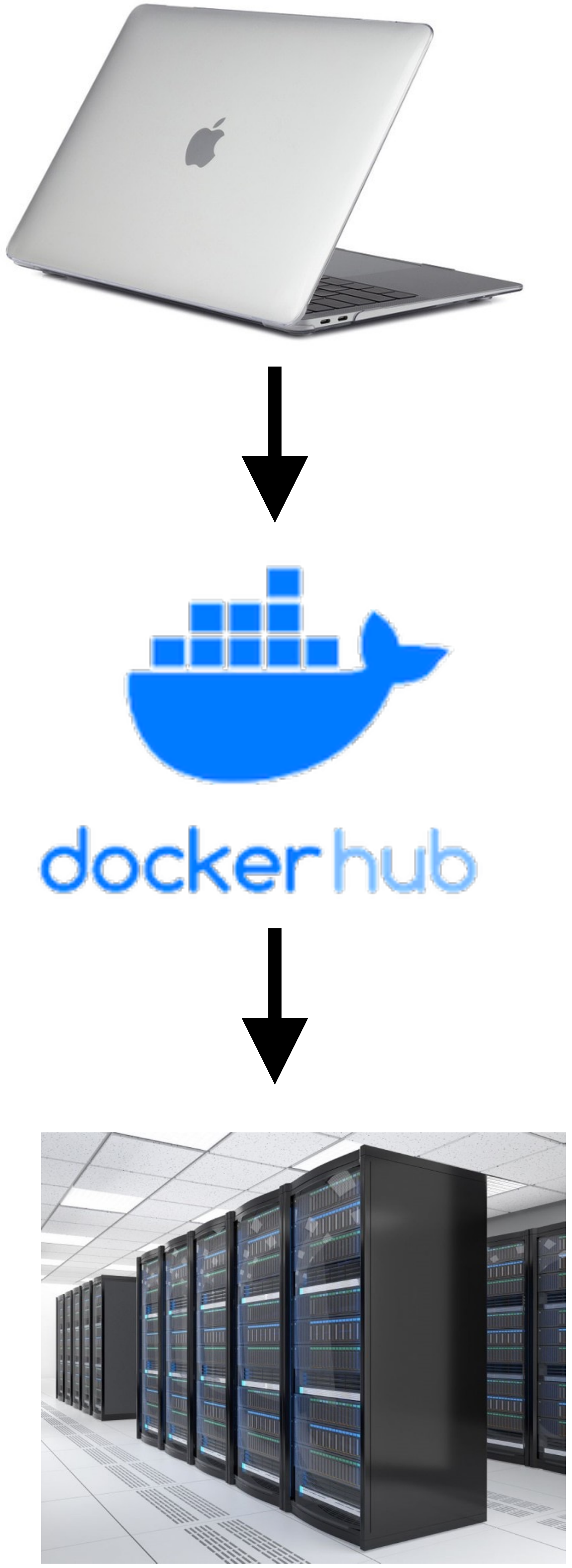
By zhan4429 • Updated 9 minutes ago

Container

Advanced Image Management

View all your images and tags in this repository, clean up unused content, recover untagged images. Available with Pro, Team and Business subscriptions.

TAG			
2.1.2.1 Last pushed **9 minutes ago** by zhan4429			docker pull zhan4429/macs:2.1.2.1
DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE
b19e36170035	linux/amd64	---	162.28 MB



```
$ singularity pull macs2.sif docker://zhan4429/macs:2.1.2.1
```

singularity pull

Download or build a container from a given URI.

singularity pull [output file] <URI>

Supported URIs include:

- **Library**: pull an image from singularity library
  library://<user>/<collection>/<image>[:tag]
- **Docker**: pull an image from Docker Hub.
  docker://<repository>/<image>[:tag]
- **Quay.io**: pull an image from Quay.io registry
  docker://quay.io/<repository>/<image>[:tag]
- **http, https**: pull an image using the http(s?) protocol
  e.g., https://library.sylabs.io/v1/imagefile/library/default/alpine:latest

Two useful image hubs

1. **DockerHub** (https://hub.docker.org)
   - Online repository of Docker container images.
   - As of Oct. 16, 2021, 8,435,580 available container images.

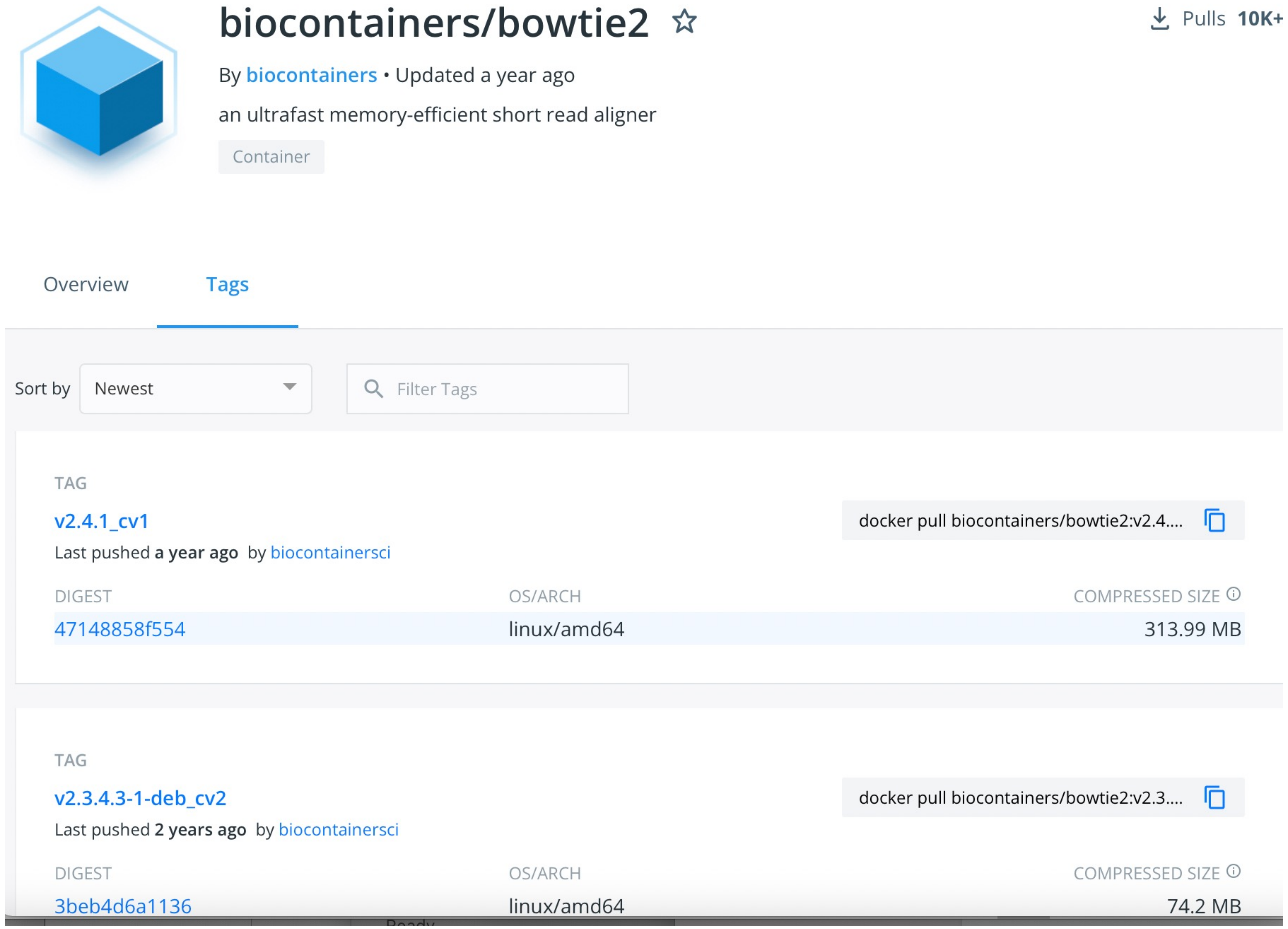
2. **Biocontainers** (https://biocontainers.pro/registry)
   - A community-driven project for bioinformatics containers.
   - 10.3K tools,41.1K versions,202.6Kcontainers and packages.
   - **For some reason, loading of the registry website is slow.**

singularity pull example

Pull the image from Docker Hub (docker://repository/image:tag)

For example:

singularity pull bowtie2_v2_4_1.sif docker://biocontainers/bowtie2:v2.4.1_cv1



biocontainers/bowtie2 ☆

Pulls 10K+

By biocontainers • Updated a year ago

an ultrafast memory-efficient short read aligner

TAG
v2.4.1_cv1
Last pushed **a year ago** by biocontainersci

docker pull biocontainers/bowtie2:v2.4....

DIGEST	OS/ARCH	COMPRESSED SIZE
47148858f554	linux/amd64	313.99 MB

TAG
v2.3.4.3-1-deb_cv2
Last pushed **2 years ago** by biocontainersci

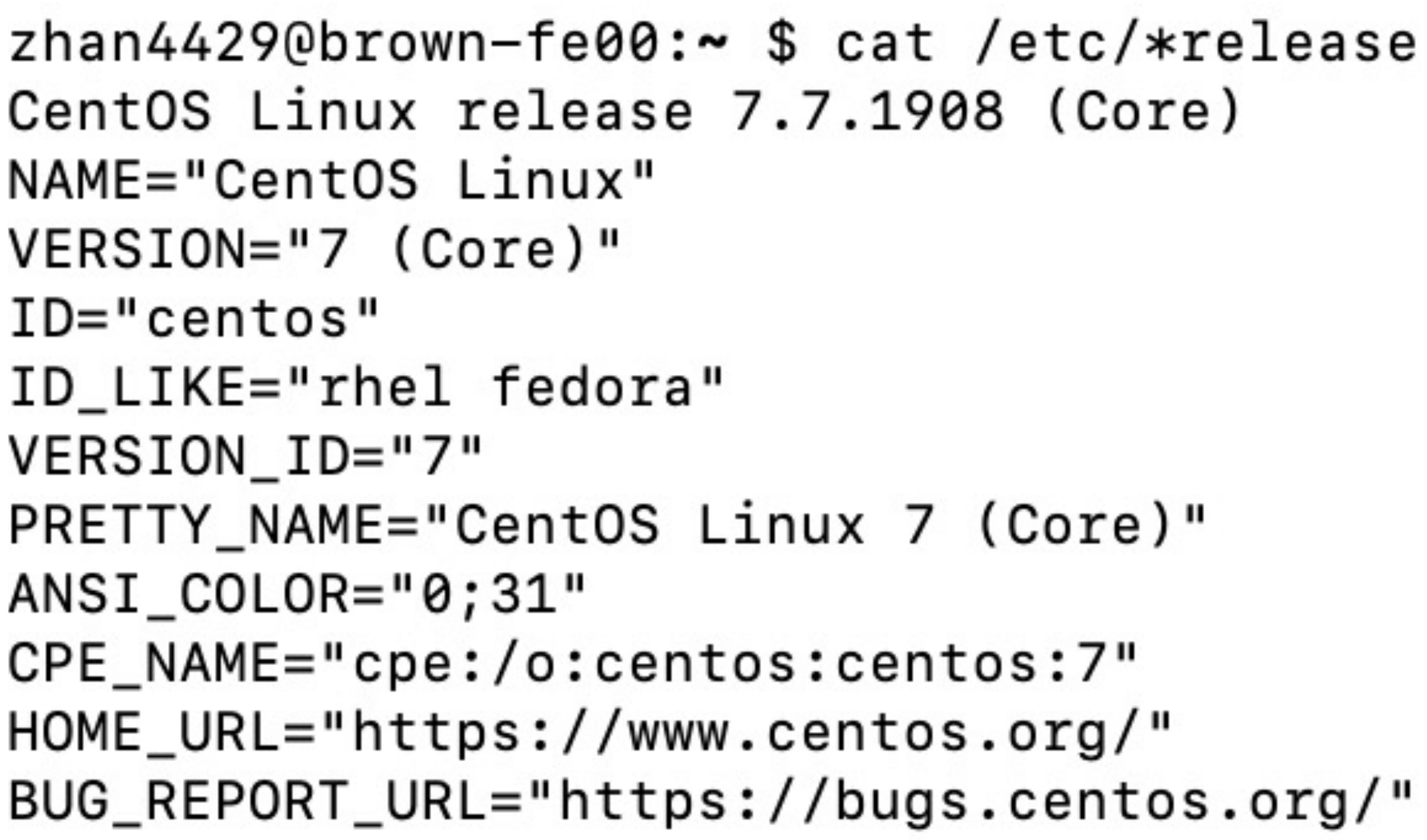
docker pull biocontainers/bowtie2:v2.3....

DIGEST	OS/ARCH	COMPRESSED SIZE
3beb4d6a1136	linux/amd64	74.2 MB

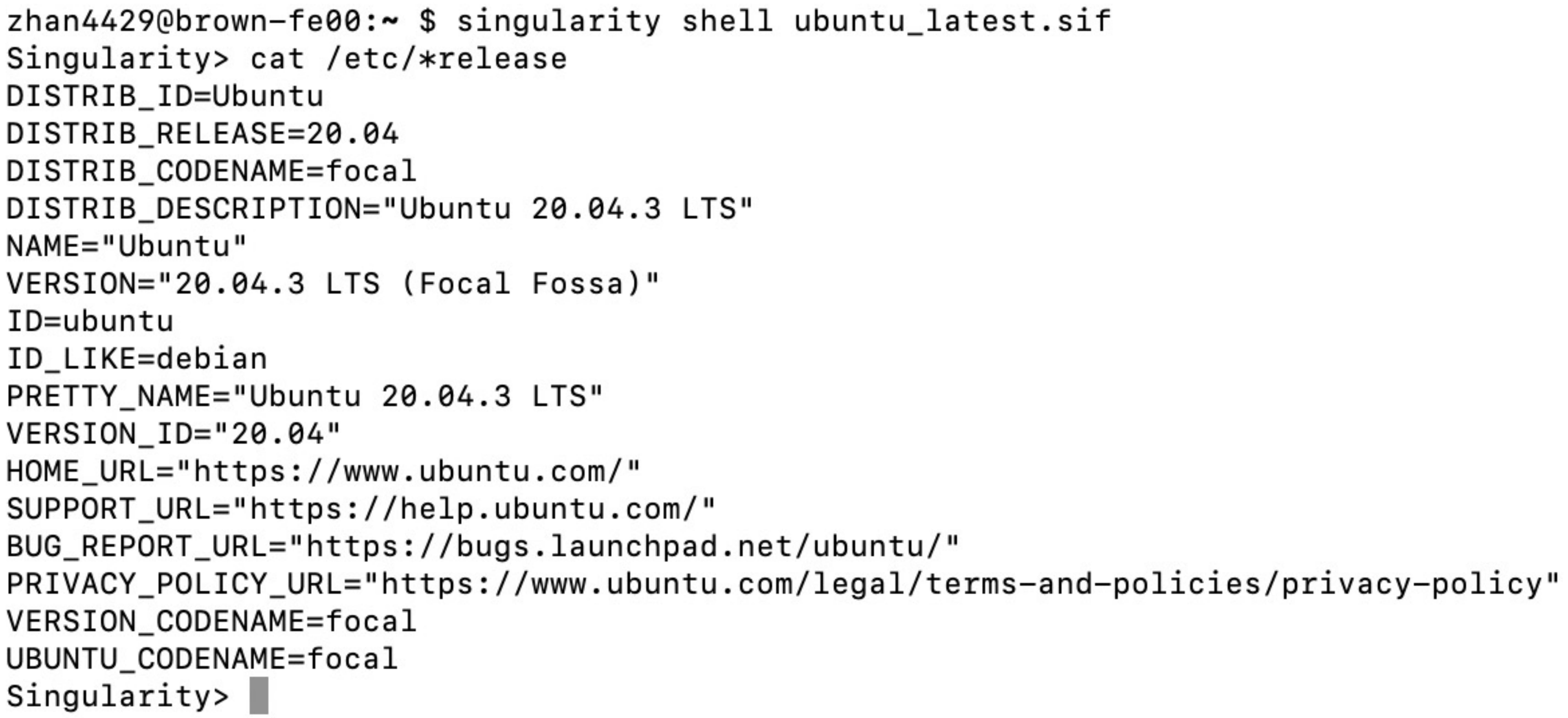
singularity shell

Start an interactive shell, and go inside the container

singularity shell myimage.sif



```
zhan4429@brown-fe00:~ $ cat /etc/*release
CentOS Linux release 7.7.1908 (Core)
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"
```



```
zhan4429@brown-fe00:~ $ singularity shell ubuntu_latest.sif
Singularity> cat /etc/*release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=20.04
DISTRIB_CODENAME=focal
DISTRIB_DESCRIPTION="Ubuntu 20.04.3 LTS"
NAME="Ubuntu"
VERSION="20.04.3 LTS (Focal Fossa)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 20.04.3 LTS"
VERSION_ID="20.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=focal
UBUNTU_CODENAME=focal
Singularity> █
```

Type “**exit**” in the interactive shell to go back to host system

Bind mounts

- Programs running inside a container will not have access to directories and files outside of your home and the current directory.
- Singularity allows you to map directories on your host system to directories within your container using bind mounts.

```
singularity shell --bind hostdir1:containerdir1 --bind hostdir2:containerdir2 myimage.sif
```

Singularity binds several directories into the container image automatically. **$HOME**, **/tmp** and **$PWD** is the default list.

We also configured singularity to bind **/apps**, **/depot**, and **/scratch** on our clusters.

singularity run

Run the user-defined default command(launch the **runscript**) within a container.

This will run the default command set for containers based on the specific image. This default command is set within the image ***runscript*** when the image is built. You do not specify a command to run when using ***singularity run***, you simply specify the image name. All arguments following the image name will be passed directly to the runscript.

```
singularity run myimage.sif [arguments]
```

To check the runscript for a container, enter:

```
singularity inspect --runscript myimage.sif
```

singularity exec

Run a command within a container

```
singularity exec myimage.sif command
```

For example:

```
singularity exec blast.2.11.0.sif blastx -query input.fasta -db swissprot -out blast.out
```

--bind option is also very useful for singularity exec

For example:

```
singularity exec --bind $HOME/data/:/data/ blast.2.11.0.sif blastx -query /data/input.fasta -db nr
```

input.fasta is located in the host directory /$HOME/data/

GPU acceleration

For many applications, CPU compute resources provide sufficient performance. However, for a certain class of applications, the massively parallel compute power offered by GPUs can speed up operations by orders of magnitude.

Run a container with GPU acceleration

For AMD GPUs:

```
singularity shell/run/exec --rocm myimage.sif [command] [argument]
```

For NVIDIA GPUs:

```
singularity shell/run/exec --nv myimage.sif [command] [argument]
```

Deployed container collections on RCAC clusters

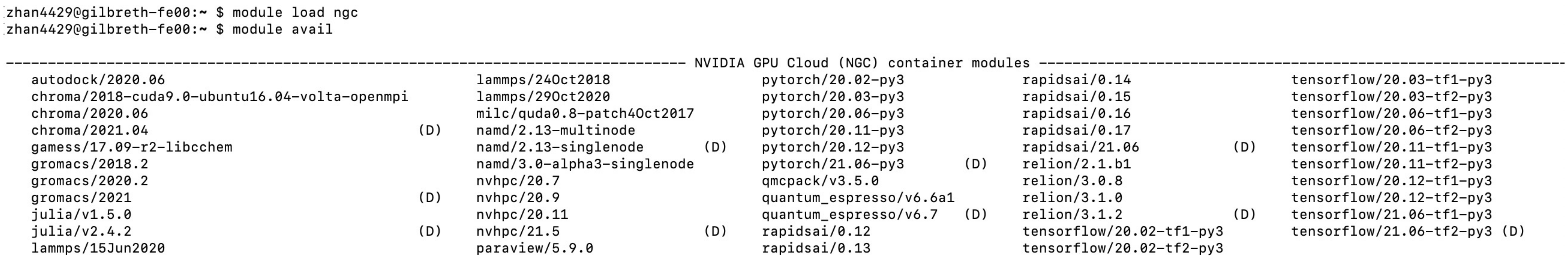
1. **NVIDIA NGC**: GPU-optimized tools for deep learning, machine learning, and high-performance computing.

2. **Biocontainers**: frequently used bioinformatic tools.

3. **ROCm**: AMD GPU software containers for HPC, AI & machine learning.

We already wrapped these containers into convenient software modules. These modules wrap underlying complexity and provide the same commands that are expected from non-containerized versions of each application.

NGC



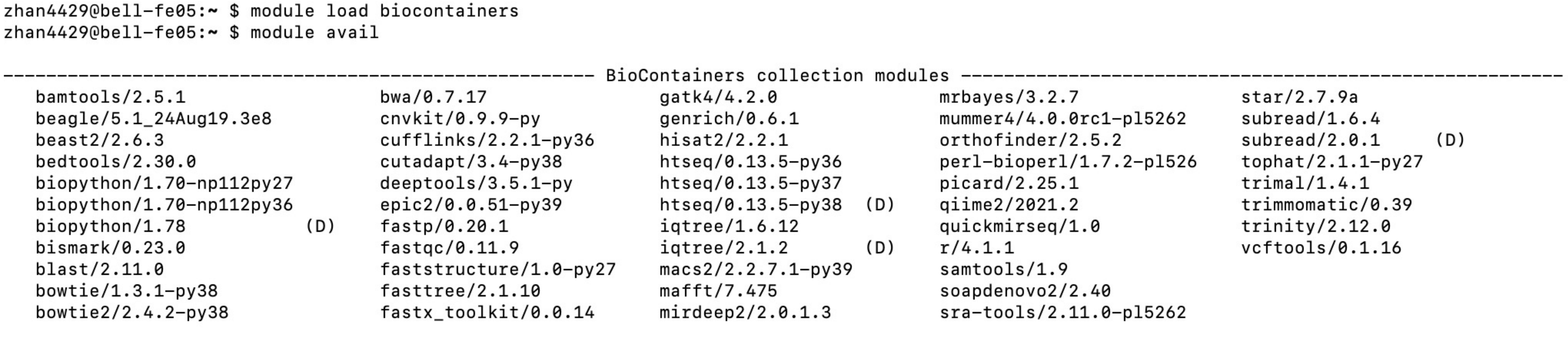
```
zhan4429@gilbreth-fe00:~ $ module load ngc
zhan4429@gilbreth-fe00:~ $ module avail

--------------------------------------------------------------------------------- NVIDIA GPU Cloud (NGC) container modules ---------------------------------------------------------------
   autodock/2020.06                                     lammps/24Oct2018                  pytorch/20.02-py3              rapidsai/0.14                   tensorflow/20.03-tf1-py3
   chroma/2018-cuda9.0-ubuntu16.04-volta-openmpi        lammps/29Oct2020                  pytorch/20.03-py3              rapidsai/0.15                   tensorflow/20.03-tf2-py3
   chroma/2020.06                                       milc/quda0.8-patch4Oct2017        pytorch/20.06-py3              rapidsai/0.16                   tensorflow/20.06-tf1-py3
   chroma/2021.04                                (D)    namd/2.13-multinode               pytorch/20.11-py3              rapidsai/0.17                   tensorflow/20.06-tf2-py3
   gamess/17.09-r2-libcchem                             namd/2.13-singlenode       (D)    pytorch/20.12-py3              rapidsai/21.06           (D)    tensorflow/20.11-tf1-py3
   gromacs/2018.2                                       namd/3.0-alpha3-singlenode        pytorch/21.06-py3       (D)    relion/2.1.b1                   tensorflow/20.11-tf2-py3
   gromacs/2020.2                                       nvhpc/20.7                        qmcpack/v3.5.0                 relion/3.0.8                    tensorflow/20.12-tf1-py3
   gromacs/2021                                  (D)    nvhpc/20.9                        quantum_espresso/v6.6a1        relion/3.1.0                    tensorflow/20.12-tf2-py3
   julia/v1.5.0                                         nvhpc/20.11                       quantum_espresso/v6.7   (D)    relion/3.1.2             (D)    tensorflow/21.06-tf1-py3
   julia/v2.4.2                                  (D)    nvhpc/21.5                 (D)    rapidsai/0.12                  tensorflow/20.02-tf1-py3        tensorflow/21.06-tf2-py3 (D)
   lammps/15Jun2020                                     paraview/5.9.0                    rapidsai/0.13                  tensorflow/20.02-tf2-py3
```

Deployed in Gilbreth, Scholar and Anvil.

For example:

```
$ module load ngc
$ module load relion/2.1.b1
$ relion
```

Biocontainers



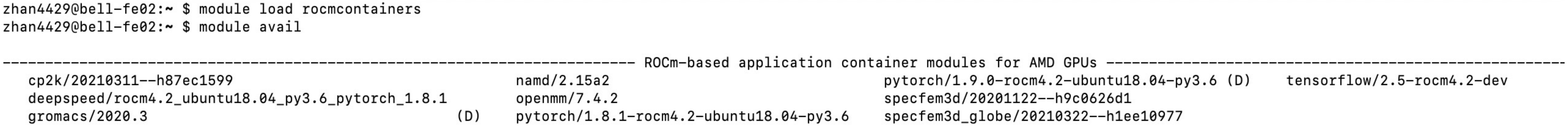
```
zhan4429@bell-fe05:~ $ module load biocontainers
zhan4429@bell-fe05:~ $ module avail

------------------------------------------------------- BioContainers collection modules --------------------------------------------------------
   bamtools/2.5.1                  bwa/0.7.17                gatk4/4.2.0               mrbayes/3.2.7               star/2.7.9a
   beagle/5.1_24Aug19.3e8          cnvkit/0.9.9-py           genrich/0.6.1             mummer4/4.0.0rc1-pl5262     subread/1.6.4
   beast2/2.6.3                    cufflinks/2.2.1-py36      hisat2/2.2.1              orthofinder/2.5.2           subread/2.0.1     (D)
   bedtools/2.30.0                 cutadapt/3.4-py38         htseq/0.13.5-py36         perl-bioperl/1.7.2-pl526    tophat/2.1.1-py27
   biopython/1.70-np112py27        deeptools/3.5.1-py        htseq/0.13.5-py37         picard/2.25.1               trimal/1.4.1
   biopython/1.70-np112py36        epic2/0.0.51-py39         htseq/0.13.5-py38  (D)    qiime2/2021.2               trimmomatic/0.39
   biopython/1.78           (D)    fastp/0.20.1              iqtree/1.6.12             quickmirseq/1.0             trinity/2.12.0
   bismark/0.23.0                  fastqc/0.11.9             iqtree/2.1.2       (D)    r/4.1.1                     vcftools/0.1.16
   blast/2.11.0                    faststructure/1.0-py27    macs2/2.2.7.1-py39        samtools/1.9
   bowtie/1.3.1-py38               fasttree/2.1.10           mafft/7.475               soapdenovo2/2.40
   bowtie2/2.4.2-py38              fastx_toolkit/0.0.14      mirdeep2/2.0.1.3          sra-tools/2.11.0-pl5262
```

Deployed in all clusters except Gilbreth. We will keep deploying new tools or versions into this collection.

For example:

```
$ module load biocontainers
$ module load bamtools/2.5.1
$ bamtools -h
```

ROCm



```
zhan4429@bell-fe02:~ $ module load rocmcontainers
zhan4429@bell-fe02:~ $ module avail

-------------------------------------------------------------------------- ROCm-based application container modules for AMD GPUs -------------------------------------------------------
   cp2k/20210311--h87ec1599                                 namd/2.15a2                                pytorch/1.9.0-rocm4.2-ubuntu18.04-py3.6 (D)    tensorflow/2.5-rocm4.2-dev
   deepspeed/rocm4.2_ubuntu18.04_py3.6_pytorch_1.8.1        openmm/7.4.2                               specfem3d/20201122--h9c0626d1
   gromacs/2020.3                                    (D)    pytorch/1.8.1-rocm4.2-ubuntu18.04-py3.6    specfem3d_globe/20210322--h1ee10977
```

Deployed only in Bell.

For example:

`$ module load rocmcontainers`

`$ module load pytorch/1.8.1-rocm4.2-ubuntu18.04-py3.6`

Get help

User guide

Singularity contains instructions for using Singularity on RCAC clusters.

Biocontainer collection contains instructions and examples for running bioinformatic containers.

ROCm container collection contains instructions and examples for running AMD GPU containers.

NGC container collection contains instructions and examples for running NVIDIA GPU containers.

Email

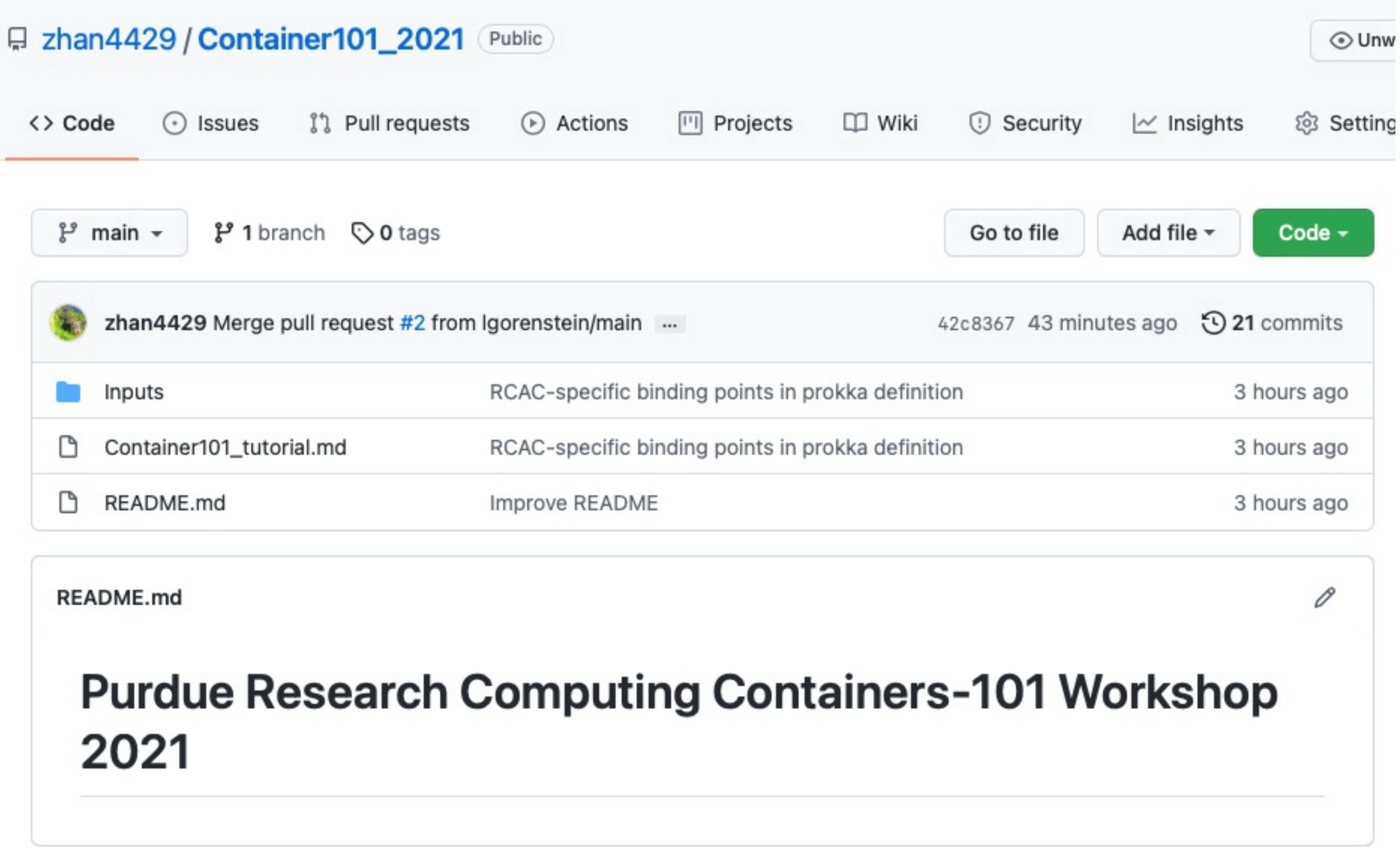
rcac-help@purdue.edu is our email support address. Send us an email any time.

Coffee hour consultations

In response to COVID-19, we are temporarily switching all our Coffee Hour Consultations to online only. We offer several slots (2:00 to 3:30pm) each afternoon (Monday to Thursday) for private one-on-one consultations or questions of up to 30 minutes.

Practice

https://github.com/zhan4429/Container101_2021



zhan4429 / Container101_2021 Public

zhan4429 Merge pull request #2 from lgorenstein/main … 42c8367 43 minutes ago 21 commits

Inputs	RCAC-specific binding points in prokka definition	3 hours ago
Container101_tutorial.md	RCAC-specific binding points in prokka definition	3 hours ago
README.md	Improve README	3 hours ago

README.md

Purdue Research Computing Containers-101 Workshop 2021

Thank you!

Questions?