Singularity containers for Bioinformatics workflows

Dr. Tannistha Nandi

Data Scientist

Research Computing Services, University of Calgary

Email: tannistha.nandi@computecanada.ca

Goals for today

I. How does High Performance Computing (HPC) environment work?

   a. Containerization of Bioinformatics tools

II. How to get Containers for Bioinformatics tools?

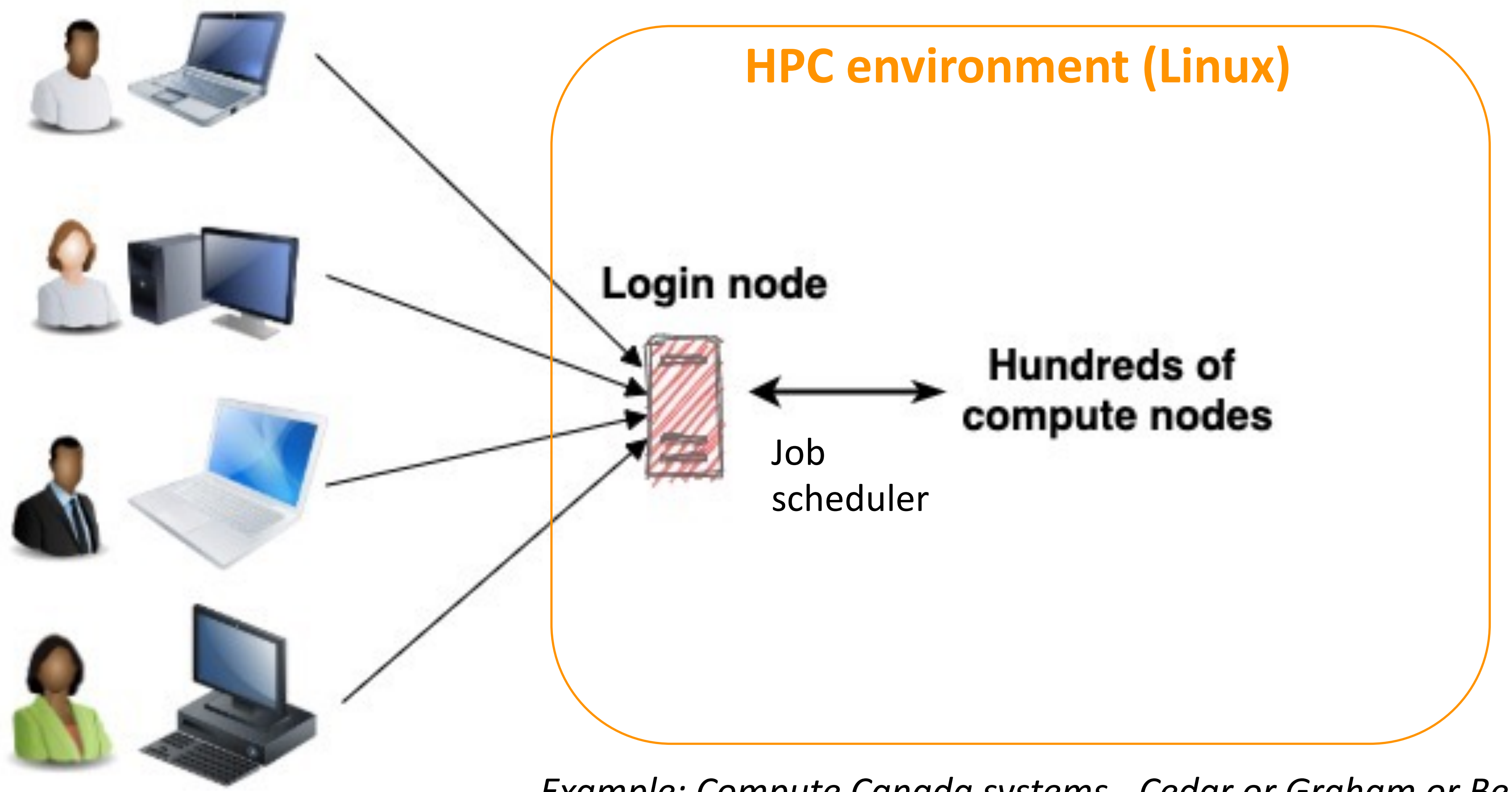
   a. Pre-built containers

   b. Customized containers

III. How to run containers on a job scheduler

How to get to a HPC environment?

Users on personal devices



HPC environment (Linux)

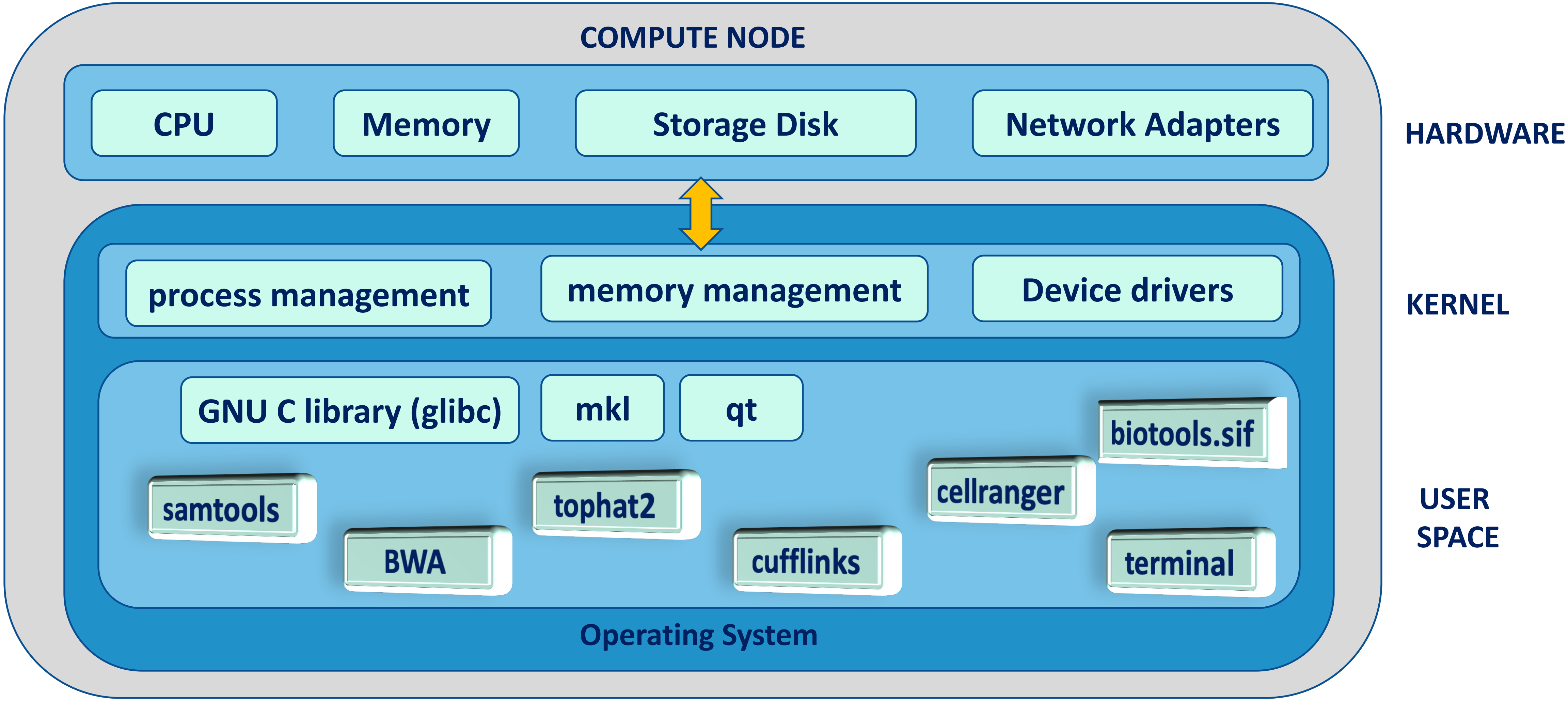
Login node

Job scheduler

Hundreds of compute nodes

Example: Compute Canada systems - Cedar or Graham or Beluga
University’s local HPC infrastructure

Bioinformatics tools in a HPC environment



COMPUTE NODE

CPU

Memory

Storage Disk

Network Adapters

HARDWARE

process management

memory management

Device drivers

KERNEL

GNU C library (glibc)

mkl

qt

biotools.sif

samtools

tophat2

cellranger

BWA

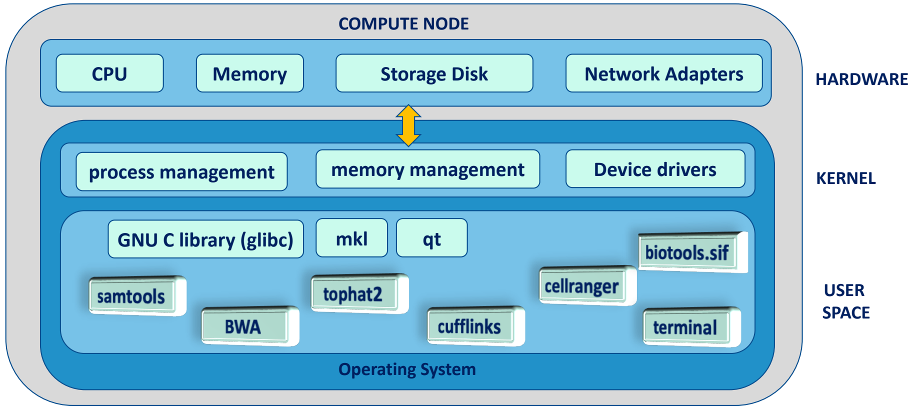
cufflinks

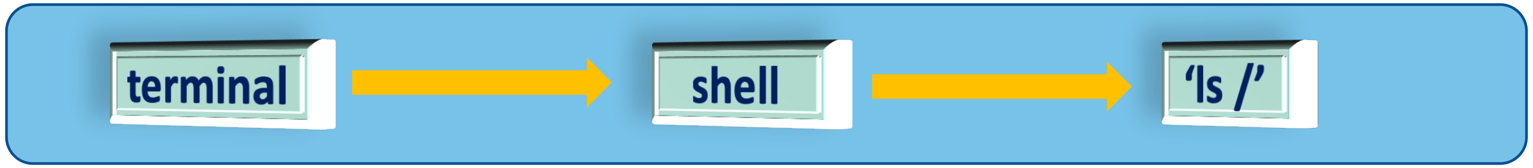
terminal

USER SPACE

Operating System

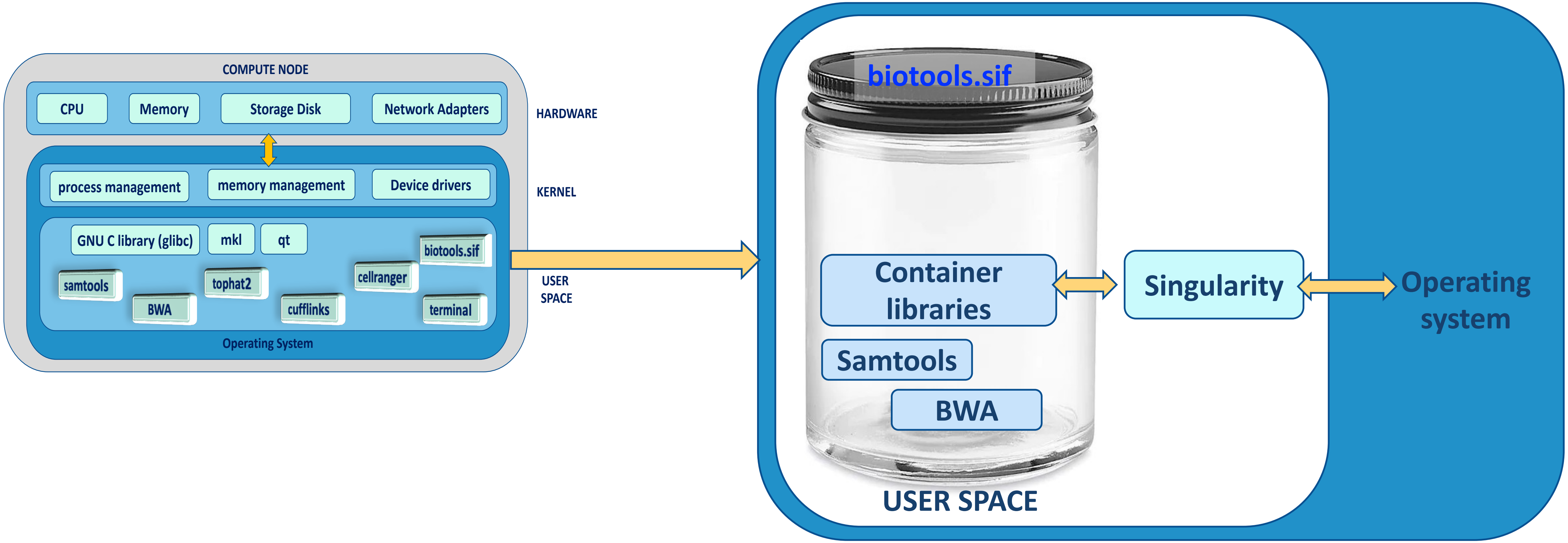
Bioinformatics tools in a HPC environment





```
[nandit@cedar1 ~]$ ls /
bin                            localscratch    project
boot                           media           root
cvmfs                          misc            run
data                           mnt             sbin
dev                            nearline        scratch
etc                            net             srv
home                           opt             sys
```

Bioinformatics tools in a HPC environment



COMPUTE NODE

CPU
Memory
Storage Disk
Network Adapters

HARDWARE

process management
memory management
Device drivers

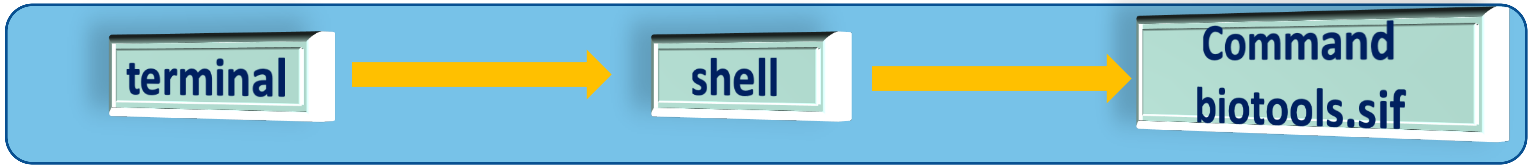
KERNEL

GNU C library (glibc)
mkl
qt
biotools.sif
samtools
tophat2
cellranger
BWA
cufflinks
terminal

USER SPACE

Operating System

biotools.sif
Container libraries
Samtools
BWA
Singularity
Operating system
USER SPACE

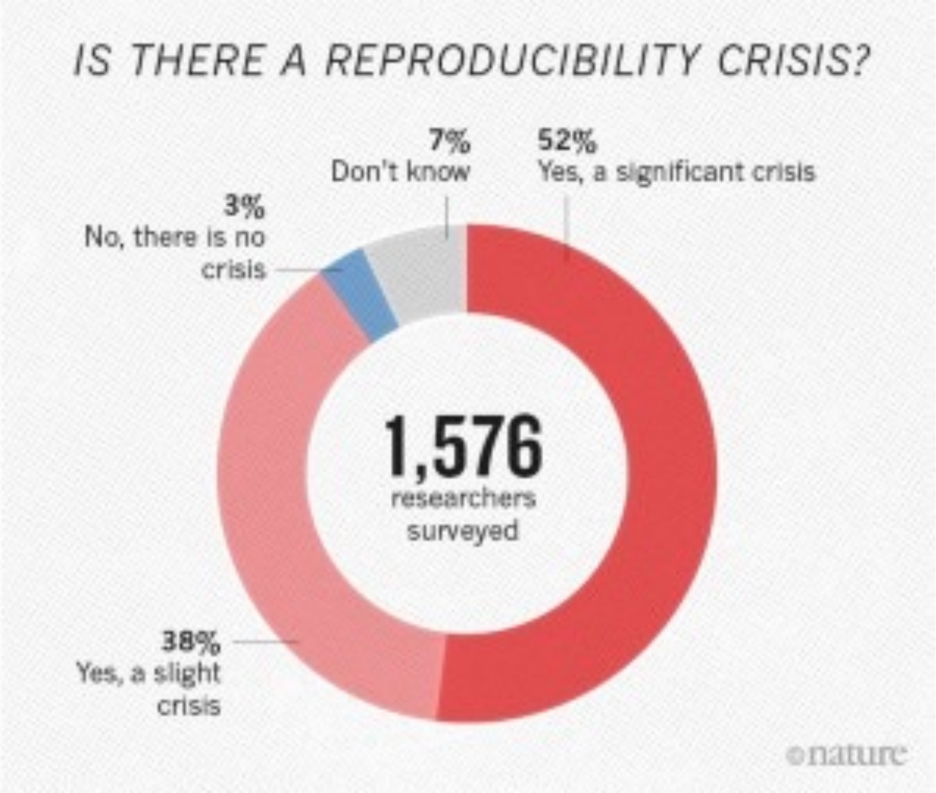


terminal
shell
Command biotools.sif

Containerization of Bioinformatics tools

Reliable way to capture and distribute software and its compute environments

- Reproducibility
- Portability



Use Case

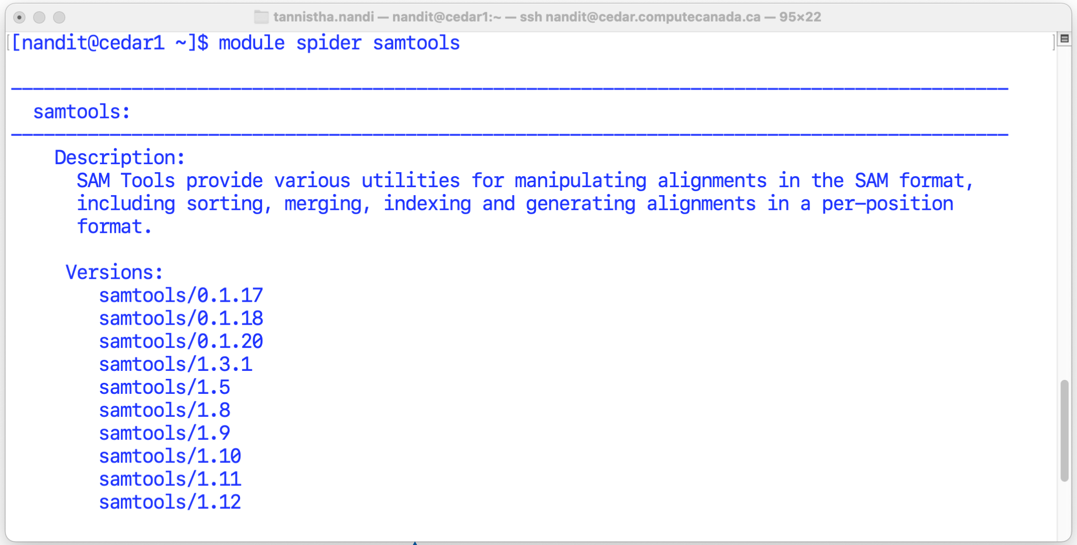
1. Reproduce results from a published journal article
2. Transfer pipelines from a test environment to a production environment

*Reference: Baker, M. Nature **533,** 452–454 (2016)*

What problems do Containers solve?

Compute Canada systems: the supporting software environment. For example, samtools

CEDAR



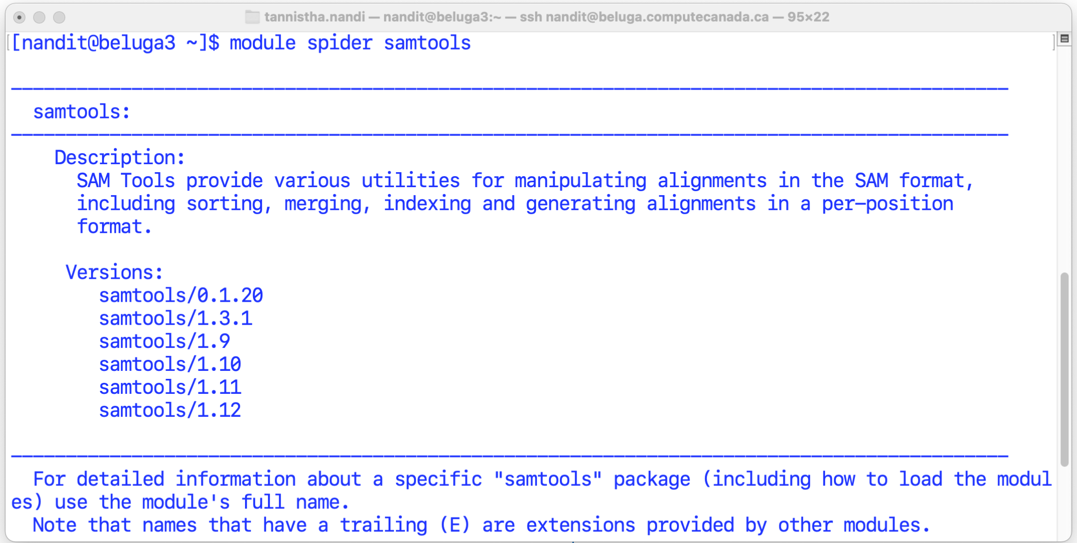
tannistha.nandi — nandit@cedar1:~ — ssh nandit@cedar.computecanada.ca — 95×22

```
[nandit@cedar1 ~]$ module spider samtools

_____________________________________________________________________________________________
  samtools:
_____________________________________________________________________________________________
    Description:
      SAM Tools provide various utilities for manipulating alignments in the SAM format,
      including sorting, merging, indexing and generating alignments in a per-position
      format.

     Versions:
        samtools/0.1.17
        samtools/0.1.18
        samtools/0.1.20
        samtools/1.3.1
        samtools/1.5
        samtools/1.8
        samtools/1.9
        samtools/1.10
        samtools/1.11
        samtools/1.12
```

BELUGA



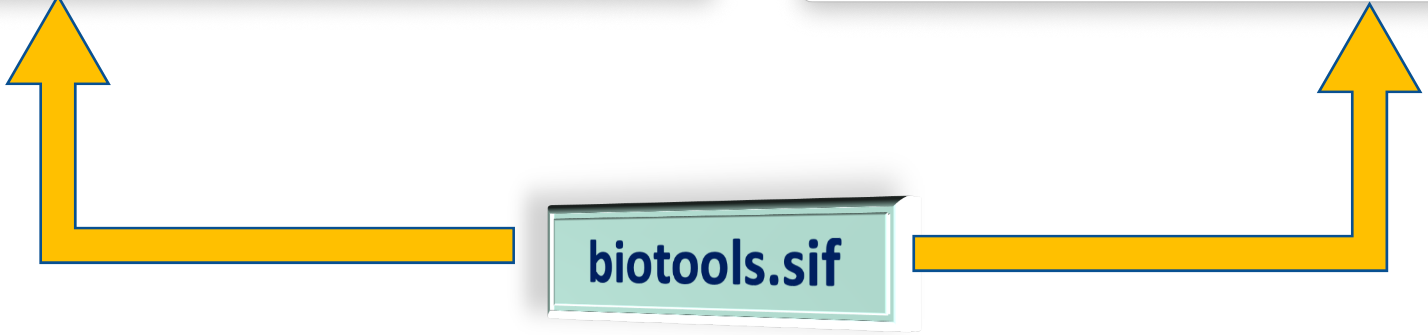
tannistha.nandi — nandit@beluga3:~ — ssh nandit@beluga.computecanada.ca — 95×22

```
[nandit@beluga3 ~]$ module spider samtools

_____________________________________________________________________________________________
  samtools:
_____________________________________________________________________________________________
    Description:
      SAM Tools provide various utilities for manipulating alignments in the SAM format,
      including sorting, merging, indexing and generating alignments in a per-position
      format.

     Versions:
        samtools/0.1.20
        samtools/1.3.1
        samtools/1.9
        samtools/1.10
        samtools/1.11
        samtools/1.12

_____________________________________________________________________________________________
  For detailed information about a specific "samtools" package (including how to load the modul
es) use the module's full name.
  Note that names that have a trailing (E) are extensions provided by other modules.
```



Accessing containers

1. Prebuilt containers

Available public / private repositories or from another researcher

- Singularity Container Library: https://cloud.sylabs.io/library
- Docker Hub: https://hub.docker.com/
- NGC: https://ngc.nvidia.com/catalog
- Biocontainers: https://biocontainers.pro/
- Research collaborator

For example:

```
$ singularity pull py39.sif docker://python:3.9
```

Custom name of the image

Registry name

Accessing containers

2. Customized Container

a. Access to a Local Linux machine with elevated ‘sudo’ privileges.

- use a device with Linux OS, install Singularity on it.
- use MacBook or Windows laptop/desktop, install a VM to provide Linux OS, then install Singularity in the VM.

b. A recipe file for the singularity container (extension .def)

```
[user@local ~]$ ls
 container.def
```

Accessing containers

2. Customized Containers – Example 1: Singularity recipe file

```
[nandit@local ~]$ cat container.def
```

```
BootStrap: docker
From: centos:latest
```

HEADER

```
%labels
   Tannistha Nandi
%runscript
    echo "Hello from inside the container..."
```

SECTIONS

- defined by a % character followed by the name of the section.

- All sections are optional.

Accessing containers

2. Customized Containers – Example 1: Singularity recipe file

```
[nandit@local ~]$ sudo singularity  build container.sif container.def
         [sudo] password for tannistha.nandi:
         INFO:    Starting build...
         Getting image source signatures
         Copying blob a1d0c7532777 done
         Copying config 8c1402b22a done
         Writing manifest to image destination
         Storing signatures
         2021/10/10 08:21:13  info unpack layer: sha256:a1d0c75327776413fa0db9ed3adcdbadedc95a662eb1d360dad82bb913f8a1d1
         INFO:    Adding labels
         INFO:    Adding runscript
         INFO:    Creating SIF file...
         INFO:    Build complete: container.sif

[nandit@local ~]$ ls
 container.def container.sif

[nandit@local ~]$./container.sif
         Hello from inside the container...
```

Accessing containers

2. Customized Containers – Example 1: Singularity recipe file

```
[nandit@local ~]$singularity inspect container.sif
    Tannistha: Nandi
    org.label-schema.build-arch: amd64
    org.label-schema.build-date: Sunday_10_October_2021_8:21:19_MDT
    org.label-schema.schema-version: 1.0
    org.label-schema.usage.singularity.deffile.bootstrap: docker
    org.label-schema.usage.singularity.deffile.from: centos:8
    org.label-schema.usage.singularity.version: 3.8.0-1.el8
```

Customized containers

2. Customized Containers – Example 2: Singularity recipe for a container to run a python script

```
[nandit@local ~]$ ls
 python.def myscript.py
[nandit@local ~]$ cat python.def

BootStrap: docker
From: python:3.9
%labels
   Tannistha Nandi
%files
    /path/to/myscript.py /opt/
%post
  apt-get -y update
  apt-get -y install pip wget git
%runscript
    python myscript.py
```

Accessing containers

2. Customized Containers – Example 2: Singularity recipe for a container to run a python script

a. Explore the environment within Singularity shell

```
[nandit@local ~]$ singularity shell python.sif
Singularity>
Singularity> ls /opt
myscript.py
Singularity> exit
```

b. Execute the python program

```
[nandit@local ~]$singularity exec python.sif python myscript.py
This is my first python script. I am happy!
[nandit@local ~]$
```

c. Run the default “runscript” in the container

```
[nandit@local ~]$ singularity run python.sif
This is my first python script. I am happy!
```

Accessing containers

2. Customized Containers – Example 3: Singularity recipe for ‘samtools’ container

%runscript: Define commands that will be executed by singularity run.

%post: Execute commands after the base OS has been installed

```
BootStrap: docker
From: centos:8

%labels
    Tannistha Nandi
%runscript
    exec “@”
%setup
    mkdir -p ${SINGULARITY_ROOTFS}/opt
    mkdir -p ${SINGULARITY_ROOTFS}/scratch
    mkdir -p ${SINGULARITY_ROOTFS}/shared
%post
   dnf -y makecache
   dnf -y group install "Development Tools"
   dnf -y install --allowerasing  hostname which dnf-utils git zlib zlib-devel \
         bzip2 bzip2-devel xz xz-devel libcurl libcurl-devel ncurses ncurses-devel \
         unzip wget gnuplot rsync java-1.8.0-openjdk java-1.8.0-openjdk-devel \
         openssl-devel libffi-devel
%environment
   export PATH=$PATH:/usr/bin:/usr/local/bin/
   LANG="en_US.UTF-8”
```

Accessing containers

2. Customized Containers – Example 3: Singularity recipe for ‘samtools’ container

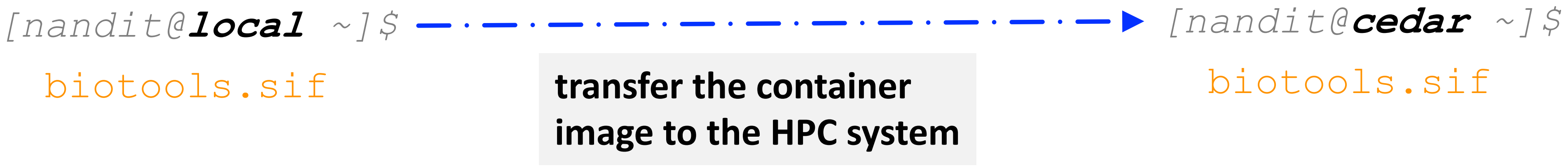
%app sections - build a single container with two or three different programs*

```
%apprun samtools
    exec samtools
%appinstall samtools
   git clone https://github.com/samtools/htslib
   cd htslib
   autoheader
   autoconf
   git submodule update --init --recursive
   ./configure
   make -j 8
   make install
   cd ..
   git clone https://github.com/samtools/samtools
   cd samtools
   autoheader
   autoconf -Wno-syntax
   ./configure
   make -j 8
   make install
   cd ..
```

%apprun: runscript for the app

%appinstall: commands to install the app (similar to *%post* but only for one app here).

Run containers on a job scheduler



[nandit@**local** ~]$

biotools.sif

transfer the container image to the HPC system

[nandit@**cedar** ~]$

biotools.sif

Container image is a file with .sif or .simg extension that contains everything needed to run applications.

Run the container without elevated privilege on the host HPC system like cedar.

Use containers

1. native command

```
$./biotools.sif
```

2. subcommand ‘shell’

```
$singularity shell biotools.sif
```

3. subcommand ‘exec’

```
$singularity exec biotools.sif samtools
```

4. subcommand ‘run’

```
$ singularity run biotools.sif
```

Run containers on a job scheduler

1. Get an interactive session

```
[nandit@cedar1~]$ salloc --account=def-<user> --node=1 --mem=6G \
--time=02:00:00 --cpus-per-task=2
[nandit@cdr767 ~]$


[nandit@cdr767~]$ module load singularity
[nandit@cdr767~]$ singularity shell biotools.sif
Singularity> samtools --help
Program: samtools (Tools for alignments in the SAM format)
Version: 1.13-14-g65a97fe (using htslib 1.13-19-g31bf087)
Usage:   samtools <command> [options]
```

Run containers on a job scheduler

2. Batch jobs

A job script (saved as jobscript.sh) to run the samtools program using Singularity

```
[nandit@cedar1~]$ cat jobscript.sh

#!/bin/bash

#SBATCH --account=def-<user>
#SBATCH --node=1
#SBATCH --time=02:00:00
#SBATCH --mem=6G
#SBATCH --cpus-per-task=2

module load singularity

singularity exec -B /path/to/input:/data biotools.sif  samtools index -bc -@ 2
/data/input.bam input.index
```

Submit the job script

```
[nandit@cedar1~]$ sbatch jobscript.sh
```

Happy to take your questions!!

Thank you for your attention……