Containers for HPC

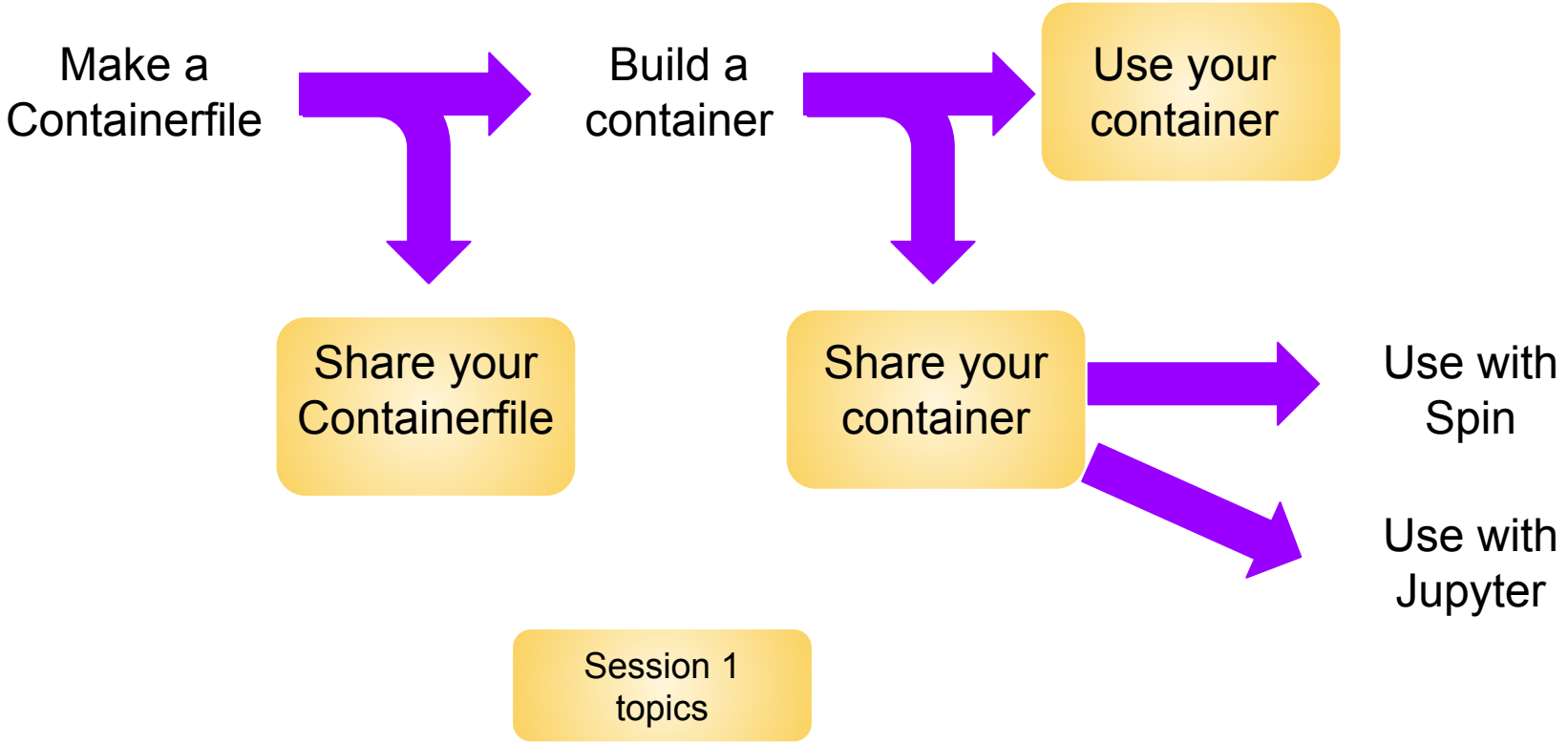
Session 2

Slides: https://bit.ly/20250313Container
Q&A Doc & Survey

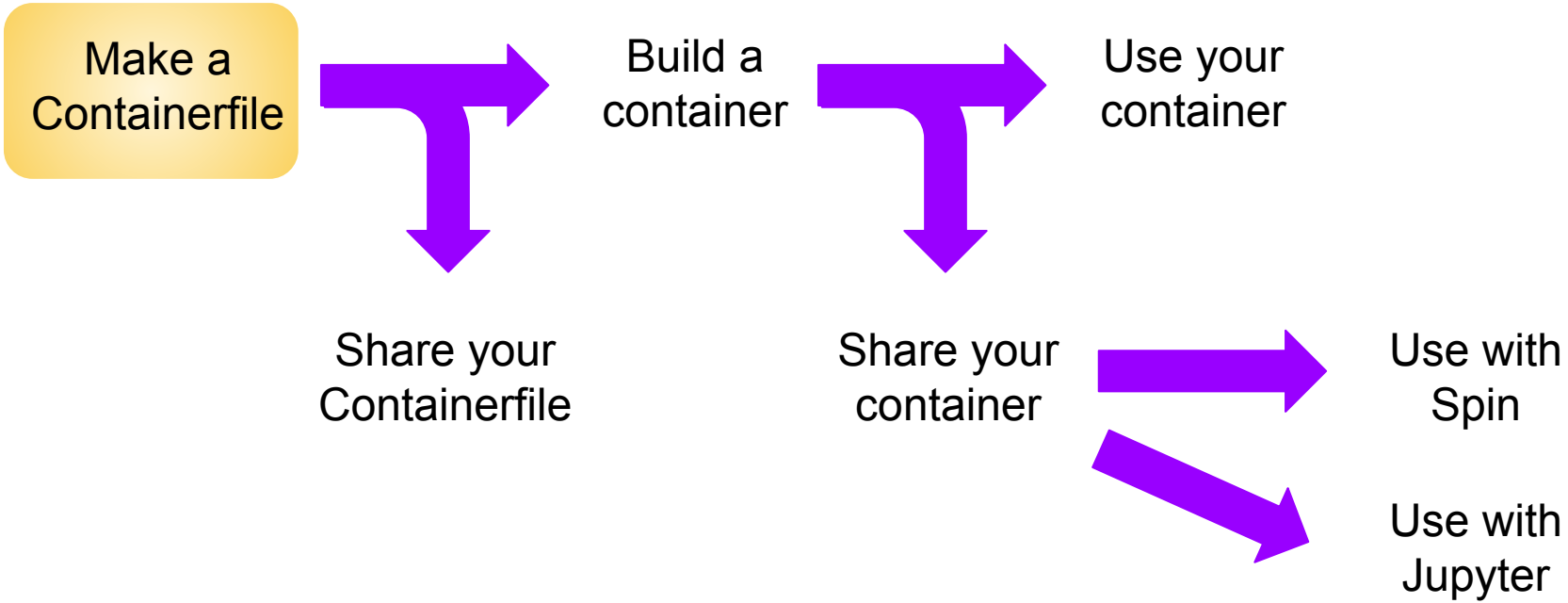
NERSC Containers Training
13 March 2025

NERSC Staff

Common Container Workflows



Containerfiles



Containerfiles - Overview

Explanations:

Grab your OS of choice

Set a working directory

Set up arguments

- Used in the container process

Use RUN to do lines of your installation

- Note: might need some system software that is already included on HPC systems
- Use Yes for all prompts

Set up environment variables

Copy files

Set up what to do

Best Practice: start with a bash script of your installation

Dummy example showing commands:

```
FROM docker.io/library/ubuntu:24.04
WORKDIR /opt
ARG code_ver=9.81

RUN apt-get update && \
    apt-get install -y wget
RUN wget https://code.org/rel/code-$code_ver.tar.gz \
    && tar xf code-$code_ver.tar.gz
ENV PATH=/opt/code/bin:$PATH
COPY codeFileA codeFileB
ENTRYPOINT echo "Hi, from the container"
```

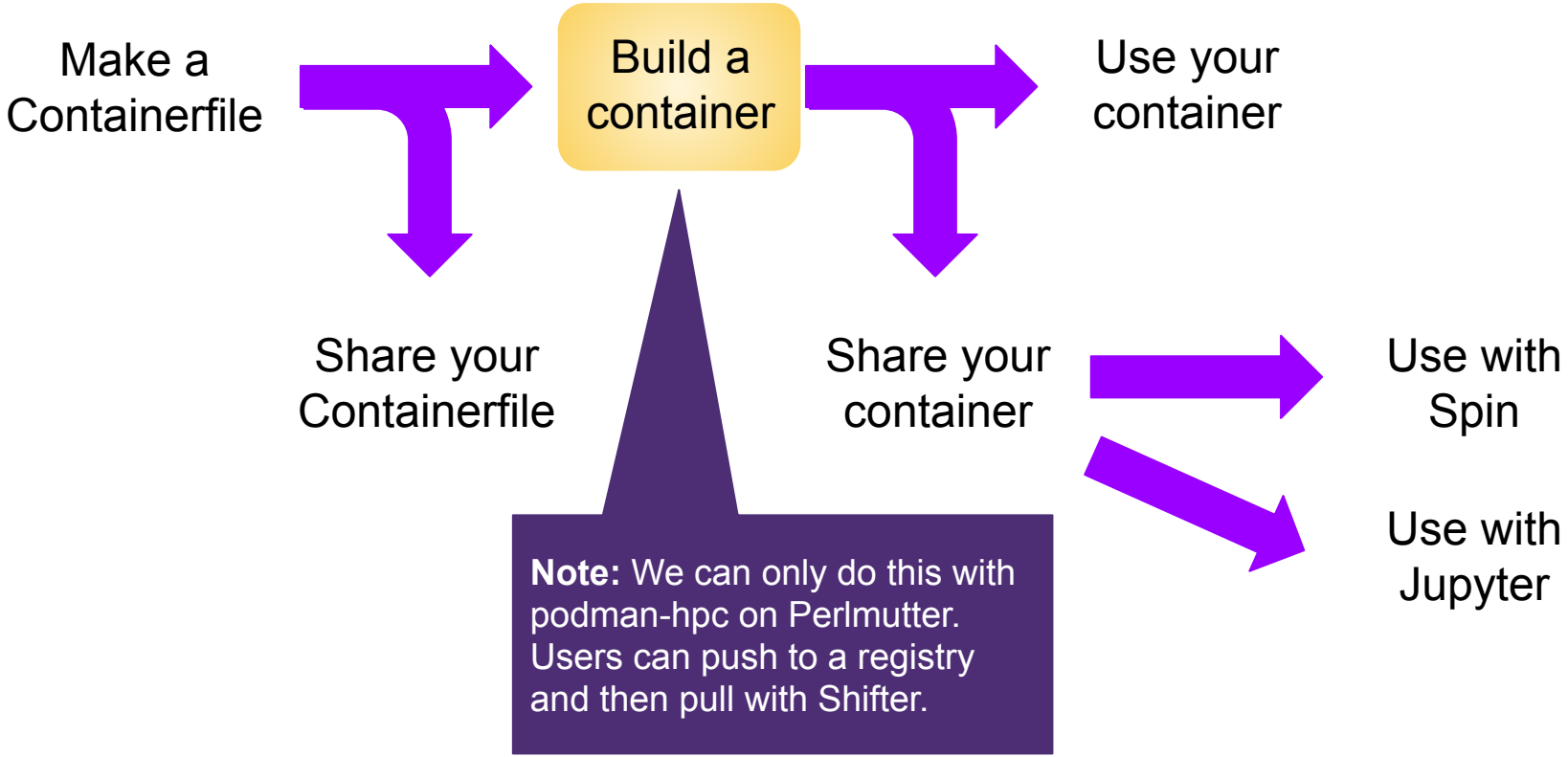
Note the use of && combining commands and \ for line wrapping

Containerfiles - Examples and Documentation

- NERSC documentation
- NERSC gitlab
- Shifter intro (slide 7)
- Containers for HPC intro (slide 23)
- Dockerfile basics
- NVIDIA container files

OCI compliance: what works in docker *should* work for podman

Building Containers



Building a Container with podman-hpc

Create a Containerfile (and call it Containerfile):

```
FROM docker.io/library/ubuntu:24.04

ENTRYPOINT echo "no lols here"
```

- Start with a base operating system
- Automatically run when you start the container

Create the container:

```
$ podman-hpc build -t nolols:1.0 .
```

Note: you must be in the same directory as Containerfile

- Use podman-hpc to build a container
- Tag this container with a name and version
- Build this container using a file called Containerfile found here

More podman-hpc Functionality

View your container:

```
$ podman-hpc images
localhost/nolols           1.0         59551900ead8   3 minutes ago   80.4 MB
docker.io/library/ubuntu   24.04       3db8720ecbf5   8 days ago      80.4 MB
```

- View the images

Migrate the container to scratch:

```
$ podman-hpc migrate nolols:1.0
```

- Use podman-hpc to move the container to scratch
- Which container we are migrating
  - Note that if you update your container on the login node you must remigrate

Note: you do this for performance[HPC] reasons. This is done automatically for containers you pull from a registry.

Helpful Building Hints

podman-hpc builds are done right in front of you

- Typically easier to troubleshoot build/run than moving between build/store/run locations
- Shifter requires you to build locally (typically with Docker), push to a registry and then pull onto Perlmutter

Migrating is important for runs in jobs

- No need to migrate for testing on the login-nodes

Builds are cached locally

- Rebuilds will take (MUCH) less time if steps are cached
- Switching login nodes means you start over :-(

Helpful Troubleshooting Hints

Container builds are done task by task

- Look at the screen/logfile for help on when failures occur
- `> STEP 4/5: RUN apt-get update && apt-get install -y --no-install-recommends wget`
- Troubleshoot by ending a Containerfile before the error, building, and then starting the container in interactive mode

Note: interactive container mode is different from an interactive (vs. batch) job

Sometimes it's helpful to break apart tasks that are grouped together

- The `&&` allows for multiple commands to be done with the same `RUN`
- Breaking this up allows you to interrogate the container mid-build

Run Containers Interactively

Common problem: Finding libraries

- Install libraries via package manager that are on PM as modules
- Libraries are in different locations - build script for PM will not work

Solution: Start your container interactively

```
$ podman-hpc run --rm -it hifrominside:1.0 /bin/bash
```

- Run interactive container mode
- Shell to enter (must be installed!)

Tip: Your bash history within the container is not saved on PM! Use copy/paste to make sure you keep your command.

- Use `find` or `locate` to find your required location
- Try your build command in the container
- Exit the container (`exit`) and update your Containerfile

Containers with MPI

Shifter and podman-hpc runtimes swap out your container's MPI

- Use the vendor optimized parallelization libraries at runtime on Perlmutter
  - Common for HPC facilities with specialized networks
- Relies on API compliance (good) and ABI compatibility (not perfect)

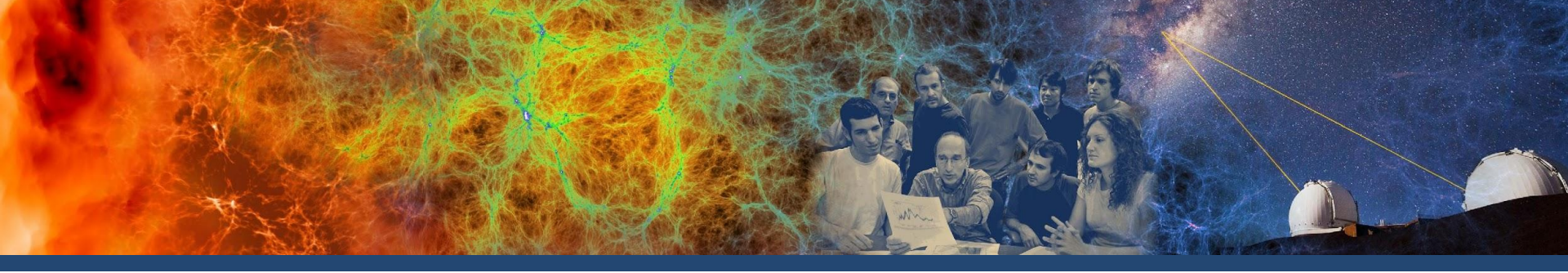
How to successfully do this?

- Build your container the way that you want using MPICH
  3.4 for mpi4py, 3.4 - 4.2 for most C/C++
- Run using the `--module=mpi` module with Shifter

```
$ srun -N 2 --module=mpich shifter command
```

- Run using the `--mpi` flag with podman-hpc

```
$ srun -n 2 podman-hpc run --rm --mpi loc/name:tag command
```



Session 2 Exercises

Exercises

1. Create a hello world container using MPI
   a. Start with this image if you don't want to build your own MPICH (info is here)
   b. Start with this repo if you don't want to build your own hello world
2. Run this on a login node with a single process
3. Migrate the container to scratch
4. Run this with an interactive job on 2 nodes
5. Run this as a batch job on 2 nodes

Bonus:

1. *Push your image to a registry*
2. *Pull your image with Shifter*
3. *Run this with 2 nodes*

Please fill out the survey!

Exercise Solutions

Legend

- Lines that start with $ are typed on Perlmutter
- Lines that start with $$ are in an interactive job
- Lines without any $s are in a file

Containerfile (using the examples)

```
FROM awlavely/mpich_ubuntu:1.0
WORKDIR /opt
RUN git clone https://github.com/adamlavely/mpihelloworldNodeName.git
RUN cd mpihelloworldNodeName                          && \
    mpicc -o hello.out mpiHelloWorldNodeAndRank.c && \
    chmod 777 hello.out                               && \
    cp hello.out /usr/local/bin/.
```

Build the container

```
$ podman-hpc build -t mpich_helloworld:1.0 .
```

Run on the login node

```
$ podman-hpc run --rm mpich_helloworld:1.0 hello.out
```

Migrate the container

```
$ podman-hpc migrate mpich_helloworld:1.0
```

Run on debug queue

```
$ salloc -N 2 -t 5 -C cpu --qos interactive
$$ srun -n 2 podman-hpc run --rm --mpi mpich_helloworld:1.0 hello.out
$$ exit
```

Submission Script (mpihello.slurm):

```
#!/bin/bash
#SBATCH --qos=debug
#SBATCH --nodes=2
#SBATCH --time=5
#SBATCH --constraint=cpu
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err
srun -n 2 podman-hpc run --rm --mpi mpich_helloworld:1.0 hello.out
```

Run as batch job

```
$ sbatch mpihello.slurm
```