



NHR for
Computational
Engineering
Science



TECHNISCHE
UNIVERSITÄT
DARMSTADT

OpenFOAM Apptainer Containers for HPC systems

We'll look at...

- 1• Introduction to Containers for HPCs
- 2• Cluster admins and container tech
- 3• Apptainer containers 101
- 4• A convenient apptainer containers mechanism
- 5• Apptainer containers for OpenFOAM
- 6• Concluding remarks

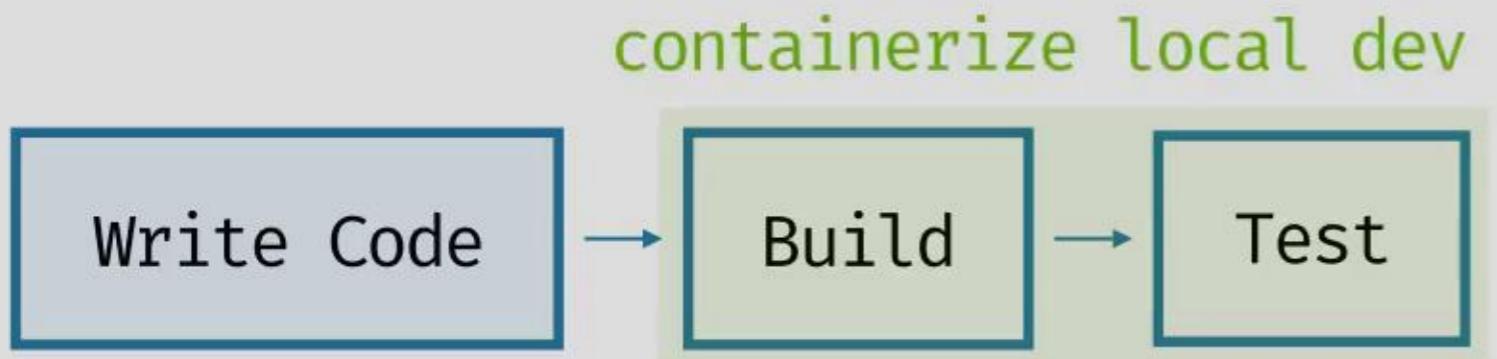
1› Introduction to Containers for HPCs

Write Code

1› Introduction to Containers for HPCs

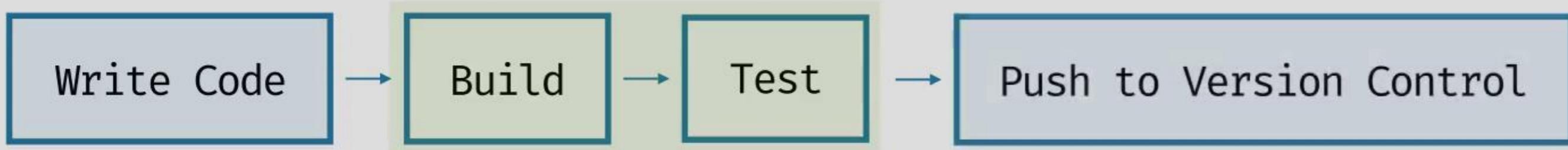


1› Introduction to Containers for HPCs

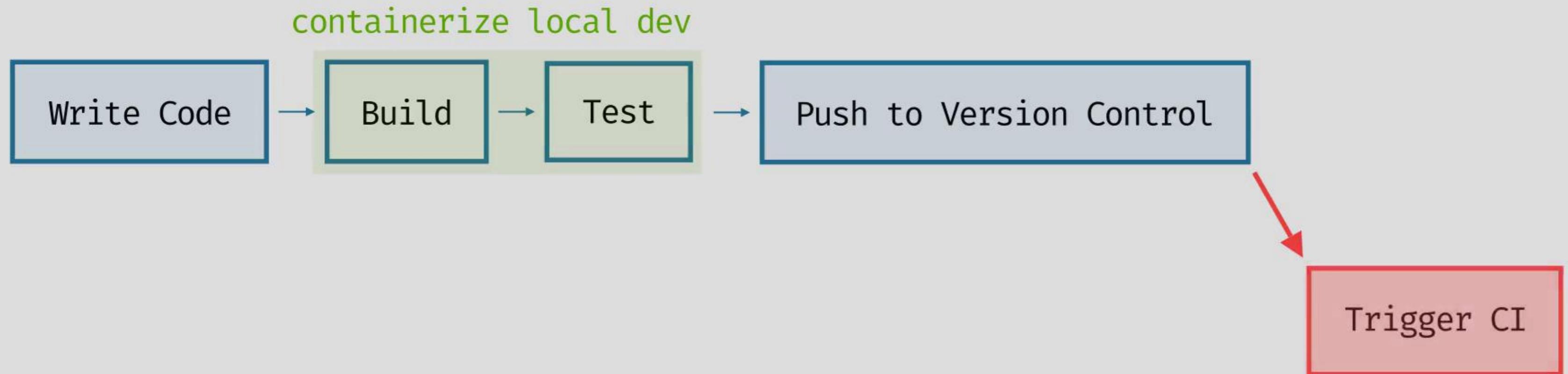


1› Introduction to Containers for HPCs

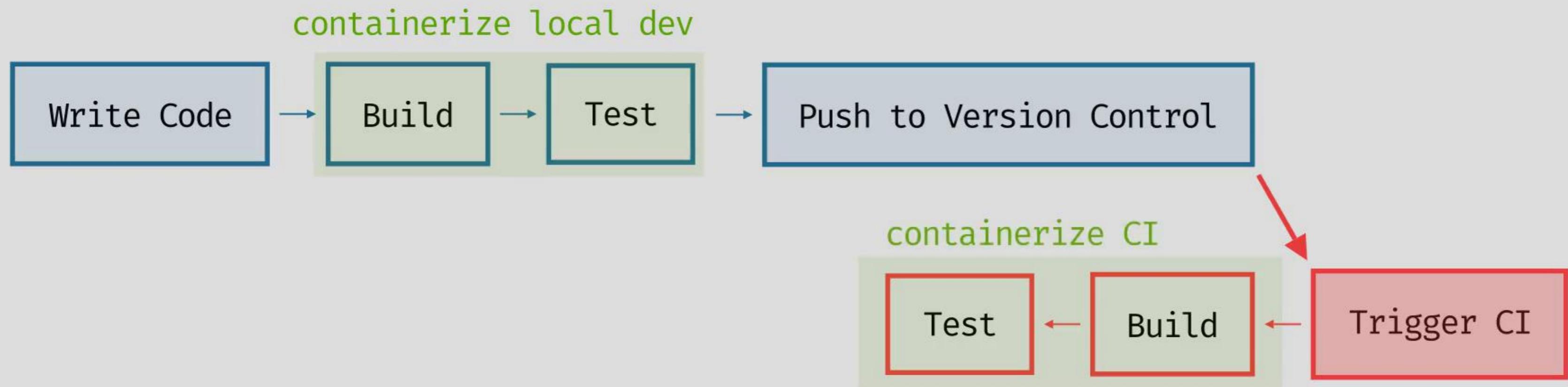
containerize local dev



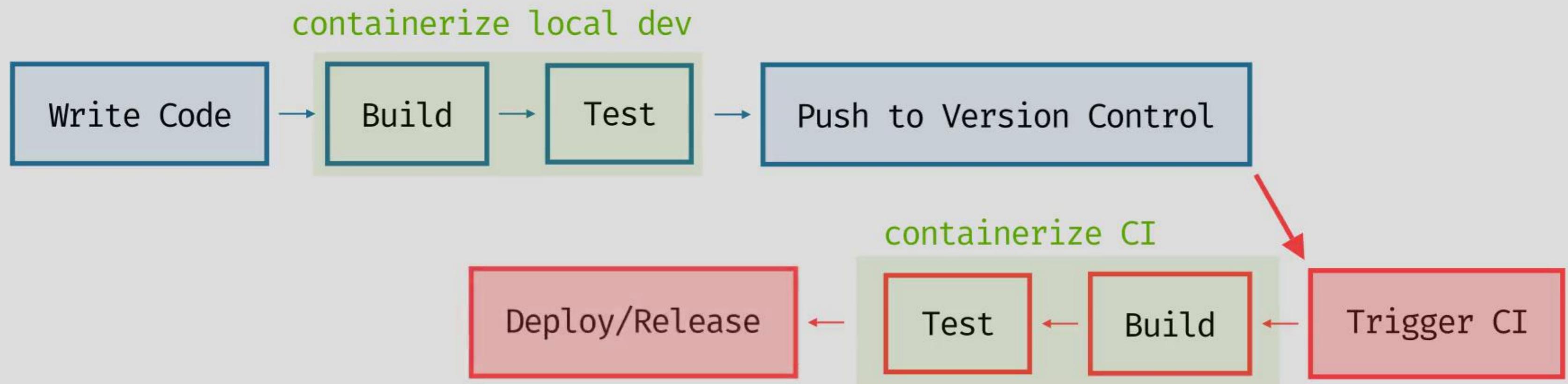
1› Introduction to Containers for HPCs



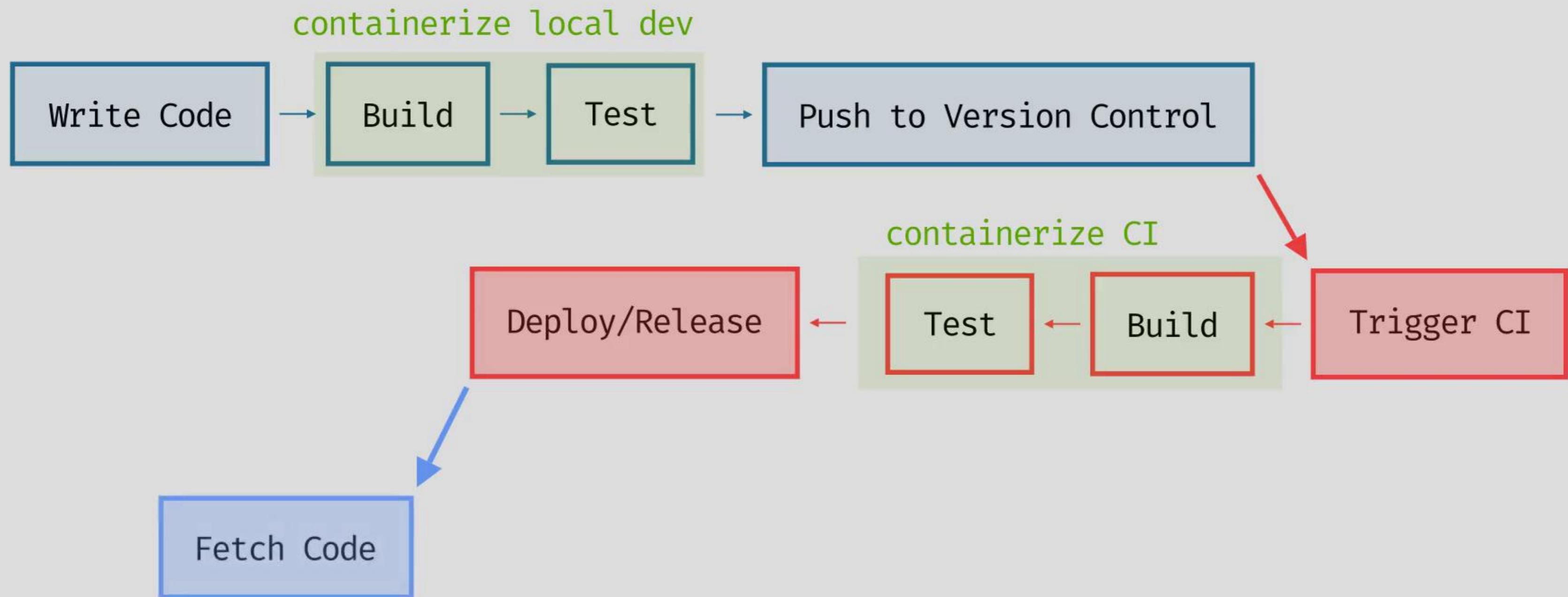
1› Introduction to Containers for HPCs



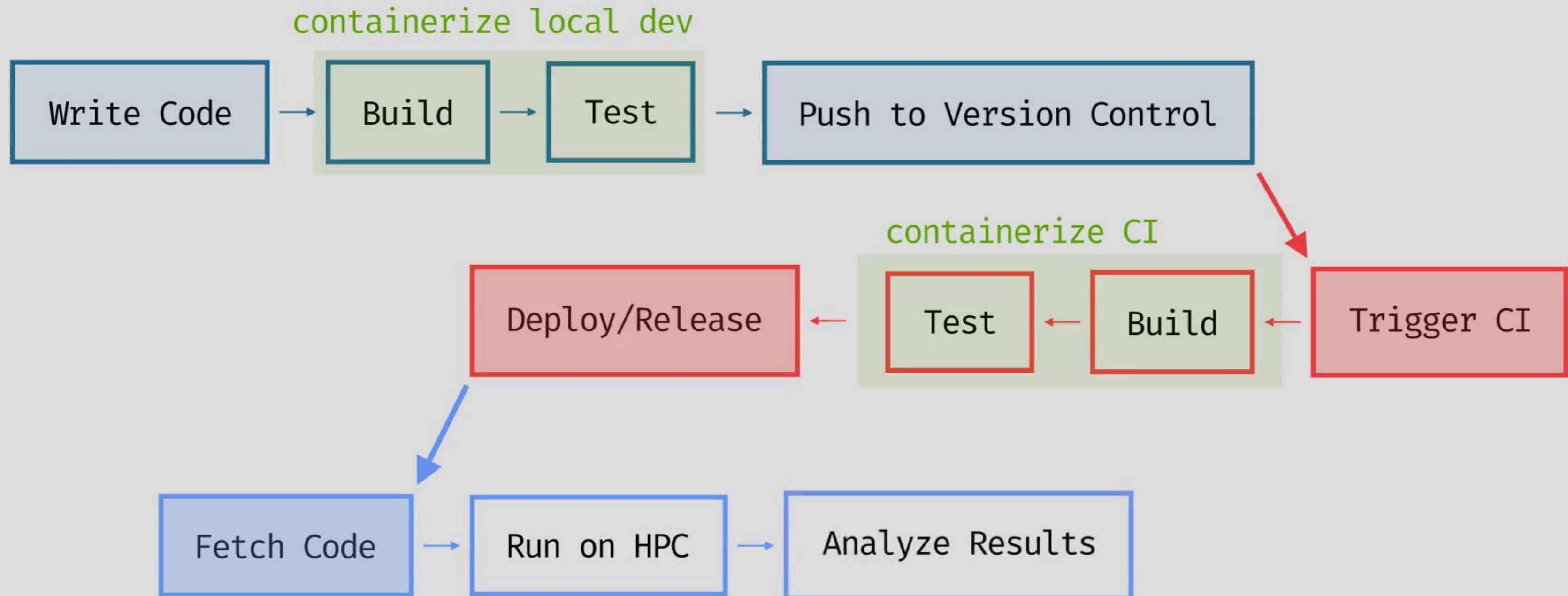
1› Introduction to Containers for HPCs



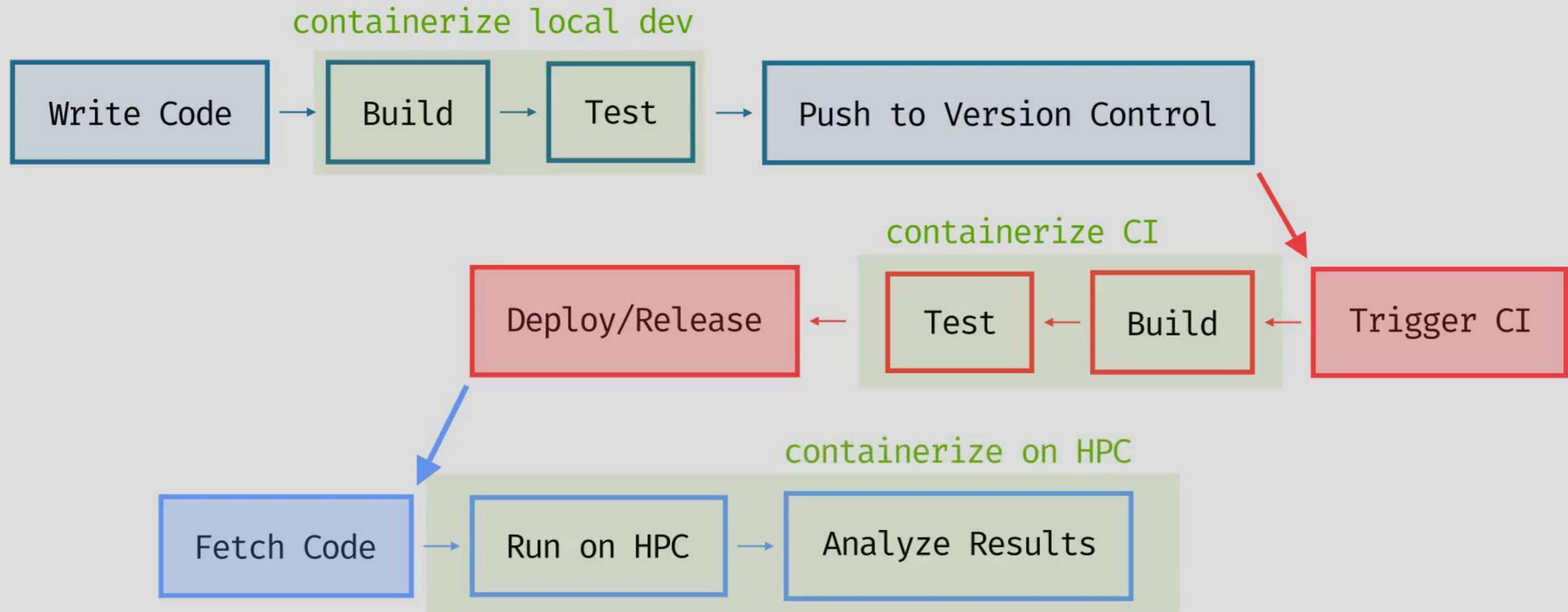
1› Introduction to Containers for HPCs



1› Introduction to Containers for HPCs



1› Introduction to Containers for HPCs



1› Introduction to Containers for HPCs

Two common types for isolation

Two common types for isolation

Containers

Virtual machines

Two common types for isolation

Containers

- 1• Partial isolation from host
- 2• Uses host's Kernel
- 3• Controlled versionning of dependencies

Virtual machines

Two common types for isolation

Containers

- 1• Partial isolation from host
- 2• Uses host's Kernel
- 3• Controlled versionning of dependencies

Virtual machines

- 1• Full isolation from host
- 2• Dedicated resources
- 3• Provisionning overhead

Two common types for isolation

Containers

- 1• Partial isolation from host
- 2• Uses host's Kernel
- 3• Controlled versionning of dependencies

Virtual machines

- 1• Full isolation from host
- 2• Dedicated resources
- 3• Provisionning overhead

Main Objective: cluster-agnostic sharing and reproducing of scientific workflows/data

1› Introduction to Containers for HPCs

Two container usage aspects

1› Introduction to Containers for HPCs

Two container usage aspects

Image creation and hosting

Container runs

Two container usage aspects

Image creation and hosting

- 1• Build from definition files?**
- 2• Can CI/CD build images?**
- 3• Main purpose of the image?**
- 4• Host images at public HUBs**

Container runs

Two container usage aspects

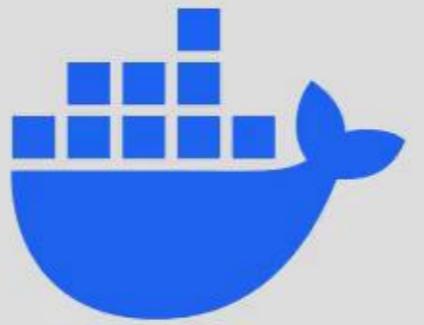
Image creation and hosting

- 1• Build from definition files?
- 2• Can CI/CD build images?
- 3• Main purpose of the image?
- 4• Host images at public HUBs

Container runs

- 1• MPI/Slurm compatibility?
- 2• Integration with HPC infrastructure
- 3• User convenience, even with minimal training
- 4• Security!!!

2› Cluster admins and container tech

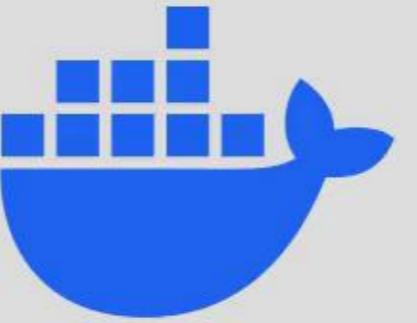


Docker: most popular

2› Cluster admins and container tech



Podman: avoids Docker's daemon

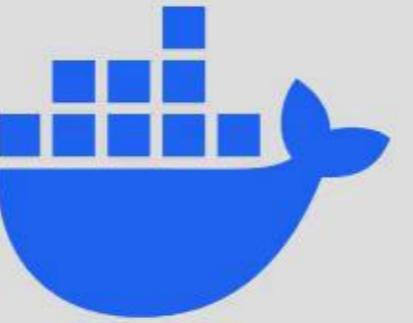


Docker: most popular

2› Cluster admins and container tech



Podman: avoids Docker's daemon



Docker: most popular



SARUS: HPC-focused

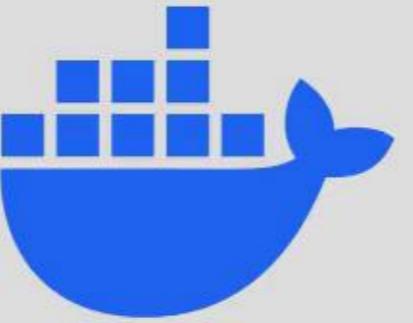
2› Cluster admins and container tech



Singularity: Single-Process-Execution



Podman: avoids Docker's daemon



Docker: most popular



SARUS: HPC-focused

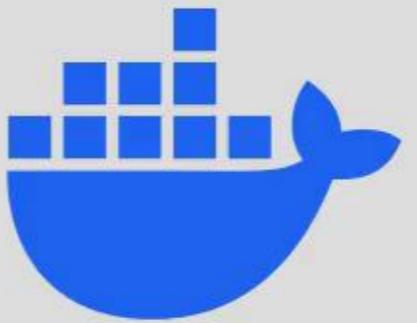
2› Cluster admins and container tech



Singularity: Single-Process-Execution



Podman: avoids Docker's daemon



Docker: most popular



Apptainer: Modernized Singularity fork



SARUS: HPC-focused

3› Apptainer containers 101

```
1 # definition file for a dummy image: test.def prodcing test.sif
2
3 Bootstrap: docker # or localimage
4 From: ubuntu:latest # <- starting image
5
6
7
8
9
10
11
12
13 %runcscript # <- what to do when 'apptainer run image.sif'
14     if [ $# -eq 0 ]; then /bin/bash; else /bin/bash -c "$@"; fi
```

```
1 # definition file for a dummy image: test.def prodcing test.sif
2
3 Bootstrap: docker # or localimage
4 From: ubuntu:latest # <- starting image
5
6 %post -c /bin/bash # <- default is sh
7     echo "echo Sourced /bashrc" > /bashrc
8
9
10
11
12
13 %runcscript # <- what to do when 'apptainer run image.sif'
14     if [ $# -eq 0 ]; then /bin/bash; else /bin/bash -c "$@"; fi
```

3› Apptainer containers 101

```
1 # definition file for a dummy image: test.def prodcing test.sif
2
3 Bootstrap: docker # or localimage
4 From: ubuntu:latest # <- starting image
5
6 %post -c /bin/bash # <- default is sh
7     echo "echo Sourced /bashrc" > /bashrc
8
9 %environment # <- set environment when run'ing/exec'ing
10    #!/bin/bash
11    source /bashrc
12
13 %runscript # <- what to do when 'apptainer run image.sif'
14     if [ $# -eq 0 ]; then /bin/bash; else /bin/bash -c "$@"; fi
```



```
1 apptainer build test.sif test.def # <- build test.sif image
2
3 apptainer run -C test.sif          # <- run in more isolated mode
4 apptainer run test.sif            # <- share env., PIDs... etc
5 apptainer run test.sif hostname   # <- run a command and exit
6
7 apptainer exec test.sif hostname  # <- skips %runcscript
8
9 apptainer push test.sif oras://ghcr.io/USER/REPO:version
10 apptainer pull oras://ghcr.io/USER/REPO:version
```

4) A convenient apptainer containers mechanism

NHR⁴
CES



Automated container building based on YAML configs using Ansible

4) A convenient apptainer containers mechanism

Automated container building based on YAML configs using Ansible



```
1 containers:
2   basic:
3     opencfd-openfoam:
4       os:
5         distro: ubuntu
6         version: 24.04
7       mpi:
8         implementation: openmpi
9         version: 4.1.5
10      framework:
11        definition: com-openfoam
12        version: 2312
13        git_ref: default
```

4) A convenient apptainer containers mechanism

Automated container building based on YAML configs using Ansible



```
1 containers:  
2   basic:  
3     opencfd-openfoam:  
4       os:  
5         distro: ubuntu  
6         version: 24.04  
7       mpi:  
8         implementation: openmpi  
9         version: 4.1.5  
10      framework:  
11        definition: com-openfoam  
12        version: 2312  
13        git_ref: default
```



```
13  projects:  
14    test:  
15      base_container: opencfd-openfoam  
16      definition: projects/test.def  
17      build_args:  
18        branch:  
19          - master
```

4) A convenient apptainer containers mechanism



```
1 # Get dependency repo to a temp location
2 git clone https://github.com/FoamScience/openfoam-apptainer-packaging /tmp/tainers
3
4 # build (in this order):
5 #     containers/basic/ubuntu-24.04-openmpi-4.1.5.sif
6 #     containers/basic/opencfd-openfoam.sif
7 #     containers/projects/test-master.sif      <- your final container
8 # original dir is where your config.yaml is located
9 ansible-playbook /tmp/tainers build.yaml \
10   --extra-vars "original_dir=$PWD" \
11   --extra-vars "@config.yaml"
```

4) A convenient apptainer containers mechanism



```
1 # definition files are now templated!
2
3 Bootstrap: localimage
4 From: {{ CONTAINERS_DIR }}/basic/{{ BASE_CONTAINER }}.sif
5
6 %post -c /bin/bash  # <- default is sh
7     echo "compiling {{ BRANCH }}"
8     # also, write meta data to /apps.json file
9
10 %environment #
11     #!/bin/bash
12     # unified sourcing of bashrcs and python envs here
13
14 %runcscript  # <- always looks like this
15     if [ $# -eq 0 ]; then /bin/bash; else /bin/bash -c "$@"; fi
```

4) A convenient apptainer containers mechanism

```
apptainer run containers/projects/test-master.sif info
```

4) A convenient apptainer containers mechanism

```
apptainer run containers/projects/test-master.sif info
```



```
1 "openmpi": {  
2   "version": "4.1.5"  
3 },  
4 "openfoam": {  
5   "fork": "com-openfoam",  
6   "branch": "default",  
7   "commit": "default",  
8   "version": "2312"  
9 },
```



```
10  "test": {  
11    "ompi_test": "/opt/OMPIFoam/ompiTest",  
12    "foam_test": "/opt/OMPIFoam/testOMPIFoam",  
13    "branch": "master"  
14 }
```

5› Apptainer containers for OpenFOAM

Scenario 1: OpenFOAM scalability study

Scenario 1: OpenFOAM scalability study

- 1• Assume we have a configuration that builds a project in Opt/Debug modes
- 2• Want to add HPCToolkit to the Debug container
- 3• Upload the SIF file to multiple clusters and compare scalability

Scenario 1: OpenFOAM scalability study

- 1• Assume we have a configuration that builds a project in Opt/Debug modes
 - 2• Want to add HPCToolkit to the Debug container
 - 3• Upload the SIF file to multiple clusters and compare scalability
-
- 1• The project's definition file is mostly a SHELL language
 - 2• So, minimal maintenance overhead for the definition files
 - 3• The YAML configuration file allows seamlessly stitching base containers!
 - 4• Also, it allows loading your own base container definitions

5› Apptainer containers for OpenFOAM



```
1 containers:
2   basic:
3     opencfd-openfoam:
4       os:
5         distro: ubuntu
6         version: 24.04
7       mpi:
8         implementation: openmpi
9         version: 4.1.5
10      framework:
11        definition: com-openfoam
12        version: 2312
13
14 #####
```

5› Apptainer containers for OpenFOAM



```
1 containers:  
2   basic:  
3     opencfd-openfoam:  
4       os:  
5         distro: ubuntu  
6         version: 24.04  
7       mpi:  
8         implementation: openmpi  
9         version: 4.1.5  
10      framework:  
11        - definition: com-openfoam  
12          version: 2312  
13        - definition: hpctoolkit  
14          version: 2024.01.99-next
```

5› Apptainer containers for OpenFOAM



```
1 containers:  
2   basic:  
3     opencfd-openfoam:  
4       os:  
5         distro: ubuntu  
6         version: 24.04  
7     mpi:  
8       implementation: openmpi  
9       version: 4.1.5  
10    framework:  
11      - definition: com-openfoam  
12        version: 2312  
13      - definition: hpctoolkit  
14        version: 2024.01.99-next
```



```
15  projects:  
16    my_project:  
17      base_container: opencfd-openfoam  
18      definition: my_project.def  
19      build_args:  
20        mode:  
21          - Debug
```

5› Apptainer containers for OpenFOAM

Scenario 2: Test things on a makeshift cluster

Scenario 2: Test things on a makeshift cluster

- 1•** Spawn Docker containers to act as "SLURM cluster" nodes
- 2•** The nodes only need to have SLURM, Apptainer and some MPI installed
- 3•** Build your container images locally
- 4•** Upload them to the head node
- 5•** Run the containers in MPI-hybrid mode

Scenario 2: Test things on a makeshift cluster

- 1• Spawn Docker containers to act as "SLURM cluster" nodes
- 2• The nodes only need to have SLURM, Apptainer and some MPI installed
- 3• Build your container images locally
- 4• Upload them to the head node
- 5• Run the containers in MPI-hybrid mode



```
1 # from the head node, these should say same things
2 srun -N 3 hostname
3 srun -N 3 apptainer run my.sif hostname
4 # also valid if hostname were to support MPI
```

6› Concluding remarks

- 1• Single process execution is "the way to go" for HPC systems!
- 2• Automating containerization helps with dev productivity
- 3• Containers ease software/data sharing at frozen versions
- 4• Apptainer is still a bit under-supported on CI/CD systems

6» Concluding remarks

- 1• Single process execution is "the way to go" for HPC systems!
- 2• Automating containerization helps with dev productivity
- 3• Containers ease software/data sharing at frozen versions
- 4• Apptainer is still a bit under-supported on CI/CD systems

For all associated materials, and the slides:

6» Concluding remarks

- 1• Single process execution is "the way to go" for HPC systems!
- 2• Automating containerization helps with dev productivity
- 3• Containers ease software/data sharing at frozen versions
- 4• Apptainer is still a bit under-supported on CI/CD systems

For all associated materials, and the slides:

- 1• <https://github.com/FoamScience/lictenberg-training-apptainers-presentation>
- 2• <https://foamscience.github.io/lictenberg-training-apptainers-presentation>

THANK YOU for your attention!