Container in the NAF

A short introduction

Beyer Christoph
Hamburg, 28.03.2025

What is a container

And do I need to give it a special treatment

What is a Linux Container

A Linux container is a set of 1 or more processes that are isolated from the rest of the system. All the files necessary to run them are provided from a distinct image, meaning Linux containers are portable and consistent as they move from development, to testing, and finally to production.

What do we provide

On the NAF we use apptainer (formerly known as singularity) as a containerization tool

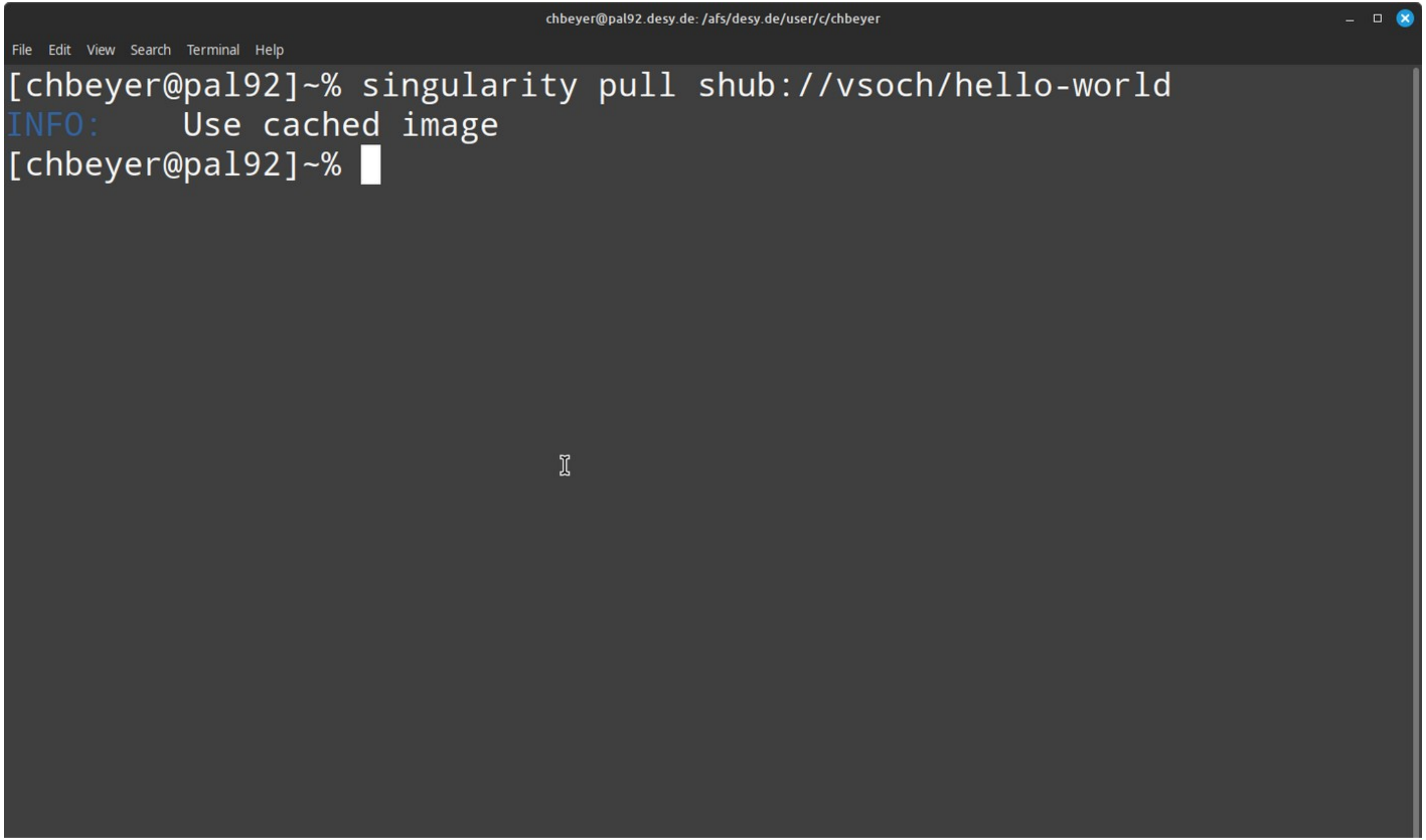
What is a Linux Container from the batchsystems view

Just something you want me to execute in your name on different computers, no matter what you call it

Why may it be intersting to let Condor know that I intend to run a container

Spoiler: Some additional preparation, mainly the bind mounts of common shares into the container will be done automatically

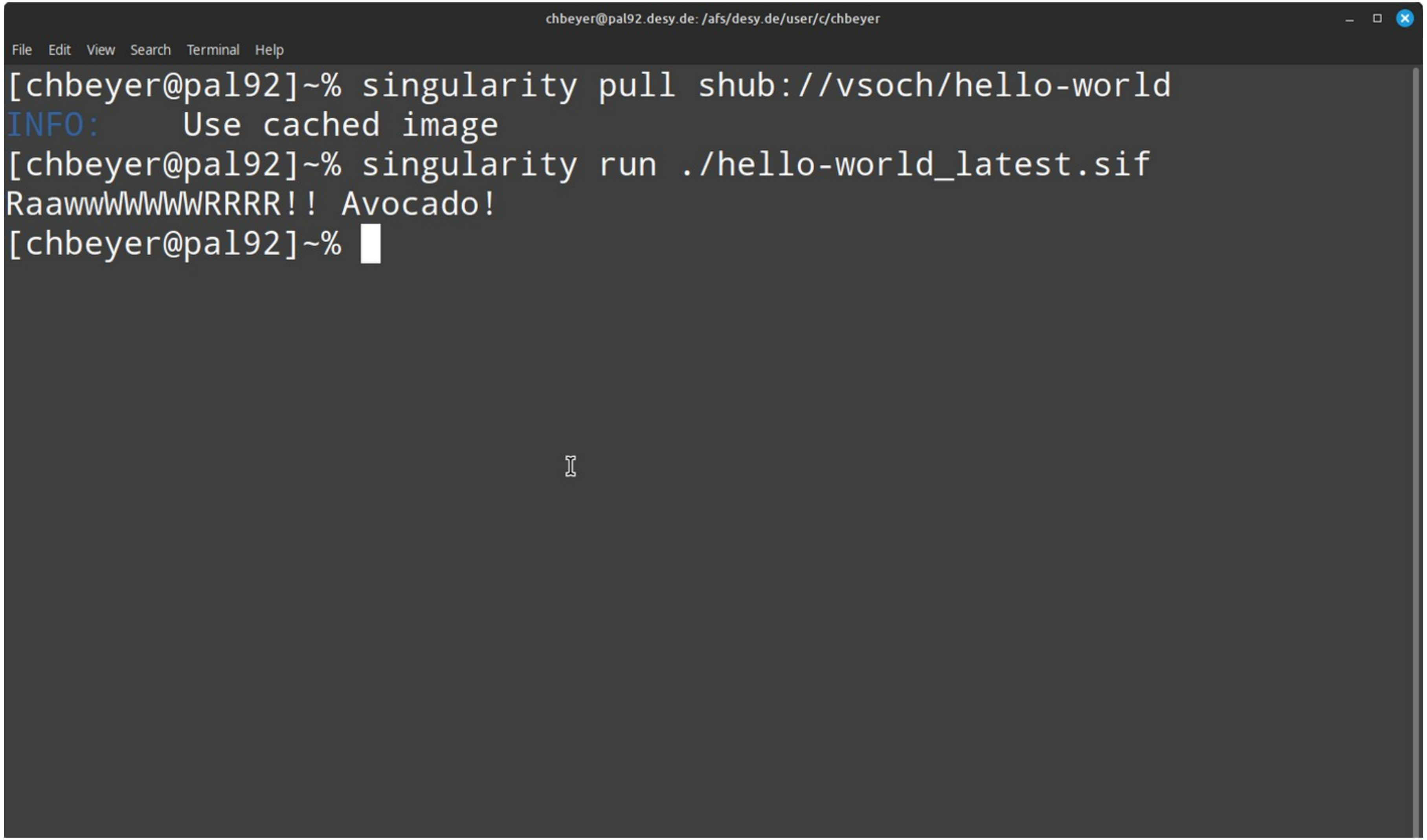
Run a container like an executable



chbeyer@pal92.desy.de: /afs/desy.de/user/c/chbeyer
File Edit View Search Terminal Help

```
[chbeyer@pal92]~% singularity pull shub://vsoch/hello-world
INFO:    Use cached image
[chbeyer@pal92]~%
```

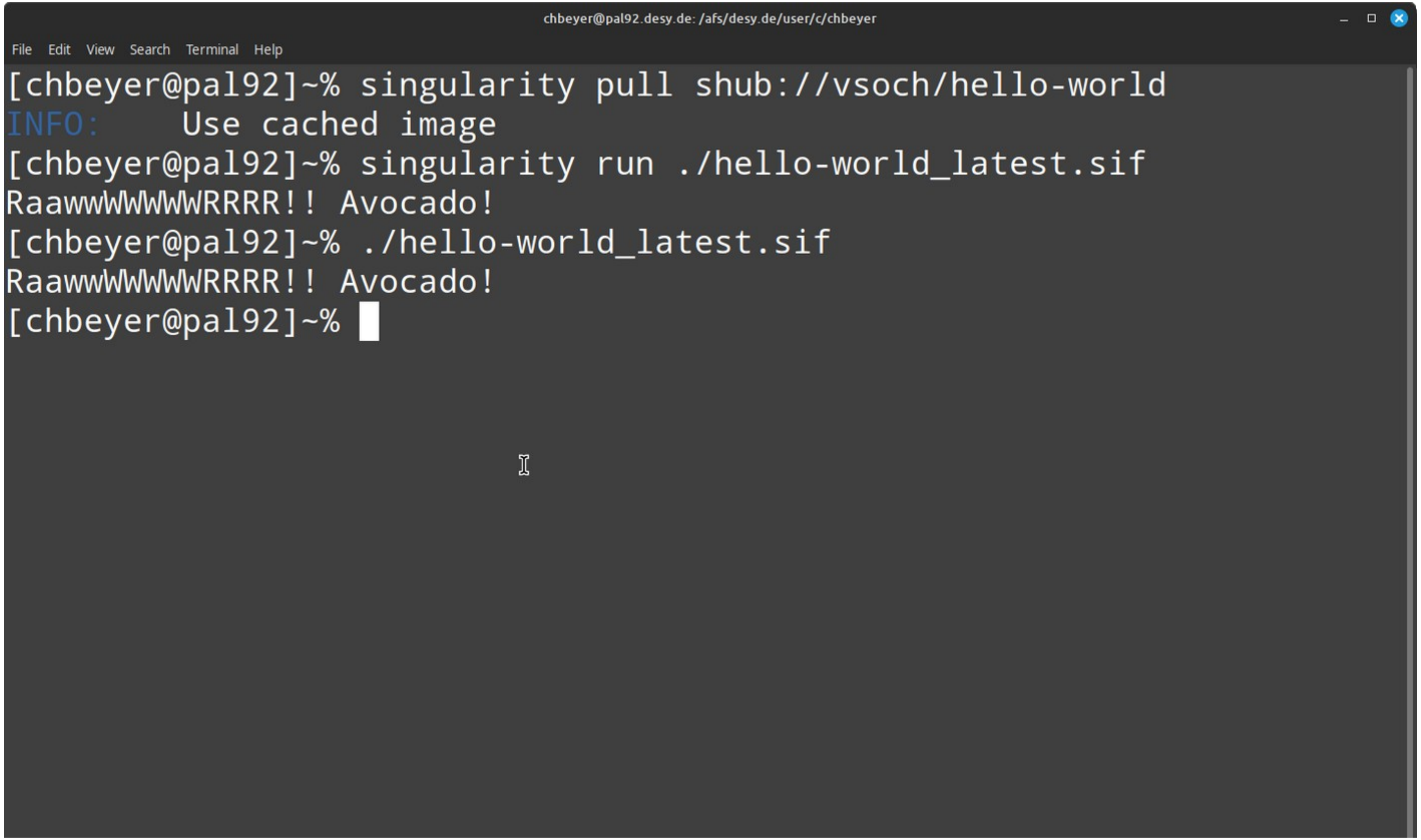
Run a container like an executable



chbeyer@pal92.desy.de: /afs/desy.de/user/c/chbeyer
File Edit View Search Terminal Help

```
[chbeyer@pal92]~% singularity pull shub://vsoch/hello-world
INFO:    Use cached image
[chbeyer@pal92]~% singularity run ./hello-world_latest.sif
RaawwWWWWWRRRR!! Avocado!
[chbeyer@pal92]~%
```

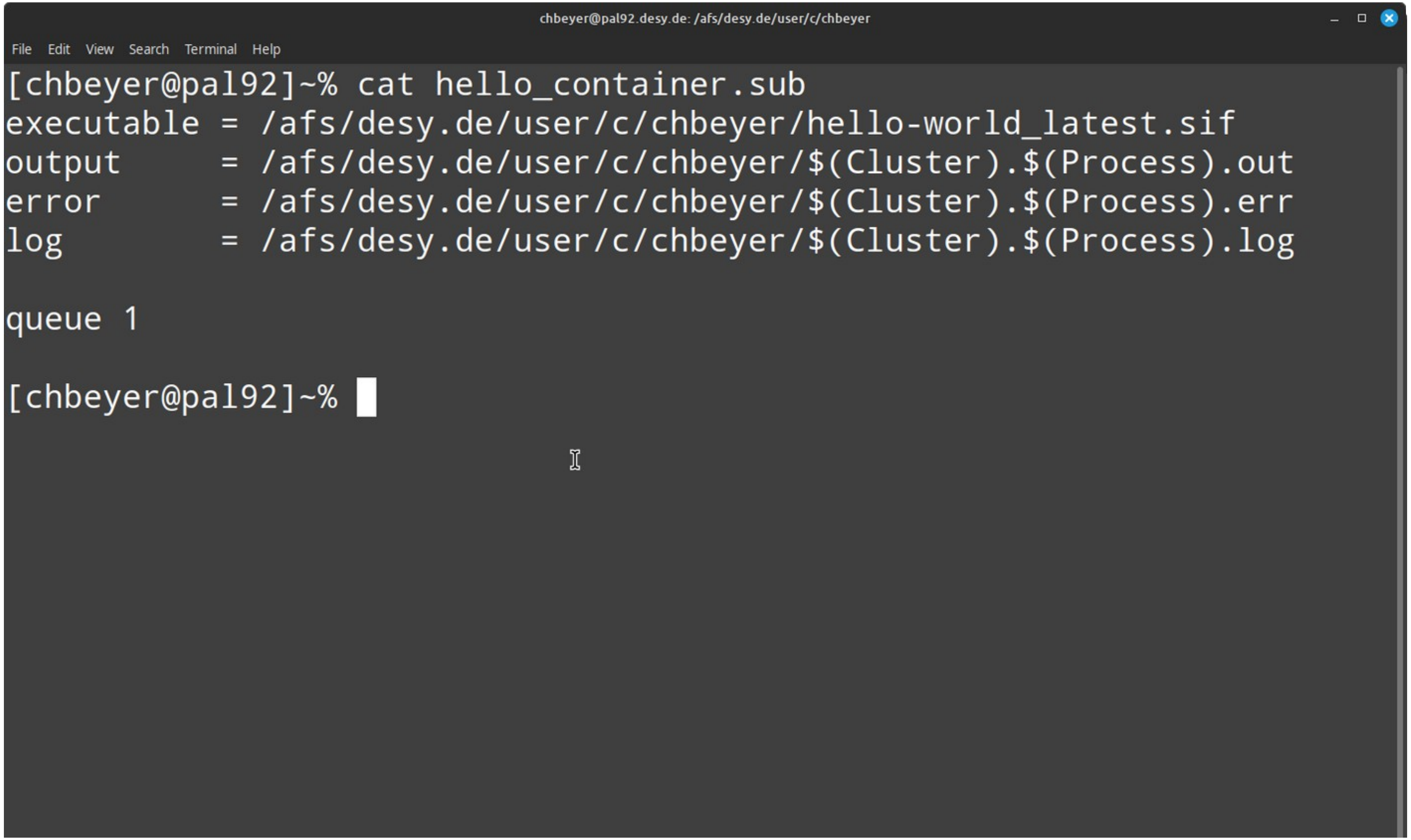
Run a container like an executable



chbeyer@pal92.desy.de: /afs/desy.de/user/c/chbeyer
File Edit View Search Terminal Help

```
[chbeyer@pal92]~% singularity pull shub://vsoch/hello-world
INFO:    Use cached image
[chbeyer@pal92]~% singularity run ./hello-world_latest.sif
RaawwWWWWWRRRR!! Avocado!
[chbeyer@pal92]~% ./hello-world_latest.sif
RaawwWWWWWRRRR!! Avocado!
[chbeyer@pal92]~%
```

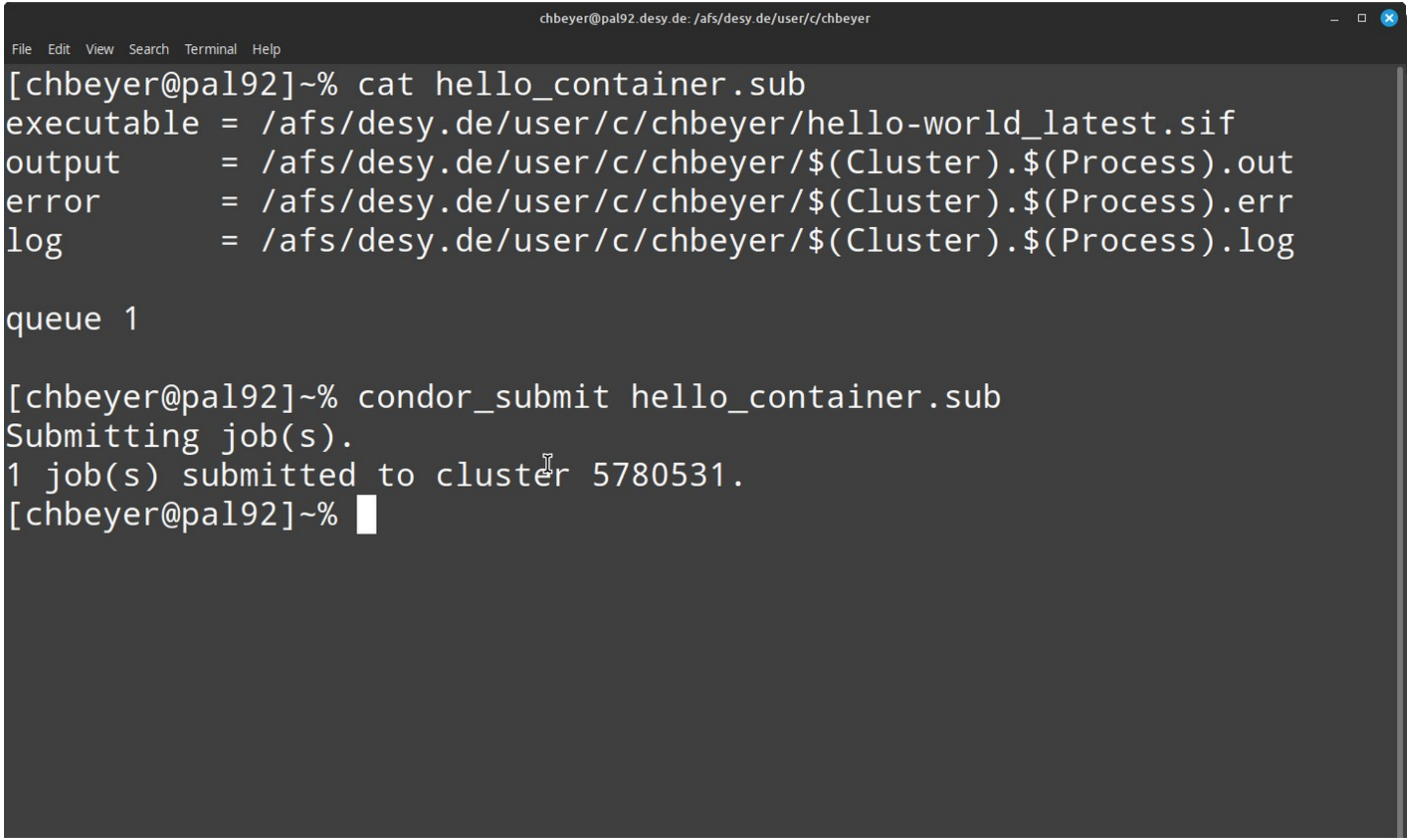
Let’s make a condor job out of it



chbeyer@pal92.desy.de: /afs/desy.de/user/c/chbeyer
File Edit View Search Terminal Help

```
[chbeyer@pal92]~% cat hello_container.sub
executable = /afs/desy.de/user/c/chbeyer/hello-world_latest.sif
output     = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).out
error      = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).err
log        = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).log

queue 1

[chbeyer@pal92]~%
```

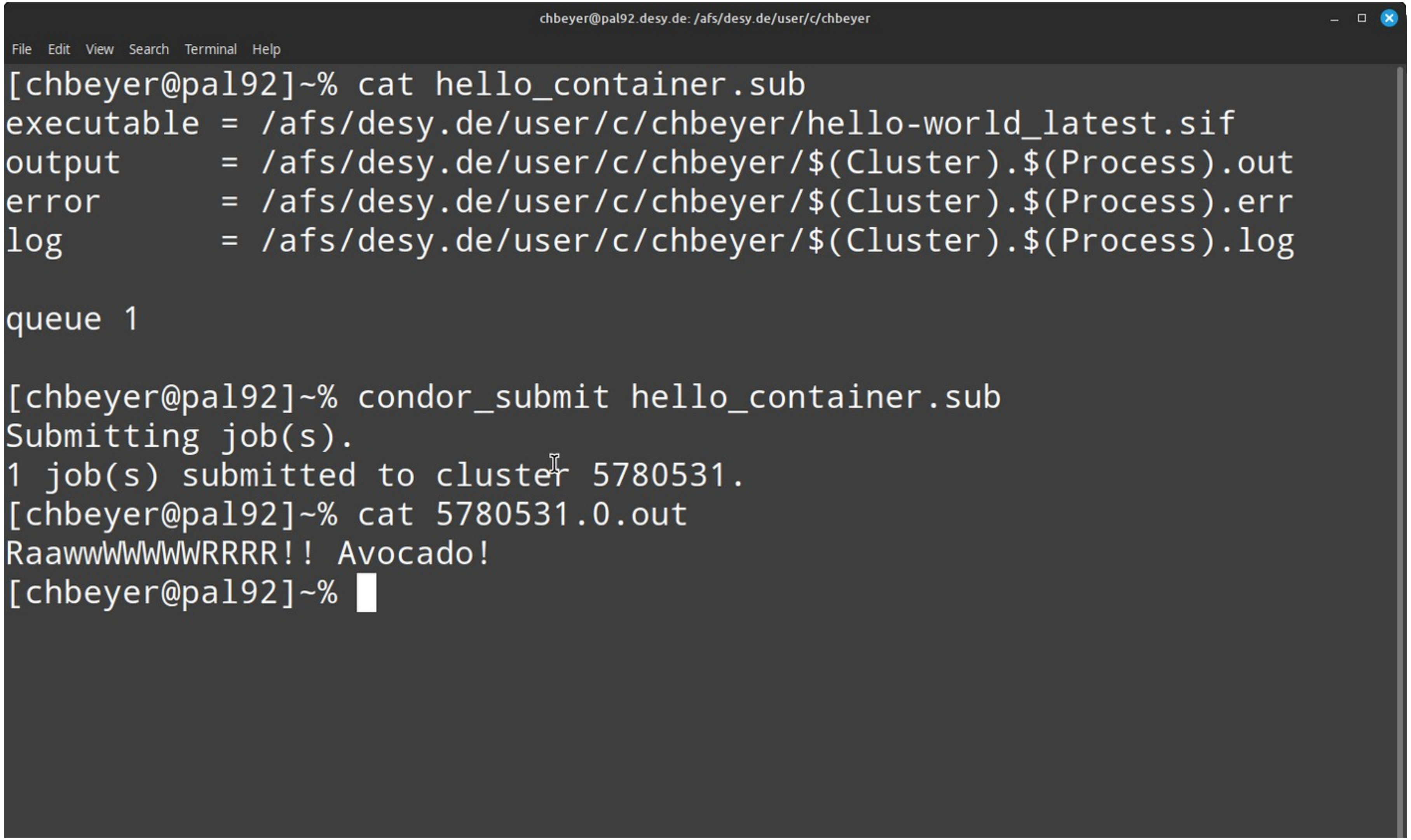
Let’s make a condor job out of it



chbeyer@pal92.desy.de: /afs/desy.de/user/c/chbeyer
File Edit View Search Terminal Help

```
[chbeyer@pal92]~% cat hello_container.sub
executable = /afs/desy.de/user/c/chbeyer/hello-world_latest.sif
output     = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).out
error      = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).err
log        = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).log

queue 1

[chbeyer@pal92]~% condor_submit hello_container.sub
Submitting job(s).
1 job(s) submitted to cluster 5780531.
[chbeyer@pal92]~%
```

Let’s make a condor job out of it



chbeyer@pal92.desy.de: /afs/desy.de/user/c/chbeyer
File Edit View Search Terminal Help

```
[chbeyer@pal92]~% cat hello_container.sub
executable = /afs/desy.de/user/c/chbeyer/hello-world_latest.sif
output     = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).out
error      = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).err
log        = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).log

queue 1

[chbeyer@pal92]~% condor_submit hello_container.sub
Submitting job(s).
1 job(s) submitted to cluster 5780531.
[chbeyer@pal92]~% cat 5780531.0.out
RaawwWWWWWRRRR!! Avocado!
[chbeyer@pal92]~% 
```

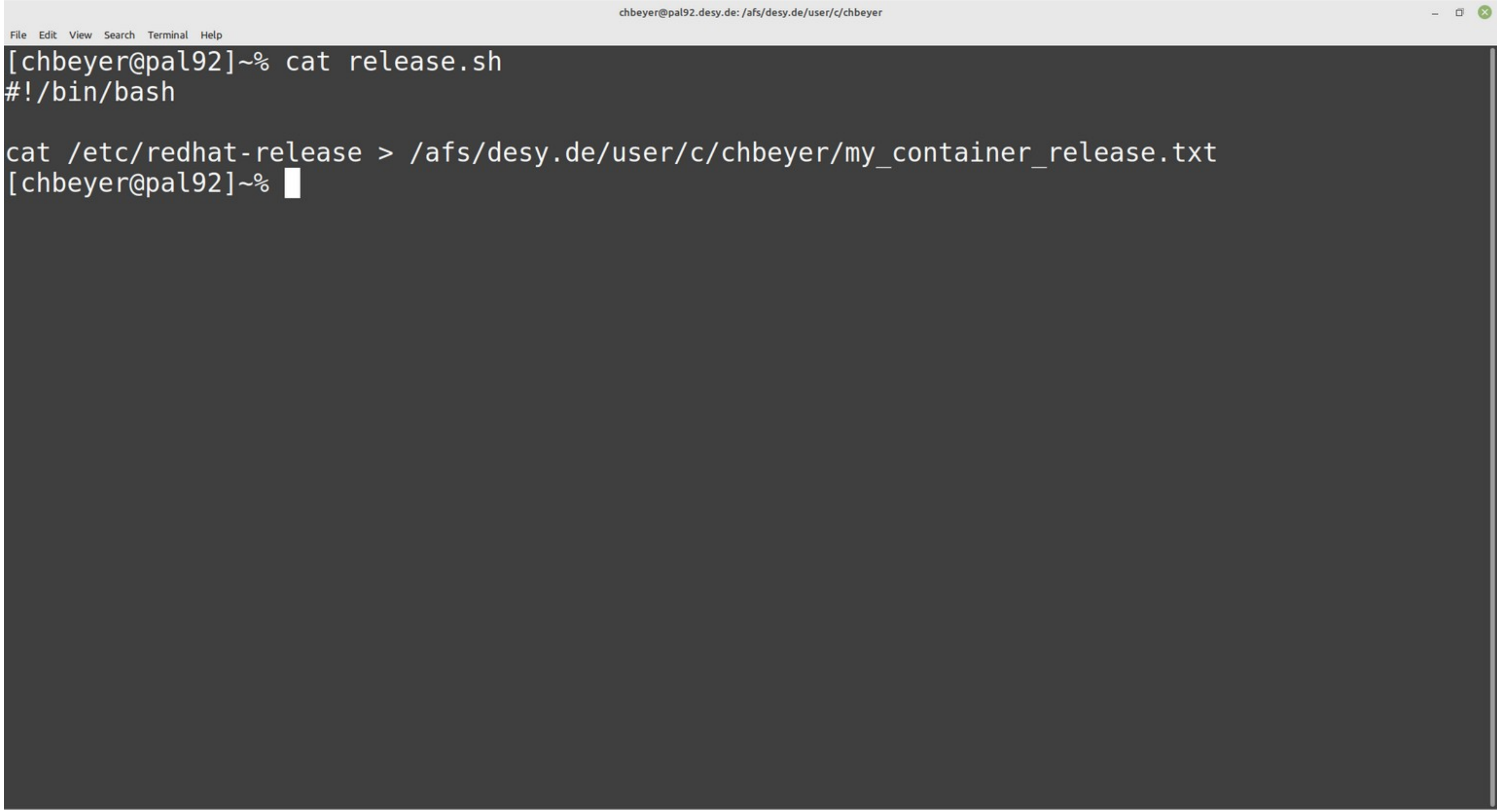
That was easy – what is all the fuzz about then

There is more to it if you take a closer look

- The container in the example is fairly simple in a way that it behaves just like any other executable
  - It writes to <STDOUT> which gets captured by HTCondor and written into the outfile location that's the only way we can write into the shared filesystem here as we did not bind-mount any of the usual worker mounts into the container
- This example will also not scale, once 1.000 jobs run the container at the same time from AFS there will be delays and problems even with a small container like this
- Nothing from inside this container could read or write into a shared filesystems e.g. DUST/AFS etc
- Let's look at a more common example that comes with some more container specific setup features
  - Use a container located in CVMFS
  - Execute a separate executable/script inside that container (no arguments needed)

Execute a mini script via condor inside an EL8 container

The exectable reads the redhat-release file

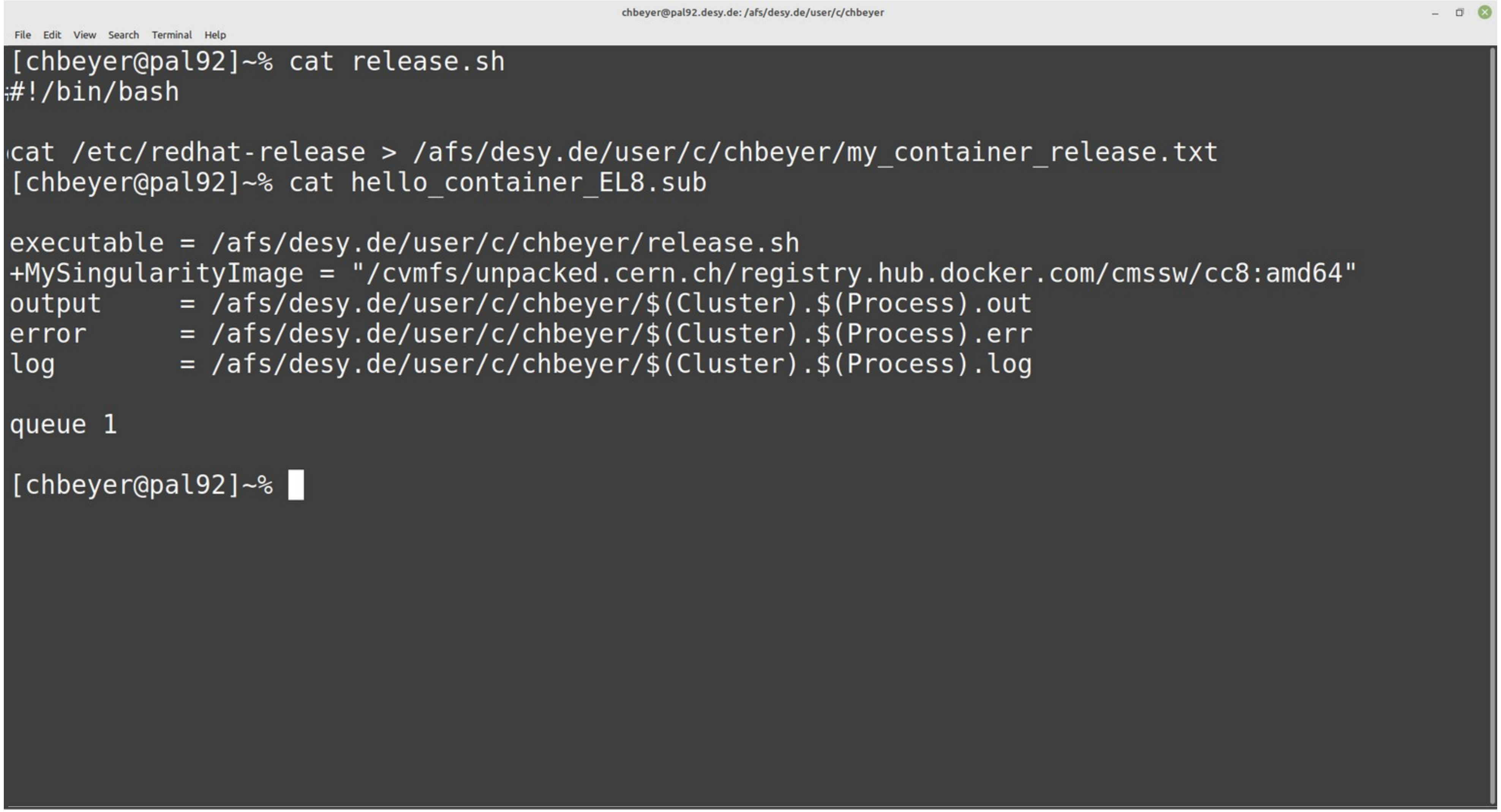


chbeyer@pal92.desy.de: /afs/desy.de/user/c/chbeyer
File Edit View Search Terminal Help

```
[chbeyer@pal92]~% cat release.sh
#!/bin/bash

cat /etc/redhat-release > /afs/desy.de/user/c/chbeyer/my_container_release.txt
[chbeyer@pal92]~%
```

Execute a mini script via condor inside an EL8 container

The submit file

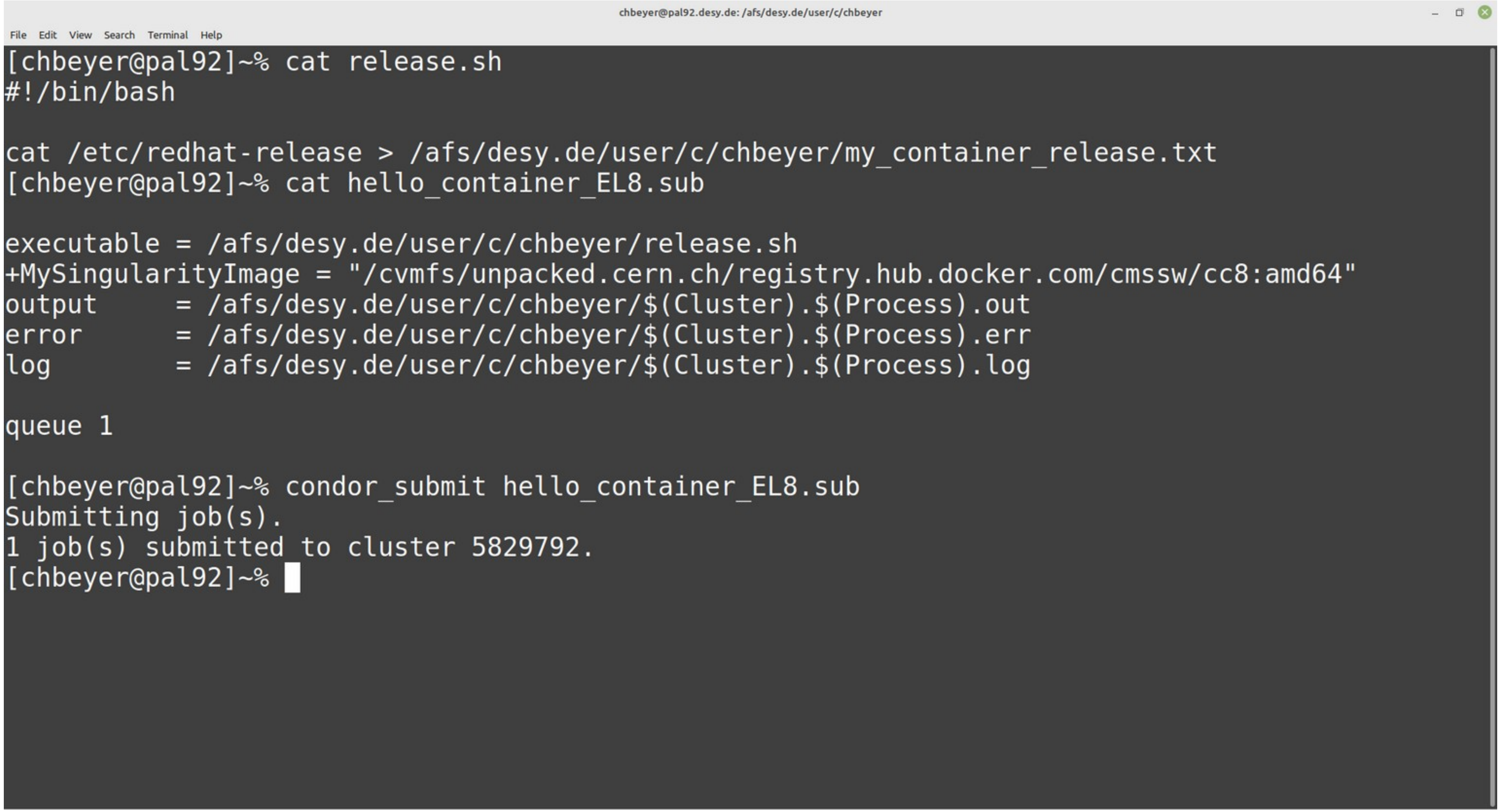


chbeyer@pal92.desy.de: /afs/desy.de/user/c/chbeyer
File Edit View Search Terminal Help

```
[chbeyer@pal92]~% cat release.sh
#!/bin/bash

cat /etc/redhat-release > /afs/desy.de/user/c/chbeyer/my_container_release.txt
[chbeyer@pal92]~% cat hello_container_EL8.sub

executable = /afs/desy.de/user/c/chbeyer/release.sh
+MySingularityImage = "/cvmfs/unpacked.cern.ch/registry.hub.docker.com/cmssw/cc8:amd64"
output     = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).out
error      = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).err
log        = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).log

queue 1

[chbeyer@pal92]~%
```

Execute a mini script via condor inside an EL8 container

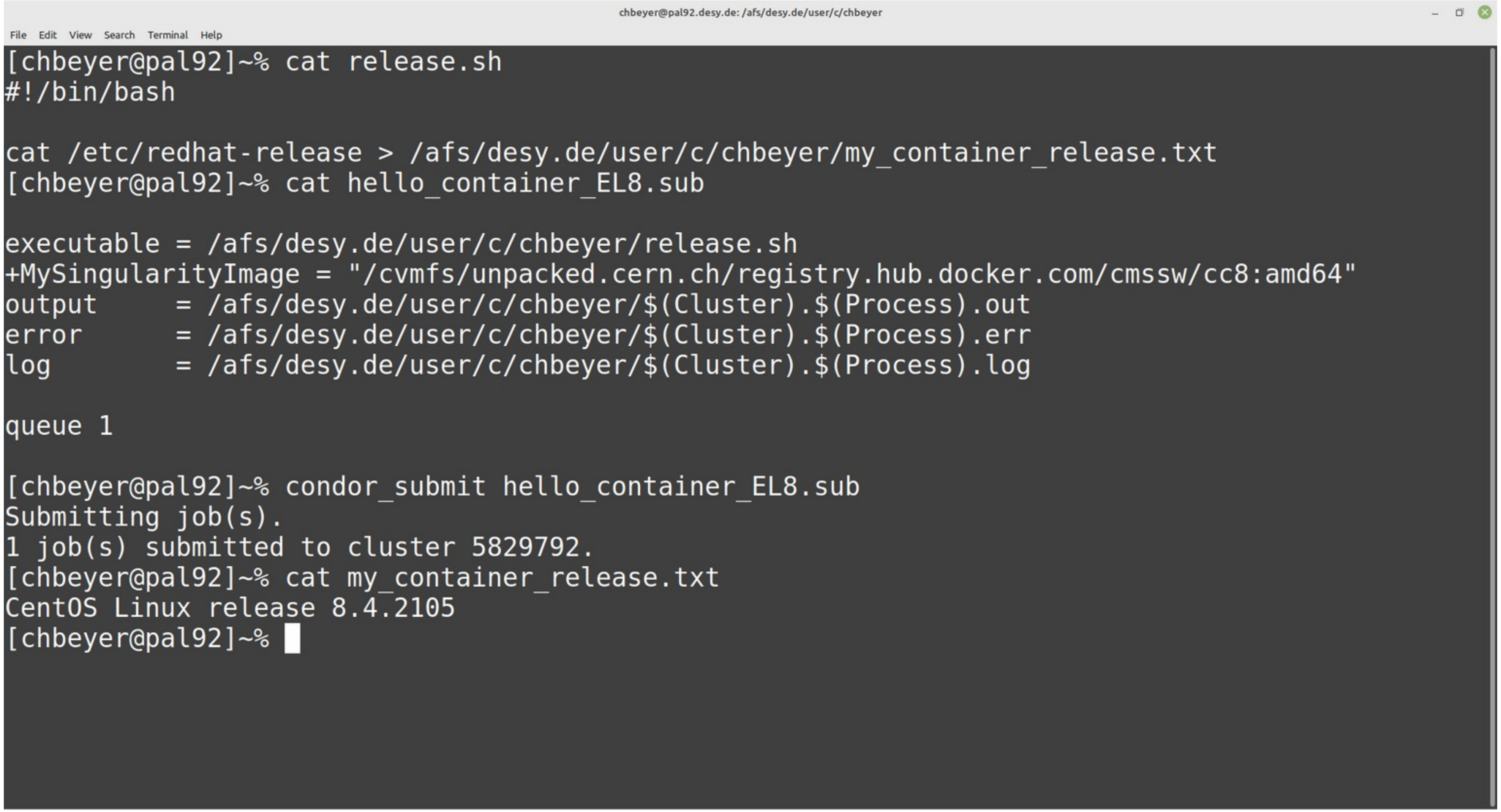
Job submit


chbeyer@pal92.desy.de: /afs/desy.de/user/c/chbeyer
File Edit View Search Terminal Help

```
[chbeyer@pal92]~% cat release.sh
#!/bin/bash

cat /etc/redhat-release > /afs/desy.de/user/c/chbeyer/my_container_release.txt
[chbeyer@pal92]~% cat hello_container_EL8.sub

executable = /afs/desy.de/user/c/chbeyer/release.sh
+MySingularityImage = "/cvmfs/unpacked.cern.ch/registry.hub.docker.com/cmssw/cc8:amd64"
output     = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).out
error      = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).err
log        = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).log

queue 1

[chbeyer@pal92]~% condor_submit hello_container_EL8.sub
Submitting job(s).
1 job(s) submitted to cluster 5829792.
[chbeyer@pal92]~% 
```

Execute a mini script via condor inside an EL8 container

Output written into the file



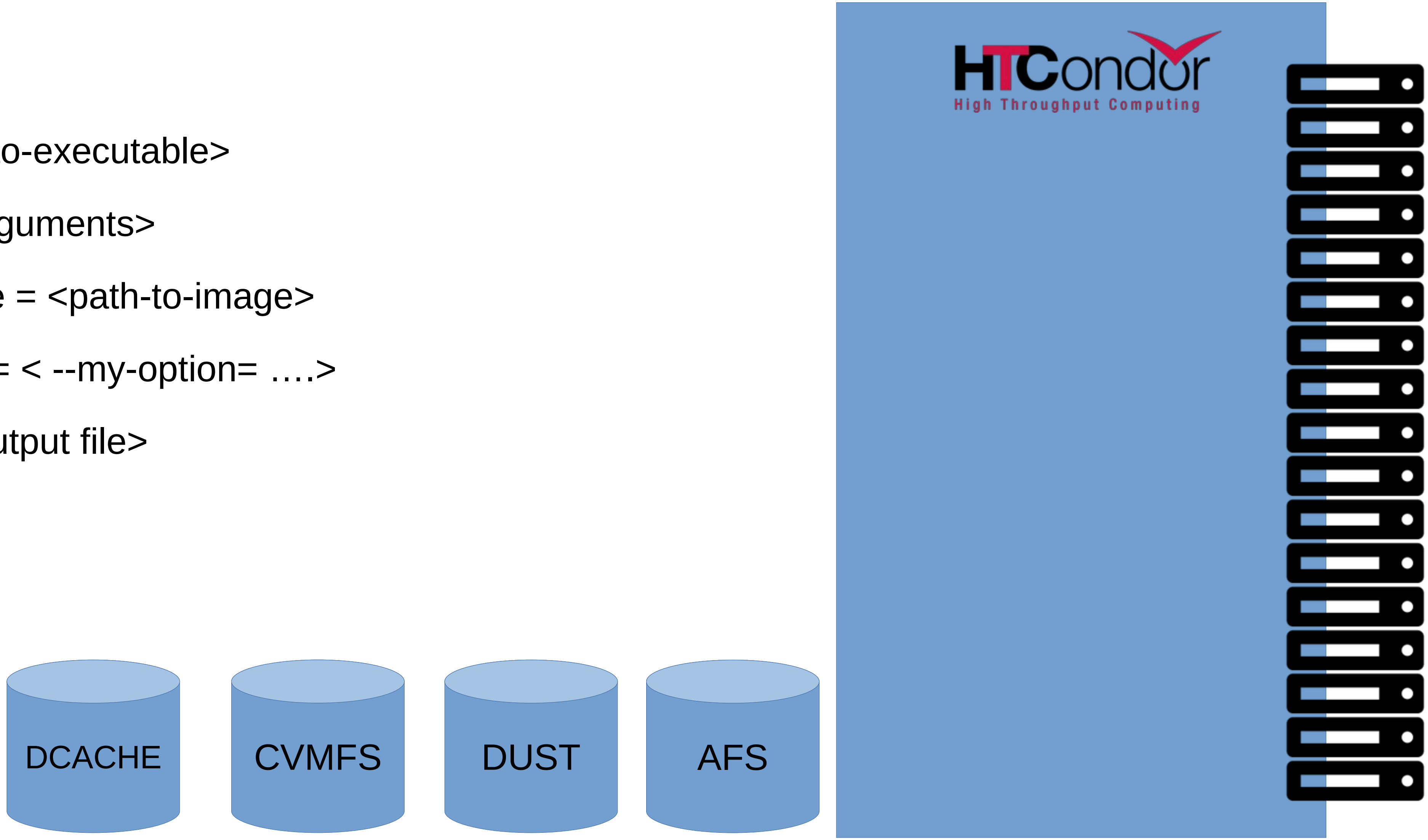
```
[chbeyer@pal92]~% cat release.sh
#!/bin/bash

cat /etc/redhat-release > /afs/desy.de/user/c/chbeyer/my_container_release.txt
[chbeyer@pal92]~% cat hello_container_EL8.sub

executable = /afs/desy.de/user/c/chbeyer/release.sh
+MySingularityImage = "/cvmfs/unpacked.cern.ch/registry.hub.docker.com/cmssw/cc8:amd64"
output     = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).out
error      = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).err
log        = /afs/desy.de/user/c/chbeyer/$(Cluster).$(Process).log

queue 1

[chbeyer@pal92]~% condor_submit hello_container_EL8.sub
Submitting job(s).
1 job(s) submitted to cluster 5829792.
[chbeyer@pal92]~% cat my_container_release.txt
CentOS Linux release 8.4.2105
[chbeyer@pal92]~%
```

Overview of a container job run

On the workernode

Job submit file

- Executable = <path-to-executable>
- Arguments = <my arguments>
- +MySingularityImage = <path-to-image>
- +MySingularityArgs = < --my-option= ….>
- Output = <path to output file>

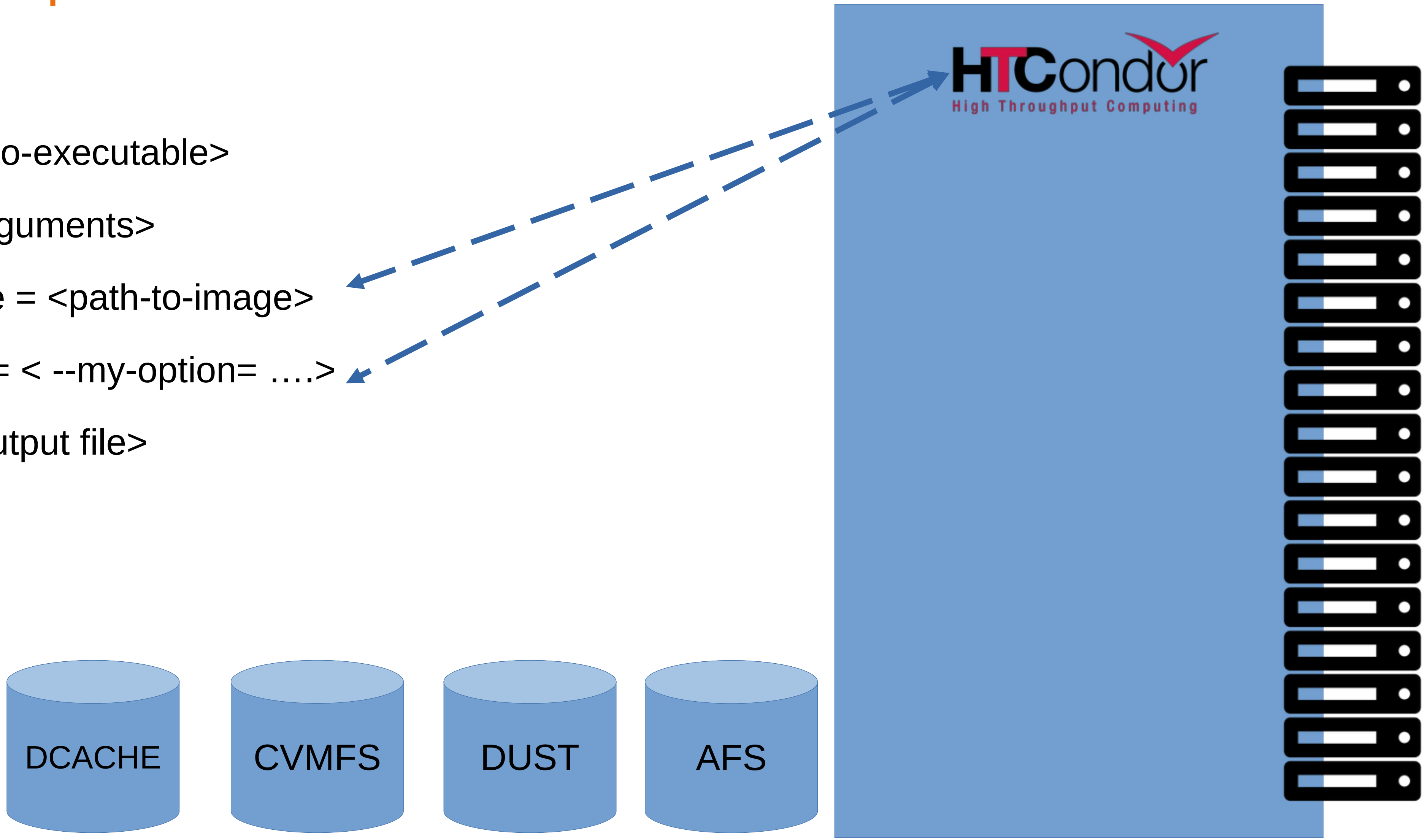


Overview of a container job run

HTC evaluates container part in submit file

Job submit file

- Executable = <path-to-executable>
- Arguments = <my arguments>
- +MySingularityImage = <path-to-image>
- +MySingularityArgs = < --my-option= ….>
- Output = <path to output file>

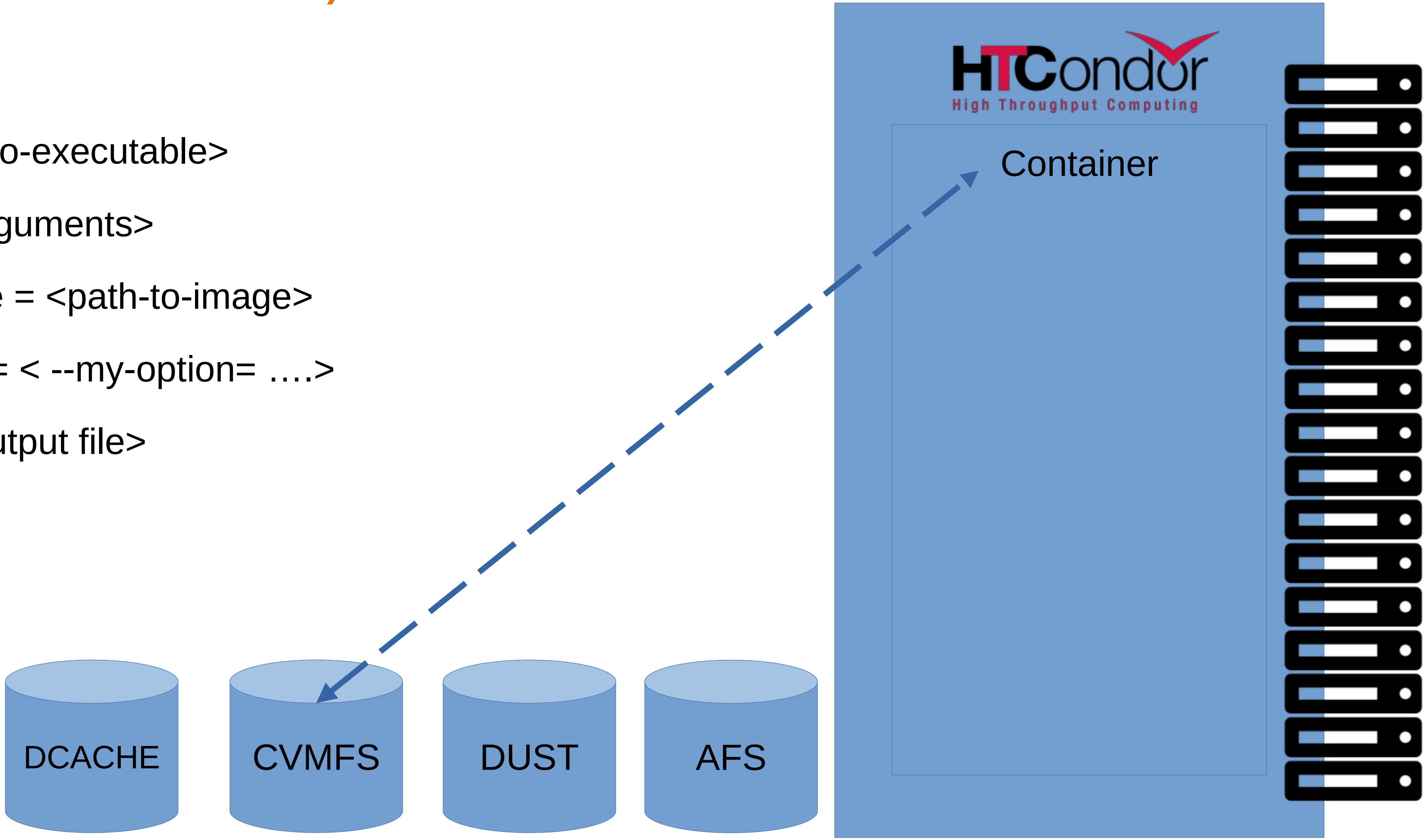


Overview of a container job run

Runs container (in this case from CVMFS)

Job submit file

- Executable = <path-to-executable>
- Arguments = <my arguments>
- +MySingularityImage = <path-to-image>
- +MySingularityArgs = < --my-option= ….>
- Output = <path to output file>

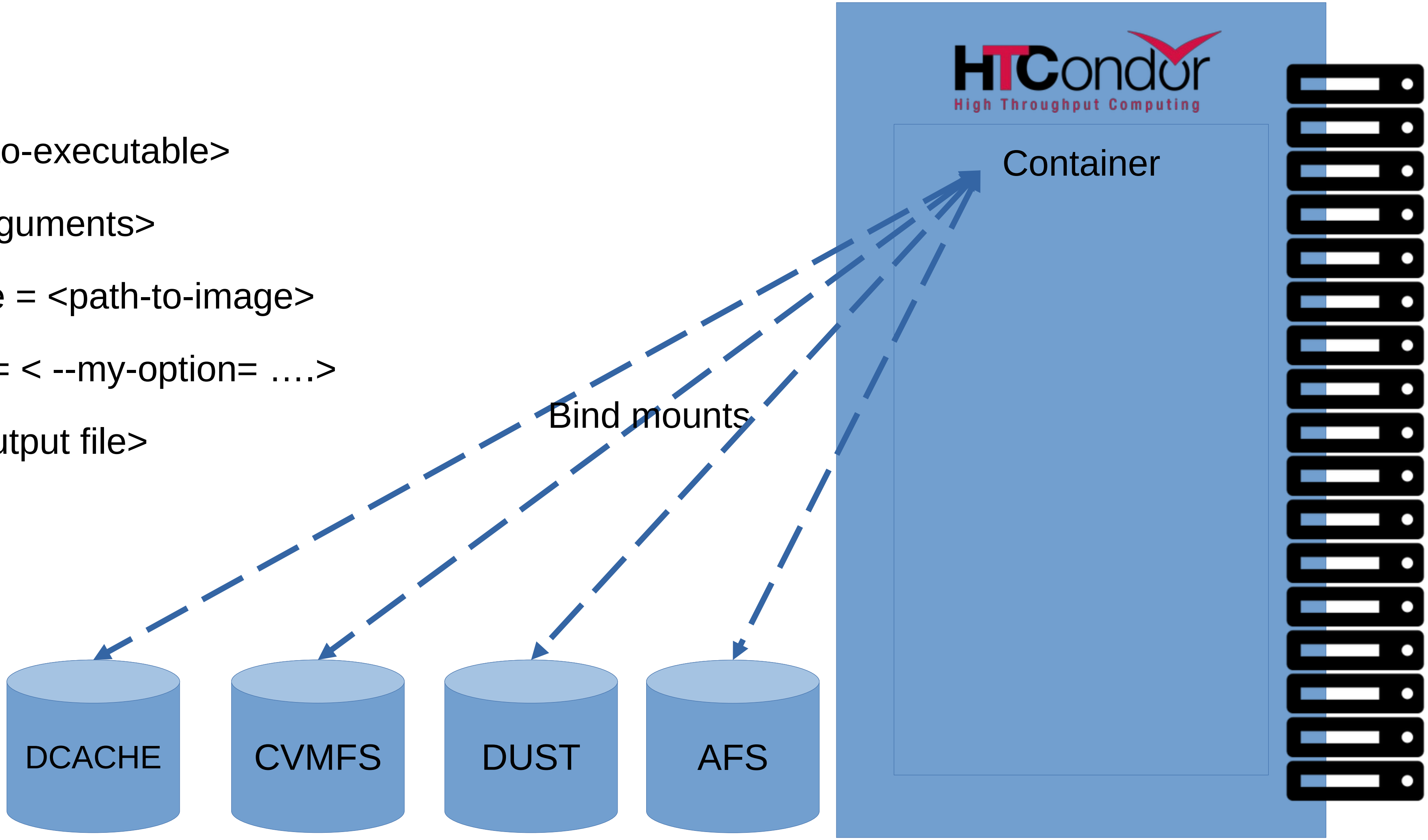


Overview of a container job run

Bind-mounts all ususal data sources

Job submit file

- Executable = <path-to-executable>
- Arguments = <my arguments>
- +MySingularityImage = <path-to-image>
- +MySingularityArgs = < --my-option= ….>
- Output = <path to output file>

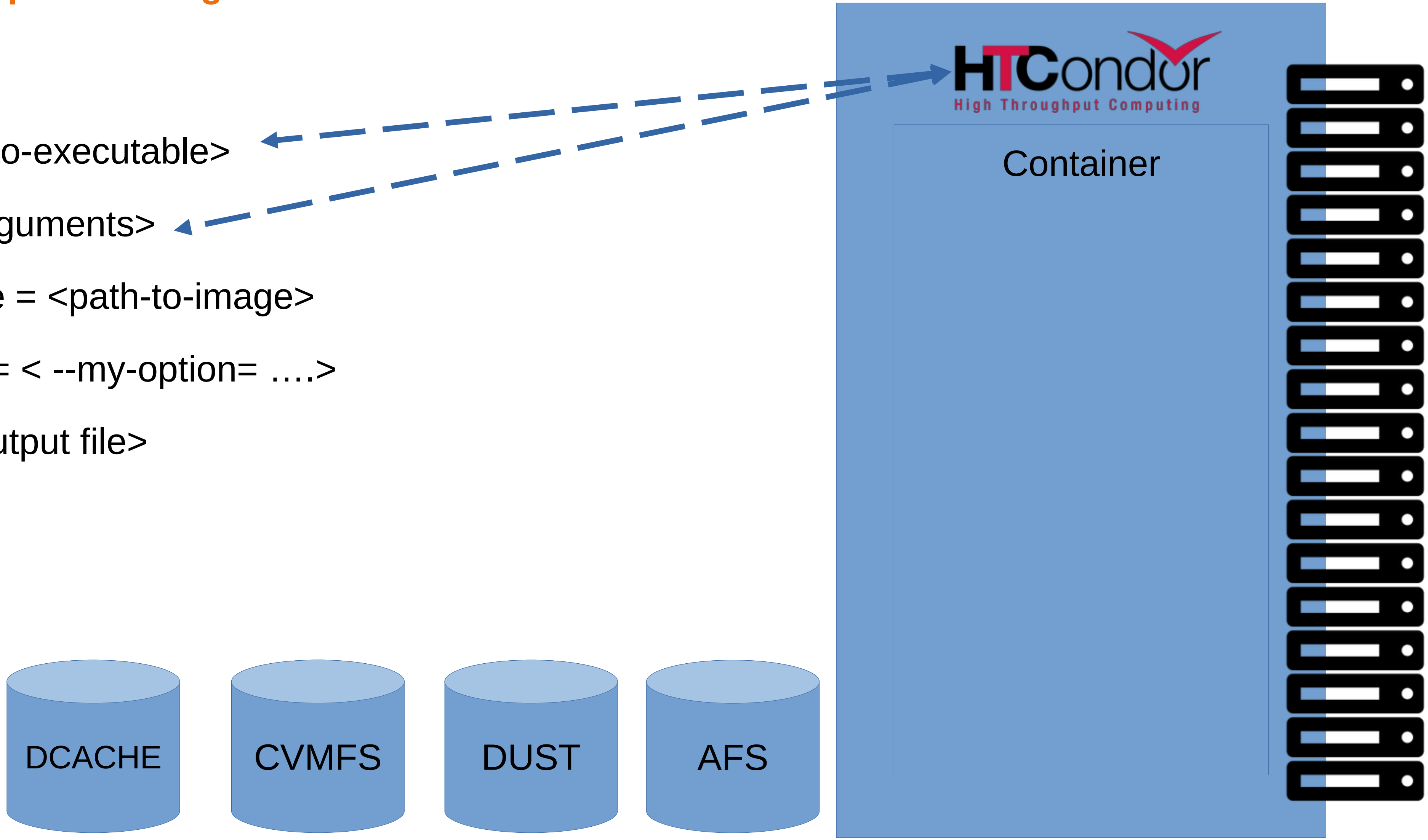


Overview of a container job run

Starts executable with possible args inside the container

Job submit file

- Executable = <path-to-executable>
- Arguments = <my arguments>
- +MySingularityImage = <path-to-image>
- +MySingularityArgs = < --my-option= ….>
- Output = <path to output file>

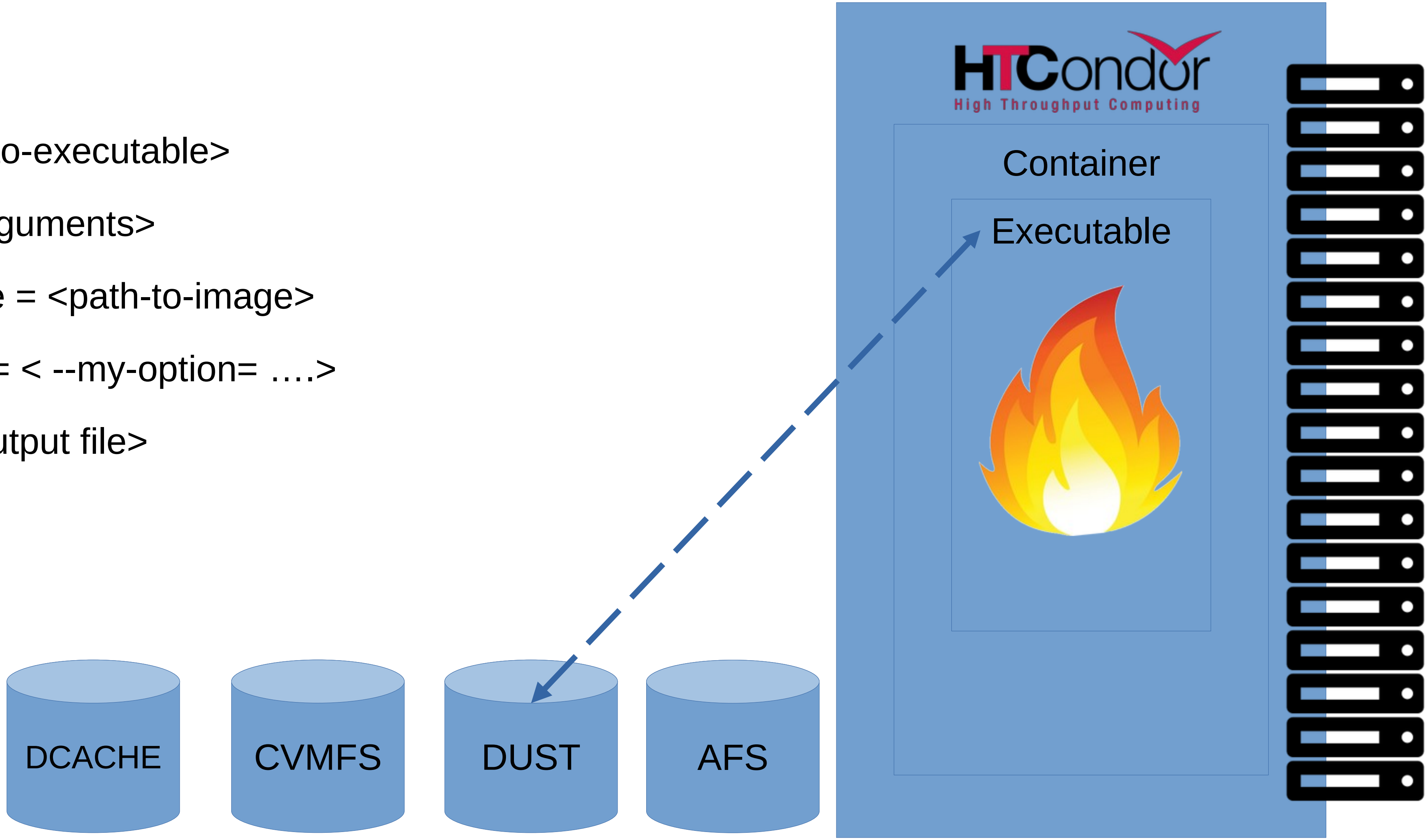


Overview of a container job run

Executable is located in DUST

Job submit file

- Executable = <path-to-executable>
- Arguments = <my arguments>
- +MySingularityImage = <path-to-image>
- +MySingularityArgs = < --my-option= ….>
- Output = <path to output file>

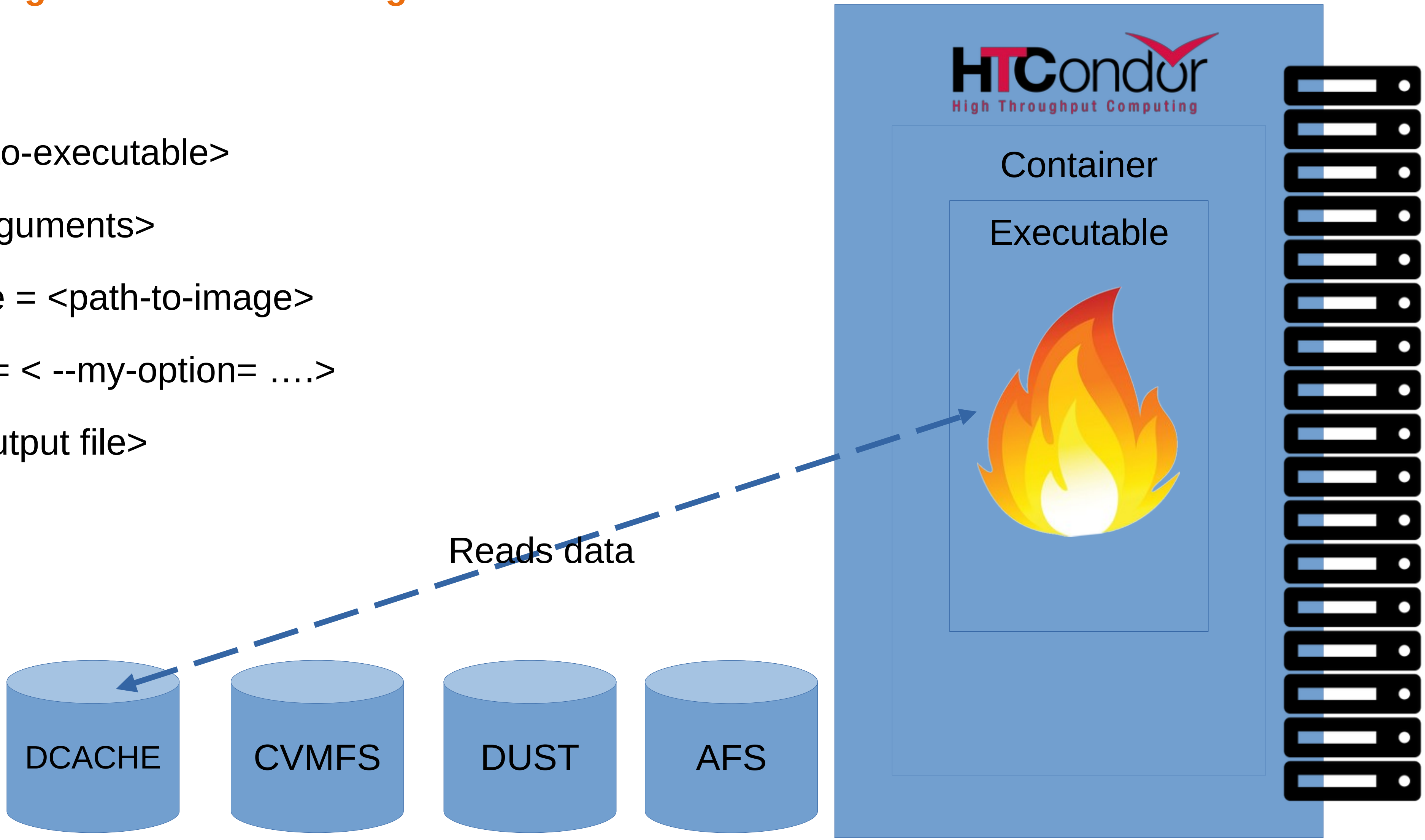


Overview of a container job run

Executable does it’s magic and reads data e.g. from DCACHE

Job submit file

- Executable = <path-to-executable>
- Arguments = <my arguments>
- +MySingularityImage = <path-to-image>
- +MySingularityArgs = < --my-option= ….>
- Output = <path to output file>

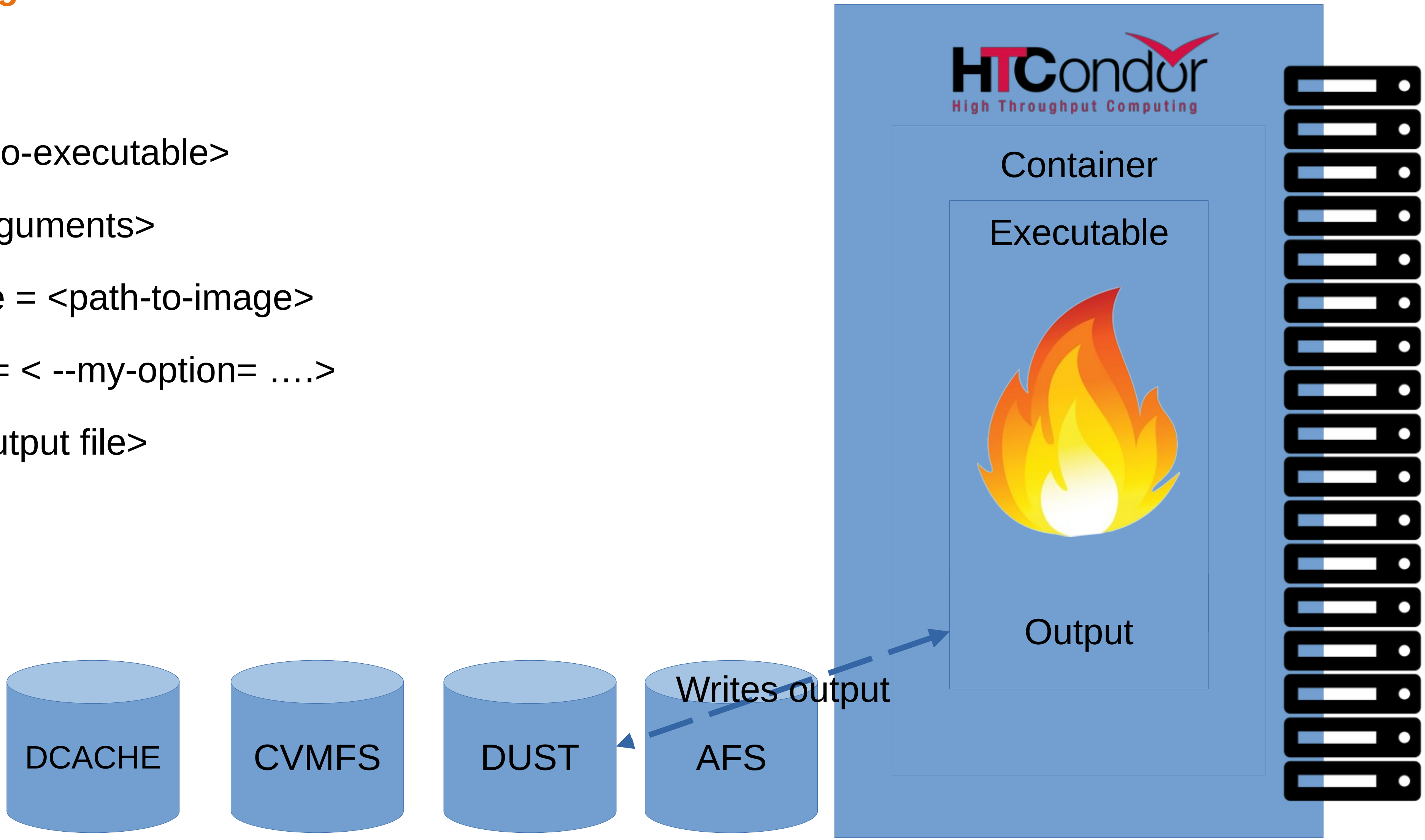


Overview of a container job run

Output get’s written e.g. to DUST

Job submit file

- Executable = <path-to-executable>
- Arguments = <my arguments>
- +MySingularityImage = <path-to-image>
- +MySingularityArgs = < --my-option= ….>
- Output = <path to output file>

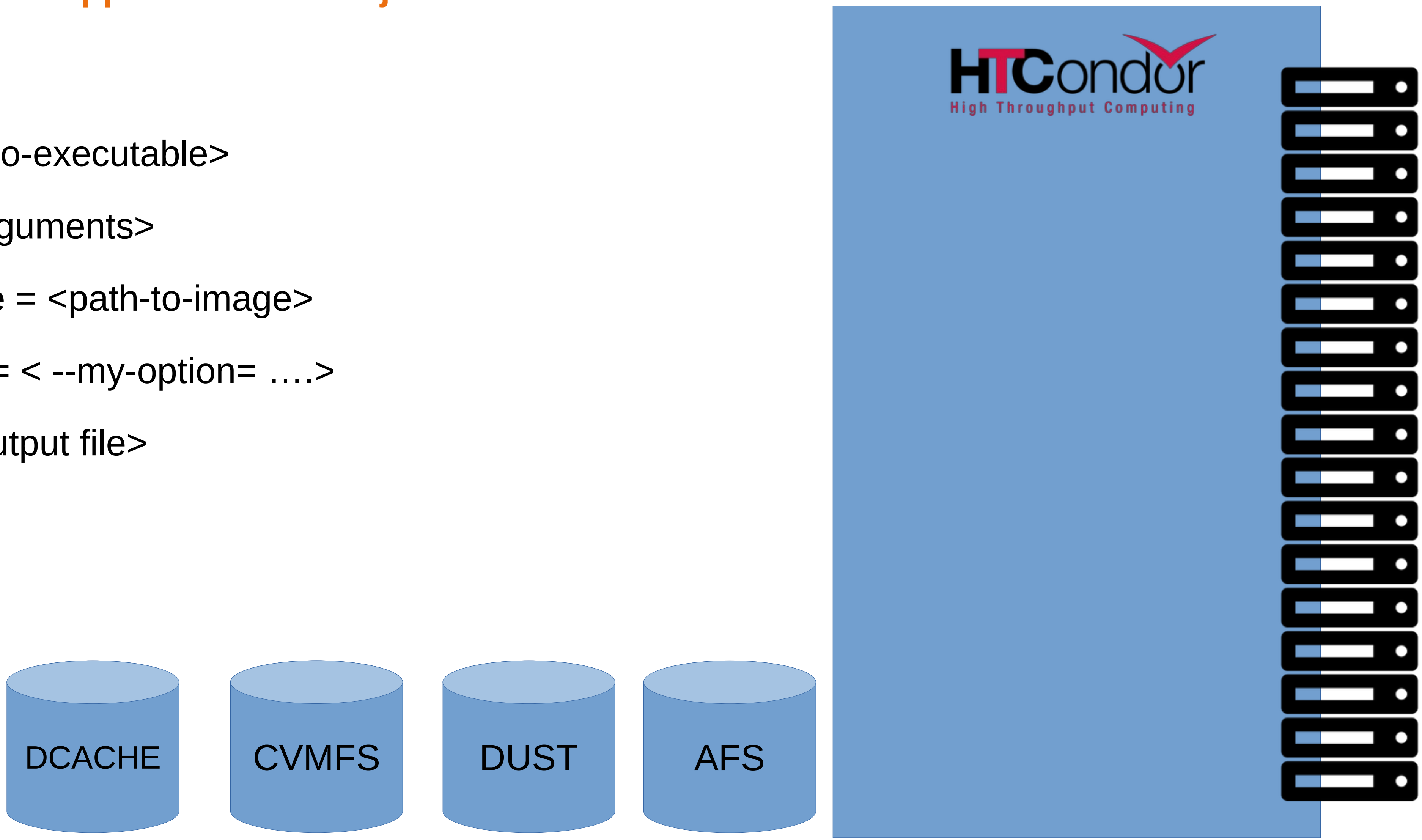


Overview of a container job run

Executable & container stopped with end of job

Job submit file

- Executable = <path-to-executable>
- Arguments = <my arguments>
- +MySingularityImage = <path-to-image>
- +MySingularityArgs = < --my-option= ….>
- Output = <path to output file>



Summary & best practice

Running jobs in containers is fairly simple

- Containers technically can be pulled at job runtime from different types of container registries which is OK for testing purposes e.g. but will not scale in production
- Containers can be staged with the job for test purposes (staging is limited to 100MB) will not scale either
- A variety of containers is available in CVMFS and if possible the usage of those is the most elegant way to profit from containers
  - Only the parts of the container image used by the job will get loaded which is very resource effective compared to staging the complete image
- If you use custom containers put them on DUST !
- Use the syntax in the submit file (differs from the official one):
  - +MySingularityImage = <path to your singularity image>
  - Optional: +MySingularityArgs = "--my-args"
  - Bind mounts will be automatically added to your startup
  - Use ‘executable = <...> ’ and ‘Arguments = <...>‘ as in usual job submits
- See here for more information: https://wiki.desy.de/naf/documentation/apptainer-support-bird/

Thank you

(Questions ?)