



THE NEW
ADVENTURES
OF
SCRUN WITH
OCI CONTAINERS

Nate Rini
SchedMD

Copyright 2022 SchedMD LLC
<https://schedmd.com>



Slurm User Group Meeting 2022

Copyright 2022 SchedMD LLC
<https://schedmd.com>

Agenda - US Mountain Time (UTC-6)



Time	Speaker	Title
9:00 - 9:50	Jason Booth	Field Notes 6: From The Frontlines of Slurm Support
10:00 - 10:20	Ole Nielsen (DTU)	Pathfinding into the clouds
10:30 - 10:55	Nate Rini	OCI Containers, and scrun
11:00 - 11:20	Wei Feinstein (LBNL)	LBNL Site Report
11:30 - 11:55	Nick Ihli	Cloudy, With A Chance of Dynamic Nodes
12:00 - 12:20	Kota Tsuyuzaki (NTT)	Burst Buffer Lua Plugin for Lustre
12:30 - 12:55	Tim Wickberg	Slurm 22.05, 23.02, and Beyond



Welcome

- Seven separate presentations, seven separate streams
- Presentations are available through the SchedMD Slurm YouTube channel
 - <https://youtube.com/c/schedmdslurm>
- Or through direct links from the agenda
 - https://slurm.schedmd.com/slurm_ug_agenda.html

Asking questions



- Feel free to ask questions throughout through YouTube's chat
- Chat is moderated by SchedMD staff
 - Tim McMullan, Ben Roberts, and Tim Wickberg
 - Also identified by the little wrench symbol next to their name
- For SchedMD presentations:
 - Questions will be relayed to the presenter by the moderators
 - Some may be deferred to the end if they cannot be relayed in a timely fashion
 - Or some may be answered by the moderators in chat directly
- For community presentations:
 - Please ask questions in the live chat
 - The presenter (if available) may respond through chat
 - Or SchedMD staff may try to answer in their absence



THE NEW
ADVENTURES
OF
SCRUN WITH
OCI CONTAINERS

Nate Rini
SchedMD

Copyright 2022 SchedMD LLC
<https://schedmd.com>


Contents




- HPC & containers
- OCI container support
- Slurm's OCI runtime proxy - scrun
- Container staging with Lua plugin in scrun
- Rootless Docker support
- Podman support
- OCI containers via podman in Scaleout
- Questions
- Container limitations in Slurm
- Questions

Slurm Container Support

Technical Preview

 This functionality has been added as a technical preview as we continue to work out all the features and site requirements. RFE tickets are always welcome.

 Functionality and interfaces may change dramatically between releases.

- Container is an ambiguous term. This is specifically Open Container Initiative (OCI) containers which follow the set of standards here:
 - <https://github.com/opencontainers>
- Slurm container documentation:
 - <https://slurm.schedmd.com/containers.html>
- Note: job_container/tmpfs is currently incompatible with the OCI Container functionality

Users ♥ Docker

- What Docker [[docker.com](https://www.docker.com)] is to users as described by Stanford:

<https://www.sherlock.stanford.edu/docs/software/using/singularity/>

Containers are a solution to the problem of how to get software to run reliably when moved from one computing environment to another. They also resolve installation problems by packaging all the dependencies of an application within a self-sustainable image, a.k.a a container.

Docker has long been the reference and the most popular container framework in DevOps and Enterprise IT environments

- <https://www.section.io/engineering-education/why-is-docker-so-popular/>
- <https://www.jobsity.com/blog/8-reasons-why-docker-matter-for-devs>
- https://developers.redhat.com/blog/2020/11/19/transitioning-from-docker-to-podman#transition_to_the_podman_cli

HPC Docker

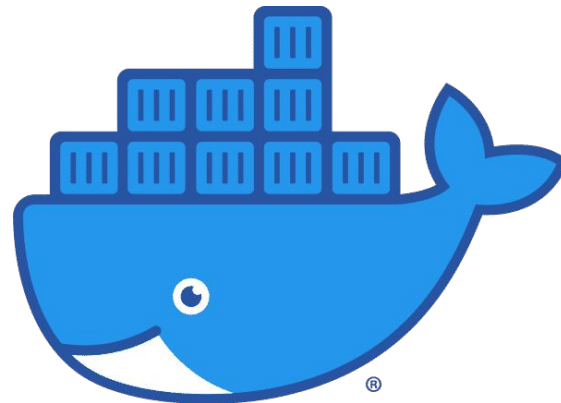


- The issues between Docker and HPC as described by Stanford:

<https://www.sherlock.stanford.edu/docs/software/using/singularity/>

- Docker requires a daemon running as root on all of the compute nodes, which has serious security implications,
- all authenticated actions (such as login, push ...) are also executed as root, meaning that multiple users can't use those functions on the same node,
- Docker uses cgroups to isolate containers, as does the Slurm scheduler, which uses cgroups to allocate resources to jobs and enforce limits. Those uses are unfortunately conflicting.
- but most importantly, *allowing users to run Docker containers will give them root privileges* inside that container, which will in turn let them access any of the clusters' filesystems as root. This opens the door to user impersonation, inappropriate file tampering or stealing, and is obviously not something that can be allowed on a shared resource.

Why not infuse Docker with Slurm?



Adding support for Docker in Slurm



Steps:

1. Slurm needs to be able to:
 - a. Run OCI Containers
 - b. Schedule jobs with containers
 - c. Track containers resources
 - d. Enforce all job rules and limits upon containers
2. Docker needs way to interface with Slurm:
 - a. Docker uses OCI Runtime to run containers
 - b. Slurm needs an OCI Runtime interface
 - c. Container images must be reliably sent to and from compute nodes

Slurm OCI Container Support



- Added '--container' (21.08) support to the following:
 - srun
 - salloc
 - sbatch
- Added viewing job container [bundle path] (21.08) and container-id (23.02) to the following:
 - scontrol show jobs
 - scontrol show steps
 - sacct
 - If passed as part of the '--format' argument using "Container"
 - slurmd, slurmstepd, slurmdbd & slurmctld logs (too many places to list)

OCI Container Support (21.08+)

- Slurm cgroups features apply to the OCI containers
 - All processes should be cleaned up even if the container anchor process dies or processes attempt to become daemons and detach from the session
 - Resource usage can be hard limited and monitored
- Slurm is only going to support unprivileged containers in 21.08, 22.05, 23.02
 - Use existing kernel support for containers
 - Users can already call all of these commands directly
 - Containers must be able to function in an existing host network
- Per host configuration via 'oci.conf' in /etc/slurm/
 - Environment variables SLURM_CONTAINER and SLURM_CONTAINER_ID (23.02) will always be set with a value (if present).

OCI Container Support (21.08+)

srun example

```
$ srun --container=/tmp/centos grep ^NAME /etc/os-release  
NAME="CentOS Linux"
```

salloc example

```
$ salloc --container=/tmp/centos grep ^NAME /etc/os-release  
salloc: Granted job allocation 65  
NAME="CentOS Linux"  
salloc: Relinquishing job allocation 65
```

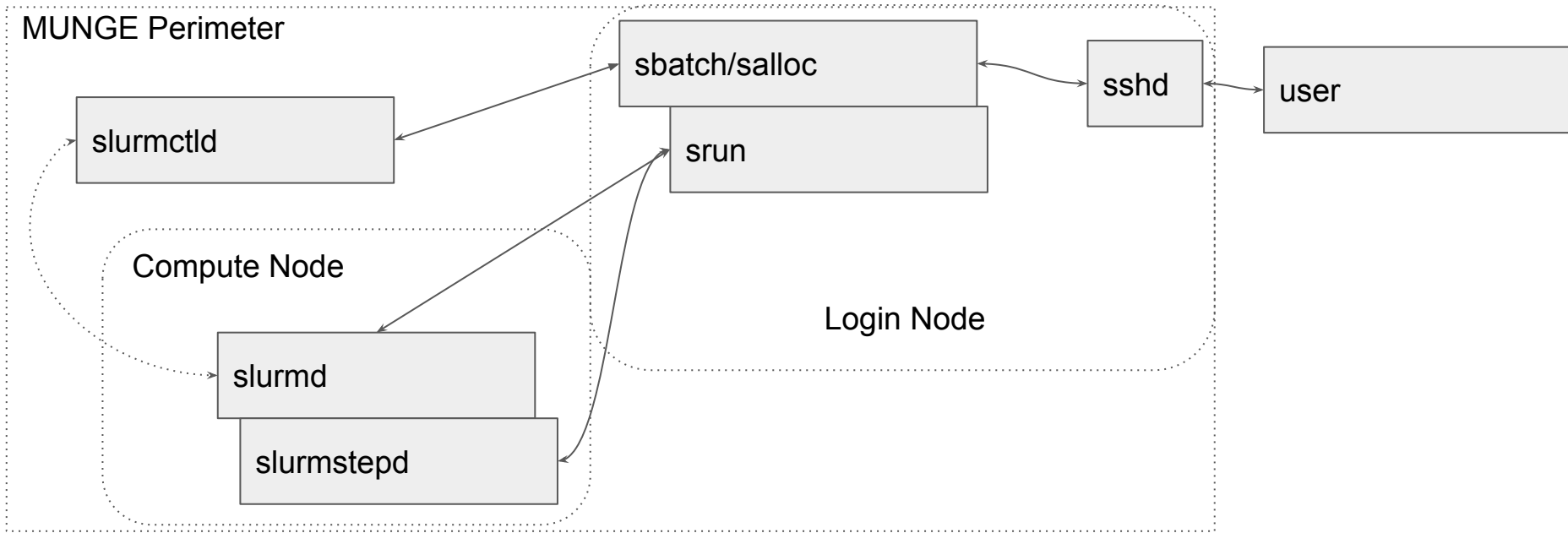
Note: containers have limited permissions and can result in pseudo terminal warnings.

OCI Container Support (21.08+)

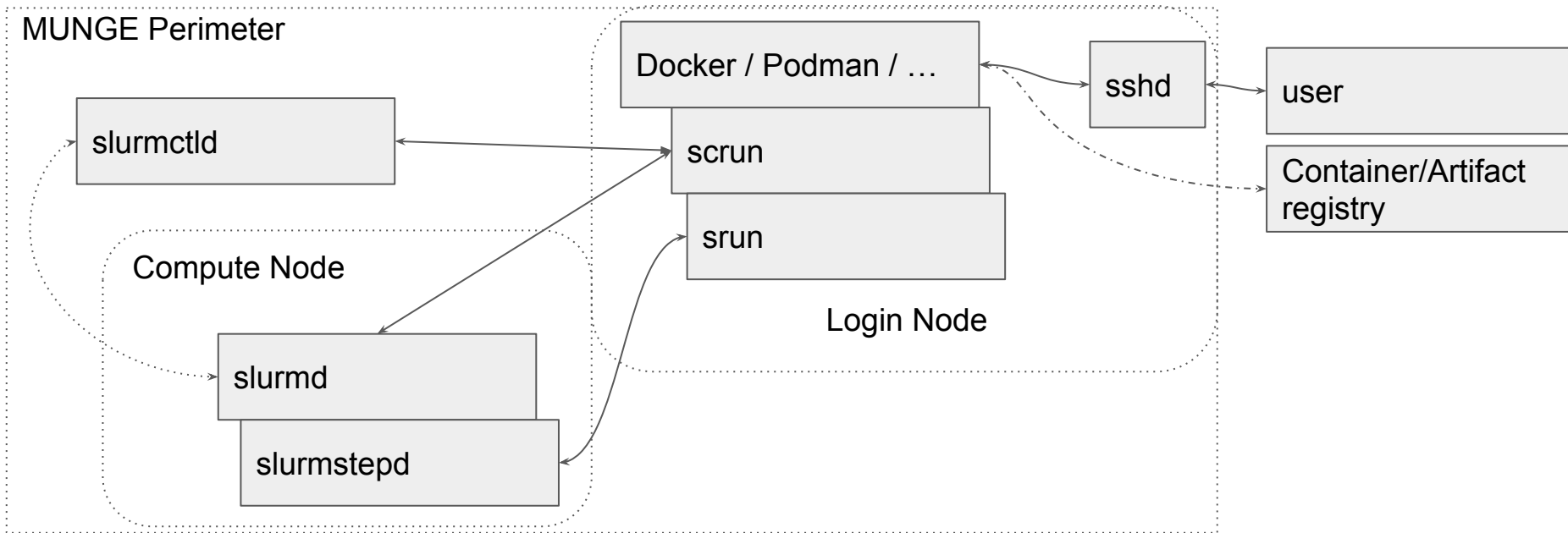
sbatch example

```
$ sbatch --container=/tmp/centos --wrap 'grep ^NAME  
/etc/os-release'  
Submitted batch job 24419  
$ cat slurm-24419.out  
NAME="CentOS Linux"
```


Batch Job Use Case (23.02)



Container Use Case (23.02)



OCI runtime proxy - scrun (23.02)

- scrun's goal is to make containers **boring for users**
 - Users have better things to do than learn about the intricacies of containers
 - Site administrators will have to do setup and maintenance on the configuration
- Use Slurm's existing infrastructure to run containers on compute nodes
- Automatic staging out and in of containers controlled by system administrators
 - End requirement that users manually prepare their images on compute nodes.
- Interface directly with OCI runtime clients (Docker or Podman or ...)

OCI runtime proxy - scrun (23.02)

- Allow users to work with the tools they want while running work on the Slurm cluster
- scrun is a new CLI command to join srun, sbatch and salloc, but no user should ever have to call it directly or even really need to be aware of it
- scrun is still very new and we welcome tickets with requests for enhancements and especially bug reports

scrun via rootless Docker (23.02)



example:

```
$ export  
DOCKER_HOST=unix://$XDG_RUNTIME_DIR/docker.sock  
$ export DOCKER_SECURITY="--security-opt label:disable  
--security-opt seccomp=unconfined --security-opt  
apparmor=unconfined --net=none"  
$ docker run $DOCKER_SECURITY -i ubuntu /bin/sh -c 'grep  
^NAME /etc/os-release'  
NAME="Ubuntu"  
$ docker run $DOCKER_SECURITY -i centos /bin/sh -c 'grep  
^NAME /etc/os-release'  
NAME="CentOS Linux"
```

Rootless Docker notes



- Rootless Docker must be fully installed and working before activating scrum configuration
- Sites should always use the latest version of Docker and rootless Docker
 - Moby Project is actively working on rootless Docker
 - Rootless Docker is currently classified as experimental
- Install procedure:
 - Host must first pass ``dockerd-rootless-setupool.sh -f check`` test
 - <https://docs.docker.com/engine/security/rootless/>
- Using user systemd to control rootless docker is recommended on login nodes but not compute nodes due to lingering daemons taking up resources

Rootless Docker config (23.02)

~/.config/docker/daemon.json

```
{
  "default-runtime": "slurm",
  "runtimes": {
    "slurm": {
      "path":
"/usr/local/slurm/sbin/scrun"
    }
  },
```

```
"experimental": true,
"iptables": false,
"bridge": "none",
"no-new-privileges": true,
"rootless": true,
"selinux-enabled": false
}
```

scrunch via rootless Podman (23.02)



example:

```
$ podman run ubuntu /bin/sh -c 'grep ^NAME /etc/os-release'  
NAME="Ubuntu"
```

```
$ podman run centos /bin/sh -c 'grep ^NAME /etc/os-release'  
NAME="CentOS Linux"
```

```
$ podman run centos /bin/sh -c 'printenv SLURM_JOB_ID'  
77
```

```
$ podman run centos /bin/sh -c 'printenv SLURM_JOB_ID'  
78
```


Podman notes

- Podman must be fully installed and working before activating scrum configuration.
- Running latest stable version of Podman is suggested.

Podman config for scrun (23.02)



~/config/containers/containers.conf:

```
[containers]
apparmor_profile = "unconfined"
cgroupns = "host"
cgroups = "enabled"
default_sysctls = []
label = false
netns = "host"
no_hosts = true
pidns = "host"
utsns = "host"
usersns = "host"
```

```
[engine]
runtime = "slurm"
runtime_supports_nocgroup
s = [ "slurm" ]
runtime_supports_json = [
"slurm" ]
remote = false

[engine.runtimes]
slurm = [
c  "/usr/local/slurm/sbin/scrun"
```

scrun - container staging



- scrun needs to stage out the image to remote host at startup
- scrun needs to stage in the image from remote host at job end
- Flexibility required as every site has a different shared file system configuration and data ingress and egress rules.
 - scrun avoids making as many assumptions about the request host vs the execution host in Slurm itself as possible.
 - Site admins must configure where and how images are staged.

scrun - container staging via Lua



- scrun's Lua staging plugin allows site to write custom and simple scripts to move the image to and back from the remote storage.
- scrun's staging lua script is located at:
 - `/etc/slurm/staging.lua`
- Lua script runs as user avoiding any additional privilege escalation risk
- Lua already has JSON support via libraries
- Sites can write a native Slurm plugin if desired instead of using the Lua plugin.

scrunch - Lua container stage in example

Simplified stage in (to compute node) hook:

```
function slurm_stage_in_allocator(id, bundle, spool_path,
config_path)
    os.execute(string.format( "/usr/bin/env rsync --numeric-ids
--delete-after --ignore-errors -a -- %s/ %s/", rootfs, dstfs))

    slurm.set_bundle_path(p)
    slurm.set_root_path(p.."rootfs")

    write_file(jc, json.encode(c))
    return slurm.SUCCESS
end
```

scrurn - Lua container stage out example

Simplified stage out (from compute node) hook: (this example only deletes the

```
function slurm_stage_out_allocator(id, bundle, spool_path, config_path)
  os.execute("rm --one-file-system --preserve-root=all -rf "..bundle)
  return slurm.SUCCESS
end
```

See Slurm's documentation for a full and functional example of the Lua script when slurm-23.02 is officially released.

Experimentation with Scaleout



- Scaleout has Podman fully configured for testing
 - Podman runs inside of Docker Compose with Slurm controlled containers on the compute hosts
 - Configured using separated nodes and filesystems to demonstrate scrun's capabilities
- Will be included with the Scaleout slurm-23.02 tagged release
- Download here: <https://gitlab.com/SchedMD/training/docker-scale-out>
 - Make sure to pick a branch: slurm-21.08 or slurm-20.11
 - Read the README
 - sysctl settings must be applied first!
 - latest release of docker and docker-compose is highly recommended

User Namespaces



- Admins will need to configure: `/etc/subuid` and `/etc/subgid`
- If subuid/subgid range is greater than 1
 - Jobs and processes may run as any of the sub user/group in the range.
 - Sites need to be aware of this and account for it in their accounting and ACLs.
- Avoiding remapping users is suggested for simple containers:
 - Docker/Podman argument: `--userns=host`
 - <https://docs.docker.com/engine/security/userns-remap/>
- MUNGE is not aware of User Namespaces but will know the real user id of the running processes. This can be any user in the subuid range.
 - subuid/subgid is not Slurm specific

Questions?



Will scrum be useful for your site?

What barrier could be stopping your site from switching to scrum?

What features could help your site in switching to scrum?

What barriers are sites experiencing with containers?

scrun - limitations (23.02)



- No network namespaces support
 - All containers must run under host network
- No cgroup/apparmor/selinux support via Docker/Podman
 - All the containers are executed remotely making the local system's security systems irrelevant to the container. Podman allows easy configuration disablement while Docker requires command line arguments
- No container annotation support implemented yet
- No automatic resource selections implemented yet
 - Use of Slurm environment variables allow job property control
 - scrun will currently run the default job with default resources requested
- Container failures may require examining slurmd logs and/or syslogs to determine root cause

scrunch - limitations (23.02)

- Lua must either be compiled with JSON support or the library must be installed.
 - Slurm may need to be compiled after the JSON library is installed in Lua in order to be able to use it.
- scrunch will not currently kill or stop the lua script while it is running.
 - If the Lua staging scripts hang, then the job time limit may be triggered and kill the job.
- scrunch has the relevant SPANK and clifilter support.
 - These hooks are not a security device and any user may override them same as srun/sbatch/salloc.
 - scrunch uses standard Slurm RPCs and user permissions. Any user may modify or ptrace their own processes. Any security must be applied at the controller.

scrun - limitations (23.02)

- One podman/docker instance per user per host
 - scrun does not provide information for jobs other than its own
 - Jobs will be visible via squeue/sacct/slurmrestd
 - docker / podman will be blind to any externally started containers
- MUNGE Authentication
 - scrun currently only works via MUNGE
 - Job submission host must have Slurm installed and be in MUNGE perimeter
- JWT Authentication
 - Not currently implemented
- Container IDs must be unique per user
 - Docker or Podman will hand the container ID to scrun verbatim.
 - scrun will try to search for the container by ID

If the local anchor process is dead.

scrun - limitations (23.02)

- All existing limitations for running containers in Slurm still apply:
 - Containers must have a compatible version of Slurm installed to call Slurm commands
 - MUNGE's socket must be mounted in container to use MUNGE based authentication
 - JWT authentication is possible from container but there are no secrets functionality currently available.
 - Slurm does not support step controls/commands via JWT currently.
- User environment must be explicitly set
 - The environment at time of calling docker/podman will not be inherited by the container unless environment variables are supported by Docker/Podman.

scrun - limitations (23.02)



- scrun will create a local process that must remain alive for the duration of the Job
 - If the local process is killed, then the job will be killed by Slurm. This is the same requirement as any job run via srun
 - scrun can be started from a batch job to avoid submission host uptime requirements
- scrun supports output of Docker JSON formatted log files
 - All output to set to STDOUT instead of being split between STDOUT and STDERR
- Docker current uses an event and poll based system for determining if a container is alive
 - This may result in higher CPU usage on the submission host than only running a container directly via srun

scrun - limitations (23.02)



- scrun requires oci.conf to be fully configured
- I/O restrictions and other limitations from the submission host will affect staging containers in and out
- Slurm (scrun) is run as one of the last steps of starting the container in Docker/Podman
- Slurm has no control over Docker/Podman
 - Docker and podman will need to be configured independently of Slurm
 - Only rootless Docker/Podman is supported
 - rootless docker has varying levels of support with older kernels
 - Sites are recommended to run the latest version of their distro and docker to avoid issues
- Not all functionality of Docker/Podman is implemented

scrun - limitations (23.02)

- Online image repositories exist independently of Slurm and may apply bandwidth or usage restrictions
 - These limitations can falsely imply scrun (and Slurm) being slow
 - Sites are suggested to set up local caching proxies if possible
 - scrun does not cache images
- scrun is not a security solution or antivirus or a new security layer
 - It does not scan or reason about the contents of the container images beyond enforcing **basic** OCI image formatting
 - It will push the images to the execution hosts where the configured and the OCI runtime in oci.conf will be executed to start the containers
 - Users are responsible to ensure the container images are following site policies and procedures while being free of malicious code

scrunch - limitations (23.02)

- scrunch will only run under the POSIX user/group neither adding or removing abilities/capabilities/permissions from the user and therefore the container processes
- Sites must configure the stage in and stage out Lua scripts to clean up cached images
 - Failure to cleanup the images may result in massive wasteful usage of the filesystems.
- Sites must configure docker/podman to cleanup cached images independently of Slurm
 - Dockers build cache can get very large!



Questions?

Copyright 2022 SchedMD LLC
<https://schedmd.com>

End Of Stream



- Thanks for watching!