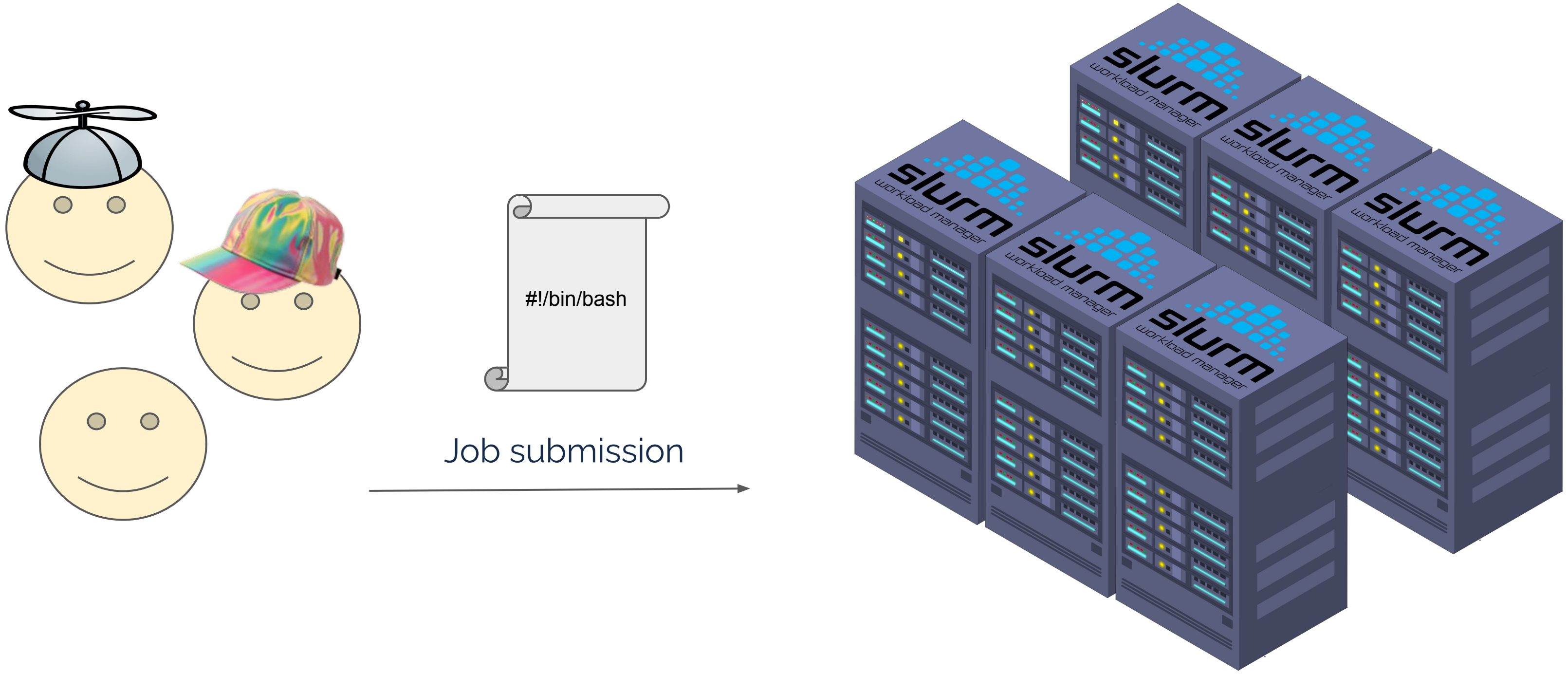
Simplifying the creation of Slurm client environments

A straw for your Slurm beverage

Pablo Llopis Sanmillán - FOSDEM 23

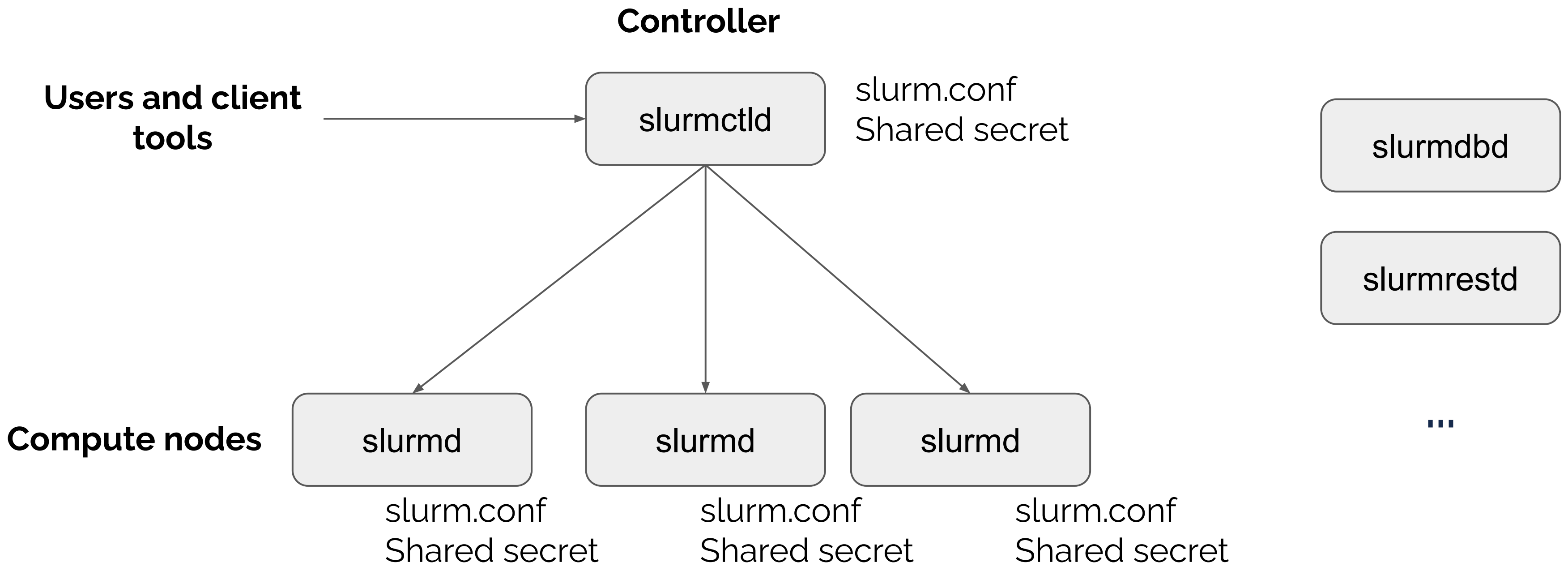
Slurm: a brief introduction

Slurm is both a Resource Manager, and a Job Scheduler



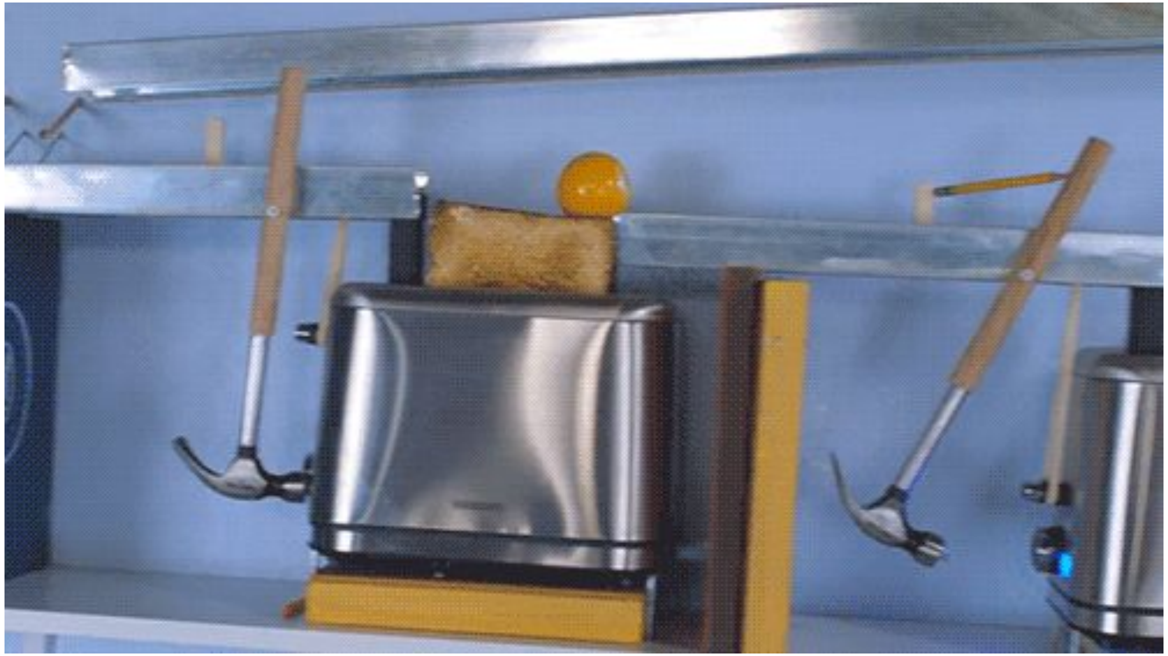
Slurm: a brief introduction

At its core, Slurm consists of a controller daemon, and client daemons



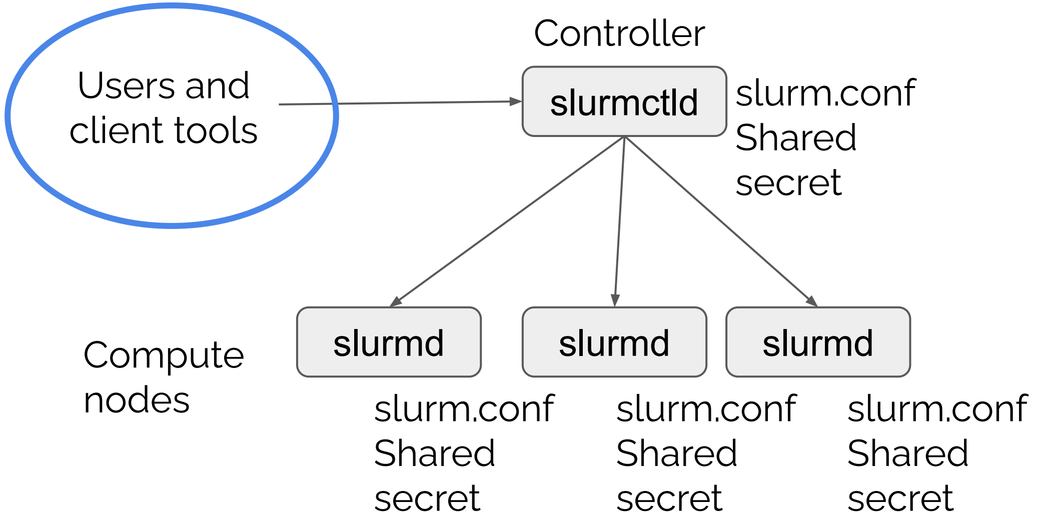
Containers

Containers are increasingly becoming a popular tool to run, automate deployments, and test modern infrastructure.



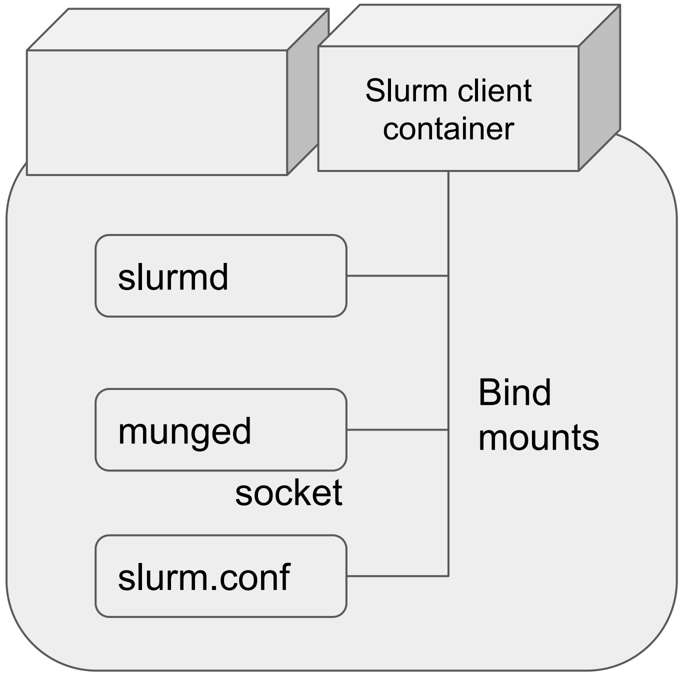
Containers + Slurm: use cases

- Monitoring
- Health checks
- Accounting
- Integration with other services

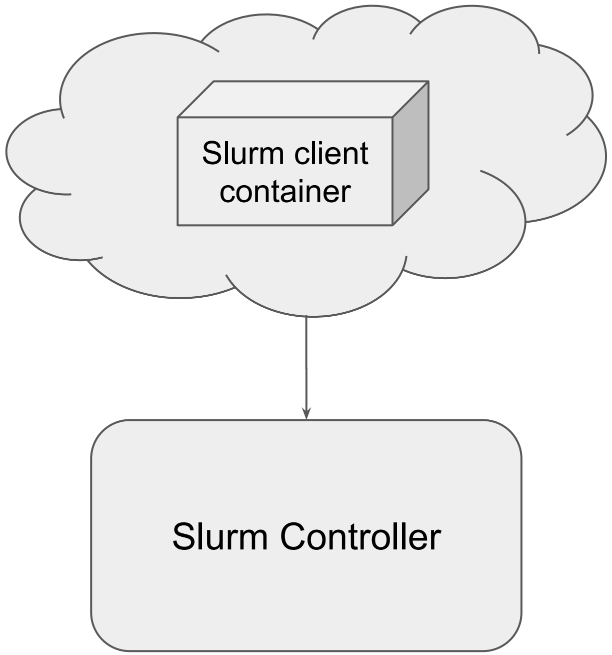


Containers + Slurm

The local use-case

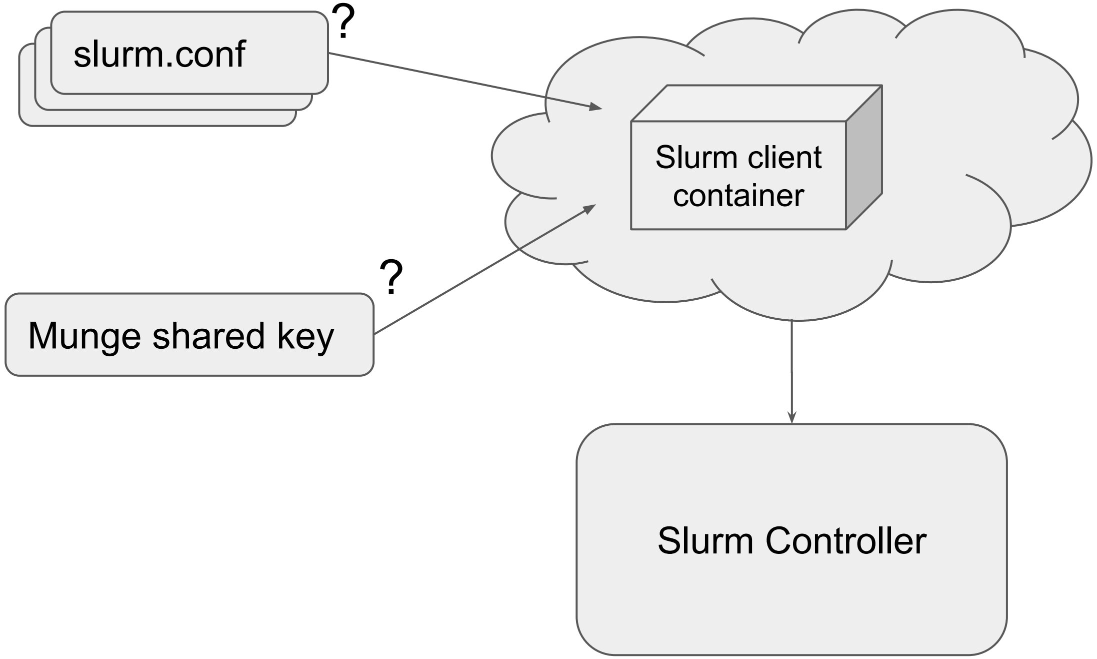


The distributed/remote use-case

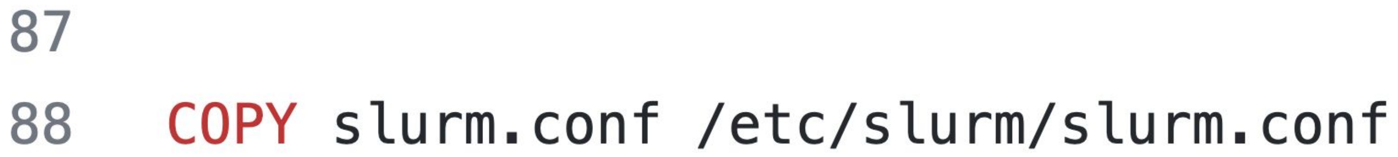


Containers + Slurm

The distributed/remote use-case



Containers + Slurm: the bad



```
COPY slurm.conf /etc/slurm/slurm.conf
```

This will absolutely work. But it’s often not necessarily good practice for maintainability.

Containers + Slurm: the bad

```

COPY slurm.conf /etc/slurm/slurm.conf
```

This will absolutely work. But it's often not necessarily good practice for maintainability.

Solution: Use Slurm's configless* feature! 🙂👍

*Since Slurm 20.02

Containers + Slurm: the ugly

```
# For some reason containers do not mount cgroups with file
# /sys/fs/cgroup/$subsystem/release_agent present.
# This file simply doesn't exist right after spawning the container (maybe due to container escape explo
# Slurm expects this file to exist. However, if we mount the cgroup subsystem manually
# after the container has already been spawned, the release_agent file will be there (??).
# We therefore umount them all, and then rely on Slurm's CgroupAutomount=yes to mount the cgroup subsyst
# ...
# Try containers, they said. It will be fun, they said.
umount /sys/fs/cgroup/freezer
umount /sys/fs/cgroup/cpuset
umount /sys/fs/cgroup/devices
umount /sys/fs/cgroup/cpuacct
umount /sys/fs/cgroup/memory
```

```
# More hacks needed for kubernetes:
# We will want to share the munge socket between containers.
# The main way to achieve this is to use the volume emptydir pattern in the pod,
# and let containers mount /var/run/munge via volumemounts.
# Munge checks and refuses to run if the directory has group write permissions enabled.
# But Kubernetes does not have a way to let us choose the directory mode and ownership.
# Therefore, make sure it is running in our deside mode and ownership.
# While we're at it, make sure we do not run into a "socket file already exists" in case
# the container is restarted independently from the pod.
# Isn't it nice when your infra is modern and declarative and doesn't need shell scripting everywher
rm -rf /var/run/munge/*
chmod 0755 /var/run/munge
chown munge:munge /var/run/munge
sudo -u munge /sbin/munged
```

```
# Another kubernetes-specific hack:
# Munge doesn't support secret as symlinks, and kubernetes forcibly
# presents secrets as symlinks. So we need to make a copy.
cp /secrets/munge.key /etc/munge/
```

Containers + Slurm: the ugly

Separate config files approach

- Manage a copy of slurm config files.

  (might be a challenge to keep a single, consistent source of truth)

- You will also need munged.

- And the munge key.

Configless approach

- Add slurmd into your container to benefit from configless.

- You will also need munged.

- And the munge key.

Containers + Slurm: the good?

A one-shot CLI tool that

- Authenticates to the controller (either munge or **JWT**)
- Fetches the Slurm config files

Containers + Slurm: the good?

A one-shot CLI tool that

- Authenticates to the controller (either munge or **JWT**)
- Fetches the Slurm config files

Straw: A tool to fetch Slurm config files 🥤

https://github.com/pllopis/straw

Straw in action

https://asciinema.org/a/e17m5iHhWM4MUaRD4fyrLAjmV

```
[pllopis@fedora straw]$ python straw.py -h
usage: straw.py [-h] [--auth {munge,jwt}] [-o OUTPUT_DIR] [-v] [-V]
[-l] server [server ...] version

positional arguments:
  server                slurmctld server in server[:port] notation
  version               Slurm major version that corresponds to that
of the slurmctld server (e.g. 22.05)

options:
  -h, --help            show this help message and exit
  --auth {munge,jwt}    Authentication method (default: jwt)
  -o OUTPUT_DIR, --output-dir OUTPUT_DIR
                        Existing output directory where config files
will be saved (default: ./)
  -v, --verbose         Increase output verbosity. Rrepetitions
allowed. (default: None)
  -V, --version         show program's version number and exit
  -l, --list            List available protocol versions (default:
False)
```

Straw in action

```
[pllopis@fedora straw]$ python straw.py -l
22.05
21.08
20.11
[pllopis@fedora straw]$ echo $SLURM_JWT
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MDE2NDI5NjgsImlhdCI6
MTY3MDEwNjk2OCwic3VuIjoicm9vdCJ9.mhesNN8venwBDgXQNolzdi__QQbmV8jYm2BV
lTRi47c
[pllopis@fedora straw]$ python straw.py --auth jwt localhost 22.05
[pllopis@fedora straw]$ python straw.py -v --auth jwt localhost 22.05
Using authentication method: jwt
Trying localhost:6817...
SlurmdSpoolDir=/var/spool/slurm/d
[pllopis@fedora straw]$
```

Conclusions

Conclusions

- Straw can simplify the **cost** of Slurm client container creation.
- Straw can increase the **security** of Slurm integrations with other services.

Caveats

- It would be even better if Straw didn't exist! Ideally this would be supported natively by Slurm.
- JWT tokens still need to belong to SlurmUser to be able to pull the config. (Slurm implementation limitation)

Try it out! https://github.com/pllopis/straw