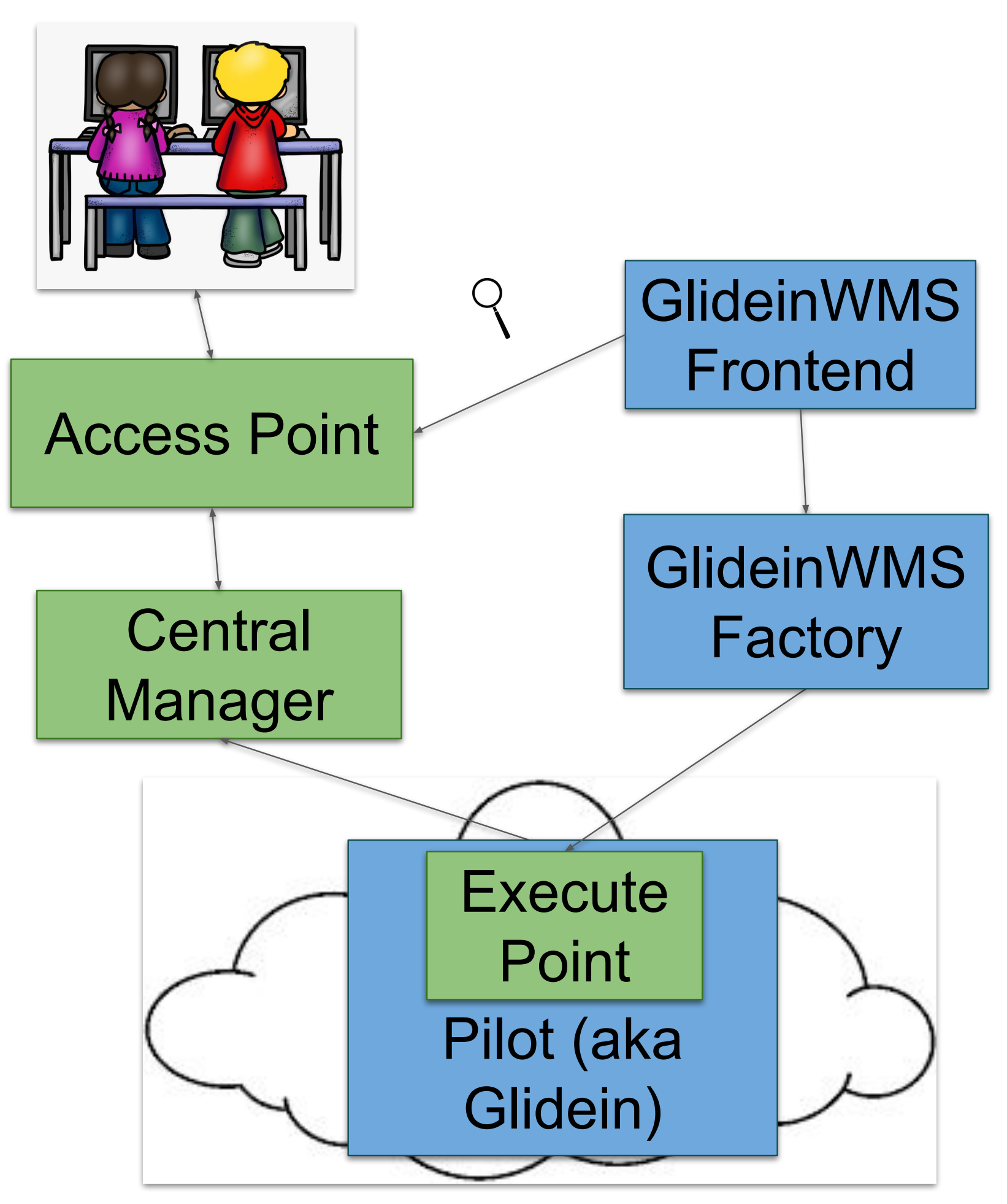
Exploiting Kubernetes to Simplify the Deployment and Management of the Multi-purpose CMS Pilot Job Factory

Jeffrey Michael Dost[1], Marco Mascheroni[1], Brian Paul Bockelman[2], Colby Walsworth[1], Edita Kizinevic[3], John Thiltges[4], Merina Albert[5], Vaiva Zokaite[6], Frank Würthwein[1]

[1]Univ. of California San Diego (US), [2]Morgridge Institute for Research, [3]CERN, [4]University of Nebraska Lincoln, [5]Fermi National Accelerator Lab, [6]Vilnius University

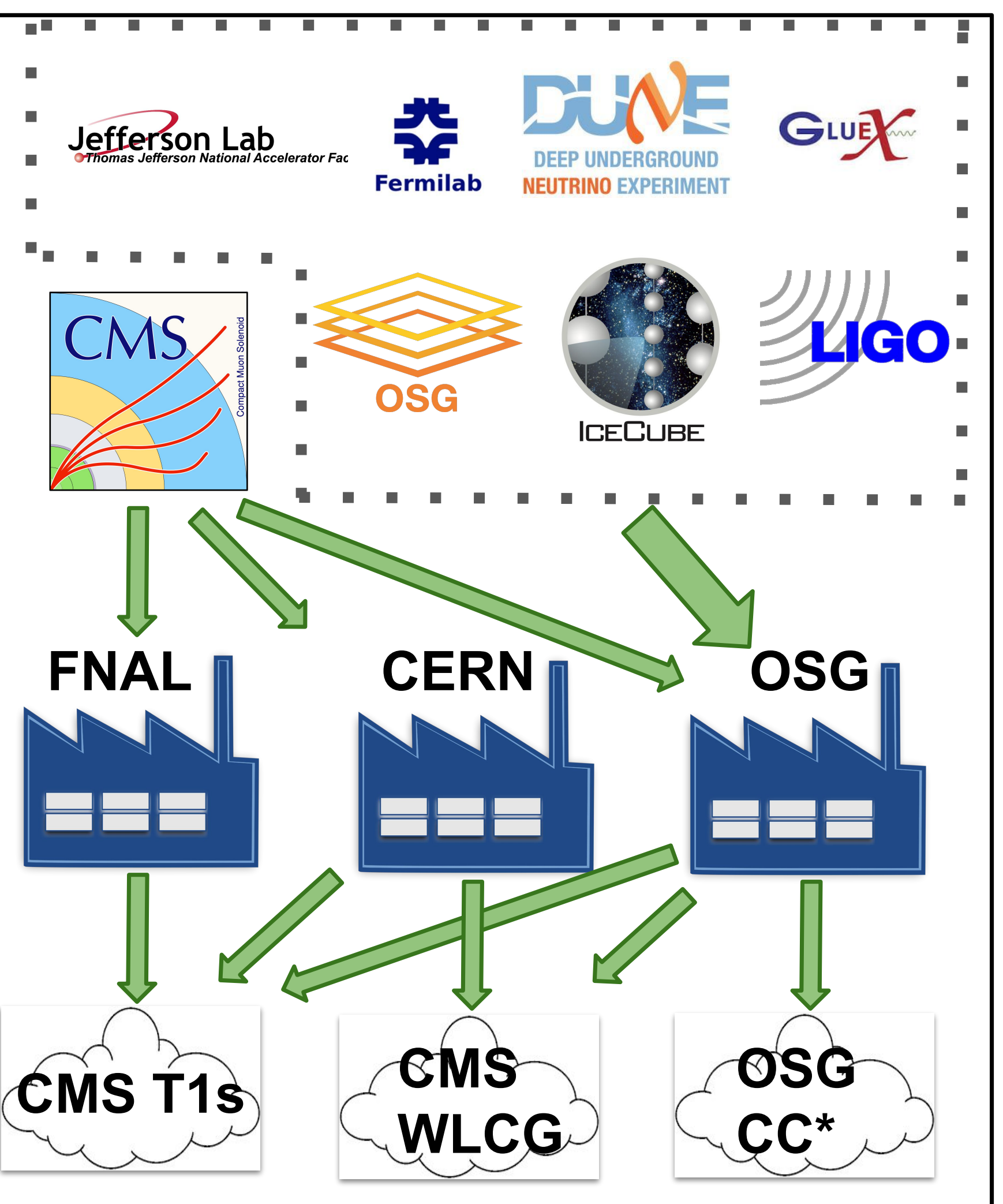
Provisioning resources

- GlideinWMS (Glidein Workflow Management System) is a pilot-based workload manager.
- Used in distributed computing, including the CMS experiment at CERN.
- Dynamically provisions execute points (glideins) across grid and cloud resources.
- Fetches jobs from a central queue for efficient execution.
- Simplifies workflow management and optimizes resource usage.
- Ensures scalable and flexible job execution in complex environments.



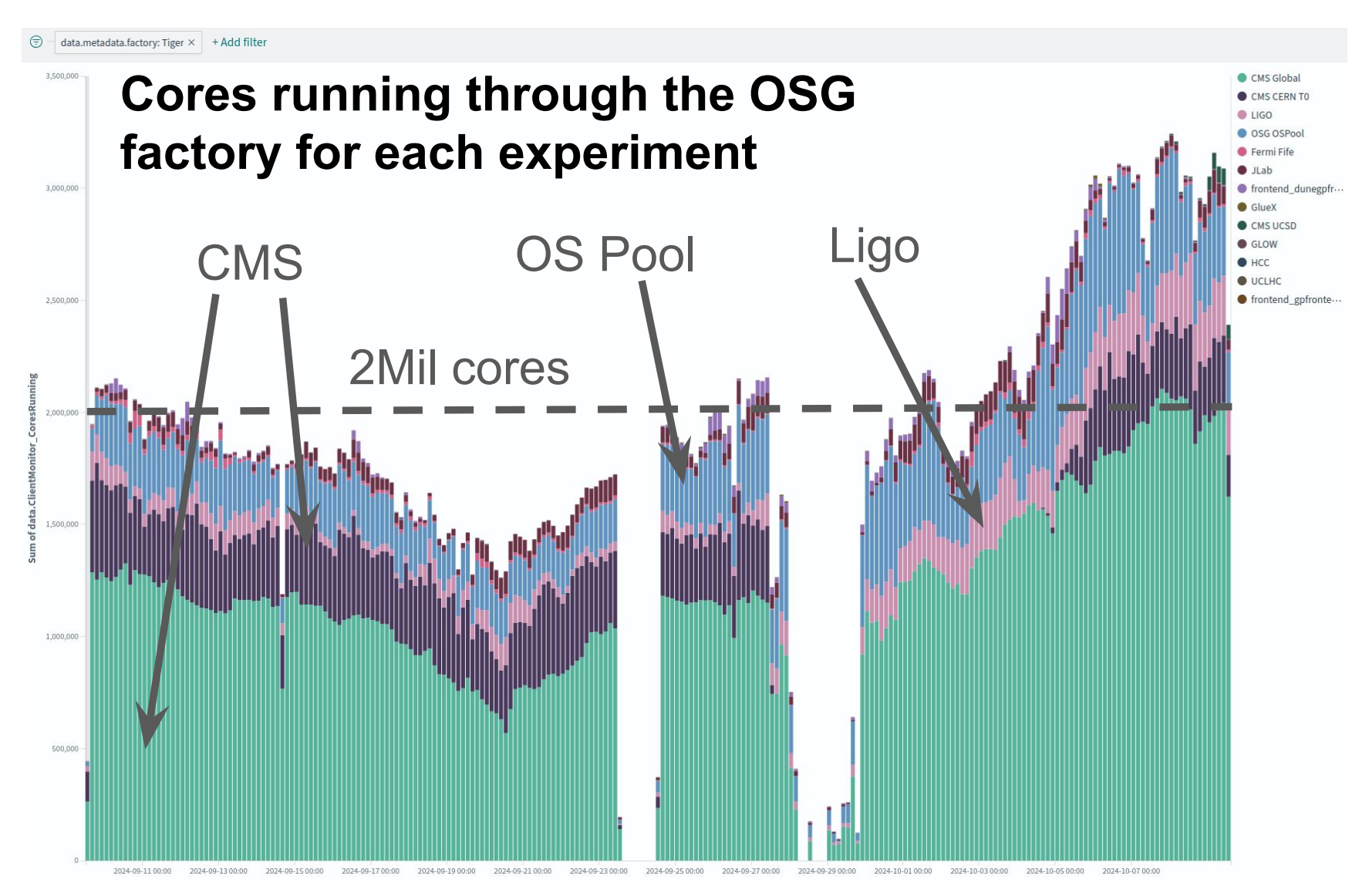
GlideinWMS deployment

- Multiple GlideinWMS factories serve different frontends (VOs - Virtual Organizations).
  - FNAL factory serves CMS, U.S. Tier-2 (T2) and Tier-1 (T1) sites.
  - CERN factory handles glidein submission for all CMS sites globally.
  - The OSG factory supports all VOs, providing a more general-purpose service.
- This setup ensures resource distribution tailored to the needs of each organization, enhancing scalability and redundancy.



Motivation

- Automating scaling, load balancing, and failover processes: high availability and resilience.
- Part of a larger effort within OSG to modernize the infrastructure.
  - OSG services such the GRACC accounting system, glidein pool frontends and central managers, hosted compute entrypoints (CEs), software repositories, and web pages have also undergone similar migrations.
- The OSG factory, which supports CMS and other VOs, is one of the core components being migrated.
  - better integration with the broader OSG infrastructure



Challenges

- The Kubernetes (K8s) factory pod installs itself from scratch every restart
  - Any state must be saved between restarts. GlideinWMS and HTCondor both depend on state written to disk
  - The factory must run a reconfig command automatically on startup

Persistent State

- K8s provides a Persistent Volume Claim (PVC) object that allows us to retain state between pod restarts
- We created 3 NVMe backed Ceph volumes to save state for:
  - `/var/lib/condor`
  - `/var/lib/gwms-factory`
  - `/var/log/gwms-facotory`
- These volumes can be increased in size as needed without requiring a pod restart

Updates to Reconfig Procedure

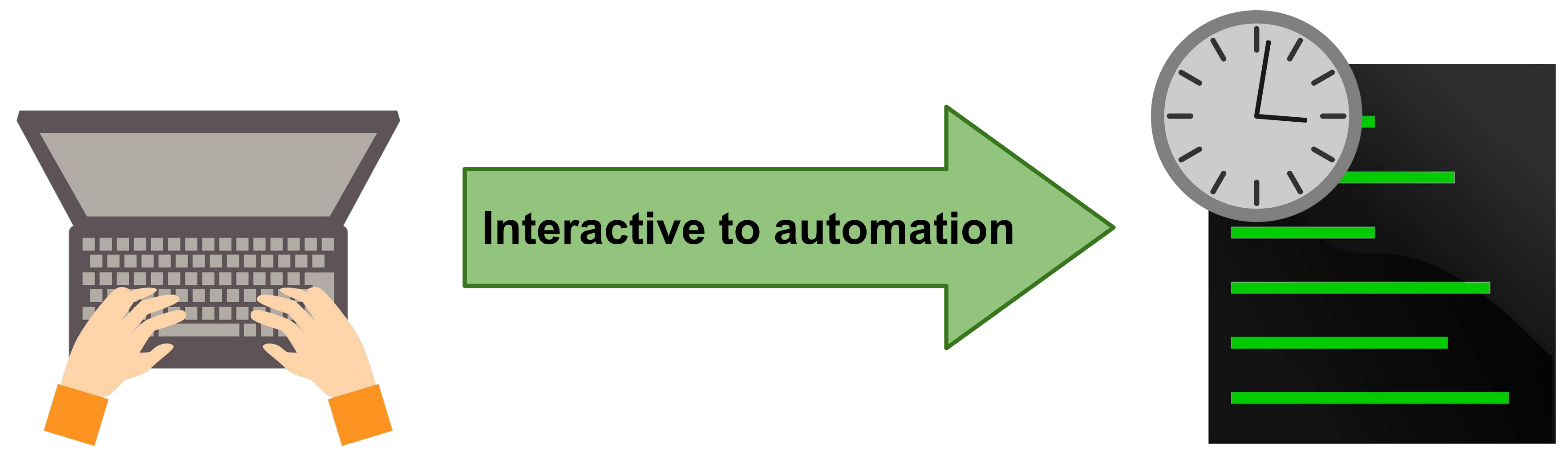
Old bare metal way

1. Update file(s) in `/etc/osg-gfactory/`
2. Manually run the following commands:

```
systemctl stop gwms-factory
gwms-factory reconfig
systemctl start gwms-factory
```

New K8s way

1. Update files(s) in osg-gfactory Git repo
2. Cron script fires off every 15 min to check for changes
3. If changes are detected, script automatically stops, reconfigures, and restarts the factory



Updates to Download HTCondor Tarballs

Old bare metal way

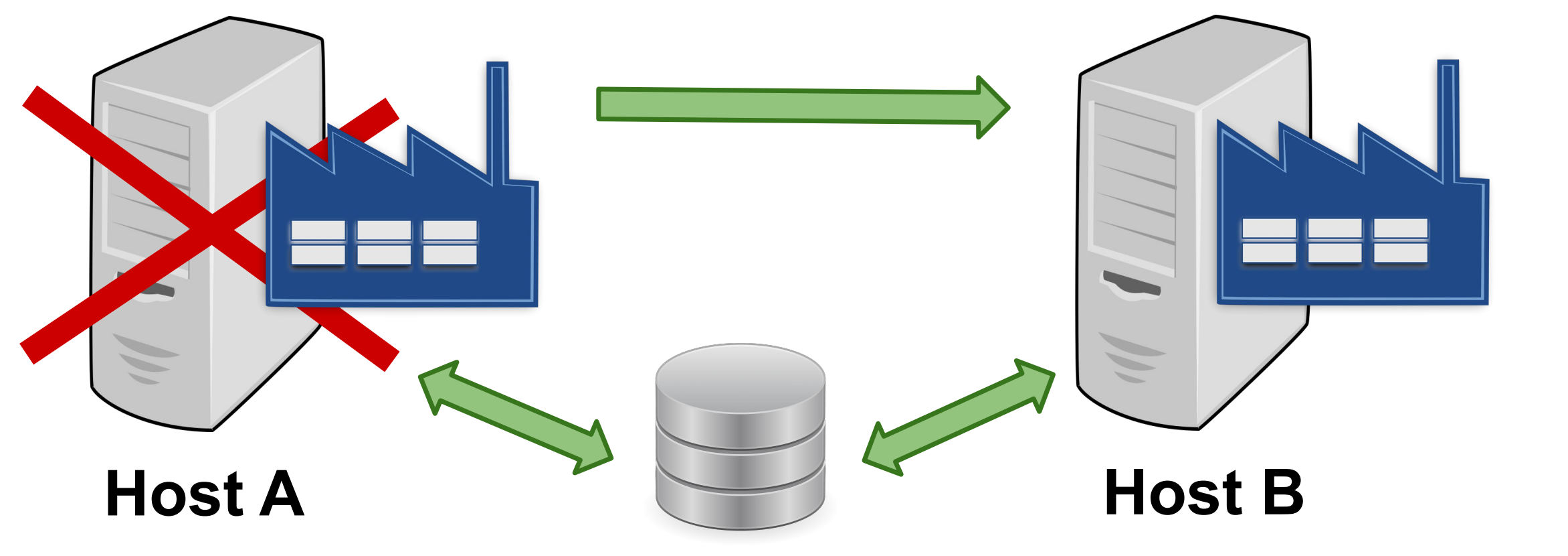
1. Manually download new tarballs from https://research.cs.wisc.edu/htcondor/tarball/ into `/var/lib/gwms-factory/condor/`
2. Run reconfig procedure described above

New K8s way

1. Update new tarball config yaml file in Git repo
2. Reconfig hook* runs the `get_tarball` script which automatically downloads any tarballs in config that the factory does not yet have

* runs whenever the new update Cron script above runs, so no manual reconfig required

Kubernetes Resilience



When a physical host goes down in Kubernetes, the factory pod will automatically restart on another host in the cluster

Conclusions

- Migrating the CMS Pilot Job Factory to Kubernetes streamlines deployment and management, improving scalability and reliability.
- Align with modern cloud-native practices, improving long-term maintainability.
  - E.g.: no need for local sys admins machine maintenance
  - Easier migrations with new OS releases
- Improved automation for reduced operational overhead

This work was partially supported by the US National Science Foundation under Grant No. 2121686 and under Grant No. 2030508