ETH zürich

IT Services

No-Touch Administration: Managing Slurm at Scale

Dr. Urban Borštnik
Scientific IT Services, HPC group

12–13. September 2024, Oslo

Contents

1. Background
2. Automating Slurm associations user a customer database
3. Automating Slurm deployment using a GitOps approach on Kubernetes

Mission

Operate a central HPC cluster at the Swiss Federal Institute of Technology (ETH Zurich)

- Research university with 25k students and 11k staff, 500 professors in 16 departments
- We operate the Euler cluster
  - on behalf of customers (shareholders) who invest into the cluster
  - to provide public computing resources to anyone at the ETH

© 2015 by Olivier Byrde, ETH Zurich

Automating administration

Euler has

- ~200 active shareholders (customers)
- ~4000 active users
- >2000 new users per year

Changes can not be processed manually.

Public domain, https://en.wikipedia.org/wiki/File:UnderwoodKeyboard.jpg

Shareholder database: Managing shareholders, users, resources

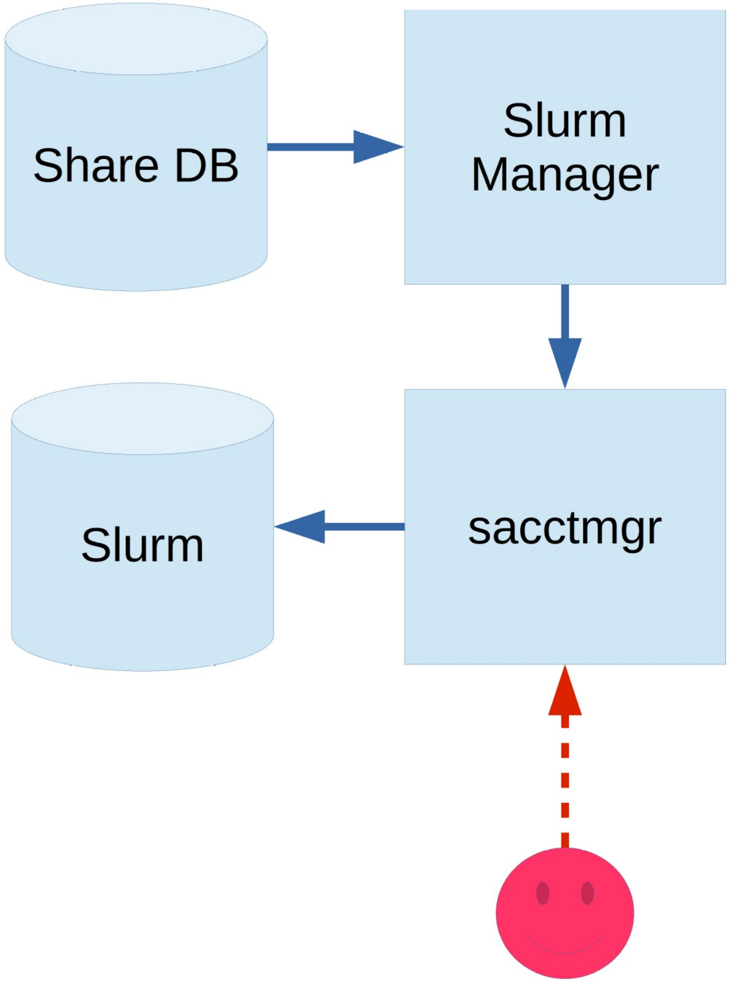
Shareholders invest into the cluster by buying hardware, giving them the right to use, on average, their resources for 4 years

Shareholder database

- **Who** is in the share (people)
  - e.g., everybody in the `DEPT-PROFESSOR` LDAP group
- **What** is in the share (resources)
  - e.g., *x* CPU nodes, *y* GPUs, *z* TB storage
  - *Validity* (start & end time)
  - Tenant
- Relies on ETH-wide LDAP directory with decentralized management
  - shareholders empowered to manage share memberships themselves

Slurm manager: Keeping Slurm users and accounts up-to-date

- Automatically manage Slurm associations
- Takes data from the share DB and updates Slurm
- Synchronize
  - users → Slurm Users
  - shareholders × resources → Slurm Accounts
  - shareholders × resources × runtimes → Slurm QOSs
  - share members × resources → Slurm Associations
- Runs periodically to sync changes from LDAP



Structure of Slurm users and accounts

- users → Slurm Users
- Account hierarchy and associations (shareholders × resources)
  - normal
    - normal/public ← anyone
    - normal/hpc_group
    - normal/chemistry_group
  - gpu
    - gpu/chemistry_group ← only they have access to GPUs
- QOS (shareholders × resources × time)
  - chemistry_group/normal/[4,24,120]h
  - Priority and limits proportional to resources, inversely to time and usage

Events: Automatic changes to Slurm users and accounts

Only active users are in Slurm (~4900).

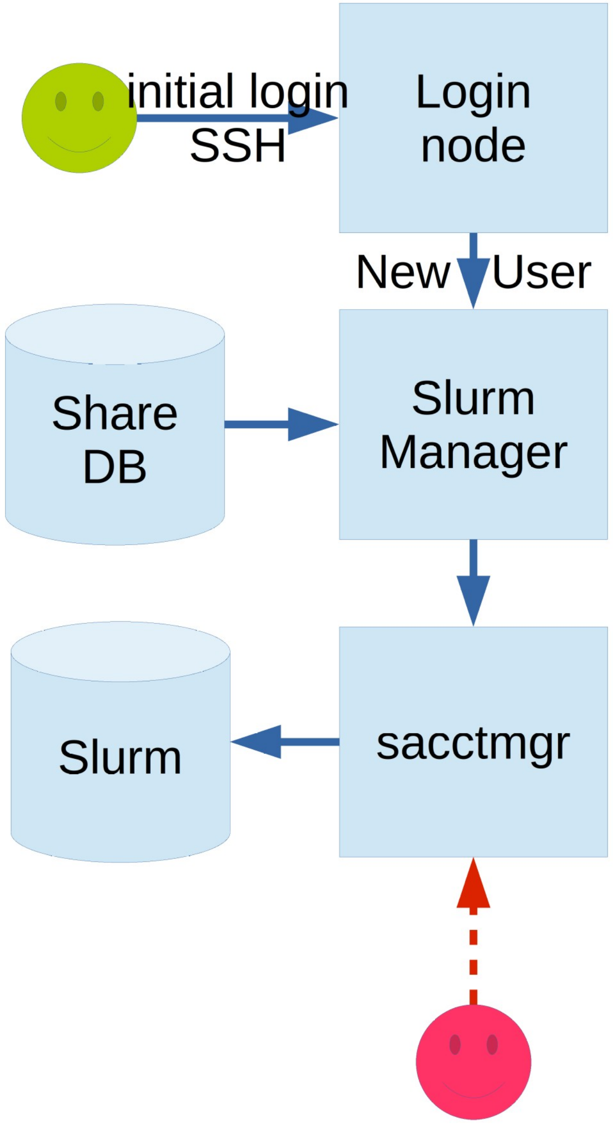
- Remove inactive users
- Add new users

First login

- Create $HOME directory
- Slurm manager notified to add user & associations

Job submission

- Check share membership and set Slurm Account and Slurm QOS



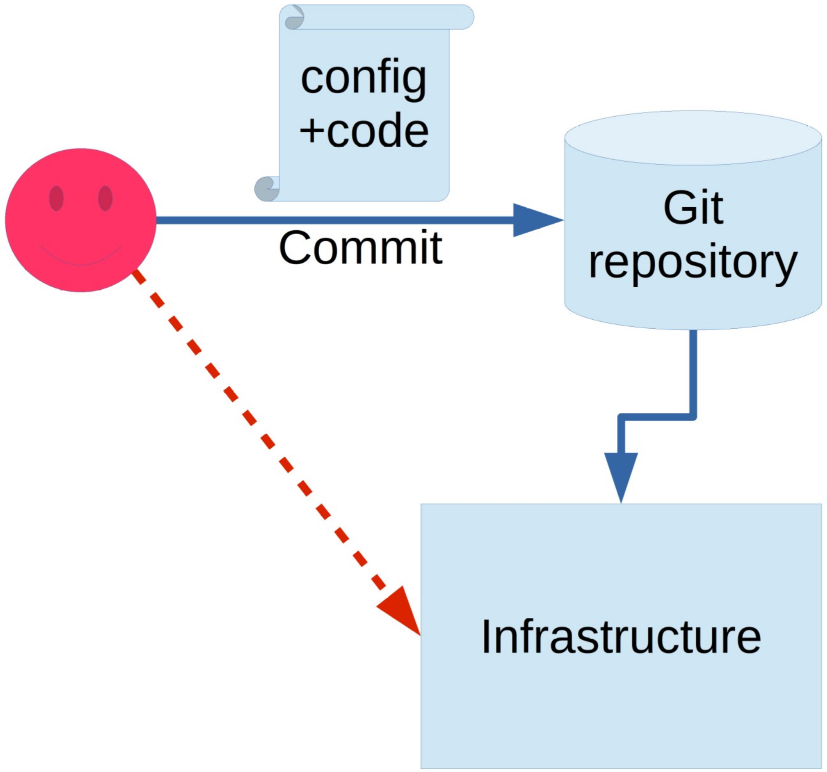
No-touch administration: Users and accounts

- There is **no** day-to-day administration of users and accounts
- Anyone can log in and start computing without delay in the correct Slurm account
- Expirations of compute resources are handled automatically
- New shareholders and new investments *are* processed manually (~150/year)

Automating Slurm deployment: GitOps

GitOps: Automatically deploy infrastructure from code and configuration stored in a Git repository.

- A Git repository is the source of truth
- Declarative deployment
  - You commit changes
  - You don't execute changes
- No pets (except for the git repository and persistent data)
- CI/CD pipelines
- Natural fit for Kubernetes (k8s)



Deploying Slurm on Kubernetes using GitOps

Why?

- Technical blank slate: we only started using Slurm in 2022
- Stable Kubernetes (k8s) cluster
  - years of Kubernetes experience
  - consolidating services on k8s if appropriate
- Ease of deployment & flexibility
  - hardware, network assignments are implied
- Rely k8s's self-healing instead of application-level high-availability (HA)
- Consistent monitoring and logging
- The possibilities!
  - Testing, experimentation, temporary clusters

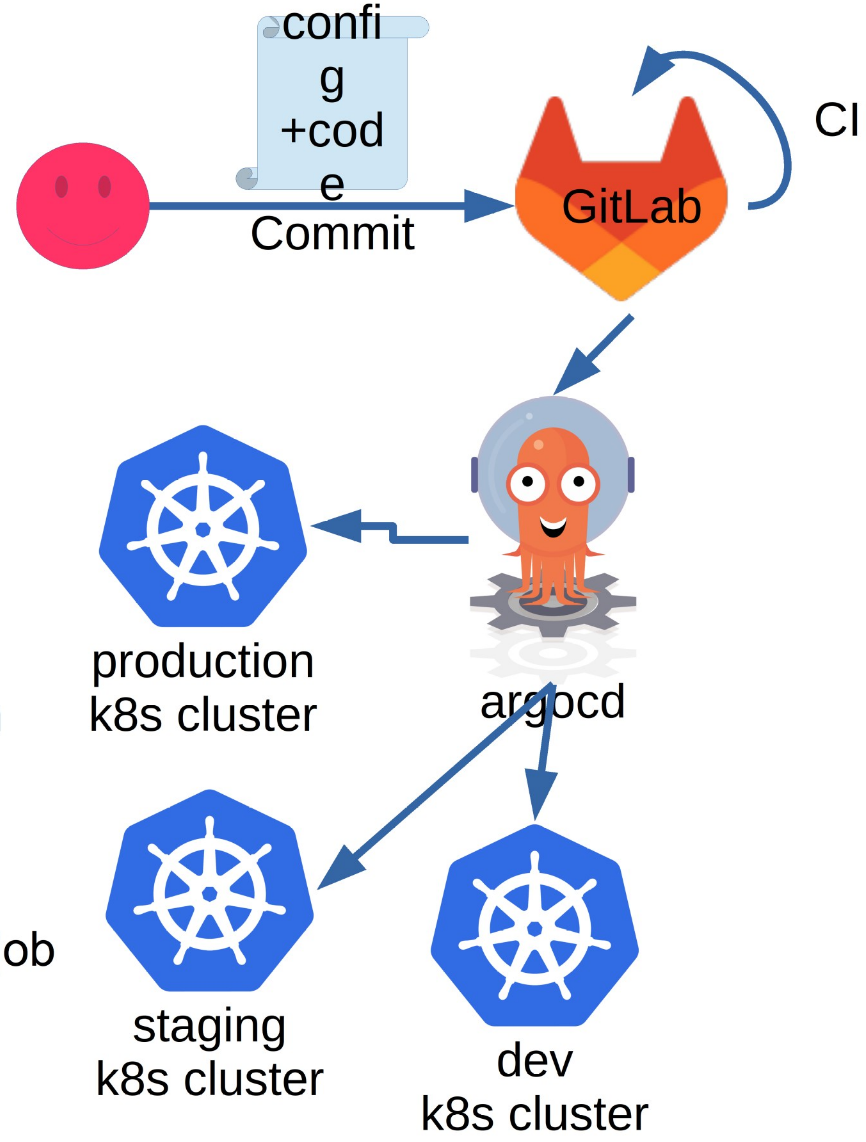
•

Slurm git repositories

- 2 Git repositories: template + values
  - s3mok8s, a Helm chart (template) to
    - build containers for the Slurm daemons, mariadb
    - define how to run the containers in k8s pods
    - define services: mapping an (IP address+port) to the Slurm daemon
    - define storage (for slurmctld, mariadb, logs, …)
    - secrets (munge key, mariadb credentials, …)
    - containerized compute nodes
    - tests
  - s3mok8s-values: “fill-in” values for the Helm chart:
    - Slurm configuration (nodes, topology, partitions, Slurm.conf options, …)
    - deployment specifics (DNS names, storage system, …)

Deploying Slurm using GitOps: Environments

- Every change to the template goes through at least these environments:
  - Dev (+ short-lived experimental during development)
    - compute containers
  - Staging:
    - 2 physical compute nodes
  - Production
    - 1200+ physical compute nodes of different types on several networks
- Functionality is tested in every environment:
  - from the Slurm configuration through job submission to job execution
  - manual testing of changes, esp. on staging



No-touch administration: Deployment

- GitOps with kubernetes allows no-touch administration:
  - There is **no** daily overhead with the deployment
  - CI/CD pipelines mean that any changes that land in production are well-tested
  - K8s provides a managed hardware, network, and kernel

Acknowledgements

- K8s DevOps: Steven Armstrong, Michál Minař, Loïc Hausammann, Nicolás Kowenski
- Hardware and Slurm Operators: Diego Moreno, Eric Müller
- K8s Persistent Storage: Allen Neeser
- User Support: Sam Fux, Nadia Marounina
- Procurement: Christian Bolliger
- Group head: Olivier Byrde

Image credits:
Gitlab logo: GitLab B.V., MIT license
ArgoCD: ArgoCD, Apache license
Kubernetes: Google, Inc, Public domain

ETH zürich

Dr. Borštnik
urban.borstnik@id.ethz.ch

ETH Zurich
Scientific IT Services
OCT G 45
Binzmühlestrasse 130
8092 Zurich
Switzerland

https://sis.id.ethz.ch/