

Simultaneous User Management for LDAP and Slurm

The [Slurm](#) cluster management system lacks direct LDAP integration, which can make user management quite cumbersome. Slurm is not automatically aware of any users in the system and what their resource limits in the cluster should be. Hence, a new user must be added to the LDAP instance and the Slurm database, which requires double bookkeeping and is error-prone (e.g. user might exist in Slurm but has been deleted in LDAP or vice versa).

Ideally, the LDAP instance is the single source of truth for what individual users are able to do on the system and even configurations specific to Slurm (e.g. resource limits) should be managed via LDAP.

This application allows for the simultaneous creation, modification, and deletion of LDAP and Slurm entities. Under the hood the [ldap3](#) client library is used to manage users in LDAP and Slurm's [sacctmgr](#) utility is called as a subprocess to add/modify/delete users in Slurm.

Additionally, this application allows for the creation of various directories on the cluster (e.g. home, nfs share etc.) and sets the appropriate user quotas for these directories via [setquota](#).

Note:

The [usermgmt](#) application expects an auxiliary LDAP [ObjectClass](#) (e.g. called [slurmRole](#)). The [ObjectClass](#) must unlock access to several [AttributeTypes](#) that can be used to manage Slurm-specific things like quality-of-service (QOS).

Currently, [usermgmt](#) expects the following [AttributeTypes](#) to present in your LDAP instance:

- [slurmDefaultQos](#): Specifies the user's default QOS. Can only exist once per user.
- [slurmQos](#): Specifies the QOS available to the user. Can be added multiple times to a specific user.

Requirements

LDAP

The LDAP instance needs an [auxiliary ObjectClass](#) (e.g. called [slurmRole](#)), which provides the [AttributeTypes](#) [slurmDefaultQos](#) and [slurmQos](#).

See documentations like [this](#) for details about the creation of new schemas in LDAP.

Slurm

The only dependency to Slurm is the [sacctmgr](#) ([Slurm Account Manager](#)), which interacts with the interface provided by [slurmdbd](#) (Slurm Database Daemon). The [sacctmgr](#) tool should be available on the control host of your cluster.

You need to point to the [sacctmgr](#) binary location in the `/etc/usermgmt/conf.toml` file.

Directory management

The application relies on SSH and common commands such as `mkdir` and `setquota` to be available on each target node. During execution of the directory management module, you will be prompted for a username and password to establish SSH connections with. Note that the provided username and password must be the same on all nodes you want to manage directories for.

Also make sure that the user can execute the following commands using sudo privileges **without password** (interactive password prompts via SSH are a bit of a hassle):

- `mkdir`
- `chown`
- `setquota`
- `mkhomedir_helper`

One way to accomplish this is by adding these commands to the `/etc/sudoers` file:

```
# /etc/sudoers
username ALL = (root) NOPASSWD: /usr/bin/mkdir
username ALL = (root) NOPASSWD: /usr/bin/chown
username ALL = (root) NOPASSWD: /usr/sbin/setquota
username ALL = (root) NOPASSWD: /usr/sbin/mkhomedir_helper
```

Replace `username` by the user you want to execute the commands with and make sure the paths to the executables are correct.

Note: Use `sudo visudo` to change the sudoers file!

Build and Install 🐛

You can build the `usermgmt` tool using Cargo:

```
cargo build
```

The following examples show how you can run the program with Cargo:

```
# Show available arguments
cargo run -- --help

# Add a user
cargo run -- add teststaff123 --group staff --firstname Martina --lastname
Musterfrau --publickey key.pub

# Modify user
cargo run -- modify teststaff123 -f Martha -m bla@blubb.de -d interactive

# Delete user
cargo run -- delete teststaff123
```

```
# List users in LDAP
cargo run -- list --ldap-users

# Run with different log-level
# Available are: error, warn, info, debug, and trace.
# Error represents the highest-priority log messages and trace the lowest.
# The default is info
RUST_LOG=warn cargo run -- delete teststaff123

# Add user in LDAP only
cargo run -- --ldap-only add teststaff123 --group staff --firstname Martina
--lastname Musterfrau
```

Create Debian Package

We use [cargo-deb](#) to automatically create a Debian package for production usage.

The package creation and installation steps are listed below:

```
# Install cargo-deb
cargo install cargo-deb
# Create Debian package in Debian package
target/debian/<project_name>_<version>_<arch>.deb
cargo deb
# Install package
dpkg -i target/debian/*.deb
# For previously installed packages, don't forget to update your conf.toml,
# in case there have been config changes
cp conf.toml /etc/usermgmt
```

Configuration

Location of configuration file

A basic configuration file ([conf.toml](#)) is loaded during runtime. This file determines a large portion of the behaviour of the program. The program tries to load the configuration file from several places. The first found configuration file is loaded. The search is conducted in the following order:

- Under user specific configuration places according to the OS.
 - Which folder on which OS is used, can be looked up [here](#)
 - In Addition under Linux: "~/.usermgmt"
- Under the system wide paths in respect to the OS
 - Under Linux: "/usr/usermgmt"
- The CWD as the last resort

The program does not create the configuration file and the locations listed above, automatically ! You do not need to create the configuration from scratch though. By using the command generate-config like this

```
usermgmt generate-config
```

You can acquire a default configuration. This output goes to the stdout of the terminal by default. You can create a local configuration file via piping. This example creates a local default configuration file at '/home/foo/conf.toml' via piping

```
usermgmt generate-config > /home/foo/conf.toml
```

Structure and content of configuration file

The `conf.toml` file looks as follows:

```
# Default value of the Slurm default QOS for the student group
student_default_qos = 'basic'
# Default value of the Slurm default QOS for the staff group
staff_default_qos = 'advanced'
# Default value of the Slurm QOS for the student group
student_qos = [
    'interactive',
    'basic',
    'gpubasic',
]
# Default value of the Slurm QOS for the staff group
staff_qos = [
    'interactive',
    'advanced',
    'gpubasic',
]
# A list of QOS against which user inputs are validated.
# Note that the values set here must also exist as actual QOS in Slurm.
valid_qos = [
    'interactive',
    'basic',
    'advanced',
]
# A list of groups against which user inputs are validated.
# Note that the values set here must also exist as actual Accounts in
Slurm.
valid_slurm_groups = [
    'staff',
    'student',
]
# Common object class values each user entity in LDAP needs to have
objectclass_common = [
    'inetOrgPerson',
    'ldapPublicKey',
    'organizationalPerson',
    'person',
]
```

```

    'posixAccount',
    'shadowAccount',
    'slurmRole',
    'top',
]
# List of compute nodes on your cluster
# Will be used to create user directories on local disks
compute_nodes = [
    'machine.test.de',
]
# Gid of the student group
student_gid = 1002
# Gid of the staff group
staff_gid = 1001
# Gid of the faculty group
# (faculty users will be treated the same as the staff group in Slurm)
faculty_gid = 1000
# Path to sacctmgr binary
sacctmgr_path = '/usr/local/bin/sacctmgr'
# Domain components used for LDAP queries
# Will be used in combination with ldap_org_unit
# and the cn of the username you provided for ldap login
ldap_domain_components = 'cn=department,dc=company,dc=com'
# Default login shell for the user
login_shell = '/bin/bash'
# Organizational unit in LDAP used to apply operations under
# This value is combined with ldap_domain_components like
# Is optional and can be omitted.
# '[ldap_org_unit,]{ldap_domain_components}'
ldap_org_unit = 'people'
# User bind prefix to be used when establishing LDAP connections.
# Binding goes like: cn=admin... or uid=someuser...
ldap_bind_prefix = 'cn'
# Use a different OU for establishing connections to you LDAP server
# Is optional and can be omitted.
# The resulting ldap path for logging is: {ldap_bind_prefix}=
# <ldap_user_name>,[ldap_bind_prefix,][ldap_domain_components]
ldap_bind_org_unit = 'ou=people'
# Protocol, host and port of your LDAP server
ldap_server = 'ldap://<hostname>:<port>'
# Read only user for ldap search queries (e.g. usermgmt list ldap)
# Is optional and can be omitted.
ldap_readonly_user = 'readonlyuser'
# Read only user password
# Is optional and can be omitted.
ldap_readonly_pw = 'secret'
# Can be used for connection with read access only
# Is optional and can be omitted.
# User bind prefix to be used when establishing LDAP connections.
# Binding goes like: cn=admin... or uid=someuser...
ldap_readonly_user_prefix = "read_only_uid"
# Can be used for connection with read access only
# Is optional and can be omitted.
# The resulting ldap path for logging is: {ldap_bind_prefix}=

```

```

<ldap_user_name>,[ldap_bind_prefix,][ldap_domain_components]
ldap_readonly_bind = "ou=readonly,ou=realm"
# Default user for SSH login during directory management.
# You can always enter a different username during application runtime
default_ssh_user = 'serveradmin'
# Hostname of the server that provides the home directories
# Assumes that a single host is responsible for home directories
# and that they are shared via nfs
home_host = 'home.server.de'
# Hostname of an nfs server to be used by cluster users
nfs_host = 'nfs.server.de'
# Slurm head node (where sacctmgr is installed)
# Required when run_slurm_remote=true
head_node = 'head.node.de'
# Root directory of the shared folders on the nfs host
nfs_root_dir = '/mnt/md0/scratch'
# Root directory of user folders on each compute node
# (must be the same on each node)
compute_node_root_dir = '/mnt/md0/user'
# Filesystem (or mountpoint) under which user quotas are to be set on the
compute nodes
filesystem = '/mnt/md0'
# Filesystem (or mountpoint) under which user quotas on the user's home
directory are to be set
home_filesystem = '/dev/sdb4'
# Filesystem (or mountpoint) under which user quotas are to be set on the
NFS
nfs_filesystem = '/dev/sda1'
# Quota softlimit on compute nodes
quota_softlimit = '200G'
# Quota hardlimit on compute nodes
quota_hardlimit = '220G'
# Quota softlimit on nfs
quota_nfs_softlimit = '200G'
# Quota hardlimit on compute nfs
quota_nfs_hardlimit = '220G'
# Quota softlimit on user home
quota_home_softlimit = '20G'
# Quota hardlimit on user home
quota_home_hardlimit = '22G'
# Create/delete/modify user on the slurm data base by default
# Can be overridden via CLI option for a command
include_slurm = true
# Create/delete/modify user on the ldap data base by default
# Can be overridden via CLI option for a command
include_ldap = true
# Use the directory management module of the application
# Note that this is somewhat experimental and quite specific to
# the THN cluster and therefore might not be suitable for
# other cluster environments
include_dir_mgmt = true
# Use the mkhomedir_helper tool to create the user home
# directory (recommended). When false, the directory will
# be created using mkdir and no skeleton configs (e.g. .bashrc) will be

```

```
copied
use_homedir_helper = true
# Execute Slurm commands from a remote client via SSH or directly on the
server
run_slurm_remote = true
# Port to be used when connecting via ssh to any node
ssh_port = 22
# If true, the application will try to authenticate via a ssh agent before
the simple password authentication
ssh_agent = false
```

The values for `student_default_qos`, `staff_default_qos`, `student_qos`, and `staff_qos` will be used when `--default-qos` and `--qos` are not explicitly set.

Usage

The following examples show the basic usage of the `usermgmt` tool:

```
# Show available arguments
usermgmt --help
# Show help for modify subcommand
usermgmt modify --help
# Add a user
usermgmt add teststaff123 --group staff --firstname Martina --lastname
Musterfrau
# Modify user
usermgmt modify teststaff123 --firstname Martha --mail bla@blubb.de --
default-qos interactive
# Delete user
usermgmt delete teststaff123
```

Adding Users

The uid integer value will be automatically determined based on the `--group` parameter provided. Currently you can choose between the two groups *staff* and *student*.

The uid for a new user will be determined based on the following rules:

- Uids for *staff* start with 1000
- Uids for *student* start with 10000
- The uid will be 1 plus the highest uid currently present in LDAP

The gids are determined based on the string provided in `--group` using the values in `conf.toml`. Therefore, a gid for each valid group must be present in the `/etc/usermgmt/conf.toml` file.

When no `--default-qos` or `--qos` parameter is set, the default values provided in the `/etc/usermgmt/conf.toml` file will be used based on the `--group` parameter given.

Modifying Users

A list of modifiable values can be obtained via `usermgmt modify --help`.

Deleting Users

User can be deleted via `usermgmt delete <username>`.

Tips and advanced usage

Use SSH agent for ssh authentication

To save yourself entering password for ssh authentication again and again, you can let the application use a running ssh agent.

Activate this feature via setting the field `ssh_agent` from false to true inside the configuration file.

Start your ssh agent in the terminal via the command

```
ssh-agent
```

Then add your private ssh key to the ssh agent via this command. You might be asked for the password to decrypt this key if a password was set during creation of the key pair.

```
ssh-agent <path_to_private_ssh_key>
```

If authentication over ssh is requested, the application will try to use one of your keys registered within the ssh agent and does not ask you for a password. If more than one key are registered within the agent, you will be asked for which key to use via a prompt in the terminal.

Show more logs

The log-level can be changed using the `RUST_LOG` environment variable. Available log-levels are *error*, *warn*, *info*, *debug*, and *trace*. *Error* represents the highest-priority log messages and *trace* the lowest. The default log-level is *info*. You'll receive the most verbose output when you set it to *debug*.

```
# Delete user with log-level debug
RUST_LOG=debug usermgmt delete teststaff123
```

Show stack trace in case of error

Many of the errors, reported by the application, can also shown with their stack trace. The stack trace is quite useful for locating the place in the code where the error was caused. This is especially handy for debugging. By default the stack trace in Rust is disabled though. You need to set the environmental variable named "RUST_BACKTRACE" to 1. This can be accomplished via this command in the terminal.


```
export RUST_BACKTRACE=1
```

Pitfalls

Make sure you execute the `userrgmt` tool with a user who has **administrative rights** for `sacctmgr`. You can check available users and their admin level via `sacctmgr list user`.

When you attempt LDAP operations, you will be prompted for a username and a password. Make sure the user has sufficient rights to add, modify, and delete entities in LDAP.

External Dependencies

- [Slurm Account Manager](#) as part of slurmdbd. Make sure this is installed on the host you're executing this tool from.

Release

You need to include the current version of your release in the `CHANGELOG.md` because the [github action](#) picks it up from there.

Also change the version number in `Cargo.toml` and `src/cli.rs` to keep everything consistent.

To add a release, you need to tag the branch with the current version and then push the tag:

```
git tag <version> main  
git push origin <version>
```

`<version>` is the version of your release (e.g. `v0.3.0`).

Note: Don't forget to push your commits to main as well (`git push`).

Build for Mac M1

Cross compilation for the M1 target via github actions currently fails due to missing dependencies for OpenSSL, but you can build the application natively on an M1:

```
cargo build --release --target aarch64-apple-darwin
```

If you want to make a release file similar to the ones created automatically via the github action do:


```
cp README.md target/aarch64-apple-darwin/release  
cp LICENSE target/aarch64-apple-darwin/release
```

```
cd target/aarch64-apple-darwin/release
tar -cvzf usermgmt-aarch64-apple-darwin.tar.gz usermgmt README.md LICENSE
```

Local development with via docker

Development of this app can be done locally via a docker container set up. As moment of writing the functionality for LDAP and Slurm can be used via docker. Directory management does not work in docker. Read this [Readme](#) for how to set up local development via docker.

GUI

The GUI version of the usermgmt tool is made by slint  Logo of slint