Linux server commands/utility guides

Collection of procedures and setups that have been done on the Linux DataJoint machine. If not specified, all commands are to be executed on the Linux machine, after connecting to it via SSH (see GitHub Readme).

Change time zone

- Display the currently active time zone with `timedatectl`
- List names of all possible time zones with `timedatectl list-timezones`
- Out of this list, choose the appropriate time zone and change to it with `sudo timedatectl set-timezone ZoneName`

Set up and edit Credentials Profile with mysql_config_editor

- The program `mysql_config_editor` stores login details (username, password, host IP) in the file `.mylogin.cnf`
- `.mylogin.cnf` is obfuscated, meaning that it can only by read by other MySQL programs and does not store credentials in plain text
- `.mylogin.cnf` can store several profiles (e.g. root and user credentials, or credentials to several databases)
- Create a new profile with the following one-line command:
  `mysql_config_editor set --login-path=`*`name`* `--host=127.0.0.1 --user=`*`username`* `--password`
  and enter the password when prompted.
- Tell MySQL programs which profile to use with the additional command `--login-path=`*`name`*.
- For some reason, this does not work for mysqldump (see "Backup" guide below)

Solve bug with text editor displaying black screen

- When opening Nano or another text editor, the screen turns black and gets stuck
- Bug sounds like this report: https://superuser.com/questions/1684867/wsl-issue-vi-vim-and-nano-are-unusable-empty-terminal-screen-same-isue-with
- Suggested solution worked:
  - If you encounter the bug, close and reopen the console and type `export TERM=xterm-color`
  - Line is also added to `~/.bashrc`, which hopefully automatically resolves the bug

Set up scripts that are executed regularly (cron jobs)

- See list of cron jobs with `crontab -l`
- Root has its own list of cron jobs for scripts that require sudo access: `sudo crontab -l`
- Edit the list of cron jobs with `(sudo) crontab -e`
- Each cron job is its own row in the file
- Produce cron tab schedule easily with https://crontab.guru/
- Executions of cron jobs are automatically logged to `/var/log/cron`
- Output (echos and errors) can be logged to log files and later inspected (for us located in `/home/hheise/logs`)
- Current sudo crontab:
  - After every reboot: `server_reboot.sh`
    `@reboot`

```
/home/hheise/datajoint_wahl/util/server_config/server_reboot.sh >
/home/hheise/logs/reboot.log 2>&1
```

- At 10 a.m. every Friday: `mouse_reminder.sh`

```
0 10 * * 5
/home/hheise/datajoint_wahl/util/server_config/mouse_reminder.sh >
/home/hheise/logs/mouse_reminder.log 2>&1
```

- Current crontab:
  - At 4 a.m. every day: `daily_backup.sh` (mysqldump does not work with sudo):

```
0 4 * * *
/home/hheise/datajoint_wahl/util/server_config/daily_backup.sh >
/home/hheise/logs/daily_backup.log 2>&1
```

  - At 12 a.m. every Sunday: `weekly_backup.sh`:

```
0 0 * * 0
/home/hheise/datajoint_wahl/util/server_config/weekly_backup.sh >
/home/hheise/logs/weekly_backup.log 2>&1
```

Update local datajoint_wahl GitHub repo

- The "datajoint_wahl" GitHub repository is stored locally on the machine under `/home/hheise/datajoint_wahl`
- If repo does not exist yet on this machine, create it with `git clone https://github.com/Hendriela/datajoint_wahl.git` while being in the `/home/hheise` directory
- If changes to the repo have been made which are important for the functioning of the server (e.g. shell scripts, structural changes to the `common_mice.Mouse` and `common_mice.Weight` tables (which the server uses for `mouse_reminder.py`), etc.), update the local repo with these steps
  - Enter the directory with `cd /home/hheise/datajoint_wahl`
  - Get repo updates with `git pull`
  - In case of a merge conflict:
    - More complete, but more work: Discard all local changes with `git stash` and re-apply changes that have to be done on the server side (easier than dealing with the conflict through the terminal):
      - Replace dummy values for the datajoint_wahl Gmail account password with actual password in `notify_server_restart.py` and `mouse_reminder.py`
      - Make all shell scripts executable with `chmod +x script_name.sh` (all shell scripts should have a green name when displayed with `ls`)
    - Usually easier: Delete the changed file with `rm changed_file.sh` and `git pull` again. Apply local changes to the removed file only
    - Possible, but no guide yet: Use command-line to `git commit` changes and resolve merge conflicts directly with git
- If login.py has not been created yet, save it as a copy of login_TEMPLATE.py and make necessary edits (remove keyring import, enter datajoint credentials, remove error raises, etc.) -> these changes are not removed after `git stash` and only have to be done once after cloning

Set up Python virtual environment for DataJoint

- Create environment in a directory (e.g. the datajoint_wahl directory) with
  `python3 -m venv /home/hheise/datajoint_wahl`
- Activate the environment with
  `source /home/hheise/datajoint_wahl/bin/activate`
- Try to run the Python script and pip install packages that are missing

Calling a Python script with a shell command

- Open (or create if nonexistent) a shell script with `nano my_script.sh`
- Enter content:
  ```
  #!/bin/bash
  python3 my_python_script.py
  ```
- If the Python script should run in the virtual environment, use that Python program instead:
  `/home/hheise/datajoint_wahl/bin/python3 my_python_script.py`
- Exit the text editor with `Ctrl+X` and save changes
- Mark the shell script executable with `chmod +x my_script.sh`
- Run the script with `./my_script.sh` or schedule it with a cron job

Create/change startup commands

- Startup commands are stored in shell script
  `/home/hheise/datajoint_wahl/util/server_reboot.sh`
- Put commands here that should be executed whenever the server reboots (restarting of the database, restoring settings that are lost during reboot, etc.)
- Script is in the GitHub repo, so you can edit the script in the repo and pull the changes on the server, or edit the file on the server directly (although then the repo file will not be up-to-date, so keep that in mind!
- Add script to the sudo crontab (sudo is necessary to start the MySQL database):
  `@reboot /home/hheise/datajoint_wahl/util/server_reboot.sh`

Permanently mount a network drive (e.g. Neurophysiology Server)

- Create a directory where the drive should be mounted, e.g.:
  `sudo mkdir /media/neurophysiology-storage1`
- Create a backup of the fstab file:
  `sudo cp /etc/fstab /etc/fstab_old`
- Create or edit a credential file to add credentials for the server:
  - `sudo nano /root/.smbcredentials`
  - Add the lines (replace values with actual username/password)
    ```
    username=winusername
    password=winpassword
    ```
  - Now, change the permissions of the file so only root can read and edit it:
    `sudo chmod 700 /root/.smbcredentials`
- Edit fstab file:
  `sudo nano /etc/fstab`
- At the end of the file, enter this one line (change server address if necessary):
  ```
  //130.60.51.15/neurophysiology-storage1
        /media/neurophysiology-storage1  cifs
  credentials=/root/.smbcredentials,iocharset=utf8,file_mode=077
  7,dir_mode=0777 0 0
  ```
- Try to mount your drives with `sudo mount -a`

- Now this drive should automatically mount after a reboot

Automatic backup

- Two cron jobs are scheduled to run `daily_backup.sh` and `weekly_backup.sh` in the datajoint_wahl repo under `/util/server_config (backup.sh` is legacy and not scheduled anymore due to disk space limitations)
- In short: All databases (schema) are backed up at 4 a.m. and compressed into a sql.gz file, which is stored locally under `/home/hheise/datajoint_backups` and then copied to the Wahl folder at `Neurophysioloy-Storage1/Wahl/Datajoint/backups/daily`
- Backups are performed with the dedicated MySQL user “backup”, which has SELECT and PROCESS rights for the entire database, the minimum privileges for a whole-db backup
- Backup files are named “daily” and “weekly” and will be overwritten at this time period
- Tables are not locked during the backup, to allow overnight computations to still access the database during the backup. This means that the database will be backed up in the state when the backup started, not when it finishes
- The whole database can be restored with `mysql < dump.sql`, a single schema from the large file with `mysql --one-database db1 < dump.sql`
- If the backup file is compressed (ends with `sql.gz`), add `gunzip < dump.sql.gz | mysql...` before the mysql command
- For more detailed explanations of dump parameters, check the shell script itself
- There is a bug in mysqldump 8, which can be fixed by setting up a config file:
    - Open/create config file in the local directory with `sudo nano ~/.my.cnf`
    - Add to the file:
      ```
      [mysqldump]
      column-statistics = 0
      ```
      and save.