GUCKY

TOOLS FOR HANDLING YOUR SERVER AND MYSQL DATABASE

From managing your server to running a web application

CONTENT

- Running things on your server
  - What OS should I run on the server?
  - Some useful tools for your server
  - Running jupyter on your server
  - Running a MySQL instance on your server
- Personal repositories that are pip-installable
- Web application or GUI for your MySQL instance
  - General apps available for MySQL
  - DataJoint-compliant apps

RUNNING THINGS ON YOUR SERVER

UBUNTU SERVER

EASY SERVER TO SETUP, FREE, AND LOTS OF SUPPORT ONLINE

USEFUL TOOLS FOR YOUR SERVER

- tmux: https://medium.com/hackernoon/a-gentle-introduction-to-tmux-8d784c404340
  - allows multiple terminal sessions to be accessed simultaneously in a single window
  - detach processes from the controlling terminals, allowing SSH sessions to remain active without being visible (i.e. when you log out of your server on your computer those processes will still be running and you go back to them after logging back in). Great for running processes on your server that take time or should always be running (such as a jupyter notebook or some analysis)
- On mac: use iterm2 instead of terminal (better for handling tmux sessions)
- For editing files you can use various terminal editors: vim, nano, emacs. Nano is the easiest to use at the beginning whereas I recommend vim for speed.
- install docker: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04
- anaconda: install for multiple user in the /opt directory of your server:
  - https://docs.anaconda.com/anaconda/install/multi-user/
- You can ask ZI Computing (and CUIMC Computing I assume) to give you a hostname for your lab server (then people don't need to remember the IP address for your server)
- Setup aliases to speed up your terminal workflow: https://www.tecmint.com/create-alias-in-linux/

JUPYTERHUB

- Allows you to run jupyter lab and notebook for all users on your server: users don’t need to individually install and maintain scientific environment
- https://jupyterhub.readthedocs.io/en/stable/quickstart.html
- you can install custom kernels for specific anaconda environments using:

```
python -m ipykernel --user --name MYENV --display-name “MYENV”
```

- Similar to your jupyter notebooks and jupyter lab you can configure jupyterhub with a configuration file: https://jupyterhub.readthedocs.io/en/stable/getting-started/config-basics.html

RUNNING A MYSQL INSTANCE ON YOUR SERVER

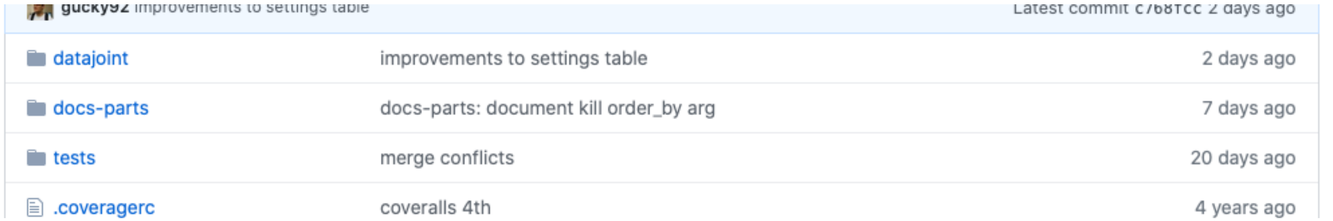
- Run a mysql instance on your server using the docker image
  - You can access your MySQL instance by either ssh into your server or if you allow direct access to the port 3306 on your server you can simply set your ‘database.host’ configuration in datajoint to your server IP/hostname.
  - If you are running a jupyterhub on your server, you can also directly access your MySQL server there as we did in the previous lectures
- Running multiple mysql instances (e.g. if you want a specific mysql instance for testing or tutorials):
  - clone the mysql-docker into different directories for your different use cases
  - In the docker-compose.yml file you will need to change two things:
    - the port
    - the volume
- Managing users and privileges in the MySQL database:
  - https://www.digitalocean.com/community/tutorials/how-to-create-a-new-user-and-grant-permissions-in-mysql

```
1 docker-compose.yml
version: '2'
services:
  db:
    # image: datajoint/mysql:5.6
    image: datajoint/mysql:5.7
    # image: datajoint/mysql:8.0
    volumes:
      - ./my.cnf:/etc/my.cnf
    ports:
      - "3308:3306"
    environment:
      - MYSQL_ROOT_PASSWORD=simple
    volumes:
      - ./data2:/var/lib/mysql2
    command: --character-set-server=utf8
```

CREATING YOUR OWN INSTALLABLE REPOSITORY

PIP-INSTALLABLE REPOSITORY

- your repository should contain one folder with the same name of the repository, which includes your actual code:
  - within this subdirectory and any subdirectories within it have a __init__.py file
- include a setup.py file
- include a requirements.txt (optional)
- Once you have cloned the repository, cd into it and run the following command:
  - pip install -e .
- Now you can simply import the code using “import YOURPACKAGENAME”
- you can also change any code and when you refresh the kernel in your session it will import the updated code (this is because we pip installed the code using the -e option)
- it is good practice to have a .gitignore file in your repositories that ignores various files when committing changes, such as .DS_Store or .pyc files, here is a template:
  - https://gist.github.com/GhostofGoes/94580e76cd251972b15b4821c8a06f59



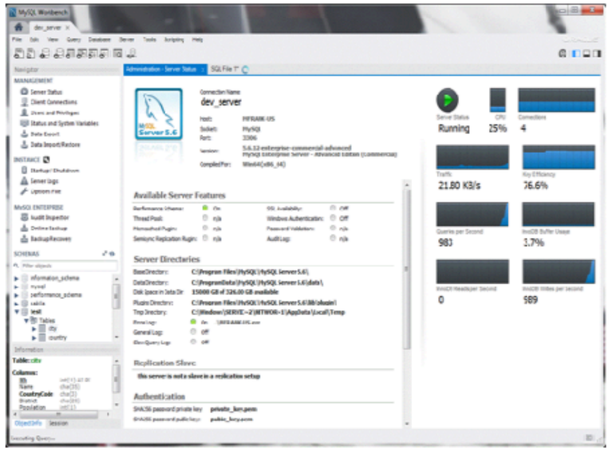

```
#!/usr/bin/env python

from setuptools import setup

setup(
    name='dreye',
    version='0.1',
    description='Dreye: Color models and stimuli for all model organisms',
    author='Matthias Christenson',
    author_email='gucky@gucky.eu',
    # install_requires=['requirements.txt'],
    # TODO requirement file
    packages=['dreye']
)
```

APPS FOR YOUR MYSQL INSTANCE

COMMONLY USED MYSQL APPS

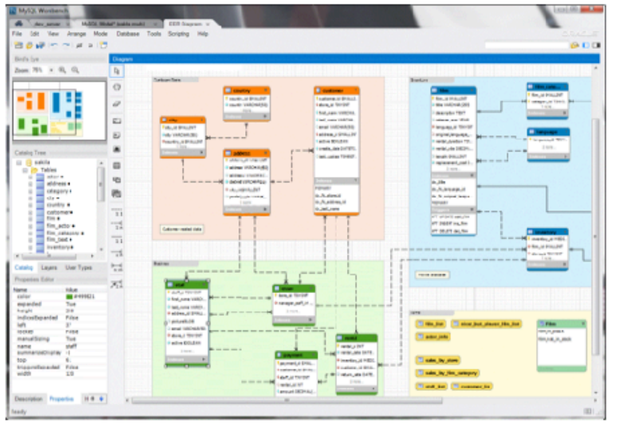
- MySQL Workbench
- Sequel Pro
- dbForge Studio
- HeidiSQL
- And many more: https://codingsight.com/10-best-mysql-gui-tools/
- IMPORTANT: These apps to do not follow the DataJoint data model

MySQL Workbench Home



View Screenshot:
Windows, Linux, OS X

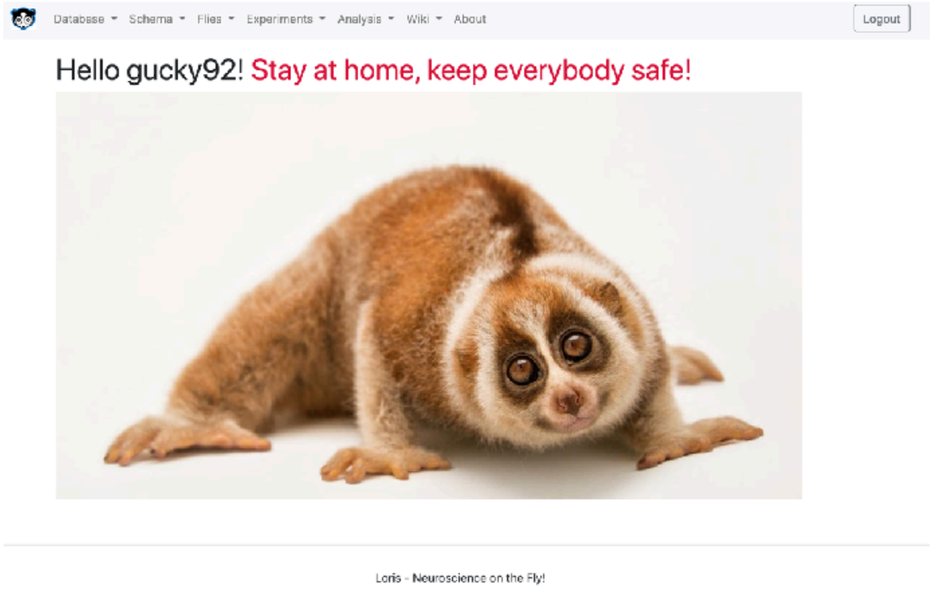
Visual Database Design



View Screenshot:
Windows, Linux, OS X

APPS DESIGNED AROUND DATAJOINT

- Helium: https://github.com/mattbdean/Helium
  - docker-installed
  - allows inserting and viewing tables
  - cannot edit/update entries and does not handle long blob datatypes
  - quite fast and simple design
- Loris: https://github.com/gucky92/loris
  - docker-installed (also intalls MySQL database), but can also be configured for specific lab use. It uses the micro web framework Flask.
  - Also a package for python, which includes various custom datatypes for datajoint and manipulating fetched data (such as putting fetched data into a long dataframe format for easy plotting)
  - includes wiki and can point to an external wiki
  - includes visualization of ERDs.
  - allows editing of existing entries.
  - create tables within the application.
  - allows joining and downloading of tables.
  - insert missing entry into upstream tables.
  - you can insert attachments and python datatypes (blob and attach)
  - Manages users and creates user-specific schemas and group project schemas
  - allows for entry specific user restrictions when trying to delete entries.
  - Various additional features, such as running experimental and analysis scripts from the web app
  - Has knowledge of AutoImported and AutoComputed tables (allows uploading of analysis scripts and running by simply clicking through options)
  - Comes with an optionally pre-defined database schema for fly labs
  - Still in development with new features to come - documentation is currently lacking
  - Generally slower than helium due to backend and extra checks being performed.



DISCUSSION

IS DATAJOINT MY ONLY OPTION?

NO

there are many other ways to interact with a database and other types of databases you can create. For example, in python there are other types of ORMs with extensive documentation that you may want to consider instead. These ORMs are generally designed for a more general audience, but also have various additional features, such as running a different databases e.g. PostgreSQL, SQLite:

•SQLAlchemy, PonyORM, Peewee, Tortoise

You may also think about using NoSQL database such as MongoDB.

We introduced DataJoint because that is what we know basically and use in our lab.

THE END