Lab Carpentry Documentation

Release 0.1

DDD

Dec 01, 2017

Contents

Contents:

CHAPTER 1

Coding and Software

These guidelines describe general approaches to storing your scientific work, to help:

- Guarantee lab-internal transparency of ongoing projects
- Ensure code lifetime extends beyond the contracts of any single lab-member
- Facilitate reproducibility in the long run

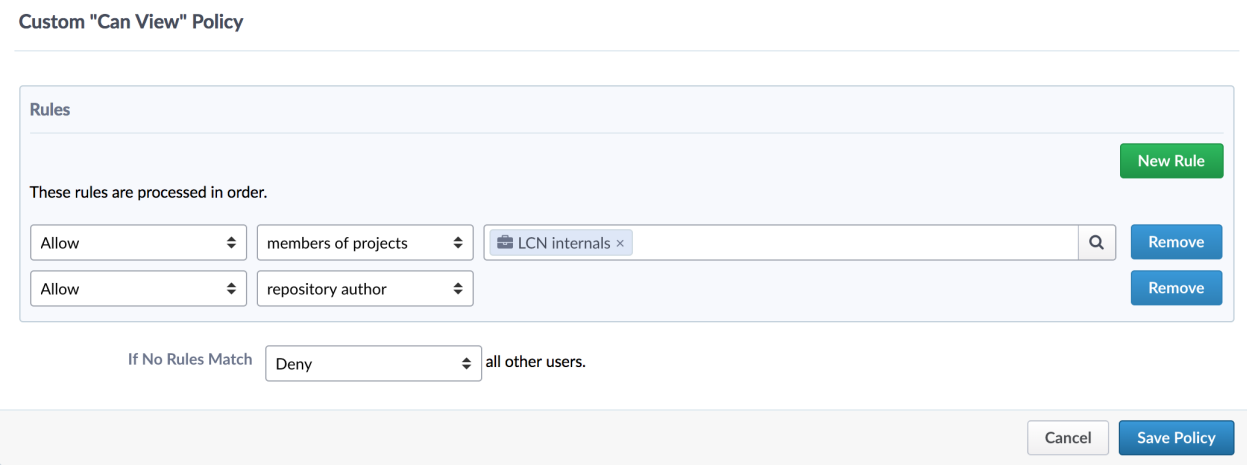
Some of the below are lab-wide policies that should be followed. These are **marked in bold below**.

1.1 Code guidelines

1.1.1 Internal/development code

Code should be stored in a **git repository on C4Science, accessible to the LCN group**. Please, exclude your data files from these repositories (see *Code vs Data*).

- For each project that you develop code for, create a Git repository at https://c4science.ch
- Make your Git repository readable for the members of the LCN-Internals project
  - Go to https://c4science.ch/diffusion/REPOSITORY_URL/manage/policies
  - Edit Policies -> "Visible to" -> Costum Policy
  - Make a Policy allowing members of LCN-Internals as well as "Repository Author", see the image below.



1.1.2 Publication code

Publicly released code should be stored in a public **Github repository**.

- Create a personal Github account
- Push your local git repository to a GitHub repository
- Your repository will then be forked into a repository at https://github.com/EPFL-LCN

This policy is established so both you, and the lab administrators, have control over the publicly available repositories.

1.1.3 Code Quality

Coming soon.

1.2 Data guidelines

1.2.1 Storage possibilities

GIT LFS

For files <1GB, you can use Git LFS as offered by C4Science.

Icfiler

For large data files, you can use shared lab storage. On all lab machines (lcncal1-5 and lcnsrv1-4), this is mounted as

```
/lcncluster
```

To access Icfiler from your own machine, you need to be connected to the EPFL network (use VPN if at home). You can then either:

1. mount the following SMB share

```
smb://icfiler2.epfl.ch/lcncluster
```

2. mount the drive using sshfs over one of our lab machines (here we use lcncal1)

```
mkdir mount_dir
sshfs gaspar_username@lcncal1:/lcncluster mount_dir
```

Switch drive

EPFL offers a free and swiss-hosted "dropbox" for all staff and students. Head over to Switchdrive to access it.

1.3 Code vs Data

In general, it is good practice to split your programming efforts into code and data.

1.3.1 Code

Programs that perform simulations, do data analysis, plot your results, and so on. Importantly, code can consume data, i.e. if your code performs analyses of data files and plots something. It also can produce data. For example, if your code simulates a neuronal system it might produce spike times. Your code can also produce derivative data (e.g. rates from spikes) if it performs analyses.

Code files have the following properties:

- Small: commonly text (non-binary) files that require a minimal amount of storage
- Dependent on interpreter: the syntax of your code will depend on whatever programming environment you write it for (e.g.: python, matlab, c, c++, perl, bash ...)
- Change often: Code will change often, e.g. as you need to implement a new functionality, fix a typo, add a comment.
- Suitable for versioned control: since the size of code files is small, incremental versions can be efficiently stored in version control systems (CVS). For example, this enables you to reproduce simulations or analyses that were produced with earlier versions of your code, even if you need to do this years later (given that you wrote down or annotated the of your code that was used).

1.3.2 Data

Most generally, data are numbers that are stored in files according to certain formats. Formats can range from simple storing each number in a row of a text file, to more intricate data format standards.

Common examples for data formats are: plain text, CVS, HDF5.

Data files have the following properties:

- Large: Depending on the volume of data and its format, data files tend to use more space than code.
- Independent of interpreter: data files can ideally be read from and written to by several interpreters, if general formats are used.
- Change rarely: most scientific data will usually be static, i.e. you will not change a single number here and there. If you often add measurements or simulations, this should produce additional data files.
- Not suitable for versioned control: due to the larger size of data files, they do not (or very rarely) lend themselves to versioned storage. On the bright side, data files also change rarely, and therefore do not benefit much from version control.