Version control with git

Git is a free and open source distributed version control system. It allows to track changes in source code and allows multiple people to work on the same code base.

We currently switching from svn to git and the repositories are/will be hosted on GitHub.

The analysis code for LCLS is currently hosted in the lcls-psana organization on GitHub.

Once the code is cleaned up and we have less repositories we will move the code to the slac-lcls organization. This organization will also be used by the DAQ group and allows to group all of your code development together.

Content

Setup git

The following steps describe how to setup git. This setup has to done only once for the machine where you are planing to develop code.

Setting the user name in git:

```
git config --global user.name <my-name>
```

Setting the email address in git:

```
git config --global user.email <my@email>
```

Confirm that your user name is correct by:

```
git config --global user.name
```

Confirm that your email is correct by:

```
git config --global user.email
```

To list all configuration values:

```
git config -l
```

Setup ssh keys for your GitHub account

To make the development more convenient it is recommended to setup ssh keys for your GitHub account.

Detailed instructions on setting up ssh keys can be found here:

Adding ssh keys can be done in the settings of your GitHub account (top right button on GitHub) and in the settings menu there is a "SSH and PGP keys" section.

Check if you already have an ssh key by checking if the following file exists: `~/.ssh/id_rsa.pub` and not empty.

> If yes, create a new ssh key on GitHub and copy the content of `~/.ssh/id_rsa.pub` to the Key section.

If you don't already have a ssh key, create a new one with the following command

```
ssh-keygen -t rsa -b 4096 -C "<my@email>"
```

Then on prompt enter the name of ssh key file, e.g. `~/.ssh/id_rsa`.

Newly created ssh key file should be added to the ssh-agent by commands

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_rsa
```

Afterwards add a new ssh key on GitHub and copy the content of `~/.ssh/id_rsa.pub` to the Key section.

Accessing GitHub from psdev (and other internal networks)

To access GitHub from psdev/psana and other internal networks that lack a direct connection to the Internet, please use psproxy as the ssh/HTTPS proxy.

For SSH access, please edit/create the file ~/.ssh/config and add the following; replacing <your_unix_id> with your unix account id.

```
Host code.stanford.edu
  Hostname code.stanford.edu
  User git
  ProxyCommand ssh <your_unix_id>@psproxy nc %h %p

Host github.com
   Hostname github.com
   User git
   ProxyCommand ssh <your_unix_id>@psproxy nc %h %p
```

NOTE: make sure you "chmod 600 ~/.ssh/config", otherwise ssh will complain, bitterly.

To access GitHub repos using HTTPS, set your Git http.proxy using

```
git config --global http.proxy http://psproxy:3128
```

Useful Commands

Check status of the repo

```
git status
```

Show current and available branches in the repository

```
git branch
```

Show local changes in the code since the last commit

```
git diff
```

Git workflow

The following section describes the workflow for developing software with git.

Clone repository

The first step is to clone the repository you want to contribute by using the flowing command:

```
git clone git@github.com:lcls-psana/<some-repository>.git
```

The repository where you clone from is typically called "upstream". This is the global copy of the repository that all developers share and work on together.

Git clone creates a local copy of the repository in a new folder with the same name as the repository. Any changes that you do to the repository only effect your local copy until you push the changes upstream.

All git commands have to be issued from inside the directory of the git repository.

Make changes to the code

After cloning the repository make changes to the code

Adding new files to the git repo

If you created new file they can be added to git with the following command:

```
git add <file-name>
```

Commit changes

After changing the code these changes have to be committed with:

```
git commit -a -m "commit message"
```

Push commits upstream

After committing the changes into the local repository they have to be pushed upstream to make them available for the other developers.

```
git push origin master
```

This can fail if upstream is ahead of you local version. In this case you first have to get the most recent changes from upstream:

Here is a blog post that explains the difference between merge and rebase

```
git fetch origin
git rebase -p origin
```

Resolving Conflicts

If the local commits and the commits from upstream will modify the same line in the same file, this will result in a merge conflict reported by the above rebase command.

It is simplest to resolve the conflict manually. The command "git diff" will show the lines that were changed locally and by the upstream version. This file can now be added to keep whatever part of the local or remote change you want.

After the merge conflict is resolved commit your changes

```
git commit -a -m "commit message"
```

And then continue with the rebase and push everything upstream

```
git rebase --continue
git push origin master
```

Tag new release

To create a new release from the current version of the local repository and push it upstream do:

```
git tag -a V00-00-01 -m "Version V00-00-01"
git push origin V00-00-01
```

Create new repository

New repositories can be created directly on GitHub in the organization

https://github.com/lcls-psana (green New button on the right)

Afterwards just clone the new repository and the follow the workflow above

Convert svn repository to git

The python 3 code to convert repositories from svn to git is at:

https://github.com/lcls-psana/psdm-svn2git

The script can be run like this to convert a svn repository to git:

```
python svn2git.py -u https://pswww.slac.stanford.edu/svn-readonly/psdmrepo/xtcav
```

References

- Conda Release System
- SSH keys