An intro to Git and GitHub

WHY TO USE VERSION CONTROL SYSTEM IN GISAS DATA ANALYSIS

Gennady Pospelov
Scientific Computing Group at MLZ

17-21 Sep, 2018, GISAS Summer School, Bayreuth/Garching
What is version control?

- A system that keeps records of your changes in files
- Allows you to revert changes and go back to previous state
What is version control?

- A system that keeps records of your changes in files
- Allows you to revert changes and go back to previous state

Version Control vs. Backup system
What is version control?

- A system that keeps records of your changes in files
- Allows you to revert changes and go back to previous state
- Allows for collaborative development
- Allows you to know who made changes and when
What is version control?

- A system that keeps records of your changes in files
- Allows you to revert changes and go back to previous state
- Allows for collaborative development
- Allows you to know who made changes and when

What is Git?

- One of most popular version control systems, industry standard

What is GitHub?

- Website with lots of cloud services build around Git platform
Life without version control

SINGLE USER MODE

To manage changes in documents
- Multiple versions of files, manually created

To face risk of HDD failure
- Regular backups

To use multiple computers to work on a project
- Working folder synchronization

To handle text/code which is no longer needed, but might be needed later on
- Commented pieces of text/code everywhere
Life without version control

COLLABORATIVE ENVIRONMENT

Multiple team members have to edit the same file(s)
- Need to find a place to store the project
- Overwriting may occur, difficult to keep up with the latest file version

Without version control

With version control
What is Git?

- Distributed version control system
  - Tracks changes in computer files
  - Coordinates work among multiple people
- Open-source, free
- Created by Linus Torvalds to aid in Linux Kernel development

Name “git” stands for
- pronounceable random three-letter combination
- global information tracker: when you're in a good mood
- "goddamn idiotic truckload of sh*t": when it breaks

https://git-scm.com/downloads
How does Git work?
How does Git works?

- The purpose of Git is to manage a set of files, as they change over time.
- Git stores this information in the same directory as a project itself.
- User decides when to take directory snapshot and asks Git to do that.

User working directory under Git control.
Key concept: **Commit**

- The act of creating a snapshot
- Can be noun or verb
  - “I commited the code”
  - “I just made a new commit”
- Essentially, a project is made up of a bunch of commits

**Commit object**
- Identifier: ed489
- Parent: da985
- Author: James
- Date: Fri May 22 18:09:34 2009
- Snapshot reference: Version 5
- Message: New chapter added
Key concept: **Repositories**

- Often shortened to ‘repo’
- Contains all change history of the project
  - Consists of all your commits and snapshot trees
- Can live on a local machine, on a remote server

**Cloning**
- The act of copying a repository from a remote server

**Pushing**
- The process of adding local changes to the remote

**Pulling**
- The process of downloading commits from remote
Key concept: Repositories

- Git is distributed version control system
- Every repository has full commit history
- At technical level there is no such thing as central repo

The name *origin* is used for repository which is *considered* as central and used for syncing.
Working with Git
Git can be complicated at first

NAME

git-checkout - Switch branches or restore working tree files

SYNOPSIS

```
git checkout [-q] [-f] [-m] [<branch>]
git checkout [-q] [-f] [-m] --detach [<branch>]
git checkout [-q] [-f] [-m] --detach <commit>
git checkout [-q] [-f] [-m] [--orphan] <new_branch> [<start_point>]
git checkout <tree-ish> [--] <pathspec>.
git checkout [-p|--patch] <tree-ish> [--] <paths>.
```

DESCRIPTION

Updates files in the working tree to match the version in the index or the specified tree. If no paths are given, git checkout will also update HEAD to set the specified branch as the current branch.

git checkout <branch>

To prepare for working on <branch>, switch to it by updating the index and the files in the working tree, and by pointing HEAD at the branch. Local modifications to the files in the working tree are kept, so that they can be committed to the <branch>.

If <branch> is not found but there does exist a tracking branch in exactly one remote (call it <remote>) with a matching name, treat as equivalent to

```
$ git checkout -b <branch> --track <remote>/<branch>
```
Git can be complicated at first

NAME

git-checkout - Switch branches or restore working tree files

SYNOPSIS

```bash
git checkout [ -q ] [ -f ] [ -m ] [ [branch] ]
git checkout [ -q ] [ -f ] [ -m ] --detag [ [branch] ]
git checkout [ -q ] [ -f ] [ -m ] [ [branch] ] --detag <commit>
git checkout [-q] [-f] [-m] [ [-b|--branch|--orphan]:<new_branch> ] [ <start_point> ]
git checkout [-t|--tree-ish] [...] <pathspec>
git checkout [-p|--patch] [-t|--tree-ish] [...] <paths>;
```

DESCRIPTION

Updates files in the working tree to match the version in the index or the specified tree. If no paths are given, git checkout will also update HEAD to set the specified branch as the current branch.

`git checkout <branch>`

To prepare for working on `<branch>`, switch to it by updating the index and the files in the working tree, and by pointing HEAD at the branch. Local modifications to the files in the working tree are kept, so that they can be committed to the `<branch>`.

If `<branch>` is not found but there does exist a tracking branch in exactly one remote (call it `<remote>`) with a matching name, treat as equivalent to

```bash
$ git checkout -b <branch> --track <remote>/<branch>
```

https://git-scm.com/docs

https://git-man-page-generator.lokaltog.net/
SmartGit

Get your commit done.
SmartGit is a graphical Git client with support for SVN and Pull Requests for GitHub and Bitbucket. SmartGit runs on Windows, macOS, and Linux.

https://www.syntevo.com/smartgit/
Demo: creating repositories, recording changes

Create local repository
- Add files to repository
- Modify files, record changes
- Undoing changes
States of the file

Untracked

- Add the file
- Remove the file

Tracked

- Unmodified
- Modified
- Staged
- Edit the file
- Stage the file
- Commit

15
Key concept: **Branches**

- All commits in git live on some branch
- The main branch in a project called master branch
Key concept: **Branches**

- All commits in git live on some branch
- The main branch in a project called master branch
Key concept: **Branches**

- All commits in git live on some branch
- The main branch in a project called master branch
Key concept: **Branches**

- All commits in git live on some branch
- The main branch in a project called master branch
Key concept: **Branches**

- All commits in git live on some branch
- The main branch in a project called master branch
Key concept: Branches

- All commits in git live on some branch
- The main branch in a project called master branch

Article was send for review
Key concept: **Branches**

- All commits in git live on some branch
- The main branch in a project called master branch

Article was send for review
Key concept: **Branches**

- All commits in git live on some branch
- The main branch in a project called master branch

---

Article was send for review

Received comments
Key concept: **Branches**

- All commits in git live on some branch
- The main branch in a project called master branch

![Branches Diagram]

<table>
<thead>
<tr>
<th>Article was send for review</th>
<th>Changes incorporated</th>
</tr>
</thead>
</table>

17
Key concept: **Branches**

- All commits in git live on some branch
- The main branch in a project called master branch
Key concept: conflicts

- In most cases Git will figure out how to integrate changes
- Conflicts occur when
  - Same line of same file was modified
  - File was deleted in one of the branch

![Diagram showing conflict resolution process]
Key concept: conflicts
Binary files in Git

- Binary files can be stored in Git too
- However Git can’t merge binary files in any way which makes sense
- During the merge you will have to make choice which one is “correct”
Demo: creating branches, resolving conflicts

Create feature branch
- Create feature branch
- Modify file, record changes

Create and resolve merge conflict
- Modify same file in master branch
- Try to merge feature branch to master
- Resolve conflict
How to collaborate on GitHub
Create GitHub account

Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 28 million developers.

https://github.com
Create remote repository

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name

Great repository names are short and memorable. Need inspiration? How about fluffy-robot.

Description (optional)

Public

Anyone can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you’re importing an existing repository.

Add .gitignore: None

Add a license: None

Create repository
Remote repository

SINGLE USER SCENARIO

GitHub cloud

Local computers

Three main commands: clone, push, pull
Remote repository

MULTI USERS SCENARIO

GitHub cloud

Local computers

Three main commands: clone, push, pull
New GitHub terminology: fork, pull request
Demo: creating repositories, recording changes

Create remote repository
- Clone it locally
- Modify content
- Push modifications to remote
- Update local from remote

Fork remote repository
- Make a fork of someone’s repository
- Clone it locally
- Make modification, push to origin, provide pull request
BornAgain on GitHub
A software to simulate and fit neutron and X-ray scattering at grazing incidence. [http://www.bornagainproject.org](http://www.bornagainproject.org)

This branch is 609 commits ahead, 6 commits behind master.

<table>
<thead>
<tr>
<th>Branch</th>
<th>Created</th>
<th>Last commit</th>
<th>Description</th>
<th>Changes ago</th>
</tr>
</thead>
<tbody>
<tr>
<td>develop</td>
<td>Nov 12</td>
<td>09:21:17</td>
<td>Merge pull request #529 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
<tr>
<td>master</td>
<td>Nov 22</td>
<td>09:30:43</td>
<td>Merge pull request #520 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
<tr>
<td>new</td>
<td>Oct 28</td>
<td>09:30:43</td>
<td>Merge pull request #519 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
<tr>
<td>gh</td>
<td>Oct 28</td>
<td>09:30:43</td>
<td>Merge pull request #518 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
<tr>
<td>travis</td>
<td>Oct 28</td>
<td>09:30:43</td>
<td>Merge pull request #517 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
<tr>
<td>Core</td>
<td>Oct 28</td>
<td>09:30:43</td>
<td>Merge pull request #516 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
<tr>
<td>Doc</td>
<td>Oct 28</td>
<td>09:30:43</td>
<td>Merge pull request #515 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
<tr>
<td>Examples</td>
<td>Oct 28</td>
<td>09:30:43</td>
<td>Merge pull request #514 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
<tr>
<td>Fit</td>
<td>Oct 28</td>
<td>09:30:43</td>
<td>Merge pull request #513 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
<tr>
<td>GUI</td>
<td>Oct 28</td>
<td>09:30:43</td>
<td>Merge pull request #512 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
<tr>
<td>Tests</td>
<td>Oct 28</td>
<td>09:30:43</td>
<td>Merge pull request #511 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
<tr>
<td>ThirdParty</td>
<td>Oct 28</td>
<td>09:30:43</td>
<td>Merge pull request #510 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
<tr>
<td>Wrap</td>
<td>Oct 28</td>
<td>09:30:43</td>
<td>Merge pull request #509 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
<tr>
<td>auto</td>
<td>Oct 28</td>
<td>09:30:43</td>
<td>Merge pull request #508 from gospelov/develop</td>
<td>2 years ago</td>
</tr>
</tbody>
</table>
BornAgain on GitHub

https://github.com/scgmlz/BornAgain
If you encounter a problem during installation or while running BornAgain or if you have a question related to BornAgain models consider using GitHub issue system.
Workflow is organized according to “A successful Git Branching Model”
BornAgain on GitHub

- BornAgain web site is maintained via git
- Hosted directly on GitHub
- Uses GitHub CI to automatize workflow

https://github.com/scgmlz/BornAgain-website

https://bornagainproject.org
GitHub vs GitLab

- Both are web-based Git repositories
- Designed for use by dev team of all sizes
- Both offer wiki, issue tracking, code review, continuous integration (CI)

<table>
<thead>
<tr>
<th></th>
<th>github.com</th>
<th>gitlab.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Popularity</td>
<td>70.000.000+ projects</td>
<td>500.000+ projects</td>
</tr>
<tr>
<td>Free plan</td>
<td>Unlimited public repo</td>
<td>Unlimited public repo</td>
</tr>
<tr>
<td></td>
<td>No private repo</td>
<td>Unlimited private repo</td>
</tr>
<tr>
<td></td>
<td>No team permissions</td>
<td>Unlimited team</td>
</tr>
<tr>
<td></td>
<td>Unlimited CI usage</td>
<td>Limited CI 2000 minutes monthly</td>
</tr>
<tr>
<td></td>
<td>No self-hosting</td>
<td>Self hosting possible</td>
</tr>
<tr>
<td>Paid plan</td>
<td>7$/month, private repo</td>
<td>4$/month, time tracking</td>
</tr>
<tr>
<td></td>
<td>25$/month, team up to 5</td>
<td>19$/month, CI up to 10000 minutes</td>
</tr>
</tbody>
</table>
Summary

- Install Git and SmartGit
- Create a new local repository
- Clone Git repository
- Make use of Git commit history
- Revert files to previous state
- Create a GitHub repository
- Add, commit and push changes
- Create and merge branches
- Send pull requests
- Resolve conflicts
Thank you!