Introduction to Version Control Systems 2

DATA 4010 Seminar – Fall 2024

Stefano Ansaloni

University of Manitoba

September 25, 2024

About me

Stefano Ansaloni

Cloud Computing Specialist at University of Manitoba
(part of the Advanced Reasearch Computing team)

Software Developer and DevOps Specialist since 2017

Linux User/Admin since 2005

Refresh – What is a Git remote?

A *remote* (or *remote repository*) is a copy of a local repository hosted on the Internet or network somewhere.

Remote repositories are used to ease the collaboration between multiple developers by pushing and pulling data to and from them when they need to share work.

`Pro Git book [2.5 Git Basics - Working with Remotes]`

Refresh – Git Credential storage

When interacting with remote repositories, you could be asked to provide some sort of credentials in order to access those.

For simple cases (i.e. when only a username and password is needed), this can be managed directly by Git.
However, an increasing number of online services is moving to multifactor authentication systems.

To handle that type of authentication, you could use an external Git credential system like *Git Credential Manager* (available at https://github.com/git-ecosystem/git-credential-manager).

Pro Git book [7.14 Git Tools - Credential Storage]

Code hosting platforms

GitHub	https://github.com
GitLab	https://gitlab.com
Bitbucket	https://bitbucket.org
Gitea	https://gitea.com
Codeberg	https://codeberg.org

Note: remember that is your responsibility to read, understand, and decide whether to accept all the policies/user agreements of the service(s) you use.

More than just remote repositories

In addition to remote repositories, those services usually provide:

- access control
- bug tracking
- task management
- continuous integration (CI)
- continuous delivery/deployment (CD)
- wiki
- ... and more

Focusing On GitHub

GitHub

GitHub is a common choice to host code for public open-source projects, or to create private repositories for personal/non-public projects.

It is also the world's largest source code host as of June 2023:

- over 100 million developers
- more than 420 million repositories
- $\sim$ 28 million public repositories

GitHub requires all users who contribute code, to enable one or more forms of two-factor authentication (2FA).

Wikipedia [GitHub]
GitHub [Securing your account with 2FA]

Creating a new GitHub repository

1. In the top-right corner of any page, use the "+" drop-down menu, and select "*New repository*"
2. Type a short name for the new repository (e.g. "first-repo")
3. Choose the repository visibility: "*Public*" or "*Private*"
4. Review the creation options
5. Click "*Create repository*"

After creating a new GitHub repository, it can be added as a remote to a local git repository.
(e.g. "`git remote add <remote_name> <new_github_repo_url>`")

GitHub [Create a repo]

Adding collaborators to a GitHub repository

1. Navigate to the main page of the desired repository
2. Under the repository name, click "*Settings*"
3. In the "Access" section of the sidebar, click "*Collaborators*"
4. Click "*Add people*"
5. Use the search field to search and select a GitHub user
6. Click "*Add <user_name> to <repo_name>*"
7. The selected user will receive an invite email to the repository (once they accept the invitation, they will have collaborator access to the repository)

GitHub [Inviting collaborators to a personal repository]

Forking a GitHub repository

A *fork* is a copy of an existing GitHub repository that shares code and visibility settings with the original GitHub upstream repository (also called “parent repository”).

1. Navigate to the existing repository to be forked
2. In the top-right corner of the page, click “*Fork*”
3. Review the forking options
4. Click “*Create fork*”

`GitHub [Fork a repo]`

Creating a GitHub pull request

A *pull request* (or *PR*) is a way to ask a GitHub upstream repository to integrate changes pushed to a GitHub repository that is a fork of the main one.

Once a *pull request* is opened, maintainers can discuss and review the changes with collaborators and add follow-up commits before the changes are merged into the main GitHub repository.

GitHub [About pull requests]

Creating a GitHub pull request

1. Navigate to the main page of the desired repository fork
2. In the "*Branch*" menu, choose the branch that contains the commits to be integrated upstream
3. Click "*Compare & pull request*" in the yellow banner (above the list of files) to create a PR for the current branch
4. Review the "*base branch*" and the "*compare branch*" (the former represent the destination, the latter is the source for the PR)
5. In the "*Title*" field, type a meaningful title for the pull request
6. In the description body field, explain why the upstream maintainer should accept the pull request
7. Click "*Create Pull Request*" to create a pull request ready for review (or use the drop-down and click "*Create Draft Pull Request*" to create a draft pull request)

GitHub [Creating a pull request]

Git merge conflicts

Merge conflicts happen when merging branches have competing commits, and Git needs human intervention to decide which changes to incorporate in the final merge.

When the changes are on different files (or different lines of the same file), Git can merge them automatically. Usually, merge conflicts happen when different changes are made to the same line of the same file, or when the same file is edited on a branch and deleted on the other merging branch.

All merge conflicts must be resolved before merging a pull request on GitHub. If there is a merge conflict between the "compare branch" and "base branch", the conflicting files are listed above the "Merge pull request" button (which will be also deactivated until all conflicts are resolved).

GitHub [About merge conflicts]

Git merge conflicts resolution

1. Print the list of conflicting files using "`git status`"

2. For each file containing conflicts
   - 2.1 Open the file with a text editor
   - 2.2 Search for the "*conflict markers*" ("`<<<<<<< HEAD`", followed by the changes from the destination branch, followed by "`=======`", followed by the changes from the source branch, followed by "`>>>>>>> <source_branch_name>`")
   - 2.3 Decide what changes to keep (it is possible to incorporate changes from both branches)
   - 2.4 Delete the "*conflict markers*"
   - 2.5 Stage resolved file for commit using "`git add <file_name>`"

3. Commit the resolved files using "`git commit`"

GitHub [Resolving a merge conflict using the command line]

Git merge conflicts resolution example

```
user@host:~/repo$ git merge feat-1
Auto-merging file1
CONFLICT (content): Merge conflict in file1
Automatic merge failed; fix conflicts and then commit the
    result.


user@host:~/repo$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
  both modified:   file1

no changes added to commit (use "git add" and/or "git commit
     -a")
```

Git merge conflicts resolution example

```
user@host:~/repo$ cat file1
<<<<<<< HEAD
ggg
=======
vvv
>>>>>>> feat-1

## Use a text editor to resolve the conflicts ##

user@host:~/repo$ cat file1
I decided to write something else...and this is fine

user@host:~/repo$ git add file1

user@host:~/repo$ git commit -m "Resolve merge conflicts"
```

Creating a GitHub issue

GitHub allows users to create and manage *issues* to track ideas, feedback, tasks, or bugs.

1. Navigate to the main page of the desired repository
2. Under the repository name, click “*Issues*”
3. Click “*New issue*”
4. In the “*Title*” field, type a meaningful title for the issue
5. In the comment body field, type a description of the issue
6. Review the other options
7. Click “*Submit new issue*”

`GitHub [About issues]`
`GitHub [Creating an issue]`

GitHub
Additional Services

Extra – Creating a GitHub gist

A *gist* is a special type of GitHub repository intended to provide a simple way to share code snippets with others (like a pastebin service).

1. In the top-right corner of any page, use the “+” drop-down menu, and select “*New gist*”
2. In the “*Filename including extension*” field, type a file name for the gist (including the file extensions)
3. In the file contents field, type the text of the gist
4. Click “*Create secret Gist*” or “*Create public gist*” (note: secret gists are not private, i.e. anyone who has the gist’s url can access them)

GitHub [Creating gists]

Extra – GitHub Pages

GitHub Pages is a static web hosting services offered by GitHub that can be used to serve websites.

It takes HTML, CSS, and JavaScript files straight from a repository and serves them using a "`github.io`" subdomain.

GitHub Pages can be integrated with Jekyll (https://jekyllrb.com) to generate the website from plain text files.

`GitHub [About GitHub Pages]`

Repositories Common Files

What is a license file

A *license* (or *software license*) is a legal document governing the use or redistribution of software.

Usually the *license* is a text file that resides in the root directory of a project (e.g. “`LICENSE`”, “`LICENSE.txt`”, ...).

When you own the source code of a software, you can choose the license type that best suits your needs.

`Wikipedia [Software license]`
`GitHub [Licensing a repository]`
`Choose-a-license website`

What is a readme file

A *readme file* (or *README*) is a text file that resides in the root directory of a project, containing the following information:

- the purpose of the project
- instructions on how to build/install the project
- guidance on how to use the project
- how to contribute to the project

By ensuring that the *readme file* is comprehensive, developers can better collaborate and maintain the project.

Medium [Why having a Readme]
GitHub [About READMEs]

Formatting a readme file

Since a readme file is just a text file, it does not allow any special formatting by itself.

However, GitHub can render readme files (and in general all text files) when they are written using the Markdown syntax, and have the "`.md`" extension (e.g. "`README.md`").

It is also important to note that GitHub offers an extended version of the basic Markdown syntax, and you can use it when writing readmes, comments and issues.

`Markdown Guide`
`GitHub [Basic writing and formatting syntax]`

Markdown cheat-sheet

Heading	# Biggest Header ... ###### Smallest Header
Bold	**bold text**
Italic	*italicized text*
Blockquote	> quoted text
Unordered list	- item 1 - item 2
Ordered list	1. item 1 2. item 2
Link	[link title](https://github.com)
Inline code	`code`

Multiline code	
	```
	code
	more code
	```

Table	| col 1 hdr | col 2 hdr | | --- | --- | | some text | other text |

Markdown Guide [Cheat Sheet]

GitHub Markdown cheat-sheet

Mentioning people or teams	`@username` `@organization/team_name`

Referencing issues or pull requests	`#issue_or_pr_number`
	`username/repo_name#issue_or_pr_number`
	`organization/repo_name#issue_or_pr_number`

Referencing commits	`commit_sha`
	`username@commit_sha`
	`username/repo_name@commit_sha`
	`organization/repo_name@commit_sha`

GitHub [Basic writing and formatting syntax]
GitHub [Autolinked references and URLs]

GitHub
Command Line Tool

GitHub command line tool

GitHub offers a command line tool called "*GitHub CLI*" (or simply "*gh*") that allows to manage the majority of aspects related to a GitHub account (i.e. managing repositories, forks, pull requests, issues, ...).

GitHub CLI is free, open-source and multiplatform (available for Windows, Mac, Linux).

Installation instructions are available on its official GitHub repository page (https://github.com/cli/cli).

GitHub CLI
GitHub CLI [Manual]

GitHub CLI – Authentication

GitHub CLI needs to be authorized before executing operations on the desired GitHub account.

To authenticate with the GitHub website you can use “`gh auth login`”.

This command will open a web browser to complete the login process (to see the authentication status you can use “`gh auth status`”).

After finishing the login process, you can check the information about your GitHub profile with “`gh status`”.

`GitHub CLI [Authentication]`
`GitHub CLI [Status]`

GitHub CLI – Repositories

To create a GitHub repository you can use "`gh repo create`".

To list your GitHub repositories you can use "`gh repo list`".

To delete a GitHub repository you can use "`gh repo delete <repo_name>`". Omitting "`repo_name`", will delete the repository pointed by the current directory.

To fork a GitHub repository you can use "`gh repo fork <owner/repo_name>`". Omitting "`repo_name`", will create a fork of the repository pointed by the current directory. Moreover, the new fork will be set as the "`origin`" remote and any existing origin remote will be renamed to "`upstream`".

GitHub CLI – Pull requests

To create a GitHub pull request you can use “`gh pr create`”. This command will use the current branch as “compare branch” and the current GitHub repository default branch as “base branch”.

To list the pull requests in the current GitHub repository you can use “`gh pr list`”.

To show the status of the pull requests in the current GitHub repository you can use “`gh pr status`”.

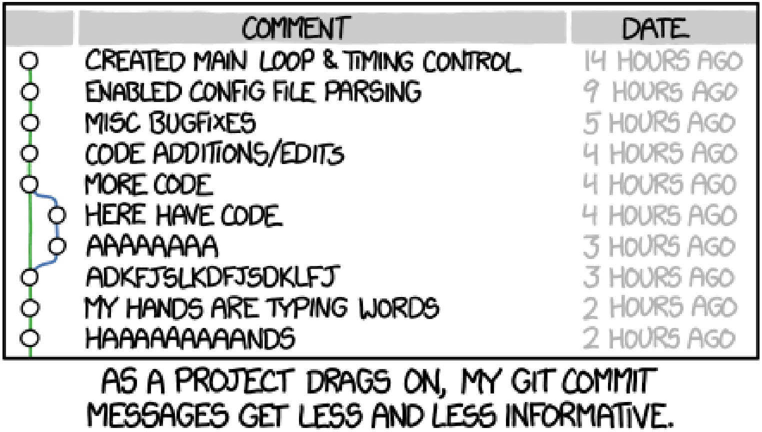
GitHub CLI – Issues

To create a GitHub issue you can use “`gh issue create`”.

To list the issue in the current GitHub repository you can use “`gh issue list`”.

To close a GitHub issue you can use “`gh issue close <issue_number_or_url>`”.

To delete a GitHub issue you can use “`gh issue delete <issue_number_or_url>`”.



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Questions?

Thank you