Git part 1

Gulnaz Shalgumbayeva

2023-08-02

##Key Points Automated Version Control

```
Version control is like an unlimited ‘undo’.
Version control also allows many people to work in parallel.
```

Setting Up Git

```
Use git config with the --global option to configure a user name, email address, editor, and other prefe
```

Creating a Repository

```
git init initializes a repository.
Git stores all of its repository data in the .git directory.
```

Tracking Changes

```
git status shows the status of a repository.
Files can be stored in a project’s working directory (which users see), the staging area (where the nex
git add puts files in the staging area.
git commit saves the staged content as a new commit in the local repository.
Write a commit message that accurately describes your changes.
```

Exploring History

```
git diff displays differences between commits.
git checkout recovers old versions of files.
```

Ignoring Things

```
The .gitignore file tells Git what files to ignore.
```

Remotes in GitHub

```
A local Git repository can be connected to one or more remote repositories.
Use the HTTPS protocol to connect to remote repositories until you have learned how to set up SSH.
git push copies changes from a local repository to a remote repository.
git pull copies changes from a remote repository to a local repository.
```

Collaborating

```
git clone copies a remote repository to create a local repository with a remote called origin automatica
```

Conflicts

```
Conflicts occur when two or more people change the same lines of the same file.
The version control system does not allow people to overwrite each other’s changes blindly, but highligh
```

Open Science

```
Open scientific work is more useful and more highly cited than closed.
```

Licensing

```
People who incorporate General Public License (GPL’d) software into their own software must make their s
The Creative Commons family of licenses allow people to mix and match requirements and restrictions on a
People who are not lawyers should not try to write licenses from scratch.
```

Citation

```
Add a CITATION file to a repository to explain how you want your work cited.
```

Hosting

```
Projects can be hosted on university servers, on personal domains, or on public forges.
Rules regarding intellectual property and storage of sensitive information apply no matter where code an
```

Supplemental: Using Git from RStudio

```
Using RStudio’s Git integration allows you to version control a project over time.
```

Git Cheatsheets for Quick Reference

```
Printable Git cheatsheets in several languages are available here
```

Links to an external site.

https://training.github.com/downloads/github-git-cheat-sheet.pdf

More material is available from the GitHub training website Links to an external site..

https://docs.github.com/en/get-started/quickstart/set-up-git

An interactive one-page visualization

http://ndpsoftware.com/git-cheatsheet.html#loc=index;

Links to an external site. about the relationships between workspace, staging area, local repository, upstream repository, and the commands associated with each (with explanations). Both resources are also available in other languages (e.g. Spanish, French, and more). “Happy Git and GitHub for the useR Links to an external site.”

https://happygitwithr.com/

is an accessible, free online book by Jenny Bryan on how to set up and use Git and GitHub with specific references on the integration of Git with RStudio and working with Git in R. Open Scientific Code using Git and GitHub

https://open-source-for-researchers.github.io/open-source-workshop/

Links to an external site. - A collection of explanations and short practical exercises to help researchers learn more about version control and open-source software.

##Glossary

```
changeset - A group of changes to one or more files that are or will be added to a single commit in a ve
commit - record the current state of a set of files (a changeset) in a version control repository. As a
conflict - A change made by one user of a version control system that is incompatible with changes made
HTTP - The Hypertext Transfer Protocol used for sharing web pages and other data on the World Wide Web.
merge - (a repository): To reconcile two sets of changes to a repository.
protocol - A set of rules that define how one computer communicates with another. Common protocols on th
remote - (of a repository) A version control repository connected to another, in such a way that both ca
repository - A storage area where a version control system stores the full history of commits of a proje
resolve - To eliminate the conflicts between two or more incompatible changes to a file or set of files
revision - A synonym for commit.
SHA-1 - SHA-1 hashes is what Git uses to compute identifiers, including for commits. To compute these, G
SSH - The Secure Shell protocol used for secure communication between computers.
timestamp - A record of when a particular event occurred.
version control - A tool for managing changes to a set of files. Each set of changes creates a new commi
```

Links to an external site. of the files; the version control system allows users to recover old commits reliably, and helps manage conflicting changes made by different users.

Setting up Git

PS1="$"

$git config –global user.name "Gulnaz Shalgumbayeva"

$git config –global user.email "gshalgum@uci.edu"

$git config –global user.core.autocrlf input

$git config –global user.editer "nano -w"

$git config –global init.defaultBranch main

$git config –global list

Creating a Repository

Questions

```
Where does Git store information?
```

Objectives

```
Create a local Git repository.
>Describe the purpose of the .git directory.
```

Key Points

$ git init initializes a repository.

Git stores all of its repository data in the .git directory

$cd desktop

$pwd

/Users/gulnazshalgumbayeva/desktop

$mkdir planets

$cd planets

$pwd

/Users/gulnazshalgumbayeva/desktop/planets

$git init

Initialized empty Git repository in /Users/gulnazshalgumbayeva/Desktop/planets/.git/

$ls -a

. .. .git

Comment: You see this hidden directory that tracks the history of your project, if you ever delete this directory .git then you will lose all the project history!!!

$git checkout -b main

Switched to a new branch 'main' $git status

Warning: By running git init in the moons subdirectory, you are telling Git to start tracking the files in that subdirectory and create a new repository that is independent of the main repository in the planets directory.

Exercise: Wolfman explains to Dracula how a nested repository is redundant and may cause confusion down the road. Dracula would like to remove the nested repository. How can Dracula undo his last git init in the moons subdirectory?

But be careful! Running this command in the wrong directory, will remove the entire Git history of a project you might want to keep. Therefore, always check your current directory using the command pwd

pwd git rm -r moons

Tracking Changes

Questions

```
How do I record changes in Git?
How do I check the status of my version control repository?
How do I record notes about what changes I made and why?
```

Objectives

```
Go through the modify-add-commit cycle for one or more files.
Explain where information is stored at each stage of that cycle.
Distinguish between descriptive and non-descriptive commit messages.
```

$pwd

/Users/gulnazshalgumbayeva/desktop/planets

$nano mars.txt

$ls

mars.txt

$cat mars.txt

Cold and dry, but everything is my favorite color.

$git status

On branch main

No commits yet

Untracked files: (use “git add . . . ” to include in what will be committed) mars.txt

nothing added to commit but untracked files present (use “git add” to track)

$git add mars.txt

$git status

On branch main

No commits yet

Changes to be committed: (use “git rm –cached . . . ” to unstage) new file: mars.txt

comment: Git knows that has to keep track on mars.txt but has not yet recorded because no commit

$git commit -m “Start notes on Mars s a base”

[main (root-commit) d47b85c] Start notes on Mars s a base 1 file changed, 1 insertion(+) create mode 100644 mars.txt

comment: Git takes everything(that you told to do), saves in .get using get add, stores copy in get directory that is get commit identifier is d47b85c]

git nano will show you text editor to write longer message

$git status

On branch main

No commits yet

Changes to be committed: (use “git rm –cached . . . ” to unstage) new file: mars.txt

$git commit -m “Start notes on Mars s a base” [main (root-commit) d47b85c] Start notes on Mars s a base 1 file changed, 1 insertion(+) create mode 100644 mars.txt $git status On branch main nothing to commit, working tree clean

git log will show all history

$nano mars.txt

cooment: I added sentence

$git status

On branch main Changes not staged for commit: (use “git add . . . ” to update what will be committed) (use “git restore . . . ” to discard changes in working directory) modified: mars.txt

no changes added to commit (use “git add” and/or “git commit -a”) comment: We want to save the changes use git add

$git add mars.txt

$git commit -m "Added information about Mars moons"

[main 4dc52e0] Added information about Mars moons 1 file changed, 2 insertions(+)

Comment: git diff was not showing anything, just wait for 5 -10 min it takes a time to sync

git status shows the status of a repository.

Files can be stored in a project's working directory (which users see), the staging area (where the next commit is being built up) and the local repository (where commits are permanently recorded).

git add puts files in the staging area.

git commit saves the staged content as a new commit in the local repository Write a commit message that accurately describes your changes.

$cat mars.txt to see the text in mars.txt git diff

diff –git a/mars.txt b/mars.txt index 2878051..5fb7635 100644 — a/mars.txt +++ b/mars.txt @@ -1,3 +1,3 @@ Cold and dry, but everything is my favorite color. -The two moons could be a problem.
- +The two moons could be a problem. + But one researcher will appreciate the lack of humidity.

explanation:

-1 +3 means first file has 3 lines

+1, 3 means added one and it has 3 lines now

Now we need to put those changes so it can report

$ git add mars.txt

$git diff –staged

$git commit -m "Discuss concers about Mars climate"

[main c8d4549] Discuss concers about Mars climate 1 file changed, 2 insertions(+), 3 deletions(-)

$git status

On branch main nothing to commit, working tree clean

Git does not track Directory, only files within them.

$mkdir spaceships

$git status

On branch main nothing to commit, working tree clean

$git add spaceships

$git status

On branch main nothing to commit, working tree clean

comment: Our newly created directory will not under list of files even though we explicitly git add to our repository, this is the reason we need git keep files otherwise empty directories

Exercise: Create a new Git repository on your computer called bio. Write a three-line biography for yourself in a file called me.txt, commit your changes Modify one line, add a fourth line Display the differences between its updated state and its original state.

$mkdir bio $cd bio $ git init $nano me.txt $ git add me.txt $ git commit -m "Added some info about me" git diff

Exploring History

Questions

```
How can I identify old versions of files?
How do I review my changes?
How can I recover old versions of files?
```

Objectives

```
Explain what the HEAD of a repository is and how to use it.
Identify and use Git commit numbers.
Compare various versions of tracked files.
Restore old versions of files.
```

Key points:

```
git diff displays differences between commits.
git checkout recovers old versions of files.
```

$git diff HEAD mars.txt

diff –git a/mars.txt b/mars.txt index 5fb7635..86acdb6 100644 — a/mars.txt +++ b/mars.txt @@ -1,3 +1,7 @@ Cold and dry, but everything is my favorite color. The two moons could be a problem. But one researcher will appreciate the lack of humidity. +An ill considered change.

git diff HEAD will show the last change where git diff HEAD~1 mars.txt it will show previous and we can use git diff HEAD ~so on to see previous changes

$git show HEAD~3 mars.txt

to remove the last line (last commit) $git checkout HEAD mars.txt

comment: can be dangerous will restore ALL files in the current directory if you do git checkout HEAD without spesifying file name

$cart mars.txt

git revert [commit ID]

is a Git command used to create a new commit that undoes the changes introduced by a specific commit. It allows you to safely correct mistakes in your Git history without rewriting commits. When you use git revert, Git generates a new commit that effectively removes the changes made in the specified commit while keeping the rest of the commit history intact.

Ignoring things

Questions

```
How can I tell Git to ignore files I don’t want to track?
```

Objectives

```
Configure Git to ignore specific files.
Explain why ignoring files can be useful
```

Key points: The .gitignore file tells Git what files to ignore.

$mkdir results $touch a.dat b.dat c.dat results/a.out results/b.out $git status comment: we will see untracked 3 files and 1 folder

nano .gitignore comment: type in text editor which items you want to ignore

*.dat results/

comment: Control + o enter control + X To see which files you are ignoring

cat .gitignore git status

Make sure to add and commit git add .gitignore git commit -m " Ignore data files and the results folder" git status comment: gitignore helps us adding files accidentally to our repository. try git add a.dat (it will not add) to see the ignored items git status –ignored

exercise: How would you ignore all .dat files in your root directory except for final.dat? Hint: Find out what ! (the exclamation point operator) *.dat !final.dat

exercise: Let us assume you have many .dat files in different subdirectories of your repository. For example, you might have:

results/a.dat data/experiment_1/b.dat data/experiment_2/c.dat data/experiment_2/variation_1/d.dat

How do you ignore all the .dat files, without explicitly listing the names of the corresponding folders?

recall data/** / *.dat matches all .dat files within the data directory and its subdirectories, no matter how deep they are nested.

nano .gitignore results/*.dat data/** / *.dat