MacEwan UNIVERSITY

Practical Programming Methodology

CMPT201 F25

Tools: Version Control, Git

Mahmoud Elsaadany, PhD, PEng

Outlines

- Version (Source) Control Systems: Why?
- Version Control Systems (VCS).
- Git Vs GitHub
- Git/GitHub: Principles

Version (Source) Control Systems: Why?

- Scenario 1:
- Your program is working
- You change “just one thing”
- Your program is not working anymore ☺
- You change few things back
- Your program is still not working

Solution→ Rollback to the code before the modification

Version (Source) Control Systems: Why?

- Scenario 2:
- Your program worked well
- You made many improvements
- You haven't gotten them to work yet
- You need to Demo your program to the client now

Solution→ Rollback to the code before the modification

Version (Source) Control Systems: Why?

- Scenario 3:
- You changed one part of a program → it works
- Your co-worker changes another part →it works
- You intergrate them together → it doesn't work

Solution→ go back to the code before the merging and work on the compatibility.

Version (Source) Control Systems: Why?

- Scenario 4:
- You make many improvements to a header file/function.
- Your co-worker makes many different improvements to the same header file/function.

Solution→ merging and check the compatibility.

Version (Source) Control Systems

- Objectives

- Keep track of multiple (older and newer) versions of everything (not just source code)
- Request comments regarding every change.
- Allow “check in” and “check out” of files so you know which files someone else is working on
- Display differences between versions.

Version (Source) Control Systems

- Benefits (Working Alone)
  - You can go back to earlier versions
  - You can support different versions (standalone, web app, etc.) of the same basic project.

- Benefits (Working with others)
  - Greatly simplifies concurrent work.
  - Ability to merge changes.

Git Vs GitHub

- **Git:**
  - a free and open-source distributed version control system.
  - Designed to handle small/very large projects with speed and efficiency.
- **GitHub:**
  - is a web-based Git repository hosting service.
  - Offers all distributed revision control and source code management (SCM) functionality of Git.
  - Add some features.

Repository

- Top-level working directory contains:
  - Many subdirectories, source code, binaries, documentation, data files, etc.
  - One subdirectory, named **.git**, saves repository objects.
- Any time, we take a "snapshot", called a commit object
- The commit object
  1. Contains a set of files
  2. Commit objects do not require huge amounts of memory
  3. References to the "parents" of the commit object.
  4. A unique "SHA1" name (secure hashing algorithm).

Repository

- When you **clone** an existing project, you created a local repository
- The repository is a subdirectory named **.git** containing various files
- The dot indicates a “hidden” directory
- Cannot work directly with the contents of that directory;
- various **git** commands do that for you

GitHub Link to student server

- Create an account on github.com (get a username using email)
- **git config --global user.name** “Your userName"
- **git config --global user.email** "youremail you used during the signup"
- **ssh-keygen -t rsa -b 2048**

Please watch the video to understand how to clone a private repository to a directory in the student server

Git/GitHub: Principles

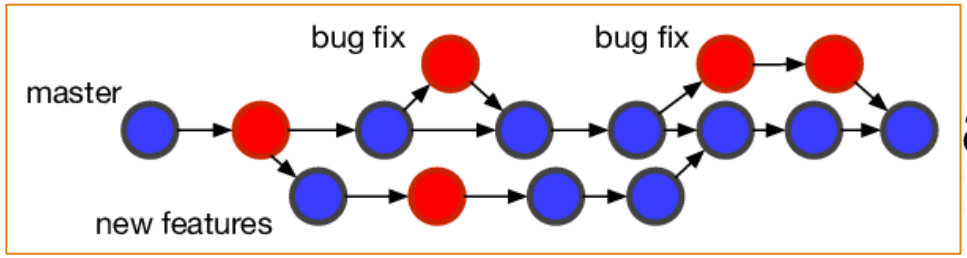
- commit

This means you take a snapshot of the current directory situation

```
git add
git commit –m “comment title” –m ”change description”
```

To reflect your local changes into the remote repository

```
git push
```



acEwan
IVERSITY

Git/GitHub: Principles

Commit objects and graphs

- Commit changes to git → creates a commit object
- A commit object → A complete state of the project

  (including all the files in the project)
- The first commit obj → **has no "parents"**
- The original commit obj → is the **parent of the new** commit obj
- Most commit objects → have a **single-parent**
- **Merge** two commit objs →New commit obj has **two parents**

Git/GitHub: Principles

Commit messages

- Commits without cost, Do them often.
- With commit, provide a one-line message
  - Bad message: “Fixed some bugs”
  - Better message: “Corrected the calculation of Avg”

- Commit messages should be very helpful (to yourself and your team members)

[Hint: You can’t say much in one line, so commit often]

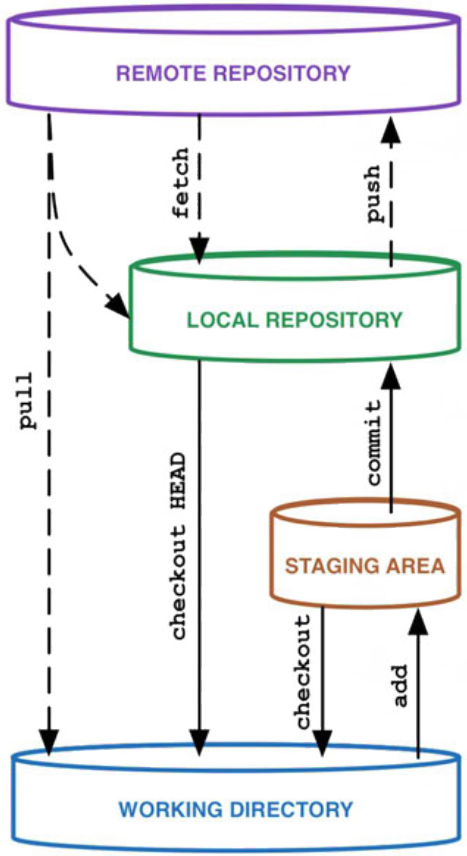
Git/GitHub: Principles

- Typical workflow
- git status
  - See what Git thinks is going on
  - Use this frequently!
- Work on your code

> git add your_edited_files

- git commit –m “Useful comments”
- git push → to save on the host

(remote repository)



Git/GitHub: Principles

Efficient Approach

1. Make sure you are current with the central/main repository (not a branch)
2. Make some improvements to your code.
3. Update the central repository before anyone else does

This approach → no resolving conflicts or working with multiple branches

"All the complexity in git comes from dealing with branches and conflicts"

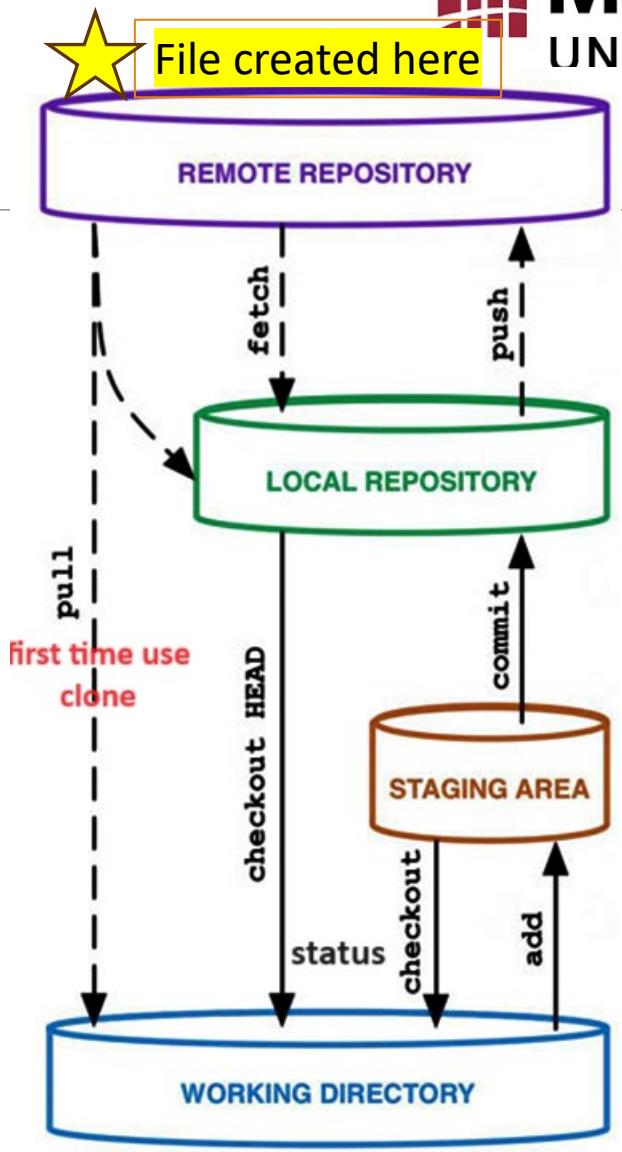
Hints:

- ❑ Make sure you are up-to-date before starting to work.
- ❑ Commit and update the central repository frequently.

Git/GitHub: Principles

GitHub created file

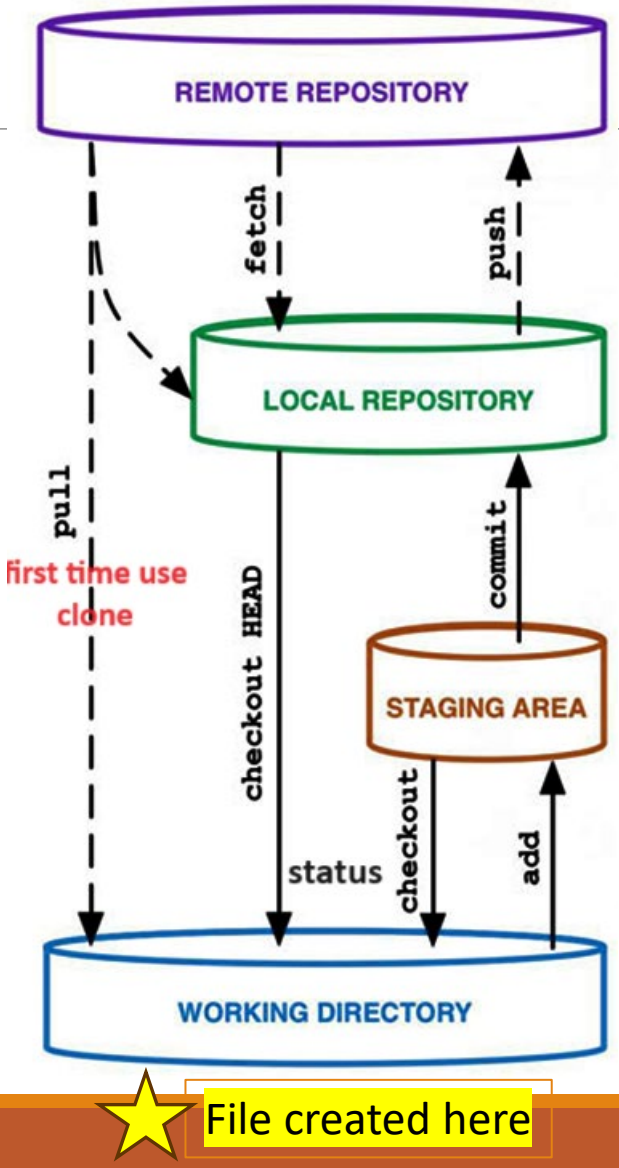
1. Create a private repository.
2. Create files inside
3. Edit the files and commit
4. Open a local directory (on the student server)
5. Inside the directory write the command
   - git clone RemoteRepAdress
6. From this step on any modification can be committed to local or remote repository



Git/GitHub: Principles

Locally created file

1. Create a directory and file inside. (recommended to add README.md)
2. Initialize the directory and commit with the commands
   - git init → create .git subdirectory
   - git add filename → linked "filename to staging area"
   - git commit filename → saving "filename" in the local rep
3. Create a destination directory on GitHub and copy the Rep address
4. Define a remote repository with the command
   - git remote add origin RepAddress
5. Define a standard upstream
   - git push –u origin master
6. From this step on any modification can be committed to the local or remote repository



Git/GitHub: Principles

Ignore files/directory

1) create a .gitignore file with following command
   - touch .getignore

2) add the names of the files to ignore (use vim .getignore and list all the files/directories that you need to ignore)

- *.o → ignore any object file
- *.txt → ignore personal txt files in the project
- test/ → ignore directory test

Git/GitHub: Principles

Ignore files/directory: Example

```
shamseldins@students:~/LocalRepEx> ls
README.md
shamseldins@students:~/LocalRepEx> touch AB.txt
shamseldins@students:~/LocalRepEx> touch AB.o
shamseldins@students:~/LocalRepEx> touch AB.md
shamseldins@students:~/LocalRepEx> mkdir Test
shamseldins@students:~/LocalRepEx> cd Test && touch XY.txt
shamseldins@students:~/LocalRepEx/Test> touch XY
shamseldins@students:~/LocalRepEx/Test> ls
XY  XY.txt
shamseldins@students:~/LocalRepEx/Test> cd..
shamseldins@students:~/LocalRepEx> ls
AB.md  AB.o  AB.txt  README.md  Test
shamseldins@students:~/LocalRepEx> git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        AB.md
        AB.o
        AB.txt
        Test/

nothing added to commit but untracked files present (use "git add" to track)
shamseldins@students:~/LocalRepEx>
```

We created multiple file and one directory

Git/GitHub: Principles

Ignore files/directory: Example

```
shamseldins@students:~/LocalRepEx> touch .gitignore
shamseldins@students:~/LocalRepEx>
```

students.cs.macewan.ca - PuTTY

```
*.txt
*.o
Test/

~
~
-- INSERT -- 4,1         All
```

.gitignore

We added a list with what we need to ignore

Git/GitHub: Principles

Ignore files/directory: Example

```
shamseldins@students:~/LocalRepEx> touch .gitignore
shamseldins@students:~/LocalRepEx>
```

```
shamseldins@students:~/LocalRepEx> vim .gitignore
shamseldins@students:~/LocalRepEx> vim .gitignore
shamseldins@students:~/LocalRepEx> git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be co
        .gitignore
        AB.md

nothing added to commit but untracked files present (use
shamseldins@students:~/LocalRepEx>
```

Now we see the non-ignored files only

Git/GitHub: Principles

Ignore files/directory: Example

```
shamseldins@students:~/LocalRepEx> touch .gitignore
shamseldins@students:~/LocalRepEx>

shamseldins@students:~/LocalRepEx> git add .gitignore AB.md
shamseldins@students:~/LocalRepEx> git commit .gitignore AB.md -m "new file
created"
[master c4055f5] new filecreated
 2 files changed, 4 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 AB.md
shamseldins@students:~/LocalRepEx> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 80 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 335 bytes | 83.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:shokryshams/DestRep1.git
   c0500fa..c4055f5  master -> master
shamseldins@students:~/LocalRepEx>
```

Git/GitHub: Principles

Remove and move

git rm filename → remove file (delete) “affect the local rep”

git mv filename dir/NewName → Move filename from “affect the local rep”

Note you can do this in two steps

rm filename → remove form working directory

git add filename → reflect the action to the local Rep

In both cases to apply the changes to the remote Rep→ we use

git push

Git/GitHub: Principles

Log reading

git log → to read a list with the commit actions

To get outside the log read press q

```
commit c4055f5b7686f7c8c3e41fcd68778cb4efccd5a6 (HEAD -> master, origin/master)
Author: shokryshams <shamseldins@macewan.ca>
Date:   Thu Nov 2 19:44:33 2023 -0600

    new filecreated

commit c0500fad34f92df2ea6eba590e240c79bed1d307
Author: shokryshams <shamseldins@macewan.ca>
Date:   Thu Nov 2 12:32:10 2023 -0600

    One line added

commit aea80609f9098a703d74672fdad187a3d97871f2
Author: shokryshams <shamseldins@macewan.ca>
Date:   Thu Nov 2 12:28:09 2023 -0600
```

Git/GitHub: Principles

Log reading

git log --oneline → more readable format

git log --graph → shows a graph with the workflow

git log --graph --oneline → shows a compressed graph with the workflow

Git/GitHub: Principles

- Branch & merge

git branch NewBranchName → open new branch

git checkout NewBranchName → Switch to the new branch

The above two commands can be replaced by one command :

git checkout -b NewBranchName → open and switch to a new branch

Git/GitHub: Principles

Branch & merge

git branch → list all existing branches

git branch --list → list all branch names

git branch -a list → remote and local branches

git branch -d branchName →delete branch (only if it's merged)

git branch -D branchName →delete branch (even if not merged)

git merge branchName →merge branchName into current branch

Git/GitHub: Principles

Branch Adding collaborators