Object-Oriented Programming

Git Basics

Michael L. Collard, Ph.D.

Department of Computer Science, The University of Akron

Git Local Repository Setup

Create local repository from remote with default directory	`git clone https://github.com/UACPSC/OOPS24-Rainfall-010.git`
Create local repository from remote with named directory	`git clone https://github.com/UACPSC/OOPS24-Rainfall-010.git Rainfall`

Git Workflow 1

Update local repository from remote repository (e.g., GitHub)	`git pull`
Edit files locally, e.g., add a header comment to the file *rainfall.cpp*	
Commit current changes to the local repository	`git commit –am "Add a header comment"`
Update remote repository from local repository	`git push`
View commit log	`git log`

Git Workflow 2

Update local repository from remote repository (e.g., GitHub)	`git pull`
Edit files locally, e.g., add a header comment to the file *rainfall.cpp*	
Stage commits in the local repository	`git add rainfall.cpp`
Commit staged changes to the local repository	`git commit -m "Add a header comment"`
View current staged and modified files	`git status`
View diff of modified files	`git diff`
View diff of staged files	`git diff --staged`
Update remote repository from local repository	`git push`

Workflow Notes

```
git pull
# edit, build, run, test/check
git commit -am "Add a header comment"
git push

git pull
# edit, build, run, test/check
git add rainfall.cpp
git commit -m "Add a header comment"
git status
git diff
git diff --cached
git push
```

- Only have to `git pull` once per session (but no harm in doing so multiple times)

- You can wait to `git push`, e.g., if you are offline

- Use this workflow *as* you develop, not just at the end

Commits

```
git pull
# edit, build, run, test/check
git commit -am "Add a header comment"
git push
```

- Commit as you finish parts of the work, *not just at the end of the work*.

- In general, you cannot commit too often but try to commit complete thoughts (something you can write a good commit message for)

- After every commit, the project should still build and work

Commit Messages

- Think about the commit messages you are using. Note that they are commands (imperative). They finish the sentence:

  If applied, this commit will <commit message>

- E.g., *If applied, this commit will* **Add a header comment**

- How to Write a Git Commit Message

Git Tags

- Way to mark particular points in development, basically a ref (reference) to a commit
- You cannot commit to a tag
- Much of project build and distribution is dependent on tags, e.g., GitHub Releases
- We will use *annotated* tags
- Tags are cheap and easy to create, delete, and therefore easy to change
- Always verify the tag is present at GitHub

Tag: v1a

Action	Command	Notes
List tags	`git tag`	List all tags
Create annotated tag	`git tag -a v1a -m "Completed v1a"`	The `-a` flag creates an annotated tag, and `-m` supplies the tag message
Push tag to remote	`git push origin v1a`	Push **only** the tag named `v1a` to the remote named `origin`
Checkout a tag	`git checkout v1a`	Check out the commit pointed to by `v1a`
Delete local tag	`git tag -d v1a`	Delete the tag **only in your local repo**
Delete remote tag	`git push origin --delete v1a`	Remove the tag `v1a` from the remote `origin`