Git for version control and collaboration
Part 1

Katia Oleinik
Research Computing Services
Outline

- Motivation
- Using Git for version control
- Collaboration using Git
- GitHub and other remote repositories
Big Picture

stash

working space

staging area (index)

remote repository

local repository
Setting up git (~/.gitconfig)

$ module load git

$ git config --global user.name "Katia Oleinik"
$ git config --global user.email koleinik@bu.edu
$ git config --global core.editor "vim"
    "emacs -nw"
    "nano" (or gedit)

$ git config --list [--global / --local]
Getting help

$ git help verb

$ man git-verb

$ git verb -h

Example: $ git config -h
Creating a local repository

- New directory/project
  git init dirname

- Existing directory
  cd /path/to/dirname
  git init

- Cloning local repository
  git clone /project/scv/dirname

- Cloning remote repository
  git clone https://github.com/bu-rcs/newpkg.git
Exploring git repository

Git keeps all of its info in one place: your `.git` directory in your project’s root:
```
tree .git
```

Check current status of your repository
```
git status
```

View history of commits
```
git log
```

Execute this commands often
Main workflow for version control

- stash
- working space
- staging area (index)
- local repository
- remote repository
Main workflow for version control

1. `git add file1 [file2 file3 ...]`
   `git add .`

2. `git commit -m "commit message"
   `git commit`
Check status of the repository

Check current status of your repository
git status

Execute this commands often
View history of commits
git log

View git directory
tree .git

List contents of a tree object
git ls-tree master .
.gitignore file

- can list file names and patterns
- patterns apply to all subdirectories, while file names - to the current directory
- each sub-directory can contain its own .gitignore file
- .gitignore file(s) should be committed
Add tag to your commit

Check log of your repository
git log

Add tag to a specific commit
git tag -a v0.1 sha1

Check log of your repository
git log
deleting and renaming files

After deleting or renaming a file, it has to be added to the staging area and then committed:

```
rm filename

git add filename

git commit -m 'deleted filename'
```
deleting and renaming files

Similarly:

mv file1 file2

git add file1 file2

git commit -m 'renamed file1 into file2'
Exploring the differences/changes

- working space
- staging area (index)
- local repository

- `git diff`
- `git diff --cached`
- `git diff HEAD`
- `git diff 07c0080b`
Remove files from staging area

Remove a single file from staging area
`git reset HEAD -- /path/to/file`

Unstage all file
`git reset`
Review the history

```
git log               # show the list of commits
git log -3            # show the list of the last 3 commits

git show sha1         # show information about specific commit
```

There are many options (can be combined):
```
git log --graph
git log --oneline
git log --stat
git log -p
```
Alias for git log

# non-colored version

git log --graph --pretty=format:'%h%Creset -%d%Creset %s (%cr) <%an>%Creset' --abbrev-commit

# colored version

'%C(red)%h%C/reset) -%C(yellow)%d%C(reset) %s %C(green)(%cr) %C(bold blue)<%an>%C(reset)'

git log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold cyan)%aD%C(reset) %C(bold green)(%ar)%C(reset)%C(bold yellow)%d%C(reset)%n'' %C(white)%s%C(reset) %C(dim white)- %an%C(reset)' --all

# create alias

git config --global alias.lg "log --all --decorate --oneline --graph"
Filtering logs

#Search commits with specific file(s) modified
```
git log -- file1 file2
```

#Filter by date
```
git log --after="2019-1-1" --before="2019-3-24"
```

#Filter by author
```
git log --author="Katia\|Brian"
```

#Search commit messages
```
git log --grep="delete"
```
View file source in a commit

- `git show HEAD:filename`  # source in the last commit
- `git show 0721696:filename`  # source in a specific commit
- `git annotate filename`  # show who made changes to a file
Travelling in time

undo staging

- git reset
- git reset -- filename

working space

staging area (index)

local repository

discard changes

- git checkout HEAD
- git checkout -- filename
Travelling in time

working space

staging area (index)

working space

staging area (index)

e5678

d4567
c3456
b2345
a1234

master branch

git checkout c3456
Travelling in time

- working space
- staging area (index)
  - e5678
  - d4567
  - c3456
  - b2345
  - a1234

master branch

git checkout master
Remote repository

To publish repository to a GitHub (or other place):

1. Create a repository in GitHub

2. In your working directory run:

```
git remote add origin https://github.com/katgit/image_prep.git
git push -u origin master
```
Collaboration

To practice collaboration we will create remote repositories on the SCC and will practice updating them.

collaborator #1 creates a "bare" repository:
```
cd /projectnb/scv/GIT
git init --bare --shared name.git
```

Then collaborator #1 clones it in another directory
```
cd /to/my/own/dir
git clone /projectnb/scv/GIT/name.git
```
Collaboration

collaborator # 1 in his directory creates a text file README and add a few lines
emacs -nw README
git add README
git commit -m"add README file"
git push origin master

collaborator # 2 clones this repository
cd /to/my/own/dir
git clone /projectnb/scv/GIT/name.git
cd name
Collaboration

Both create 1 file with a different name (file1 and file2) and then add them to repo

touch file1
git add file1
git commit -m"add file1"

Each tries to execute the following command

git push origin master

One will succeed the other one will fail.

! [rejected] master -> master (fetch first)
error: failed to push some refs to '/projectnb/scv/GIT/ka.git/'
Collaboration

The collaborator who had an error runs:
- `git pull origin master`
- *within the editor* edits merge commit message
- `git push origin master`

The other collaborator pulls new changes:
`git pull origin master`
Collaboration

Both edit the same README file
nano README
git add README
git commit -m "Katia fixed README"

Each tries to execute the following command
git push origin master

One will succeed the other one will fail.

! [rejected] master -> master (fetch first)
error: failed to push some refs to '/projectnb/scv/GIT/ka.git/'
Collaboration

The collaborator who had an error runs:

- `git pull origin master`
- `edit README file to remove conflict`
- `git add README`
- `git commit -m "merged conflict in README file"`
- `git push origin master`

The other collaborator pulls new changes:

`git pull origin master`
The End