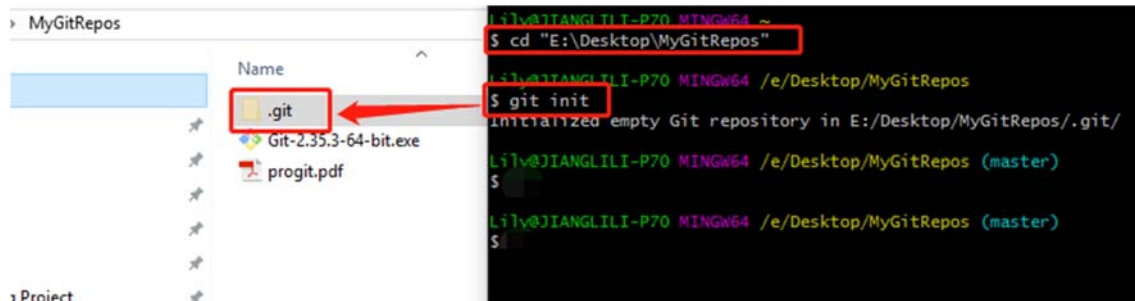


1. Git common commands operation practice

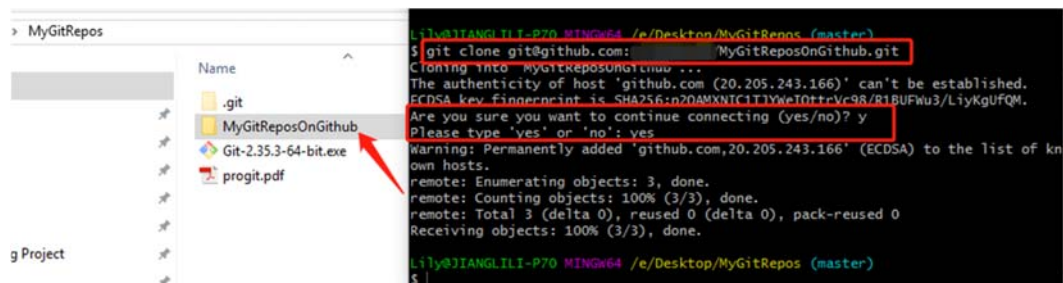
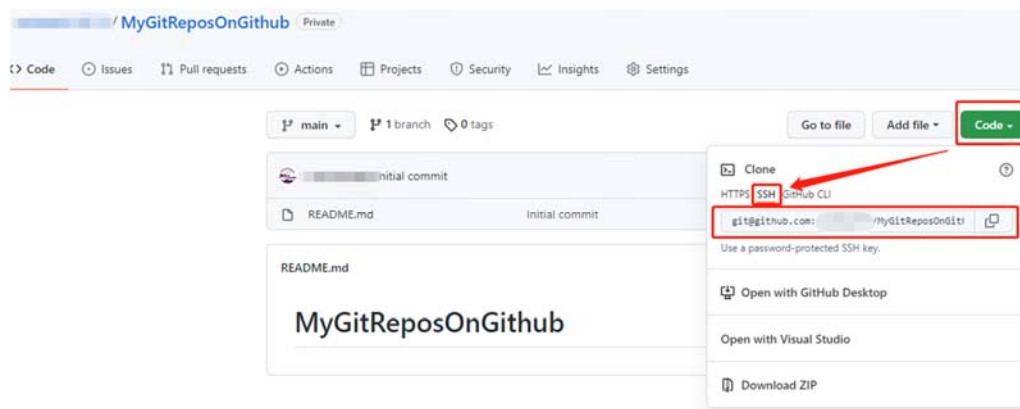
```
$ git init
```

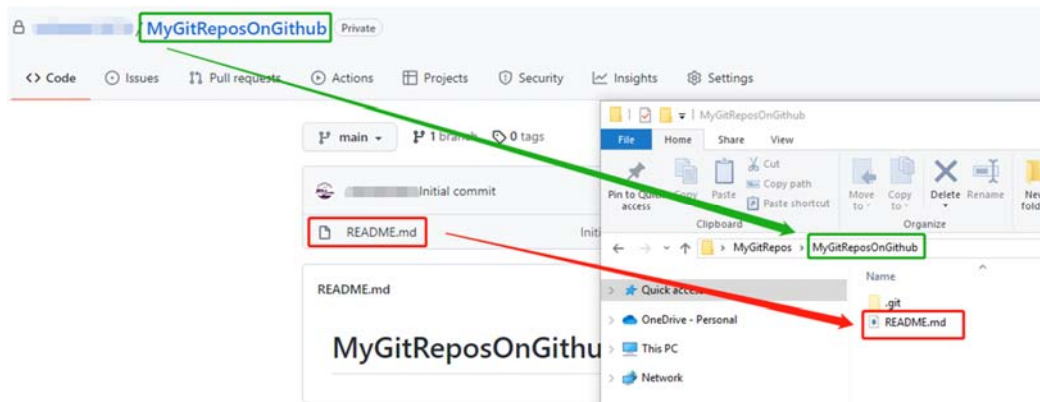
This creates a new subdirectory named `.git` that contains all of your necessary repository files — a Git repository skeleton. At this point, nothing in your project is tracked yet.



```
$ git clone git@github.com:xxx/MyGitReposOnGithub.git
```

With this method, you can quickly get a read/write Git server up and running for a handful of developers.



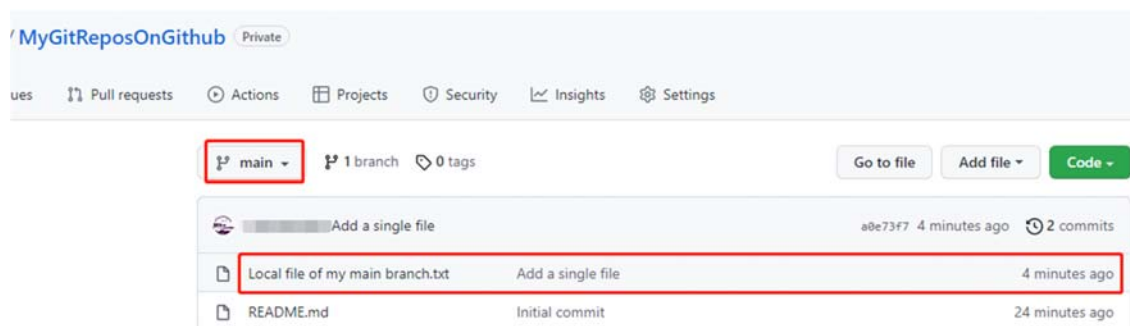
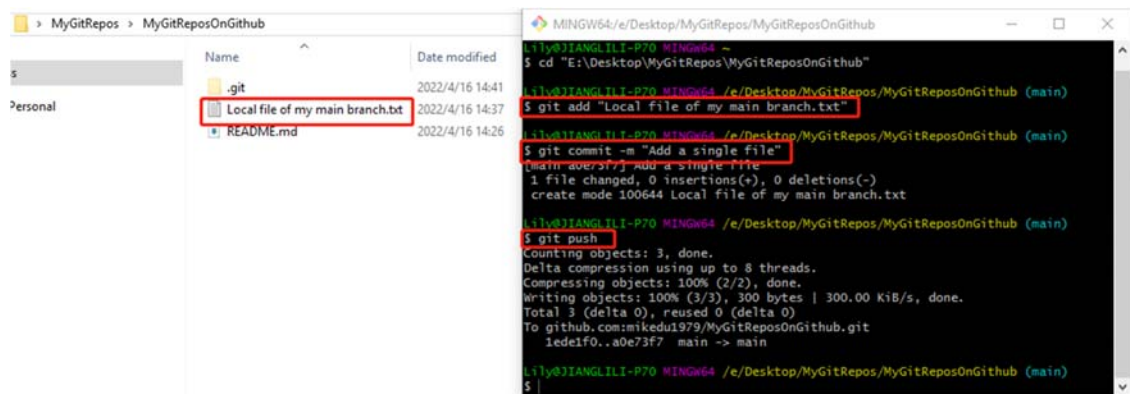


```
$ git add "Local file of my main branch.txt"
```

```
$ git commit -m "Add a single file"
```

```
$ git push
```

Submit a single local file to the GitHub repository

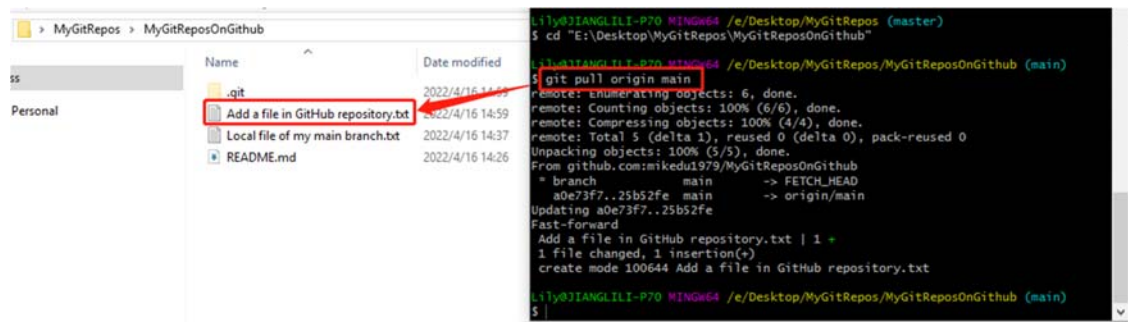
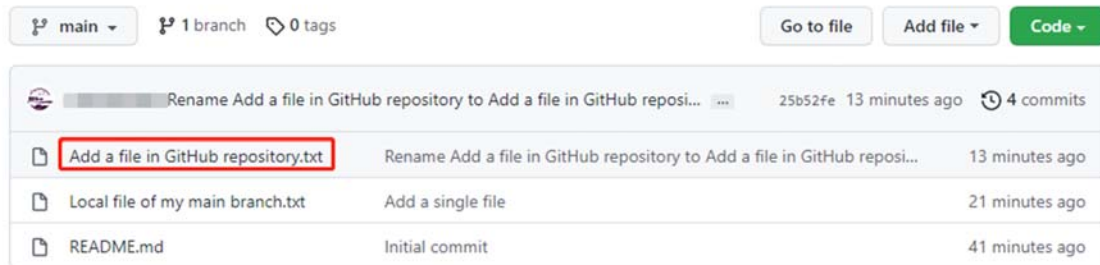


```
$ git add .
```

Add all changed files in the current directory to the temporary storage area

```
$ git pull origin main
```

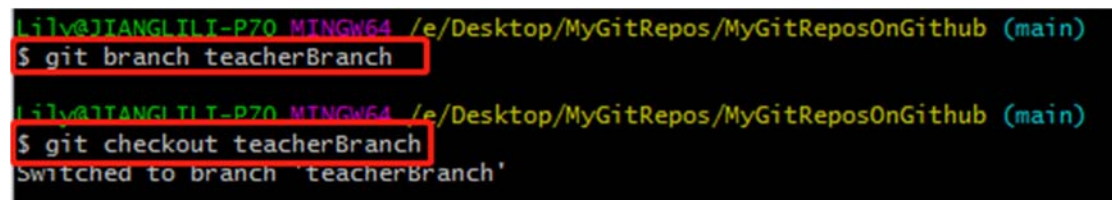
Get the newly added files under the main branch in the GitHub repository



```
$ git branch teacherBranch
```

```
$ git checkout teacherBranch
```

Create a new branch and switch to the new branch



```
$ git branch studentBranch01
```

```
$ git checkout studentBranch01
```

```
$ git push --set-upstream origin studentBranch01
```

Create a new branch and switch to the new branch, then push the current branch and set the remote as upstream

```
Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (teacherBranch)
$ git branch studentBranch01

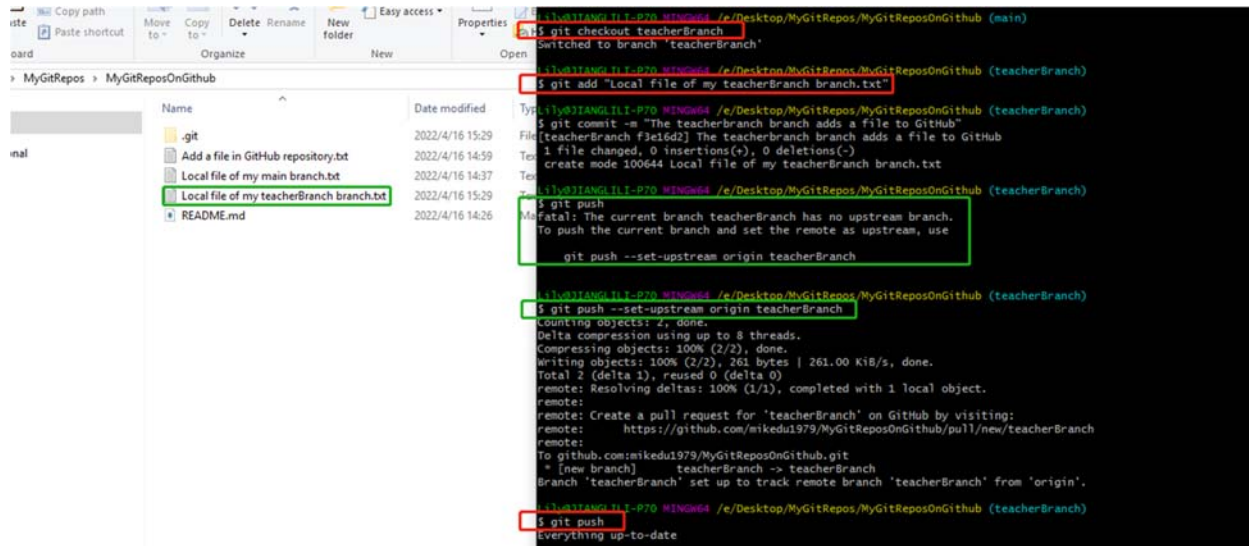
Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (teacherBranch)
$ git checkout studentBranch01
Switched to branch 'studentBranch01'

Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (studentBranch01)
$ git push --set-upstream origin studentBranch01
total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'studentBranch01' on GitHub by visiting:
remote:   https://github.com/mikedu1979/MyGitReposOnGithub/pull/new/studentBranch01
remote:
To github.com:mikedu1979/MyGitReposOnGithub.git
 * [new branch]      studentBranch01 -> studentBranch01
Branch 'studentBranch01' set up to track remote branch 'studentBranch01' from 'origin'.

Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (studentBranch01)
$
```

The screenshot shows the GitHub interface for a repository named 'MyGitReposOnGithub'. At the top, there are navigation links for Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. Below this, there are two yellow notification banners. The first banner indicates that 'teacherBranch' had recent pushes 12 minutes ago, with a 'Compare & pull request' button. The second banner indicates that 'studentBranch01' had recent pushes 3 minutes ago, also with a 'Compare & pull request' button. Below the banners, there is a section for branches and tags. A dropdown menu is open, showing a list of branches: 'main' (checked), 'studentBranch01' (highlighted with a red box), and 'teacherBranch'. To the right of the dropdown, there is a table of recent commits. The table has columns for the commit message, the commit hash, the time ago, and the number of commits. The first commit is 'repository to Add a file in GitHub reposi...' with hash '25b52fe', pushed 37 minutes ago, and it is the 4th commit. Other commits include 'Rename Add a file in GitHub repository to Add a file in GitHub reposi...', 'Add a single file', and 'Initial commit'. At the bottom of the screenshot, the repository name 'MyGitReposOnGithub' is displayed.

Switch to different branches and you can see that the files are different



The screenshot shows a Windows file explorer window on the left and a terminal window on the right. The file explorer shows the contents of the 'MyGitRepos' directory, which includes a '.git' folder and three text files: 'Add a file in GitHub repository.txt', 'Local file of my main branch.txt', and 'Local file of my teacherBranch branch.txt'. The terminal window shows the following commands and output:

```
Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (main)
$ git checkout teacherBranch
Switched to branch 'teacherBranch'

Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (teacherBranch)
$ git add "Local file of my teacherBranch branch.txt"

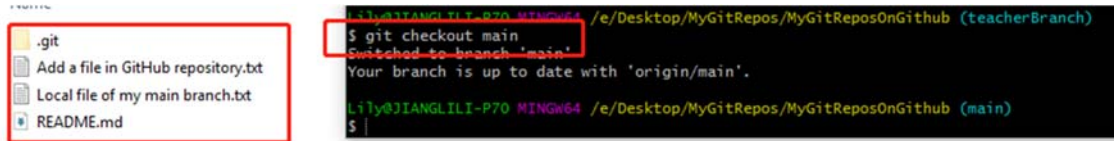
Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (teacherBranch)
$ git commit -m "The teacherBranch branch adds a file to GitHub"
[teacherBranch f3e16d2] The teacherBranch branch adds a file to GitHub
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Local file of my teacherBranch branch.txt

Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (teacherBranch)
$ git push
fatal: The current branch teacherBranch has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin teacherBranch

Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (teacherBranch)
$ git push --set-upstream origin teacherBranch
Counting objects: 2, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 261 bytes | 261.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'teacherBranch' on GitHub by visiting:
remote:   https://github.com/mikedu1979/MyGitReposOnGithub/pull/new/teacherBranch
remote:
To github.com:mikedu1979/MyGitReposOnGithub.git
+ [new branch] teacherBranch -> teacherBranch
Branch 'teacherBranch' set up to track remote branch 'teacherBranch' from 'origin'.

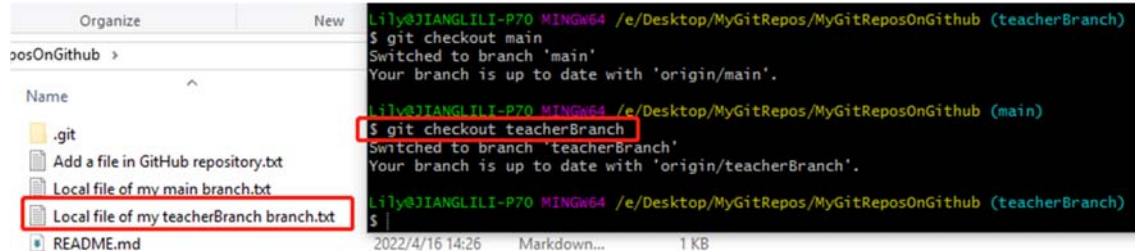
Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (teacherBranch)
$ git push
Everything up-to-date
```



The screenshot shows a Windows file explorer window on the left and a terminal window on the right. The file explorer shows the contents of the 'MyGitRepos' directory, which includes a '.git' folder and three text files: 'Add a file in GitHub repository.txt', 'Local file of my main branch.txt', and 'Local file of my teacherBranch branch.txt'. The terminal window shows the following commands and output:

```
Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (teacherBranch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (main)
$
```

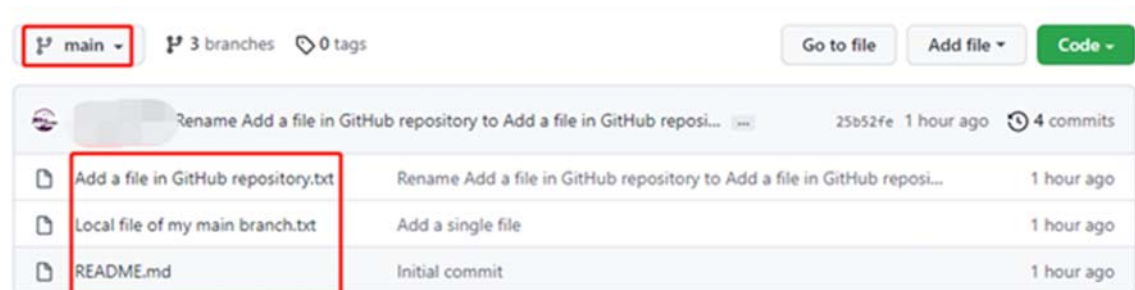


The screenshot shows a Windows file explorer window on the left and a terminal window on the right. The file explorer shows the contents of the 'MyGitRepos' directory, which includes a '.git' folder and three text files: 'Add a file in GitHub repository.txt', 'Local file of my main branch.txt', and 'Local file of my teacherBranch branch.txt'. The terminal window shows the following commands and output:

```
Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (teacherBranch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (main)
$ git checkout teacherBranch
Switched to branch 'teacherBranch'
Your branch is up to date with 'origin/teacherBranch'.

Lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (teacherBranch)
$
```



The screenshot shows a GitHub repository page for 'mikedu1979/MyGitReposOnGithub'. The 'main' branch is selected. The commit history table shows the following commits:

Commit	Message	Author	Date	Commits
25b52fe	Rename Add a file in GitHub repository to Add a file in GitHub repository...	mikedu1979	1 hour ago	4 commits
f3e16d2	Add a file in GitHub repository.txt	mikedu1979	1 hour ago	
...	Local file of my main branch.txt	mikedu1979	1 hour ago	
...	Initial commit	mikedu1979	1 hour ago	

teacherBranch 3 branches 0 tags Go to file Add file Code

This branch is 1 commit ahead of main. Contribute

The teacherbranch branch adds a file to GitHub f3e16d2 29 minutes ago 5 commits

Add a file in GitHub repository.txt	Rename Add a file in GitHub repository to Add a file in GitHub reposi...	1 hour ago
Local file of my main branch.txt	Add a single file	1 hour ago
Local file of my teacherBranch branch...	The teacherbranch branch adds a file to GitHub	29 minutes ago
README.md	Initial commit	1 hour ago

```
$ git checkout main
```

```
$ git merge teacherBranch
```

Merge the "local file of my teacherbranch branch.txt" file under the "teacherbranch" branch into the "main" branch

MyGitReposOnGithub

Name	Date modified
.git	2022/4/16 15:49
Add a file in GitHub repository.txt	2022/4/16 14:59
Local file of my main branch.txt	2022/4/16 14:37
Local file of my teacherBranch branch.txt	2022/4/16 15:49
README.md	2022/4/16 14:26

```
MINGW64/e/Desktop/MyGitRepos/MyGitReposOnGithub (teacherBranch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
MINGW64/e/Desktop/MyGitRepos/MyGitReposOnGithub (main)
$ git merge teacherBranch
Updating 25b2fe..f3e16d2
Fast-forward
 Local file of my teacherBranch branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Local file of my teacherBranch branch.txt
lily@JIANGLILI-P70 MINGW64 /e/Desktop/MyGitRepos/MyGitReposOnGithub (main)
$
```

2. Git common commands (reference material)

(1) Working with local repositories

git init

This command turns a directory into an empty Git repository. This is the first step in creating a repository. After running git init, adding and committing files/directories is possible.

Usage:

```
# change directory to codebase
$ cd /file/path/to/code

# make directory a git repository
$ git init
```

In Practice:

```
# change directory to codebase
$ cd /Users/computer-name/Documents/website

# make directory a git repository
$ git init
Initialized empty Git repository in /Users/computer-name/Documents/website/.git/
```

git add

Adds files in the to the staging area for Git. Before a file is available to commit to a repository, the file needs to be added to the Git index (staging area). There are a few different ways to use git add, by adding entire directories, specific files, or all unstaged files.

Usage:

```
$ git add <file or directory name>
```

In Practice:

```
# To add all files not staged:
$ git add .

# To stage a specific file:
$ git add index.html

# To stage an entire directory:
$ git add css
```

git commit

Record the changes made to the files to a local repository. For easy reference, each commit has a unique ID.

It's best practice to include a message with each commit explaining the changes made in a commit. Adding a commit message helps to find a particular change or understanding the changes.

Usage:

```
# Adding a commit with message
$ git commit -m "Commit message in quotes"
```

In Practice:

```
$ git commit -m "My first commit message"
[SecretTesting 0254c3d] My first commit message
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 homepage/index.html
```

git status

This command returns the current state of the repository.

git status will return the current working branch. If a file is in the staging area, but not committed, it shows with *git status*. Or, if there are no changes it'll return *nothing to commit, working directory clean*.

Usage:

```
$ git status
```

In Practice:

```
# Message when files have not been staged (git add)
$ git status
On branch SecretTesting
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    homepage/index.html

# Message when files have been not been committed (git commit)
$ git status
On branch SecretTesting
Your branch is up-to-date with 'origin/SecretTesting'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
```



```
new file: homepage/index.html

# Message when all files have been staged and committed
$ git status
On branch SecretTesting
nothing to commit, working directory clean
```

git config

With Git, there are many configurations and settings possible. *git config* is how to assign these settings. Two important settings are user.name and user.email. These values set what email address and name commits will be from on a local computer. With *git config*, a *--global* flag is used to write the settings to all repositories on a computer. Without a *--global* flag settings will only apply to the current repository that you are currently in.

There are many other variables available to edit in *git config*. From editing color outputs to changing the behavior of *git status*. Learn about *git config* settings in the official [Git documentation](#).

Usage:

```
$ git config <setting> <command>
```

In Practice:

```
# Running git config globally
$ git config --global user.email "my@emailaddress.com"
$ git config --global user.name "Brian Kerr"

# Running git config on the current repository settings
$ git config user.email "my@emailaddress.com"
$ git config user.name "Brian Kerr"
```

git branch

To determine what branch the local repository is on, add a new branch, or delete a branch.

Usage:

```
# Create a new branch
$ git branch <branch_name>

# List all remote or local branches
$ git branch -a

# Delete a branch
$ git branch -d <branch_name>
```

In Practice:

```
# Create a new branch
```

```
$ git branch new_feature

# List branches
$ git branch -a
* SecretTesting
  new_feature
  remotes/origin/stable
  remotes/origin/staging
  remotes/origin/master -> origin/SecretTesting

# Delete a branch
$ git branch -d new_feature
Deleted branch new_feature (was 0254c3d).
```

git checkout

To start working in a different branch, use *git checkout* to switch branches.

Usage:

```
# Checkout an existing branch
$ git checkout <branch_name>

# Checkout and create a new branch with that name
$ git checkout -b <new_branch>
```

In Practice:

```
# Switching to branch 'new_feature'
$ git checkout new_feature
Switched to branch 'new_feature'

# Creating and switching to branch 'staging'
$ git checkout -b staging
Switched to a new branch 'staging'
```

git merge

Integrate branches together. *git merge* combines the changes from one branch to another branch. For example, merge the changes made in a staging branch into the stable branch.

Usage:

```
# Merge changes into current branch
$ git merge <branch_name>
```

In Practice:

```
# Merge changes into current branch
$ git merge new_feature
```

```
Updating 0254c3d..4c0f37c
Fast-forward
 homepage/index.html | 297 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 297 insertions(+)
 create mode 100644 homepage/index.html
```

(2) Working with remote repositories

git remote

To connect a local repository with a remote repository. A remote repository can have a name set to avoid having to remember the URL of the repository.

Usage:

```
# Add remote repository
$ git remote <command> <remote_name> <remote_URL>

# List named remote repositories
$ git remote -v
```

In Practice:

```
# Adding a remote repository with the name of beanstalk
$ git remote add origin
git@account_name.git:beanstalkapp.com:/account_name/repository_name.git

# List named remote repositories
$ git remote -v
origin git@account_name.git:beanstalkapp.com:/account_name/repository_name.git
(fetch)
origin git@account_name.git:beanstalkapp.com:/account_name/repository_name.git
(push)
```

Note: A remote repository can have any name. It's common practice to name the remote repository 'origin'.

git clone

To create a local working copy of an existing remote repository, use *git clone* to copy and download the repository to a computer. Cloning is the equivalent of *git init* when working with a remote repository. Git will create a directory locally with all files and repository history.

Usage:

```
$ git clone <remote_URL>
```

In Practice:

```
$ git clone
git@account_name.git.beanstalkapp.com:/account_name/repository_name.git
Cloning into 'repository_name'...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (5/5), 3.08 KiB | 0 bytes/s, done.
Checking connectivity... done.
```

git pull

To get the latest version of a repository run *git pull*. This pulls the changes from the remote repository to the local computer.

Usage:

```
$ git pull <branch_name> <remote_URL/remote_name>
```

In Practice:

```
# Pull from named remote
$ git pull origin staging
From account_name.git.beanstalkapp.com:/account_name/repository_name
* branch          staging    -> FETCH_HEAD
* [new branch]     staging    -> origin/staging
Already up-to-date.

# Pull from URL (not frequently used)
$ git pull git@account_name.git.beanstalkapp.com:/account_name/repository_name.git staging
From account_name.git.beanstalkapp.com:/account_name/repository_name
* branch          staging    -> FETCH_HEAD
* [new branch]     staging    -> origin/staging
Already up-to-date.
```

git push

Sends local commits to the remote repository. *git push* requires two parameters: the remote repository and the branch that the push is for.

Usage:

```
$ git push <remote_URL/remote_name> <branch>
```

```
# Push all local branches to remote repository
$ git push -all
```

In Practice:

```
# Push a specific branch to a remote with named remote
```

```

$ git push origin staging
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 734 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
To git@account_name.git.beanstalkapp.com:/account_name/repository_name.git
    ad189cb..0254c3d SecretTesting -> SecretTesting

# Push all local branches to remote repository
$ git push --all
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 373 bytes | 0 bytes/s, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To git@account_name.git.beanstalkapp.com:/account_name/repository_name.git
    0d56917..948ac97 master -> master
    ad189cb..0254c3d SecretTesting -> SecretTesting

```

(3) Advanced Git Commands

git stash

To save changes made when they're not in a state to commit them to a repository. This will store the work and give a clean working directory. For instance, when working on a new feature that's not complete, but an urgent bug needs attention.

Usage:

```

# Store current work with untracked files
$ git stash -u

# Bring stashed work back to the working directory
$ git stash pop

```

In Practice:

```

# Store current work
$ git stash -u
Saved working directory and index state WIP on SecretTesting: 4c0f37c Adding new file to branch
HEAD is now at 4c0f37c Adding new file to branch

# Bring stashed work back to the working directory

```

```
$ git stash pop
```

On branch SecretTesting

Your branch and 'origin/SecretTesting' have diverged,
and have 1 and 1 different commit each, respectively.

(use "git pull" to merge the remote branch into yours)

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

```
    modified:   index.html
```

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

Dropped refs/stash@{0} (3561897724c1f448ae001edf3ef57415778755ec)

git log

To show the chronological commit history for a repository. This helps give context and history for a repository. *git log* is available immediately on a recently cloned repository to see history.

Usage:

```
# Show entire git log
```

```
$ git log
```

```
# Show git log with date parameters
```

```
$ git log --<after/before/since/until>=<date>
```

```
# Show git log based on commit author
```

```
$ git log --<author>="Author Name"
```

In Practice:

```
# Show entire git log
```

```
$ git log
```

```
commit 4c0f37c711623d20fc60b9cbcf393d515945952f
```

```
Author: Brian Kerr <my@emailaddress.com>
```

```
Date:   Tue Oct 25 17:46:11 2016 -0500
```

Updating the wording of the homepage footer

```
commit 0254c3da3add4ebe9d7e1f2e76f015a209e1ef67
```

```
Author: Ashley Harpp <my@emailaddress.com>
```

```
Date:   Wed Oct 19 16:27:27 2016 -0500
```

My first commit message

```
# Show git log with date parameters
```



```
$ git log --before="Oct 20"
commit 0254c3da3add4ebe9d7e1f2e76f015a209e1ef67
Author: Ashley Harpp <my@emailaddress.com>
Date:   Wed Oct 19 16:27:27 2016 -0500
```

My first commit message

```
# Show git log based on commit author
$ git log --author="Brian Kerr"
commit 4c0f37c711623d20fc60b9cbcf393d515945952f
Author: Brian Kerr <my@emailaddress.com>
Date:   Tue Oct 25 17:46:11 2016 -0500
```

Updating the wording of the homepage footer

git rm

Remove files or directories from the working index (staging area). With *git rm*, there are two options to keep in mind: force and cached. Running the command with force deletes the file. The cached command removes the file from the working index. When removing an entire directory, a recursive command is necessary.

Usage:

```
# To remove a file from the working index (cached):
$ git rm --cached <file name>

# To delete a file (force):
$ git rm -f <file name>

# To remove an entire directory from the working index (cached):
$ git rm -r --cached <directory name>

# To delete an entire directory (force):
$ git rm -r -f <file name>
```

In Practice:

```
# To remove a file from the working index:
$ git rm --cached css/style.css
rm 'css/style.css'

# To delete a file (force):
$ git rm -f css/style.css
rm 'css/style.css'

# To remove an entire directory from the working index (cached):
```

```
$ git rm -r --cached css/
rm 'css/style.css'
rm 'css/style.min.css'

# To delete an entire directory (force):
$ git rm -r -f css/
rm 'css/style.css'
rm 'css/style.min.css'
```

More reference: <https://git-scm.com/book/en/v2>