Using Git For Collaboration
Remotes and Branches
Using the Git Command-line to work with branches on your local repository

Using The Github UI
Working with collaborators using the Github UI

CI/CD
Understanding Continuous Integration and Continuous Deployment/Delivery
REMOTE
Repositories live on a GitHub server.

LOCAL
Your computer talks to the GitHub server with terminal.

Browser lets you access repository and send changes back to the server.

LOCAL
Someone else’s computer talks to the GitHub server.
Git Local vs Remote

Local

- working directory
- staging area
- local repo

Local processes:
- `git add`
- `git commit`
- `git checkout`
- `git merge`

Remote

- remote repo

Remote processes:
- `git push`
- `git fetch`
Some Basic Terminology

Repository (Repo)
A folder for your project that contains all of your project's files along with their revision history.

Local repository (Workspace)
Your local copy of a repository on your computer.

origin
The remote, original copy of a repository.
**git status**

Gives you a current overview of your repository. It telling you which files have or haven’t been saved and what changes are in staging.

**git push**

Pushes changes in your locally committed repository to the remote repository.

**git pull**

Downloads or updates the contents of a remote repository to your local device.
SSH Keys

- ssh keys are special, unique files that allow the user to access secure data (in this case our code)

- Git implements a similar level of security and requires user to ssh with ssh keys.

- If a user does not have a key, they will not gain access to anything

---

**Generate your SSH keys**

Enter the following command with the email associated with your GitHub account:

```
ssh-keygen
```

Generating public/private rsa key pair.
Enter file in which to save the key (/Users/sanjay/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/sanjay/.ssh/id_rsa.
Your public key has been saved in /Users/sanjay/.ssh/id_rsa.pub.

Accept the default save location by pressing enter and continue by answering the prompts and run the following command:

```
pbcopy < ~/.ssh/id_rsa.pub
```

The above command copies the ssh key to your clipboard
SSH keys continued . . .

1. Navigate and sign into GitHub in your browser

1. In the top right corner, press on your icon and then in the menu bar, select Settings
SSH keys continued...

3. Once in settings, navigate to SSH and GPG keys.

4. Navigate to the button that is labeled “new SSH key” and then paste what is in your clipboard into “Key”

5. Press “Add SSH key” when finished.
Making our repository

Use the “git init” command to make a new repository called Test_Repo

$ git init Test_Repo
Initialized empty Git repository in /Users/sanjay/Desktop/classes/git_examples/Test_Repo/.git/

Now simply add any files you want to the directory

$ Test_Repo git:(master) touch test.txt
$ Test_Repo git:(master) ls
  test.txt
Moving test.txt to the Staging Area

run “git add test.txt” to move the file to the staging area

[→ Test_Repo git:(master) × git add test.txt]
Committing Our Changes

run "git commit" to move the file from the staging area. We add the "-m" option to include a message

```
[→ Test_Repo git:(master) × ]$ git commit -m "V1"
[master (root-commit) 1a7c555] V1
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test.txt
```
Checking on Our Progress

run “git status” to check what is in our staging area and in our master branch.

$ Test_Repo git:(master) git status
On branch master
nothing to commit, working tree clean

You can also run “git log” to see the all your commit history

commit 1a7c555f1a720a00d80778861d464624dc43b315 (HEAD -> master)
Author: Sanjay S <srikumar.sanjay@gmail.com>
Date: Mon Nov 16 23:17:50 2020 -0500
Making our remote repository

1. Navigate and sign into GitHub in your browser

1. In the top right corner, press on your icon and then in the menu bar, select Your repositories
Remote Repository continued . . .

1. Find the above search bar, and look for the button labeled new

1. In the top right corner, press on your icon and then in the menu bar, select Your repositories
Give your repository a name and then select Create Repository.
GitHub will then give you a SSH URL to use, along with some commands we’ve gone over to get you started:

### Quick setup — if you’ve done this kind of thing before

![GitHub URL](https://avatars.githubusercontent.com/u/14150526?v=4&size=40)

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

#### ...or create a new repository on the command line

```bash
echo "# hi" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:Sanjays04/hi.git
git push -u origin main
```

#### ...or push an existing repository from the command line

```bash
git remote add origin git@github.com:Sanjays04/hi.git
git branch -M main
git push -u origin main
```

#### ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code]
"git remote add (origin remote repository URL) creates a remote connection called origin from your local repository.

"git push origin master" simply pushes any changes from your local master branch to the remote master branch. (don't worry about branching just yet)

```bash
$ Test_Repo git:(main) git remote add origin git@github.com:Sanjay404/Test_Repo.git
$ Test_Repo git:(main) git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 209 bytes | 209.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:Sanjay404/Test_Repo.git
 * [new branch] main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```
Viewing Repository on Github
2. Working With Branches

Using the Git CLI to work with branches on your local repository
What are Branches?

- A version of the repository that you can edit that has been separated from the master branch
- Allows you to create edits on a new feature you are developing without changing the main code
- These branches can be merged later with the master branch
Checking Your Current Branch

To check the branch you are currently on, you can run the `git status` command within your repository:

```
[TestRepo] $ git status
On branch master
Your branch is up to date with 'origin/master'.
nothing to commit, working tree clean
TestRepo $
```
Getting a List of Branches

To get a list of branches that currently exist, you can use the `git branch` command within your repository:
Creating a New Branch

To create a new branch you can use the `git branch branch_name` command within your repository:
Switching Branches

To create a new branch you can use the `git checkout branch_name` command within your repository:
To merge a branch to an existing branch, you need to first checkout the branch you want to merge to with `git checkout dest_branch` and merge with `git merge --no-ff src_branch` command within your repository:

```
[TestRepo $]
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
[TestRepo $]
$ git merge --no-ff test_branch
Merge made by the 'recursive' strategy.
  README.md | 2 ++
  1 file changed, 2 insertions(+)
TestRepo $]
```
To push a branch to a remote repository, you can push your new branch with `git push -u origin your_branch` command within your repository. **Caution:** you will need to pull changes before pushing new changes.
To pull changes from a remote repository, you can use the `git pull` command within your repository. **Caution:** The `git status` command will not show you if the remote branch has new commits.

```bash
TestRepo $ git status
On branch test_branch
Your branch is up to date with 'origin/test_branch'.

nothing to commit, working tree clean
TestRepo $ git pull
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 11 (delta 2), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (11/11), done.
From github.com:sagars729/TestRepo
  22f6593..b560609  test_branch -> origin/test_branch
  4bfc5eb..b1544f5  master    -> origin/master
Updating 22f6593..b560609
Fast-forward
  README.md   | 2 ++
1 file changed, 2 insertions(+)
TestRepo $
```
3. Using The Github UI

Working with collaborators using the Github UI
Viewing The Repository

- Repository Owner and Name
- Settings
- Current Branch
- Files
- Project README
- Commits
Adding Collaborators

Manage Access Tab

Invite Collaborator
Adding Collaborators

Add A Collaborator
Viewing Commits and Older Versions

Current Branch
Commit Hash
Commit Message
Old Versions
Viewing Changes

Commit Message
Commit Hash
Changed Lines
Switch Branch

Create A Pull Request
Switch Branch
Choose Branch
Creating A Pull Request

1. Select Branch To Merge To
2. Commit Message For Merge
3. Longer Merge Description
4. Create Pull Request
Adding Assignee and Reviewer

Click To Choose Reviewer
Click Here To Choose Assignee
Select a Team Member
Adding Reviewer

Click To Choose Reviewer
Select a Team Member

Merge is Blocked Until Review
Adding Review

Click To Add Your Review
Reviewing Changes

Click To Add Your Review

View Changes In Pull Request
Approving Changes

Choose To Comment, Approve, or Request Changes

Add Your Review

Submit Review
Merge Your Pull Request

See The Review

Merge Pull Request
Understanding Continuous Integration and Continuous Deployment/Delivery
What is Continuous Integration?

- The coding practice of implementing small changes and frequently committing those changes
- Allows for big projects to be divided into smaller more manageable sub-tasks
- Enables large teams to work collaboratively on a single project by having each team member pick up a sub-task
- Frequently committing allows for each team member’s contributions to be used by the rest of the team without much delay
What is Continuous Delivery?

- The coding practice of automating the code release process
- As changes are continuously integrated into a system by developers, they are also continuously delivered to clients
- Requires at least one manual step of approving the deployment of code once code is ready to be delivered
Continuous Delivery vs Continuous Deployment

- Continuous Delivery requires at least one step of manually approving the deployment of code once it is ready
- Continuous Deployment automates the step of approving the deployment of code
- Requires rigorous testing, monitoring, and ability to scale back
A Quick Scenario

- git push
- git pull

latest code

packaged code

Give Feedback, Report Errors

Give Feedback, Report Errors
CI/CD

Image Source: saviantconsulting.com