GIT

Class 2
New
Version

Agenda

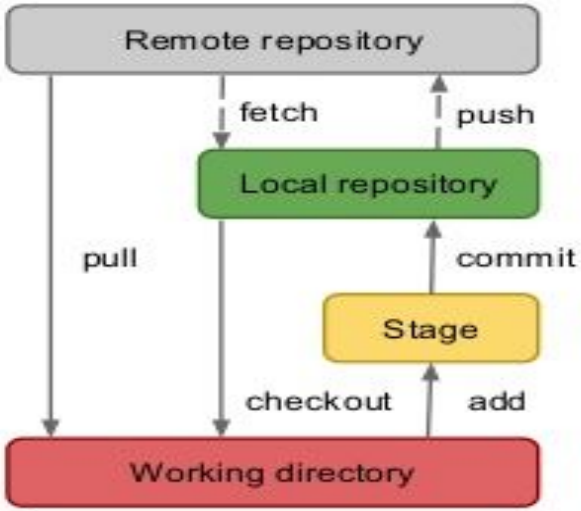
Git Workflow

Git Commands

Git Workflow

Let’s understand the complete flow:

Understanding of workflow



- Obtain a repository
  - *git init* or *git clone*
- Make some changes
- Stage your changes
  - *git add*
- Commit changes to the local repository
  - *git commit -m “My message”*
- Push changes to remote
  - *git push remotename remotebranch*

Git Workflow

```
Asele4ka:GitTest assele4ka$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .classpath
        new file:   .gitignore
        new file:   .project
        new file:   .settings/org.eclipse.jdt.core.prefs
        new file:   .settings/org.eclipse.m2e.core.prefs
        new file:   pom.xml
        new file:   src/test/java/TestClass.java
```

Working directory is simply the folder that contains the .git folder. It is the directory within which you have checked out a branch of your project.

Whenever we modify any files in working directory they do not show up in our local repository unless we commit those changes.

Git Workflow

Before commit we use “git add ” command - that is when we are adding files to the **stage** (staging environment or index)

Stage is nothing but the **pre-step** for committing the files. So the first step before commit we are adding files into the **stage**.

Once we added files to the stage environment and now if we run “git status” we will see notice “Changes to be committed”.

Git Workflow

```
Asele4ka:GitTest assele4ka$ git add .
Asele4ka:GitTest assele4ka$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .classpath
        new file:   .gitignore
        new file:   .project
        new file:   .settings/org.eclipse.jdt.core.prefs
        new file:   .settings/org.eclipse.m2e.core.prefs
        new file:   pom.xml
        new file:   src/test/java/TestClass.java
        new file:   src/test/java/TestClass2.java
```

Once we added files to the stage environment and now if we run “git status” we will see notice “Changes to be committed”.

Git Workflow

```
8 files changed, 69 insertions(+)
create mode 100644 .classpath
create mode 100644 .gitignore
create mode 100644 .project
create mode 100644 .settings/org.eclipse.jdt.core.prefs
create mode 100644 .settings/org.eclipse.m2e.core.prefs
create mode 100644 pom.xml
create mode 100644 src/test/java/TestClass.java
create mode 100644 src/test/java/TestClass2.java
```

After files are staged now we can commit them to the local repo.
Committing is nothing but adding our staged files into local repository
Commit command is used to commit the staged files, we can also set the message for the commit, messages could be your simple description about the changes.

Git Workflow

```
Asele4ka:GitTest assele4ka$ git log
commit fea5880c24f0dc3dc851f21679d786342ad7ad9b (HEAD -> master)
Author: Asel <assele4ka@Asele4ka.fios-router.home>
Date:   Thu Dec 6 14:14:41 2018 -0500

    Adding 3 Test Case

commit f7f4bd8cd67eb7522ee956f9b54799dd888cb86b
Author: Asel <assele4ka@Asele4ka.fios-router.home>
Date:   Thu Dec 6 14:12:54 2018 -0500

    Adding 2 Test Case
```

We can see the commit history by using log command, this will tell us what branch we are in and he author, date, commit message.

Git Workflow

```
Asele4ka:GitTest assele4ka$ git push origin master
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 8 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (20/20), 1.94 KiB | 992.00 KiB/s, done.
Total 20 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/SyntaxTechnologies/GitTest/pull/new/master
remote:
To https://github.com/SyntaxTechnologies/GitTest.git
 * [new branch]      master -> master
```

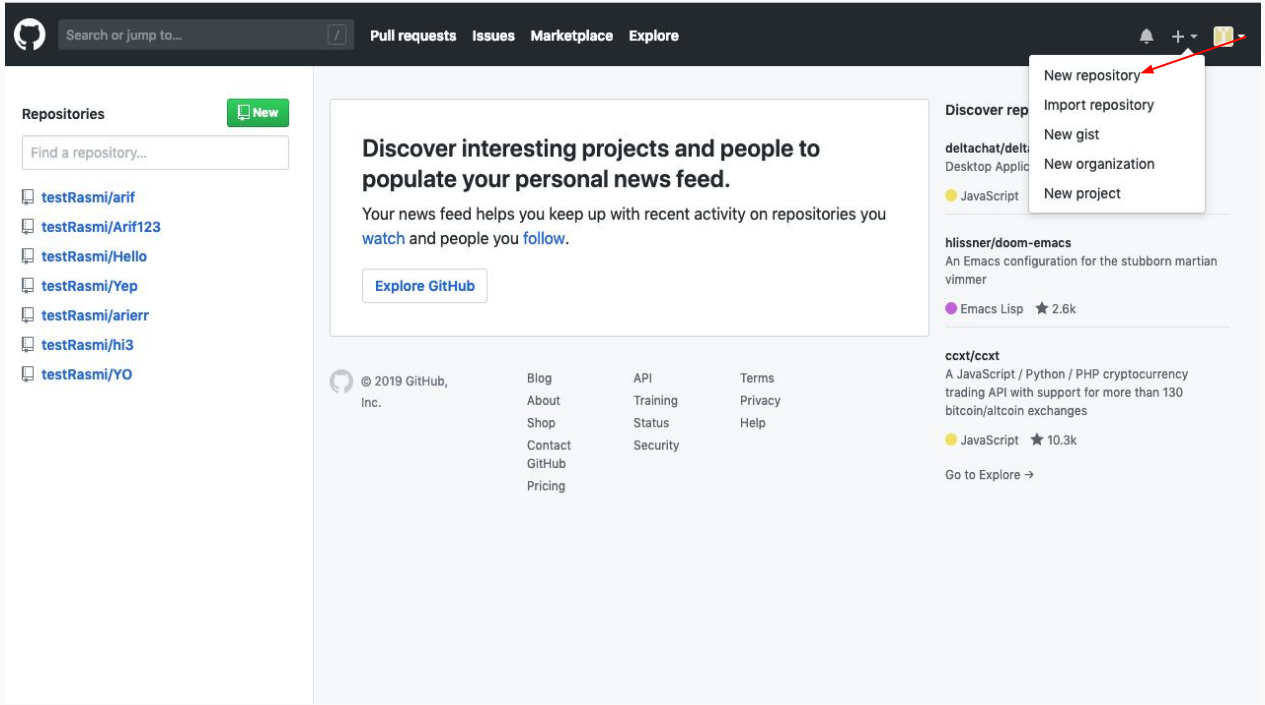
If we only want to keep track of our code locally, we don't need to use GitHub. But if we want to work with a team, we can use GitHub to collaboratively modify the project's code.

To get code from local repo to the GitHub(Remote repo) we need to push commit.

Create a new repository

Click on the plus button.

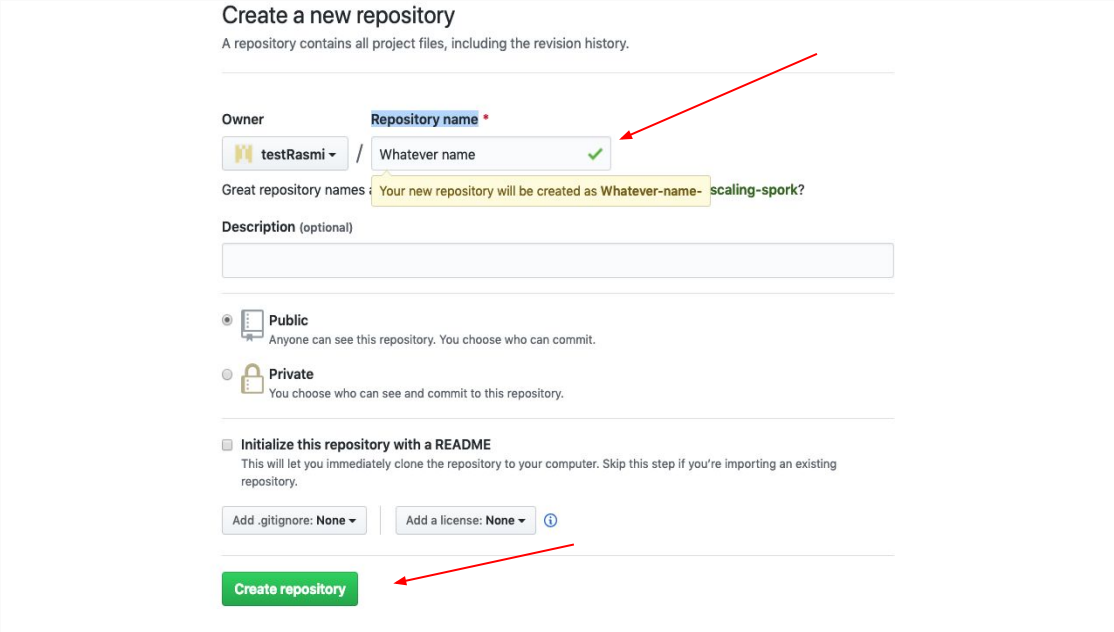
Then click on new repository



Create a new repository

Enter whatever name into the Repository name

Then click on create repository button



testRasmi / Whatever-name-

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

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

Set up in Desktop or HTTPS SSH https://github.com/testRasmi/Whatever-name-.git

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

```
echo "# Whatever-name-" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/testRasmi/Whatever-name-.git
git push -u origin master
```

Git Add

Type this command into the terminal:

git add HelloWorld.txt

Then type this command into the terminal:

git status

```
Arifs-MacBook-Pro:gitTest arif$ git add HelloWorld.txt
Arifs-MacBook-Pro:gitTest arif$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   HelloWorld.txt

Arifs-MacBook-Pro:gitTest arif$
```

Git Commit

Type this command into the terminal:

git commit -m “My first commit”

```
Arifs-MacBook-Pro:gitTest arif$ git commit -m"My first commit"
[master 7104c35] My first commit
 1 file changed, 1 insertion(+)
 create mode 100644 HelloWorld.txt
Arifs-MacBook-Pro:gitTest arif$
```

Git remote

Type this command into the terminal:

git remote add origin https://github.com/YOUR U NAME/REPOSITORY NAME-.git

```
Arifs-MacBook-Pro:gitTest arif$ git remote add origin https://github.com/testRasmi/Whatever-name-.git
Arifs-MacBook-Pro:gitTest arif$
```

Git Push

Push
A push is how we get our local files to a remote repo so team will have the option to pull our latest modifications.

Git Push

Type this command into the terminal:

git push -u origin master

```
Arifs-MacBook-Pro:gitTest arif$ git push -u origin master
Password for 'https://testRasmi@github.com':
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 478 bytes | 478.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/testRasmi/Whatever-name-.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
Arifs-MacBook-Pro:gitTest arif$
```

testRasmi / Whatever-name-

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

No description, website, or topics provided.

Edit

Manage topics

2 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

testRasmi My first commit — Latest commit 7104c35 27 minutes ago

Hello.txt	My first commit	an hour ago
HelloWorld.txt	My first commit	27 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README

Good practices for pushing to the remote repository

- Make sure to communicate what you are working on with teammates before you start any change or feature to avoid duplication of efforts.
- Before pushing, make sure you have pulled all current files or changes from the master/remote.
- Never commit anything that is not complete.
- Commit often.
- Use descriptive commit messages