GIT

Class 1

Agenda

What is version control system? Why do we need version control tools?

What is GIT? GitHub?

GIT Workflow

Adding user details in GIT

Create a Git Repository

Git Commands

What is version control system?

Let’s think of the scenario:

1. You make a number of improvements to a class.
   Your co-worker makes a number of different improvements to the same class
   How can you merge these changes?

2. You change one part of a program--it works
   Your co-worker changes another part--it works
   You put them together--it doesn’t work
   Some change in one part must have broken something in the other part. What were all the changes?

What is version control system?

- Version Control System (VCS) is a software that helps software developers to work together and maintain a complete history of their work.
- Version Control System is the management of changes to documents, computer programs, large websites and other collection of information.
- We can think of a Version Control System (aka: VCS) as a kind of database.
- Version Controls keeps track of all the version of our project, every small change create a new version of the project.

What is version control system?

- VCS Keeps multiple (older and newer) versions of everything (not just source code).
- VCS requests comments regarding every change.
- VCS allows “check in” and “check out” of files so you we which files someone else is working on.
- VCS displays differences between versions

Benefits of version control

- Gives us a “time machine” for going back to earlier versions.
- Gives us great support for different versions (standalone, web app, etc.) of the same basic project.
- Greatly simplifies concurrent work, merging changes.
- Allows developers/ automation engineers to work simultaneously.
- Does not allow overwriting each other’s changes.

Version control tools

The tool which handles all the backups are called as Version control tools. Most popular vc tools:

- GIT
- SVN
- CVS
- Bazaar
- Assembla
- Mercurial

What is Git?

Git is an open source version control system tool for tracking changes in computer files and coordinating work on those files among multiple people.

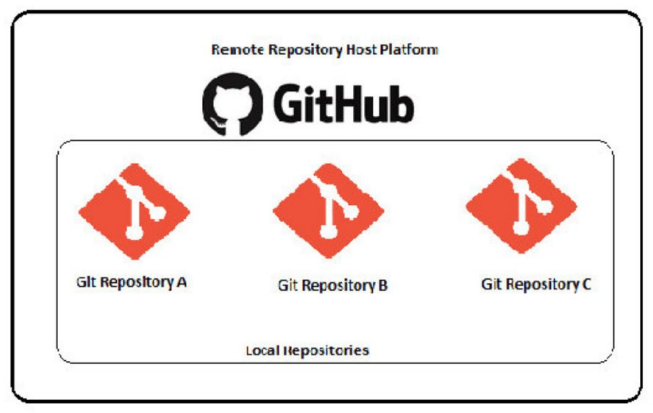
Git is distributed control system. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.

Git is an open source program to keep track of changes in our files. It just takes screenshots of your files when you save them, and then it compares the files for differences.

Git vs GitHub

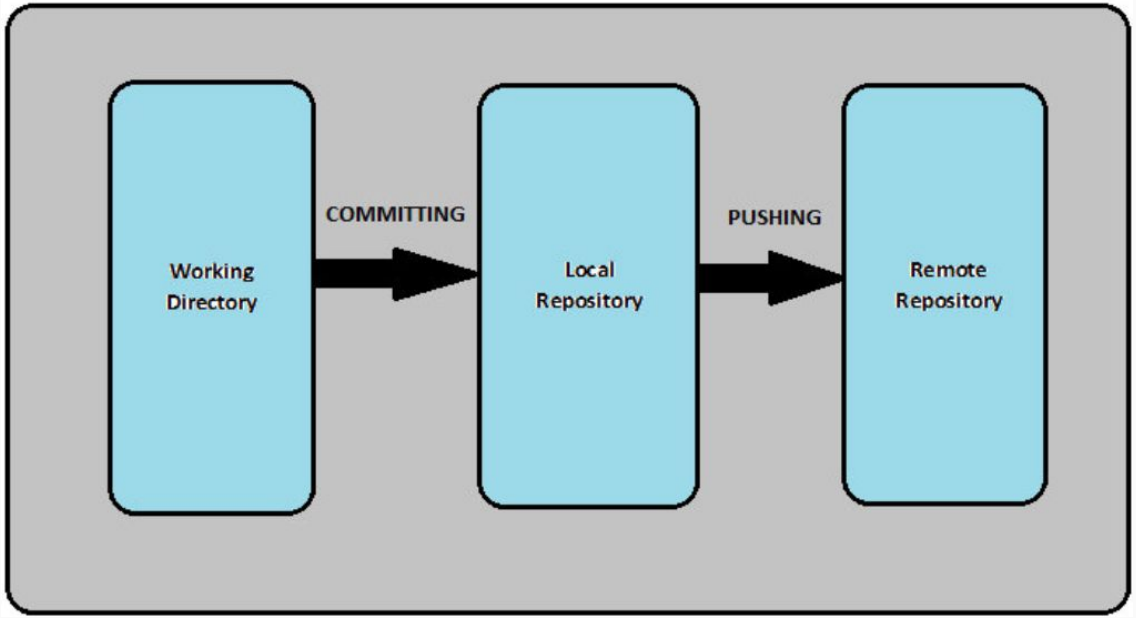
Git: It is a Distributed Version Control System for tracking versions of files.

Github: It is web portal and cloud hosting service for your Git repositories.



We use commands of Git to track versions of your files. And Github is just a remote platform where these files are hosted.

Git Workflow



Step 1: We modify a file from the working directory.
Step 2: You commit these files to the local repository.
Step 3: We push code from local to remote repository, it stores the changes permanently to the Git repository.

Basics of Git CMD

Command line/CMD commands in git:

pwd
Get the current folder details

mkdir folderToCreate
Create a new folder

cd folderName
Move into a sub-folder

ls
List the files in the current folder

cd ..
Step out of current directory

Adding user details in Git

Setting user details like user name and email is important for tracking purpose.

Before setting any values in git we need to open git bash terminal.
--global is nothing but the scope of the user details

$ git config --global user.name "[name]"
Sets the name you want a ached to your commit transactions

$ git config --global user.email "[email address]"
Sets the email you want a ached to your commit transactions

Git Repository

Our top-level working directory contains everything about our project: source code, binaries, documentation, data files, etc.

At any time, we can take a “snapshot” of everything (or selected things) in our project directory, and put it in our repository. This “snapshot” is called a commit object.

Commit objects do not require huge amounts of memory. We can work as much as we like in our working directory, but the repository isn’t updated until we commit something.

Create a Git Repository

Start a new repository or obtain one from an existing URL

$ git init [project-name]
Creates a new local repository with the specified name

$ git clone [url]
Downloads a project and its entire version history

Git Status

git status command shows the difference between the working folder and the local repository. When we add files in the current directory it does not mean they are present in the local repository. Those two files are just present in the working directory, even though the working directory and local repository are same.

Files will be moved to local repository only when we commit them into local repository

The untracked files are nothing but the files that you have created but not committed.

Make changes

A **git commit** is when you tell git that a change (or addition) you have made is
ready to be included in the project
When you commit your change to git, it creates a commit object

When you commit, you must provide a one-line message stating what you have done. Commit messages can be very helpful, to yourself as well as to your team members

Make changes

If you create new files and/or folders, they are not tracked by Git unless you ask it to do so:
git add newFile1

Committing makes a “snapshot” of everything being tracked into your repository. A message telling what you have done is required:
git commit –m “My first commit”

Make changes

If you create new files and/or folders, they are not tracked by Git unless you ask it to do so:
git add newFile1 newFolder1
or
git add .

Committing makes a “snapshot” of everything being tracked into your repository. A message telling what you have done is required:
git commit –m “My first commit”
git commit

Format of the commit message:
One line containing the complete summary
If more than one line, the second line must be blank

Git Push

Push operation copies changes from a local repository instance to a remote one.
This is used to store the changes permanently into the Git repository and allows other people to see the changes you've made.

git push origin yourbranchname

GIT

Class 2

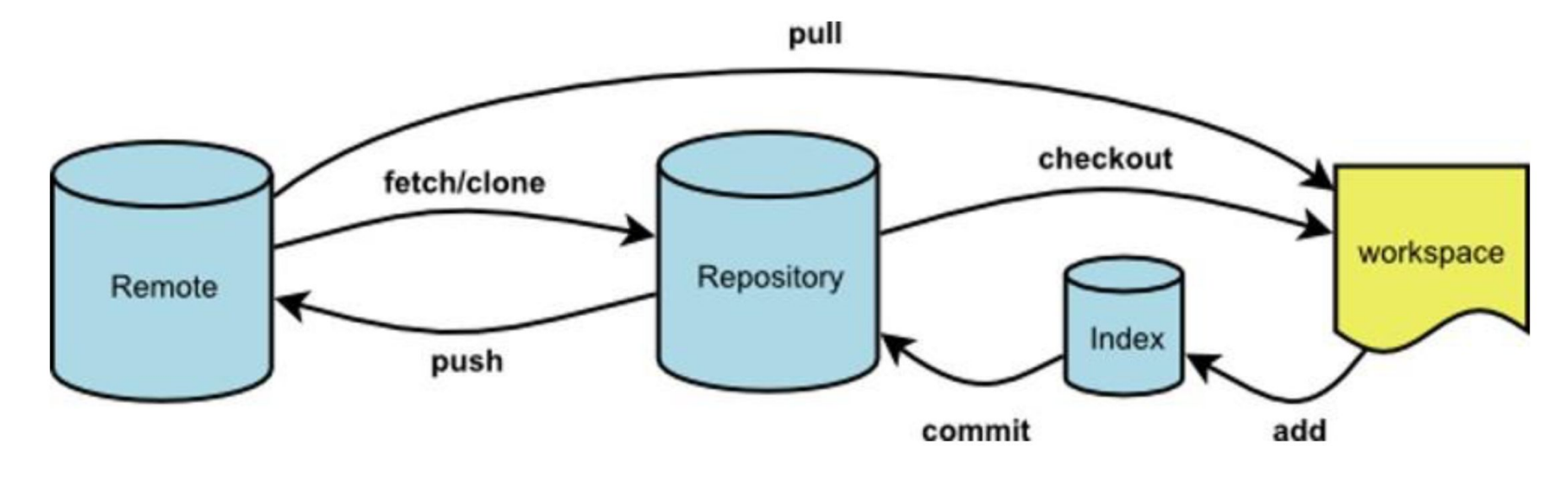
Agenda

Git Workflow

Git Commands

Git Workflow

Let’s understand the complete flow:



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.