

Git:

git is used to store a code in a single place called repository and it can access to all the users(developers) at a time.

Advantage of Git:

Code Security

Easy Accessible

Maintain version

Track the code

Basically in git we will have two repository, Even in real projects are also we are going to have two repositories

1. Main Branch

2. Dummy Branch

Every time we will push the code to dummy branch only, in some companies they will directly push the code to master branch only.

1. Why we need to push our code to dummy branch?

```
git branch new_branch_name
```

```
git branch -d branch_name (d for delete)
```

Git Repository Setup:

First we need to create a git account and create a repository

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner

Repository name \*



RajeshJakkula

/

Scala\_Coding



Great repository names are short and memorable. Need inspiration? How about **fuzzy-engine**?

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.



**Initialize this repository with a README**

This will let you immediately clone the repository to your computer.

Add .gitignore: **None**

Add a license: **None**



Create repository

Once repository is created, below screen shot we can see

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

or

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

---

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

```

git remote add origin https://github.com/RajeshJakkula/Scala_Coding.git
git push -u origin master
  
```

---

**...or import code from another repository**

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

These are commands we need to do in terminal once repository is created

```

echo "# Scala_Coding" >> README.md (here we
initializing our repository in our terminal)
git init
git add README.md (adding want file to commit)
git commit -m "first commit". ( committing with message,
before pushing into repository we need to commit
without this commit command we cann't push the file to
repo)
git remote add origin https://github.com/RajeshJakkula/
Scala_Coding.git
git push -u origin master. ( end step push file to git
repo)
  
```

```

[Rajeshs-MBP:Python_Coding rajeshjakkula$ git branch
* master
Rajeshs-MBP:Python_Coding rajeshjakkula$ █
  
```

git remote add origin(local rename of repository, we can assign any name) [https://github.com/RajeshJakkula/Scala\\_Coding.git](https://github.com/RajeshJakkula/Scala_Coding.git)  
 for first time pushing into repository we need to mention the locally rename name, if you

have multiple branch locally.

ex:1

```
git remote add Scala_coding https://github.com/RajeshJakkula/Scala_Coding.git
```

```
git push -u Scala_coding
```

```
git pull <remote-repo> <remote-branch>
```

```
git fetch <remote-repo> <remote-branch>
```

**Git pull:** when you do a ***git pull***, it gets all the changes from the remote or central repository and attaches it to your corresponding branch in your local repository.

**Git fetch:** when you do a ***git fetch***, it gets all the changes from the remote repository, stores the changes in a separate branch in your local repository and if you want to reflect those changes in your corresponding branches, use a ***git merge*** to do that.

```
git clone [url]
```

```
git clone https://github.com/RajeshJakkula/Python\_Coding
```

Where this clone command is useful?

for example two peoples are working in project and there a dependency for your task

first person need to develop the code and you need that code in your machine. you need to clone the entire project into your machine

clone( entire project are file level also).

**git status:** once you developed few files and you want to what are all the file you made changes

```

Rajeshs-MBP:Python_Coding rajeshjakkula$ git status
On branch master
Your branch is up to date with 'github/master'.

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:   Context_Manager.py
        modified:   Generators.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .idea/
        data.txt
        foo.txt

no changes added to commit (use "git add" and/or "git commit -a")
Rajeshs-MBP:Python_Coding rajeshjakkula$ █

```

git log:  
 what the last commits, who did the last commit to a  
 file( informatica version)

```

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .idea/
        data.txt
        foo.txt

no changes added to commit (use "git add" and/or "git commit -a")
Rajeshs-MBP:Python_Coding rajeshjakkula$ git log
commit bc1e29e5bf1abb899444585a8ff88f27430950a1 (HEAD -> master, git
Author: Rajesh Jakkula <rajeshjakkula@Rajeshs-MBP.home>
Date:   Mon Oct 21 15:39:45 2019 -0700

    Scala Code first to ScalaRepo

commit cacff70aeb0569d4cc0e51dd3b83c23e662b56bb
Author: Rajesh Jakkula <rajeshjakkula@Rajeshs-MBP.home>
Date:   Mon Oct 21 15:35:04 2019 -0700

    scala first commit

```

```
commit 3b6f3a3f3e1dfb0593b672927b6b92a28e0f829e
Author: Rajesh Jakkula <rajeshjakkula@Rajeshs-MBP.home>
Date:   Mon Oct 21 15:32:35 2019 -0700
```

#### Scala Code

```
commit 268f5a3e12fce63b12fb1a81b99c9c6c606fc667
Author: Rajesh Jakkula <rajeshjakkula@Rajeshs-MBP.home>
Date:   Sun Oct 20 19:00:21 2019 -0700
```

#### Generatos with number example

```
commit 0a8421d432aad8e5b1b0e1eca20d7d642eae8d3d
Author: Rajesh Jakkula <rajeshjakkula@Rajeshs-MBP.home>
Date:   Sat Oct 19 22:59:37 2019 -0700
```

#### Cursor Position in a file

```
commit 76f589aea89e5a5eef8e3c981aadeb9a8d4bdcf4
Author: Rajesh Jakkula <rajeshjakkula@Rajeshs-MBP.home>
Date:   Sat Oct 19 19:23:48 2019 -0700
```

#### Changes

```
commit 2e237da65951554839ac85a6718f32c88fe67069
Author: Rajesh Jakkula <rajeshjakkula@Rajeshs-MBP.home>
Date:   Sat Oct 19 19:20:45 2019 -0700
```

#### First Commit

```
Rajeshs-MBP:Python_Coding rajeshjakkula$
```

### Undo commit:

Once you committed and you want to revert back those commits  
**git revert commit\_id**

**git revert 2e237da.....**

once revert is also committed your commit id 2e237df and you want to set it back

git reset —hard 2e237df, now your able to see your code changes in your file.

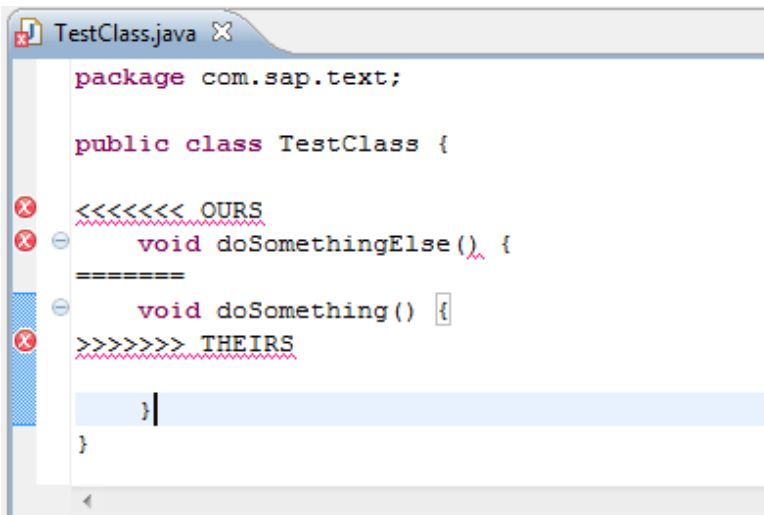
git stash:( interview question)

git stash list  
git stash apply  
git stash pop( unstash the stash files)

git rebase:( interview question)

merge conflicts:( interview question)

if you two people's are trying to commit the code into same file at a time your get merge conflicts



```
TestClass.java
package com.sap.text;

public class TestClass {

<<<<<< OURS
    void doSomethingElse() {
=====
    void doSomething() {
>>>>>> THEIRS
    }
}
```

how to resolve merge conflicts:  
manually you need to delete

\* for every commit into repository you will have one ID.

**git checkout:**

**git checkout -, -filename.py ( remove ,)**

you made changes and do the checkout to the file, your changes will be remove from the file.

git checkout —. ( for all files)

for example you committed a code and any other person don't want to make any change on that file

you will do checkout that file

your commit file id 8483.

git check 8483.( now this file is going to be checked out no one is not able to make any

changes, until again it is checked in)

undo checkout:

git revert

git reset —hard

**git diff: ( unix dif**

**we will compare the old and new code change files.**

