**GIT Assignment 2**

1Q. What does the command 'git status' do?

Ans.

***Git status command is used in Git to know the status of the***

***working tree***. It shows the state of your working directory and helps

you see all the files which are untracked by Git, staged, or unstaged.

In shorter terms, Git will show you any difference in the current tree

and the HEAD pointer (Refer [***Git Terminologies***](https://www.toolsqa.com/git/first-commit-in-git/)). Along with this Git

status will also show you the changed or new file in the repository.

We will explore each of these ones through examples. We will cover

the following cases of using Git Status.

* ***On a newly created file***
* ***On a modified file***
* ***On a deleted file***

***Git Status when working tree is Clean***

Before we make any new changes let's see the status of the Git repository in

which we are working (*First Project*). Refer [***Git Repository***](https://www.toolsqa.com/git/create-git-repository/)

1.Open ***Git Bash***

2.Navigate to the directory of the repository (*First Project*).

3.Type the following command git status and Press ***enter*** to execute the

command.

As clearly visible, there is nothing to commit and the working tree is

clean i.e. there are no untracked files.

### *****Git Status when a new file is Created*****

Now let's make some changes and see what happens.

1.Create a file ***ABC.txt*** this using command:

touch ABC.txt

***Note***: The ***touch command*** is the easiest way to create new, empty files.

2.Press ***enter*** to create the file.

3.Once the file is created, execute the git status command again.

The message is quite clear that ***there is nothing to commit but***

***untracked files are present***. Untracked files as discussed in

the ***[Gitcommit](https://www.toolsqa.com/git/first-commit-in-git/)*** tutorial is the file that is not yet added to the staging

area.

There are two types of untracked files in Git. One is the

normal project file that we saw above and the other is the binary

files tha are inside the repository like *.obj* or .exe.

Git in the above image also provides suggestion so as to use ***git***

***add*** to track the file. These suggestions will be provided according to

the situation by the status command. You will get different

suggestions in different cases as we will see throughout this tutorial.

4.Add the file to the staging area.

After you have added the file, again look at the git

status command and what does it say this time.

***new file: ABC.txt***: This tells you that a new file is present (with its

name) and also been added to staging.

This comes very handily when you have to commit, before

committing blindly. Looking at the git status can help a lot to avoid

the changes that we don't want to commit at all.

5. ***Commit*** this file. ([***Git Commit***](https://www.toolsqa.com/git/first-commit-in-git/) ***tutorial***)

After committing the file successfully, you can once again check

the git status and it will show you that there is nothing to commit.

2Q. How to delete a Git local branch?

Whether you use GitFlow, GitHub Flow or any other branch driven

development strategy, you will inevitably end up with a local Git repository

filled with branches you no longer need. Which is why it’s good to know the

command to delete Git branches locally and permanently so they don’t

pollute your local development environment.

## How to delete local Git branches

To issue the command to delete a local Git branch, follow these steps:

1. Open a [Git BASH window](https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/How-to-customize-Git-Bash-Shell-prompt-settings) or Command Window in the root of your Git repository
2. If necessary, use the git switch or checkout command to move off the branch you wish to delete
3. Issue the

git branch –delete <branchname>

command to delete the local branch

1. Run the git branch -a command to verify the local Git branch is

Deleted

**Delete local Git branch command**

The command to delete a local git branch can take one of two forms:

* git branch –delete old-branch
* git branch -d old-branch

The only difference is the fact that the second local branch delete Git

command uses an abbreviated syntax. Both commands do the exact same

thing.

3Q. How can I add a project to Git that already exists?

[**Introduction**](https://www.digitalocean.com/community/tutorials/how-to-push-an-existing-project-to-github#introduction)

GitHub is a cloud-hosted Git management tool. Git is *distributed* version

control, meaning the entire repository and history lives wherever you put it.

People tend to use GitHub in their business or development workflow as a

managed hosting solution for backups of their repositories. GitHub takes

this even further by letting you connect with coworkers, friends,

organizations, and more.

## [Prerequisites](https://www.digitalocean.com/community/tutorials/how-to-push-an-existing-project-to-github#prerequisites)

To initialize the repo and push it to GitHub you’ll need:

1. [A free GitHub Account](https://github.com/)

2.[git installed on your local machine](https://www.digitalocean.com/community/tutorials/how-to-contribute-to-open-source-getting-started-with-git#check-if-git-is-installed)

## [Step 1 — Create a new GitHub Repo](https://www.digitalocean.com/community/tutorials/how-to-push-an-existing-project-to-github#step-1-create-a-new-github-repo)

Sign in to GitHub and [create a new empty repo](https://github.com/new). You can choose to

either initialize a README or not. It doesn’t really matter because we’re

just going to override everything in this remote repository anyways.

## [Step 2 — Initialize Git in the project folder](https://www.digitalocean.com/community/tutorials/how-to-push-an-existing-project-to-github#step-2-initialize-git-in-the-project-folder)

From your terminal, run the following commands after navigating to the

folder you would like to add.

### [Initialize the Git Repo](https://www.digitalocean.com/community/tutorials/how-to-push-an-existing-project-to-github#initialize-the-git-repo)

Make sure you are in the root directory of the project you want to push to

GitHub and run:

$git init

This step creates a hidden .git directory in your project folder, which

the git software recognizes and uses to store all the metadata and

version history for the project.

### [Add the files to Git index](https://www.digitalocean.com/community/tutorials/how-to-push-an-existing-project-to-github#add-the-files-to-git-index)

### $git add -A

The git add command is used to tell git which files to include in a

commit, and the -A (or --all) argument means “include all”.

### [Commit Added Files](https://www.digitalocean.com/community/tutorials/how-to-push-an-existing-project-to-github#commit-added-files)

### git commit -m 'Added my project'

The git commit command creates a new commit with all files that

have been “added”. The -m (or --message) sets the message that will

be included alongside the commit, used for future reference to

understand the commit. In this case, the message is: 'Added my

project'.

### [Add a new remote origin](https://www.digitalocean.com/community/tutorials/how-to-push-an-existing-project-to-github#add-a-new-remote-origin)

### git remote add origin [git@github.com:sammy/my-new-project.git](mailto:git@github.com:sammy/my-new-project.git)

In git, a “remote” refers to a remote version of the same repository,

which is typically on a server somewhere (in this case, GitHub). “origin”

is the default name git gives to a remote server (you can have multiple

remotes) so git remote add origin is instructing git to add the

URL of the default remote server for this repo.

### [Push to GitHub](https://www.digitalocean.com/community/tutorials/how-to-push-an-existing-project-to-github#push-to-github)

### git push -u -f origin main

The -u (or --set-upstream) flag sets the remote origin as

the upstream reference. This allows you to later perform git push and git pull commands without having to specify an origin since we always want GitHub in this case.

The -f (or --force) flag stands for force. This will automatically overwrite everything in the remote directory. We’re using it here to overwrite the default README that GitHub automatically initialized.

### [All together](https://www.digitalocean.com/community/tutorials/how-to-push-an-existing-project-to-github#all-together)

### [All together](https://www.digitalocean.com/community/tutorials/how-to-push-an-existing-project-to-github#all-together)

1. git init
2. git add -A
3. git commit -m 'Added my project'
4. git remote add origin git@github.com:sammy/my-new-project.git
5. git push -u -f origin main

### 4Q.What exactly is a Git fork? What are the distinctions between a fork, a branch, and a clone?

### Ans.

### A fork in Git is simply a copy of an existing repository in which the

### new owner disconnects the codebase from previous committers. A

### fork often occurs when a developer becomes dissatisfied or

### disillusioned with the direction of a project and wants to detach

### their work from that of the original project. When a git fork occurs,

### previous contributors will not be able to commit code to the new

### repository without the owner giving them access to the forked

### repo, either by providing developers the publicly accessible Git

### URL, or by providing explicit access through user permission in

### tools like GitHub or GitLab.

### Difference between branch, fork and clone

Whether you use either branching or forking, and to what extent, depends on your working environment. There are lots of ways colleagues can work with and combine fork and branch functionalities. You can [google for discussions about this](https://www.google.com/search?q=dvcs+fork+branch). Generally, for hosted systems, forks work well in situations where, as a repository admin:

* You don't want to manage user access on your repository.
* You want fine-grain control over merging.
* You expressly want to support independent branches.
* You want to discard experiments and changes easily.

We recommend branching for development organizations on Bitbucket; We use a modified form of [Vincent Driessen's GitFlow](http://nvie.com/posts/a-successful-git-branching-model/)technique. Bitbucket branches are useful when:

* You have a small group of programmers who trust each other and are in close communication.
* You are willing to give the development organization write access to a repository.
* You have a rapid iteration cycle.

Ultimately, though it is your choice – branch or fork – Bitbucket supports both.

5Q.What is the difference between HEAD, working tree, and index in GIT?

Ans.

Git as a version-control-system manages and manipulates three trees in its normal operation:

* **HEAD:**Last commit snapshot, next parent
* **Index:** Proposed next commit snapshot
* **Working Directory:**Sandbox

**Head**

HEAD is the pointer to the current branch reference, which is in turn a pointer to the last commit made on that branch. That means HEAD will be the parent of the next commit that is created. It’s generally simplest to think of HEAD as the snapshot of your last commit on that branch.

**What does it contain?**

Use***git ls-files -s***to see what it looks like. You should see something like this

**Working Tree**

This is where your files reside and where you can try changes out before committing them to your staging area (index) and then into history.

Let’s see how these three trees work together?

Git’s typical workflow is to record snapshots of your project in successively better states, by manipulating these three trees.

To get a good visualized understanding, consider this scenario. Say you go into a new directory with a single file in it. Call this v1 of the file. It is indicated in blue. Running ***git init*** will create a Git repository with a HEAD reference that points to the unborn master branch

If we run git status, we’ll see no changes, because all three trees are the same.

6Q. Which GIT command is used to change branches?

Ans.

To switch to an existing branch, you can use git

checkout again (without the -b flag) and pass the name of the

branch you want to switch to:

(my-feature)$ git checkout master

Switched to branch 'master'

(master)$

There is also a handy shortcut for returning to the previous

branch you were on by passing -  to git checkout instead of a

branch name:

(my-feature)$ git checkout -

Switched to branch 'master'

(master)$ git checkout -

Switched to branch 'my-feature'

(my-feature)$

7Q. What is the difference between GitHub and Git?

Ans.

**Git:** Git is a distributed version control system for tracking changes in

source code during software development. It is designed for

coordinating work among programmers, but it can be used to track

changes in any set of files. Its goals include speed, data integrity, and

support for distributed, non-linear workflows.

**GitHub:** GitHub is a web-based Git repository hosting service, which

offers all of the distributed revision control and source code

management (SCM) functionality of Git as well as adding its own

features.

| **S.No.** | **Git** | **GitHub** |
| --- | --- | --- |
| 1. | Git is a software. | GitHub is a service. |
| 2. | Git is a command-line tool | GitHub is a graphical user interface |
| 3. | Git is installed locally on the system | GitHub is hosted on the web |
| 4. | Git is maintained by linux. | GitHub is maintained by Microsoft. |
| 5. | Git is focused on version control and code sharing. | GitHub is focused on centralized source code hosting. |
| 6. | Git is a version control system to manage source code history. | GitHub is a hosting service for Git repositories. |
| 7. | Git was first released in 2005. | GitHub was launched in 2008. |
| 8. | Git has no user management feature. | GitHub has a built-in user management feature. |
| 9. | Git is open-source licensed. | GitHub includes a free-tier and pay-for-use tier. |
| 10. | Git has minimal external tool configuration. | GitHub has an active marketplace for tool integration. |
| 11. | Git provides a Desktop interface named Git Gui. | GitHub provides a Desktop interface named GitHub Desktop. |
| 12. | Git competes with CVS, Azure DevOps Server, Subversion, Mercurial, etc. | GitHub competes with GitLab, Bit Bucket, AWS Code Commit, etc. |

8Q.What are some of the advantages of using the Variation Control

System? Which programming language is used in Git?

Ans.

Having a GitHub repo makes it easy for you to keep track of collaborative and

personal projects - all files necessary for certain analyses can be held together and

people can add in their code, graphs, etc. as the projects develop. Each file on

GitHub has a history, making it easy to explore the changes that occurred to it at

different time points. You can review other people’s code, add comments to certain

lines or the overall document, and suggest changes. For collaborative projects,

GitHub allows you to assign tasks to different users, making it clear who is

responsible for which part of the analysis. You can also ask certain users to

review your code. For personal projects, version control allows you to keep track o

f your work and easily navigate among the many versions of the files you create,

whilst also maintaining an online backup.

Github supported language

Core languages for GitHub features include C, C++, C#, Go, Java, JavaScript, PHP,

Python, Ruby, Scala, and TypeScript. For features that support package managers, the

currently supported package managers are included in the table with their relevant

languages.

9Q. List out some Git repository features?

Ans.

## **Features of Git**

* Tracks history
* Free and open source
* Supports non-linear development
* Creates backups
* Scalable
* Supports collaboration
* Branching is easier
* Distributed development