GIT HUB RELATED QUESTIONS

1.WHAT IS THE DIFFERENCE BETWEEN GIT AND GITHUB?

ANSWER:--- Git is a revision control system, a tool to manage your source code history. GitHub is a hosting service for Git repositories. So they are not the same thing: Git is the tool, GitHub is the service for projects that use Git.

2.WHAT IS THE COMMAND FOR REMOTE?

ANSWER:--- git remote origin

3.WHAT WE CAN WRITE INSTEAD OF ORIGIN IN GITHUB?

ANSWER:--- We can write remote url instead of origin in github.

4.HOW TO PUSH FROM MASTER BRANCH?

ANSWER:--- The git push command takes two arguments:

* A remote name, for example, origin
* A branch name, for example, master

5.WHAT IS THE COMMAND OF STATUE IN GITHUB?

ANSWER:--- Shows you the status of files in the index versus the working directory. It will list out files that are untracked (only in your working directory), modified (tracked but not yet updated in your index), and staged (added to your index and ready for committing). Example:

1.git status

# On branch master #

# Initial commit #

# Untracked files: #

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

6.WHAT IS THE COMMAND FOR ADDING A FILE?

ANSWER:--- Adds files changes in your working directory to your index. Example:

1.git add .

7.WHAT IS THE COMMAND TO SEE THE DIFFERENCE IN GIT?

ANSWER:---- Generates patch files or statistics of differences between paths or files in your git repository, or your index or your working directory. Example:

1.git diff

8.WHAT IS THE UNTRACKED FILE IN GIT?

ANSWER:--- **untracked file** is each **file** that was not there in previous snapshot of the repository(**file** that is newly introduced). ... You can start tracking using **git** add, a **file** directly goes to staging area when you use **git** add, now that **file** is both tracked and staged. You can unstage any staged **file** using**git** reset -- filename.

9.HOW TO IGNORE FILE IN GIT

ANSWER:--- Run **git** reset name\_of\_file to unstage the **file** and keep it. In case you want to also remove given **file** from the repository (after pushing), use **git** rm --cached name\_of\_file . Add the path(s) to your **file**(s) which you would like to **ignore** to your .**gitignore file** (and commit them).

10.HOW TO CREATE A BRANCH?

#### ANSWER:--- To create a branch from Bitbucket

1. From the repository, click **+** in the global sidebar and select **Create a branch** under **Get to work**.
2. From the popup that appears, select a **Type** (if using the [Branching model](https://confluence.atlassian.com/x/TwlODQ#branching_model)), enter a **Branch name** and click **Create**.
3. After you create a branch, you need to check it out from your local system. Use the fetch and checkout commands that Bitbucket provides, similar to the following: EXAMPLE:- $git fetch && git checkout <feature>
4. Make your changes locally and then add, commit, and push your changes to the <feature> branch: EXAMPLE:-- $ git add .   
   $ git commit -m "adding a change from the feature branch"   
   $ git push origin <feature>
5. Click the **Source** page of your repository. You should see both the master and the <feature> branch in the branches dropdown. When you make commits to the feature branch, you'll see the files specific to that branch.

#### To create a branch locally

You can create a branch locally as long as you have a cloned version of the repo.

1. From your terminal window, list the branches on your repository.

$ git branch 

\* master

This output indicates there is a single branch, the master and the asterisk indicates it is currently active.

1. Create a new feature branch in the repository

$ git branch <feature\_branch>

1. Switch to the feature branch to work on it.

$ git checkout <feature\_branch>

You can list the branches again with the git branch command.

1. Commit the change to the feature branch:

$ git add .   
$ git commit -m "adding a change from the feature branch"

1. Switch back to the master branch.

$ git checkout master

1. Push the feature branch to Bitbucket:

$ git push origin <feature\_branch>

1. View the **Source** page of your repository in Bitbucket. You should see both the master and the feature branch. When you select the feature branch, you see the **Source** page from that perspective. Select the feature branch to view its **Recent commits**.

11.WHAT IS A MASTER BRANCH.

ANSWER:- A **branch in Git** is simply a lightweight movable pointer to one of these commits. The default **branch**name in **Git** is **master**. As you initially make commits, you're given a **master branch** that points to the last commit you made. Every time you commit, it moves forward automatically.

12.WHAT IS MASTER IN GITHUB.

In Git, "master" is a naming convention for a branch. After cloning (downloading) a project from a remote server, the resulting local repository has a single local branch: the so-called "master" branch. This means that "master" can be seen as a repository's "default" branch.