**Git commands**

**Git bash:**

Git Bash is an application for Microsoft Windows environments which provides an emulation layer for a Git command line experience. Bash is an acronym for Bourne Again Shell. A shell is a terminal application used to interface with an operating system through written commands. Bash is a popular default shell on Linux and macOS. Git Bash is a package that installs Bash, some common bash utilities, and Git on a Windows operating system.

**Git bash commands:**

Git Bash is packaged with additional commands that can be found in the /usr/bin directory of the Git Bash emulation. Git Bash can actually provide a fairly robust shell experience on Windows. Git Bash comes packaged with the following shell commands which are outside the scope of this document: [Ssh](https://man.openbsd.org/ssh.1), [scp](https://linux.die.net/man/1/scp), [cat](http://man7.org/linux/man-pages/man1/cat.1.html), [find](https://linux.die.net/man/1/find).

In addition the previously discussed set of Bash commands, Git Bash includes the full set of Git core commands discussed through out this site. Learn more at the corresponding documentation pages for [git clone](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-clone), [git commit](https://www.atlassian.com/git/tutorials/saving-changes/git-commit), [git checkout](https://www.atlassian.com/git/tutorials/using-branches/git-checkout), [git push](https://www.atlassian.com/git/tutorials/syncing/git-push), and more

Here are some the Git commands :

* ****git config****
* ****git init****
* ****git clone****
* ****git add****
* ****git commit****
* ****git diff****
* ****git reset****
* ****git status****
* ****git rm****
* ****git log****
* ****git show****
* ****git tag****
* ****git branch****
* ****git checkout****
* ****git merge****
* ****git remote****
* ****git push****
* ****git pull****
* ****git stash****

### **git config**

Usage: git config –global user.name “[name]”

Usage: git config –global user.email “[email address]”

This command sets the author name and email address respectively to be used with your commits.

IMG_256

### **git init**

Usage: git init [repository name]

This command is used to start a new repository.

IMG_257

### **git clone**

Usage: git clone [url]

This command is used to obtain a repository from an existing URL.



### **git add**

Usage: git add [file]

This command adds a file to the staging area.

IMG_259

Usage: git add \*

This command adds one or more to the staging area.

IMG_260

### **git commit**

Usage: git commit -m “[ Type in the commit message]”

This command records or snapshots the file permanently in the version history.



Usage: git commit -a

This command commits any files you’ve added with the git add command and also commits any files you’ve changed since then.

IMG_262

### **git diff**

Usage: git diff

This command shows the file differences which are not yet staged.



 Usage: git diff –staged

This command shows the differences between the files in the staging area and the latest version present.



Usage: git diff [first branch] [second branch]

This command shows the differences between the two branches mentioned.



### **git reset**

Usage: git reset [file]

This command unstages the file, but it preserves the file contents.



Usage: git reset [commit]

This command undoes all the commits after the specified commit and preserves the changes locally.

IMG_267

Usage: git reset –hard [commit]  This command discards all history and goes back to the specified commit.

IMG_268

Learn how to [connect Git secrets with a Jenkins pipeline](https://dzone.com/articles/jenkins-cicd-with-git-secrets).

### **git status**

Usage: git status

This command lists all the files that have to be committed.



### **git rm**

Usage: git rm [file]

This command deletes the file from your working directory and stages the deletion.

IMG_270

### **git log**

Usage: git log

This command is used to list the version history for the current branch.



Usage: git log –follow[file]