Basic GIT Commands

Saturday, 28. November 2020 17:47

Need to learn some basic GIT commands? You've come to the right place. Read on to discover our handy cheat sheet that you can use
Let's get started!

Understanding the GIT Workflow

GIT is the **most widely used** open-source VCS (version control system) that allows you to track changes made to files. Companies and p
applications.
A GIT project consists of three major sections: **the working directory**, **the staging area**, and **the git directory**.
The working directory is where you add, delete, and edit the files. Then, the changes are staged (indexed) in the staging area. After you
git directory.
Everyone can use GIT as it is available for **Linux**, **Windows**, **Mac**, and **Solaris**. The software may have a steep learning curve, but there a

Basic GIT Commands

Here are some basic GIT commands you need to know:

- **git init** will create a new local GIT repository. The following Git command will create a repository in the current directory:

git init

- Alternatively, you can create a repository within a new directory by specifying the project name:

git init [project name]

- **git clone** is used to copy a repository. If the repository lies on a remote server, use:

git clone username@host:/path/to/repository

- Conversely, run the following basic command to copy a local repository:

git clone /path/to/repository

- **git add** is used to add files to the staging area. For example, the basic Git following command will index the temp.txt file:

git add <temp.txt>

- **git commit** will create a snapshot of the changes and save it to the git directory.

git commit –m "Message to go with the commit here"

- Note that any committed changes won't make their way to the remote repository.

- **git config** can be used to set user-specific configuration values like email, username, file format, and so on. To illustrate, the c

git config --global user.email youremail@example.com

- The –global flag tells GIT that you're going to use that email for all local repositories. If you want to use different emails for dif

git config --local user.email youremail@example.com

- **git status** displays the list of changed files together with the files that are yet to be staged or committed.

git status

- **git push** is used to send local commits to the master branch of the remote repository. Here's the basic code structure:

git push origin <master>

- Replace <master> with the branch where you want to push your changes when you're not intending to push to the master br

- **git checkout** creates branches and helps you to navigate between them. For example, the following basic command creates a

command git checkout -b <branch-name>

- To switch from one branch to another, simply use:

git checkout <branch-name>

- **git remote** lets you view all remote repositories. The following command will list all connections along with their URLs:

for daily reference.

programmers usually use GIT to collaborate on developing software and

u commit your changes, the snapshot of the changes will be saved into the

are lots of **tutorials** ready to help you.

ommand for setting up an email will look like this:

fferent repositories, use the command below:

anch.

new branch and automatically switches you to it:

git remote lets you view all remote repositories. The following command will list all connections along with their URLs:

```
git remote –v
```

- To connect the local repository to a remote server, use the command below:

```
git remote add origin <host-or-remoteURL>
```

- Meanwhile, the following command will delete a connection to a specified remote repository:

```
git remote rm <name-of-the-repository>
```

- **git branch** will list, create, or delete branches. For instance, if you want to list all the branches present in the repository, the c

```
git branch
```

- If you want to delete a branch, use:

```
git branch –d <branch-name>
```

- **git pull** merges all the changes present in the remote repository to the local working directory.

```
git pull
```

- **git merge** is used to merge a branch into the active one.

```
git merge <branch-name>
```

- **git diff** lists down conflicts. In order to view conflicts against the base file, use

```
git diff --base <file-name>
```

- The following basic command is used to view the conflicts between branches before merging them:

```
git diff <source-branch> <target-branch>
```

- To list down all the present conflicts, use:

```
git diff
```

- **git tag** marks specific commits. Developers usually use it to mark release points like v1.0 and v2.0.

```
git tag <insert-commitID-here>
```

- **git log** is used to see the repository's history by listing certain commit's details. Running the command will get you an output t

```
commit 15f4b6c44b3c8344caasdac9e4be13246e21sadw
Author: Alex Hunter <alexh@gmail.com>
Date:   Mon Oct 1 12:56:29 2016 -0600
```

- **git reset** command will reset the index and the working directory to the last git commit's state.

```
git reset --hard HEAD
```

- **git rm** can be used to remove files from the index and the working directory.

```
git rm filename.txt
```

- **git stash** command will temporarily save the changes that are not ready to be committed. That way, you can go back to that p

```
git stash
```

- **git show** is a command used to view information about any git object.

```
git show
```

- **git fetch** allows users to fetch all objects from the remote repository that don't currently reside in the local working directory.

```
git fetch origin
```

- **git ls-tree** allows you to view a tree object along with the name, the mode of each item, and the blob's SHA-1 value. Let's say

```
git ls-tree HEAD
```

- **git cat-file** is used to view the type and the size information of a repository object. Use the -p option along with the object's SI

```
git cat-file –p d670460b4b4aece5915caf5c68d12f560a9fe3e4
```

- **git grep** lets users search through committed trees, working directory, and staging area for specific phrases and words. To sea

```
git grep "www.hostinger.com"
```

- **gitk** shows the graphical interface for a local repository. Simply run:

```
gitk
```

- **git instaweb** allows you to browse your local repository in the git-web interface. For instance:

```
git instaweb –httpd=webrick
```

- **git gc** will clean unnecessary files and optimize the local repository.

ommand should look like this:

that looks like this:

roject later on.

you want to see the HEAD, use:

HA-1 value to view the information of a specific object, for example:

rch for www.hostinger.com in all files, use:

git gc will clean unnecessary files and optimize the local repository.

```
git gc
```

- **git archive** lets users create a zip or a tar file containing the constituents of a single repository tree. For instance:

```
git archive --format=tar master
```

- **git prune** deletes objects that don't have any incoming pointers.

```
git prune
```

- **git fsck** performs an integrity check of the git file system and identifies any corrupted objects.

```
git fsck
```

- **git rebase** is used to apply certain changes from one branch to another. For instance:

```
git rebase master
```

Basic GIT Commands Cheat Sheet in .pdf

If you are just starting out with GIT, it can be hard to remember even the basic commands. For that reason, we've put together a GIT c
or print it out so you'll always have it ready when you're stuck remembering GIT commands.

Download (size:1.2MB)

Conclusion

Learning basic GIT commands will go a long way for developers as they can easily control the projects' source code. It might take some
sheet will be helpful for you.

heat sheet to help you master the software. Save the file to your devices

e time to commit to remembering all of them, but hopefully, our GIT cheat