|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Git init এর মাধ্যমে ইনিশিয়ালাইজ করতে হবে। git ইনস্টল করার পর রাইট বাটন ক্লিক করলে Git GUI here ও Git Bash here পপআপ মেনু আসবে। গিট এর গুইন এবং গ্রাফিক্যাল ইউজার ইনটার ফেজ এই ২টি ভার্সন ব্যবহার করা যাবে।আমরা এখন দেখব কমান্ড এর মাধ্যমে কিভাবে গিট ব্যবহার করা যায়।কমান্ড প্রম্পট চালূ করে git --version লিখে এন্টার করলে ইনস্টলকৃত গিট এর ভার্সন দেখা যাবে।প্রথমে কাজ করার জন্য আমরা একটি ফোল্ডার তৈরী করব। ফোল্ডারটি ওপেন করে ফোল্ডারে প্রবেশ করে রাইট বাটনে ক্লিক করে git Bash here এ ক্লিক করব।গিটের কমান্ড লাইন চালু হলে git init লিখে এন্টার দিব। তাহলে গিটের রিপোজিটরী তৈরী হবে।git status কমান্ড দিয়ে অবস্থান দেখা যাবে।গিটে কাজ করার জন্য ফাইল তৈরী করতে হবে।আমরা index.html নামে একটি ফাইল তৈরী করলাম। ফাইল তৈরী করলে ওয়ার্কিং এরিয়া তৈরী হবে।এটি হবে আনট্রেক ফাইল (untracked files). এখন git add index.html কমান্ড দিলে ফাইলটি ওয়ার্কিং এরিয়া হতে staging area তে আসবে।অনেক গুলো ফাইল আনট্রাক্ট অবস্থায় থাকলে ফাইলগুলোকে স্টেজিং এরিয়াতে নেওয়ার জন্য git add. কমান্ড ব্যবহার করলে আনট্রাক্ট অবস্থায় থাকা সকল ফাইল স্টেজিং এরিয়াতে আসবে। এখন ফাইলটি কমিট করার উপযোগী হিসেবে থাকবে। ফাইল স্টেজিং এরিয়াতে থাকা অবস্থায় git commit -m "message" কমান্ড দিলে স্টেজিং এরিয়া হতে ফাইলটি রিপোজিটরী এরিয়াতে আসবে। এখানে "message" এর মধ্যে ফাইলটি কেন রিপোজিটরী এরিয়াতে নেওয়া হল তার মেইনিং ফূল অর্থ লিখতে হবে।ফাইল রিপোজিটরীতে আনার পর ফাইলের মধ্যে সকল পরিবর্তন রেকর্ড হবে।এখন আমরা index.html ফাইলটি খোলে তাতে কিছু কোড লিখে ফাইলটি সেভ করি। এখন গিট কমান্ড লাইনে git status লিখে এন্টার করলে modified : index.html দেখাবে। তার মানে ফাইলটি পরিবর্তন করা হয়েছে।এখন ফাইলটি রিপোজিটরীতে আছে। এখন গিটের কমান্ড লাইনে git add index.html লিখে এন্টার করলে ফাইলটি রিপোজিটরী হতে স্টেজিং এরিয়াতে আসবে।এখন git commit -m "second comit" লিখে এন্টার করলে ২য় কমিট সম্পন্ন হবে।git log কমান্ড দিয়ে কমিট গুলো দেখা যাবে। কমিট গুলোকে এক লাইনে দেখার জন্য git log --oneline কমান্ড দিতে হবে।  কখনো যদি মনে হয় ভূল করে কমিট করা হয়েছে তাহলে পূর্বের কমিটে ফেরৎ যাওয়ার জন্য git checkout commitnumber কমান্ড দিলে পূর্বের কমিটে ফেরৎ যাওয়া যায়। |  |  |
| 3 | গিটে অনেক লোক মিলে কাজ করা যায়। তাই গিটে ব্যবহার কারীদের জন্য আডেনটিফিকেশন দরকার। এজন্য প্রত্যেক ব্যবহার কারীর জন্য ইউজার নেম ও ইমেইল সেট করে দিতে হয়। এটি দুইভাবে করা যায়। প্রত্যেক প্রজেক্টের জন্য একই ইমেইল ও পাসওয়ার্ড গ্লোবালি সেট করে দেওয়া যায় অথবা প্রত্যেক প্রজেক্টের জন্য লোকালী সেট করে দেওয়া যায়। গ্লোবালী সেট করার জন্য কমিট করার আগে গিট কমান্ড লাইনে git config --global user.name "Name here" লিখে এন্টার করতে হবে এবং ইমেইল দেওয়ার জন্য git config -- global user.email "email here" লিখে এন্টার করতে হবে। ইউজার নেম ও ইমেইল দেওয়ার আগে কোন কমিট করা যাবে না।ইউজার নেম ও ইমেইল আগে থেকে সেট করা আছে কিনা বা কি ইউজার নেম ও ইমেইল সেট করা আছে তা জানার জন্য git config --list কমান্ড দিতে হবে। স্টেজিং এরিয়া হতে ওয়াকিং এরিয়াতে ফেরৎ আসতে চাইলে git reset HEAD filename কমান্ড দিলে স্টেজিং এরিয়ার ফাইল ওয়ার্কিং এরিয়াতে আসবে। |  |  |
| 4 | Git init কমান্ড এর মাধ্যমে রিপোজিটরী তৈরী করে আমরা cold.txt, hot.txt নামে ২টি ফাইল তৈরী করি।এখন git status কমান্ড এর মাধ্যমে দেখলে দেখা যাবে Untracked files হিসেবে লাল কালারের ২টি ফাইলের নাম দেখা যাবে। এখন ২টি ফাইলকে আমরা git add . কমান্ড দিয়ে স্টেজিং অবস্থায় নিয়ে যাবো। এখন git status কমান্ড দিলে দেখা যাবে ২টি ফাইলের নাম সবুজ কালার হয়েগেছে । এখন git commit -m "2 txt file added" এ কমান্ডের মাধ্যমে আমরা ২টি ফাইলকে কমিটি করি। এখন আমরা chocolate.txt নামে আরও ১টি ফাইল তৈরী করে গিট এড কমান্ডের মাধ্যমে স্টেজিং এরিয়াতে নিয়ে যায় এবং git commit -m "chocolate added" দিয়ে কমিট করি।  এখন cold.txt ফাইলটি খুলে এতে ice cube added টাইপ করে ফাইলটি সেভ করি। git add . কমান্ডের মাধ্যমে ফাইলটি স্টেজিং এরিয়াতে নিয়ে যায় এবং git commit -m " ice cube added " কমান্ড দিয়ে কমিটি করি।এখন মনে আমরা ভূল করে cold.txt ফাইলটিতে ice cube added লেখাটি ভূল করে যুক্ত করেছি। এখন আমরা ice cube added এই লেখাটি লেখার আগের অবস্থায় ফিরে যেতে চাই।এখন আমরা git log --oneline কমান্ড দিয়ে কমিট গুলো দেখি।এখানে ৩টি কমিট দেখতে পাচ্ছি।  git log --oneline  2e10e4b (HEAD -> master) ice cube added  2c41b0e chocolate added (আমরা এই কমিটে ফিরে যেতে চায়)  d83f090 2 txt file added  এজন্য আমরা git checkout 2c41b0e এখন এন্টার প্রেস করি। এখন আমরা cold.txt ফাইলটি খোলে দেখি যে cube added আর নেই।অর্থাত লেখাটি লেখার আগের অবস্থায় ফিরে গেছে। এখন আরও আগের কমিটে ফিরে যেতে চায়। এখন আমরা git checkout d83f090 লিখে এন্টার করি। তাহলে এই কমিটে ফিরে আসবে। এখন আমরা লক্ষ্য করলে দেখবো যে chocolate.txt ফাইলটি আর নেই। অর্থাত তারও আগের অবস্থায় ফিরে গেছে।  এখন আমরা git checkout 'master' লিখে এন্টার প্রেস করি। তাহলে দেখবো আমরা মাস্টার ব্রান্সে চলে যাবো।  এখন আমরা chocolate.txt ফাইলটি খুলে some chocolate added লিখে সেভ করি। এরপর git status কমান্ড দিলে দেখবো modified: chocolate.txt আসবে। এখন git diff লিখে এন্টার করলে সর্বশেষ কমিটের পর যে পরিবর্তন তা দেখা যাবে। এখন আমরা git log --oneline লিখে এন্টার করি।  git log --oneline  2e10e4b (HEAD -> master) ice cube added  2c41b0e chocolate added  d83f090 2 txt file added  এখানে ৩টি কমিট দেখা যাচ্ছে। আমরা চাচ্ছি 2c41b0e chocolate added  কি পরিবর্তন হয়েছিল তা দেখতে ।এখন আমরা git show 2c41b0e  লিখে এন্টার করলে দেখতে পাবো  commit 2c41b0ef9426567c9e9f053de0118d7d4a9a376a  Author: IBRAHIM <1ibrahimtsc@gmail.com>  Date: Sat Mar 23 11:02:56 2019 +0600  chocolate added আসবে।  এখন chocolate.txt ফাইলটি খুলে তাতে dark chocolate added লিখে সেভ করি এবং স্টেংজিং করে কমিট করি।  এখন git log --oneline কমান্ড দিয়ে দেখি কি কি কমিট করেছি।  git log --oneline  e035835 (HEAD -> master) dark chocolate added  87524d4 some chocolate added  2e10e4b ice cube added  2c41b0e chocolate added  d83f090 2 txt file added  এখন e035835 (HEAD -> master) dark chocolate added ও  87524d4 some chocolate added এ ২টি কমিটের মধ্যে আমরা পার্থক্য জানতে চাচি্‌ছ।এখন git diff 87524d4 e035835 লিখে কমান্ড দিলে নিচের মত আসবে।  **diff --git a/chocolate.txt b/chocolate.txt**  **index 42f2c8e..088c6e2 100644**  **--- a/chocolate.txt আগের অবস্থা**  **+++ b/chocolate.txt পরের অবস্থা**  @@ -1 +1,3 @@  -some chocolate added  \ No newline at end of file  +some chocolate added আগেছিল  +  +dark chocolate added পরে যোগ হয়েছে।  এভাবে পার্থক্য দেখা যাবে।  এখন chocolate.txt ফাইলটি খুলে more chocolate added নতুন ১টি লাইন যুক্ত করে স্টেজিং করি।এখন যদি gif diff কমান্ড দিয়ে দেখতে চাই তাহলে তা দেখা যাবে না। স্টেজিং এরিয়াতে পার্থক্য দেখতে চাইলে git diff --staged এভাবে দেখতে হবে।  এখন কোন ফাইল মুছার জন্য git rm hot.txt দেই।  এখন git status কমান্ড দিলে আসবে  On branch master  Changes to be committed:  (use "git reset HEAD <file>..." to unstage)  modified: chocolate.txt  deleted: hot.txt এনাইনটি স্মৃতি থেকে মুছার জন্য  এভাবে কমান্ড দিতে হবে git reset HEAD hot.txt  এখন git status কমান্ড দিলে  On branch master  Changes to be committed:  (use "git reset HEAD <file>..." to unstage)  modified: chocolate.txt  Changes not staged for commit:  (use "git add/rm <file>..." to update what will be committed)  (use "git checkout -- <file>..." to discard changes in working directory)  deleted: hot.txt  LOBABA@LOBABA-PC MINGW64 /e/Git Tutorial for Beginner(গিট বাংলা টিউটোরিয়াল ) by fswebmaster/h drive/Git Tutorial for Beginner(গিট বাংলা টিউটোরিয়াল ) by frswebmaster/COFFEE (master) |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | Git by [kuthi devs](https://www.youtube.com/channel/UCydh5-RNzP9_MI2GTyEJXdw)  নতুন ব্রান্স তৈরী করার জন্য git branch branch\_name, কি কি ব্রান্স আছে চেক করার জন্য git branch, কোন ব্রান্সে যাওয়ার জন্য git checkout branch\_name, master branch এ থাকা অবস্থায় কোন ব্রান্স কে মাস্টার ব্রান্সের সাথে মার্জ করার জন্য git merge branch\_name, রিমোট সার্ভারের সাথে যুক্ত হওয়ার জন্য git remote add origin https://github/kuthidevs/test1.git, রিমোট সার্ভার এ কোন ব্রান্সকে পুশ করার জন্য git push origin branch\_name, সার্ভার হতে পুল করার জন্য git pull origin master, কোন ব্রান্স কে মুছার জন্য git branch -D branch\_name, git branch কমান্ড দিলে ব্রান্সের নামগুলো দেখাবে একটিভ ব্রান্সের আগে স্টার চিন্হ থাকবে। ভিসুয়াল স্টুডিও কোডে গিট নিয়ে কাজ করলে অনেক সহজ হয়। |  |  |

**Branch**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#branch)

Branches represent specific versions of a repository that “branch out” from your main project. Branches allow you to keep track of experimental changes you make to repositories and revert to older versions.

**Commit**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#commit)

A commit represents a specific point in your project’s history. Use the commit command in conjunction with the git add command to let git know which changes you wish to save to the local repository. Note that commits are not automatically sent to the remote server.

**Checkout**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#checkout)

Use the git checkout command to switch between branches. Just enter git checkout followed by the name of the branch you wish to move to, or enter git checkout master to return to the master branch. Mind your commits as you switch between branches.

**Fetch**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#fetch)

The git fetch command copies and downloads all of a branch’s files to your device. Use it to save the latest changes to your repositories. It’s possible to fetch multiple branches simultaneously.

**Fork**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#fork)

A fork is a copy of a repository. Take advantage of “forking” to experiment with changes without affecting your main project.

**Head**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#head)

The commit at the tip of a branch is called the head. It represents the most current commit of the repository you’re currently working in.

**Index**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#index)

Whenever you add, delete or alter a file, it remains in the index until you are ready to commit the changes. Think of it as the staging area for Git. Use the git status command to see the contents of your index. Changes highlighted in green are ready to be committed while those in red still need to be added to staging.

**Master**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#master)

The master is the primary branch of all your repositories. It should include the most recent changes and commits.

**Merge**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#merge)

Use the git merge command in conjunction with pull requests to add changes from one branch to another.

**Origin**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#origin)

The origin refers to the default version of a repository. Origin also serves as a system alias for communicating with the master branch. Use the command git push origin master to push local changes to the master branch.

**Pull**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#pull)

Pull requests represent suggestions for changes to the master branch. If you’re working with a team, you can create pull requests to tell the repository maintainer to review the changes and merge them upstream. The git pull command is used to add changes to the master branch.

**Push**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#push)

The git push command is used to update remote branches with the latest changes you’ve committed.

**Rebase**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#rebase)

The git rebase command lets you split, move or get rid of commits. It can also be used to combine two divergent branches.

**Remote**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#remote)

A remote is a clone of a branch. Remotes communicate upstream with their origin branch and other remotes within the repository.

**Repository**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#repository)

Git repositories hold all of your project’s files including branches, tags and commits.

**Stash**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#stash)

The git stash command removes changes from your index and “stashes” them away for later. It’s useful if you wish to pause what you’re doing and work on something else for a while. You can’t stash more than one set of changes at a time.

**Tags**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#tags)

Tags provide a way to keep track of important commits. Lightweight tags simply serve as pointers while annotated tags get stored as full objects.

**Upstream**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#upstream)

In the context of Git, upstream refers to where you push your changes, which is typically the master branch.

See the [Git docs reference](https://git-scm.com/docs) guide for more in depth explanations of Git related terminology.

**Commands for Configuring Git**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#commands-for-configuring-git)

Set the username:

git config –global user.name

Set the user email:

git config –global user.email

Create a Git command shortcut:

git config –global alias.

Set the preferred text editor:

git config –system core.editor

Open and edit the global configuration file in the text editor:

git config –global –edit

Enable command line highlighting:

git config –global color.ui auto

**Commands for Setting Up Git Repositories**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#commands-for-setting-up-git-repositories)

Create an empty repository in the project folder:

git init রিপোজিটরী তৈরী করার জন্য

Clone a repository from GitHub and add it to the project folder:

git clone (repo URL)

Clone a repository to a specific folder:

git clone (repo URL) (folder)

Display a list of remote repositories with URLs:

git remote -v

Remove a remote repository:

git remote rm (remote repo name)

Retrieve the most recent changes from origin but don’t merge:

git fetch

Retrieve the most recent changes from origin and merge:

git pull

**Commands for Managing File Changes**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#commands-for-managing-file-changes)

Add file changes to staging:

git add (file name)

Add all directory changes to staging:

git add .

Add new and modified files to staging:

git add -A

Remove a file and stop tracking it:

git rm (file\_name)

Untrack the current file:

git rm –cached (file\_name)

Recover a deleted file and prepare it for commit:

git checkout <deleted file name>

Display the status of modified files:

git status

Display a list of ignored files:

git ls-files –other –ignored –exclude-standard

Display all unstaged changes in the index and the current directory:

git diff

Display differences between files in staging and the most recent versions:

git diff –staged

Display changes in a file compared to the most recent commit:

git diff (file\_name)

**Commands for Declaring Git Commits**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#commands-for-declaring-git-commits)

Commit changes along with a custom message:

git commit -m "(message)" filename.extention

Commit and add all changes to staging:

git commit -am "(message)"

Switch to a commit in the current branch:

git checkout <commit> পূর্বের অবস্থায় ফেরৎ যাওয়ার জন্য

git checkout <commit> filename.extention পূর্বের অবস্থায় নির্ধারিত ফাইলকে ফেরৎ নেওয়ার জন্য

Show metadata and content changes of a commit:

git show <commit> কমিটে কি পরিবর্তন হয়েছে তা দেখার জন্য show এরপর কমিট নম্বর দিতে হবে 0812546

Discard all changes to a commit:

git reset –hard <commit>

Discard all local changes in the directory:

git reset –hard Head

Show the history of changes:

git log

Stash all modified files:

git stash

Retrieve stashed files:

git stash pop

Empty stash:

git stash drop

Define a tag:

git tag (tag\_name)

Push changes to origin:

git push

**Commands for Git Branching**[**#**](https://www.keycdn.com/blog/git-cheat-sheet#commands-for-git-branching)

Display a list of all branches:

git branch

Make a new branch and switch to it:

git checkout -b <branchname>

Switch to a branch:

git checkout <branchname>

Delete a branch:

git branch -d <branchname>

Merge a different branch with your active branch:

git merge <branchname>

Fetch a branch from the repository:

git fetch remote <branchname>

View merge conflicts between branches:

git diff <sourcebranch> <targetbranch>

Preview changes before merging branches:

git diff <sourcebranch> <targetbranch>

Push all local branches to a designated remote repository:

git push –all

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | git init | Create an empty repository in the project folder |  |
| 2 | git add (file name) | Add file changes to staging |  |
| 3 | git add . | Add all files & directory changes to staging |  |
| 4 | git add -A | Add new and modified files to staging |  |
| 5 | git rm (file\_name) | Remove a file and stop tracking it |  |
| 6 | git rm –cached (file\_name) | Untrack the current file: |  |
| 7 | git checkout <deleted file name> | Recover a deleted file and prepare it for commit |  |
| 8 | git status | Display the status of modified files |  |
| 9 | git ls-files –other –ignored –exclude-standard | Display a list of ignored files |  |
| 10 | git diff | Display all unstaged changes in the index and the current directory: |  |
| 11 | git diff –staged | Display differences between files in staging and the most recent versions: |  |
| 12 | git diff (file\_name) | Display changes in a file compared to the most recent commit |  |
| 13 | git config –global user.name "name" | Set the user name |  |
| 14 | git config –global user.email "email" | Set the user email |  |
| 15 | git config –global alias. | Create a Git command shortcut: |  |
| 16 | git config –system core.editor | Set the preferred text editor |  |
| 17 | git config –global –edit | Open and edit the global configuration file in the text editor |  |
| 18 | git config –global color.ui auto | Enable command line highlighting |  |
| 19 | git clone (repo URL) | Clone a repository from GitHub and add it to the project folder |  |
| 20 | git clone (repo URL) (folder) | Clone a repository to a specific folder |  |
| 21 | git remote -v | Display a list of remote repositories with URLs |  |
| 22 | git remote rm (remote repo name) | Remove a remote repository |  |
| 23 | git fetch | Retrieve the most recent changes from origin but don’t merge |  |
| 24 | git pull | Retrieve the most recent changes from origin and merge |  |
| 25 | git commit -m "(message)" filename.extention | Commit changes along with a custom message |  |
| 26 | git commit -am "(message)" | Commit and add all changes to staging |  |
| 27 | git checkout <commit> | Switch to a commit in the current branch |  |
| 28 | git checkout <commit> filename.extention | Switch to a commit in the current branch  পূর্বের অবস্থায় নির্ধারিত ফাইলকে ফেরৎ নেওয়ার জন্য |  |
| 29 | git show commitId | কমিটে কি পরিবর্তন হয়েছে তা দেখার জন্য show এরপর কমিট নম্বর দিতে হবে 0812546 |  |
| 30 | git reset –hard <commit> | Discard all changes to a commit |  |
| 31 | git reset –hard Head | Discard all local changes in the directory: |  |
| 32 | git log | Show the history of changes |  |
| 33 | git stash | Stash all modified files |  |
| 34 | git stash pop | Retrieve stashed files |  |
| 35 | git stash drop | Empty stash |  |
| 36 | git tag (tag\_name) | Define a tag |  |
| 37 | git push | Push changes to origin: |  |
| 38 | git branch | Display a list of all branches |  |
| 39 | git checkout -b <branchname> | Make a new branch and switch to it |  |
| 40 | git checkout <branchname> | Switch to a branch |  |
| 41 | git branch -d <branchname> | Delete a branch |  |
| 42 | git merge <branchname> | Merge a different branch with your active branch |  |
| 43 | git fetch remote <branchname> | Fetch a branch from the repository |  |
| 44 | git diff <sourcebranch> <targetbranch> | View merge conflicts between branches |  |
| 45 | git diff <sourcebranch> <targetbranch> | Preview changes before merging branches |  |
| 46 | git push –all | Push all local branches to a designated remote repository |  |