# Download

## Git

<https://git-scm.com/downloads>

## VS Code

Download: <https://code.visualstudio.com/>

## Git Desktop

Alternative to running the commands manually

<https://desktop.github.com/>

## TortoiseGit

Alternative to running the commands manually

https://tortoisegit.org/

# Commands

## Windows Commands

|  |  |  |
| --- | --- | --- |
| * cd: print current path * dir: list items * cls: clear command prompt * cd ..: change directory | * echo text > name(.type) : create file * mkdir foldername: create dir * del file: delete file | * rmdir folder: delete folder * copy file: copy file * move file: move file * move folder: move folder |

## Git Commands

|  |  |
| --- | --- |
| >git init  >git --version  >git status  >git log  >git log origin/feature..feature [view committed changes not yet pushed]  >git log --merge [log merge changes]  >git ls-files [viewing the stages files] [check online use]  >git ls-remote [showing remote files]  >git fetch origin  >git reflog [full reference log]  >git remote [displays remote servers – origin]  >git remote show origin  >git add .  >git add <file>  >git commit -m "added first text file"  >git commit -a -m "something" [adds/commits one step]  >git branch  >git branch -a [displays local and remote tracking branches]  >git branch -r [displays remote tracking branches only]  >git branch -vv [shows more info about local tracking branch]  >git branch <NewBranchName> [switch to branch]  >git branch -M main [renames the branch to main]  >git switch <branchName>  >git switch -c <NewBranchName> [create branch also]  >git checkout <BranchName>  >git checkout <commitid>  >git checkout -b <NewBranchName> [create branch also]  >git merge third-branch [Fast-Forward]  >git merge --squash feature [Fast-Forward. Will merge but not commit, do add and commit manually. Single commit]  >git merge --no-ff feature [Recursive merge, has additional commit. When undoing only reset the top merged commit]  >git merge –abort [abort the merge when conflict etc]  >git rebase master [switch to target and specify the source branch to rebase from. Rebase creates to Ids]  >git cherry-pick <ID> [switch to target and issue the command. It will create a copy with new ID]  >git stash [uncommitted and unstaged changes]  >git stash push -m "Third line added"  >git stash list [view list of all stashes]  >git stash apply [bring stash back, keeps in stash]  >git stash apply 2 [bring specific stash back, keeps in stash]  >git stash pop 0 [Removes it from stash, add commit]  >git stash drop 0 [Drop specific stash]  >git stash clear [Drop all stashes]  >git tag [Any tags already created]  >git tag 1.0 <CommitID> [Creates new tag]  >git tag -a 2.0 -m "message" [Creates new tag on latest commit]  >git show 1.0 [use the tag id, displays commit associated with the tag]  >git checkout 1.0 [puts in detached head, same as checkout with commitid]  >git tag -d 1.0 | If the email and name are not setup then add those  >git config user.email "someone@abc.com"  >git config user.name "Name"  or  >git config --global user.email "someone@abc.com"  >git config --global user.name "Name"  >git remote add origin <URL> [Tie local repo to remote repo]  >git push -u origin main [With local branch is available then will create the local tracking branch against the remote tracking branch]  >git push origin main  >git push --force origin main  >git pull origin master [Pull is combination of **git fetch** and **git merge** together]  >git fetch origin  >git branch --track <NAME> origin/<NAME> [git fetch, and then pull branch into new local tracking branch. In this case the local tracking branch gets created as well, [see this](#_Option2:_Create_a) also below switch with -t]  >git switch -t origin/<NAME> [Create the local tracking branch with same name]  >git branch -u upstream/<name> [ties the local tracking branch to remote tracking branch branch]  >git branch -u upstream/<name> <namelocal> [ties the local tracking branch to remote tracking branch branch when not in local branch]  >git clone <URL>  >git clone <URL> . [to clone in the same folder]  **Removing and Deleting –** [**Check This**](#_Deleting)  >git rm <file> [can use add as well] [DELETE]  Undoing Unstaged changes in tracked files  >git checkout <FileName>  >git restore <filename>  >git checkout . [Multiple files]  >git restore . [Multiple files]  Undoing new file which is not tracked  >git clean -dn [To list the files]  >git clean -df [To remove the files]  Undoing Staged Changes  >git reset <FileName>  >git checkout <FileName>  Or  >git restore --<FileName>  >git restore --staged .  >git checkout <FileName>  >git checkout .  Deleting Commits  ~1 is the nth number of commit, 2 means delete both 1 and 2  >git reset --soft HEAD~1 [SOFT Delete, keeps the file staged, then issue git restore command to remove the changes]  >git reset HEAD~1 [delete, will even unstaged the file if it was staged, then issue git clean commands as above]  >git reset --hard HEAD~1 [hard delete, removes all. May need to issue git clean commands as above]  >git revert <id>  Undoing deleted content that is committed  >git reflog  >git reset --hard 02f0176 [Id from reflog for the commit you want to bring back]  Deleting Branches  >git branch -D fourth-branch  >git branch -D fourth-branch third-branch  >git branch --delete --remotes origin/feature [delete remote tracking branch]  >git push origin --delete feature [delete remote branch on guthub]  Undoing Branches  >git reflog  >git checkout fd9d9ff [Checkout the commit, detached head]  >git switch -c feature |

# Git Info

## Git

* Version Control System
* Manage Code History
* Track Changes
* Local tool installed on the machine

## GitHub

* Companies build, ship and maintain software
* Cloud hosting and collaboration provider
* Git repository hosting provider

## How Git Works

Main branch is the master

A screenshot of a computer

Description automatically generated with medium confidence

## Working Dir vs Repository

Changes added to staging area and then committed.

Important to note that nothing pushed to GitHub yet.

A screenshot of a computer

Description automatically generated with medium confidence

## Branches & Commits

A screenshot of a computer

Description automatically generated with medium confidence

Branches help with people working on different features and then merged back to the running version.

## What is Head

A screenshot of a computer

Description automatically generated with low confidence

Click [here](#_Check_Head) for more details

## What is Detached Head

A picture containing screenshot, text, circle, diagram

Description automatically generated

Click [here](#_Check_Detached_Head) for more details

## Merge Types

A screenshot of a computer screen

Description automatically generated with low confidence

### Fast-Forward Merge

A picture containing text, screenshot, circle, font

Description automatically generated

### Recursive Merge

A diagram of a flowchart

Description automatically generated with low confidence

## Rebase

A screenshot of a computer

Description automatically generated with medium confidence

### When to do Rebase

A picture containing text, screenshot, font, design

Description automatically generated

## Merge vs Rebase vs Cherry-Pick

A screenshot of a cell phone

Description automatically generated with low confidence

Important: Cherry-Pick creates copies of the commit with a new ID

## Stash, RefLog, Merge, Rebase, Cherry-Pick Terminology Summary

A purple rectangles with white text

Description automatically generated with low confidence

## Connecting Local to Empty Remote Repo

A screenshot of a computer

Description automatically generated with low confidence

## Local Branches – Remote Tracking Branches

A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer program

Description automatically generated with low confidence

Pull is combination of fetch and merge.

## Clone vs Fork/PullRequests

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated with medium confidence | A picture containing text, screenshot, font  Description automatically generated |

# Git – Project 1 – 2023 Git Basics

## Creating Project

* Create a folder with name 2023 Git Basics
* Open the folder in VS Code
* Create a file initial-commit.txt

## Initialize Git Repository

* Open the path in the command prompt
* Issue command git init to initialize the git repository

>git init

Initialized empty Git repository in C:/Users/XXX/Desktop/2023 Git/2023 Git Basics/.git/

A picture containing text, font, screenshot, logo

Description automatically generated

## Change 1

### Check Status

>git status

We have the change but nothing added to the staging area

A screen shot of a computer

Description automatically generated with low confidence

### Stage the Change

> git add . [for all the files]

> git add <file> [for the single file]

And then do git status again

>git status

A picture containing text, screenshot, font

Description automatically generated

### Commit the Change

>git commit -m "added first text file"

The first time may get into an issue as following

A screen shot of a computer

Description automatically generated with medium confidence

#### Setup the User and Name

>git config user.email "[someone@abc.com](mailto:someone@abc.com)"

>git config user.name "Name"

#### Issue Commit Again

>git commit -m "added first text file"

A picture containing text, font, screenshot, line

Description automatically generated

### View Commits

>git log

A picture containing text, screenshot, font, line

Description automatically generated

## Change 2

Create a new file “second-commit.txt” and save it.

>git status

A black screen with white text

Description automatically generated with low confidence

>git add .

To add the file

>git status

A screen shot of a computer screen

Description automatically generated with low confidence

>git commit -m "added second text file"

A picture containing text, font, screenshot, number

Description automatically generated

>git status



>git log

Will see now two commits

A picture containing text, screenshot, font

Description automatically generated

## Change 3 – New Branch

Make sure that you are in the branch off which new branch is needed

>git branch



>git branch second-branch

To create a new branch

>git branch



>git checkout second-branch

Switch to second branch

>git checkout -b third-branch

Create branch and checkout in the same step

>git branch

A picture containing text, font, graphics, screenshot

Description automatically generated

>git status



## Change 4 – Adding File to third-branch

Switch to third branch

>git checkout third-branch

And then stage and commit the change

>git add .

>git commit -m "file added to third-branch"

A black background with white text

Description automatically generated with low confidence

>git status



>git log

A screenshot of a computer program

Description automatically generated with low confidence

## Merge third-branch to master

Switch to source branch

>git checkout master

>git merge third-branch

Updating 3a9d0dc..37fdfbb

Fast-forward

working-with-branches.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 working-with-branches.txt

## Check Head

>git log

A picture containing text, screenshot, software, font

Description automatically generated

## Check Detached Head

>git branch

A picture containing text, font, screenshot, graphics

Description automatically generated

>git checkout second-branch

A picture containing text, screenshot, font, software

Description automatically generated

>git checkout 3a9d0dcda284a113878a18ae977e71cfde9dca1f

A screenshot of a computer program

Description automatically generated with medium confidence

>git branch

We are not in any branch at this time

A picture containing text, font, screenshot, graphics

Description automatically generated

To get out of the detach head

>git checkout second-branch

>git branch

A picture containing text, font, graphics, screenshot

Description automatically generated

Check this for [committing](#_Committing_Detached_Head) the changes for detached head.

## Listing files in the staging area

>git ls-files

A picture containing text, font, screenshot, typography

Description automatically generated

## Deleting

### Deleting Working Dir Files

Switch to master using either of the following commands

>switch master

>switch checkout master

Hard delete the file from the working dir “**working-with-branches.txt**”

>git ls-files

The file still shows here

A picture containing text, font, screenshot, typography

Description automatically generated

>git status

A screen shot of a computer

Description automatically generated with medium confidence

Then use one of the following to stage the change. I used “rm” command

>git add .

>git add working-with-branches.txt

>git rm working-with-branches.txt [used this here]



>git ls-files



>git status

A screen shot of a computer

Description automatically generated with low confidence

>git commit -m "deleted file from working dir"

A picture containing text, font, screenshot

Description automatically generated

### Undoing Unstaged Changes in Tracked File

#### Single File

|  |  |
| --- | --- |
| Go to master branch  Open initial-commit.txt  Add some text to it and save | A screenshot of a computer  Description automatically generated with medium confidence |

>git status

A screen shot of a computer

Description automatically generated with medium confidence

>git checkout initial-commit.txt **or**

>git restore initial-commit.txt

Here we are doing a checkout to the most latest commit that we have



#### Multiple Files

Now make changes to both the files in Master branch and do status check

>git status

A screen shot of a computer

Description automatically generated with medium confidence

>git checkout . **or**

>git restore .



#### Undoing new file created

Create a new file “test.txt”

>git status

A black screen with white text

Description automatically generated with low confidence

In this case always first check and then then remove the file

>git clean -dn



>git clean -df



### Undoing Staged Changes

Make changes to a file

And then add

>git add initial-commit.txt

>git status

A screen shot of a computer code

Description automatically generated with low confidence

|  |  |
| --- | --- |
| Old Way | New Way |
| >git reset initial-commit.txt  Unstaged changes after reset:  M initial-commit.txt  >git checkout initial-commit.txt  Updated 1 path from the index | >git restore --staged initial-commit.txt  >git checkout initial-commit.txt  Updated 1 path from the index |

### Deleting Commits With git reset

#### Difference between revert and reset

Revert: If you want to undo changes that happened in a specific commit then issue revert command

Reset: To undo every change that happened since a given commit occurred, use reset command

#### Create and Commit File

Create new file, stage it and then commit it as well.

>git status

A black background with white text

Description automatically generated with low confidence

>git add .

>git commit -m "test file"

A picture containing text, font, screenshot, typography

Description automatically generated

>git log

A picture containing text, screenshot, font

Description automatically generated

#### Way 1 – Soft Reset – Commit

Keeps the file staged if staged and committed

>git reset --soft HEAD~1

Resetting one commit so 1

>git log [ the commit will not be there any more]

>git ls-files [the file will still show as we only removed the last commit]

A picture containing text, font, screenshot, graphics

Description automatically generated

Either commit the changes again

>git commit -m "added file again"

A picture containing text, font, screenshot, typography

Description automatically generated

Or follow to [remove the stagged](#_Undoing_Staged_Changes) file above.

#### Way 2 – Reset – Commit

Unstages the file as well if staged and committed

>git reset HEAD~1

>git log [ the commit is gone]

>git status

A black background with white text

Description automatically generated with low confidence



>git clean -dn



>git clean -df



#### Way 3 – Hard Reset – Commit

>git reset --hard HEAD~1

>git clean -dn

>git clean -df

#### Way 4 – Git Revert – Commit

Change any file and commit the changes

>git log

A screen shot of a computer

Description automatically generated with low confidence

>git reflog

A screenshot of a computer program

Description automatically generated with medium confidence

>git revert 0571a7d **or**

>git revert 966058d15deb024a9540a5e403117b360046cab2

It will open the note pad, put the commit message and close it.



### Deleting Branches

Pass the flag as

* -d only will delete the merged branches
* -D will delete any branch

>git branch -D fourth-branch

>git branch -D fourth-branch third-branch

### Summary Deleting

A screenshot of a computer program

Description automatically generated with medium confidence

## Committing Detached Head

Create a new file dummy.txt, add it and commit it.

>git add .

>git commit -m "dummy added"

Now do a git log to look at master

>git log

You’ll see multiple commits with latest to oldest. Pick the second last commit and checkout

|  |  |
| --- | --- |
| >git checkout 405b0d121bb0204c75a807ac60890435ec8a1171  A screenshot of a computer screen  Description automatically generated with medium confidence | >git branch |

Do two actions here

1. Now add a line to “second-commit.txt”

A screenshot of a computer

Description automatically generated with medium confidence

1. Also add a new file in this mode “detached.txt”

>git status

A screen shot of a computer

Description automatically generated with medium confidence

>git add .

>git status

A screen shot of a computer

Description automatically generated with medium confidence

>git commit -m "changes made in detached mode"

A black background with white text

Description automatically generated with low confidence

|  |  |
| --- | --- |
| >git switch master  A warning will display with a solution as well. Take note of the id provided.  A screenshot of a computer program  Description automatically generated with medium confidence  >git branch detached-head 26d1c15  Create a new branch with the commit id.  >git branch    >git switch detached-head  Switched to branch 'detached-head' | To avoid the warning.  Create the branch first here  >git branch detached-head  This will push the change to new branch  And then switch to master  >git switch master |

>git switch master

>git merge detached-head

A screenshot of a computer program

Description automatically generated with medium confidence

# Git – Project 2 – 2023 Git Deep Dive

## Basics

|  |  |
| --- | --- |
| Initialize repository  >git init  And then add a file file1.txt | >git add .  >git commit -m "file1 added"  A screen shot of a computer  Description automatically generated with low confidence  >git log  A picture containing text, screenshot, font  Description automatically generated |

## Stash Change

Un committed and un staged changes can be stashed.

### Stash

So make a change in file1.txt and save it

>git stash or

>git stash push -m "Third line added"



You’ll be back to the previous commit at this point. Check with git log

### View Stash List

>git stash list

A screen shot of a computer

Description automatically generated with low confidence

### Bring Stash Back

>git stash apply

A black background with white text

Description automatically generated with low confidence

### Bring Back specific Stash

Latest is index 0

>git stash apply 2

### Remove from Stash and add to project

>git stash pop 0

A screen shot of a computer

Description automatically generated with medium confidence

>git add .

>git commit -m "added stashed changes"

### Dropping Stash

>git stash drop 0

Dropped refs/stash@{0} (509cad802d3b364c92e6c9d8566040601db3f2bc)

Or to drop all stashes

>git stash clear

## Bringing back deleted Changes with REFLOG

### Create File2.txt

Create a new file2.txt, add and commit it

### Delete the commit for File2.txt

>git reset --hard HEAD~1 or

>git revert <id>

### Check the Delete with REFLOG

>git reflog

A screen shot of a computer program

Description automatically generated with low confidence

### Reset to Head1 to bring back file

>git reset --hard 02f0176



## Bringing back deleted branches with REFLOG

### Create a new branch and switch to it

>git switch -c feature

### Add a file to it

Add File3.txt and commit it

### Switch to Master and Delete Feature

>git switch master

>git branch -D feature



>git branch



### Undoing Deleted Branch

>git reflog

A screen shot of a computer

Description automatically generated with medium confidence

>git checkout fd9d9ff

It will put us in detached head situation

>git branch



>git switch -c feature

New branch feature is created with all the commits upto selected commit

At this point can do merge to master etc.

# Git – Project3 – 2023 Git Branches

## Basics

Initialize repository

>git init

## Initial Commits “master” Branch

### Commit 1

|  |  |
| --- | --- |
| Create a folder “master”  Add a file in folder master “m1.txt” | >git add .  >git commit -m "m1" |

### Commit 2

|  |  |
| --- | --- |
| Add a file in folder master “m2.txt” | >git add .  >git commit -m "m2" |

### Initial State of Master Branch

>git log

A picture containing text, screenshot, font

Description automatically generated

## “feature” Branch

### Create “feature” Branch

>git switch -c feature



>git branch



### “feature” Commits

#### Commit 1

|  |  |
| --- | --- |
| Create a folder “feature”  Add a file in folder master “f1.txt” | >git add .  >git commit -m "f1" |

#### Commit 2

|  |  |
| --- | --- |
| Add a file in folder master “f2.txt” | >git add .  >git commit -m "f2" |

### State of “feature” Branch

A picture containing text, screenshot, software

Description automatically generated

## Merge “feature” to “master” – Fast-Forward

### Option 1

>git switch master

>git merge feature

* Head of the master branch has been fast-forward to feature branch with the merge. Check below log
* All “feature” commits are available in master now

A screen shot of a computer

Description automatically generated with medium confidence

>git log

A picture containing text, screenshot, software, multimedia software

Description automatically generated

#### Undo commits from “master” due to merge

>git reset --hard HEAD~2



### Option 2

>git merge --squash feature

* Combines all commits of feature into single commit in master
* Nothing gets committed

A screen shot of a computer

Description automatically generated with medium confidence

>git status

A screen shot of a computer code

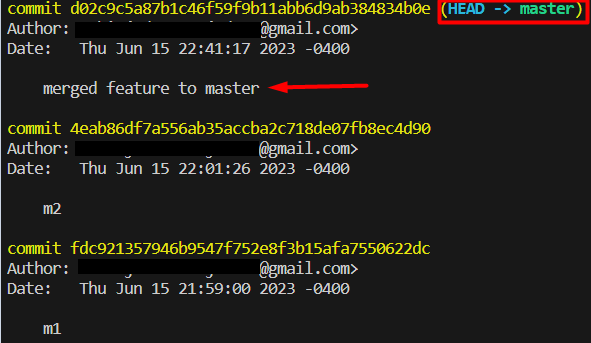
Description automatically generated with low confidence

>git commit -a -m "merged feature to master"

A picture containing text, font, screenshot, typography

Description automatically generated

>git log



#### Undo commits from “master” due to merge

>git reset --hard HEAD~1



## Merge “feature” to “master” – Recursive

### Basic Recursive merge

>git merge --no-ff feature

A screen shot of a computer screen

Description automatically generated with low confidence

>git log

A screenshot of a computer program

Description automatically generated with medium confidence

#### Undo commits from “master” due to merge

>git reset --hard HEAD~1



### Advance Recursive merge

#### Master branch

Create m3.txt file, add and commit

#### Merge feature to master

>git merge feature

A recursive merge is applied this time

A screen shot of a computer

Description automatically generated with medium confidence

>git log

A screenshot of a computer

Description automatically generated

#### Reset the merge

>git reset --hard HEAD~1

### Advance Recursive merge with --squash

>git merge --squash feature

A picture containing text, screenshot, font, line

Description automatically generated

>git status

A screen shot of a computer code

Description automatically generated with low confidence

>git commit -a -m "master and feature merged"

A picture containing text, font, screenshot, typography

Description automatically generated

>git log

A screen shot of a computer

Description automatically generated with medium confidence

>git reset --hard HEAD~1

Reset again so to use the same for Rebase

## Rebase

Switch to feature branch

>git branch

Now look at the git log to keep track of the ids for all the commit to feature branch

>git log

A screenshot of a computer

Description automatically generated with medium confidence

>git rebase master

* Rebasing created new commits
* M3 has been brought over to feature



>git log

A screenshot of a computer

Description automatically generated with medium confidence

Finally merge feature into master

>git switch master

>git merge feature

A screen shot of a computer

Description automatically generated with medium confidence

>git log

A screenshot of a computer

Description automatically generated with medium confidence

## Handling Merge Conflicts

### Change in Master

>git switch master

Open feature/f1.txt

Add text “This is the change made in master branch” and commit it.

### Change in feature

>git switch feature

Open feature/f1.txt

Add text “I changed this in feature branch” and commit it.

### Merge

>git switch master

>git merge feature

A picture containing text, screenshot, font, line

Description automatically generated

A screen shot of a computer

Description automatically generated with medium confidence

>git status

Git status will give more details about the conflict as well

A screenshot of a computer screen

Description automatically generated with low confidence

>git log –merge

A picture containing text, screenshot, font, software

Description automatically generated

#### Accept Current Change

Click accept current change meaning keeping master content then commit it

>git commit -a -m "merged master and feature for f1 file"

## Cherry-Pick

### Master branch

Switch to master branch

#### Change master/m1.txt

Open master/m1.txt and add a text to it, as you see there is typo “impotent”

A screen shot of a computer

Description automatically generated with medium confidence

And add and commit

>git commit -a -m "added text to m1"

### Create a feature2 branch and switch to it

>git switch -c feature2



#### Create folder and file in feature2

Create folder feature2

Create file in feature2 as f2-1.txt

Add and commit

>git add .

>git commit -m "f2-1.txt"

#### Fix typo in folder master, file m1.txt

A picture containing text, font, screenshot, line

Description automatically generated

Commit the change

>git commit -a -m "Fixed typo in m1.txt"

#### Create a new file in feature2 folder

Create a new file f2-2.txt

>git add .

>git commit -m "f2-2.txt"

### Chery-Pick only TypoFix Commit and move to master

#### Recent commits in Feature2 branch

A screenshot of a computer program

Description automatically generated with medium confidence

Pick the id of the typo fix

45d451e318a12bf278a4f707300f866b8472b545

#### What Cherry-Pick will do

It will create copy of this commit with a new ID

#### Switch to Master branch

>git switch master

#### Cherry-Pick TypoFix from feature2 branch

>git cherry-pick 45d451e318a12bf278a4f707300f866b8472b545

A picture containing text, font, screenshot, typography

Description automatically generated

Typo is fixed now

A screen shot of a computer

Description automatically generated with medium confidence

>git log

Fixed typo has been brought over with a new ID

A screenshot of a computer

Description automatically generated with medium confidence

# Git – Project 4 – 2023 Git Tagging Commits

## Basics

Tags are important stages in our commit history

Initialize repository

>git init

## Adding 3 Commits

|  |  |  |
| --- | --- | --- |
| Add file1.txt and add/commit it  >git add .  >git commit -m "file1.txt" | Add file2.txt and add/commit it  >git add .  >git commit -m "file2.txt" | Add file3.txt and add/commit it  >git add .  >git commit -m "file3.txt" |

## Check current Tags

>git tag

## Adding Tag to second commit

>git log

A screenshot of a computer

Description automatically generated with medium confidence

>git tag 1.0 43245249e11f6ac3b0dc7f0d4da34c9397a286b1

>git tag



## Checking the commits associated with tag

>git show 1.0

A screen shot of a computer

Description automatically generated with medium confidence

## Checkout the commit with Tag Id

This will put us in the detached head mode (check these [Check Detached Head](#_Check_Detached_Head), [Committing Detached Head](#_Committing_Detached_Head), [Undoing Deleted Branch](#_Undoing_Deleted_Branch)).

>git checkout 1.0

Same as checkout with the commit id

To get out of detached head use either of the following

>git switch -

>git switch <branch>

## Deleting tag

>git tag -d 1.0



## Adding Tag to latest commit

>git tag -a 2.0 -m "This is the latest version"

>git tag



>git show 2.0

A picture containing text, screenshot, software, font

Description automatically generated

>git tag -d 2.0

# Working with GitHub - 2023-git-basics

## Setting up remote repository

Login or create account and then create a repository with name “2023-git-basics”.

On **code tab** you’ll be presented with multiple options for the setup

A screenshot of a computer

Description automatically generated with medium confidence

## Creating a local repository

Create a new folder “2023-git-basics” on your machine and open it in VS Code

Add a folder and a file

A screenshot of a computer

Description automatically generated with medium confidence

>git init

>git add .

>git commit -m "m1"

>git status



>git branch



## Setup and push Local Repository to Remote Repository

As specified above will be using third option so copy the url which will be like

<https://github.com/USERNAME/2023-git-basics.git>

>git remote add origin <https://github.com/USERNAME/2023-git-basics.git>

>git branch -M main [rename the master branch to main]

>git branch



>git push -u origin main

A screen shot of a computer program

Description automatically generated with low confidence

Go to GitHub and the commit will be there.

### Login Prompt

If prompted with login then setup via access token

* Go to GitHub
* Click the account icon
* Click setting from the pop up
* Scroll down on the resulting page
* Go to Developer Settings to generate a new token

A screenshot of a computer

Description automatically generated with low confidence

* Click Generate a personal access token
* Provide a note, expiration date, select scopes
* And finally click generate token
* Copy the token
* Use the token to signin

A picture containing text, screenshot, font, design

Description automatically generated

Note for MAC OS users

A screenshot of a computer program

Description automatically generated with low confidence

## Adding a Second Commit

Create a new file in master/m2.txt

Add, commit and push it.

>git push -u origin main

## View Local Branches along with Remote Tracking Branch as well

>git branch -a



>git branch -r



Any push or pull involves remote tracking branch

## Create new Local Branch “feature”

>git switch -c feature

|  |  |  |
| --- | --- | --- |
| >git branch | >git branch -r | >git branch -a  A picture containing text, font, screenshot, number  Description automatically generated |

Add a new folder “feature”, a file in it “f1.txt”

Add, commit and push the changes

>git add .

>git commit -m "f1"

>git push -u origin feature

A screen shot of a computer

Description automatically generated with medium confidence

>git branch -r



>git branch -a

A black screen with red and green text

Description automatically generated with low confidence

Both branches showing on GitHub as well

A screenshot of a computer

Description automatically generated with medium confidence

## Create new Remote Branch “feature2”

### Creating branch and content on GitHub

Create a branch feature2 on GitHub

A screenshot of a computer

Description automatically generated with medium confidence

Feature2 will show now

A screenshot of a computer

Description automatically generated with medium confidence

Create a new file and commit it

A picture containing text, font, screenshot, line

Description automatically generated

### Pulling New Branch feature2 and Content

>git ls-remote

A screenshot of a computer screen

Description automatically generated with low confidence

>git fetch origin

>git branch -a

A picture containing text, font, screenshot, design

Description automatically generated

#### Option1: Create a new Local Tracking Branch to Link with Remote Branch [**Avoid**]

>git branch --track feature2-remote origin/feature2 [Bad the two names are different]



>git branch -a

A screen shot of a computer

Description automatically generated with low confidence

Now when you switch to “feature2-remote” and make changes you need to use “**git push**”.

A screenshot of a computer program

Description automatically generated with low confidence

So delete the branch

>git branch -d feature2-remote or

>git branch -D feature2-remote

#### Option2: Create a new Local Tracking Branch to Link with Remote Branch [**Good**]

>git branch --track feature2 origin/feature2 [Note the name is the same]



Or

>git switch -t origin/feature2 [Note read feature3 as feature2 in below image]



>git branch -a

A screen shot of a computer

Description automatically generated with low confidence

>git switch feature2

Now add a file “t1-remote.txt”, add and commit it

>git add .

>git commit -m "t1-remote.txt"

>git push or

>git push origin or

>git push origin feature2

>git branch -vv

A picture containing text, font, screenshot, number

Description automatically generated

>git branch -a

A screen shot of a computer

Description automatically generated with low confidence

## Deleting Remote Branches and Public Commits

>git fetch origin

|  |  |
| --- | --- |
| >git branch -a  A screen shot of a computer  Description automatically generated with medium confidence | >git branch -vv  A picture containing text, font, screenshot, number  Description automatically generated |

### Delete Branch, then remote tracking branch, finally remote branch

|  |  |
| --- | --- |
| >git branch -D feature | >git branch -a |
| >git branch --delete --remotes origin/feature | >git branch -a |

>git ls-remote

A screenshot of a computer program

Description automatically generated with low confidence

Still available on git hub

>git push origin --delete feature



>git ls-remote

A screenshot of a computer code

Description automatically generated with medium confidence

### Delete Remote Branch First

|  |  |
| --- | --- |
| >git branch -a | >git branch -vv |

>git push origin --delete feature2

A picture containing text, font, screenshot, line

Description automatically generated

>git ls-remote

A screenshot of a computer screen

Description automatically generated with low confidence

Deletes the remote tracking branch as well

The branch remains as that has not be delete

|  |  |  |
| --- | --- | --- |
| >git branch -a | >git branch -r | >git branch -vv |

### Removing the Public Commits

>git reset --hard HEAD~1

>git push --force origin master

## Go to GitHub and Delete Repository

Delete the repository completely from GitHub since this is not needed any more.