Hands-On Git : Commands, Workflows & Real-Time Tasks

What is Git?

Git is a distributed version control system used for tracking changes in source code during software development. It enables multiple developers to work on a project simultaneously, provides powerful branching and merging features, and ensures full history and traceability of every change.

- Git is installed on the local machine and manages code locally.
- **GitHub** (or GitLab, Bitbucket) is a cloud-based hosting service for Git repositories used for remote collaboration.

Why Git?

- **Track Changes:** Every change to the code is recorded, so you can go back to any previous version.
- **Team Collaboration:** Developers can work independently using branches and then merge their work.
- **Backup:** Your code can be pushed to a remote repository (like GitHub), ensuring it is not lost.
- **Experiment Safely:** Branching lets you try new features or fixes without affecting the main project.
- **Conflict Resolution:** Git offers tools to handle merge conflicts and ensure consistency.

How Git Works?

Git works through three main areas:

1. **Working Directory (Workspace):**
   Where you make changes to your files. These are untracked or modified files.
2. **Staging Area (Index):**
   A place to group changes you want to include in your next commit using git add.
3. **Local Repository:**
   Where your changes are permanently recorded after using git commit. This is your local history.

After committing locally, you can use git push to send changes to a **Remote Repository** (e.g., GitHub).

Git Installation

Command:

yum install -y git

```
[root@ip-172-31-45-141 ~]# yum install -y git
Amazon Linux 2023 Kernel Livepatch repository                              176 kB/s |  19 kB     00:00
Dependencies resolved.
========================================================================================================
 Package                  Architecture     Version                        Repository              Size
========================================================================================================
Installing:
 git                      x86_64           2.50.1-1.amzn2023.0.1          amazonlinux             53 k
Installing dependencies:
 git-core                 x86_64           2.50.1-1.amzn2023.0.1          amazonlinux            4.9 M
 git-core-doc             noarch           2.50.1-1.amzn2023.0.1          amazonlinux            2.8 M
 perl-Error               noarch           1:0.17029-5.amzn2023.0.2       amazonlinux             41 k
 perl-File-Find           noarch           1.37-477.amzn2023.0.7          amazonlinux             25 k
 perl-Git                 noarch           2.50.1-1.amzn2023.0.1          amazonlinux             41 k
 perl-TermReadKey         x86_64           2.38-9.amzn2023.0.2            amazonlinux             36 k
 perl-lib                 x86_64           0.65-477.amzn2023.0.7          amazonlinux             15 k

Transaction Summary
========================================================================================================
Install  8 Packages

Total download size: 7.9 M
Installed size: 41 M
Downloading Packages:
(1/8): git-2.50.1-1.amzn2023.0.1.x86_64.rpm                                614 kB/s |  53 kB     00:00
(2/8): perl-Error-0.17029-5.amzn2023.0.2.noarch.rpm                        2.0 MB/s |  41 kB     00:00
(3/8): git-core-doc-2.50.1-1.amzn2023.0.1.noarch.rpm                        22 MB/s | 2.8 MB     00:00
(4/8): perl-File-Find-1.37-477.amzn2023.0.7.noarch.rpm                     1.2 MB/s |  25 kB     00:00
(5/8): git-core-2.50.1-1.amzn2023.0.1.x86_64.rpm                            30 MB/s | 4.9 MB     00:00
(6/8): perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64.rpm                     955 kB/s |  36 kB     00:00
(7/8): perl-lib-0.65-477.amzn2023.0.7.x86_64.rpm                           668 kB/s |  15 kB     00:00
(8/8): perl-Git-2.50.1-1.amzn2023.0.1.noarch.rpm                           478 kB/s |  41 kB     00:00
--------------------------------------------------------------------------------------------------------
Total                                                                       33 MB/s | 7.9 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                               1/1
  Installing       : git-core-2.50.1-1.amzn2023.0.1.x86_64                                         1/8
  Installing       : git-core-doc-2.50.1-1.amzn2023.0.1.noarch                                     2/8
  Installing       : perl-lib-0.65-477.amzn2023.0.7.x86_64                                         3/8
  Installing       : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64                                   4/8
  Installing       : perl-File-Find-1.37-477.amzn2023.0.7.noarch                                   5/8
  Installing       : perl-Error-1:0.17029-5.amzn2023.0.2.noarch                                    6/8
  Installing       : perl-Git-2.50.1-1.amzn2023.0.1.noarch                                         7/8
```

Screenshot 1: Installing Git package

Repository Initialization

mkdir /dir1

cd /dir1

git init

```
[root@ip-172-31-45-141 ~]# mkdir /dir1
[root@ip-172-31-45-141 ~]# cd /dir1
[root@ip-172-31-45-141 dir1]# git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
hint:
hint: Disable this message with "git config set advice.defaultBranchName false"
Initialized empty Git repository in /dir1/.git/
```

Screenshot 2: Directory creation and git init

Git Structure Post Initialization:

- **Workspace**: Where untracked/modified files reside
- **Staging Area**: Files added using git add
- **Local Repository**: Files committed using git commit

Tracking Files

touch file file2 file3 file4 file5

git status

```
[root@ip-172-31-45-141 dir1]# touch file1 file2 file3 file4 file5
[root@ip-172-31-45-141 dir1]# git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1
        file2
        file3
        file4
        file5

nothing added to commit but untracked files present (use "git add" to track)
[root@ip-172-31-45-141 dir1]#
```

Screenshot 3: Files in red (workspace)

git add file1

git status

```
[root@ip-172-31-45-141 dir1]# git add file1
[root@ip-172-31-45-141 dir1]# git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file2
        file3
        file4
        file5
```

Screenshot 4: file1 in green (staging area)

git rm --cached file1 # remove single file from staging

git rm --cached * # remove all from staging

git add * # add all to staging

git commit -m "Initial commit"

```
[root@ip-172-31-45-141 dir1]# git add *
[root@ip-172-31-45-141 dir1]# git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1
        new file:   file2
        new file:   file3
        new file:   file4
        new file:   file5
```

Screenshot 5: Status before commit in green (staging area)

```
[root@ip-172-31-45-141 dir1]# git rm --cached file1
rm 'file1'
[root@ip-172-31-45-141 dir1]# git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file2
        new file:   file3
        new file:   file4
        new file:   file5

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1
```

Screenshot 6: Remove single file from staging

```
[root@ip-172-31-45-141 dir1]# git rm --cached *
rm 'file1'
rm 'file2'
rm 'file3'
rm 'file4'
rm 'file5'
[root@ip-172-31-45-141 dir1]# git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1
        file2
        file3
        file4
        file5
```

Screenshot 7: Remove All files from staging

```
[root@ip-172-31-45-141 dir1]# git commit -m "Initial commit"
[master (root-commit) 415e580] Initial commit
 Committer: root <root@ip-172-31-45-141.eu-west-3.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 5 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1
 create mode 100644 file2
 create mode 100644 file3
 create mode 100644 file4
 create mode 100644 file5
[root@ip-172-31-45-141 dir1]# git status
On branch master
nothing to commit, working tree clean
```

Screenshot 8: Status after commit

Cleaning Untracked Files

git clean -fd

```
[root@ip-172-31-45-141 dir1]# touch file6
[root@ip-172-31-45-141 dir1]# git clean -fd
Removing file6
[root@ip-172-31-45-141 dir1]# git status
On branch master
nothing to commit, working tree clean
```

Screenshot 9: Cleaning workspace

Git Log and Commit History

git log

git log --oneline

git log --oneline | head -2

git log filename

git commit --amend -m "Updated commit message"

```
[root@ip-172-31-45-141 dir1]# git log file5
commit 415e580cc18c683aa0aabf12c4c4547933bffd2c (HEAD -> master)
Author: root <root@ip-172-31-45-141.eu-west-3.compute.internal>
Date:   Fri Aug 1 06:42:24 2025 +0000

    Initial commit
[root@ip-172-31-45-141 dir1]# git commit --amend -m "Updated commit message"
[master 06cf039] Updated commit message
 Date: Fri Aug 1 06:42:24 2025 +0000
 Committer: root <root@ip-172-31-45-141.eu-west-3.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 5 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1
 create mode 100644 file2
 create mode 100644 file3
 create mode 100644 file4
 create mode 100644 file5
```

Screenshot 10: Log and amend command output

Git Identity Configuration

git config --global user.name "jayshri"

git config --global user.email "jayshri567@gmail.com"

Edit config

git config --global --edit

```
[root@ip-172-31-45-141 dir1]# git config --global user.name "jayshri"
[root@ip-172-31-45-141 dir1]# git config --global user.email "jayshri567@gmail.com"
```

Screenshot 11: Git global config setup

Connecting to Remote Repository

git remote add origin <repo_url>

git push origin master # Use GitHub token when prompted

```
[root@ip-172-31-45-141 dir1]# git push origin master
Username for 'https://github.com': jayshrilandge30
Password for 'https://jayshrilandge30@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 250 bytes | 250.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/jayshrilandge30/devops-auto-deploy-website.git
 * [new branch]      master -> master
[root@ip-172-31-45-141 dir1]#
```

Screenshot 12: Pushing to GitHub with token authentication

Cloning and Pulling

git clone <repo_url> # For copying repo first time

git pull <repo_url> # For syncing latest changes

```
[root@ip-172-31-45-141 dir1]# gir clone https://github.com/jayshrilandge30/UnitConverter.git
-bash: gir: command not found
[root@ip-172-31-45-141 dir1]# git clone https://github.com/jayshrilandge30/UnitConverter.git
Cloning into 'UnitConverter'...
remote: Enumerating objects: 72, done.
remote: Counting objects: 100% (72/72), done.
remote: Compressing objects: 100% (69/69), done.
remote: Total 72 (delta 18), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (72/72), 65.63 KiB | 800.00 KiB/s, done.
Resolving deltas: 100% (18/18), done.
[root@ip-172-31-45-141 dir1]# git pull https://github.com/jayshrilandge30/UnitConverter.git
remote: Enumerating objects: 72, done.
remote: Counting objects: 100% (72/72), done.
remote: Compressing objects: 100% (69/69), done.
remote: Total 72 (delta 18), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (72/72), 65.61 KiB | 2.34 MiB/s, done.
From https://github.com/jayshrilandge30/UnitConverter
 * branch            HEAD       -> FETCH_HEAD
hint: You have divergent branches and need to specify how to reconcile them.
hint: You can do so by running one of the following commands sometime before
hint: your next pull:
hint:
hint:   git config pull.rebase false  # merge
hint:   git config pull.rebase true   # rebase
hint:   git config pull.ff only       # fast-forward only
hint:
hint: You can replace "git config" with "git config --global" to set a default
hint: preference for all repositories. You can also pass --rebase, --no-rebase,
hint: or --ff-only on the command line to override the configured default per
hint: invocation.
fatal: Need to specify how to reconcile divergent branches.
```

Screenshot 13: Clone and pull commands

Differences:

- **Clone** = First-time copy
- **Pull** = Sync changes

Git Push vs Git Pull vs Git Clone

Command	Action
git push	Sends commits from local to remote repo
git pull	Gets updates from remote to local repo
git clone	Makes a local copy of the entire repository

Git Tags

git tag # View tags

git tag imp-tag <commit_id> # Lightweight tag

git tag -d imp-tag # Delete tag

git show imp-tag1 # View tag details

git tag imp-tag3 <commit_id> -m "it is very imp tag"

```
[root@ip-172-31-32-25 dir1]# git tag imp-tag3 ea2f16fc81695348557120683b4d9d9357d8d2eb -m "it is very imp tag"
[root@ip-172-31-32-25 dir1]# git log
commit ea2f16fc81695348557120683b4d9d9357d8d2eb (HEAD -> master, tag: imp-tag3)
Author: jayshri <jayshrilandge56789@gmail.com>
Date:   Fri Aug 1 08:24:47 2025 +0000

    file3

commit f13ba555e2a9ee89dd80893b218b1e8d0997914b (tag: imp-tag)
Author: jayshri <jayshrilandge56789@gmail.com>
Date:   Fri Aug 1 08:24:34 2025 +0000

    file2

commit 6465a458b5a637ae3ef9200f8db6348156854274 (tag: imp-tag1)
Author: root <root@ip-172-31-32-25.eu-west-3.compute.internal>
Date:   Fri Aug 1 08:21:42 2025 +0000

    file1
[root@ip-172-31-32-25 dir1]# git show imp-tag3
tag imp-tag3
Tagger: jayshri <jayshrilandge56789@gmail.com>
Date:   Fri Aug 1 08:33:22 2025 +0000

it is very imp tag

commit ea2f16fc81695348557120683b4d9d9357d8d2eb (HEAD -> master, tag: imp-tag3)
Author: jayshri <jayshrilandge56789@gmail.com>
Date:   Fri Aug 1 08:24:47 2025 +0000

    file3

diff --git a/file3 b/file3
new file mode 100644
index 0000000..e69de29
```

Screenshot 14: Tagging workflow

Resetting Commits

Practical with at least 2 committed files:

git reset HEAD~1 # Mixed reset (to workspace)

git reset HEAD~1 --soft # Soft reset (to staging)

git reset HEAD~1 --hard # Hard reset (delete commit)

```
[root@ip-172-31-39-113 dir1]# git reset HEAD~1
[root@ip-172-31-39-113 dir1]# git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file3
        file4
        file5
```

Screenshot 15: Mixed reset type

```
[root@ip-172-31-39-113 dir1]# git reset HEAD~1 --soft
[root@ip-172-31-39-113 dir1]# git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file5
```

Screenshot 16: Soft reset type

```
[root@ip-172-31-39-113 dir1]# git reset HEAD~1 --hard
HEAD is now at dbd96c9 file1
[root@ip-172-31-39-113 dir1]# git status
On branch master
nothing to commit, working tree clean
[root@ip-172-31-39-113 dir1]# ls
file1  file2
```

Screenshot 17: Hard reset type

Git Stash Workflow

git stash # Temporarily save changes

git stash list # List stashes

git stash apply stash@{0} # Re-apply stashed changes

git stash drop stash@{0} # Delete specific stash

git stash clear # Clear all

```
[root@ip-172-31-32-25 dir1]# git stash
Saved working directory and index state WIP on master: 706b189  adding file1
[root@ip-172-31-32-25 dir1]# git stash list
stash@{0}: WIP on master: 706b189 adding file1
stash@{1}: WIP on master: 706b189 adding file1
stash@{2}: WIP on master: ea2f16f file3
```

Screenshot 18: Stash command

```
[root@ip-172-31-32-25 dir1]# git stash apply stash@{0}
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file2

no changes added to commit (use "git add" and/or "git commit -a")
```

Screenshot 19: apply command

```
[root@ip-172-31-32-25 dir1]# git stash drop stash@{1}
Dropped stash@{1} (036eb824ebeff9ffe4429833560033e2eb45b914)
[root@ip-172-31-32-25 dir1]# git stash list
stash@{0}: WIP on master: 706b189 adding file1
stash@{1}: WIP on master: ea2f16f file3
[root@ip-172-31-32-25 dir1]# git stash clear
[root@ip-172-31-32-25 dir1]# git stash list
```

Screenshot 20: drop and clear commands

Branching in Git

git branch # List branches

git branch branchA # Create new branch

git checkout branchA # Switch to branch

git checkout -b feature-login # Create & switch

```
[root@ip-172-31-38-118 dir1]# git branch
* master
[root@ip-172-31-38-118 dir1]# git branch branchA
[root@ip-172-31-38-118 dir1]# git branch
  branchA
* master
[root@ip-172-31-38-118 dir1]# git checkout branchA
A       file3
Switched to branch 'branchA'
[root@ip-172-31-38-118 dir1]# git branch
* branchA
  master
```

Screenshot 21: Branch creation and switching

Merging Branches

git checkout master

git merge branchA

```
[root@ip-172-31-38-118 dir1]# git branch
  branchA
* master
[root@ip-172-31-38-118 dir1]# git checkout branchA
A       file3
Switched to branch 'branchA'
[root@ip-172-31-38-118 dir1]# ls
file1  file2  file3  file7
[root@ip-172-31-38-118 dir1]# git merge master
Updating 837f88c..4dd39ba
Fast-forward
 file8 | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file8
[root@ip-172-31-38-118 dir1]# ls
file1  file2  file3  file7  file8
```

Screenshot 22: Before and after merge

Merge Conflict Example

yum install -y git

mkdir rohan && cd rohan

git init

cat > file1

1. **Create branchA** and modify file1
2. **Create branchB** and modify same line in file1

git branch branchA

git branch branchB

git checkout branchA

git add file1

git commit -m "file1 in branchA"

git checkout branchB

git add file1

git commit -m "file1 in branchB"

git checkout master

git merge branchA

git merge branchB # ⚠️ Conflict here

vim file1   # Resolve manually

git add file1

git commit -m "Resolved conflict between A & B"

```
[root@ip-172-31-38-118 rohan]# git checkout branchA
Switched to branch 'branchA'
[root@ip-172-31-38-118 rohan]# git branch
* branchA
  branchB
  master
[root@ip-172-31-38-118 rohan]# ls
file1
[root@ip-172-31-38-118 rohan]# cat file1
HELLO
THIS IS JAYSHRI
[root@ip-172-31-38-118 rohan]# cat >> file1
GIT IS VERSION CONTROL TOOL
[root@ip-172-31-38-118 rohan]# cat file1
HELLO
THIS IS JAYSHRI
GIT IS VERSION CONTROL TOOL
```

Screenshot 23: BranchA content

```
[root@ip-172-31-38-118 rohan]# git checkout branchB
Switched to branch 'branchB'
[root@ip-172-31-38-118 rohan]# cat file1
HELLO
THIS IS JAYSHRI
[root@ip-172-31-38-118 rohan]# cat >> file1
GIT IS VERY EASY
[root@ip-172-31-38-118 rohan]# cat file1
HELLO
THIS IS JAYSHRI
GIT IS VERY EASY
```

Screenshot 24: BranchB content

```
[root@ip-172-31-38-118 rohan]# git merge branchB
Auto-merging file1
CONFLICT (content): Merge conflict in file1
Automatic merge failed; fix conflicts and then commit the result.
[root@ip-172-31-38-118 rohan]#
```

Screenshot 25: Merge Conflict error

```
[root@ip-172-31-38-118 rohan]# git commit -m "Resolve merge conflict between branchA and branchB"
[master a91b422] Resolve merge conflict between branchA and branchB
 Committer: root <root@ip-172-31-38-118.eu-west-3.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

[root@ip-172-31-38-118 rohan]# git branch
  branchA
  branchB
* master
[root@ip-172-31-38-118 rohan]# ls
file1
[root@ip-172-31-38-118 rohan]# cat file1
HELLO
THIS IS JAYSHRI
GIT IS VERSION CONTROL TOOL
GIT IS VERY EASY
```

Screenshot 26: Conflict prompt and resolution

Managing Git Remote Repository & Local Repository

1. Add First Remote Repository and Push Code

git remote add origin <repo1_url> # Add Remote Repo 1

git push origin master # Push code to Repo 1

(This command links your local repo to the first remote repository (origin) and pushes the code to the master branch.)

```
[root@ip-172-31-32-7 dir2]# git remote add origin https://github.com/jayshrilandge30/repo1.git
[root@ip-172-31-32-7 dir2]# touch file1
[root@ip-172-31-32-7 dir2]# git add file1
[root@ip-172-31-32-7 dir2]# git commit -m "file1"
[master (root-commit) 753e466] file1
 Committer: root <root@ip-172-31-32-7.eu-west-3.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1
[root@ip-172-31-32-7 dir2]# git push origin master
Username for 'https://github.com': jayshrilandge30
Password for 'https://jayshrilandge30@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 218 bytes | 218.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/jayshrilandge30/repo1.git
 * [new branch]      master -> master
[root@ip-172-31-32-7 dir2]# ls
file1
```

Screenshot 27: Add-Remote1-and-Push

2. Add Second Remote Repository and Push Code

git remote add origin2 <repo2_url> # Add Remote Repo 2

git push origin2 master # Push code to Repo 2

(Here we add another remote (named origin2) and push the same code to the second repository.)

```
[root@ip-172-31-32-7 dir2]# git remote add origin2 https://github.com/jayshrilandge30/devops-auto-deploy-website.git
[root@ip-172-31-32-7 dir2]# touch file9
[root@ip-172-31-32-7 dir2]# git add file9
[root@ip-172-31-32-7 dir2]# git commit -m "file9"
[master 32830c4] file9
 Committer: root <root@ip-172-31-32-7.eu-west-3.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file9
[root@ip-172-31-32-7 dir2]# git push origin2 master
```

Screenshot 28: Add-Remote2-and-Push

3. Clone the First Repository and Pull Latest Changes

git clone <repo1_url> # Clone Repo 1 locally

(This command creates a local copy of the repository from GitHub.)

git pull <repo1_url> # Pull latest changes from Repo 1

(Fetches and integrates the latest changes from the remote repo into your local repo.)

```
[root@ip-172-31-32-88 ~]# git clone https://github.com/jayshrilandge30/repo1.git
Cloning into 'repo1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
[root@ip-172-31-32-88 ~]# ls
repo1
[root@ip-172-31-32-88 ~]# cd /repo1
-bash: cd: /repo1: No such file or directory
[root@ip-172-31-32-88 ~]# cd repo1
[root@ip-172-31-32-88 repo1]# ls
file1
[root@ip-172-31-32-88 repo1]# git pull https://github.com/jayshrilandge30/repo1.git
From https://github.com/jayshrilandge30/repo1
 * branch            HEAD       -> FETCH_HEAD
Already up to date.
[root@ip-172-31-32-88 repo1]# touch file2
[root@ip-172-31-32-88 repo1]# git add file2
[root@ip-172-31-32-88 repo1]# git commit -m "file2"
[master 8b1da57] file2
 Committer: root <root@ip-172-31-32-88.eu-west-3.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file2
[root@ip-172-31-32-88 repo1]# git push origin master
Username for 'https://github.com': jayshrilandge30
Password for 'https://jayshrilandge30@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 244 bytes | 244.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/jayshrilandge30/repo1.git
   753e466..8b1da57  master -> master
[root@ip-172-31-32-88 repo1]# ls
file1  file2
```

Screenshot 29: Clone-Repo1 & Pull-from-Repo1

.gitignore Usage

Create .gitignore to:

- Temporarily hide/untrack specific files
- Prevent pushing certain files to GitHub

```
[root@ip-172-31-34-88 dir1]# touch file11
[root@ip-172-31-34-88 dir1]# vim .gitignore
[root@ip-172-31-34-88 dir1]# git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
[root@ip-172-31-34-88 dir1]# rm -rf .gitignore
[root@ip-172-31-34-88 dir1]# git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file11

nothing added to commit but untracked files present (use "git add" to track)
```

Screenshot 30: Example .gitignore contents and effect

Final Hands-On Knowledge Recap

- Initialized Git repositories and configured global settings
- Practiced git add, commit, reset, stash, and tagging
- Cloned, pulled, pushed, and managed multiple remotes
- Handled branching, merging, and **merge conflicts**
- Worked with .gitignore and tag management

Conclusion

This hands-on practice reflects strong practical experience in using Git for version control and collaboration. It showcases real-world scenarios such as repository setup, file tracking, remote sync, conflict resolution, branching strategies, and usage of advanced Git features like reset, stash, and tags. Each command was practiced step-by-step, ensuring clear understanding and execution of Git workflows.