**GIT 1:**

**Step 1: Setup your machine with Git Configuration**

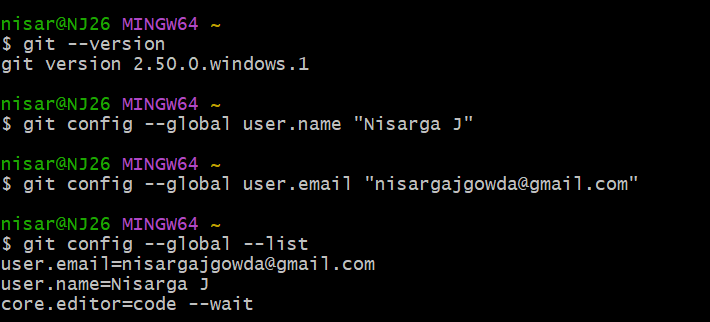
git –version

git config --global user.name "Nisarga J"

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

git config --global –list

**RESULT:**



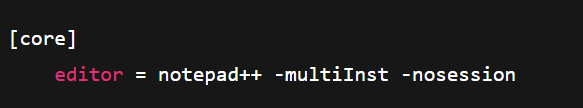
**Step 2: Integrate notepad++.exe to Git and make it a default editor**

alias np='notepad++'

git config --global core.editor "notepad++ -multiInst -nosession"

git config --global -e

**RESULT:**



**Step 3: Add a file to source code repository**

mkdir GitDemo

cd GitDemo

git init

echo "Welcome to Git Demo" > welcome.txt

ls

cat welcome.txt

git status

git add welcome.txt

git commit

**RESULT:**

Initialized empty Git repository in C:/path/to/GitDemo/.git/

welcome.txt

Welcome to Git Demo

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

welcome.txt

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: welcome.txt

[master (root-commit) abc1234] Initial commit

1 file changed, 1 insertion(+)

create mode 100644 welcome.txt

**Step 4: Push to Remote Repository**

git remote add origin [https://gitlab.com/ nisarga](https://gitlab.com/%20nisarga) j/GitDemo.git

git pull origin master

git push origin master

**RESULT:**

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Writing objects: 100% (3/3), 243 bytes | 243.00 KiB/s, done.

To https://gitlab.com/nisarga j/GitDemo.git

\* [new branch] master -> master

**GIT 2:**

Create .log file and log folder

echo "This is a log file" > debug.log

mkdir log

echo "Log folder content" > log/info.txt

Create .gitignore file

echo "\*.log" > .gitignore

echo "log/" >> .gitignore

Check .gitignore content

cat .gitignore

Check status

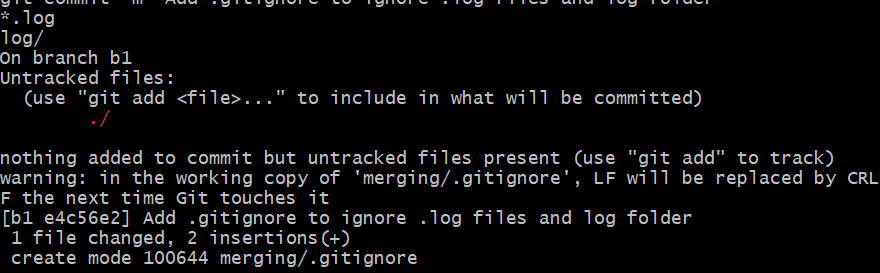
git status

Commit .gitignore

git add .gitignore

git commit -m "Add .gitignore to ignore .log files and log folder"

**RESULT:**

****

**GIT 3:**

Create a new branch

git branch GitNewBranch

List all branches

git branch -a

Switch to new branch

git checkout GitNewBranch

Add file and commit

echo "Branch specific content" > branchfile.txt

git add branchfile.txt

git commit -m "Add branch-specific file"

Check status

git status

Switch to master

git checkout master

Show differences (CLI)

git diff master GitNewBranch

Show differences (Visual with P4Merge)

git difftool master GitNewBranch

Merge branch into master

git merge GitNewBranch

Show log after merge

git log --oneline --graph --decorate

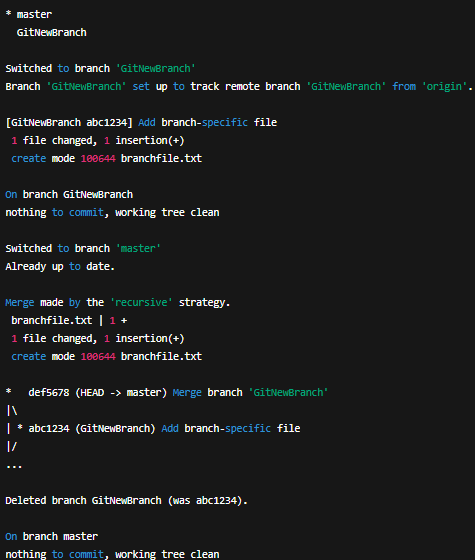
Delete merged branch

git branch -d GitNewBranch

Check status

git status

**RESULT:**

****

**GIT 4:**

Verify clean state

git status

Create new branch

git branch GitWork

git checkout GitWork

Add file in branch

echo "<message>Hello from branch</message>" > hello.xml

git add hello.xml

git commit -m "Add hello.xml in branch"

Update file in branch

echo "<message>Hello from branch - updated</message>" > hello.xml

git commit -am "Update hello.xml in branch"

Switch to master

git checkout master

Add different version of hello.xml

echo "<msg>Hello from master</msg>" > hello.xml

git add hello.xml

git commit -m "Add hello.xml in master"

Show log

git log --oneline --graph --decorate --all

Show differences

git diff master GitWork

git difftool master GitWork

Merge branch into master (will cause conflict)

git merge GitWork

Resolve conflict using 3-way merge tool (P4Merge)

(Edit hello.xml to keep correct content, then save & close)

Mark as resolved and commit

git add hello.xml

git commit -m "Resolve merge conflict in hello.xml"

Add backup file to .gitignore

echo "\*.orig" >> .gitignore

git add .gitignore

git commit -m "Ignore backup files"

List branches

git branch -a

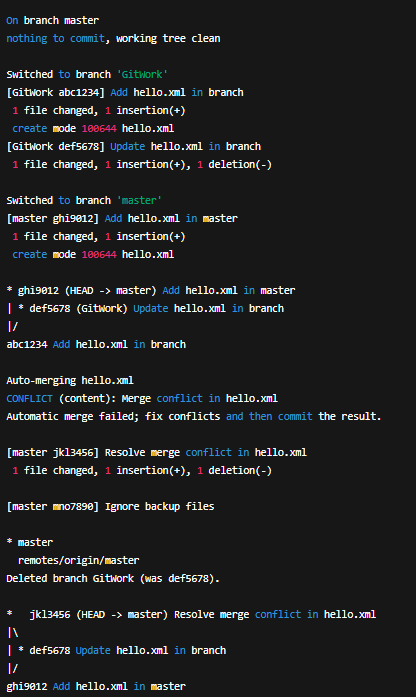
Delete branch after merge

git branch -d GitWork

Show final log

git log --oneline --graph --decorate

**RESULT:**

****

**GIT 5:**

git status

git branch -a

git pull origin master

git push origin master

**RESULT:**

