GIT CHALLENGE

**1.Resolve Merge Conflicts**

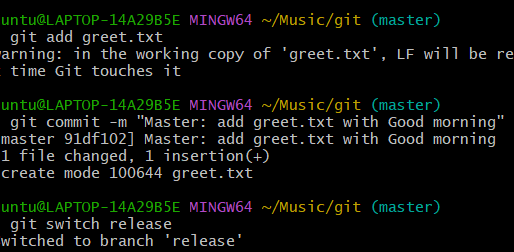
\*created 2 branches with similar contents in it

\*merge 2 those branches

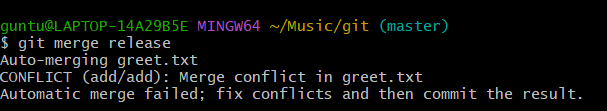
\*conflicts arrives

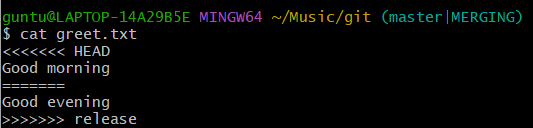
\*master branch = good morning

\*release branch = good evening



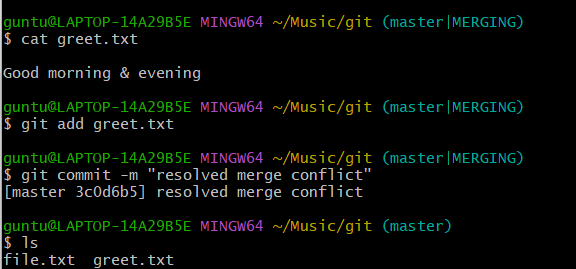
\*merge the 2 branches conflict arrives





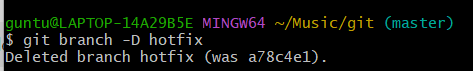
\*git add .

\*git commit – “merge conflict resolved”



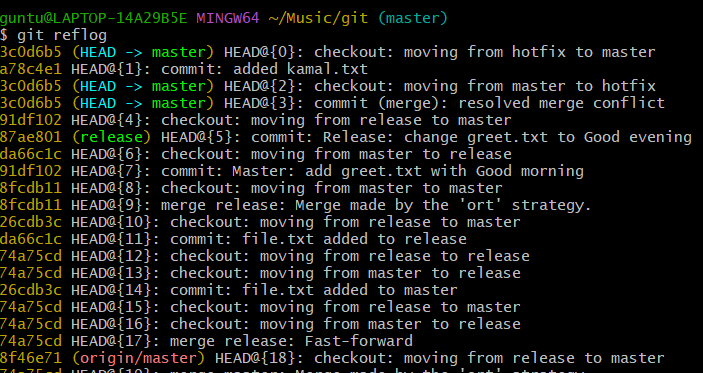
2. **Recover Deleted Branch**

**\***delete existing branch commad = git branch -D hotfix



\*check commit id

\*command= git reflog

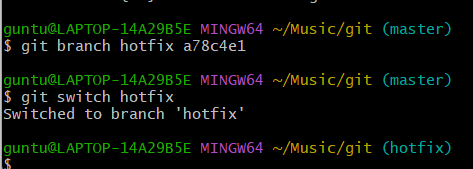


\*copy the commit id of hotfix

\*now to retrive the branch

\*git branch hotfix (commitid)

\*to verify git switch hotfix

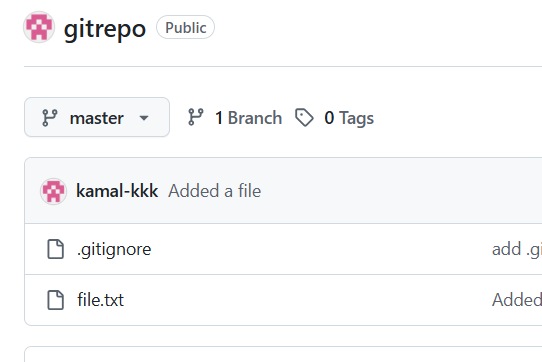


3. **Undo Wrong Push**

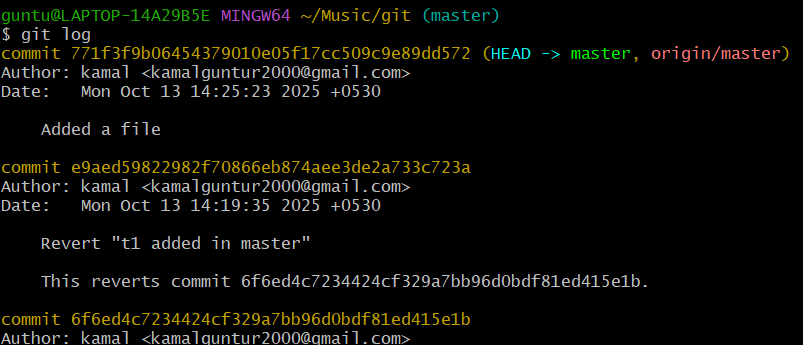
**\***git add .

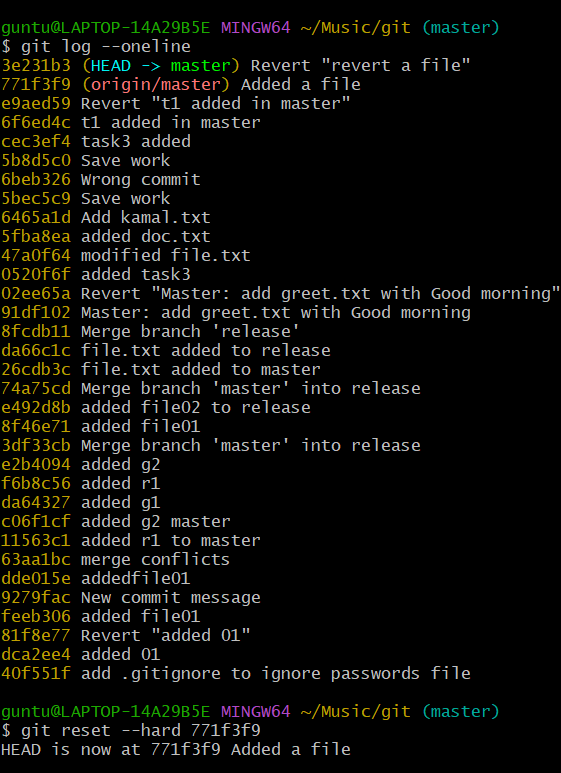
\*git commit -m “added to master”

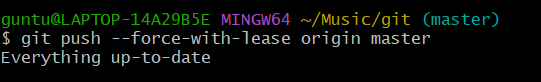
\*git push



\*git log for commit id







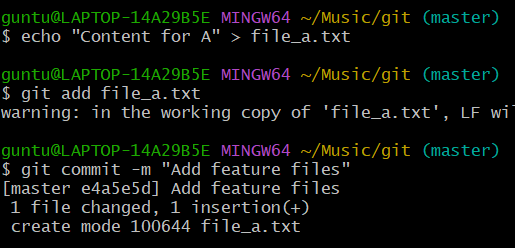
4. **Amend a Commit**

* + Make a commit, then add a missing file to it using git commit --amend.

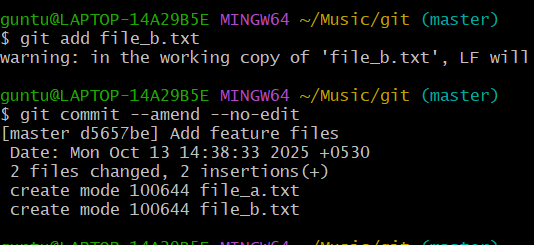
\*Created a file a

\*Git add.

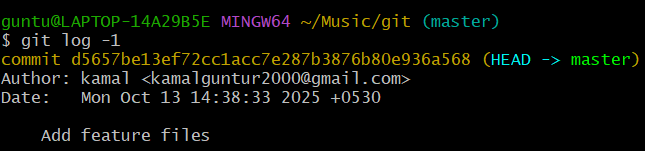
\*git commit -m



\*amend



\*to verify git log -1



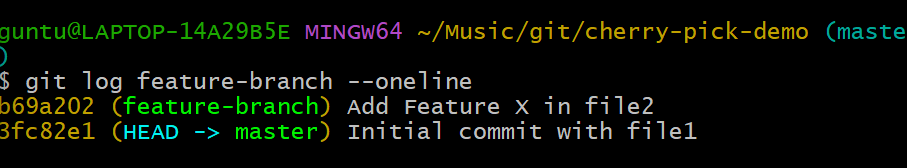
5. **Cherry-pick a Commit**

* Take a specific commit from one branch and apply it to another branch

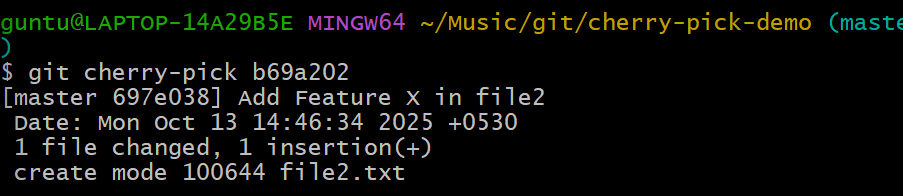
\*Create one file in 2 different branch

\*we different commit ids

\*git log feature-branch –oneline



\*git cherry pick and commit id



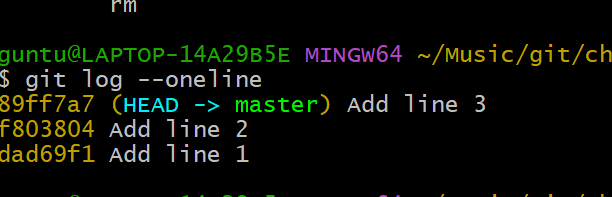
6. **Interactive Rebase**

Reorder and squash multiple commits into a single clean commit.

\*create 3 files

\*Add 1 line

\*git log –oneline

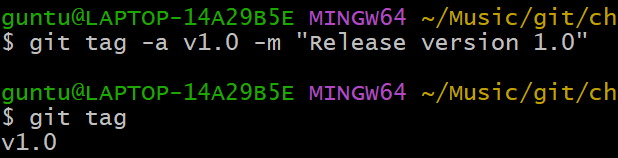
7. **Tagging & Release**

Create a version tag (v1.0), push it to GitHub, then delete and restore it.

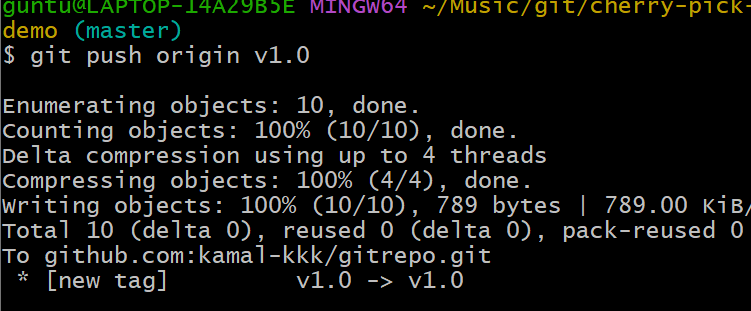
\*create version tag

\* git tag -a v1.0 -m "Release version 1.0"

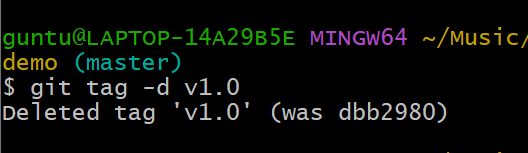
\*verify git tag



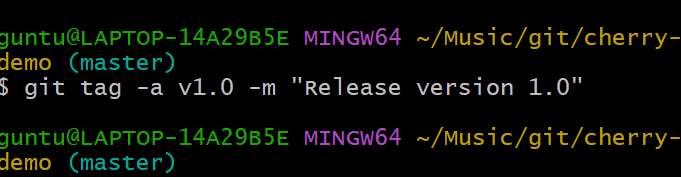
\*git push origin v1.0

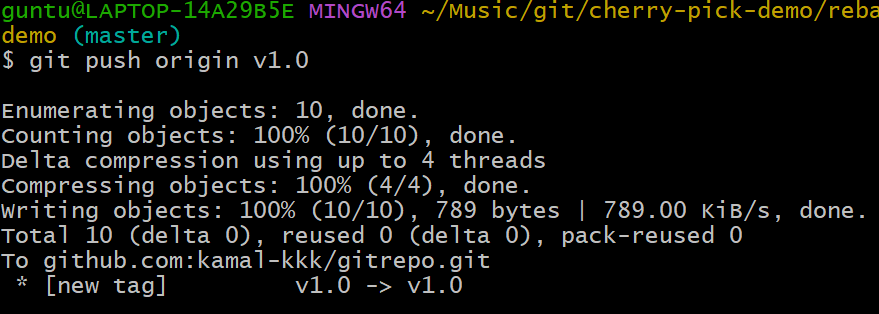


\*then delete it



\*to restore



\*verify git push origin v2.0

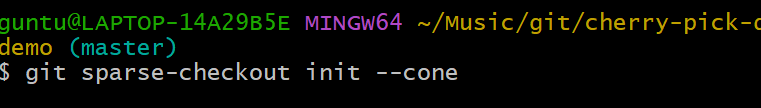
8. **Clone with Sparse Checkout**

Clone only a subdirectory of a repo using sparse checkout

\* git clone --no-checkout git@github.com:kamal-kkk/gitrepo.git myrepo

cd myrepo

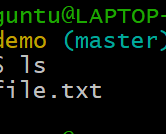
\* git sparse-checkout init –cone



\*select the subdirectory

Git sparse-checkout set src

\*ls to verify



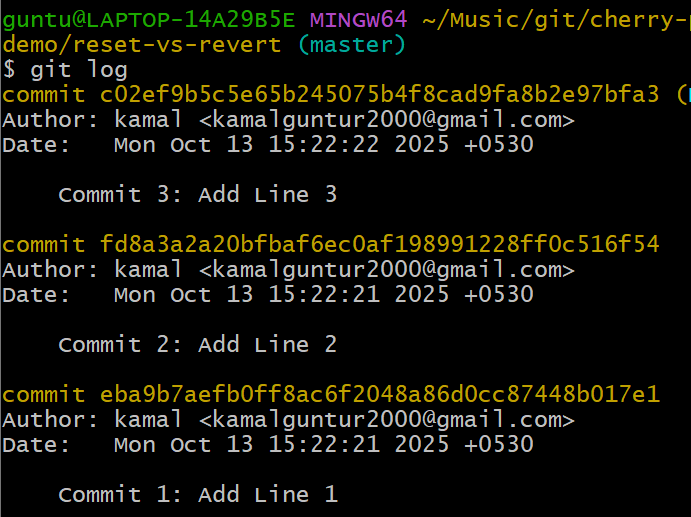
9. **Reset vs Revert Challenge**

Demonstrate the difference between git reset --hard and git revert in a repo.

\*create a repo reset nd revert

\*get init

\*created commit ids b files



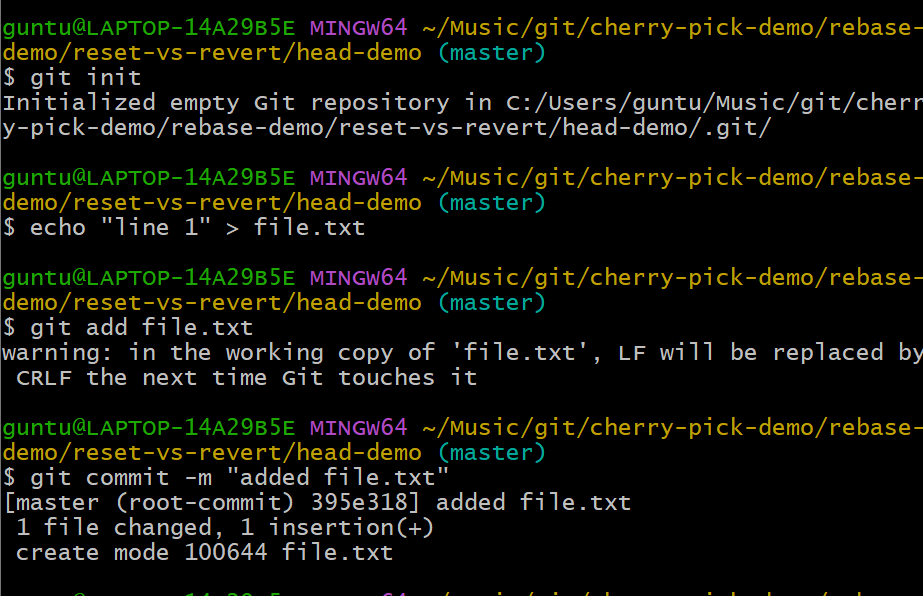
\*git log –oneline



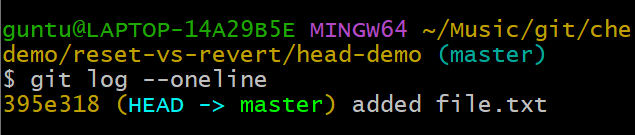
10. **Detached HEAD Challenge**

Checkout a specific commit (detached HEAD state) and create a new branch from it.

\*create a file

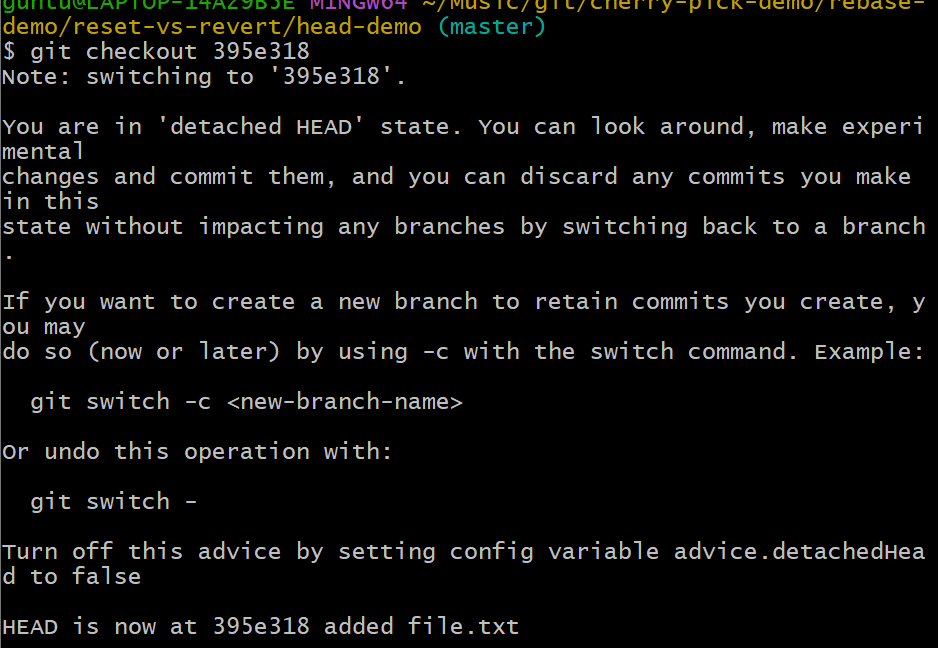


\*git log –oneline for commit ids

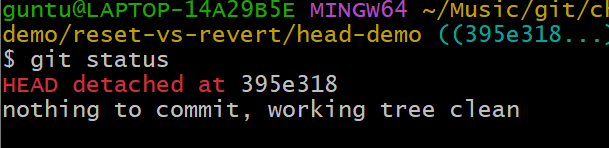


\*git checkout head

\*git checkout (commit id)



\*to verify

Git checkout (commit id)  


11. **Git Hooks Challenge**

\*create commit-msg

\*vi commit

\*add script

#!/bin/bash

# Get the commit message

commit\_msg=$(cat "$1")

# Regex to check for JIRA ticket at the start

pattern="^JIRA-[0-9]+:"

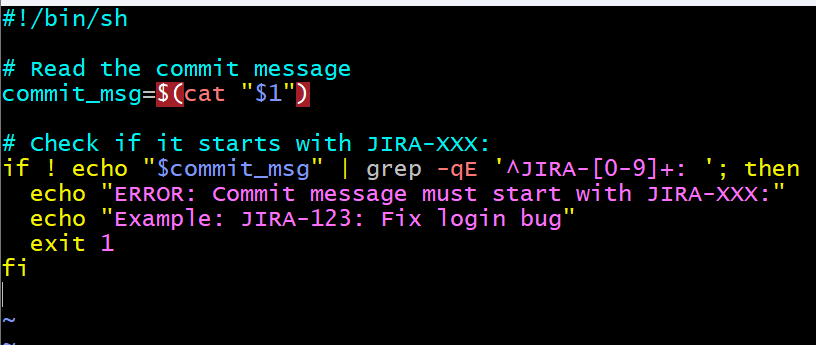
if ! [[ "$commit\_msg" =~ $pattern ]]; then

echo "ERROR: Commit message must start with JIRA-XXX:"

echo "Example: JIRA-101: Add new feature"

exit 1 # Exit with non-zero to reject commit

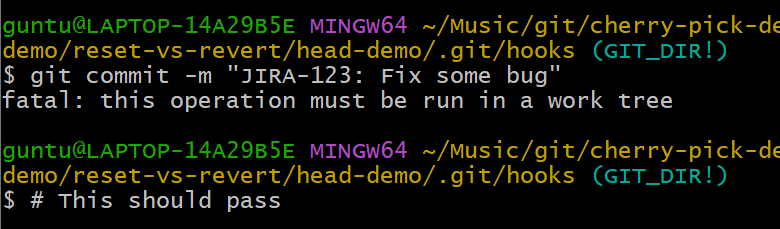
fi



\*

Git commit -m “JIRA-123: fix some bug”

\*this will pass



12. **Squash Merge vs Rebase Merge**

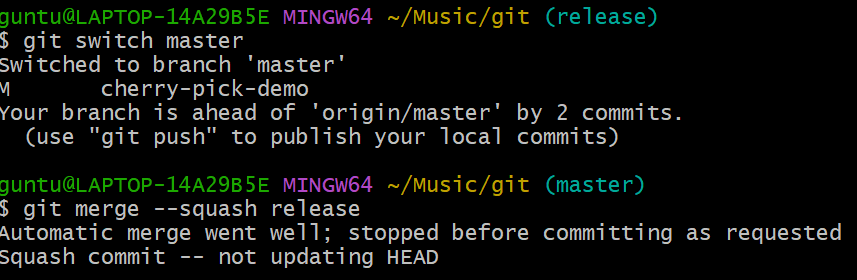
**\***created branch master and release

\*master branch has 1 file

\*release branch has 3 files

\*all the files are in tracking zone and committed

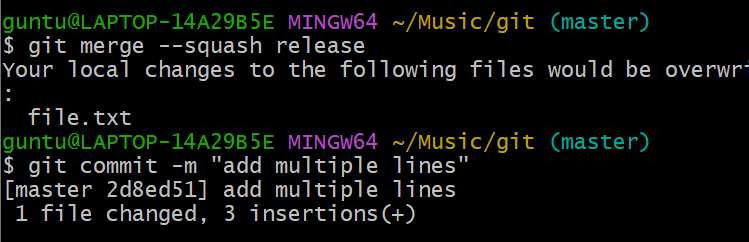
\*to squash merge



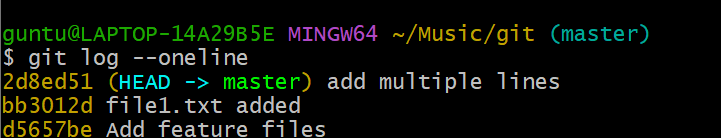
\*switch to master branch

\*Git merge –squahs merge release

\*git commit -m “add multiple line”



\*to verify git log –oneline



REBASE MERGE

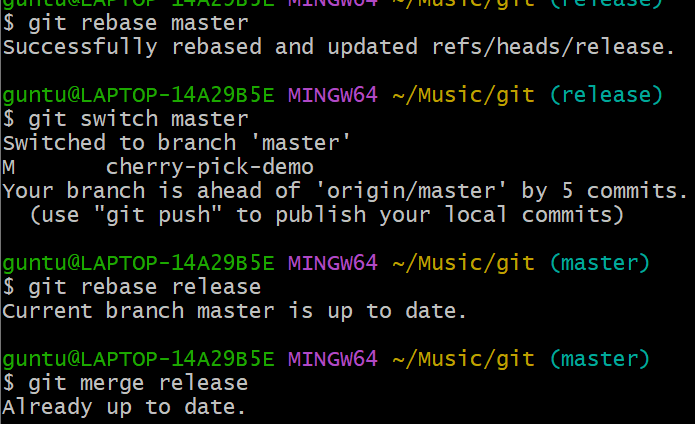
\*switch to release

\*git rebase master

\*now switch to master

\*git rebase release

\*git merge release



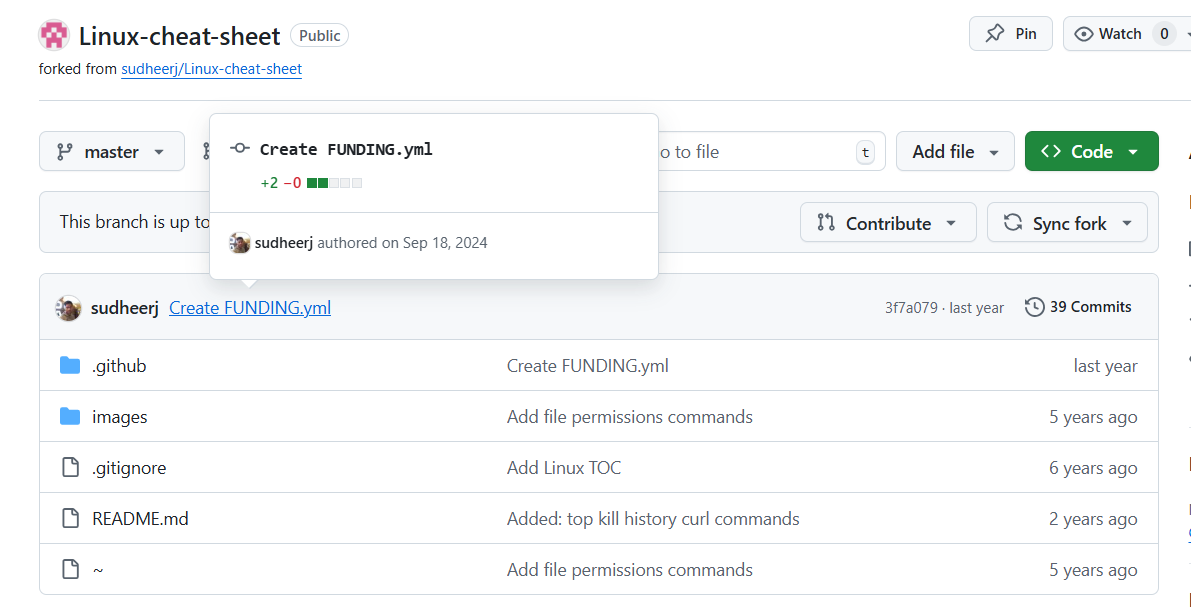
\*to verify git log –oneline



13. **Fork & Pull Request Workflow**

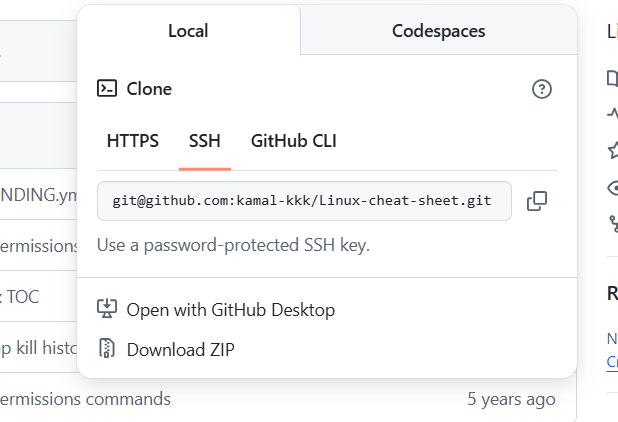
Fork a repo, make a change, and submit a pull request to the original repo.

\*

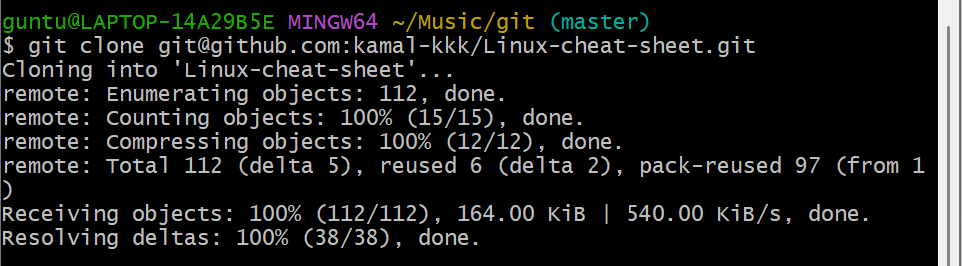


\*cloning

\*git clone [git@github.com:kamal-kkk/Linux-cheat-sheet.git](mailto:git@github.com:kamal-kkk/Linux-cheat-sheet.git)

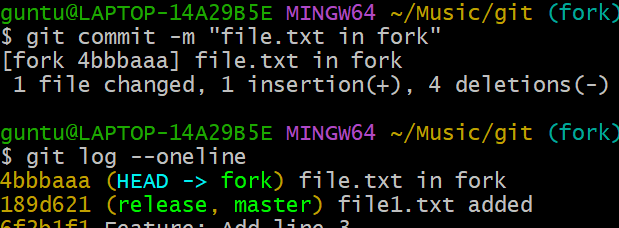


\*command- git clone [git@github.com:kamal-kkk/Linux-cheat-sheet.git](mailto:git@github.com:kamal-kkk/Linux-cheat-sheet.git)



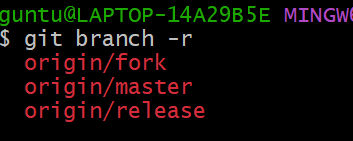
\*created new branch called fork

\*git add . in tracking zone and committed

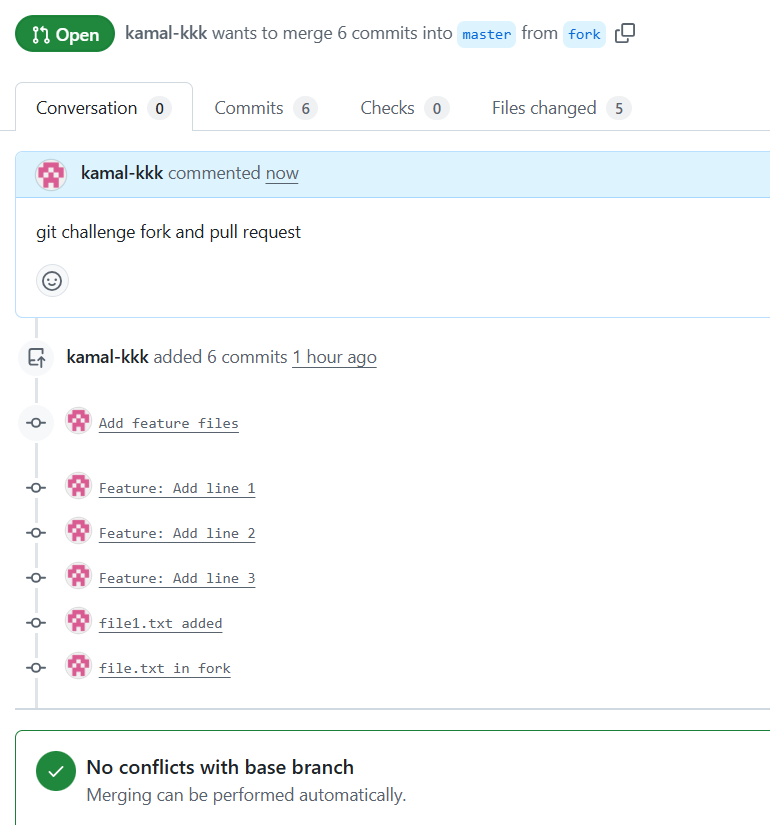


\*git push origin fork

\*git branch -r



\*pull request



14. **Recover Lost Commit**

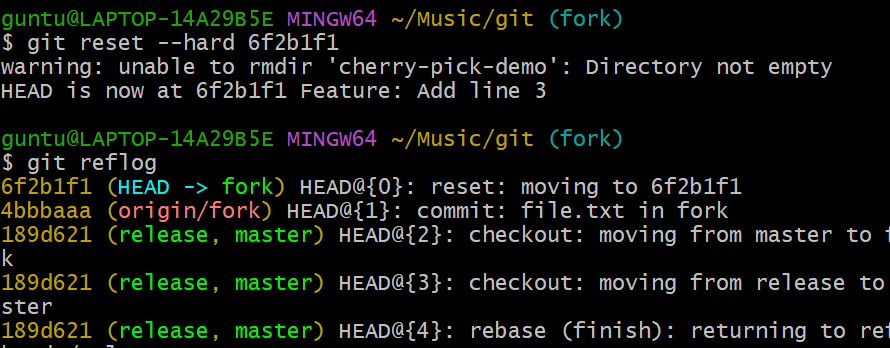
Commit something, reset hard, and then recover it using git reflog.

\*git log –oneline

\*choose the commit id

\*Git reset --hard 6f2b1f1

\*git reflog



\*git switch 6fb1f1

\*restore permanently git reset –hard 6fb1f1

\*git log –oneline

\*to verify

