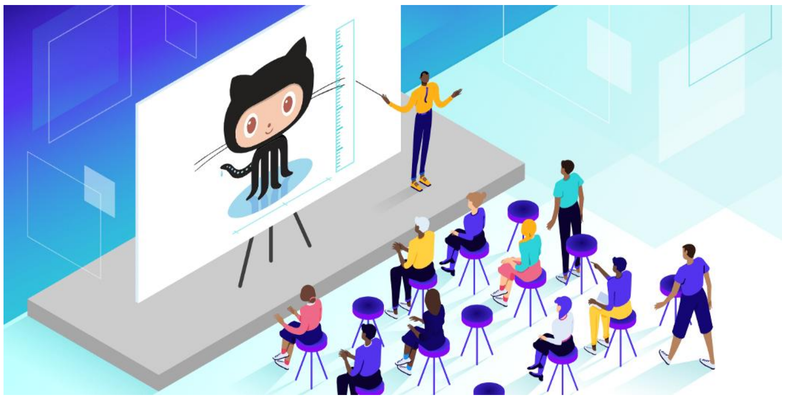
Day 10: Advance Git & GitHub

This is #90DaysofDevops challenge under the guidance of Shubham Londhe sir.

Day 10 TASK

check this for task:

https://github.com/LondheShubham153/90DaysOfDevOps/blob/master/2023/day10/tasks.md



Git Branching

Git branching is a feature in the Git version control system that allows developers to create multiple independent branches of their codebase, enabling them to work on different features or fixes simultaneously without affecting the main codebase.

Changes made to a branch can be merged back into the main codebase once they are complete and have been reviewed. Git branching is a powerful tool that allows developers to manage code changes more efficiently and collaboratively.

Git Revert and Reset

Git revert undoes a specific commit by creating a new commit that undoes the changes made by the original commit. This is a safe way to undo changes, as it does not change the commit history and allows you to easily undo the undo if needed.

Git reset, on the other hand, allows you to move the current branch to a specific commit, which can potentially change the commit history. It’s important to use reset with caution, as it can cause data loss and make it difficult to undo changes.

What Is Git Rebase?

Git rebase is a command that allows you to apply the changes from one branch onto another. This can be useful for keeping your branch up-to-date with changes made in a separate branch or for creating a linear history before merging your changes into the main branch.

What Is Git Merge?

Git merge is a command that allows you to combine the changes from two or more branches into a single branch. When you run `git merge`, Git will automatically try to merge the changes made in the source branch into the target branch.

If there are no conflicts between the changes, Git will create a new commit that includes all the changes from the merged branch. If there are conflicts, Git will prompt you to manually resolve them before the merge can be completed.

Merging is a common way to integrate changes from other branches into your working branch. It creates a new commit that represents the combination of changes.

TASK 1 :

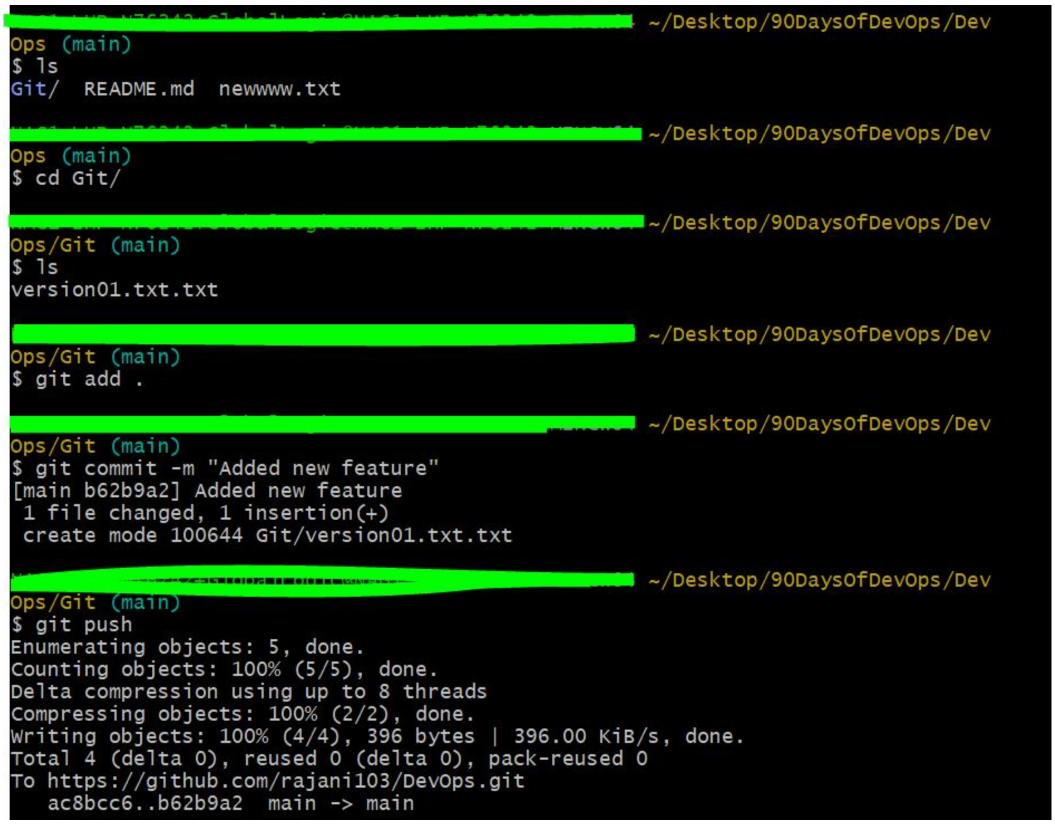
Add a text file called version01.txt inside the Devops/Git/ with "This is first feature of our application" written inside. This should be in a branch coming from `master`, [hint try `git checkout -b dev`], swithch to `dev` branch (Make sure your commit message will reflect as "Added new feature"). [Hint use your knowledge of creating branches and Git commit command]

- version01.txt should reflect at local repo first followed by Remote repo for review. [Hint use your knowledge of Git push and git pull commands here]

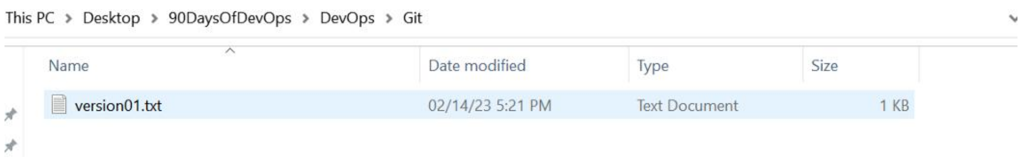
Add new commit in `dev` branch after adding below mentioned content in Devops/Git/version01.txt: While writing the file make sure you write these lines

- 1st line>> This is the bug fix in development branch
- Commit this with message " Added feature2 in development branch"
- 2nd line>> This is gadbad code
- Commit this with message " Added feature3 in development branch
- 3rd line>> This feature will gadbad everything from now.

- Commit with message " Added feature4 in development branch
-

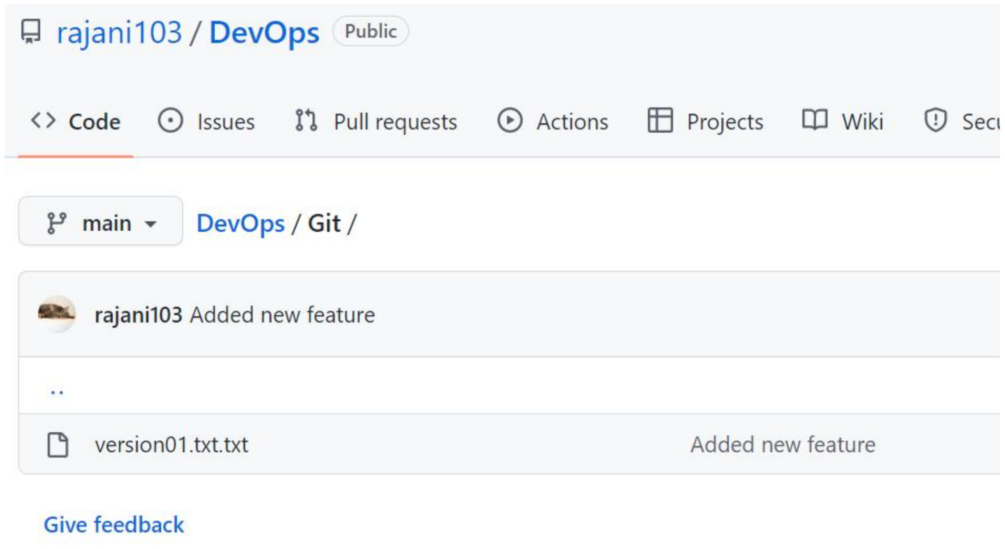


```
[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops (main)
$ ls
Git/  README.md  newwww.txt

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops (main)
$ cd Git/

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (main)
$ ls
version01.txt.txt

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (main)
$ git add .

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (main)
$ git commit -m "Added new feature"
[main b62b9a2] Added new feature
 1 file changed, 1 insertion(+)
 create mode 100644 Git/version01.txt.txt

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 396 bytes | 396.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/rajani103/DevOps.git
   ac8bcc6..b62b9a2  main -> main
```



This PC > Desktop > 90DaysOfDevOps > DevOps > Git

Name	Date modified	Type	Size
version01.txt	02/14/23 5:21 PM	Text Document	1 KB



rajani103 / DevOps Public

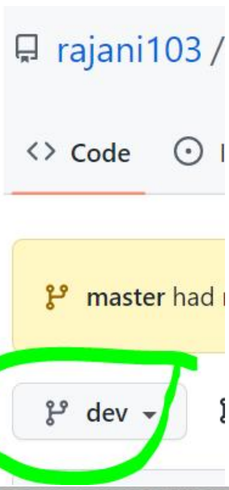
DevOps / Git /

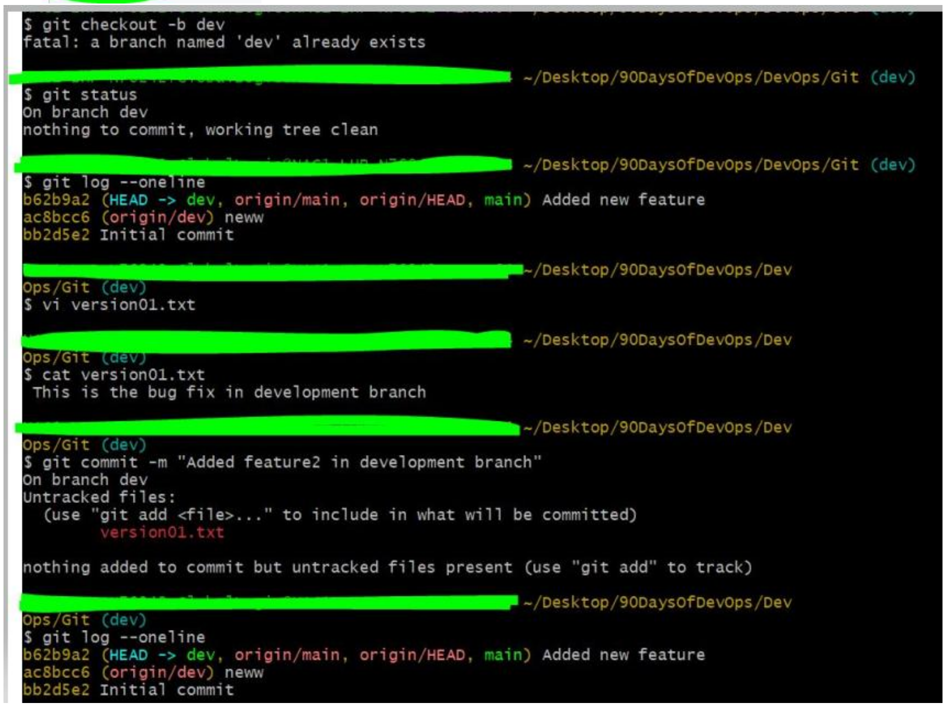
rajani103 Added new feature

..

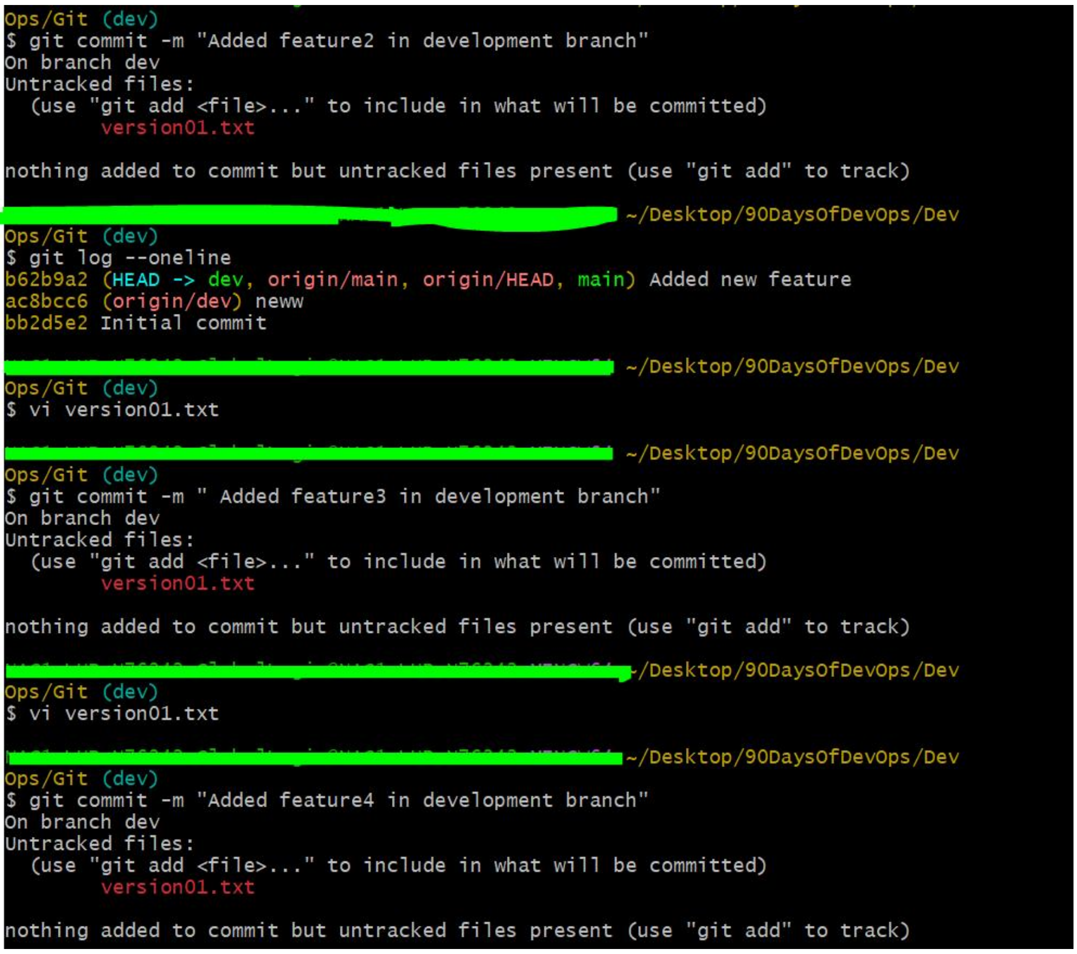
version01.txt.txt   Added new feature

2. Create a new Branch "dev" and commit the changes



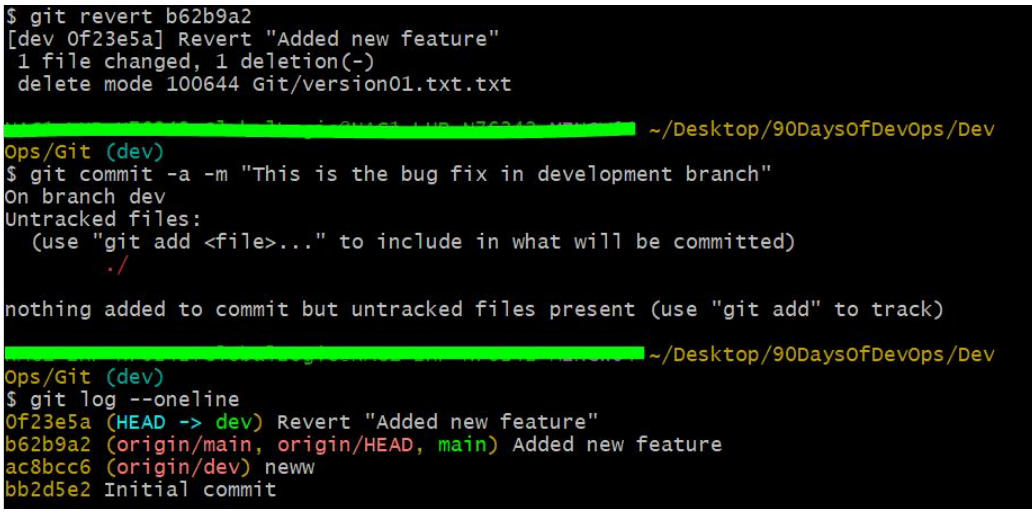


```
$ git checkout -b dev
fatal: a branch named 'dev' already exists

[redacted] ~/Desktop/90DaysOfDevOps/DevOps/Git (dev)
$ git status
On branch dev
nothing to commit, working tree clean

[redacted] ~/Desktop/90DaysOfDevOps/DevOps/Git (dev)
$ git log --oneline
b62b9a2 (HEAD -> dev, origin/main, origin/HEAD, main) Added new feature
ac8bcc6 (origin/dev) neww
bb2d5e2 Initial commit

[redacted]~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ vi version01.txt

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ cat version01.txt
 This is the bug fix in development branch

[redacted]~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ git commit -m "Added feature2 in development branch"
On branch dev
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        version01.txt

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

[redacted]~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ git log --oneline
b62b9a2 (HEAD -> dev, origin/main, origin/HEAD, main) Added new feature
ac8bcc6 (origin/dev) neww
bb2d5e2 Initial commit
```

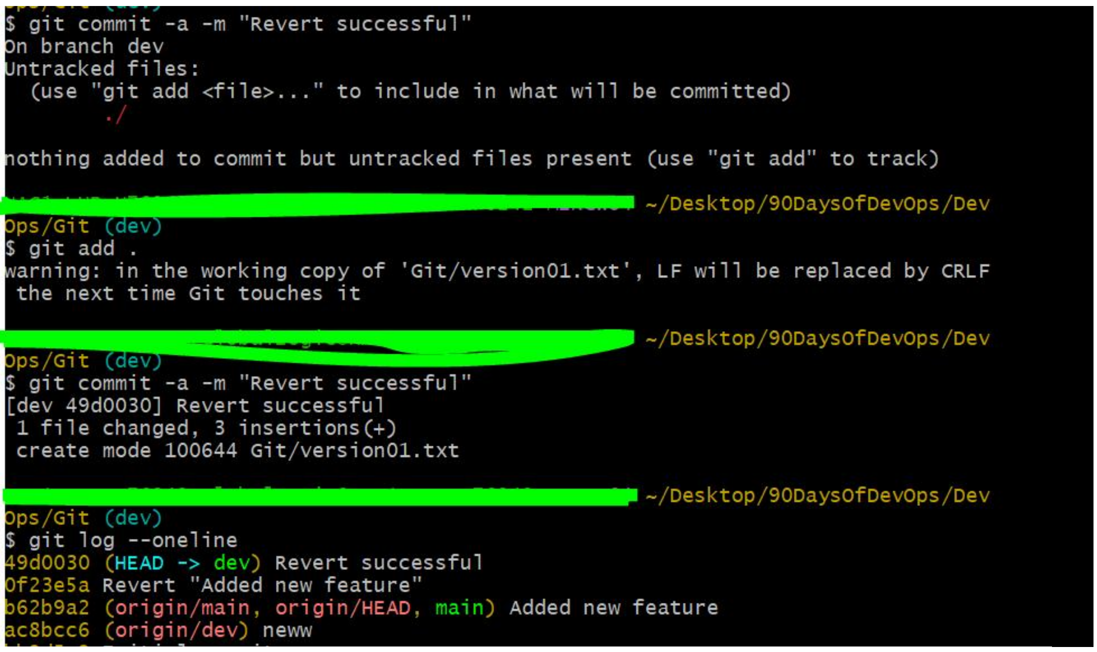


```
Ops/Git (dev)
$ git commit -m "Added feature2 in development branch"
On branch dev
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        version01.txt

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

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ git log --oneline
b62b9a2 (HEAD -> dev, origin/main, origin/HEAD, main) Added new feature
ac8bcc6 (origin/dev) neww
bb2d5e2 Initial commit

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ vi version01.txt

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ git commit -m " Added feature3 in development branch"
On branch dev
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        version01.txt

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

[redacted]~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ vi version01.txt

[redacted]~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ git commit -m "Added feature4 in development branch"
On branch dev
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        version01.txt

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

3. Use git reset command to go back to correct file. And after that switch to main branch

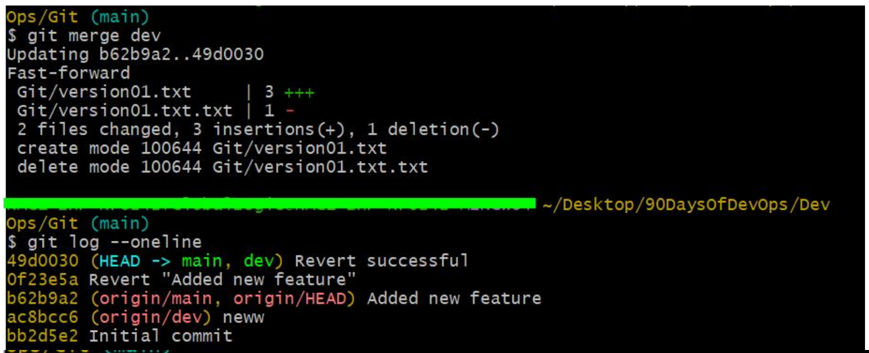


```
$ git revert b62b9a2
[dev 0f23e5a] Revert "Added new feature"
 1 file changed, 1 deletion(-)
 delete mode 100644 Git/version01.txt.txt

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ git commit -a -m "This is the bug fix in development branch"
On branch dev
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ./

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

[redacted]~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ git log --oneline
0f23e5a (HEAD -> dev) Revert "Added new feature"
b62b9a2 (origin/main, origin/HEAD, main) Added new feature
ac8bcc6 (origin/dev) neww
bb2d5e2 Initial commit
```

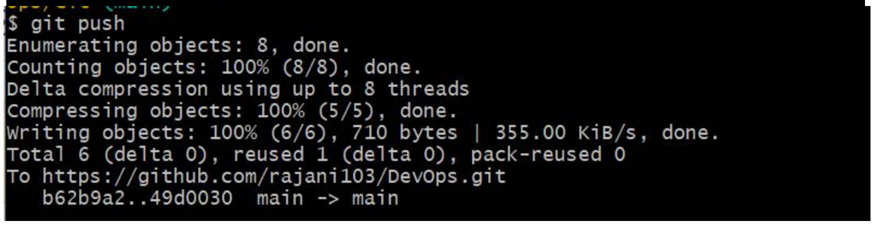


```
$ git commit -a -m "Revert successful"
On branch dev
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ./

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

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ git add .
warning: in the working copy of 'Git/version01.txt', LF will be replaced by CRLF
 the next time Git touches it

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ git commit -a -m "Revert successful"
[dev 49d0030] Revert successful
 1 file changed, 3 insertions(+)
 create mode 100644 Git/version01.txt

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ git log --oneline
49d0030 (HEAD -> dev) Revert successful
0f23e5a Revert "Added new feature"
b62b9a2 (origin/main, origin/HEAD, main) Added new feature
ac8bcc6 (origin/dev) neww
```

4. Merge dev branch to main branch and push the changes to remote Repository

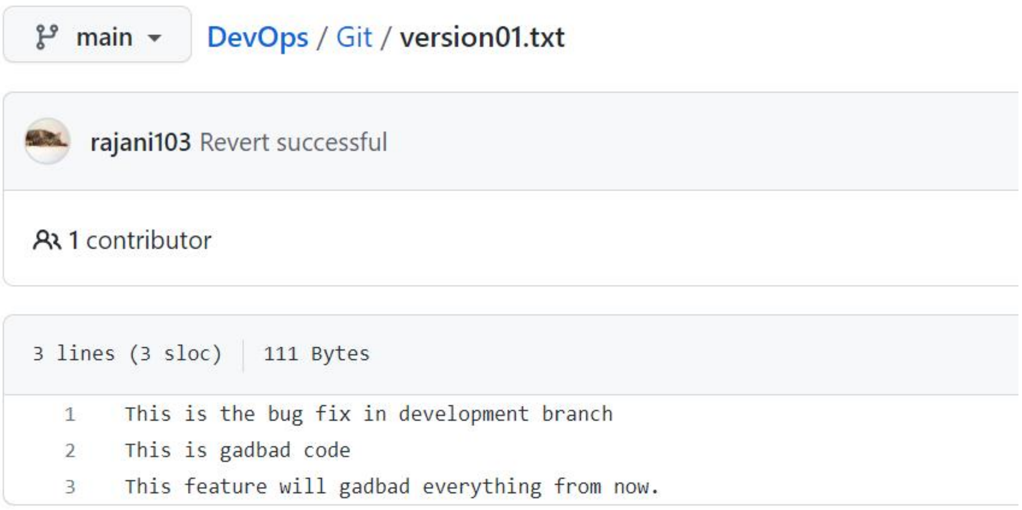


```
Ops/Git (main)
$ git merge dev
Updating b62b9a2..49d0030
Fast-forward
 Git/version01.txt     | 3 +++
 Git/version01.txt.txt | 1 -
 2 files changed, 3 insertions(+), 1 deletion(-)
 create mode 100644 Git/version01.txt
 delete mode 100644 Git/version01.txt.txt

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (main)
$ git log --oneline
49d0030 (HEAD -> main, dev) Revert successful
0f23e5a Revert "Added new feature"
b62b9a2 (origin/main, origin/HEAD) Added new feature
ac8bcc6 (origin/dev) neww
bb2d5e2 Initial commit
```



```
$ git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 710 bytes | 355.00 KiB/s, done.
Total 6 (delta 0), reused 1 (delta 0), pack-reused 0
To https://github.com/rajani103/DevOps.git
   b62b9a2..49d0030  main -> main
```

5. Change are Updated on Remote Repository



main

DevOps / Git / version01.txt

rajani103 Revert successful

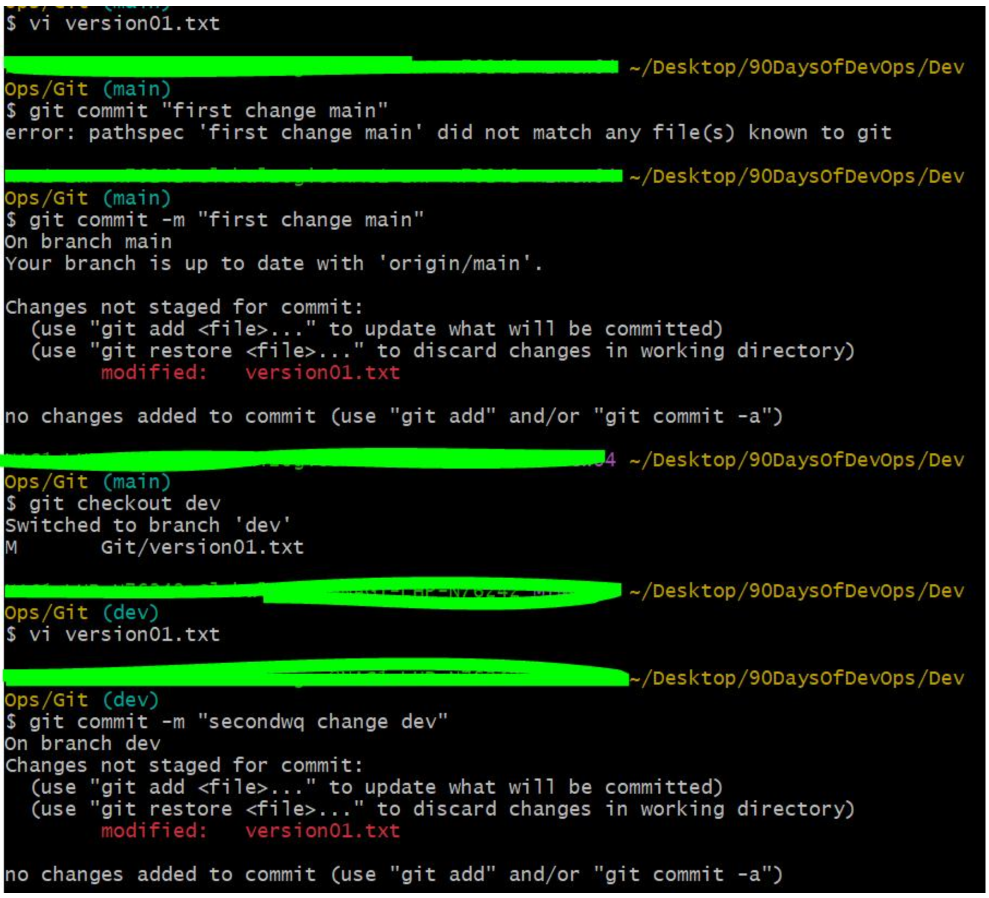
1 contributor

3 lines (3 sloc) | 111 Bytes

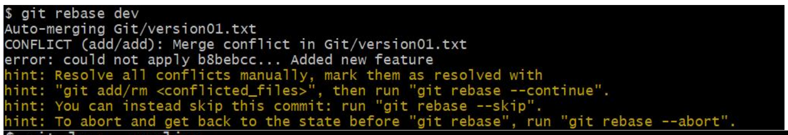
```
This is the bug fix in development branch
This is gadbad code
This feature will gadbad everything from now.
```

Task 2:

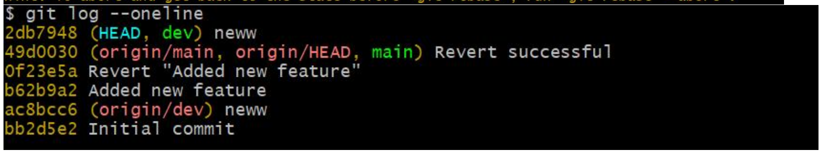
- Demonstrate the concept of branches with 2 or more branches with screenshot.
- add some changes to `dev` branch and merge that branch in `master`
- as a practice try git rebase too, see what difference you get.
-



```
$ vi version01.txt

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (main)
$ git commit "first change main"
error: pathspec 'first change main' did not match any file(s) known to git

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (main)
$ git commit -m "first change main"
On branch main
Your branch is up to date with 'origin/main'.

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:   version01.txt

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

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (main)
$ git checkout dev
Switched to branch 'dev'
M       Git/version01.txt

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ vi version01.txt

[redacted] ~/Desktop/90DaysOfDevOps/Dev
Ops/Git (dev)
$ git commit -m "secondwq change dev"
On branch dev
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:   version01.txt

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



```
$ git rebase dev
Auto-merging Git/version01.txt
CONFLICT (add/add): Merge conflict in Git/version01.txt
error: could not apply b8bebcc... Added new feature
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
```



```
$ git log --oneline
2db7948 (HEAD, dev) neww
49d0030 (origin/main, origin/HEAD, main) Revert successful
0f23e5a Revert "Added new feature"
b62b9a2 Added new feature
ac8bcc6 (origin/dev) neww
bb2d5e2 Initial commit
```

Please, feel free to drop any questions in the comments below. I would be happy to answer them.

If this post was helpful, please do follow and click the clap

_Thank you for reading

_Rajani