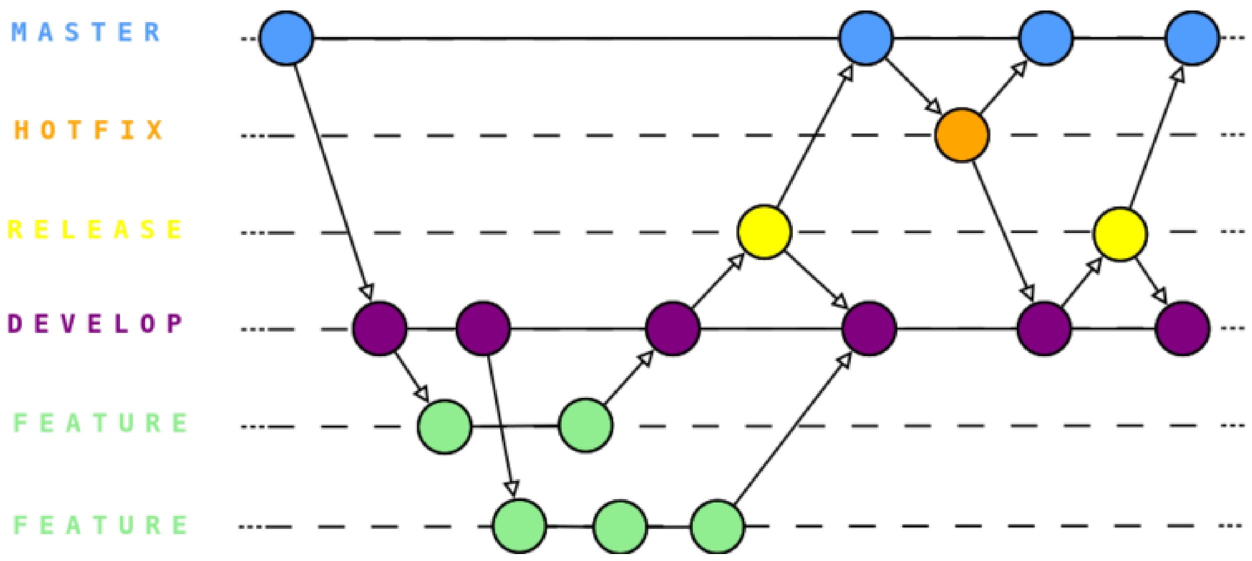
GIT FUNDAMENTAL HOMEWORK

1. Git flow & branching model

The Git flow Workflow defines a strict branching model designed around the project release and provides a robust framework for managing larger projects. This flow is described as below:



There are basically 5 different branches:

- master – branch without direct access for the developers. The purpose of this single branch is to keep the most stable deployable version of the code.

- develop – this is the focus point for the developers where all feature branches are created from and later merged to. Same as with master, no one operates directly on this branch.

- feature – a set of branches. Here you develop new functionalities! Always derived from develop and always merged with it.

- release – a set of branches. When your development feature reaches the form, you want to release a new version of the app, this is release.

- hotfix – a set of branches. When a new bug is discovered on production you create a new hotfix branch from master and after fixing it you merge it back to master as well as develop.

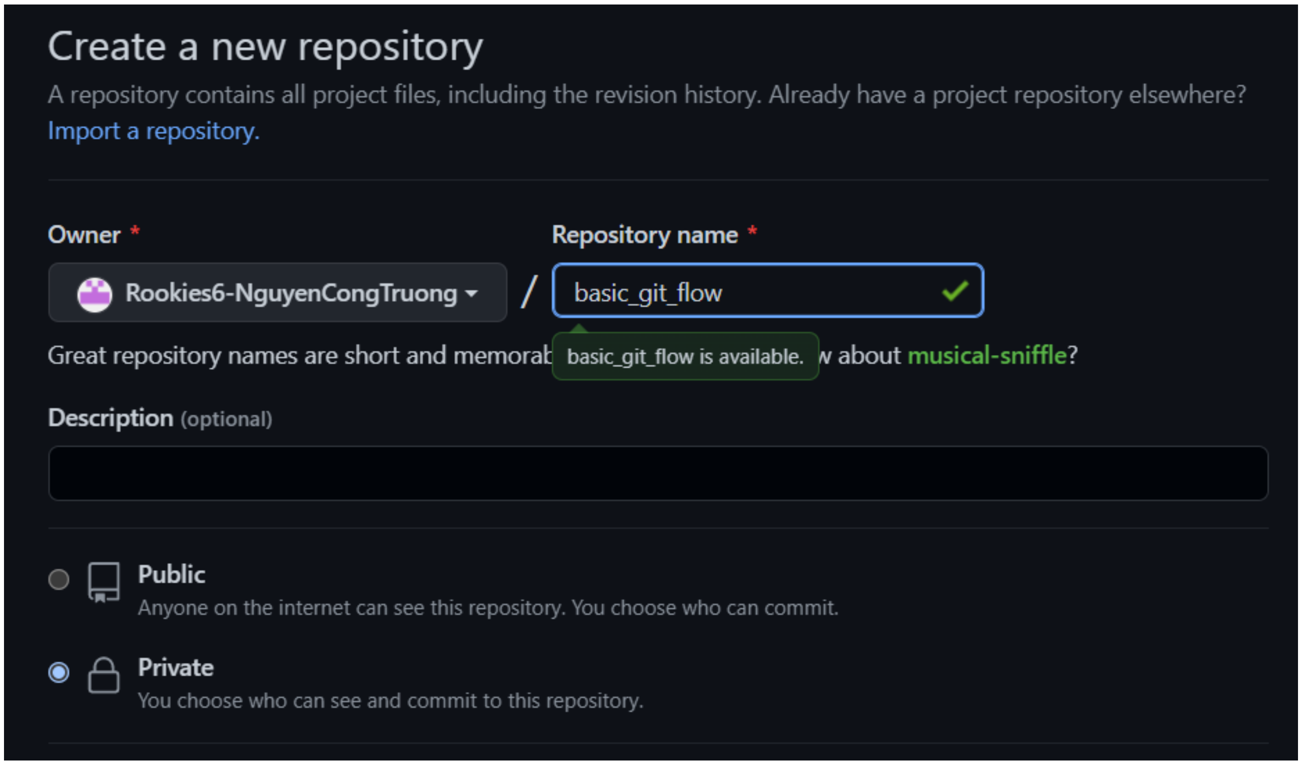
	Derived from	Merge to (use PRs)	Life cycle
master	-	-	Initiate
develop	-	-	Initiate
feature	develop	develop	Deleted after merge
release	develop	master, develop	Deleted after release
hotfix	master	master, develop	Deleted after merge

2. Questions

Scenario: the team has 5 developers (Team leader – TL, DevA, DevB, DevC, DevD), following Scrum/Agile methodology and **uses basic Git flow (section 1)**.

Q1. For source code management, at sprint 0, TL want to create a private Git repository named “basic_git_flow” on Github with default “master/main” branch. (hints: go to Github and do somethings then take a screen shot with browser url).

Answer:



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * Rookies6-NguyenCongTruong / Repository name * basic_git_flow

Great repository names are short and memorab basic_git_flow is available. w about musical-sniffle?

Description (optional)

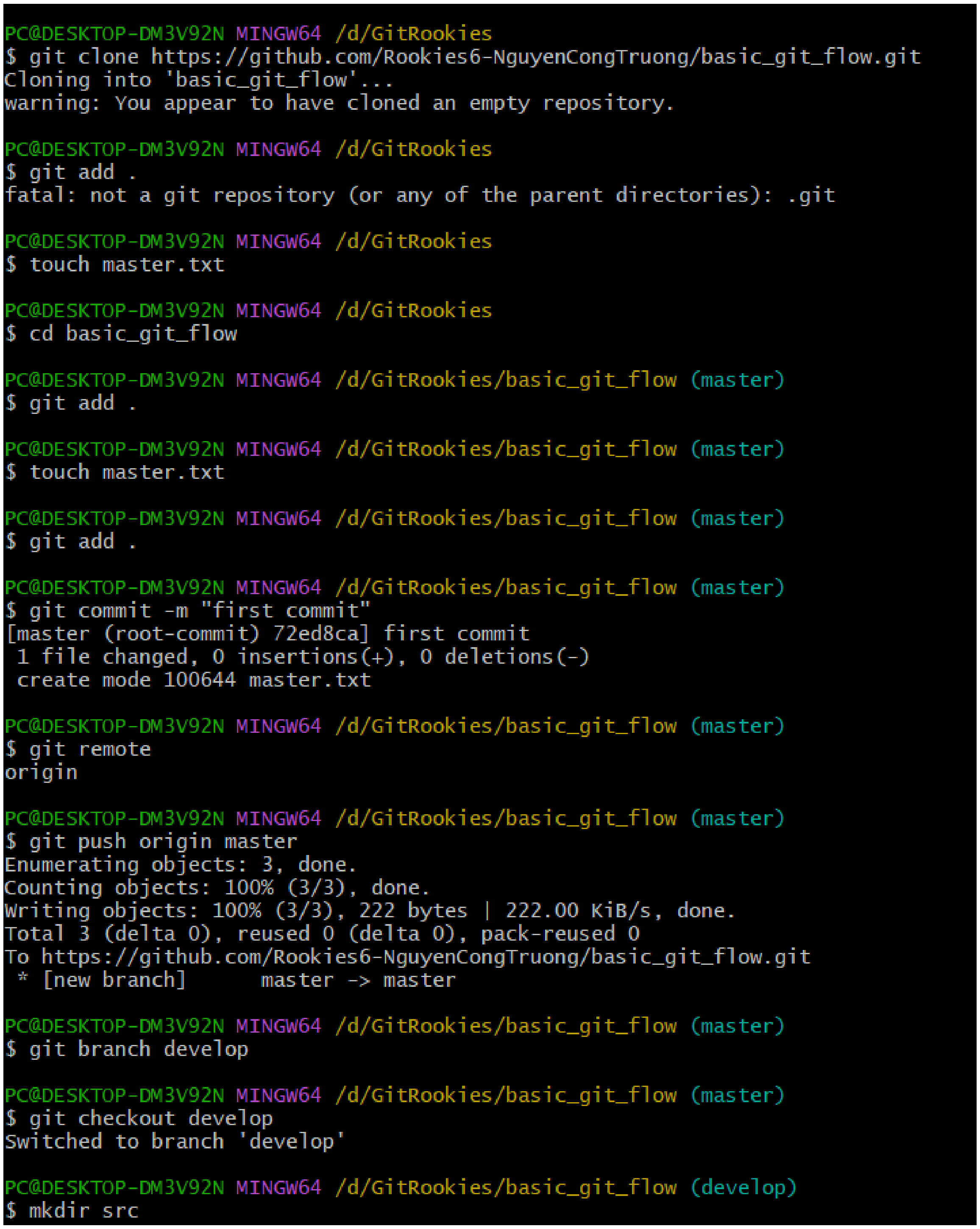
○ Public
Anyone on the internet can see this repository. You choose who can commit.

● Private
You choose who can see and commit to this repository.

Q2. At sprint 0, TL want to create “develop” branch, branch contains “src” folder and file “commonBehavior” in “src” folder. Write Git commands to the requirement. (hints: clone repo, checkout master, create new branch, add some files, stage changes, commit, push)

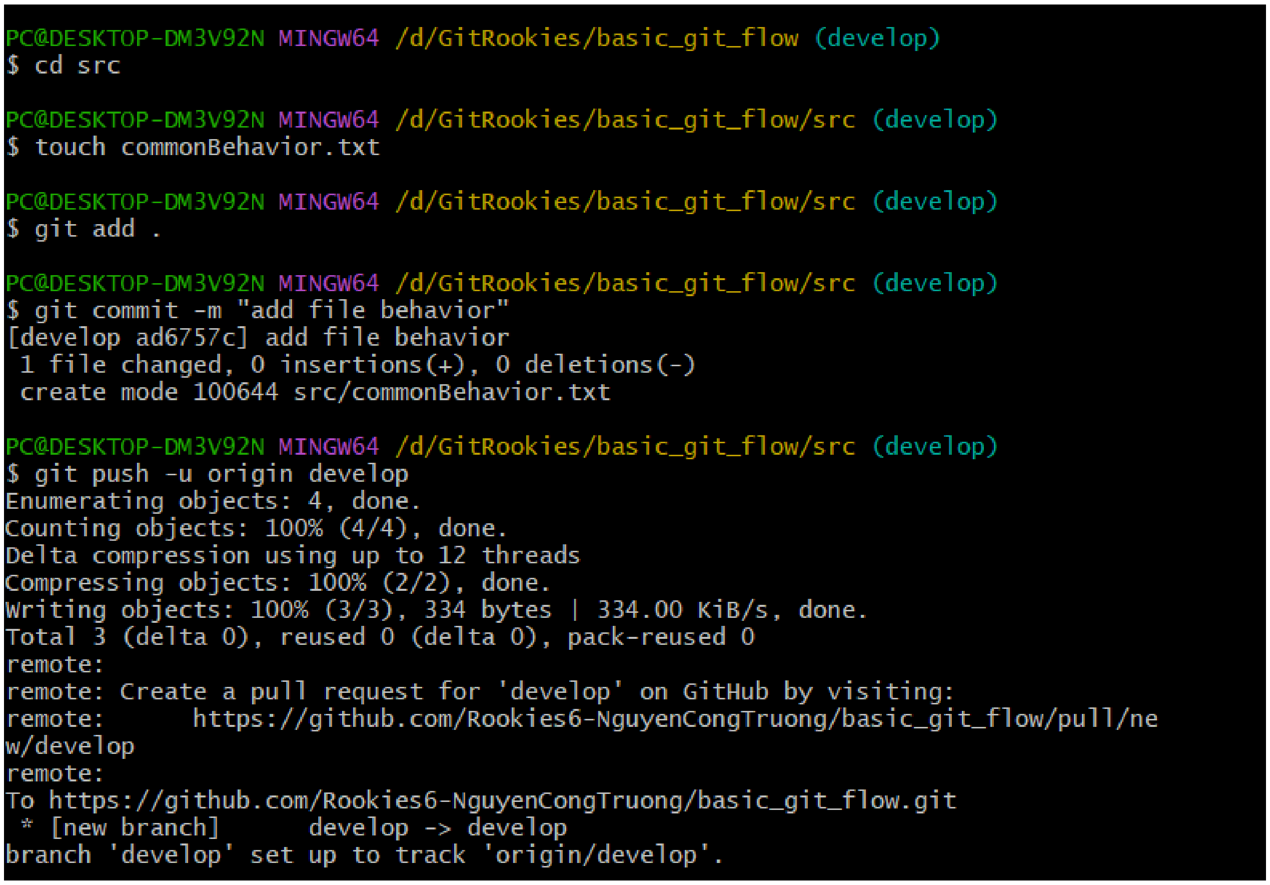
Answer:

❖ Clone the repository, create new branch and make folder src



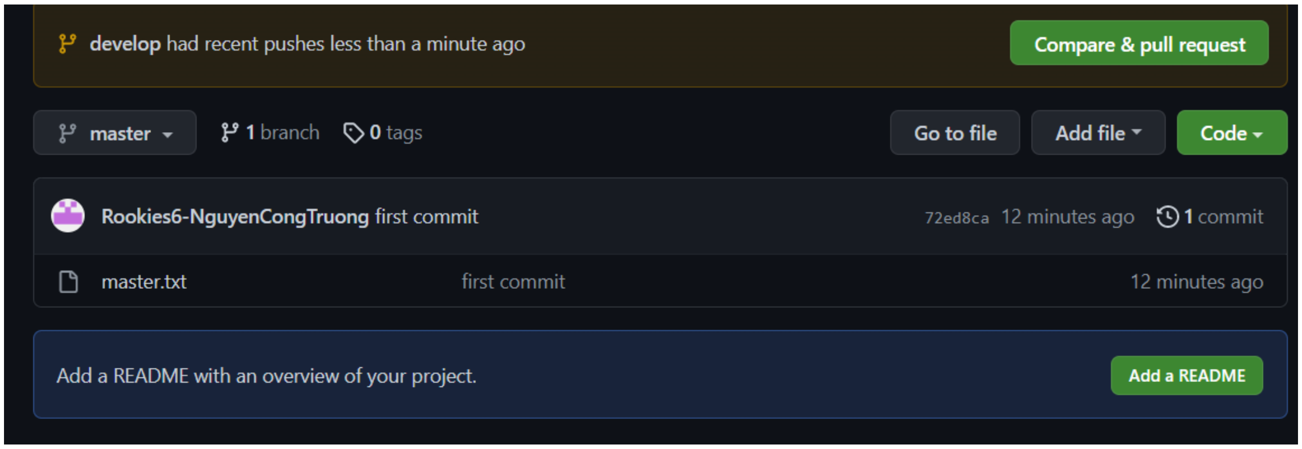
```
PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies
$ git clone https://github.com/Rookies6-NguyenCongTruong/basic_git_flow.git
Cloning into 'basic_git_flow'...
warning: You appear to have cloned an empty repository.

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies
$ git add .
fatal: not a git repository (or any of the parent directories): .git

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies
$ touch master.txt

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies
$ cd basic_git_flow

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow (master)
$ git add .

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow (master)
$ touch master.txt

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow (master)
$ git add .

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow (master)
$ git commit -m "first commit"
[master (root-commit) 72ed8ca] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 master.txt

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow (master)
$ git remote
origin

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 222 bytes | 222.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Rookies6-NguyenCongTruong/basic_git_flow.git
 * [new branch]      master -> master

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow (master)
$ git branch develop

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow (master)
$ git checkout develop
Switched to branch 'develop'

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow (develop)
$ mkdir src
```

❖ Create file, stage file, commit and push -u (upstream)



```
PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow (develop)
$ cd src

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (develop)
$ touch commonBehavior.txt

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (develop)
$ git add .

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (develop)
$ git commit -m "add file behavior"
[develop ad6757c] add file behavior
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 src/commonBehavior.txt

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (develop)
$ git push -u origin develop
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 334 bytes | 334.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:      https://github.com/Rookies6-NguyenCongTruong/basic_git_flow/pull/ne
w/develop
remote:
To https://github.com/Rookies6-NguyenCongTruong/basic_git_flow.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.
```

❖ The branch being push on GitHub



develop had recent pushes less than a minute ago

Rookies6-NguyenCongTruong first commit 72ed8ca 12 minutes ago 1 commit

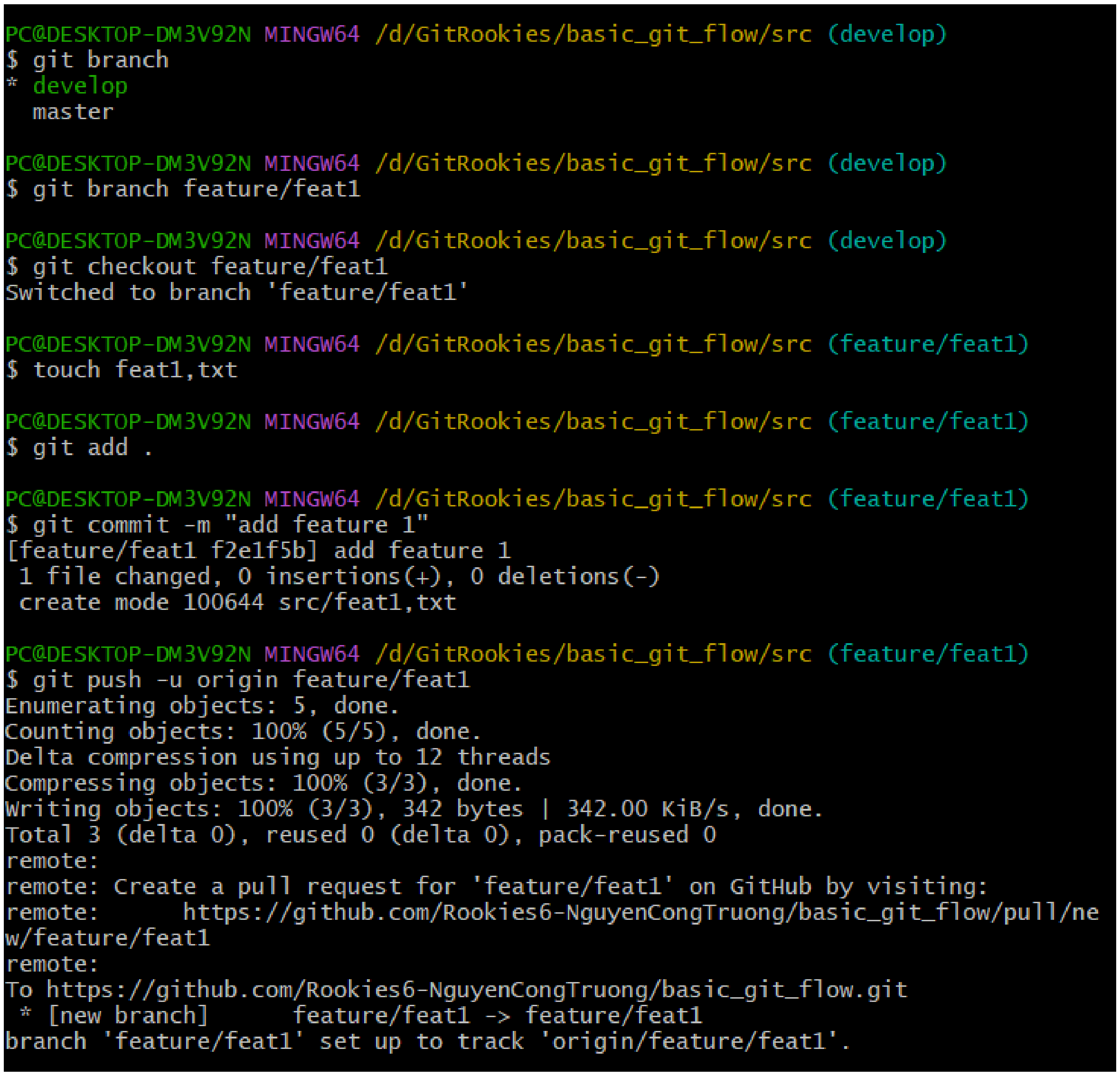
master.txt first commit 12 minutes ago

Add a README with an overview of your project.

Q3. At Sprint 0, DevA must do task which is "feat1", then DevA creates new branch named "feature/feat1" based on "develop" branch and add file "feat1" to "src" folder. After finishing this task, DevA creates a pull request to merge code from "feature/feat1" into "develop" branch. Write Git commands to do the requirement then go to Github merge the PR.
(hints: as same as Q2 hints and what is git pull request command)

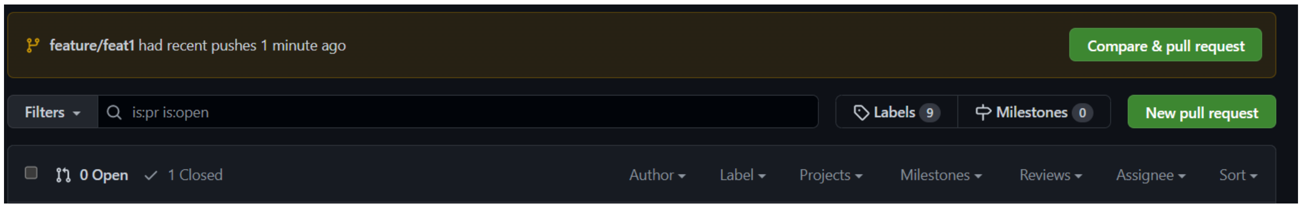
Answer:

- ❖ Create branch feature/feat1, add file to src of that branch, push to upstream to GitHub



```
PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (develop)
$ git branch
* develop
  master

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (develop)
$ git branch feature/feat1

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (develop)
$ git checkout feature/feat1
Switched to branch 'feature/feat1'

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat1)
$ touch feat1,txt

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat1)
$ git add .

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat1)
$ git commit -m "add feature 1"
[feature/feat1 f2e1f5b] add feature 1
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 src/feat1,txt

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat1)
$ git push -u origin feature/feat1
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 342 bytes | 342.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature/feat1' on GitHub by visiting:
remote:      https://github.com/Rookies6-NguyenCongTruong/basic_git_flow/pull/ne
w/feature/feat1
remote:
To https://github.com/Rookies6-NguyenCongTruong/basic_git_flow.git
 * [new branch]      feature/feat1 -> feature/feat1
branch 'feature/feat1' set up to track 'origin/feature/feat1'.
```

- ❖ Pull request


feature/feat1 had recent pushes 1 minute ago

Compare & pull request

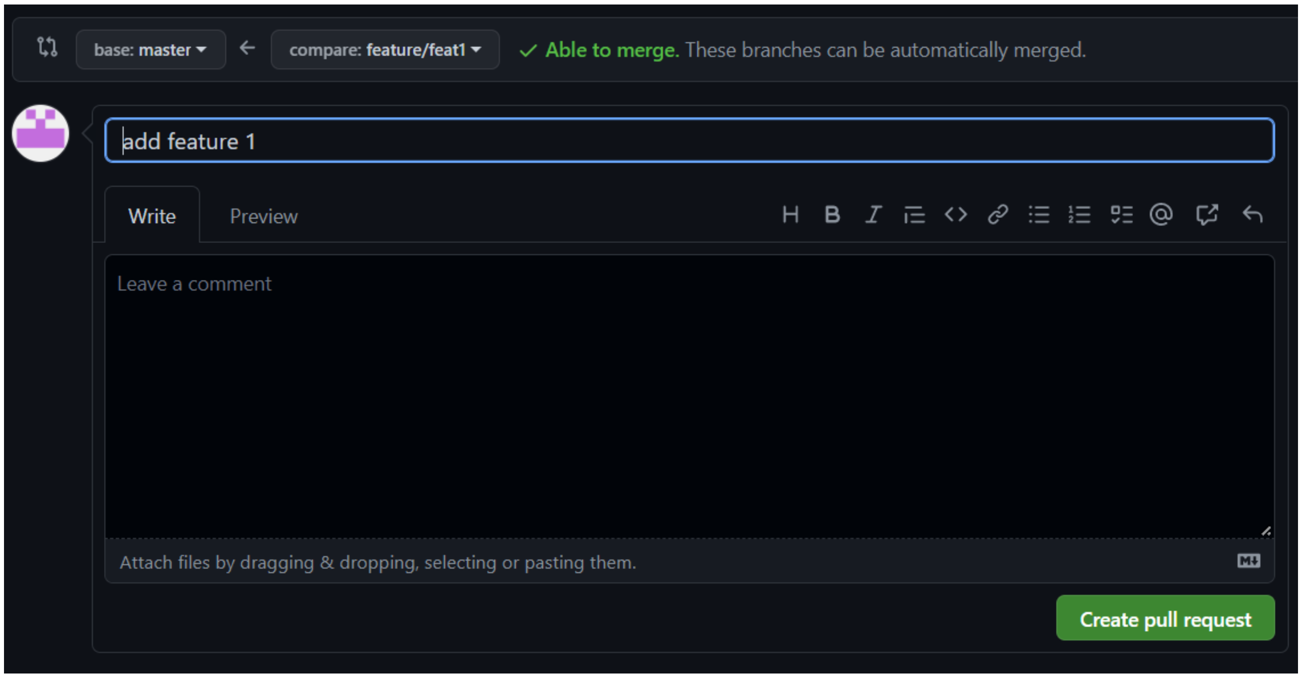
Filters

is:pr is:open

Labels 9 · Milestones 0 · New pull request

0 Open · 1 Closed

Author · Label · Projects · Milestones · Reviews · Assignee · Sort


base: master ← compare: feature/feat1 ✓ Able to merge. These branches can be automatically merged.

add feature 1

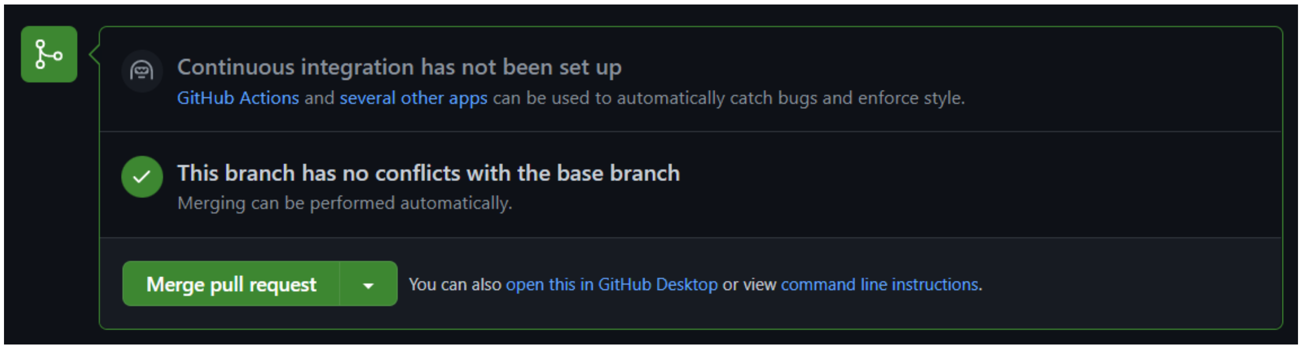
Write · Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

- Merge pull request


Continuous integration has not been set up

GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch

Merging can be performed automatically.

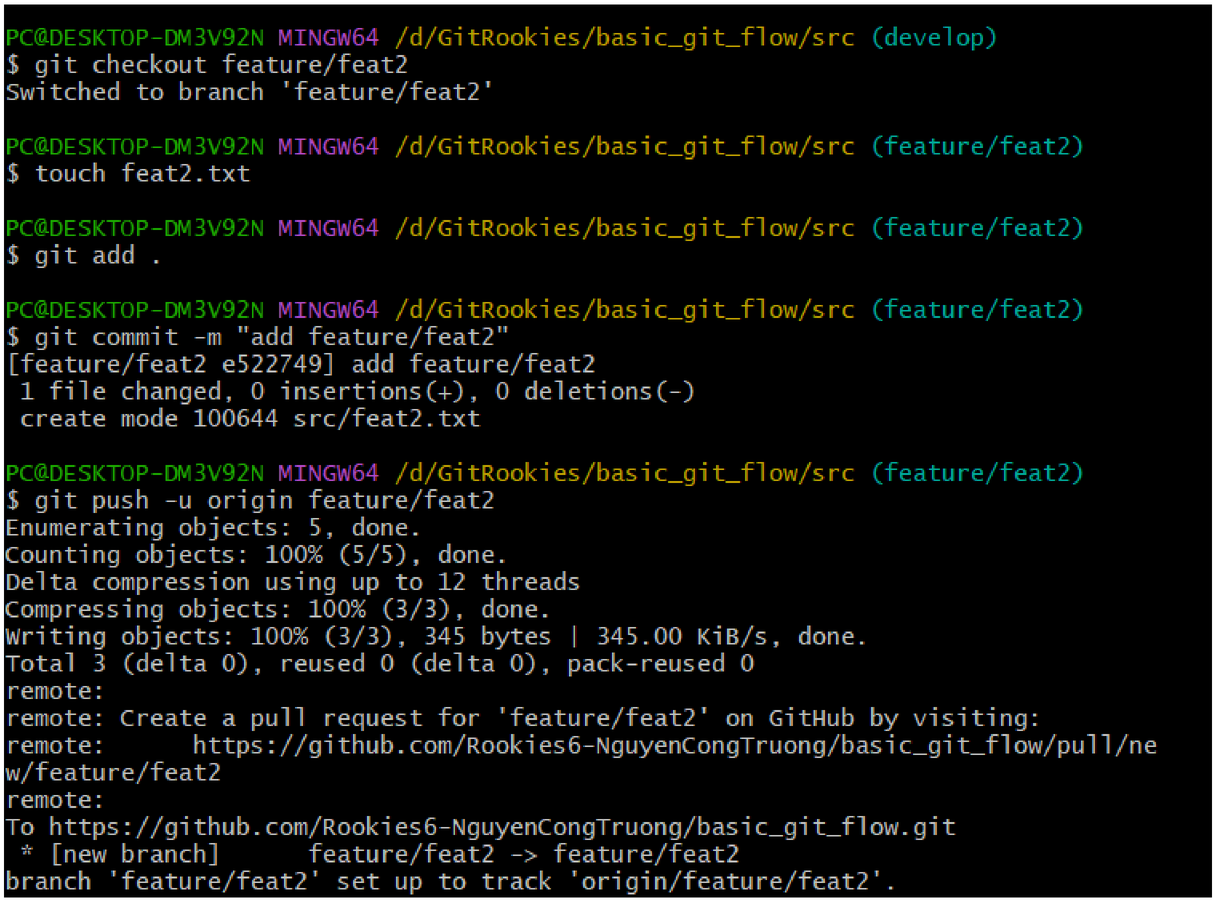
Merge pull request

You can also open this in GitHub Desktop or view command line instructions.

Q4. At Sprint 0, DevC and DevD must do task which is “feat2”. DevC creates new branch named “feature/feat2” based on “develop” branch and add file “feat2” to “src” folder. DevC and DevD work on the same file “feat2”. After finishing this task, DevD creates a pull request to merge code from “feature/feat2” into “develop” branch. Write Git commands to do the requirement (include commands to use merge tools, assumes that conflicts are automatically merge).

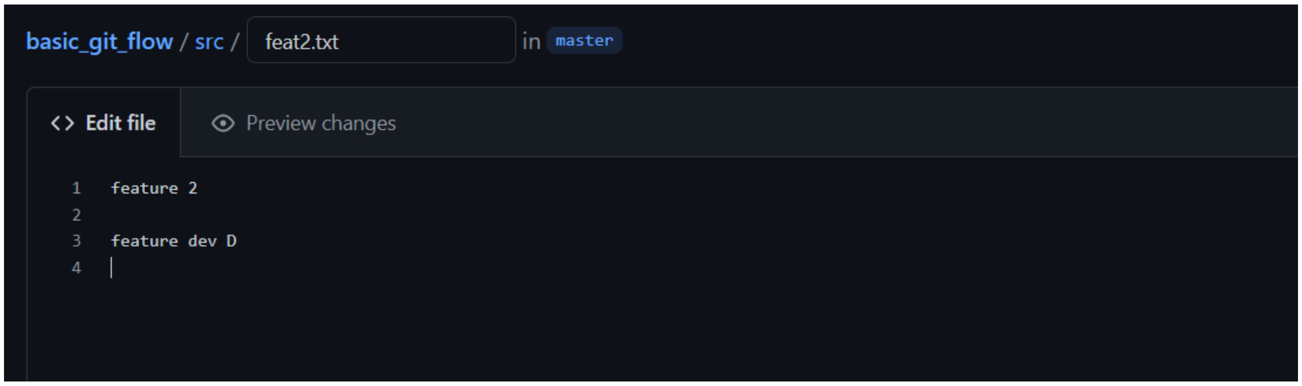
(hints: how to decrease the conflicts when DevC/DevD commit changes on the same file/branch or what they do before each committing changes)
Answer:

- ❖ Create branch feature/feat2, add file feat2 to src of that branch, push to upstream to GitHub



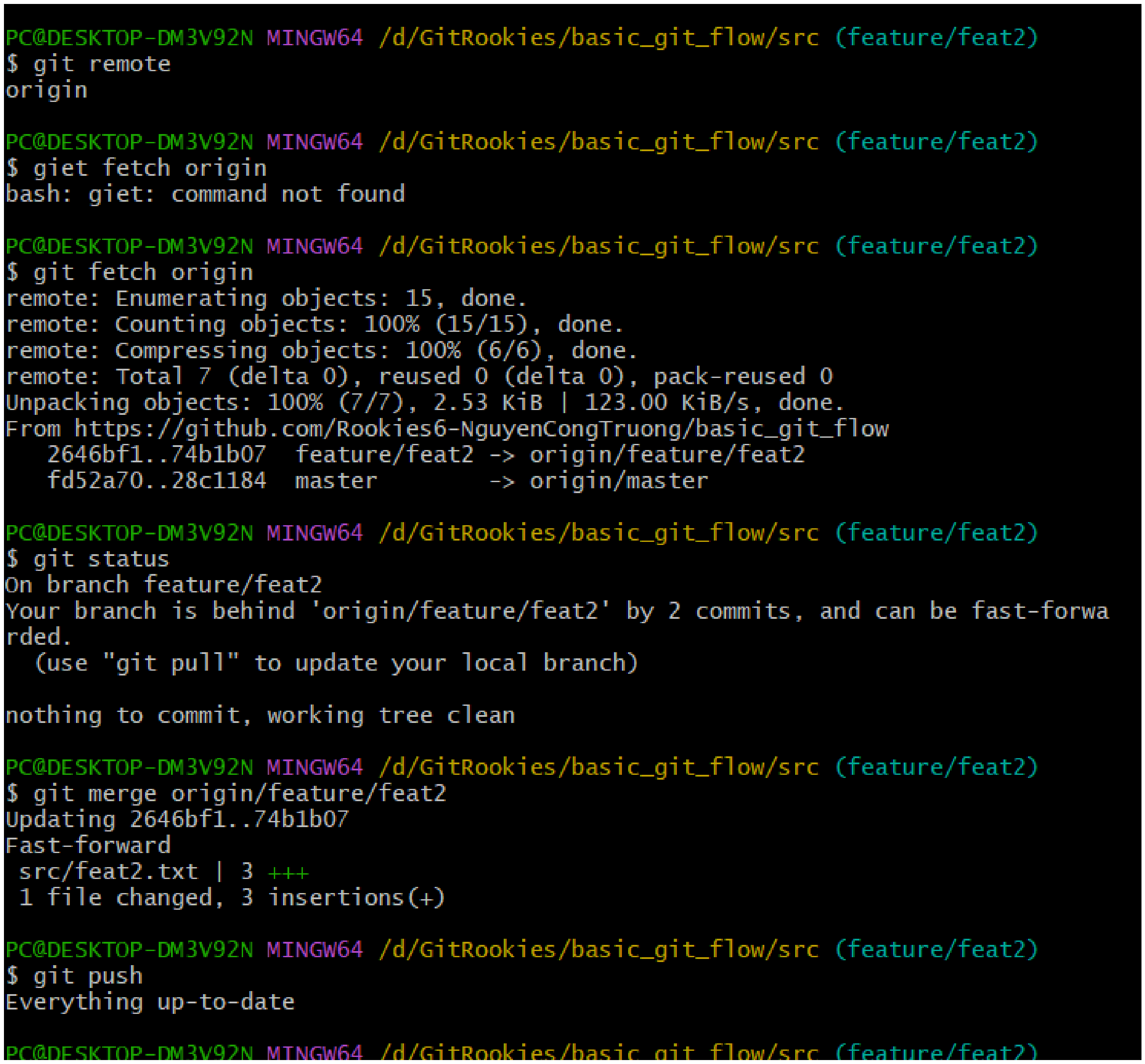
```
PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (develop)
$ git checkout feature/feat2
Switched to branch 'feature/feat2'

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat2)
$ touch feat2.txt

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat2)
$ git add .

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat2)
$ git commit -m "add feature/feat2"
[feature/feat2 e522749] add feature/feat2
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 src/feat2.txt

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat2)
$ git push -u origin feature/feat2
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 345 bytes | 345.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature/feat2' on GitHub by visiting:
remote:      https://github.com/Rookies6-NguyenCongTruong/basic_git_flow/pull/ne
w/feature/feat2
remote:
To https://github.com/Rookies6-NguyenCongTruong/basic_git_flow.git
 * [new branch]      feature/feat2 -> feature/feat2
branch 'feature/feat2' set up to track 'origin/feature/feat2'.
```

- ❖ Edit file feat2 to cause conflict on github



```
feature 2

feature dev D
```

❖ Resolve conflicts and push back

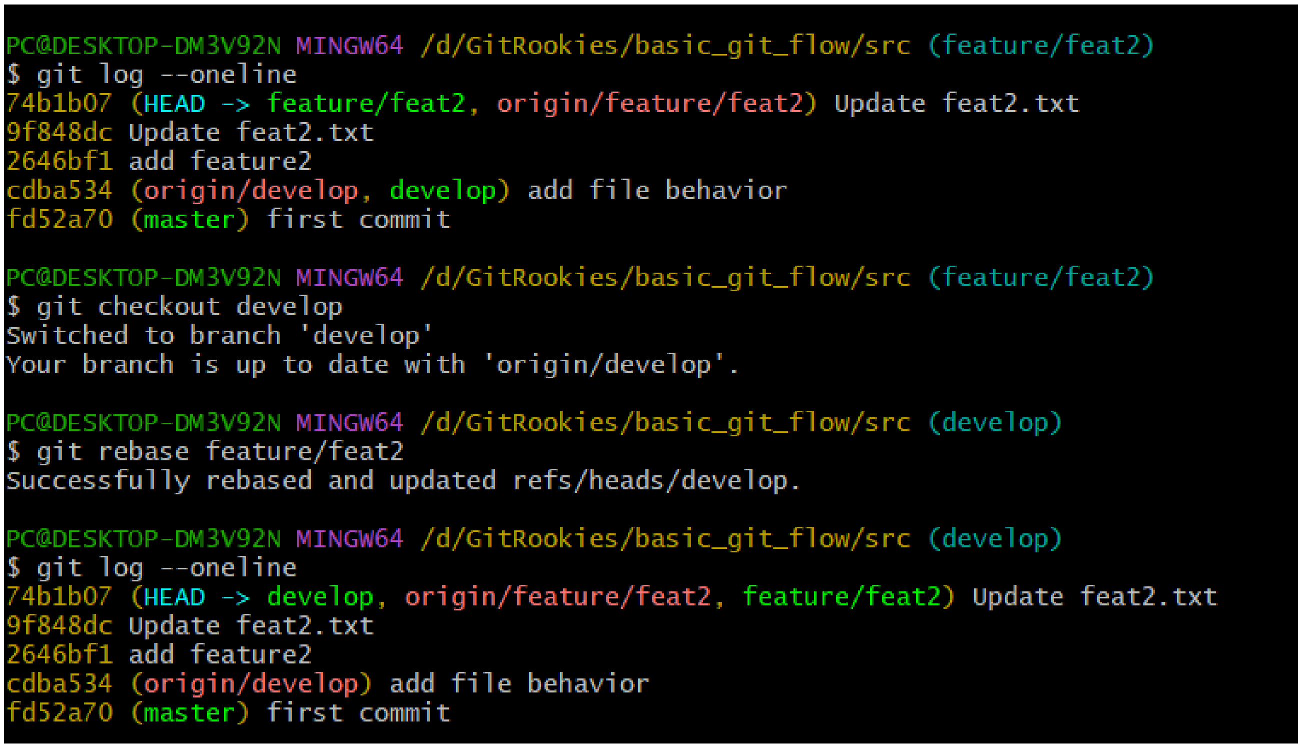


```
PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat2)
$ git remote
origin

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat2)
$ giet fetch origin
bash: giet: command not found

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat2)
$ git fetch origin
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), 2.53 KiB | 123.00 KiB/s, done.
From https://github.com/Rookies6-NguyenCongTruong/basic_git_flow
   2646bf1..74b1b07  feature/feat2 -> origin/feature/feat2
   fd52a70..28c1184  master        -> origin/master

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat2)
$ git status
On branch feature/feat2
Your branch is behind 'origin/feature/feat2' by 2 commits, and can be fast-forwa
rded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat2)
$ git merge origin/feature/feat2
Updating 2646bf1..74b1b07
Fast-forward
 src/feat2.txt | 3 +++
 1 file changed, 3 insertions(+)

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat2)
$ git push
Everything up-to-date

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat2)
```

❖ How to decrease the conflicts when DevC/DevD commit changes on the same file/branch or what they do before each committing changes
- They should ask, communicate to each other from what part they going to change in the code if they are working in the same branch.
- Don’t change thing that can easily cause conflict like import …

Q5. At sprint 0, after DevD create a pull request to merge code from "feature/feat2" into "develop". TL review the PR and see that there are many commits on "feature/feat2". TL wants to do "rebase" commits on this branch before merging PR. Write Git commands to do the requirement then go to Github to merge the PR. (hints: make 5 - 8 commit changes, git rebase interactive/squash on one branch, force push after finishing rebase process).

Answer:



```
PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat2)
$ git log --oneline
74b1b07 (HEAD -> feature/feat2, origin/feature/feat2) Update feat2.txt
9f848dc Update feat2.txt
2646bf1 add feature2
cdba534 (origin/develop, develop) add file behavior
fd52a70 (master) first commit

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (feature/feat2)
$ git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (develop)
$ git rebase feature/feat2
Successfully rebased and updated refs/heads/develop.

PC@DESKTOP-DM3V92N MINGW64 /d/GitRookies/basic_git_flow/src (develop)
$ git log --oneline
74b1b07 (HEAD -> develop, origin/feature/feat2, feature/feat2) Update feat2.txt
9f848dc Update feat2.txt
2646bf1 add feature2
cdba534 (origin/develop) add file behavior
fd52a70 (master) first commit
```

Q6. At sprint 1, DevA must do task which is "feat3". DevA creates new branch named "feature/feat3" based on "develop" branch and add file "feat3" to "src" folder. At the same time, DevB do task to add some extra behavior to project. DevB creates new branch named "feature/ex_behavior" based on "develop" branch and add file "extraBehavior" to "src" folder. After some days, DevA and Dev realize that "feature/feat3" can re-use code of "feature/ex_behavior". They accept merge code from "feature/ex_behavior" into "feature/feat3" and only make PR to merge code from "feature/feat3" to "develop". Write Git commands to do the requirement and at the part of [merge code from "feature/ex_behavior" into "feature/feat3"], provide 2 ways of merging branches (git merge and git rebase).

(hints: git merge branches, git rebase onto, force push after finishing rebase process)

Answer:

Q7. After n-1 sprint, team must create a release branch to go live some features. TL create release branch named “release/r1” based on “develop”. For uat/production environments, team must update some log message, then team adds “extraLogMessage” file to “src” folder. As your understanding of Git flow (section 1), write Git commands to ensure that release branch and its changes are merged into develop/master.

Answer: