**SWE 525 Git Version Control Assignment 1 8/10/16**

A. Answer following questions briefly:

1. List out the key difference between a centralized version control system and distributed version control system

**Answer:**

**Centralized Version Control System**:

Provides a server software component which stores and manages the different versions of the files and let developer copy(checkout) a certain version onto their individual computer.

There is a single point of failure in centralized version control system. The server machine is a single point of failure here.

It gets harder to work in parallel on different features.

**Distributed Version Control system**:

Each user has a complete copy of a repository on his individual computer. The user can copy an existing repository.

There is a central server for keeping a repository but each cloned repository is a full copy of this repository. There is no problem of a single system failure since a copy of the repository is saved locally and uploaded only what is required.

Distributed version control system makes it easier to work in parallel on different features.

1. List down any two centralized version control system and 2 distributed version control system

**Answer**:

Two Centralized Version control system: Perforce, Subversion

Two Distributed Version Control system: git, mercurial.

1. What are the advantages of git VCS over other VCS

**Answer**:

The main difference between Git and any other VCS is the way Git thinks about its data. Git thinks of its data more like set of snapshots of a miniature filesystem.

Git maintains a set of configuration values within each repository. Git manages and inspects configuration and setup information on a per-site, per-user, and per-repository basis.

1. What are the different states of a file in the Git VCS

Committed, modified and staged. Committed means, that the data is safely stored in your local database. Modified means that you have changed the file but have not committed it to your database yet. Staged means that you have marked a modified file in its current version to go into your next commit snapshot.

**B. GIT REMOTE REPOSITORIES:** Perform following tasks and explain how you performed each operation. Draw a flow diagram as you progress through the steps. Add all git commands you used and push the repository in your github. Add your github public repository (for these following tasks) link with the homework.

https://github.com/Supriyankumar/Assignment1\_git.git

1. Clone an existing repository on Github created during course and configure your local repo to point to the remote repository

Answer:

Used git clone to clone the existing repository on Github.

Last login: Sat Aug 20 08:04:09 on console

SUPRIYAs-MacBook-Pro:~ suppi$ git --version

git version 1.7.10.2 (Apple Git-33)

SUPRIYAs-MacBook-Pro:~ suppi$ cd SWE525

-bash: cd: SWE525: No such file or directory

SUPRIYAs-MacBook-Pro:~ suppi$ cd desktop

SUPRIYAs-MacBook-Pro:desktop suppi$ cd SWE525

SUPRIYAs-MacBook-Pro:SWE525 suppi$ ls

SUPRIYAs-MacBook-Pro:SWE525 suppi$ git init

Initialized empty Git repository in /Users/suppi/Desktop/SWE525/.git/

SUPRIYAs-MacBook-Pro:SWE525 suppi$ ls

SUPRIYAs-MacBook-Pro:SWE525 suppi$ ls -a

. .. .git

SUPRIYAs-MacBook-Pro:SWE525 suppi$ git status

# On branch master

#

# Initial commit

#

nothing to commit (create/copy files and use "git add" to track)

SUPRIYAs-MacBook-Pro:SWE525 suppi$ git clone https://github.com/treehouse/treehouse\_android\_utilities.git

Cloning into 'treehouse\_android\_utilities'...

remote: Counting objects: 21, done.

remote: Total 21 (delta 0), reused 0 (delta 0), pack-reused 21

Unpacking objects: 100% (21/21), done.

SUPRIYAs-MacBook-Pro:SWE525 suppi$ ls

treehouse\_android\_utilities

Using git remote to view the existing repos

SUPRIYAs-MacBook-Pro:SWE525 suppi$ git clone https://github.com/treehouse/treehouse\_android\_utilities.git

Cloning into 'treehouse\_android\_utilities'...

remote: Counting objects: 21, done.

remote: Total 21 (delta 0), reused 0 (delta 0), pack-reused 21

Unpacking objects: 100% (21/21), done.

SUPRIYAs-MacBook-Pro:SWE525 suppi$ ls

treehouse\_android\_utilities

SUPRIYAs-MacBook-Pro:SWE525 suppi$ git remote

SUPRIYAs-MacBook-Pro:SWE525 suppi$ ls

treehouse\_android\_utilities

SUPRIYAs-MacBook-Pro:SWE525 suppi$ cd treehouse\_android\_utilities

SUPRIYAs-MacBook-Pro:treehouse\_android\_utilities suppi$ git remote

origin

SUPRIYAs-MacBook-Pro:treehouse\_android\_utilities suppi$ git remote -v

origin https://github.com/treehouse/treehouse\_android\_utilities.git (fetch)

origin https://github.com/treehouse/treehouse\_android\_utilities.git (push)

SUPRIYAs-MacBook-Pro:treehouse\_android\_utilities suppi$

The above snapshot of the bash shell shows the remote repository linked to the local. Git remote shows the origin in the local repo.

1. Perform some operation like add, remove, modify and finally push your changes to the remote repository

SUPRIYAs-MacBook-Pro:SWE525 suppi$ git clone https://github.com/Supriyankumar/Treehouse.git

Cloning into 'Treehouse'...

warning: You appear to have cloned an empty repository.

SUPRIYAs-MacBook-Pro:SWE525 suppi$ ls

Assignment1-Git treehouse\_android\_utilities

Treehouse

SUPRIYAs-MacBook-Pro:SWE525 suppi$ cd Treehouse

SUPRIYAs-MacBook-Pro:Treehouse suppi$ echo "# Treehouse" >> README.md

SUPRIYAs-MacBook-Pro:Treehouse suppi$ ls

README.md

SUPRIYAs-MacBook-Pro:Treehouse suppi$ git add README.md

SUPRIYAs-MacBook-Pro:Treehouse suppi$ git status

# On branch master

#

# Initial commit

#

# Changes to be committed:

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

#

# new file: README.md

#

SUPRIYAs-MacBook-Pro:Treehouse suppi$ git commit -m "new README addition"

[master (root-commit) 050ac52] new README addition

1 file changed, 1 insertion(+)

create mode 100644 README.md

SUPRIYAs-MacBook-Pro:Treehouse suppi$ git status

# On branch master

nothing to commit (working directory clean)

SUPRIYAs-MacBook-Pro:Treehouse suppi$ git remote

origin

SUPRIYAs-MacBook-Pro:Treehouse suppi$ git push -u origin master

Username for 'https://github.com': 89706.kumar@students.itu.edu

Password for 'https://89706.kumar@students.itu.edu@github.com':

Counting objects: 3, done.

Writing objects: 100% (3/3), 239 bytes, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/Supriyankumar/Treehouse.git

\* [new branch] master -> master

Branch master set up to track remote branch master from origin.

SUPRIYAs-MacBook-Pro:Treehouse suppi$ cd SWE525

-bash: cd: SWE525: No such file or directory

3.Pull the latest changes from the repository to get the updates from others in to your local repo and merge the changes

Answer: Used git pull origin : automatically updated and

SUPRIYAs-MacBook-Pro:desktop suppi$ cd SWE525

SUPRIYAs-MacBook-Pro:SWE525 suppi$ cd Treehouse

SUPRIYAs-MacBook-Pro:Treehouse suppi$ git pull origin

remote: Counting objects: 3, done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

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

From https://github.com/Supriyankumar/Treehouse

050ac52..1f40e22 master -> origin/master

Updating 050ac52..1f40e22

Fast-forward

README.md | 1 +

1 file changed, 1 insertion(+)

SUPRIYAs-MacBook-Pro:Treehouse suppi$

SUPRIYAs-MacBook-Pro:Assignment1-Git suppi$ git fetch origin

remote: Counting objects: 3, done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

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

From https://github.com/Supriyankumar/Assignment1-Git

3fd725e..d55139a master -> origin/master

SUPRIYAs-MacBook-Pro:Assignment1-Git suppi$

SUPRIYAs-MacBook-Pro:Assignment1-Git suppi$ git merge origin/master

Updating 3fd725e..d55139a

Fast-forward

README.md | 2 ++

1 file changed, 2 insertions(+)

create mode 100644 README.md

SUPRIYAs-MacBook-Pro:Assignment1-Git suppi$

1. Try fetching the changes and perform the merge to get the difference between the pull and the merge command

Solution:

unknown20c9d0d31df9:Assignment1\_git suppi$ git pull origin

remote: Counting objects: 3, done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0

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

From https://github.com/Supriyankumar/Assignment1\_git

ec20ca8..8f75570 master -> origin/master

Updating ec20ca8..8f75570

Fast-forward

fetch.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 fetch.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git fetch origin

remote: Counting objects: 3, done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0

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

From https://github.com/Supriyankumar/Assignment1\_git

8f75570..c1c5867 master -> origin/master

unknown20c9d0d31df9:Assignment1\_git suppi$ git merge

fatal: No commit specified and merge.defaultToUpstream not set.

unknown20c9d0d31df9:Assignment1\_git suppi$ git merge origin

fatal: origin - not something we can merge

unknown20c9d0d31df9:Assignment1\_git suppi$ git fetch origin

unknown20c9d0d31df9:Assignment1\_git suppi$ git merge origin/master

Updating 8f75570..c1c5867

Fast-forward

pull.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 pull.txt

1. Perform some changes and before committing the changes, stash your changes and then pull the changes and finally apply the changes to understand how stashing works
2. Create a feature branch and do some file operations in the branch and commit the changes to the branch

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout feature

Switched to branch 'feature'

unknown20c9d0d31df9:Assignment1\_git suppi$ git branch feature

fatal: A branch named 'feature' already exists.

unknown20c9d0d31df9:Assignment1\_git suppi$ echo "#this is the file to be merged" >> newfile.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git add newfile.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git commit -m "some file"

[feature 81fcfa6] some file

1 file changed, 1 insertion(+)

create mode 100644 newfile.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git status

# On branch feature

nothing to commit (working directory clean)

1. Merge the changes using the rebase command and finally perform a safe deletion of the feature branch

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout master

Switched to branch 'master'

unknown20c9d0d31df9:Assignment1\_git suppi$ echo "rebase this to rebasetest later" > rebasefile.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git add rebasefile.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git commit -m "new rebase file"

[master f661229] new rebase file

1 file changed, 1 insertion(+)

create mode 100644 rebasefile.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout feature

Switched to branch 'feature'

unknown20c9d0d31df9:Assignment1\_git suppi$ git rebase master

First, rewinding head to replay your work on top of it...

Applying: some file

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout master

Switched to branch 'master'

Your branch is ahead of 'origin/master' by 1 commit.

unknown20c9d0d31df9:Assignment1\_git suppi$ git merge feature

Updating f661229..fd9063f

Fast-forward

newfile.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 newfile.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git branch -D feature

Deleted branch feature (was fd9063f).

1. Create another feature branch and this time after committing the changes to the feature branch, merge the changes using fast forward merge and then delete the feature branch

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout -b feature

Switched to a new branch 'feature'

unknown20c9d0d31df9:Assignment1\_git suppi$ echo "creating one new file" >> new\_file.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git add new\_file.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git commit -m "new file on branch"

[feature ec20ca8] new file on branch

1 file changed, 1 insertion(+)

create mode 100644 new\_file.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git status

# On branch feature

nothing to commit (working directory clean)

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout master

Switched to branch 'master'

Your branch is ahead of 'origin/master' by 2 commits.

unknown20c9d0d31df9:Assignment1\_git suppi$ git merge feature

Updating fd9063f..ec20ca8

Fast-forward

new\_file.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 new\_file.txt

**C. GIT BRANCHING AND MERGING:** Perform following tasks and explain how you performed each operation. Draw a flow diagram as you progress through the steps. Add all git commands you used and push the repository in your github. Add your github public repository (for these following tasks) link with the homework.

1. Create a local branch using git checkout -b branchname command

Solution:

unknown20c9d0d31df9:SWE525 suppi$ git clone https://github.com/Supriyankumar/Assignment1\_git.git

Cloning into 'Assignment1\_git'...

warning: You appear to have cloned an empty repository.

unknown20c9d0d31df9:SWE525 suppi$ echo "# This is the solutions repo" >> README.md

unknown20c9d0d31df9:Assignment1\_git suppi$ git branch

\* master

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout -b feature

Switched to a new branch 'feature'

unknown20c9d0d31df9:Assignment1\_git suppi$ git branch

\* feature

master

unknown20c9d0d31df9:Assignment1\_git suppi$ echo "this is the branch file" >> file1.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ ls

README.md file1.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git add file1.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git commit -m "one new file1"

[feature 3b508fd] one new file1

1 file changed, 1 insertion(+)

create mode 100644 file1.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git log

commit 3b508fde2d5a8ceac53953bd2ad934241a985ded

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 28 22:23:19 2016 -0700

one new file1

commit f1c3aa30a21b907a3ad90a039dca42895daa8a30

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 28 22:18:53 2016 -0700

readme file

commit 8efab29ce1c3128136b35866b6518fc6ddeaf627

Author: Supriyankumar <89706.kumar@students.itu.edu>

Date: Sun Aug 28 22:17:34 2016 -0700

new line addition

1. Observe the difference by doing some file operations and switch back to the master branch and see if you can see the changes done on the branch

Solution:

unknown20c9d0d31df9:Assignment1\_git suppi$ git branch

\* feature

master

unknown20c9d0d31df9:Assignment1\_git suppi$ echo "this is the branch file" >> file1.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ ls

README.md file1.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git add file1.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git commit -m "one new file1"

[feature 3b508fd] one new file1

1 file changed, 1 insertion(+)

create mode 100644 file1.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git log

commit 3b508fde2d5a8ceac53953bd2ad934241a985ded

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 28 22:23:19 2016 -0700

one new file1

commit f1c3aa30a21b907a3ad90a039dca42895daa8a30

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 28 22:18:53 2016 -0700

readme file

commit 8efab29ce1c3128136b35866b6518fc6ddeaf627

Author: Supriyankumar <89706.kumar@students.itu.edu>

Date: Sun Aug 28 22:17:34 2016 -0700

new line addition

unknown20c9d0d31df9:Assignment1\_git suppi$ git branch

\* feature

master

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout master

Switched to branch 'master'

unknown20c9d0d31df9:Assignment1\_git suppi$ git branch

feature

\* master

1. Now switch back to the branch and commit the changes and switch to master

branch. Now see if you can still see the changes in the master branch

Solution:

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout feature

Switched to branch 'feature'

unknown20c9d0d31df9:Assignment1\_git suppi$ echo "This is the second line" >> file1.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git add file1.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git diff

unknown20c9d0d31df9:Assignment1\_git suppi$ git diff file1.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git commit -m "New line"

[feature d41c2b7] New line

1 file changed, 1 insertion(+)

unknown20c9d0d31df9:Assignment1\_git suppi$ git diff file1.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout master

Switched to branch 'master'

1. Now switch back to the branch name and stash the changes and apply the changes to the master branch by switching to the master branch

Solution:

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout feature

Switched to branch 'feature'

unknown20c9d0d31df9:Assignment1\_git suppi$ git stash

No local changes to save

unknown20c9d0d31df9:Assignment1\_git suppi$ git branch

\* feature

master

unknown20c9d0d31df9:Assignment1\_git suppi$ git commit -m "all files"

# On branch feature

# Untracked files:

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

#

# file2.txt

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

unknown20c9d0d31df9:Assignment1\_git suppi$ git add file2.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git diff file1.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git diff file2.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git commit -m "second file"

[feature bbbd28a] second file

1 file changed, 1 insertion(+)

create mode 100644 file2.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git stash

No local changes to save

unknown20c9d0d31df9:Assignment1\_git suppi$ git status

# On branch feature

nothing to commit (working directory clean)

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout master

Switched to branch 'master'

1. Try merging the changes from the branch to the master branch using all the three merge strategies and then view the git log

Solution:

unknown20c9d0d31df9:Assignment1\_git suppi$ git merge -s ours feature

Merge made by the 'ours' strategy.

unknown20c9d0d31df9:Assignment1\_git suppi$ git log

commit 42481668a06de8466eb402aec695c846a242796a

Merge: f1c3aa3 bbbd28a

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 28 22:54:20 2016 -0700

Merge branch 'feature'

commit bbbd28af40381d33c68bb7fca2af5bf7a541a59b

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 28 22:50:05 2016 -0700

second file

commit d41c2b739d32c8991c1e2654507516e62c40bd18

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 28 22:32:42 2016 -0700

New line

commit 3b508fde2d5a8ceac53953bd2ad934241a985ded

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 28 22:23:19 2016 -0700

unknown20c9d0d31df9:Assignment1\_git suppi$ git branch feature1

unknown20c9d0d31df9:Assignment1\_git suppi$ git branch

feature

feature1

\* master

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout -b feature1

fatal: A branch named 'feature1' already exists.

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout feature1

Switched to branch 'feature1'

unknown20c9d0d31df9:Assignment1\_git suppi$ echo "This isthe new file on new branch" >> test\_file.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git add test\_file.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git commit -m "one new file "

[feature1 9df615b] one new file

1 file changed, 1 insertion(+)

create mode 100644 test\_file.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git status

# On branch feature1

nothing to commit (working directory clean)

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout master

Switched to branch 'master'

Your branch is ahead of 'origin/master' by 4 commits.

unknown20c9d0d31df9:Assignment1\_git suppi$ git merge feature feature1

Updating 4248166..9df615b

Fast-forward

test\_file.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 test\_file.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout feature2

Switched to branch 'feature2'

unknown20c9d0d31df9:Assignment1\_git suppi$ echo "third file" >> file3.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git add file3.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ echo "new line added" >> file3.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git add file3.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git diff file3.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git commit -m "third file"

[feature2 dcd43fa] third file

1 file changed, 2 insertions(+)

create mode 100644 file3.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git diff file3.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git status

# On branch feature2

nothing to commit (working directory clean)

unknown20c9d0d31df9:Assignment1\_git suppi$ git checkout master

Switched to branch 'master'

Your branch is ahead of 'origin/master' by 5 commits.

unknown20c9d0d31df9:Assignment1\_git suppi$ git merge -s recursive -X theirs feature2

Updating 9df615b..dcd43fa

Fast-forward

file3.txt | 2 ++

1 file changed, 2 insertions(+)

create mode 100644 file3.txt

unknown20c9d0d31df9:Assignment1\_git suppi$ git log

commit dcd43faf4fca55eaffa19e1c12d8c64aad57cfa2

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 28 23:07:55 2016 -0700

third file

commit 9df615b7d705ebbeb834245594eca9bae5c096bb

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 28 22:59:07 2016 -0700

one new file

commit 42481668a06de8466eb402aec695c846a242796a

Merge: f1c3aa3 bbbd28a

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 28 22:54:20 2016 -0700

Merge branch 'feature'

commit bbbd28af40381d33c68bb7fca2af5bf7a541a59b

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 28 22:50:05 2016 -0700

1. Push the local branch to the remote repository and see if the branch is present on the remote repository – Github

Solution:

unknown20c9d0d31df9:Assignment1\_git suppi$ git push origin

Username for 'https://github.com': 89706.kumar@students.itu.edu

Password for 'https://89706.kumar@students.itu.edu@github.com':

Counting objects: 12, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (7/7), done.

Writing objects: 100% (10/10), 1.10 KiB, done.

Total 10 (delta 0), reused 0 (delta 0)

To https://github.com/Supriyankumar/Assignment1\_git.git

d41c2b7..bbbd28a feature -> feature

f1c3aa3..dcd43fa master -> master

**Solve following Git Reciepe Problems: Please give explanation and git commands to accomplish the tasks assigned**

1. Committing a file removed with the standard rm command

Problem: Your repository is in a clean state and contains the removed.txt file. The file is committed. You want to remove the file using the standard $ rm command and then commit this modification into the repository.

Solution:

We consider the file “removed.txt” as an example.

1. Remove removed.txt file with $rm removed.txt
2. Check the status of the repository with $git status
3. # On branch master
4. # Changes not staged for commit:
5. # (use "git add/rm <file>..." to update what will be committed)
6. # (use "git checkout -- <file>..." to discard changes in working directory)
7. #
8. # deleted: removed.txt
9. #
10. no changes added to commit (use "git add" and/or "git commit -a")
11. The file is listed under Changes not staged for commit. Therefore, the file is unstaged.
12. 3. Check the simplified form of status. The output of $ git status -s will be:
13. \_D removed.txt
14. The state is denoted by two characters: a space and a letter D.
15. 4. Commit the changes with
16. $ git commit -a -m "Staging and committing removed file"
17. 5. Check the status with the $ git status -s command. The output is empty, therefore, the repository is clean. The working directory doesn't contain the removed.txt file.
18. Converting an unmodified file into an untracked file  Problem:  The repository is in a clean state and the working directory contains one file—untracked.txt. The file is unmodified. You want to convert it into an untracked state.

Solution:

Right now the repository is clean and the working directory contains one file—untracked.txt.

1. Remove the untracked.txt file with:

$ git rm --cached untracked.txt

2. Check the contents of the repository with the $ ls command. As you can see the file was

not deleted—it still exists in the working directory.

3. Check the status of the repository. The command

$ git status prints:

# On branch master

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# deleted: untracked.txt

#

# Untracked files:

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

#

# untracked.txt

The file is listed as both staged (Changes to be committed) and untracked (Untracked files).

4. Check the simplified form of the status. The output of

$ git status -s will be:

D\_ untracked.txt

?? untracked.txt

The file is listed twice: as D\_ and ??.

5. Commit the changes with $ git commit -m "Committing removed file"

6. Check the status with the $ git status -s command.

The output is: ?? untracked.txt

The file is not tracked anymore and is not included in the latest snapshot.

3.Committing in a wrong branch

You have just created a new revision only to find out that it should go to a different branch. If your modifications do not collide while switching branches you can easily forget to check out the appropriate branch before you commit. After creating the revision in the wrong branch you want to move it to its correct destination.

Solution:

Consider a file named m1.txt

$ echo A new text > m1.txt

Let’s suppose you intended this change to be committed in the master branch. The modification doesn’t collide

with the doc branch. You can checkout the doc branch without any problems using: $ git checkout doc. The checkout

command outputs a line that give you information about the changes:

M m1.txt

But it can be easily overlooked.

After some time, you are completely unaware that the current branch is not the master. You create the wrong

revision with:

$ git snapshot Recipe 5-8: a revision in a wrong branch

Now that you realized your failure, you want to move the revision to the master branch.

You can do it in two separate steps:

• Copy the revision from wrong branch to correct one.

• Remove the revision from wrong branch.

Before you proceed with the procedures given below verify that the new revision is present in doc branch and

absent from master branch. Here are the commands you need:

$ git log --oneline doc

$ git log --oneline master

Here is the procedure you need to copy the latest revision from the doc branch to the master branch:

1. Change the current branch to master with the $ git checkout master command.

2. Copy the tip revision from the doc branch to your current branch (which is the master)

with the $ git cherry-pick doc command.

As of now, the new revision is present in both branches. The commands $ git log --oneline -1 master and

$ git log --oneline -1 doc print the output shown in Listing 5-3. Although your actual hashes will be different,

you should notice that the two SHA-1 names of your commits are different.

The output of $ git log --oneline -1 for the two branches

# the output of $ git log --oneline -1 master

43336a3 Recipe 5-8: a revision in a wrong branch

# the output of $ git log --oneline -1 doc

7ad4187 Recipe 5-8: a revision in a wrong branch

1. Change the current branch to doc with the $ git checkout doc command.

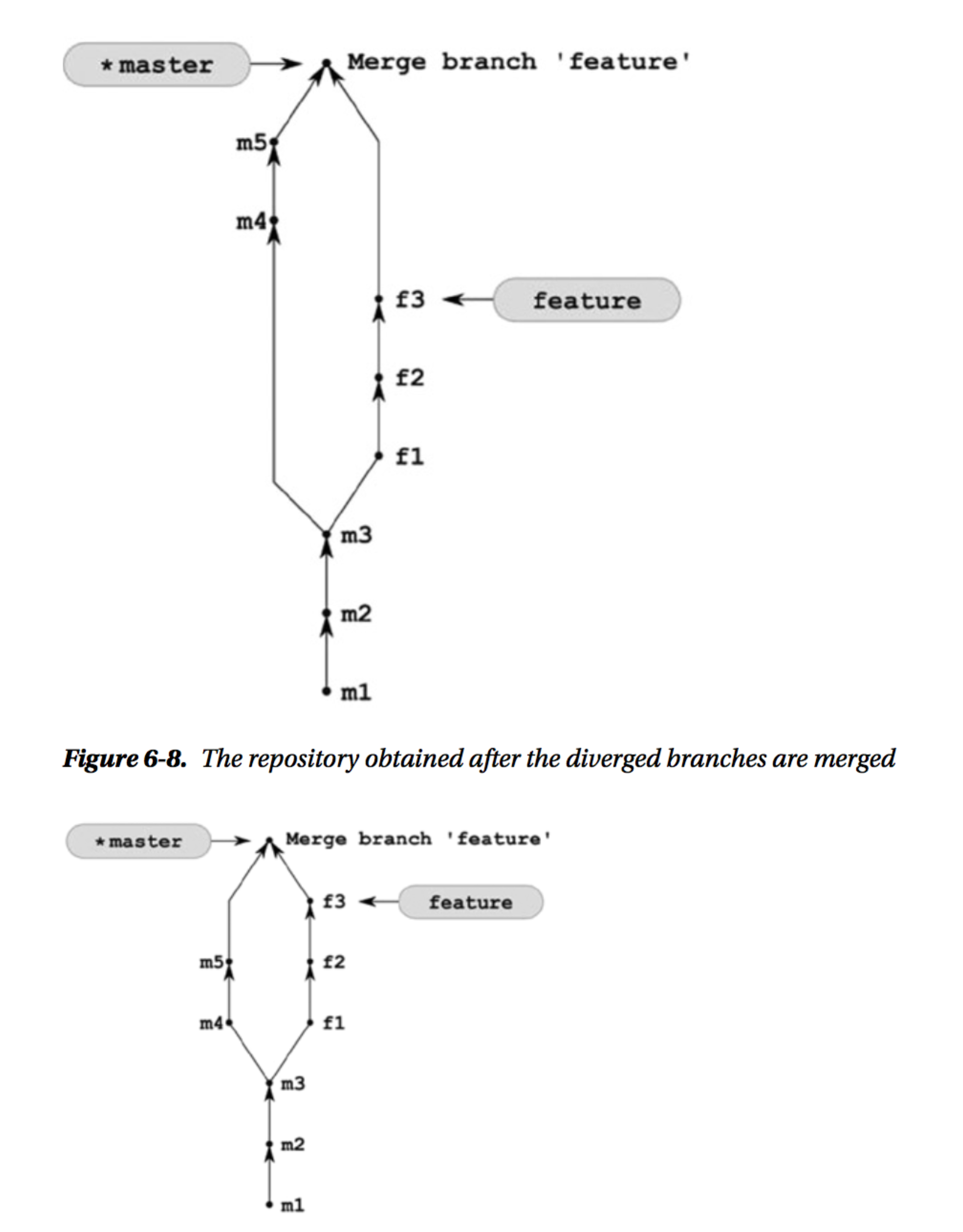
2. Remove the revision with the $ git reset --hard HEAD~ command.

The output of $ git l master and $ git l doc proves that the new revision is included only in the master branch.

You also can verify it with one command to display all branches: $ git log --oneline --graph --decorate --all.

4. Merging diverged branches

You want to merge the branches shown in Figure 6-5. The feature branch is to be merged into the master branch. The repository you want to obtain is presented in Figure 6-8. Figure 6-8 underlines the order in which revisions m4, m5 and f1, f2, f3 were created. In this recipe, this order is not important, therefore Figure 6-8 could also be drawn as in Figure 6-9



Solution: In cases where fast-forward is not possible the $ git merge command creates an additional revision, called merge

commit. This commit differs from the commits you have created so far because it contains more than one parent.

It joins two or more different revisions. This gives us the opportunity to classify every commit as either a non-merge

commit or a merge commit. A merge commit is a commit that has two or more parents. A non-merge commit is a

commit with exactly one parent. Obviously, the commit created in this recipe has two parents thus it a merge commit.

When inspecting the history with $ git log or $ gitk, you can filter out both types of commits. The command:

$ git log --oneline --merges

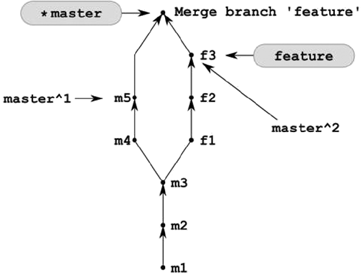
outputs only merge commits, while

$ git log --oneline --no-merges

prints only non-merge commits. You also can set the expected minimal and maximal number of parents with:

$ git log --oneline --max-parents=X --min-parents=Y

where X and Y are arbitrary positive integers.



Merge branch 'X'

where the X is the name of the branch you merged in (feature, in our recipe). You can memorize the above rules,

remembering that, when on the master branch, the $ git log --oneline -1 command prints:

6fb2 Merge branch 'feature'

As you can guess the working directory now contains all the files from both branches. The command $ ls

outputs the files: f1.txt, f2.txt, f3.txt, m1.txt, m2.txt, m3.txt, m4.txt, m5.txt.

The merge can be undone exactly as in Recipe 6-3. Only this time you can use not only reflog and SHA-1, but also

ancestor and n-th parent references. Assuming that you are in the master branch both the following commands will

undo the merge discussed in this recipe:

$ git reset --hard master^

$ git reset --hard master~

1. Rebasing divergent branches  Problem: You work in a repository with two branches named master and feature. The branches diverged and your repository  now looks like Figure 7-1(a). You want to transform the feature branch in such a way that:
   1. The history is linear (that means that the branches are not divergent anymore).
   2. The master branch is merged into the feature branch.
   3. All of the commits that were made in the feature branch are at the very top of the master branch.  The repository you want to achieve is presented in Figure 7-1(b).

Solution:

1. Checkout the feature branch with the $ git checkout feature command.

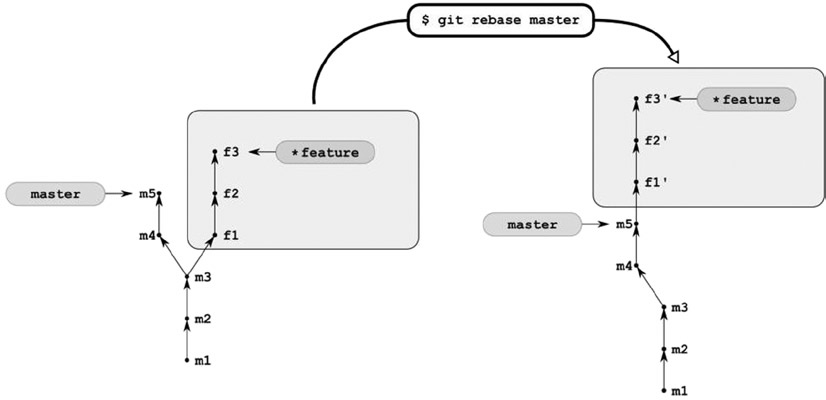
2. Rebase the feature branch onto the master branch with the $ git rebase master

command.

3. Checkout the master branch with the $ git checkout master command

The transformation performed by $ git rebase master can be described as applying the changes introduced by the

commits from the current branch at the top of another branch



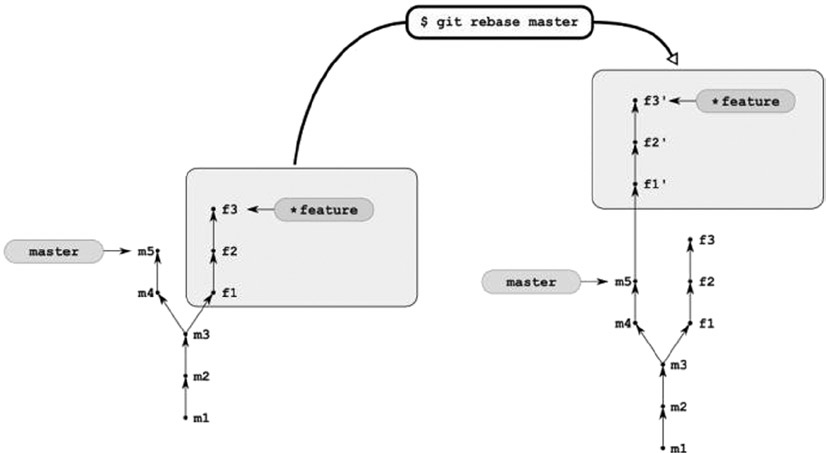
create a new revision that will have the same comment and that will introduce the same changes to your files.

What happens to the original revisions f1, f2, and f3? Nothing. They are left intact. To be more accurate,

rebasing can be depicted as seen in Figure. The original revisions f1, f2, and f3 are not referred to by any branch

anymore—they became dangling revisions. But they remain unchanged in the database. At least as long as you do not

expire the reflog and prune the database.



Let’s find the original revision f3. As always—you can use the $ git reflog command. But this time its output

can be misleading. Probably it will be easier to explore reflog with the $ git log command. We want to get the list of

all the commits that:

• Are included in reflog

• Have comments containing f3 string

$ git log --walk-reflogs --grep=f3 --pretty="%h %s %cd" | sort | uniq

$ git branch -D feature

$ git checkout -b feature [SHA-1]

$ git checkout -B feature [SHA-1]

# current branch is feature

$ git rebase master

$ git merge master

$ git rebase a b

$ git rebase a

$ git rebase a HEAD

Change the current branch to feature and use one parameter for rebase as in:

$ git checkout feature

$ git rebase master

• Use two parameters for rebasing—your current branch is not important then:

$ git rebase master feature

Whichever is the case, feature is the current branch after successful rebasing.

6. Diverging three branches Problem

Your repository contains two divergent branches master and feature, as shown in Figure 7-8(a). First you want to work on some new idea, basing your work on the latest revision in your feature branch. You need to create a new branch called brave-idea and to commit your changes as revisions b1 and b2. Next you want to switch to the feature branch and create three new revisions f4, f5, and f6. The repository you want to achieve is shown in Figure 7-8(b).

1. Create and checkout the brave-idea branch with $ git checkout -b brave-idea feature

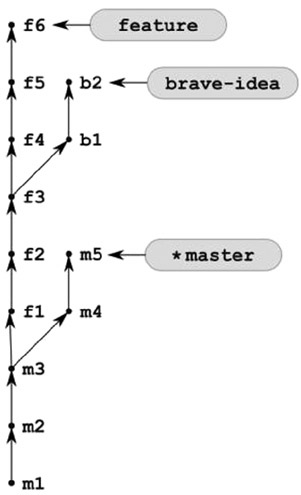
2. Create two revisions in the brave-idea branch with $ git simple-commit b1 b2

3. Checkout the feature branch with $ git checkout feature

4. Create three revisions in the feature branch with $ git simple-commit f4 f5 f6

5. Change the current branch to master with $ git checkout master

The result of $ git log --oneline --graph --decorate --all for a repository



The common ancestor of two branches is the latest revision included in two branches. For branches feature and

brave-idea it is f3. For master and feature it is m3. You can find the common ancestor using:

$ git merge-base feature brave-idea

If you want to get the common ancestor for more than two branches, use --octopus parameter. The command:

$ git merge-base --octopus feature brave-idea master

prints the SHA-1 of m3 commit.

The range of commits was already discussed in Recipe 7-2. A special operator .. is interpreted as a difference of

branches. The command:

$ git log --oneline master..brave-idea

prints commits b2, b1, f3, f2, and f1, while:

$ git log feature..master

outputs revisions m4 and m5.

The set of new commits introduced by two branches is resolved by the ... operator. It is a symmetrical difference

of branches. The output of:

$ git log feature...brave-idea

consists of f6, f5, f4, b2, and b1.

The even more verbose way to specify included and excluded revisions is to use --not operator. The command:

$ git log a b c --not d --not e --not f

prints the revisions included in a, b, or c and excluded from d, e, and f. This can also be written as:

$ git log a b c ^d ^e ^f

f6, f5, f4—commits introduced in feature branch

b2, b1—commits introduced in brave-idea branch

m5, m4—commits introduced in master branch

$ git log --oneline

master feature brave-idea

^`git merge-base master feature`

^`git merge-base feature brave-idea`

master feature brave-idea

^`git merge-base master feature`

^`git merge-base feature brave-idea`

$ git log --format="%h" --grep=XXX --all