Git

Tool

Version Control System Tool

Git –core part written in c

## Linus Trovald, 2005

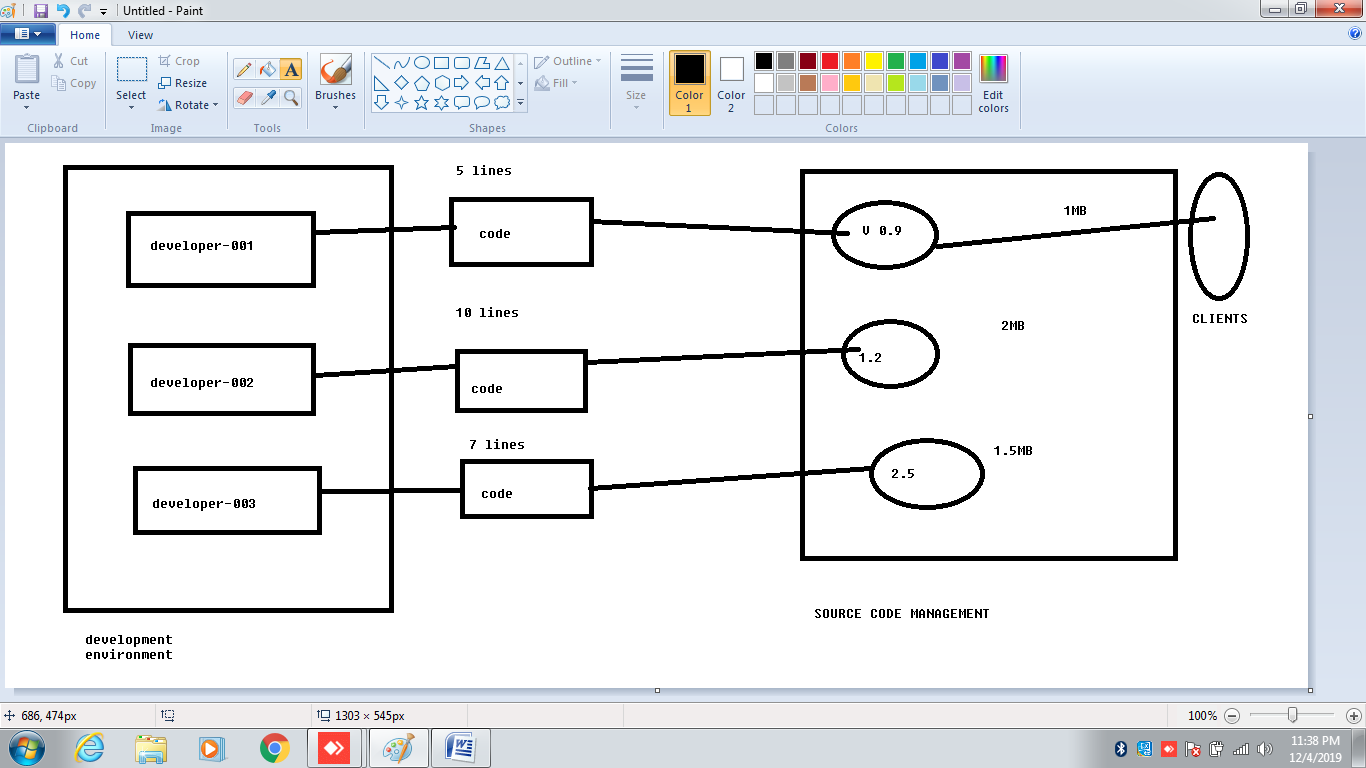
Linux Kernel Development

Designed and Developmeny -----------git

Git

Open Source Tool

Version Control System Tool



VERSION CONTROL

SYSTEM

ALLOWS YOU TO KEEP TRACK OF CHANGES

MADE THE YOUR CODE OVERTIME

COLLABORATION ON THE SAME WORK

NEEDS SPECIFIC CHANGES

ASSOCIATED CON TRIBUTORS ARE TRACKED.

VERSION CONTROL

SOFTWARE

HELPS SOFTWARE DEVELOPERS TO WORK TOGETHER

MAINTAIN A COMPLETE HISTORY OF THEIR WORK

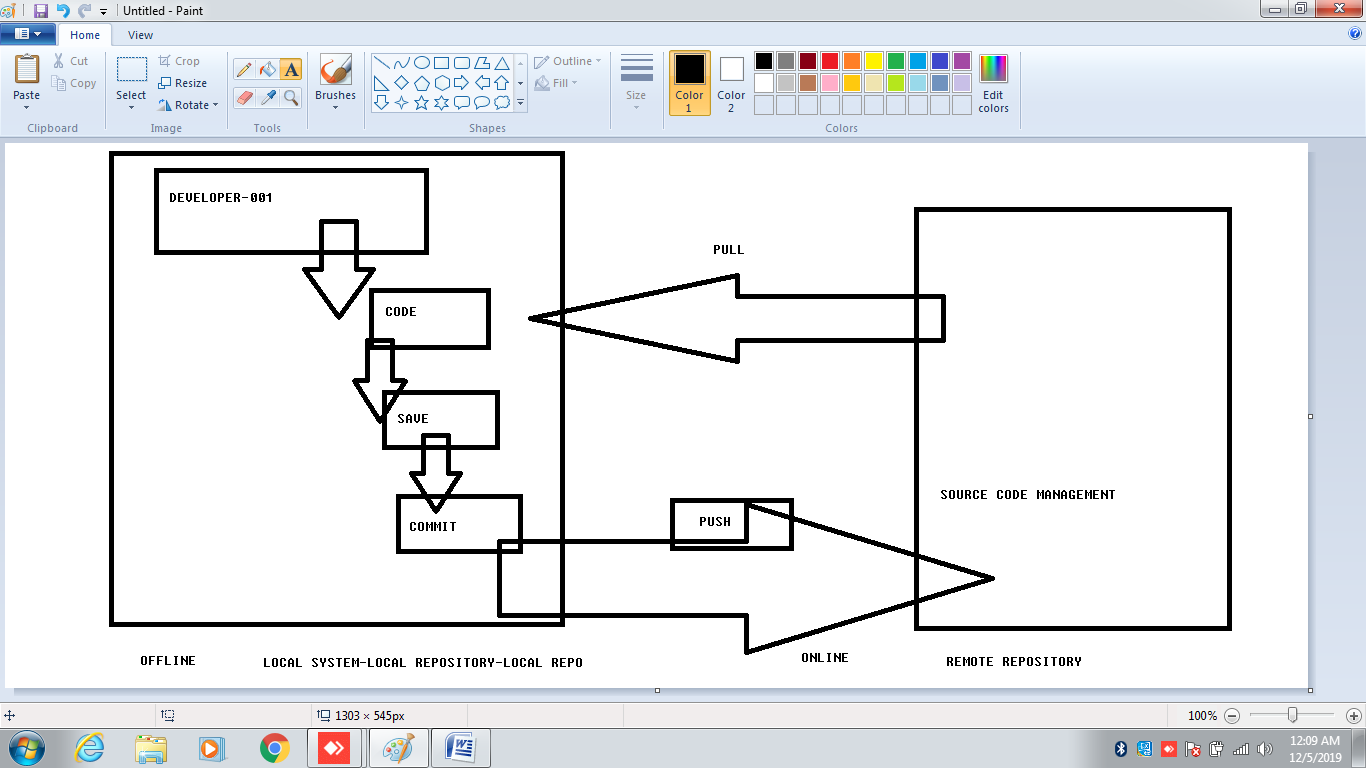
TYPES OF VCS

1.CVCS-----CENTRAL SERVER DOWN-------------------DATA LOSS OF BACKUP-------------------------

2.DVCS----------DISTRIBUTED VERSION CONTROL SYSTEM------SERVER GOES DOWN----------------CLIENT SIDE REPOSITORY-----------COPIED BACK----------SERVER TO RESTORE IT-------------------------OFFLINE WORKS

OFFLINE WORKS

1. COMMIT CHANGES
2. CREATE BRANCHES
3. VIEW LOGS



DISTRIBUTED VERSION CONTROL SYSTEM

SERVER GOES DOWN

CLIENT SIDE REPOSITORY(LOCAL REPO)

COPIED BACKUP

SERVER TO RESTORE IT

OFFLINE WORKS

1.COMMIT CHANGES

2.CREATE BRANCHES

3.VIEW LOGS

THEN AFTER NEED NETWORK

YOUR CHANGES AND TAKE THE LATEST CHANGES

ADVANTAGES

1.FREE /OPEN SOURCE

2.SMALL

3.FAST

4.BACKUP

5.SECURITY

GIT

OPEN SOURCE TOOL

VERSION CONTROL SYSTEM TOOL

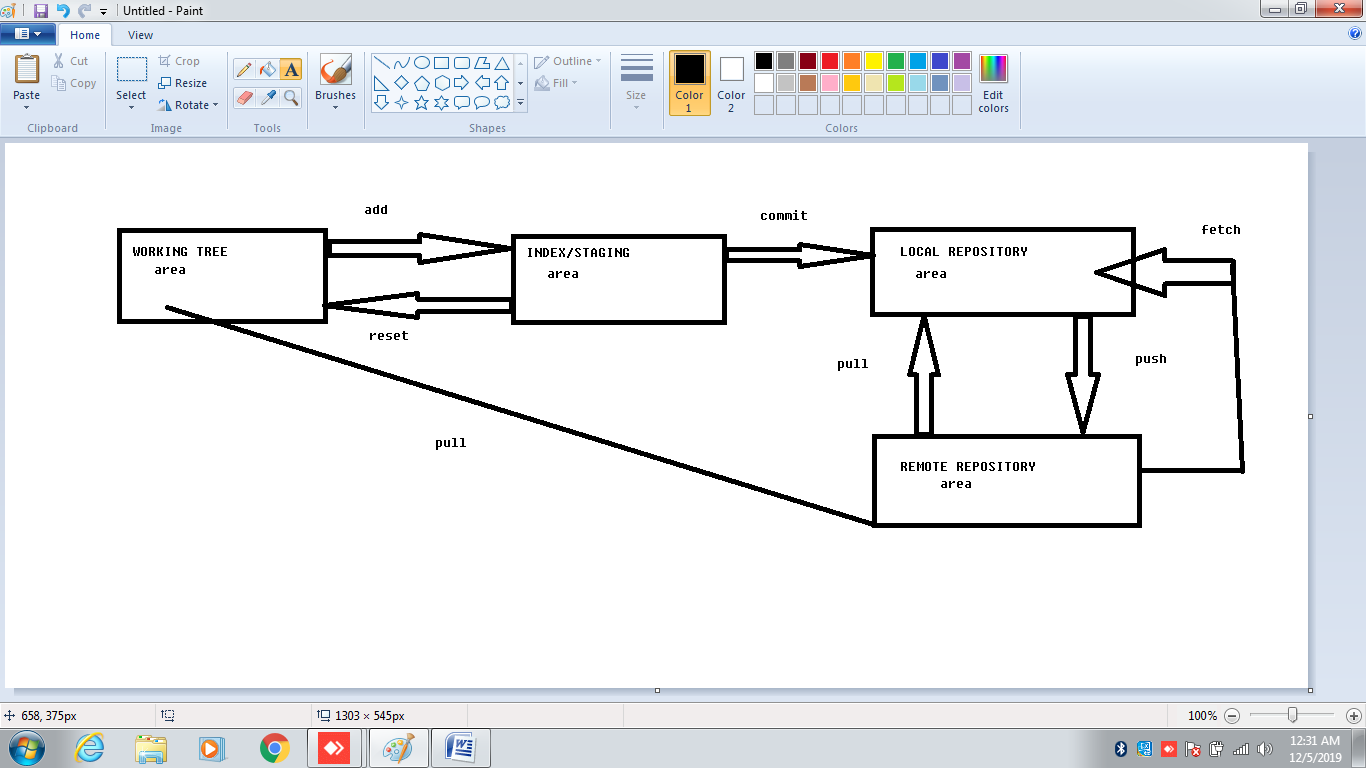
A.DISTRIBUTED REVISION CONTROL

B.SOURCE CODE MANAGEMENT SYSTEM

PROJECTS(SMALL/BIG)

SPEED AND EFFICIENCY

COMMONLY USED FO SOURCE CODE MANAGEMENT



Server

network

GIT TREE STRUCTURE

1. SETUP AND CONFIG

GIT

CONFIG

HELP

1. GETTING PROJECTS

INIT

CLONE

1. BASIC SNAPSHOTTING

ADD

STATUS

DIFF

RM

MV

1. BRANCHING AND MERGING

BRANCH

CHECKOUT

MERGE

LOG

TAG

WORKTREE

1. SHARING AND UPDATIN PROJECTS
2. INSPECTION AND COMPARISON
3. PATCHING
4. DEBUGGIND

GREP

1. GUIDES

GLOSSARY

WORKFLOWS

1. EMAILS

AM

APPLY

SEND-EMAIL

1. EXTERNAL SYSTEMS

SVN

FAST-IMPORT

1. ADMINSTRATION

CLEAN

GC

BUNDLE

1. SERVER ADMIN

DAEMON

UPDATE-SERVER-INFO

1. PLUMBING COMMANDS

CAT-FILE

CHECK-IGNORE

CHECKOUT –INDEX

COMMIT-TREE

COUNT-OBJECTS

LS-FILES

READ-TREE

WRITE-TREE

1.$git init

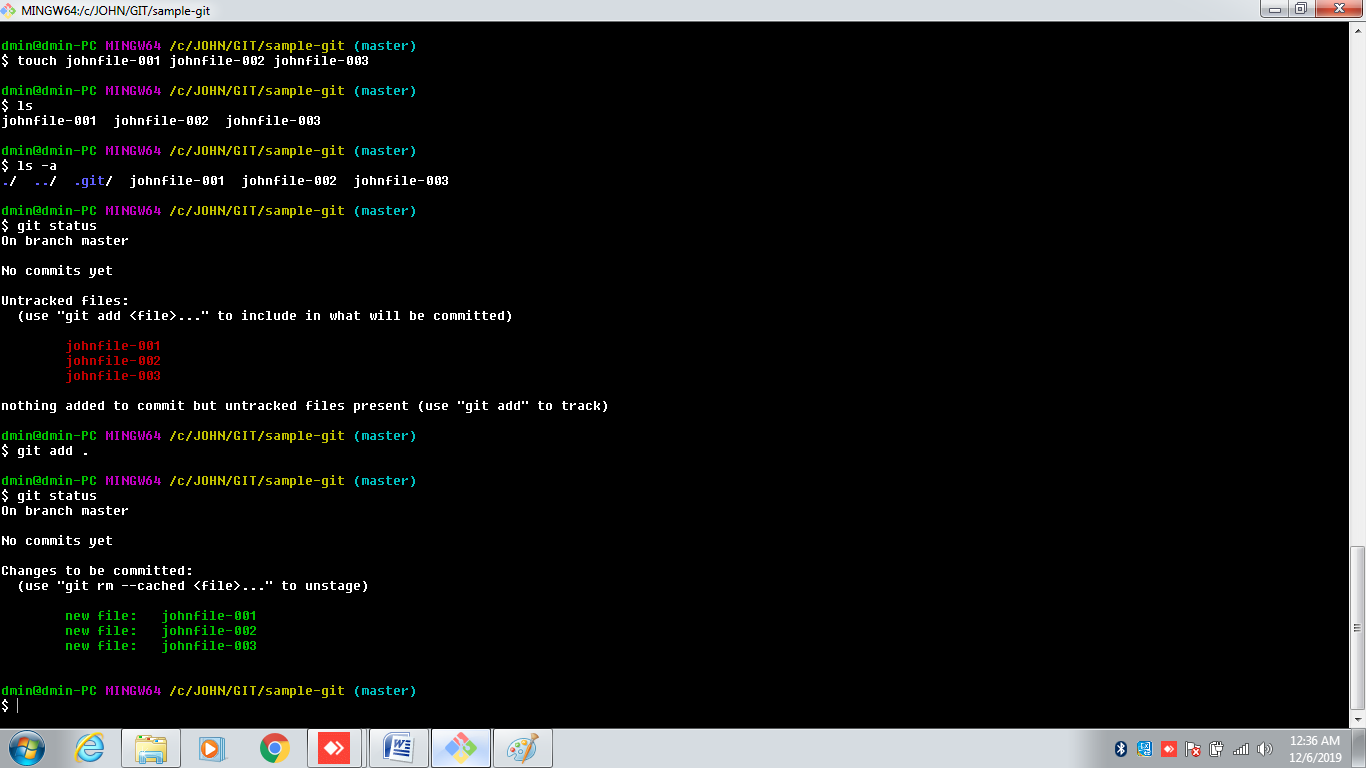
2. touch

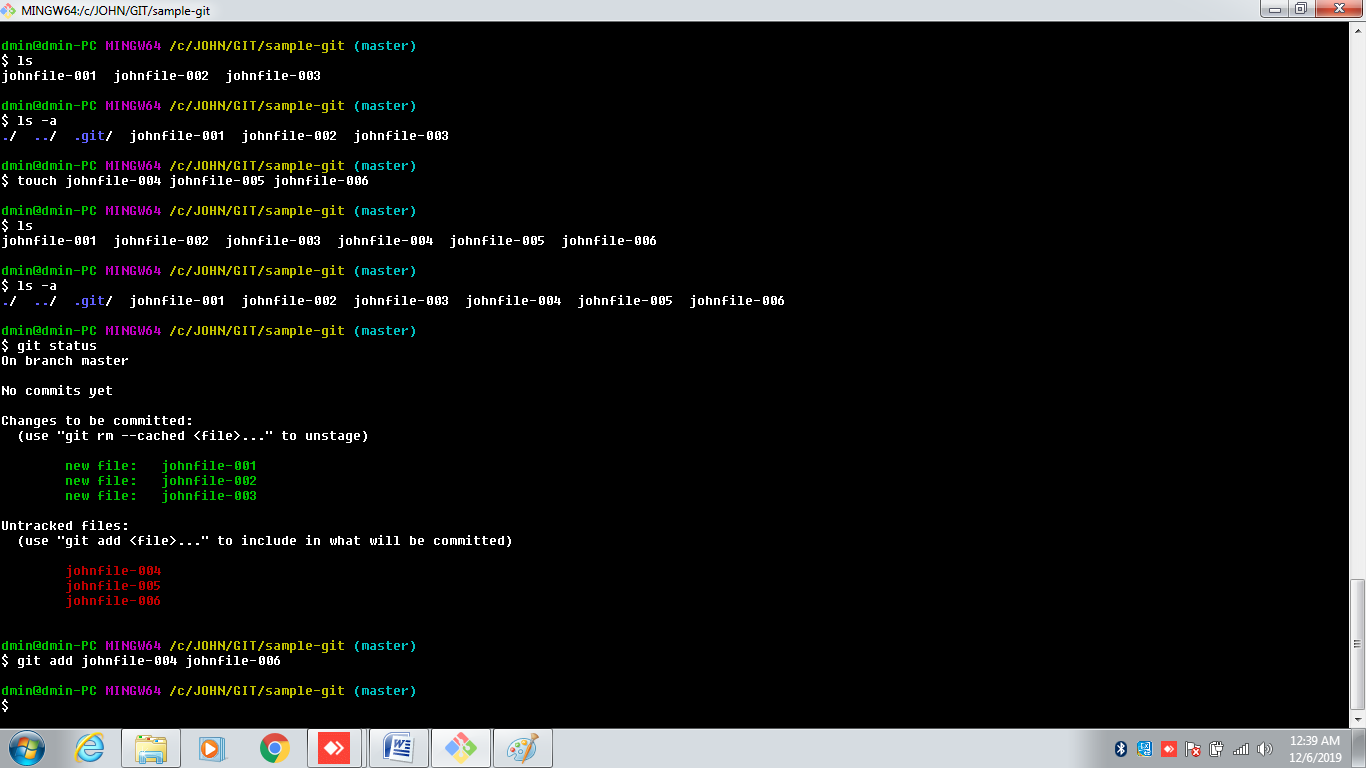
3.$git status

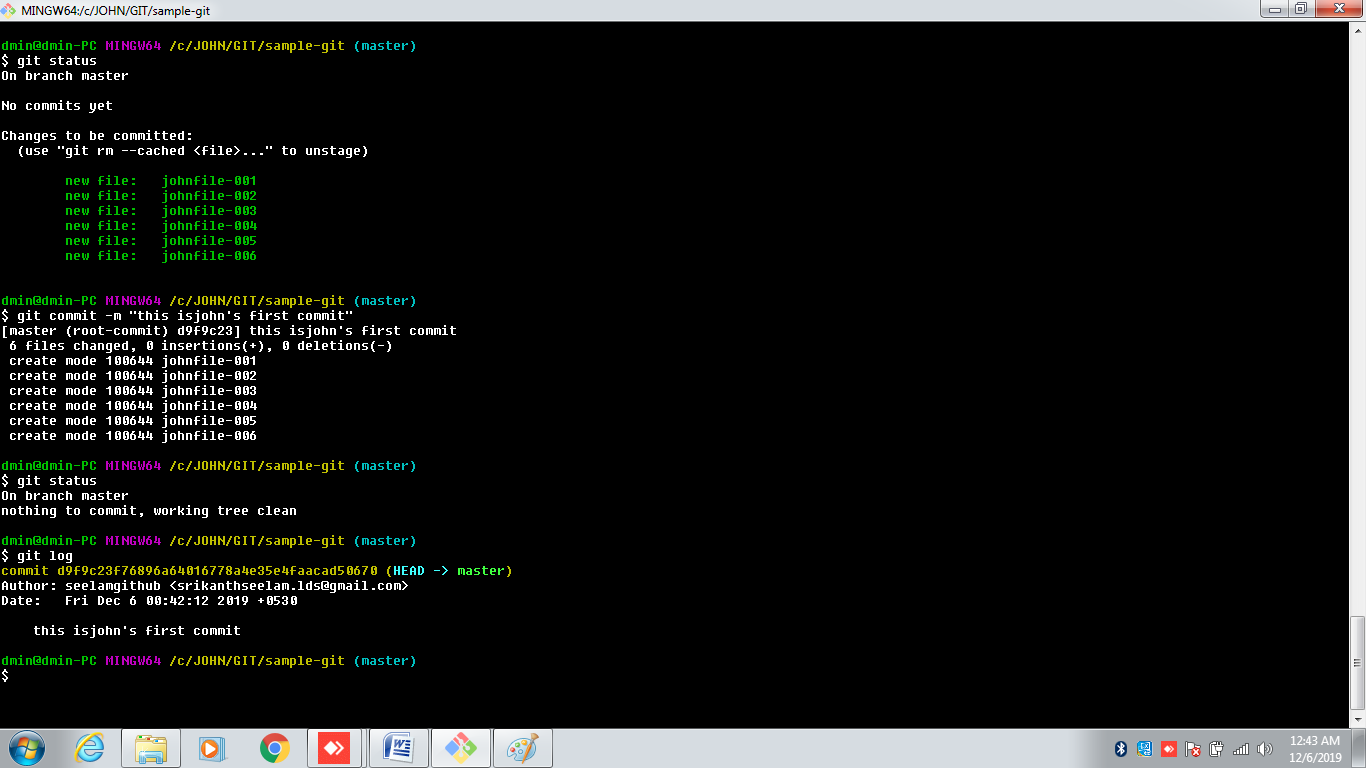
4.$git add .

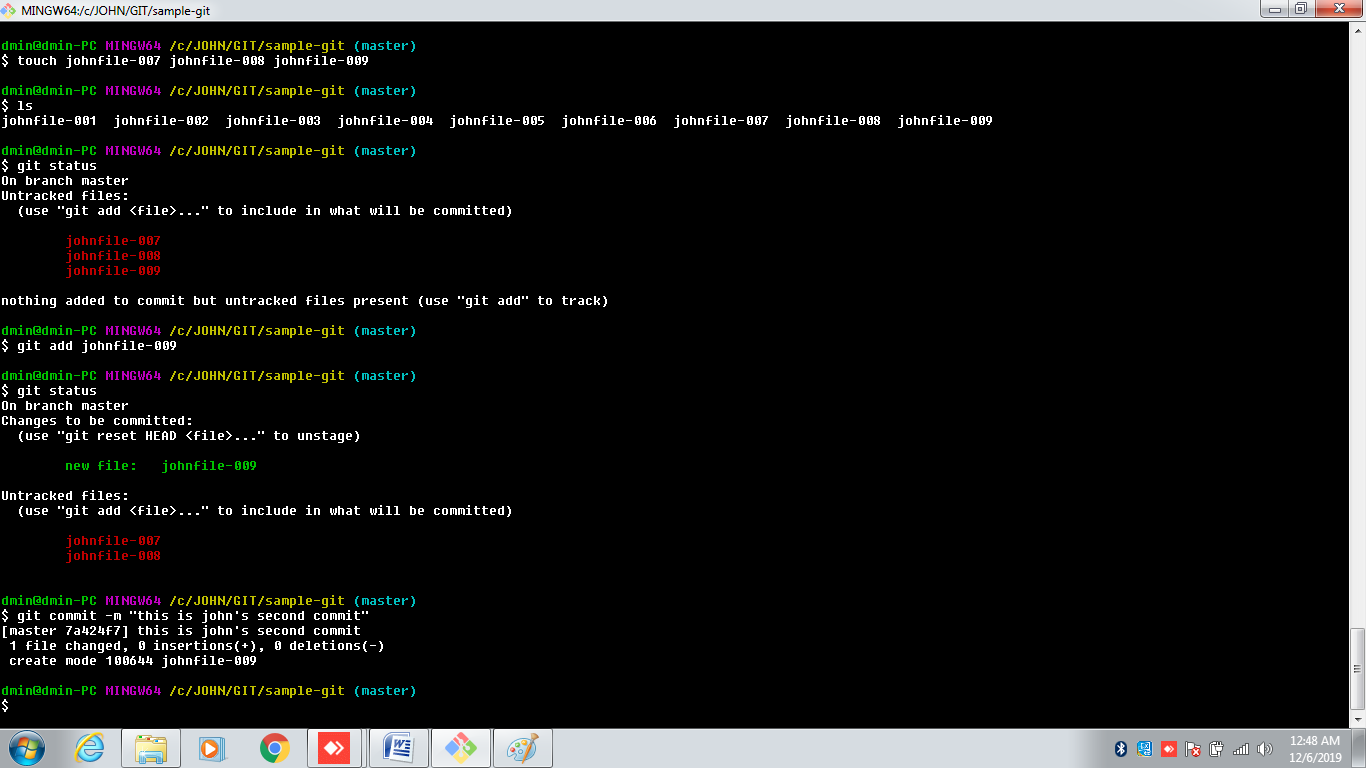
5.$git commit –m “write your message”

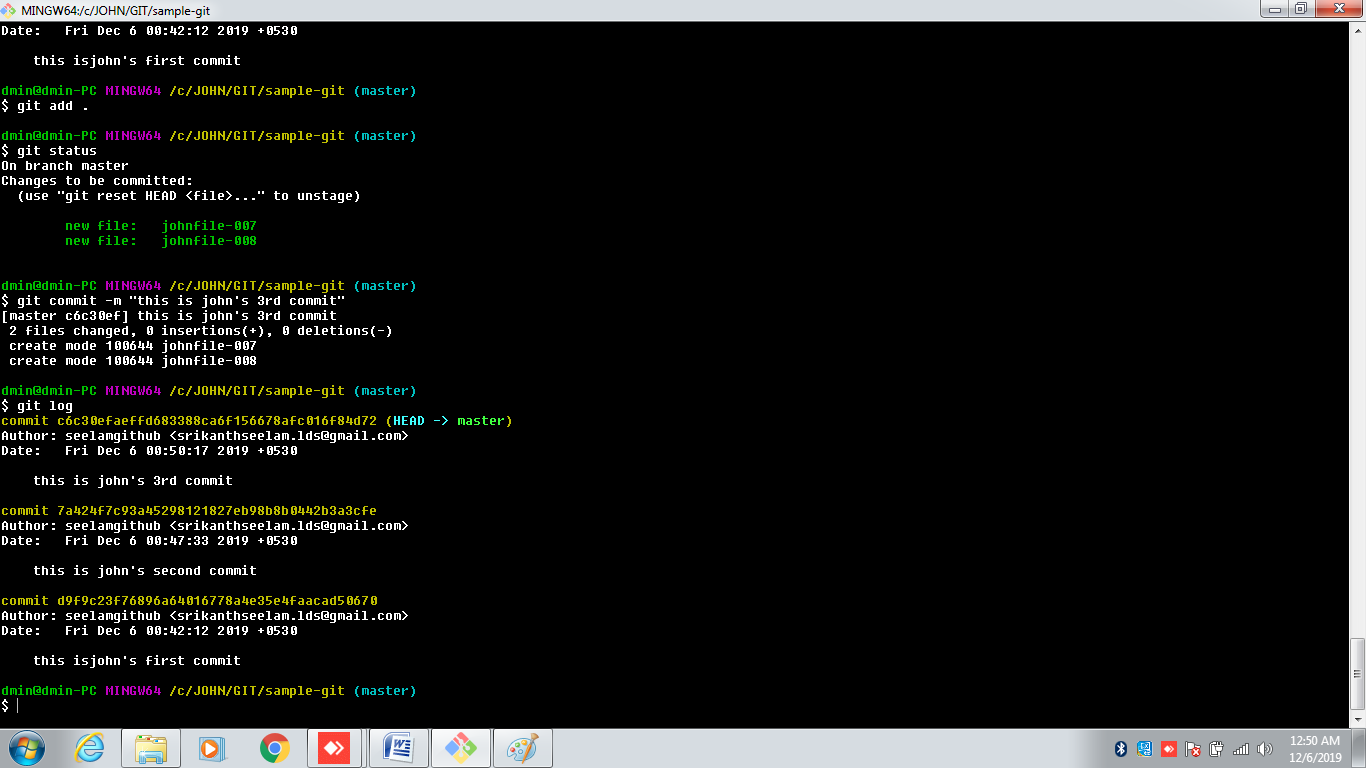
6.$git log











C:/----------🡪JOHN----------🡪GIT------------🡪sample-git

$git init

$touch file1 file2 file3

$ls

$git status

$git add .

$git status

$git commit –m “first commit”

$git stauts

$git log

////////////////////////////////////////////////////////$git push origin master

1.START WORKING AREA

clone

init

2.WORKING ON THE CURRENT CHANGES

add

mv

reset

rm

3.EXAMINE THE HISTORY AND STATE

status

show

log

grep

bisect

4.COMMON HISTORY

commit

checkout

branch

diff

merge

tag

rebase

5.COLLABORATE

push

pull

fetch

6.TEMPORARY AREA

Stash

CONFIGURATION FILE FROM LOCAL REPOSITORY

[core]

repositoryformatversion = 0

filemode = false

bare = false

logallrefupdates = true

symlinks = false

ignorecase = true

CONFIGURATION FILE FROM REMOTE REPOSITORY

[core]

repositoryformatversion = 0

filemode = false

bare = false

logallrefupdates = true

symlinks = false

ignorecase = true

[remote "origin"]

url = https://github.com/githubseelam/JOHNLEE.git

fetch = +refs/heads/\*:refs/remotes/origin/\*

[branch "master"]

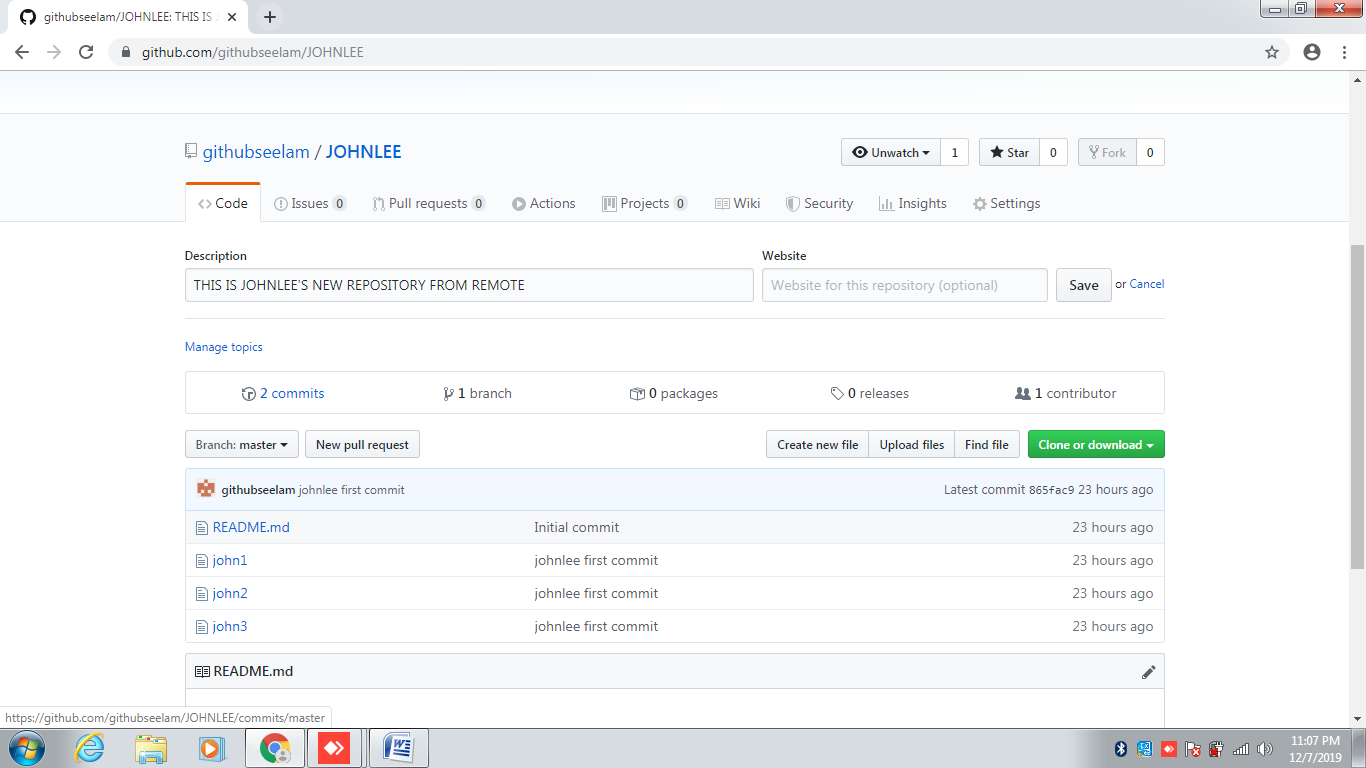
remote = origin

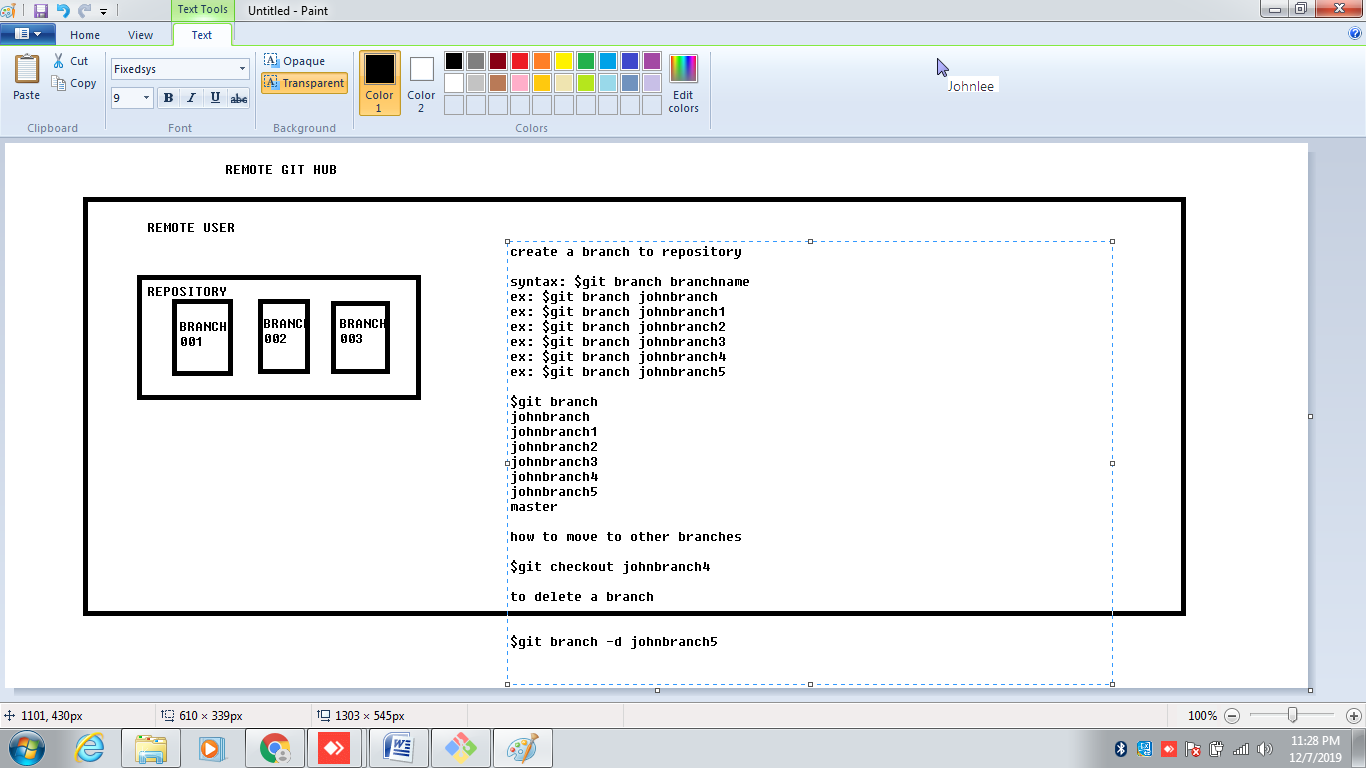
merge = refs/heads/master

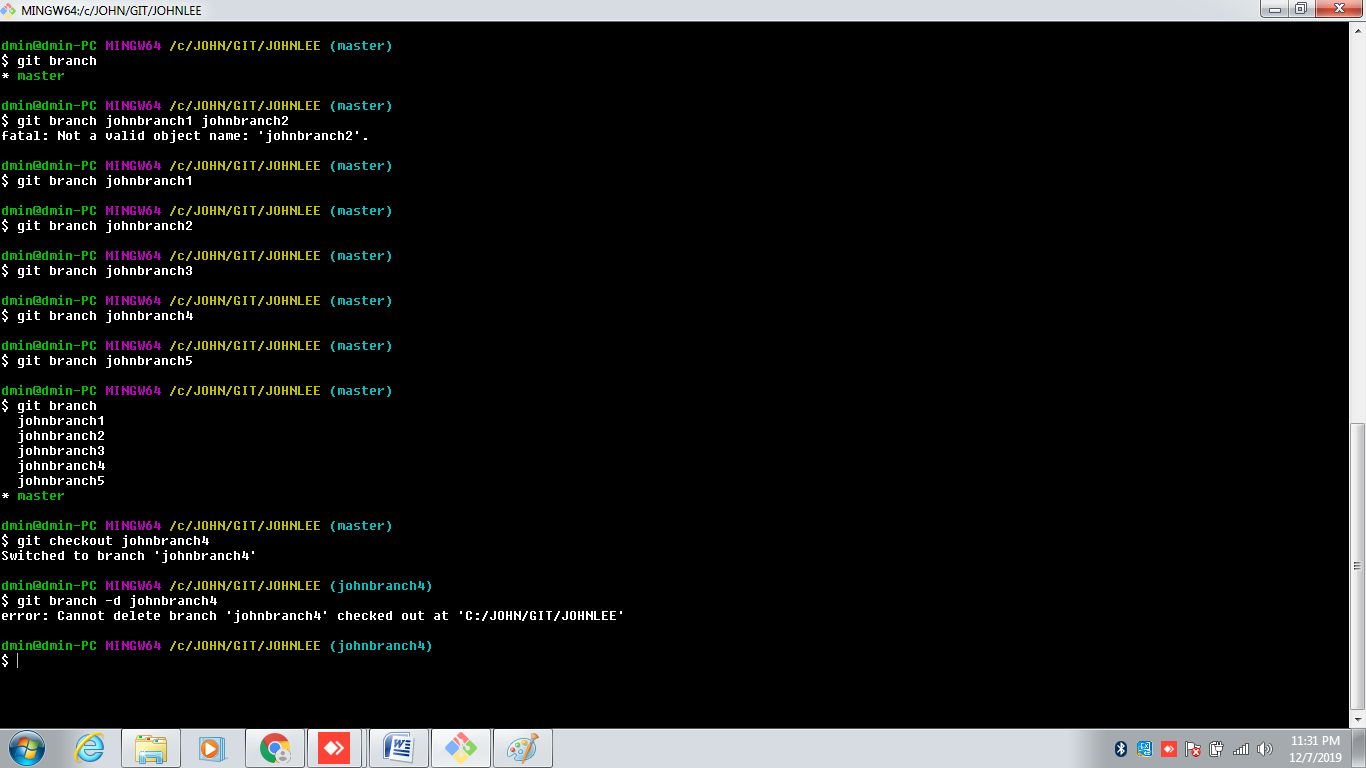
ASSINGMENT

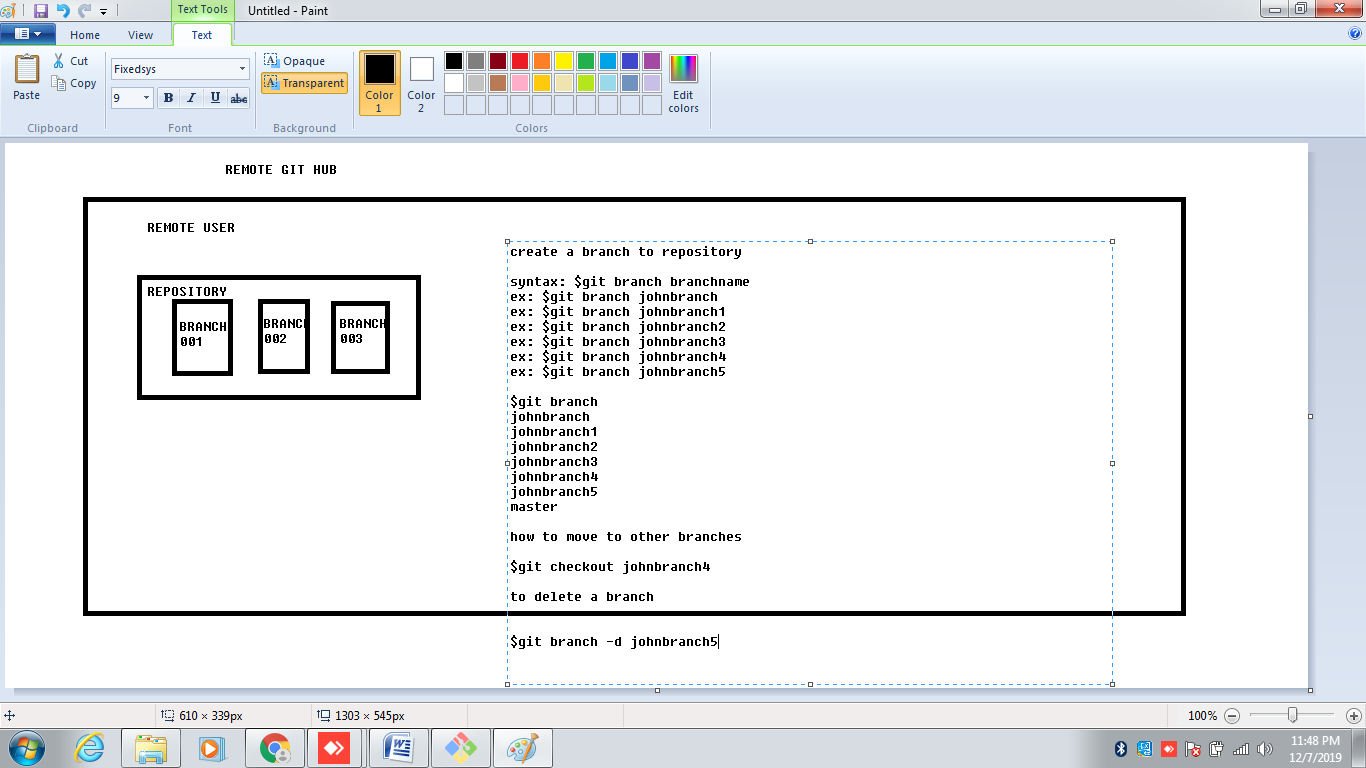
SCENARIO-001

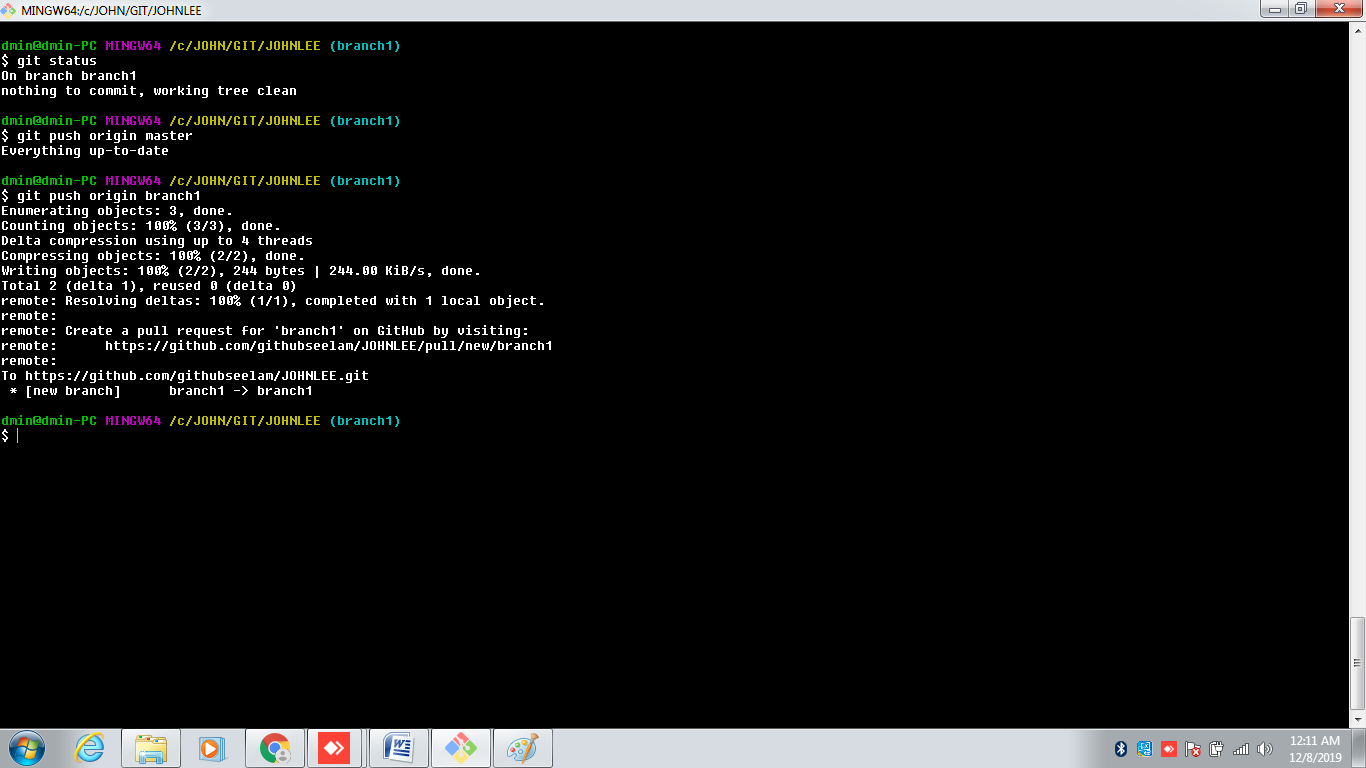
1. SING UP GIT HUB ACCOUNT
2. LOGIN INTO GIT HUB
3. YOUR PROFILE
4. REPOSITORIES
5. NEW
6. REPOSITORY NAME
7. DESCRIBE
8. SELECT PUBLIC
9. SELECT INITIALIZED WITH README.MD FILE
10. CREATE REPOSITORY
11. MOVE TO YOUR REPOSITORY
12. CLONE /DOWNLOAD
13. COPY URL
14. MOVE TO YOUR LOCAL SYSTEM
15. CHOOSE ANY FOLDER
16. GIT BASE OVER THERE
17. CLONE URL($git clone paste url)
18. Ls
19. Change remote repository
20. Create files with Touch command
21. Ls
22. Git status($git status)
23. $git add .
24. $git commit –m “your message”
25. Git push origin master
26. Go back to your github and check wheather files are created or not
27. Create files in your remote repository
28. Back to your local system
29. $git pull origin master

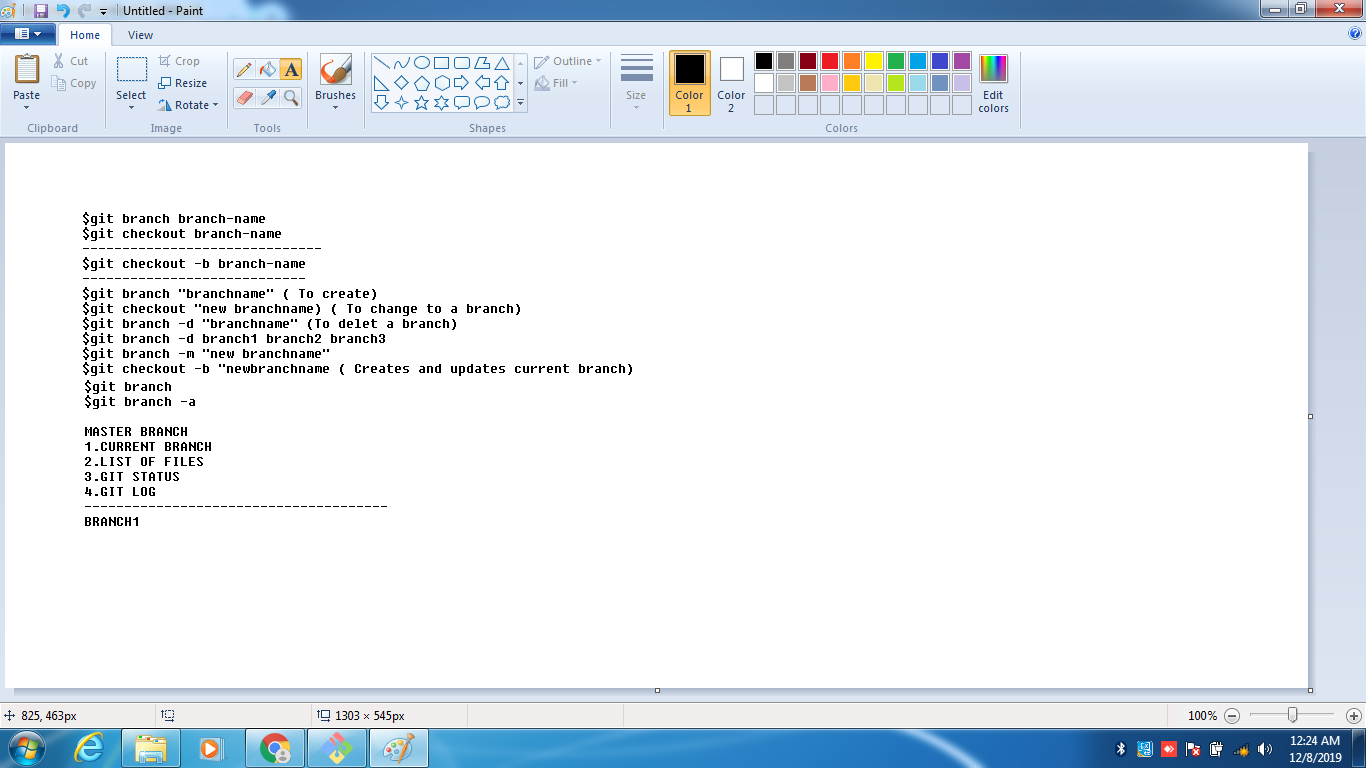


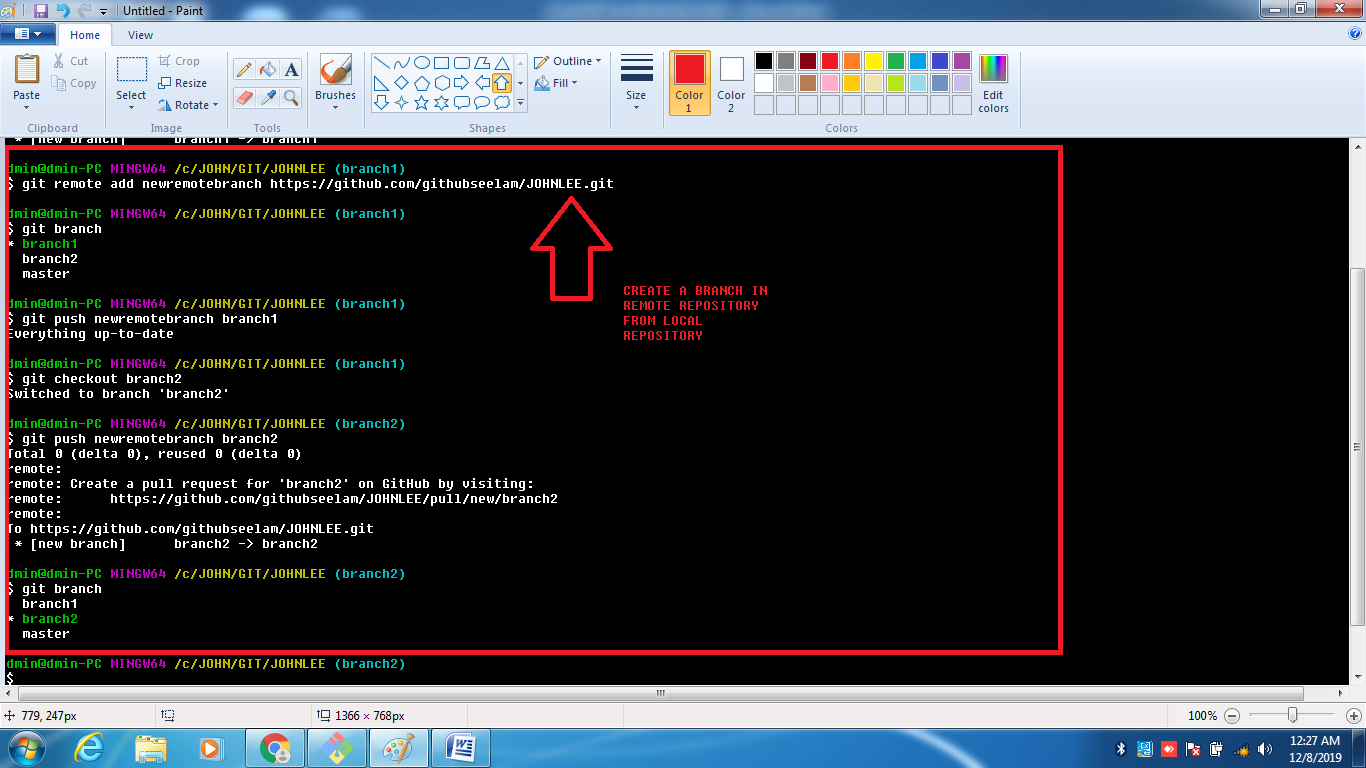












08TH DEC, 2019

COMMON HISTORY COMMANDS

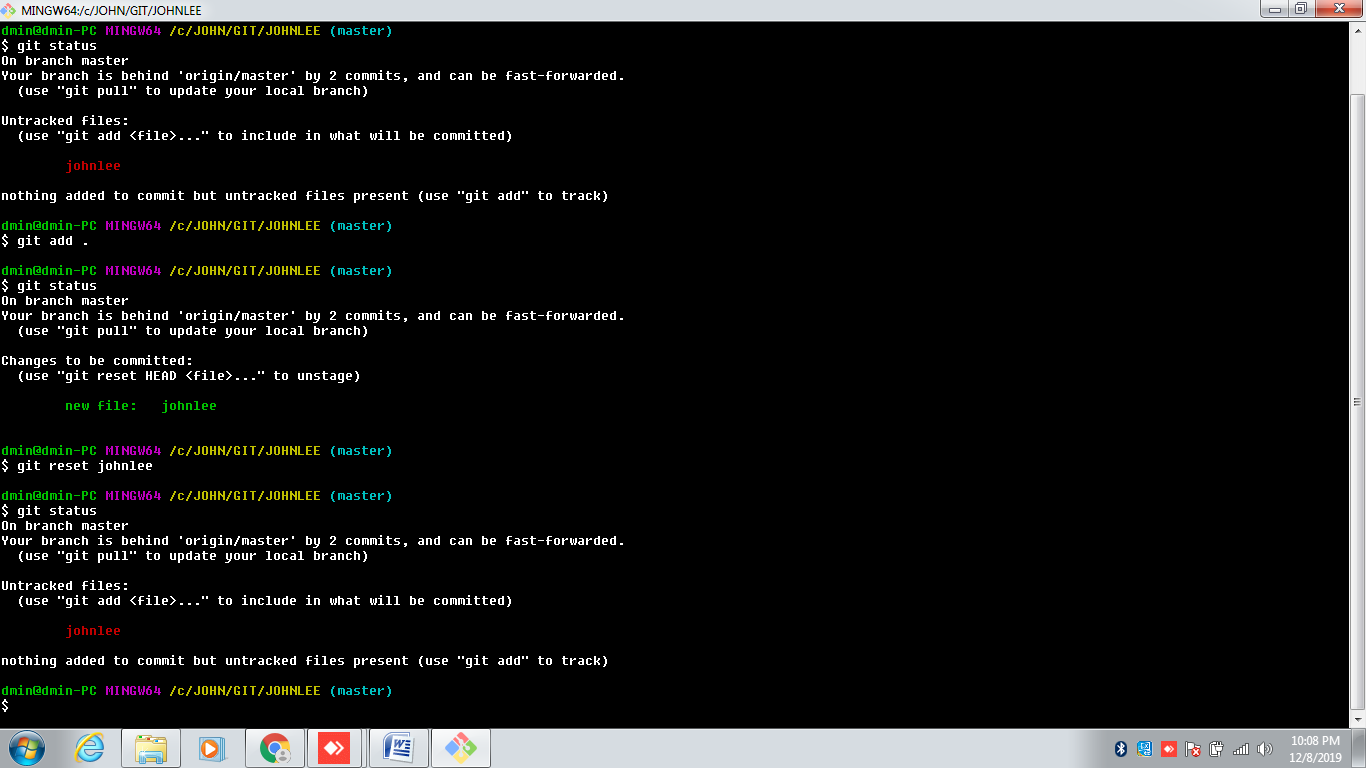
1.RESET

2.COMMIT

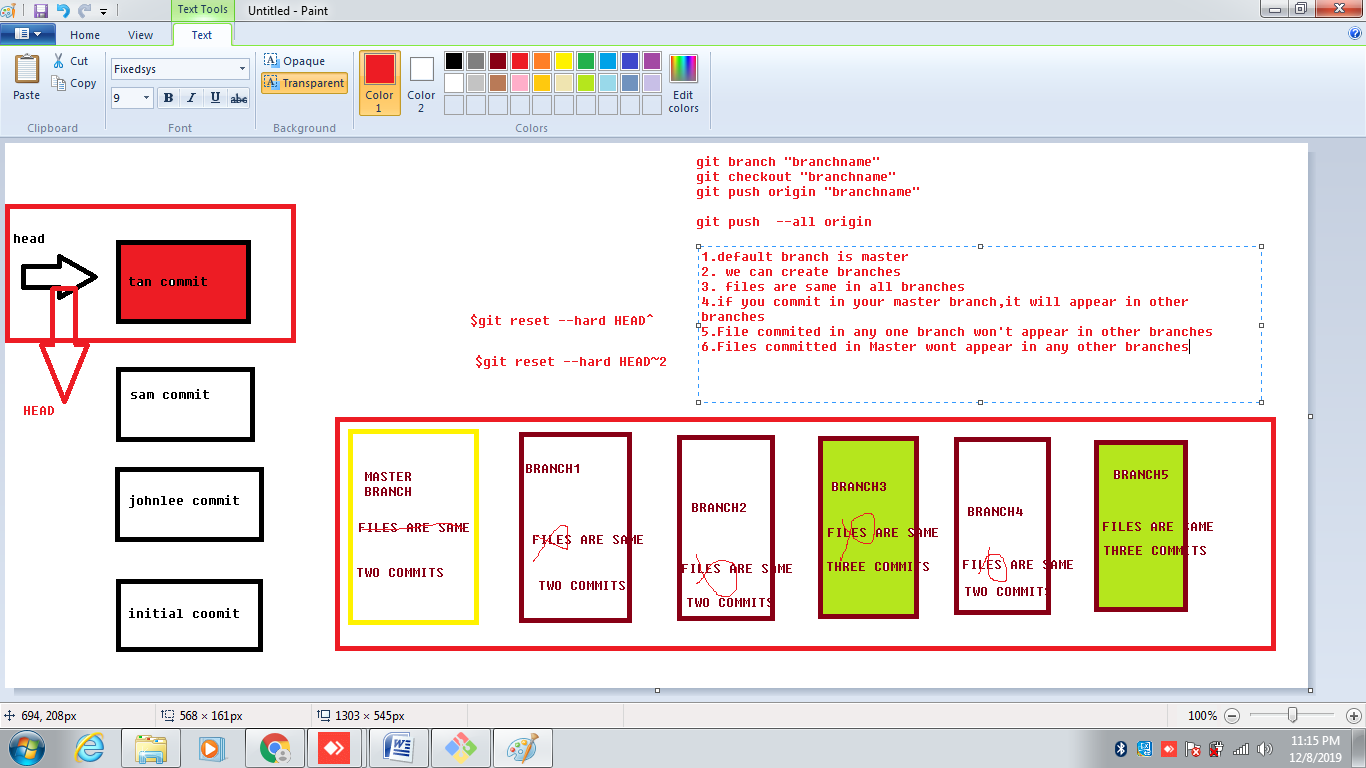
3.BRANCH

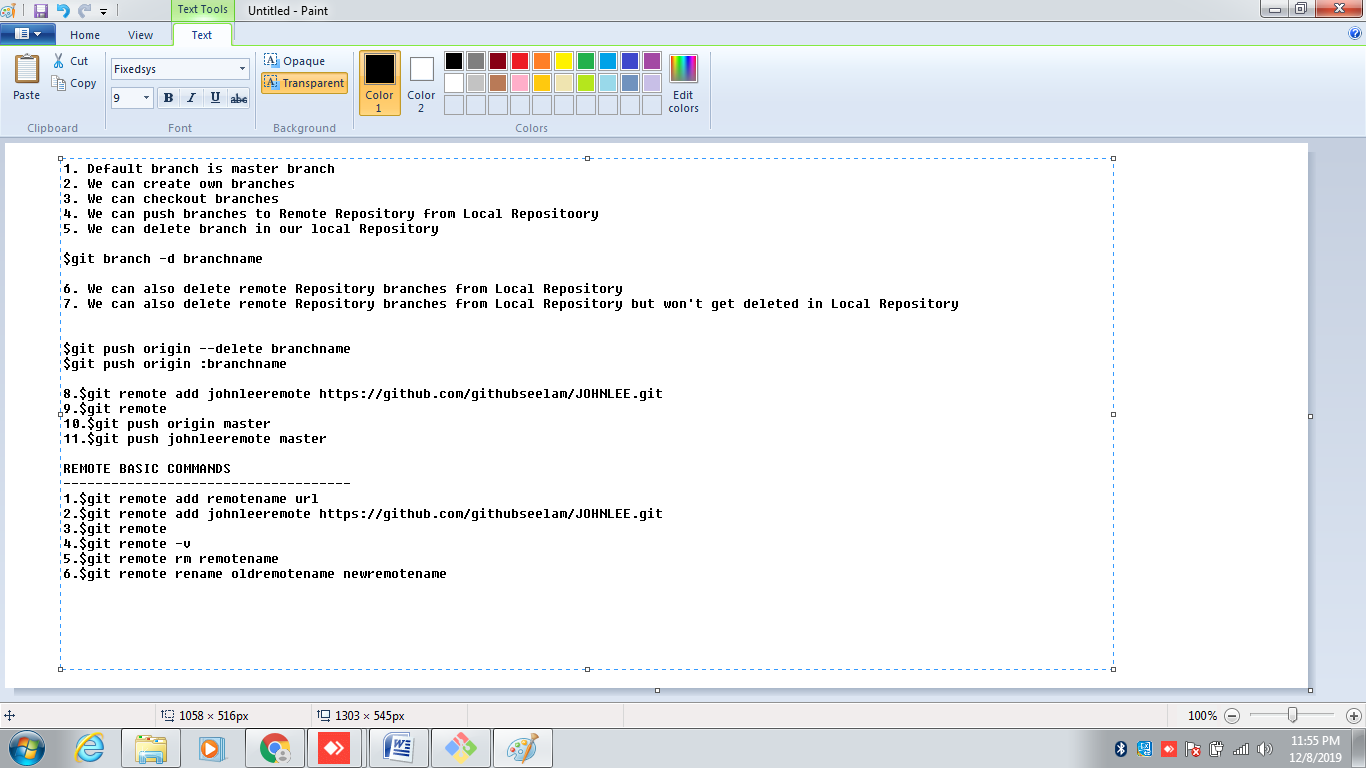
4.DIFF

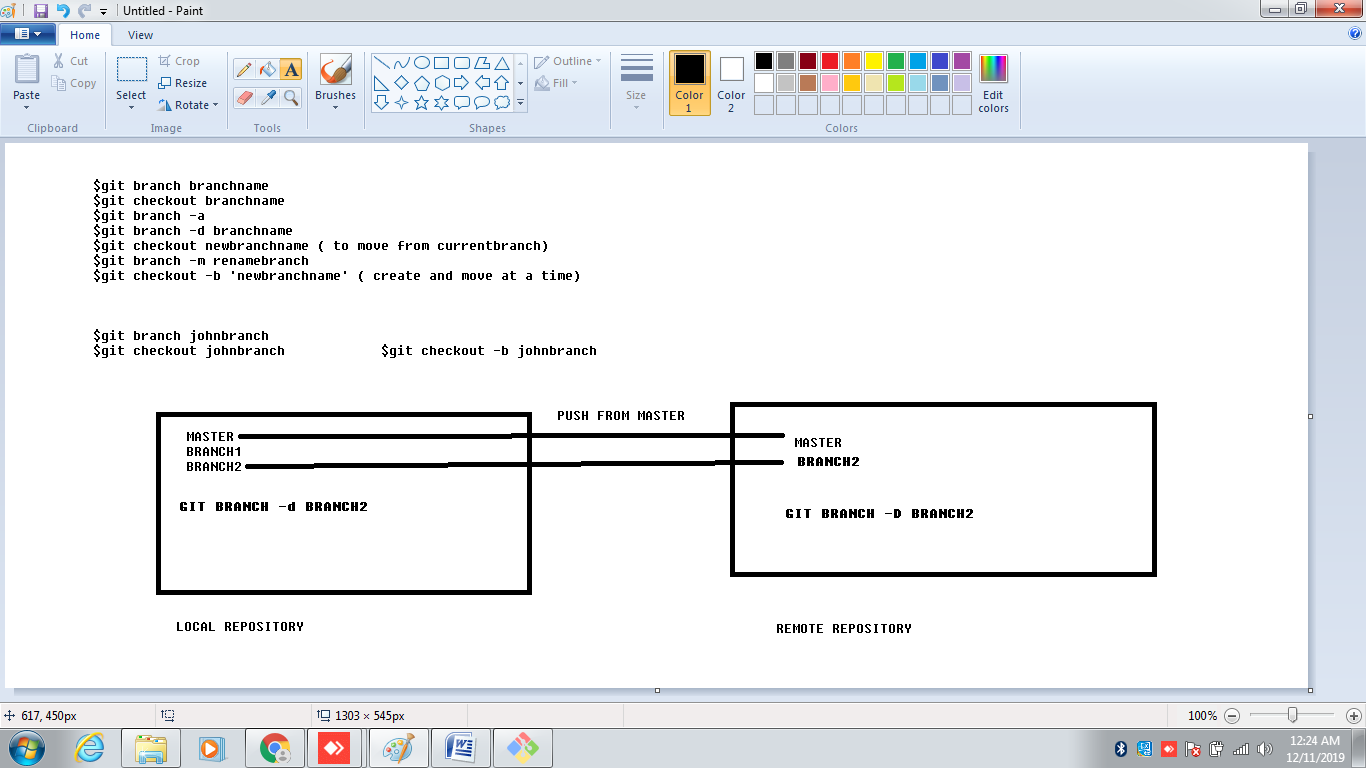
5.MERGE



$ git commit --amend -m "this is new commit"







**$git branch Branch001**

**$git branch Branch002**

**$git checkout Branch001**

**$git checkout -b Branch003**

**$git branch**

**Branch001**

**Branch002**

**Branch003**

**master**

**To Delete branches from Local Repository**

**$git branch -d Branch002**

**$git branch -D Branch003**

**To Delete branches from Remote Repository**

**$git push origin --delete Branch001--------------(or)---$git push origin :Branch001**

$git branch

$git remote

$git branch –a

$git remote –v

$git push origin master

$git push remotename master

$git remote add NewRemoteName ‘URL’

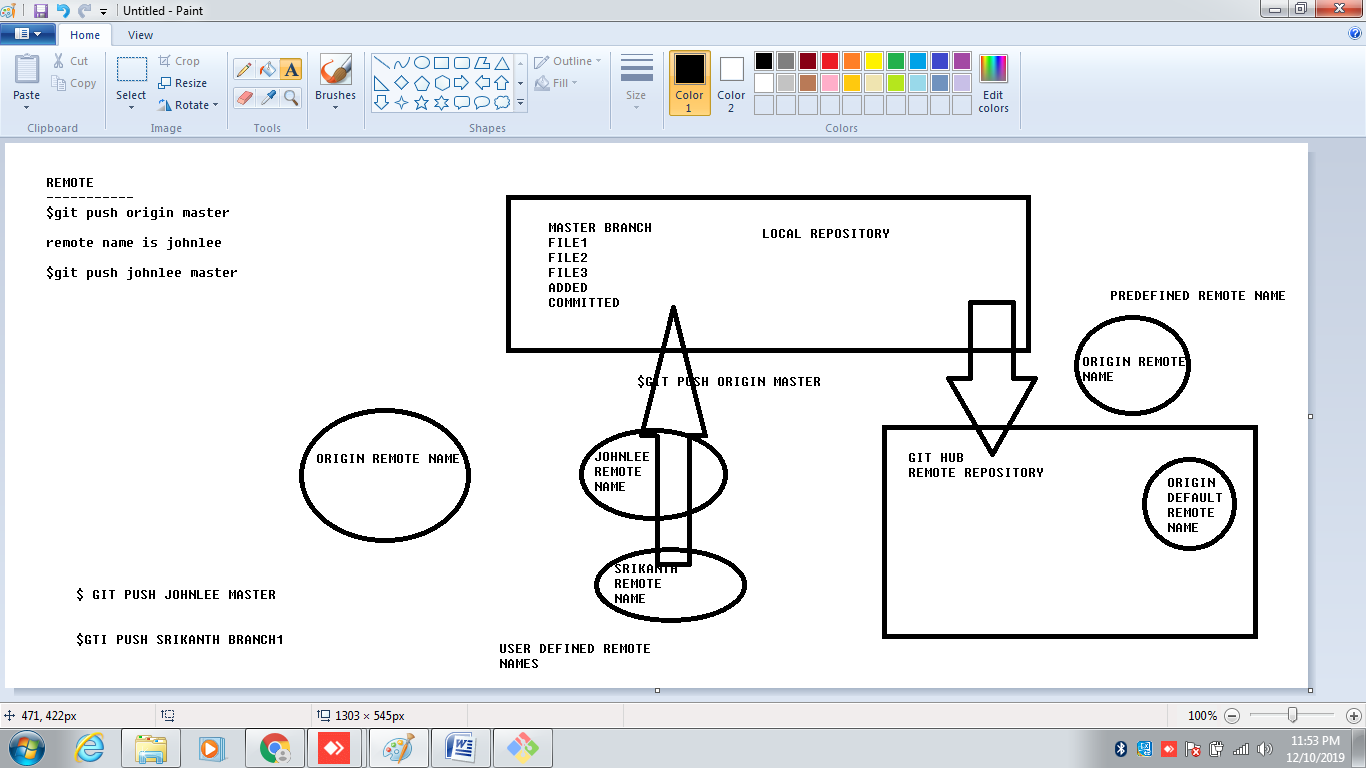
$git remote

$git remote rm ‘newremotename’

$git push newremotename master/branchname

$git push remotename branchname

$git remote rename Oldremote newremote



**$git branch Branch001**

**$git branch Branch002**

**$git checkout Branch001**

**$git checkout -b Branch003**

**$git branch**

**Branch001**

**Branch002**

**Branch003**

**master**

**To Delete branches from Local Repository**

**$git branch -d Branch002**

**$git branch -D Branch003**

**To Delete branches from Remote Repository**

**$git push origin --delete Branch001--------------(or)---$git push origin :Branch001**

$git branch Branch001

$git branch Branch002

$git checkout Branch001

$git checkout -b Branch003

$git branch

Branch001

Branch002

Branch003

master

$git branch -m NewBranchname

To Delete branches from Local Repository

$git branch -d Branch002

$git branch -D Branch003

To Delete branches from Remote Repository

$git push origin --delete Branch001--------------(or)---$git push origin :Branch001

REMOTE

$git remote add Remotename URL

$git remote

$git remote -v

$git push Remotename Branchname

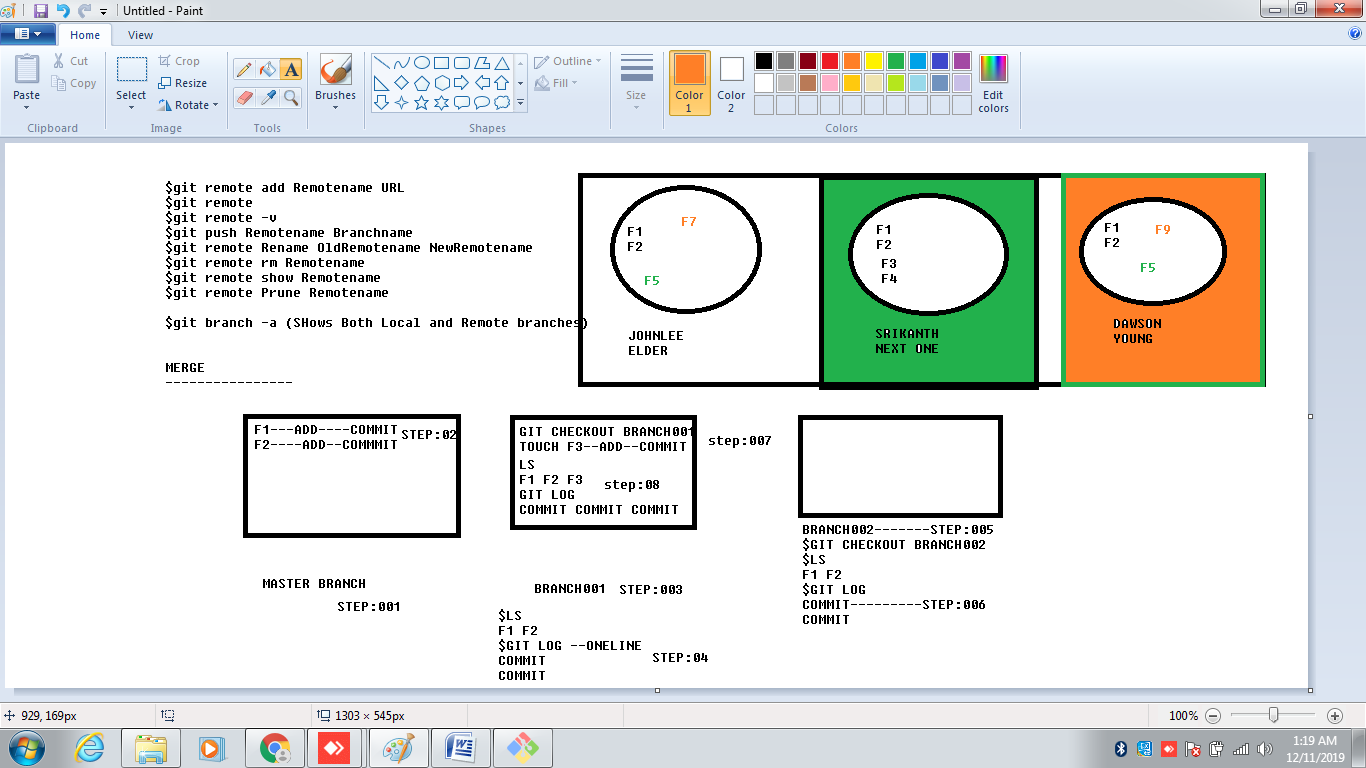
$git remote Rename OldRemotename NewRemotename

$git remote rm Remotename

$git remote show Remotename

$git remote Prune Remotename

$git branch -a (SHows Both Local and Remote branches)



1.$git init

2.$touch

3.$git status

4:$git add .

4.$git add filename

5.$git commit -m "description"

6.$git push origin master

7.$git pull origin master

8.$git push --all master

9.$git branch

10.$git branch branchname

11.$git checkout branchname

12.$git branch -a

13.$git branch -d branchname

14.$git branch –D branchname

15.$git branch -m renamebranch

16.$git checkout -b newbranchname

17.$git push origin --delete branchname

18.$git push origin :branch2

19.$git branch -d branchname

20.$git log

21.$git log --oneline

22.$git commit --amend -m new commit message

23.$git reset --hard HEAD~2

23.$git reset filename

24.$git remote add newremotename 'URL(from)GitHub'

25.$git remote

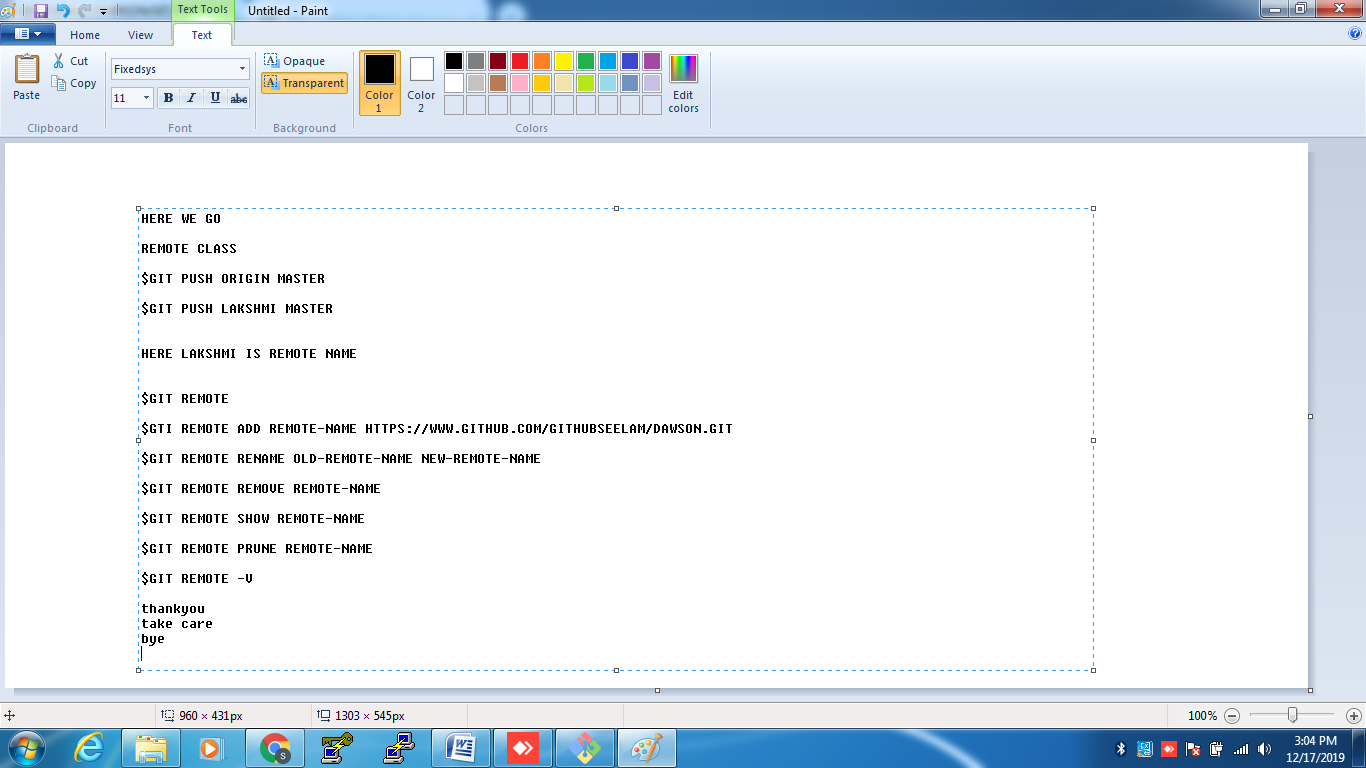
25.$git remote -v

26.$git remote rm remotename

27.$git remote show remotename

28.$git remote prune remotenametode

29.$git remote rename oldremote newremote



GIT DIFF

----------------------

DIFFERENCES BETWEEEN THE COMMITTED OR SAVED VERSIONS OF A FILE IN THE REPOSITORY

1.SHOW LOCAL CHANGES

2.SHOW DIFFERENCE BETWEEN STAGGED AND REMOTE REPOSITORY

3.SHOW DIFFERENCE BETWEEN TWO COMMITS

4.SHOW DIFFERENCE BETWEEN TWO FILES

5.SHOW DIFFERENCE BETWEEN TWO BRANCHES

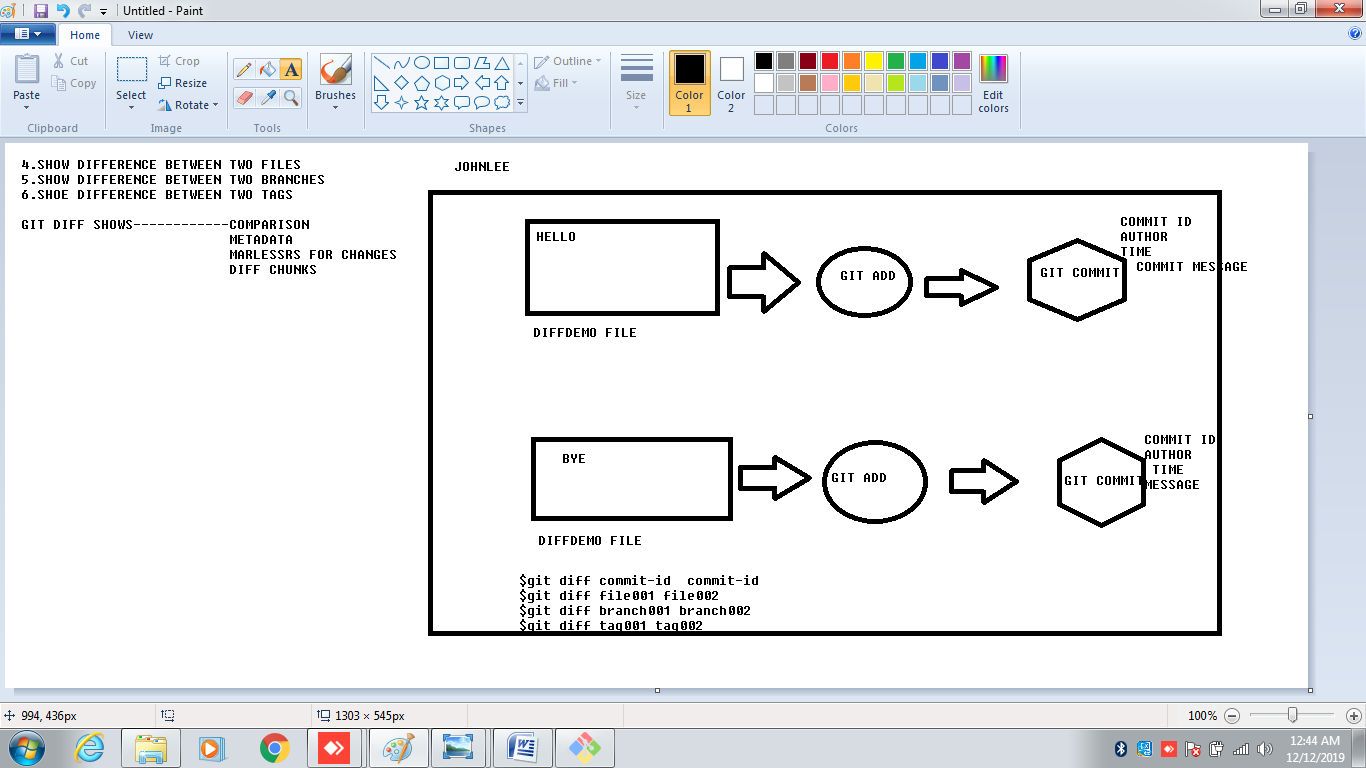
6.SHOE DIFFERENCE BETWEEN TWO TAGS

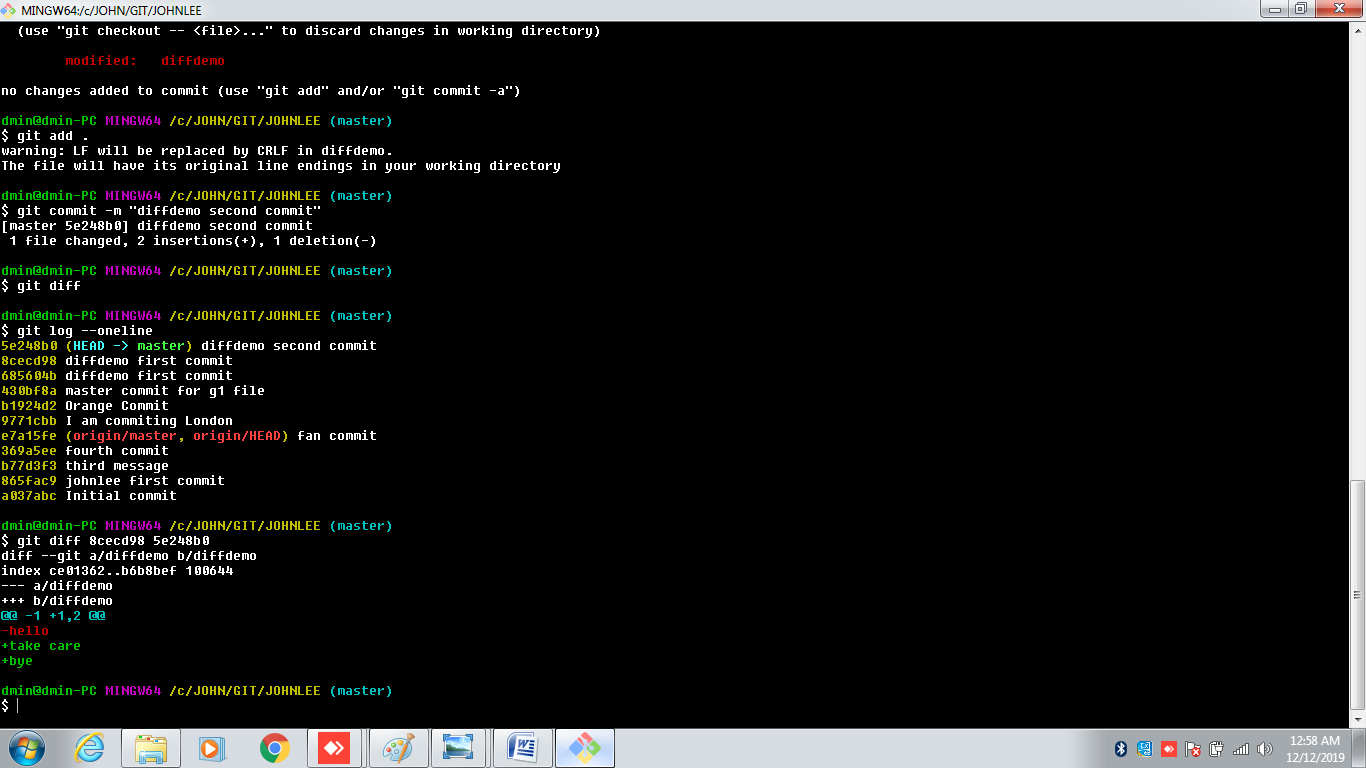
GIT DIFF SHOWS------------COMPARISON

METADATA

MARLESSRS FOR CHANGES

DIFF CHUNKS





**diff --git a/diffdemo b/diffdemo**

**index ce01362..b6b8bef 100644**

**--- a/diffdemo**

**+++ b/diffdemo**

@@ -1 +1,2 @@

-hello

+take care

+bye

GIT MERGE

---------------

GIT MERGE IS USED TO COMBINE TWO BRANCHES

--------------------------------------------

WILL COMBINE MULTIPLE SEQUENCES OF COMMITS INTO ONE UNIFIED HISTORY

----------------------------------------------------------

$GIT MERGE BRANCHNAME

-------------------------------------

TWO TYPES

A. FAST FORWARD MERGE

B. THREE-WAY MERGE

-----------------------------------

FAST FORWARD MERGE

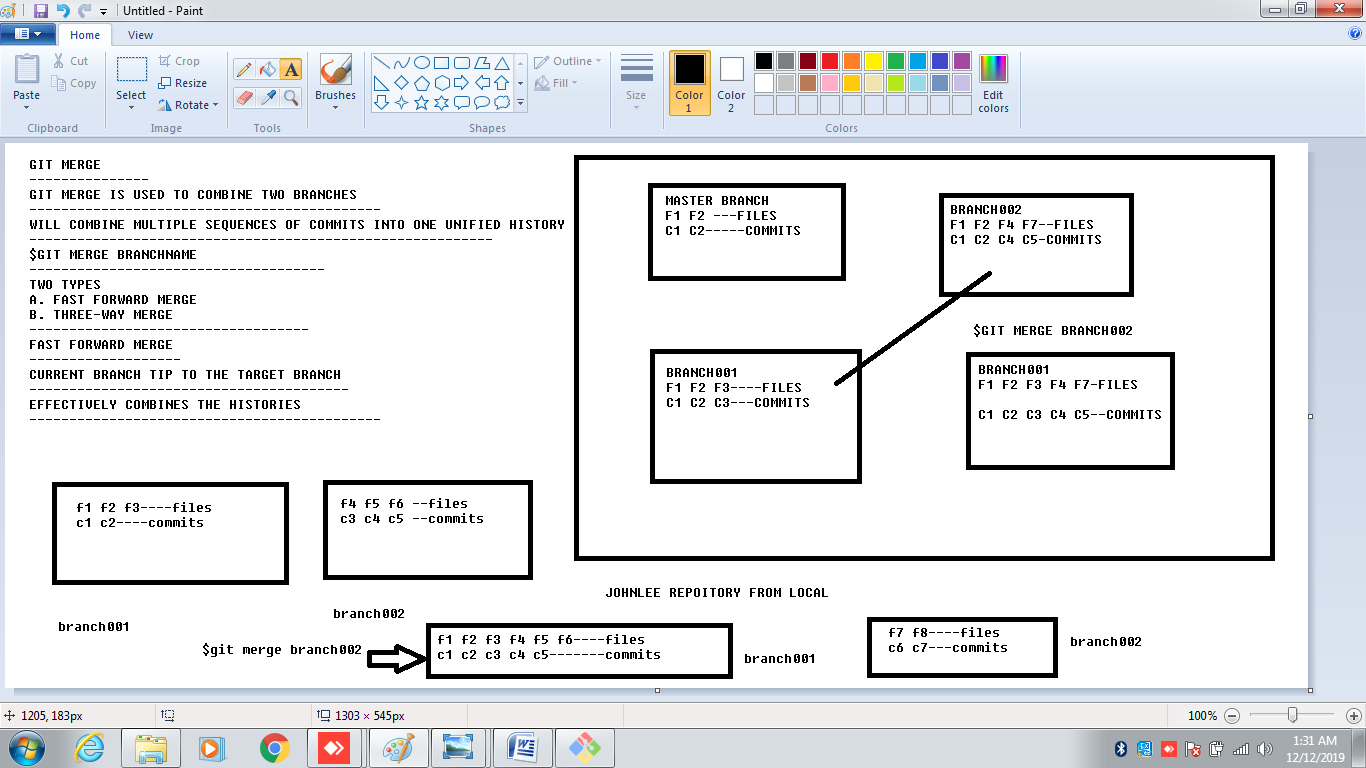
-------------------

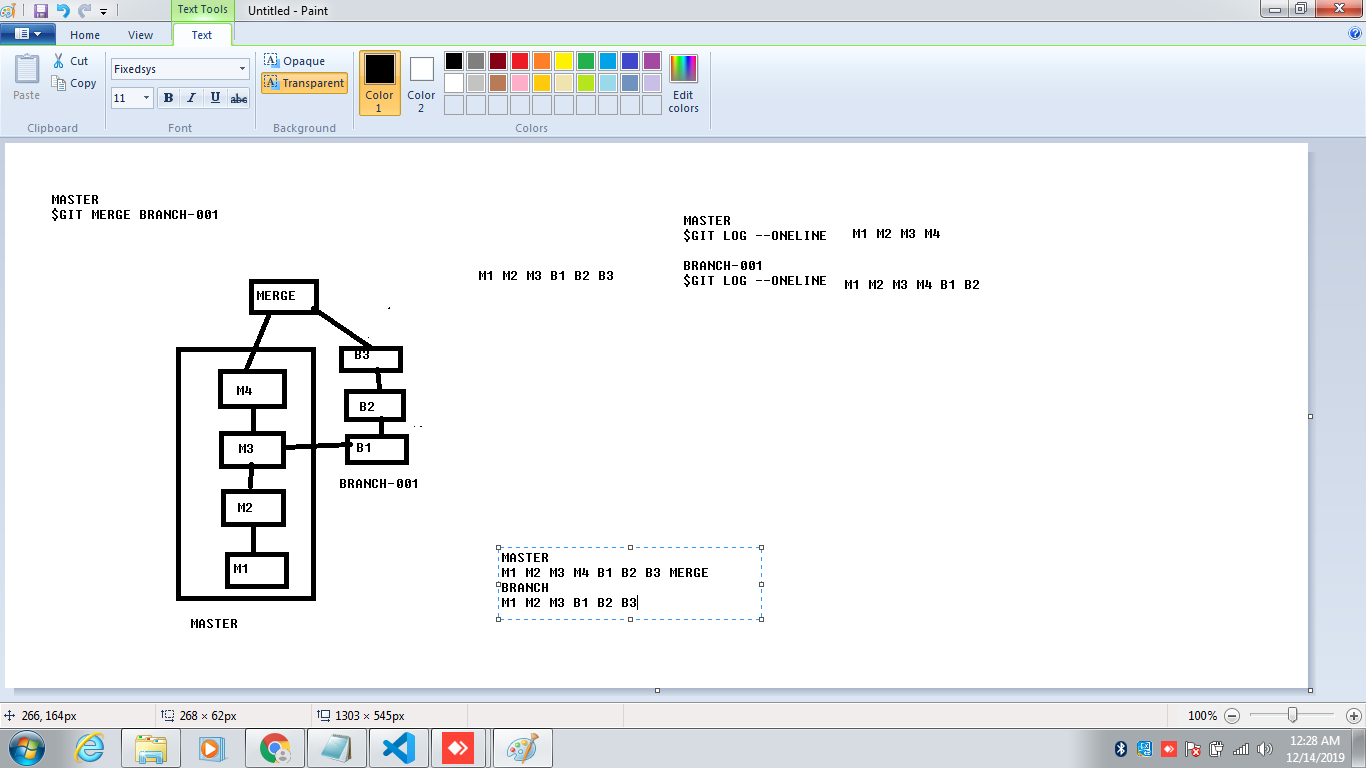
CURRENT BRANCH TIP TO THE TARGET BRANCH

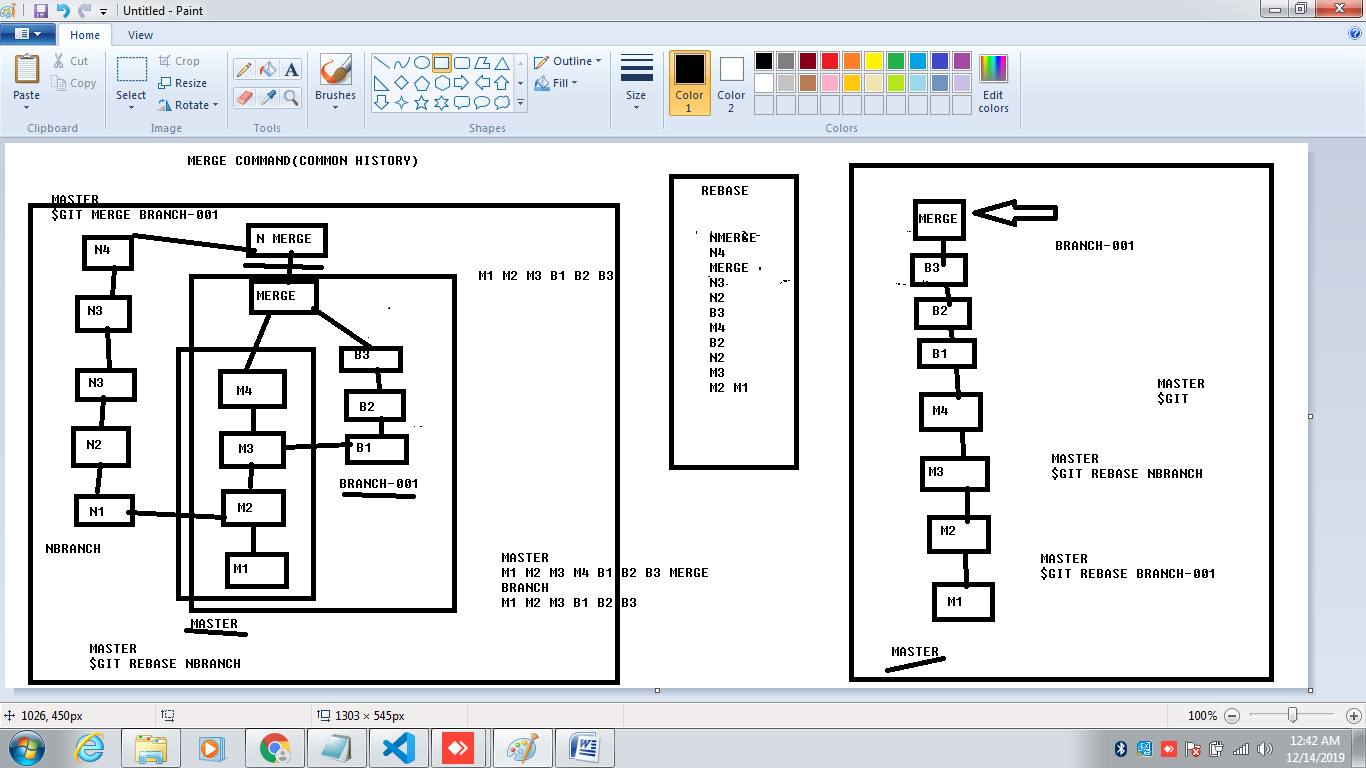
----------------------------------------

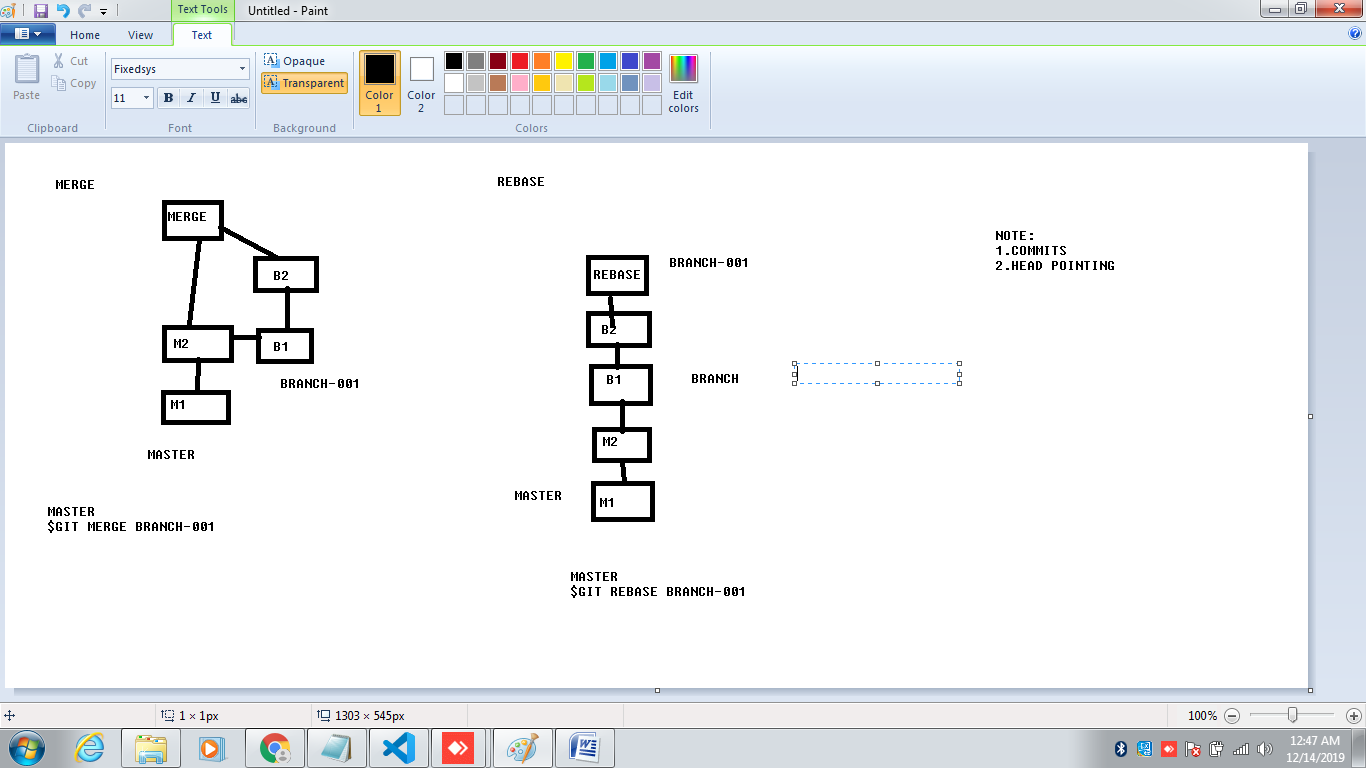
EFFECTIVELY COMBINES THE HISTORIES

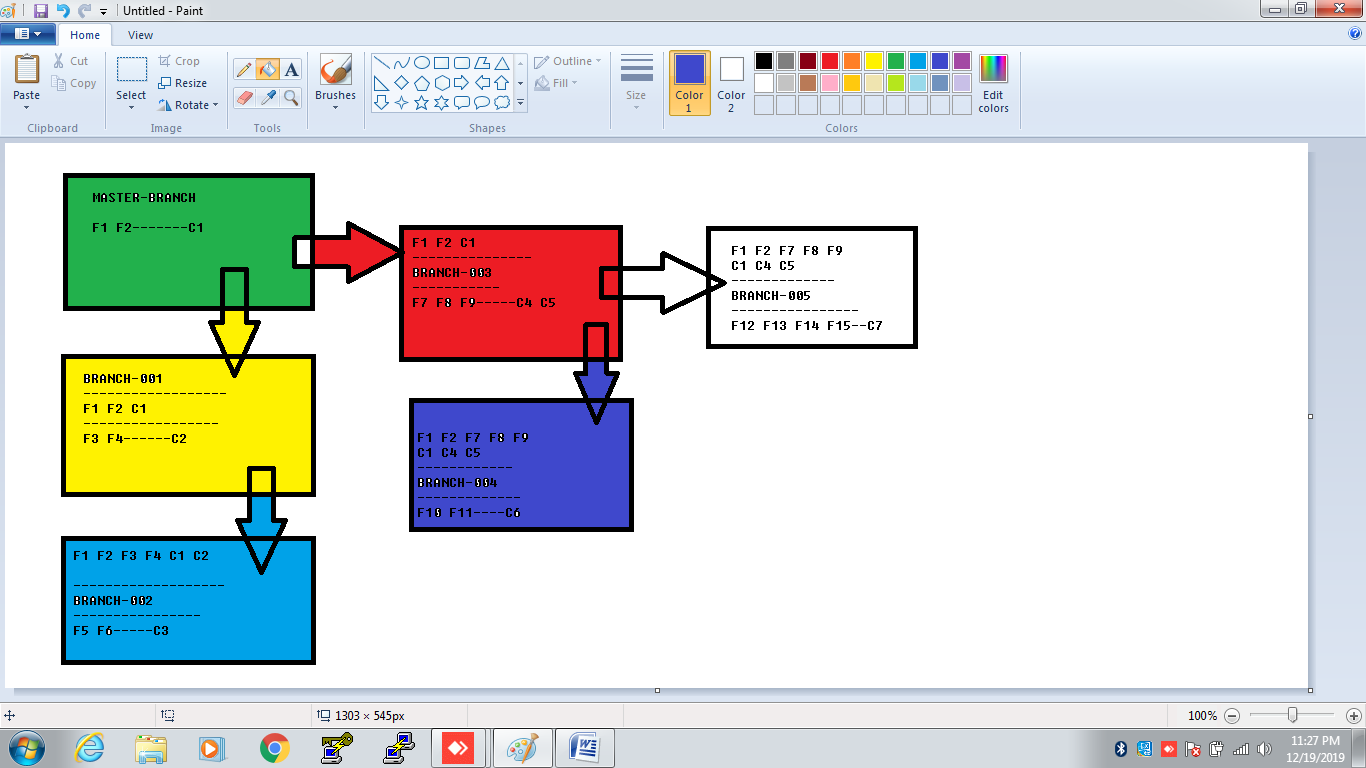
--------------------------------------------

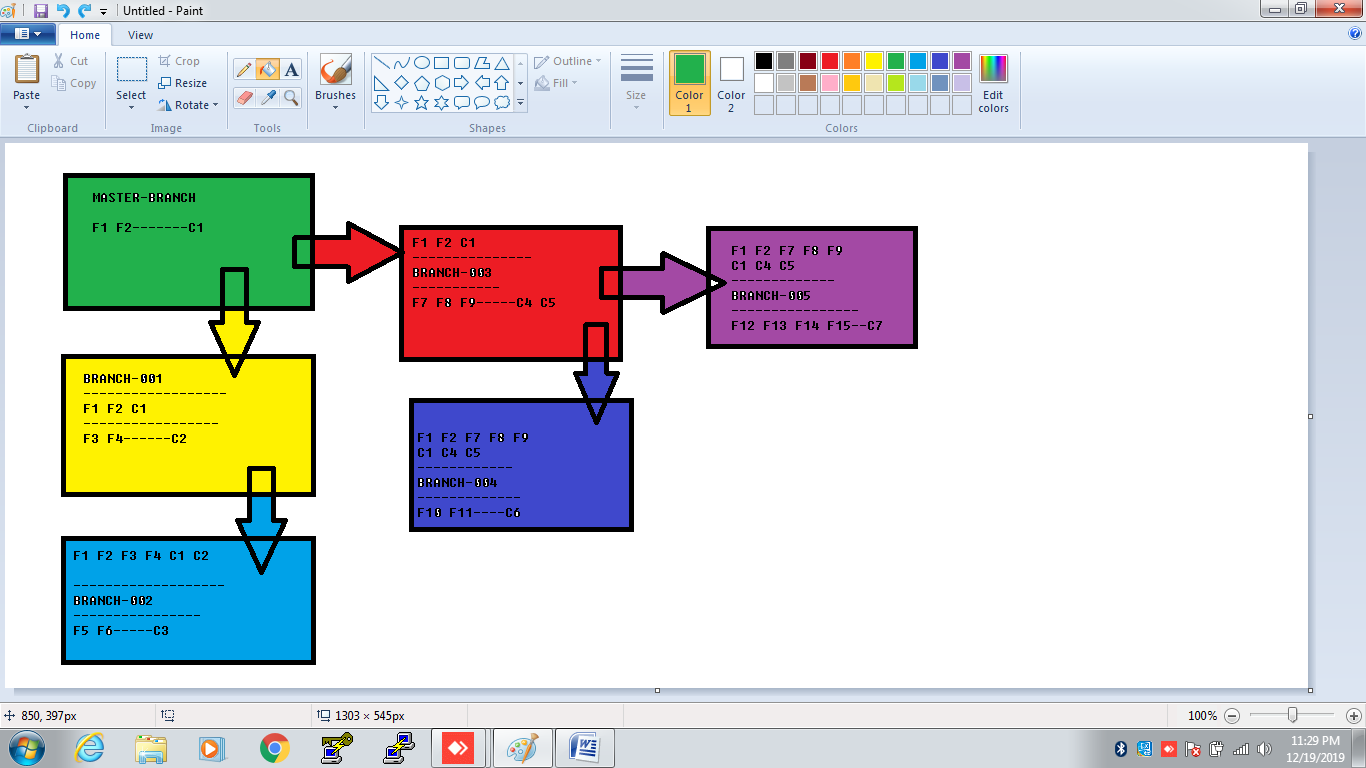


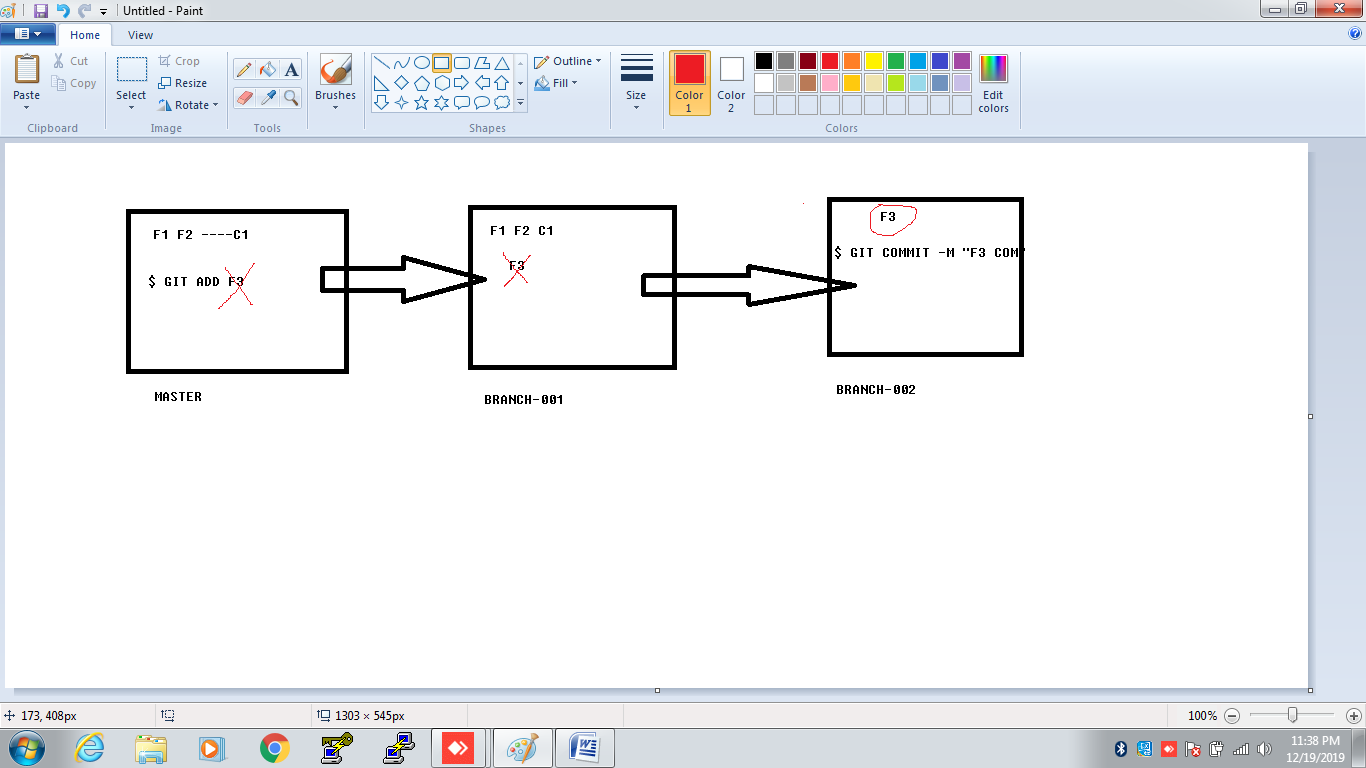












1.WE cant create another branch until and unless we create a single commit in Master branch

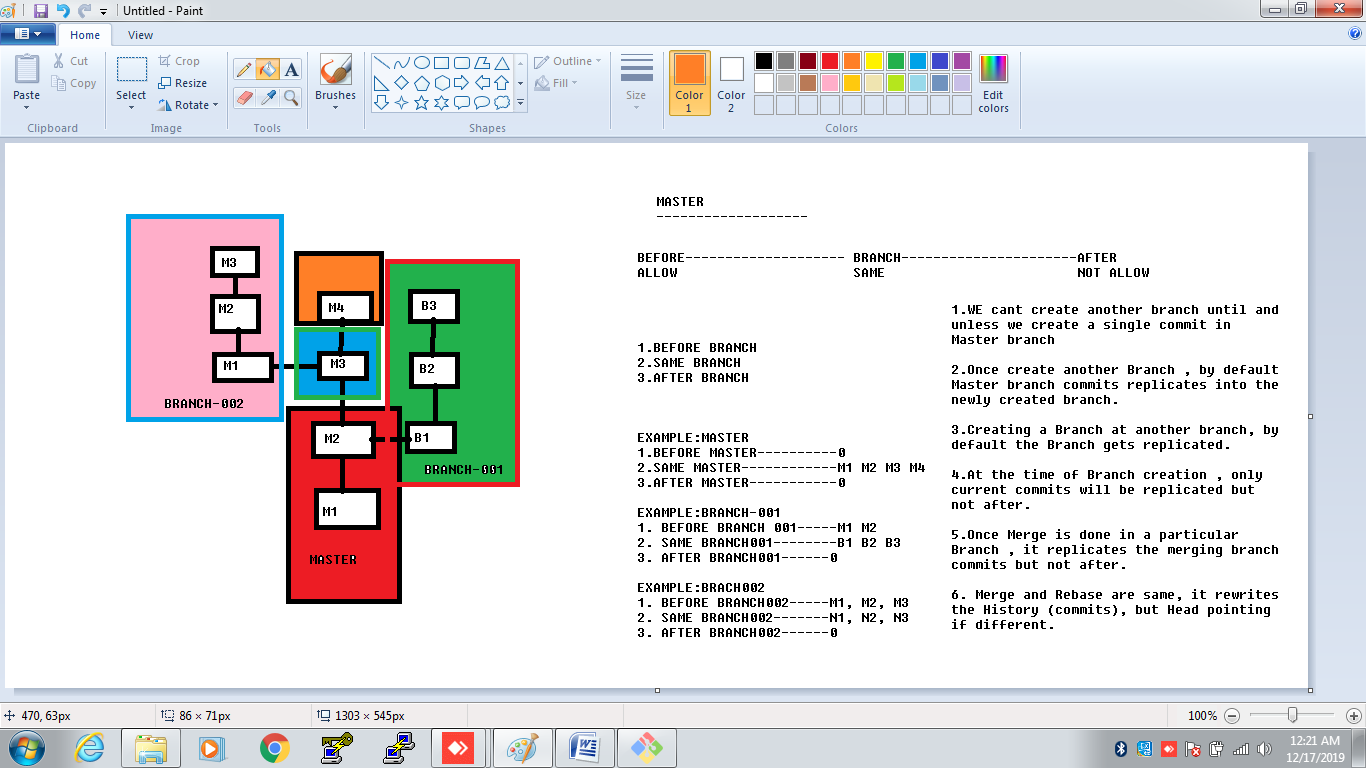
2.Once create another Branch , by default Master branch commits replicates into the newly created branch.

3.Creating a Branch at another branch, by default the Branch gets replicated.

4.At the time of Branch creation , only current commits will be replicated but not after.

5.Once Merge is done in a particular Branch , it replicates the merging branch commits but not after.

6. Merge and Rebase are same, it rewrites the History (commits), but Head pointing if different.



1. $git init

2. $git clone URL

3. $touch filename

4. $git add .

5. $git add file1 file2 file3

6. $git commit -m "commit text"

7. $git log -oneline

8. $git log

9. $git status

10. $git push origin master

11. $git pull origin master

12. $git reset filename

13. $git reset --hard HEAD^

14. $git reset --hard HEAD~3

15. $git commit --amend -m "newcommit message" (To chnage the latest commit message)

16. $git branch

17. $git branch branchname

18. $git checkout branchname

19. $git branch -m newbranchname (rename)

20. $git branch -d branchname

21. $git branch -D branchname

22. $git checkout -b branchname ( To create and checkout to new branch)

23. $git branch -a

24. $git remote add remotename URL

25. $git remote

26. $git remote -v ( complete date of remotes)

27. $git remote rename oldname newname

28. $git remote remove remotename

29. $git remoteshow remotename

30. $git remote prune remotename

31. $git push remotename branchname

32. $git pull remotename branchname

33. $git push remotename --delete branchname

33a. $git push remotename :branchname

34. $git merge branchname

35. $git rebase branchname

36. $git diff commit-id1 commit-id2

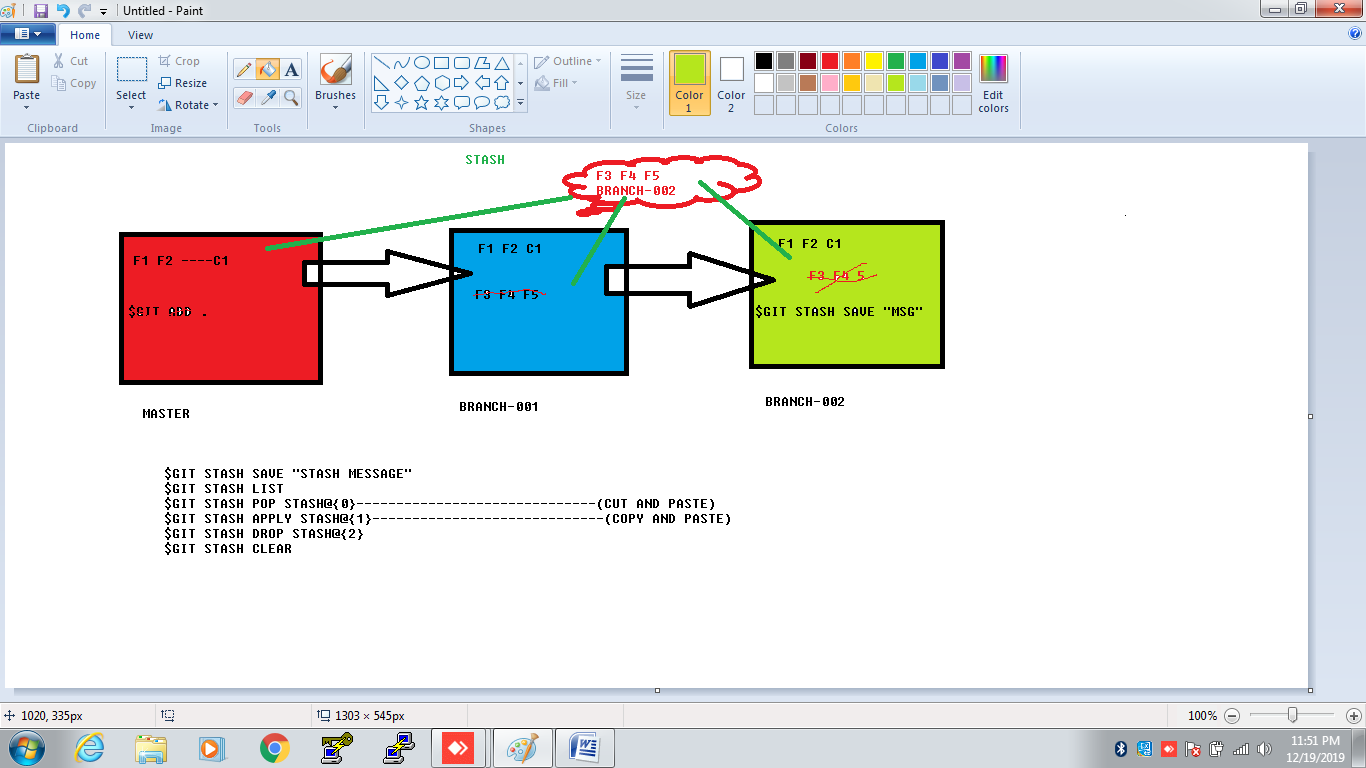
37. 1. $git init

37. $git diff file001 file002

38. $git diff tag001 tag002

39. $git show commitid

40. $git push -u origin master (



STASH

--------------------------------------

$GIT STASH SAVE "STASH MESSAGE"

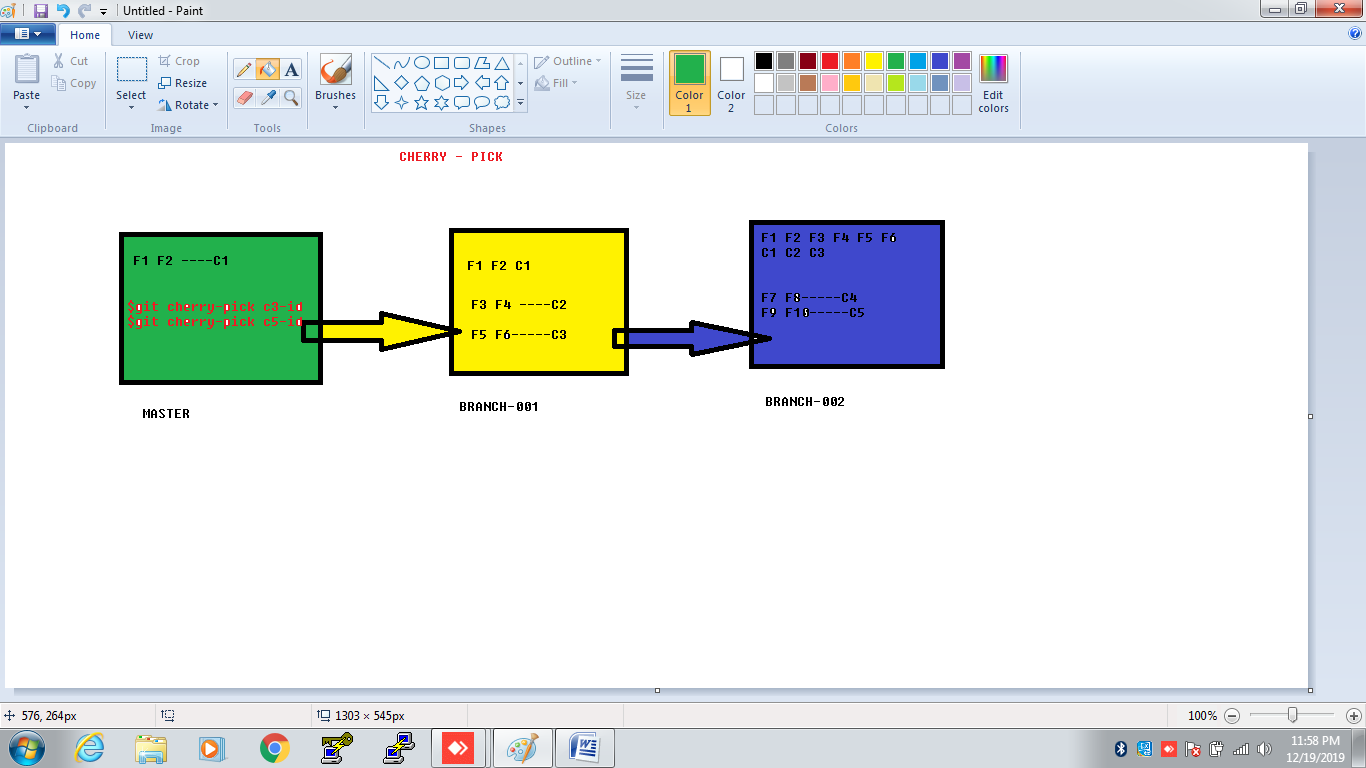
$GIT STASH LIST

$GIT STASH POP STASH@{0}------------------------------(CUT AND PASTE)

$GIT STASH APPLY STASH@{1}-----------------------------(COPY AND PASTE)

$GIT STASH DROP STASH@{2}

$GIT STASH CLEAR



CHERRY-PICK

--------------------

MASTER

--------------

F1 F2 F3

COMMIT-001

COMMIT-002

BRANCH-001

----------

F4 F5

COMMIT-003

COMMIT-004

BRANCH-002

------------

F6 F7 F8

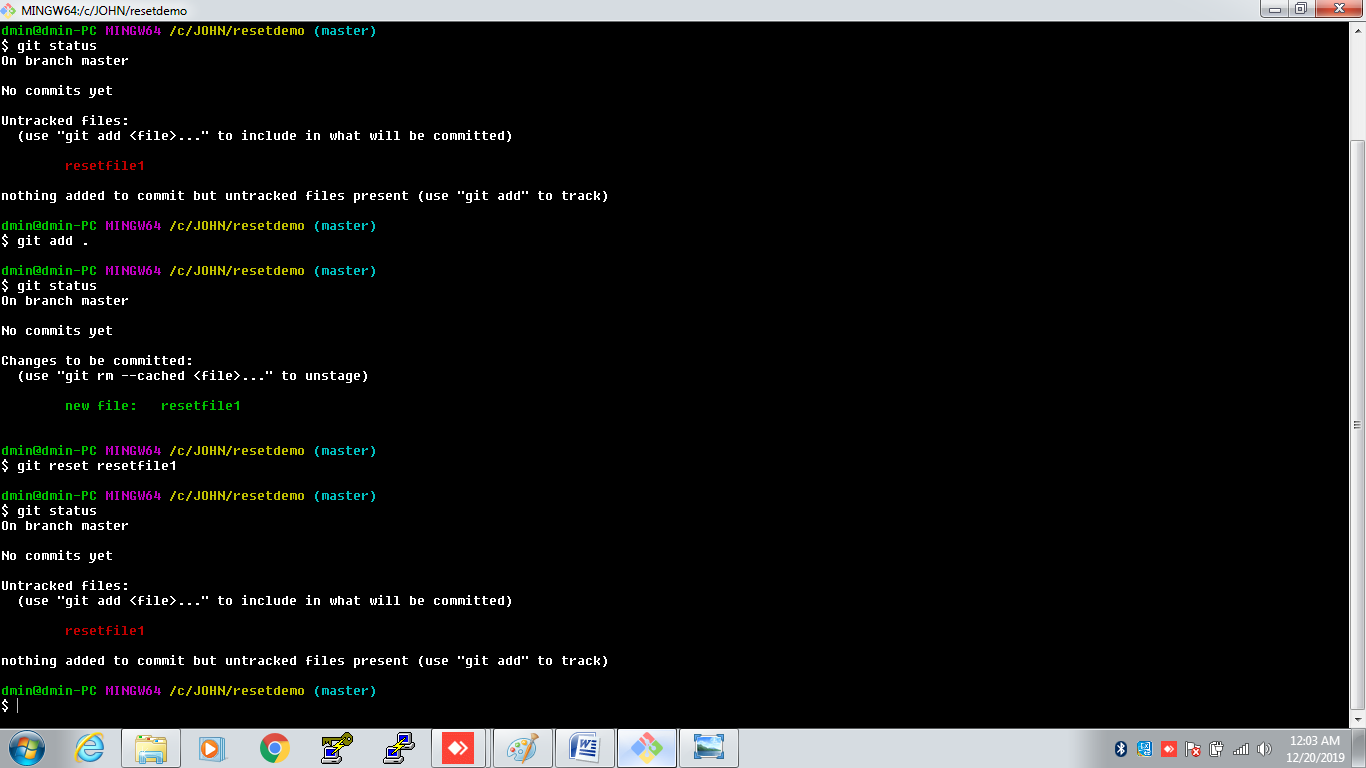
COMMIT-005

COMMIT-006

MASTER

--------------

$GIT CHERRRY-PICK COMMIT-006-ID



$git reset --hard HEAD^

$git reset --hard HEAD~3

REMOVE(rm):

git----remodemo-----$git init

stage:001

$touch f1 f2 f3

$git status

$rm f1 f2 f3

----it will all files and working tree was clean

------------------------------------------------------

Stage :002

$touch f1 f2 f3

$git status

$git add .

$rm f1 f2 f3

tracked files-----f1 f2 f3

deleted files-----f1 f2 f3----(untracked files)

$git commit -m "f1,f2,f3 are committed"

$git add .

tracked files---f1 f2 f3

$git commit -m "f1,f2,f3 are again committed"

--------------------------------------------

Stage:003

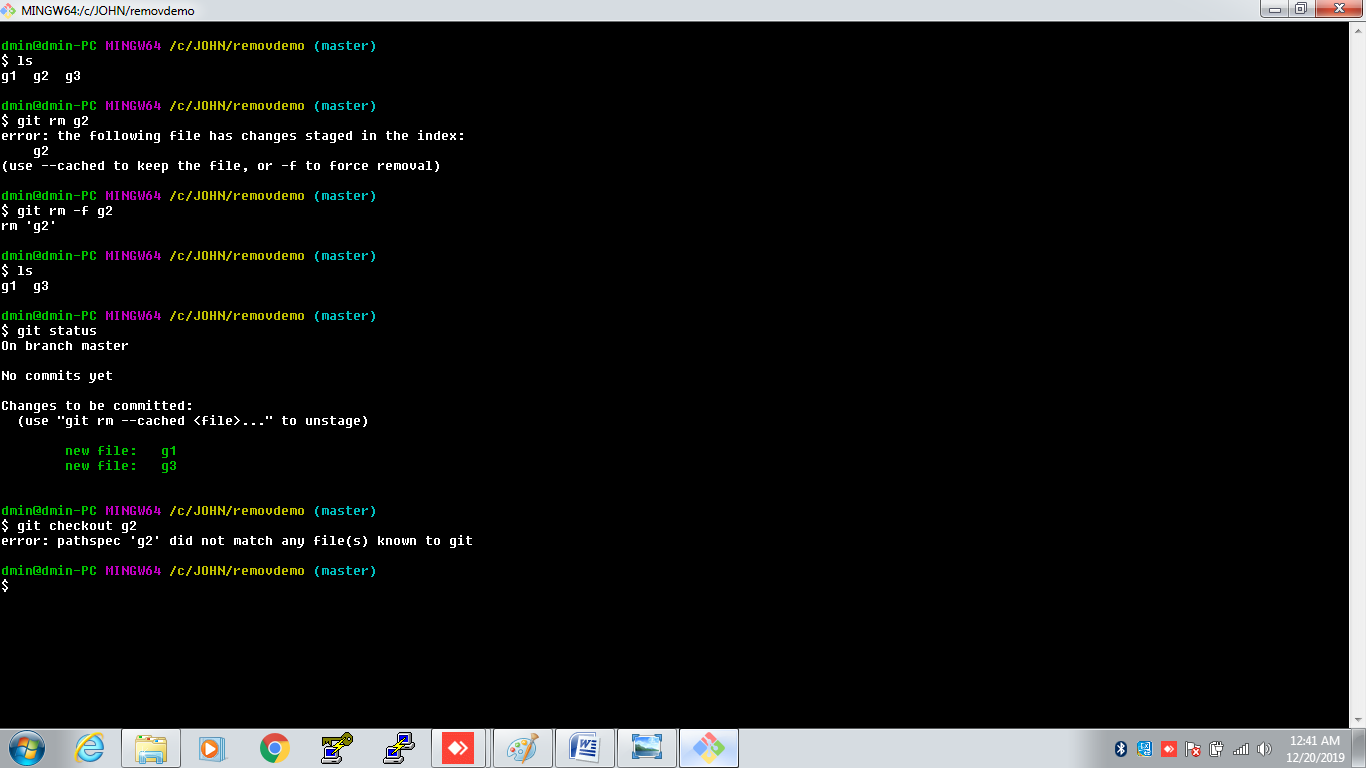
$touch f1 f2 f3

$git add .

$git rm f2

$git status

f1 f3 -------------------tracked files



MOVE(mv):

git----movedemo-----$git init

$touch f1 f2 f3

$git status

$mv f1 f2 f3

----it will all files and working tree was clean

------------------------------------------------------

$touch f1 f2 f3

$git status

$git add .

$mv f1 f2 f3

tracked files-----f1 f2 f3

deleted files-----f1 f2 f3----(untracked files)

$git commit -m "f1,f2,f3 are committed"

$git add .

tracked files---f1 f2 f3

$git commit -m "f1,f2,f3 are again committed"

--------------------------------------------

$touch f1 f2 f3

$git add .

$git mv f2

$git status

f1 f3 -------------------tracked files

