**Task – 1**

**● Create a new folder**

**● Put the following files in the folder**

**○ Code.txt ○ Log.txt ○ Output.txt**

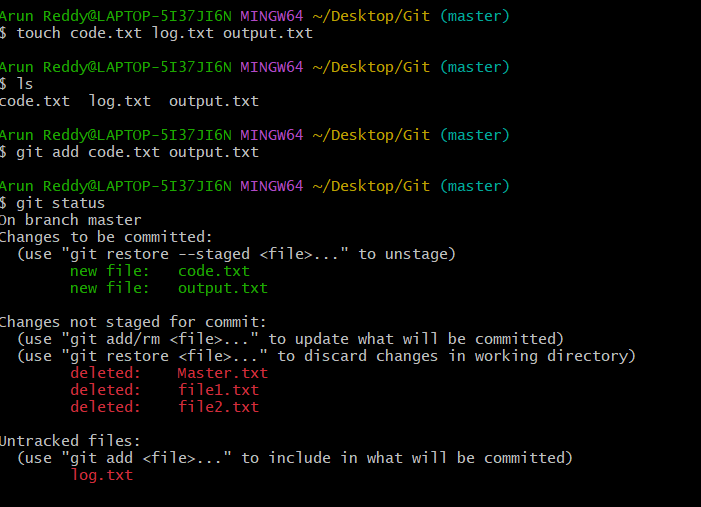
**● Stage the Code.txt and Output.txt files**

**● Commit them**

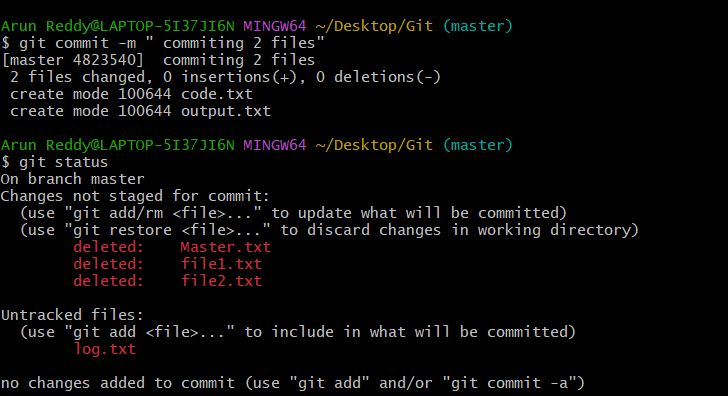
**● And finally push them to github Please share the commands for the above points**

Commands:

* Mkdir Git
* Git init
* Touch code.txt output.txt Log.txt
* Git add code.txt output.txt



* Git commit -m “first commit”



After creating the git repository , follow the below commands to push the data from local to global



* Git remote -v 🡪 to check white repos we are in
* Git remote add origin repo-url 🡪 adding the remote repo named “**origin”**



* Git push origin master 🡪 to push the data from local to global

Commands used in bash:

* mkdir Git
* cd Git
* git init
* touch code.txt log.txt output.txt
* ls
* git add code.txt output.txt
* git status
* git commit -m " commiting 2 files"
* git status
* git status
* git remote -v
* git remote add origin https://github.com/ArunReddy1012/Git-Assignment1.git
* git remote -v
* git push origin master
* ls

**Files added/ pushed into git remote repository**

**Task – 1 completed.**

**Task-2**

**● Create a git working directory with feature1.txt and feature2.txt in the master branch**

**● Create 3 branches develop, feature1 and feature2**

**● In develop branch create develop.txt, do not stage or commit it**

**● Stash this file, and checkout to feature1 branch**

**● Create new.txt file in feature1 branch, stage and commit this file**

**● Checkout to develop, unstash this file and commit**

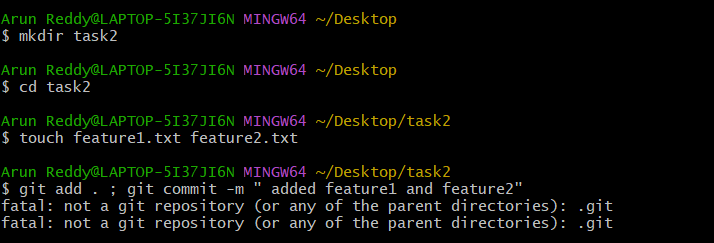
**Mkdir task2**

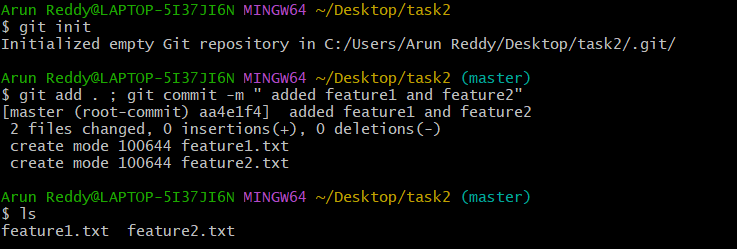
**Cd ttask2**

**Git init**

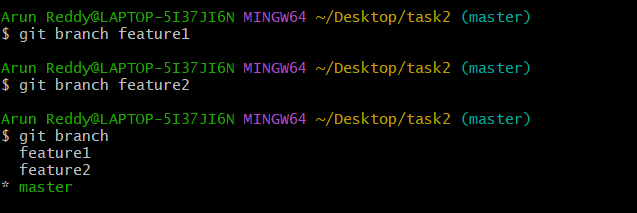
**Touch feature1.txt feature2.txt**

**Git add .; git commit -m “jgwdk”**

****

****

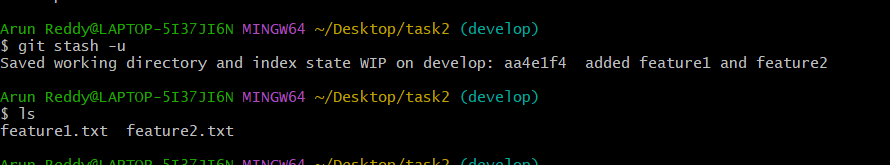
**Git branch feature1 and 2 develop**



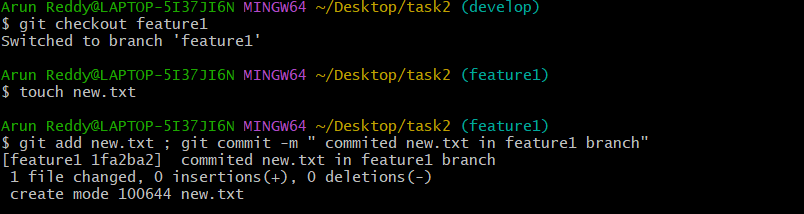
* **Git checkout develop**
* **Touch develop.txt**



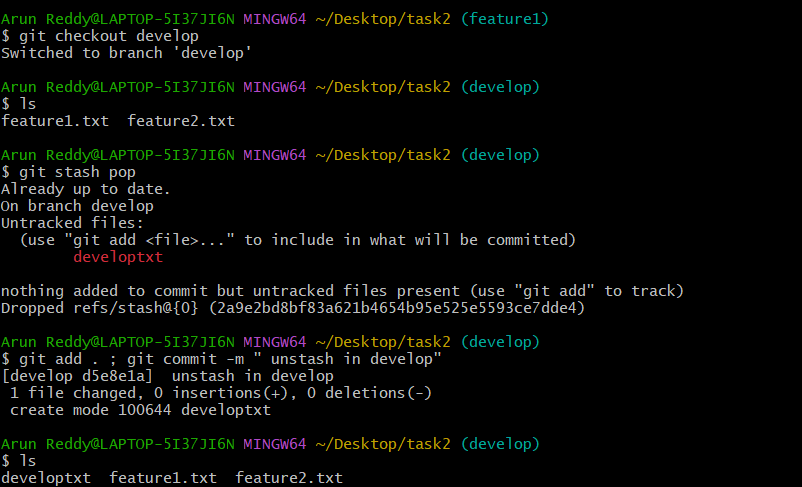
* Git stash -u



* **Git checkout feature1**
* **Touch new.txt**
* **Git add new.txt ; git commit -m “hi”**



* Git checkout develop
* Git stash pop 🡪 unstash all the files



Commands :

* 436 mkdir task2
* 437 cd task2
* 438 touch feature1.txt feature2.txt
* 439 git add . ; git commit -m " added feature1 and feature2"
* 440 git init
* 441 git add . ; git commit -m " added feature1 and feature2"
* 442 ls
* 443 git add . ; git commit -m " added feature1 and feature2"
* 444 git branch feature1
* 445 git branch feature2
* 446 git branch
* 447 git branch develop
* 448 git branch
* 449 git checkout develop
* 450 touch developtxt
* 451 ls
* 452 git stash -u
* 453 ls
* 454 git checkout feature1
* 455 touch new.txt
* 456 git add new.txt ; git commit -m " commited new.txt in feature1 branch"
* 457 ls
* 458 git checkout develop
* 459 ls
* 460 git stash pop
* 461 git add . ; git commit -m " unstash in develop"
* 462 ls

**Task – 2 completed.**

**Task – 3**

● Create a git working directory, with the following branches

○ Develop ○ F1 ○ f2

● In the master branch, commit main.txt file

● Put develop.txt in develop branch, f1.txt and f2.txt in f1 and f2 respectively

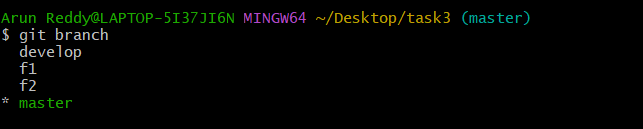
● Push all these branches to github

● On local delete f2 branch

● Delete the same branch on github as well

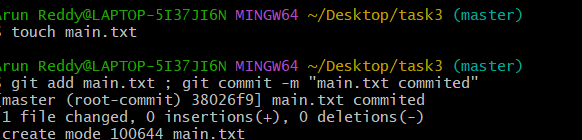
Commands:

* Mkdir task3
* Cd task3
* Git init
* Touch file.txt
* Git add . ; git commit -m “ first commit”
* Git branch develop
* Git branch f1
* Git branch f2



Touch main.txt (in master)

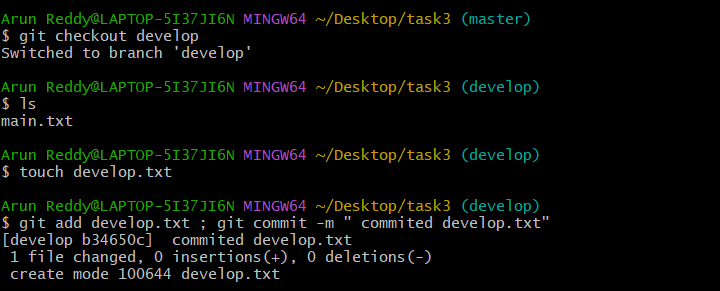
Git add . ; git commit -m “ main.txt commited”



Git checkout develop

Touch develop.txt (in develop)

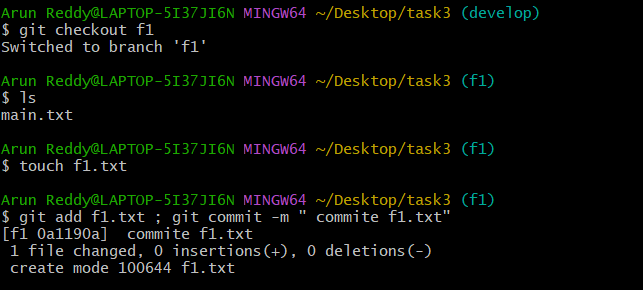
Git add . ; git commit -m “ develop.txt commited”



Git checkout f1

Touch f1.txt (in f1)

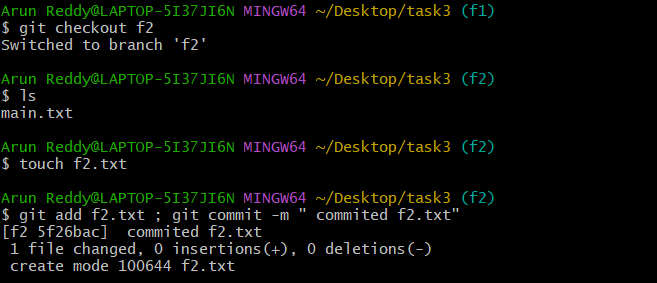
Git add . ; git commit -m “ f1.txt commited”



Git checkout f2

Touch f2.txt (in f2)

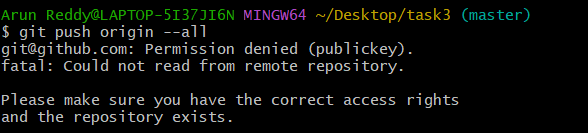
Git add . ; git commit -m “ f2.txt commited”



Git remote add origin [git@github.com:ArunReddy1012/Git-Assg-1.3.git](mailto:git@github.com:ArunReddy1012/Git-Assg-1.3.git) --> adding a remote repo

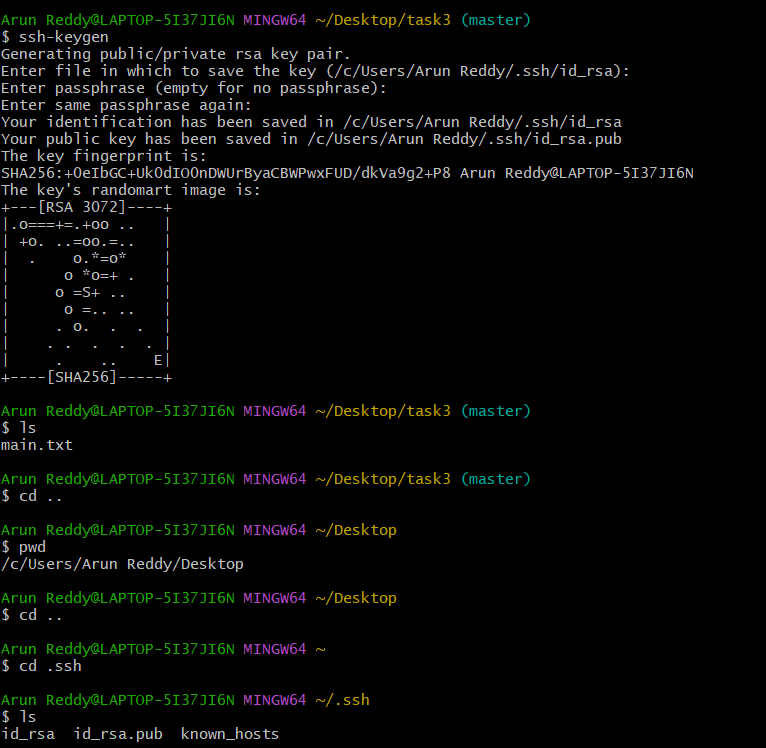
Git push --all 🡪 to push all the branches

* We got the permission denied because, we still haven’t created key and added to our github



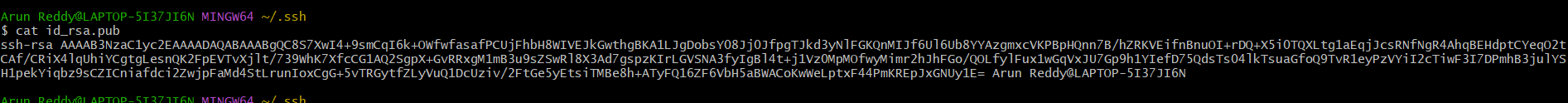


ssh-keygen 🡪 generating key(It will generate 2 keys id\_rsa and id\_rsa.pub)

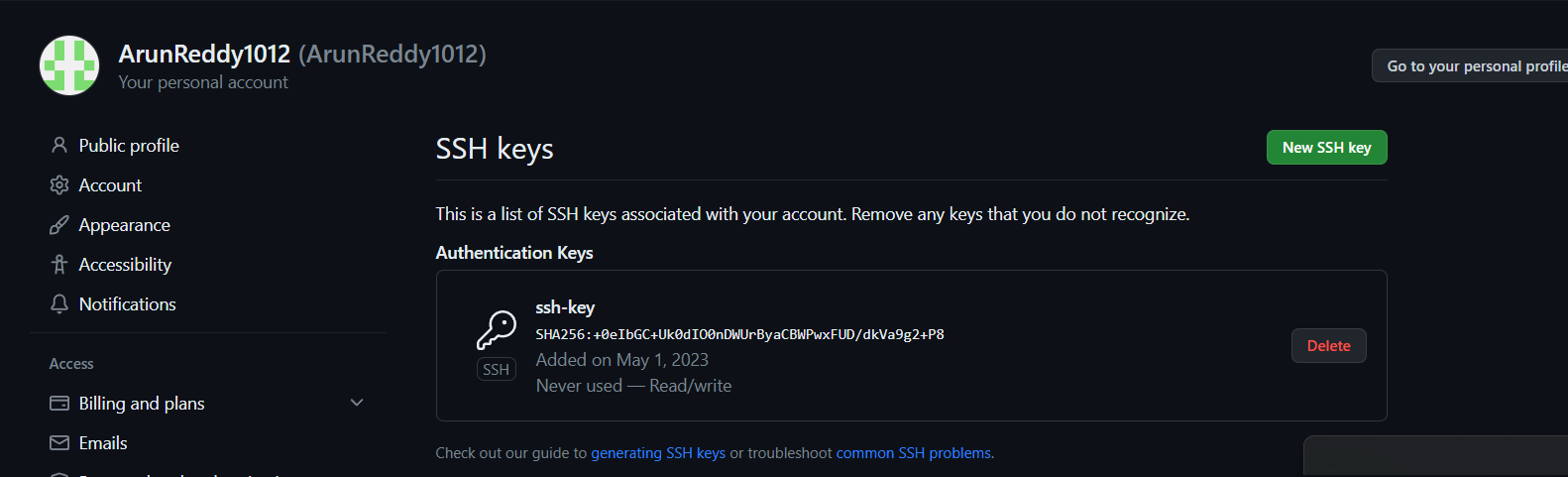




cat id\_rsa.pub (copy the path and paste it in git hub ssh key)



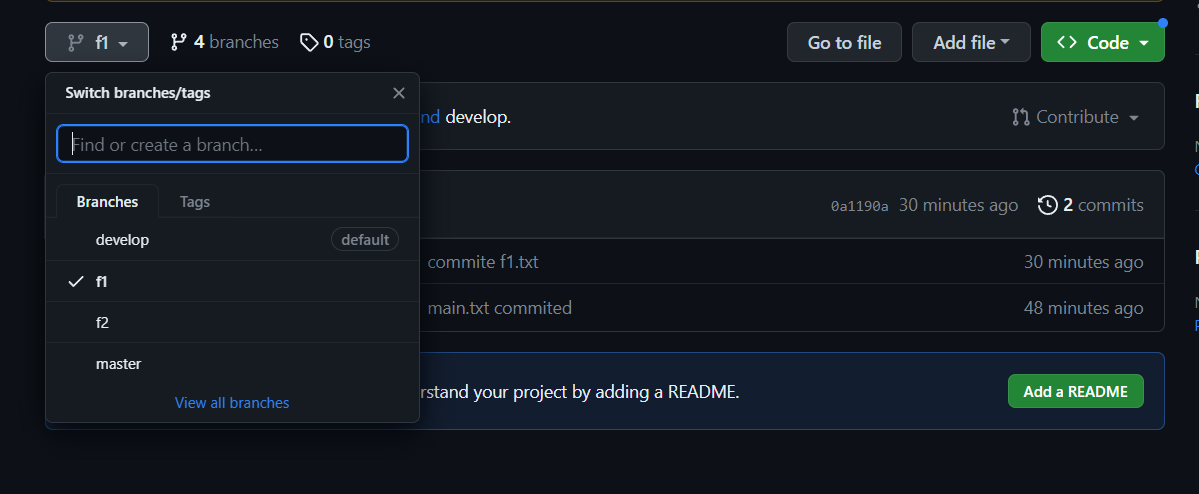




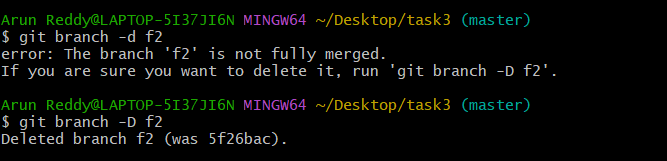


Now in git bash, go back to the working directory and push it

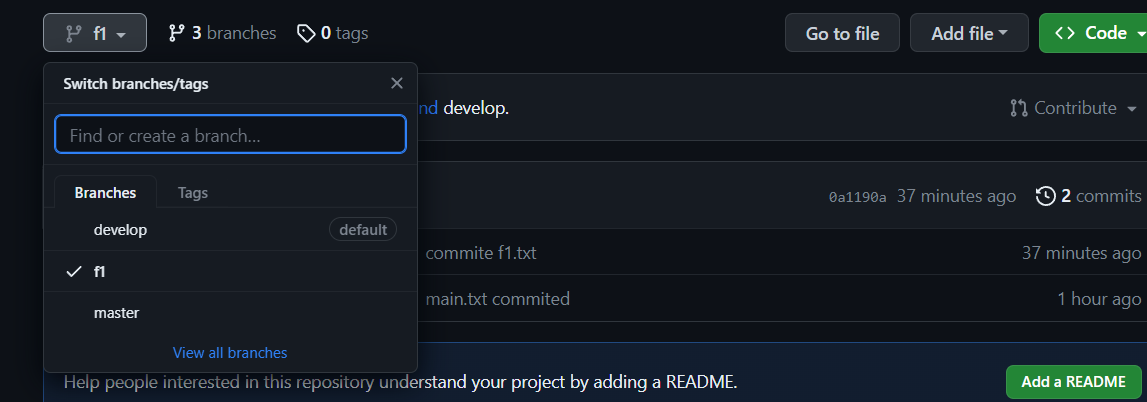
All the four branches were pushed into git hub

****

git branch -D f2 🡪 to delete f2 branch in local

****

git push origin --delete f2 🡪 to delete f2 branch in remote .



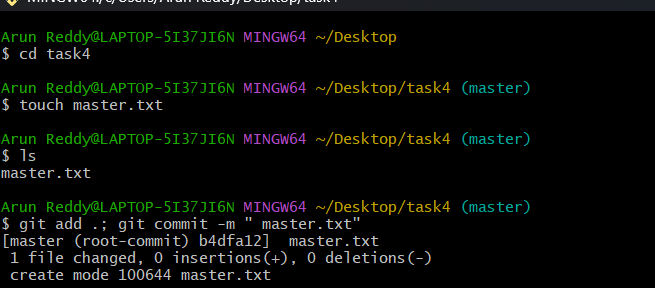
**Task - 3 completed.**

**Task – 4**

* Put master.txt on master branch, stage and commit
* Create 3 branches: public1, public2 and private
* Put public1.txt on public 1 branch, stage and commit
* Merge public 1 on master branch
* Merge public 2 on master branch
* Edit master.txt on private branch, stage and commit
* Now update branch public 1 and public 2 with new master code in private
* Also update new master code on master
* Finally update all the code on the private branch

Commands:

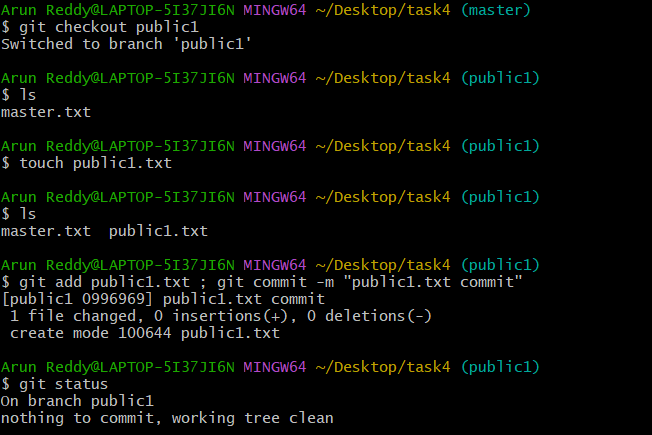
* Mkdir task4
* Cd task4
* Git init
* Touch master.txt
* Git add . ; git commit -m “ commit master.txt”



Create three branches:



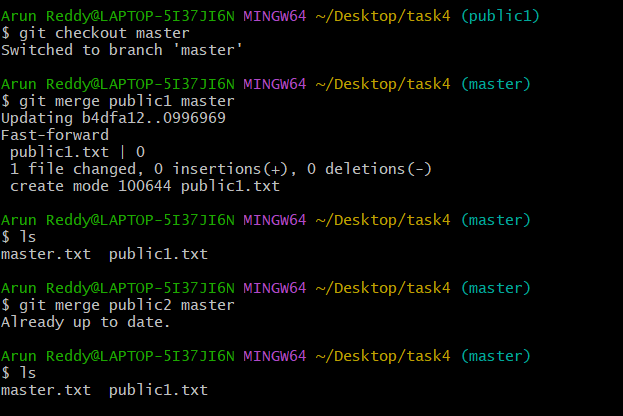
* Put public1.txt on public1 branch, stage and commit



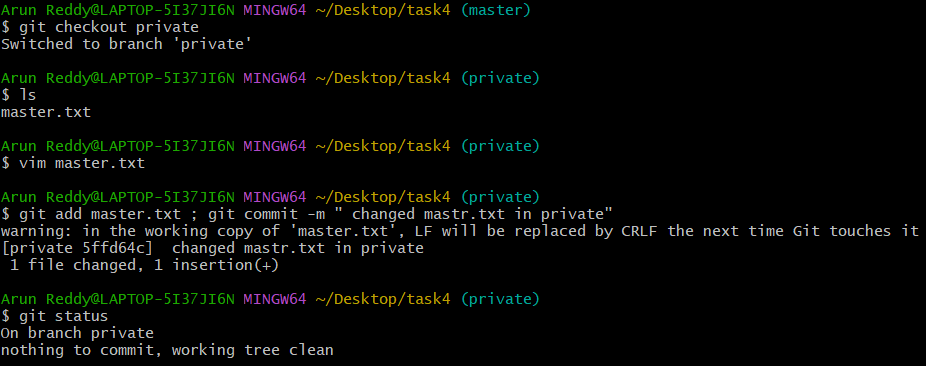


Checkout to master branch and perform

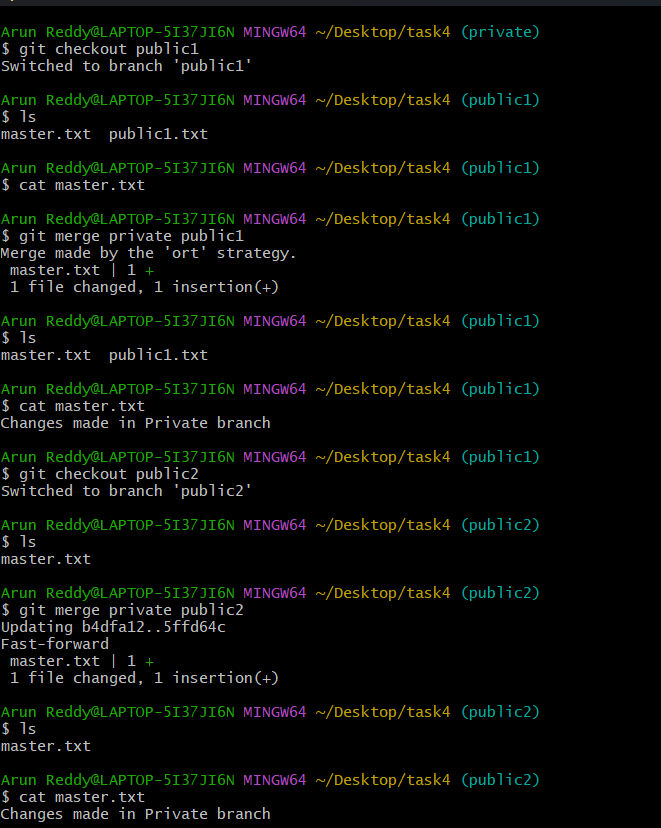
* Merge public 1 on master branch
* Merge public 2 on master branch



* Edit master.txt on private branch, stage and commit

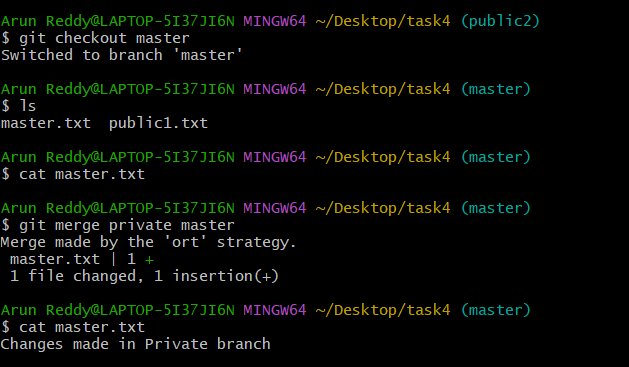


* Now update branch public 1 and public 2 with new master code in private





* Also update new master code on master

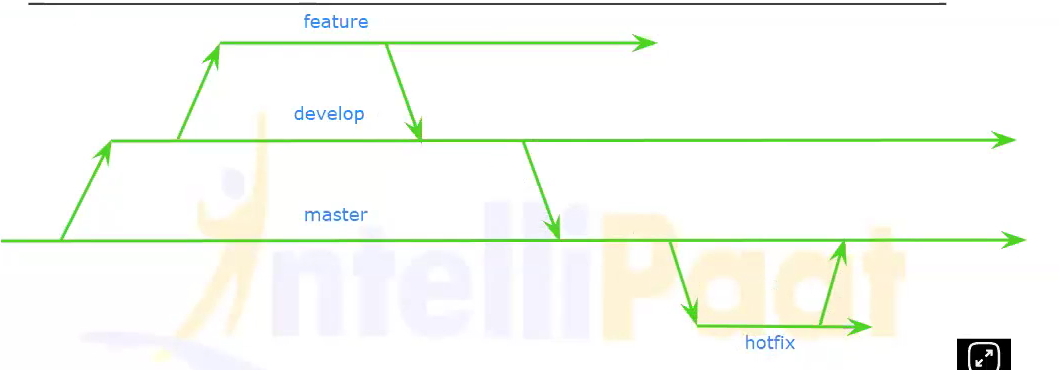


**Task – 4 is completed.**

**TASK – 5**

* **Create a gitflow workflow architecture on git**
* **Create all the required branches**
* **Starting from the feature branch, push the branch to the master, following the architecture**
* **Push a urgent.txt on master using hotfix**

**Architecture**

****

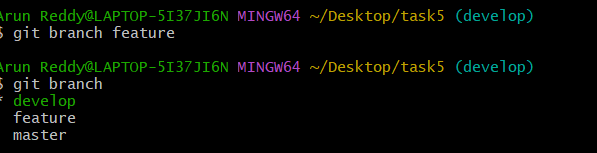


**Commands:**

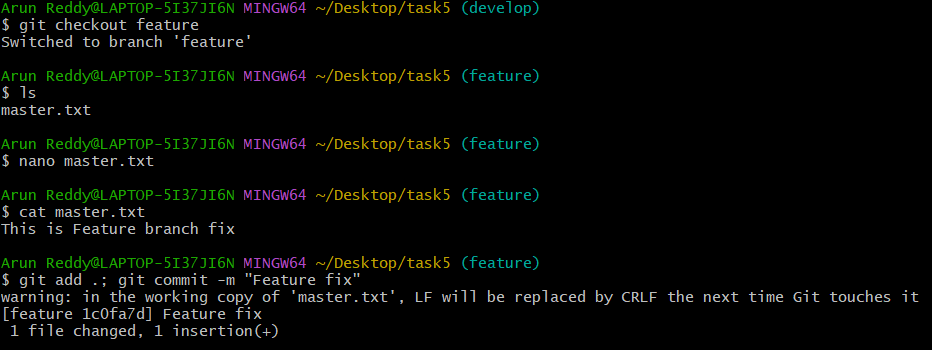
**Create the architecture where we have the master branch, we create the develop branch, from the develop branch we create feature branch on which we going to work on some features**

* **Create a master branch**
* **Git branch develop**
* **Git branch feature**

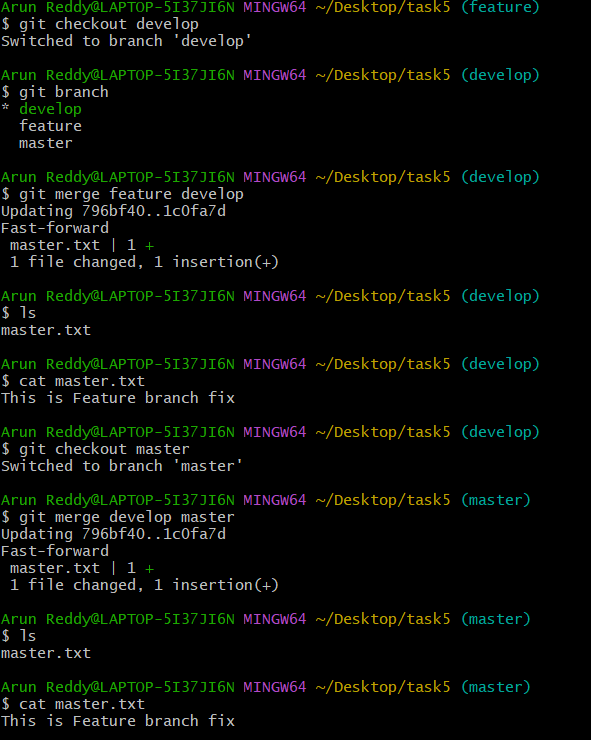
****

****

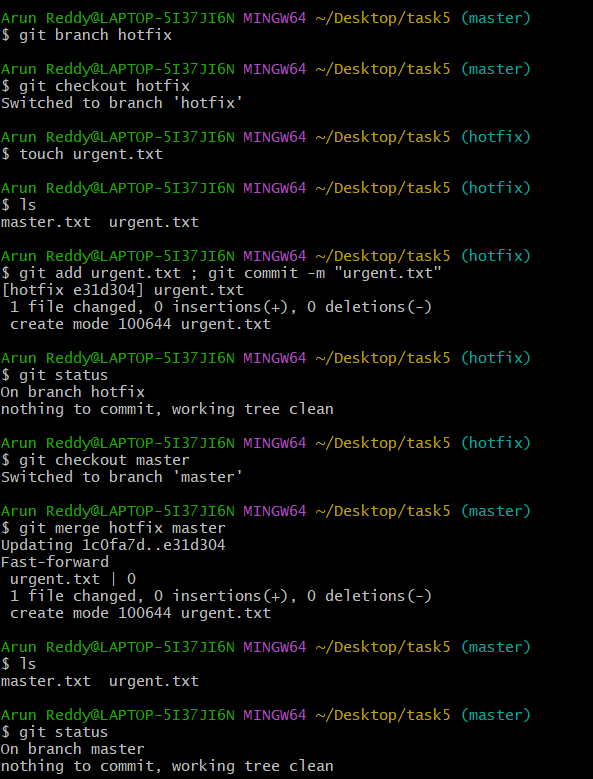
* **In Feature branch edit the master.txt file after that stage and commit the file**

****

* **Merge feature with develop and develop with Master**

****

* **Git branch hotfix**
* **Git checkout hotfix**
* **Touch urgent.txt**
* **Git add urgent.txt ; git commit -m “urgent.txt commit”**
* **Git checkout master**
* **Git merge hotfix master**

****

**TASK – 5 completed.**