CourseCube
(Formerly Java Learning Center)

DevOps

Module 3

Working with GIT/GitHub
Part 2

Author
Srinivas Dande

Java Learning Center

18. Branches and Merging

Architect Tasks:

Task 13: Setup Remote repositoty

In this Task, You will do

1) Create Empty Remote Repository in GitHub
2) Create the Local Repository
3) Add the Remote Origin to local Repository
4) Push to Remote Repository
5) Doing Commits to local master
6) Pushing to Remote master

1) Create Empty Private Repository in GitHub with name **myjlc-bookstore-1** without any files
2) Do the following in your local Machine

```
mkdir myjlc-bookstore-1
cd myjlc-bookstore-1
echo "# myjlc-bookstore-1" >> README.md
git status
git init
git status
git add README.md
git status
git commit -m "my first commit - README.md added"
git status
```

3) Change the Branch name to **jlcmaster**

```
git branch -M jlcmaster
```

4) Add the Remote Origin

```
git remote add origin git@github.com:DandesClasses/myjlc-bookstore-1.git
```

5) Push to remote first time

```
git push -u origin jlcmaster
```

6) Write new File , commit and push

write Hello.java

```
class Hello{
public static void main(String as[]){
System.out.println("Hello Guys !!!");
}
}
```

```
git add Hello.java
git commit -m "my second commit  - Hello.java added"
git status

git push
```

7) Write new File , commit and push

write Hai.java

```
class Hai{
public static void main(String as[]){
System.out.println("Hai Guys !!!");
}
}
```

```
git add Hai.java
git commit -m "my third commit  - Hai.java added"
git status

git push
```

19. Working On Branch

Developer -1 Tasks:

Task 14: Working On Branch

In this Task, You will do

1) Clone the Repository
2) Create the Branch - **my-new-feature-1**
3) Doing Commits to **my-new-feature-1**
4) Pushing to Remote **my-new-feature-1**

1) Clone the Remote Repository

```
mkdir developer-1
cd developer-1
git clone git@github.com:DandesClasses/myjlc-bookstore-1.git
```

2) Practice the git branch related Commands

```
git branch
git branch -r
git branch -a
git checkout -b my-new-feature-1  //Create and Switch
git branch -a
git checkout -b jlcmaster  //Error
git checkout jlcmaster   //Switch
git branch -a
git checkout -b my-new-feature-1    //Error
git checkout my-new-feature-1 //Switch
```

3) Switch to my-new-feature-1 branch

```
git checkout my-new-feature-1 //Switch
```

4) Write your Code and Commit

```
Write New-Feature-1A.java
git status
git add New-Feature-1A.java
git status
git commit -m "my commit -1 for my-new-feature-1"
git status
```

5) Push to Remote Origin

```
git push --set-upstream origin my-new-feature-1
```

6) Write your Code Again and Commit

```
Write New-Feature-1B.java
git status
git add New-Feature-1B.java
git status
git commit -m "my commit -2 for my-new-feature-1"
git status
```

7) Push to Remote Origin

```
git push
```

8) Update Hello.java and Commit

```
Update Hello.java
git status
git add Hello.java
git status
git commit -m "my commit -3 for my-new-feature-1"
git status
```

9) Push to Remote Origin

```
git push
```

20. git diff command

Developer -1 Tasks:

Task 15: Clone Remote repositoty

In this Task, You will do

1) Difference between two commits
2) Difference between two branches

1) Difference between two branches

```
git diff   jlcmaster my-new-feature-1

jlcmaster + Hello Update + New File1A.java + New File 1B.java => my-new-feature-1
```

2) Difference between two branches

```
git diff   my-new-feature-1 jlcmaster

my-new-feature-1 - Hello Update - New File1A.java -New File 1B.java => jlcmaster
```

3) Difference between two commits

git diff db61bed a545f9c

git diff Commit -2 Commit - 3

Commit - 2 + Hello Update => Commit - 3

4) Difference between two commits

git diff a545f9c db61bed

git diff Commit - 3 Commit -2

Commit - 3 - Hello Update => Commit - 2

21. Branch Merging

Developer -1 Tasks:

Task 16: Merge feature Branch with master

In this Task, You will do

1) Swicth to master branch
2) Merge the feature branch with master

1) Check the commits of Feature branch

git log –oneline

jlcmaster branch 6 commits

2) First we should be in master branch to merge our branch with master

git checkout jlcmaster

git branch

3) Check the commits of Master branch

```
git log –oneline

jlcmaster branch has 3  commits
```

4) Now Run Merge Command

```
git merge my-new-feature-1

//jlcmaster branch has 6  commits which is same as my-new-feature-1
```

5) Push local Master to Remote Master

```
git push
```

22. Working On Multiple Branches

Developer -1 Tasks:

Task 17: Working On Multiple Branches

In this Task, You will do

1) Create the Branch - **my-new-feature-2**
2) Doing Commits to **my-new-feature-2**
3) Pushing to Remote **my-new-feature-2**
4) Create the Branch - **my-new-feature-3**
5) Doing Commits to **my-new-feature-3**
6) Pushing to Remote **my-new-feature-3**
7) Merge the both the feature branches with master

1) Create and Switch to my-new-feature-2 branch

```
git checkout -b my-new-feature-2  //Create and switch
```

2) Write your Code and Commit in my-new-feature-2

```
Write New-Feature-2A.java
git add New-Feature-2A.java
git commit -m "my commit -1 for my-new-feature-2"
git status

git push --set-upstream origin my-new-feature-2
```

3) Create and Switch to my-new-feature-3 branch

```
git checkout -b my-new-feature-3  //Create and switch
```

4) Write your Code and Commit in my-new-feature-3

```
Write New-Feature-3A.java
git add New-Feature-3A.java
git commit -m "my commit -1 for my-new-feature-3"
git status

git push --set-upstream origin my-new-feature-3
```

5) Check the Branch Differences

```
git diff jlcmaster my-new-feature-1
git diff jlcmaster my-new-feature-2
git diff jlcmaster my-new-feature-3
```

6) Now Bracnhes are having the Commits are as follows

```
jlcmaster => 6 commits
my-new-feature-1 => 6 commits
my-new-feature-2 => 7 commits
my-new-feature-3 => 7 commits
```

7) Merge **my-new-feature-2** and **my-new-feature-3** Branch with jlcmaster

```
git merge my-new-feature-2  my-new-feature-3
```

8) Push to Remote Origin

```
git push
```

9) Now Bracnhes are having the Commits are as follows

```
jlcmaster => 9 commits
my-new-feature-1 => 6 commits
my-new-feature-2 => 7 commits
my-new-feature-3 => 7 commits
```

10)Switch to the my-new-feature-2 Branch to Work for New Feature2

```
git checkout my-new-feature-2  //switch
```

11)Update Hello.java and Commit

```
Update Hello.java
git add Hello.java
git commit -m "my commit -2 for my-new-feature-2"
git status
git push
```

12) Switch to the my-new-feature-3 Branch to Work for New Feature3

```
git checkout my-new-feature-3//switch
```

13)Update Hello.java and Commit

```
Update Hello.java
git add Hello.java
git commit -m "my commit -2 for my-new-feature-3"
git status
git push
```

14)Now Bracnhes are having the Commits are as follows

```
jlcmaster => 9 commits
my-new-feature-1 => 6 commits
my-new-feature-2 => 8 commits
my-new-feature-3 => 8 commits
```

15)Merge **my-new-feature-2** and **my-new-feature-3** Branch with jlcmaster

```
git merge my-new-feature-2  my-new-feature-3
```

16)This may cause the Conflicts. If any conflicts at Merge time , then merge will ne failed.

Then you have two Options

1)Cancel the Merge

git merge --abort

2) Resolve the Conflicts and Commit again

=> Resolve Manually

=> git add Hello.java

=> git commit -m "After merging F2 and F3 and Resolving Conflicts"

17)Push to Remote Origin

```
git push
```

18)Now Bracnhes are having the Commits are as follows

```
jlcmaster => 12 commits
my-new-feature-1 => 6 commits
my-new-feature-2 => 7 commits
my-new-feature-3 => 7 commits
```

23. Tag You Branch

Task 18: Tag You Branch

In this Task, You will do

1) See the Avaulable Tags
2) Add New Tag
3) Push Tag to Remote

1) Tagging realated Commands

```
git tag
git tag 1.0.0
git tag
git push --tag
```

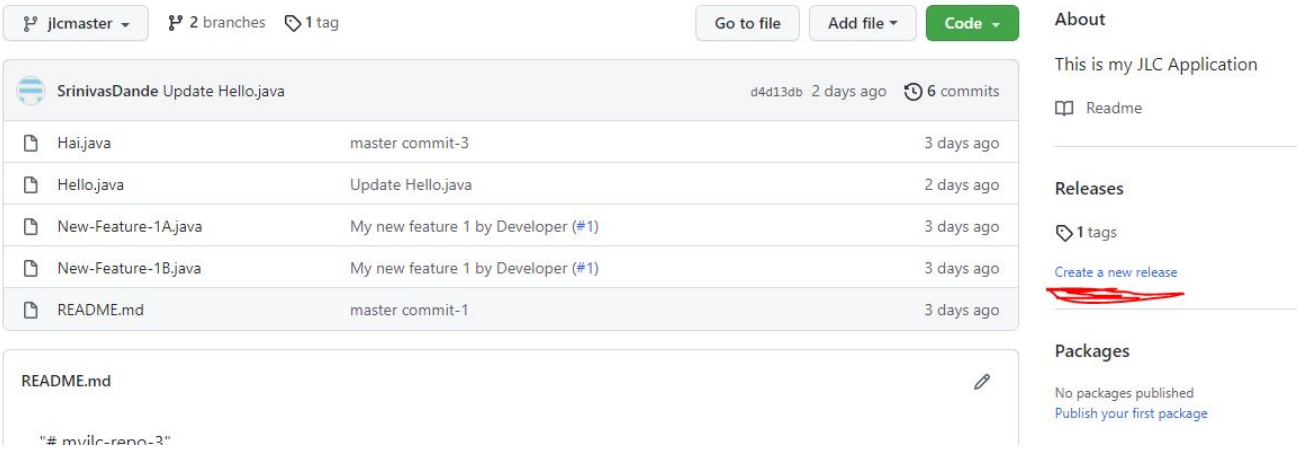
24. Creating Release Version

Task 19: Creating Release Version

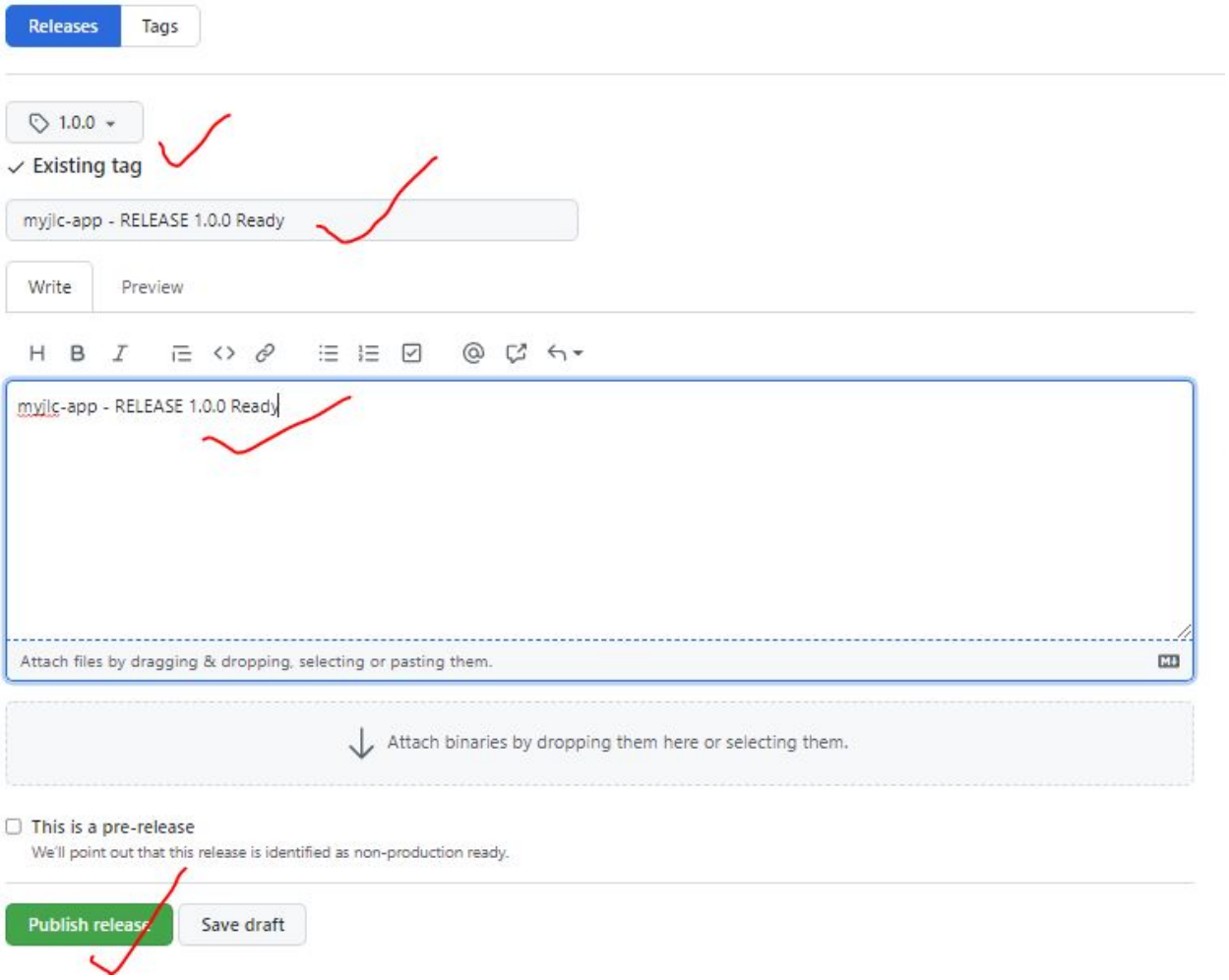
In this Task, You will do

1) Create the Release Version

1) Open the Repository in GitHub, You can see following



2) Click on **Create a New Release,** Fill as shown below and Click on **Publish Release**



25. Delete the Branches

Task 20: Delete the Branches

In this Task, You will do

2) Create the Release Version
3) Publish the Release Version

1) Branch Deletion realated Commands

git branch

git branch -r

git branch -a

git branch -d my-new-feature-1 //Deletes local branch

git push origin --delete my-new-feature-1 //Deletes Remote branch

git branch –a

26. Save your Work with git stash

Task 21: Save your Work with git stash

In this Task, You will do

4) Add new Files or Update Existing Files
5) Save them with git stash
6) Check the stash list
7) Use the Saved Work later in the same branch or other branch
8) Remove Saved work from Stash

2) Switch to my-new-feature-2 branch

```
git checkout my-new-feature-2
```

3) Check any work is saved with Stash

```
git stash list
```

4) Write new Files, add to Index and Save it

```
echo "I am test1.java" >> test1.java
git add test1.java
git stash

echo "I am test2.java" >> test2.java
git add test2.java
git stash

echo "I am test3.java" >> test3.java
git add test3.java
git stash
```

```
echo "I am test4.java" >> test4.java
git add test4.java
git stash
```

5) Check any work is saved with Stash

```
git stash list
```

6) Write new Files, add to Index , Commit it and then push

```
echo "I am test5.java" >> test5.java
git add test5.java
git commit -m "commit -1 -on F2-- test5.java - added"

git push
```

7) Check any work is saved with Stash

```
git stash list
git stash show
git stash show stash@{0}
git stash show stash@{1}
git stash show stash@{2}
git stash show stash@{3}
```

8) Switch to my-new-feature-3 branch

```
git checkout my-new-feature-3
```

9) Check any work is saved with Stash

```
git stash list
```

10)Apply the Saved Work and Commit

```
git stash apply stash@{3}
git commit -m "commit -1 -on F3-- test1.java - added"
```

11)Switch to my-new-feature-2 branch

```
git checkout my-new-feature-2
```

12)Check any work is saved with Stash

```
git stash list
```

13)Apply the Saved Work and Commit

```
git stash pop
git commit -m "commit -2 -on F2-- test4.java - added"
```

14)Check any work is saved with Stash

```
git stash list


git stash show stash@{0}
git stash show stash@{1}
```

15)Apply the Saved Work and Commit

```
git stash apply stash@{2}
git commit -m "commit -3 -on F2-- test1.java - added"
```

16)Apply the Saved Work and Commit

```
git stash pop
git commit -m "commit -4 -on F2-- test3.java - added"
```

17)Check any work is saved with Stash

```
git stash list
git stash show stash@{0}
git stash show stash@{1}
```

18)Switch to my-new-feature-2 branch

```
git checkout my-new-feature-3
```

19)Check any work is saved with Stash

```
git status
git stash list
```

20)Apply the Saved Work and Commit

```
git stash apply
git stash pop
git commit -m "commit -2 -on F3-- test2.java - added"
```

21)Check any work is saved with Stash

```
git stash list
git stash show stash@{0}
git stash show stash@{1}
```

22)Merge my-new-feature-2 and my-new-feature-3 Branch with jlcmaster

```
git merge my-new-feature-2  my-new-feature-3
```

23)Push to Remote Origin

```
git push
```

24)Tagging realated Commands

```
git tag
git tag 1.7.9-RELEASE
git tag
git push --tag
```

27. Create the Branch to Work for Experimental Features

1) Switch to new branch called **my-experimental-features**

```
git checkout –b my-experimental-features
```

2) Check the Current Branch

```
git branch -a
```

3) Write new File, add to Index and Commit it

```
echo "I am MyLogic1.java" >> MyLogic1.java
git add MyLogic1.java
git commit -m "commit-1 for my-experimental-features - MyLogic1"
git status
```

4) Write new File, add to Index and Commit it

```
echo "I am MyLogic2.java" >> MyLogic2.java
git add MyLogic2.java
git commit -m "commit-2 for my-experimental-features - MyLogic2"
git status
```

5) Write new File, add to Index and Commit it

```
echo "I am SuperLogic1.java" >> SuperLogic1.java
git add SuperLogic1.java
git commit -m "commit-3 for my-experimental-features - SuperLogic1"
git status
```

6) Write new File, add to Index and Commit it

```
echo "I am SuperLogic2.java" >> SuperLogic2.java
git add SuperLogic2.java
git commit -m "commit-4 for my-experimental-features - SuperLogic2"
git status
```

7) Write new File, add to Index and Commit it

```
echo "I am SuperDuperLogic.java" >> SuperDuperLogic.java
git add SuperDuperLogic.java
git commit -m "commit-5 for my-experimental-features - SuperDuperLogic"
git status
```

28. Creating and Applying Paches

Task 22: Creating and Applying Paches

In this Task, You will do

1) Create the Patch for the Differences
2) Create the Patch for selective Commit Id
3) Apply the Patch

1) Switch to existing branch called **my-experimental-features**

```
git checkout  my-experimental-features
```

2) See the Commits of **my-experimental-features**

```
git log –oneline
```

Display below commits

ccc8645 commit-5 for my-experimental-features - SuperDuperLogic

1b17ccb commit-4 for my-experimental-features - SuperLogic2

1a6deec commit-3 for my-experimental-features - SuperLogic1

f1b0c85 commit-2 for my-experimental-features - MyLogic2

88c9b3e commit-1 for my-experimental-features - MyLogic1

3) See the Bracch difference

```
git diff  jlcmaster my-experimental-features
```

4) Create the Paches

```
git format-patch jlcmaster -o mypatches
```

5) See the Patches created

```
cd mypatches
ls
0001-commit-1-for-my-experimental-features-MyLogic1.patch
0002-commit-2-for-my-experimental-features-MyLogic2.patch
0003-commit-3-for-my-experimental-features-SuperLogic1.patch
0004-commit-4-for-my-experimental-features-SuperLogic2.patch
0005-commit-5-for-my-experimental-features-SuperDuperLogi.patch
```

6) Goto the Branch where you want apply the Patch

```
git checkout jlcmaster
```

7) Apply the Patch Required

```
git am mypatches\0001-commit-1-for-my-experimental-features-MyLogic1.patch
```

8) Apply the Another Patch

```
git am mypatches\0003-commit-3-for-my-experimental-features-SuperLogic1.patch
```

9) Check whether Two Pacthes Aplyed to Current Branch or not

```
git log --oneline
```

10)Switch to existing branch called **my-experimental-features**

```
git checkout  my-experimental-features
```

11)Create the patch for Required Commit Id

```
git format-patch -1 f1b0c85 -o patches
```

12)See the Patch created

```
cd patches

ls

0001-commit-2-for-my-experimental-features-MyLogic2.patch
```

13)Goto the Branch where you want apply the Patch

```
git checkout jlcmaster
```

14)Apply the Patch Required

```
git am patches/0001-commit-2-for-my-experimental-features-MyLogic2.patch
```

15)Check whether Two Pacthes Aplyed to Current Branch or not

```
git log --oneline
```

16)You can Apply the Required Commit in any branch with **cherry-pick**

```
git cherry-pick ccc8645
```

17)Check whether checy-pick applied to to Current Branch or not

```
git log --oneline
```

29. Merge Commit Vs Rebase Vs Squash

You can merge the Feature bracn with master in 3 different Ways

1) Merge Commit
2) Rebase and Merge
3) Squash and Merge

29.A. Merge Commit

Architect Tasks: Setup Remote repositoty

8) Create Empty Private Repository in GitHub with name **myjlc-repo-2** without any files

9) Do the following in your local Machine

```
cd architect

mkdir myjlc-repo-2

cd myjlc-repo-2

echo "# myjlc-repo-2" >> README.md

git init

git status

git add README.md

git commit -m "master commit-1"

git status
```

10)Change the Branch name to **jlcmaster**

```
git branch -M jlcmaster
```

11)Add the Remote Origin

```
git remote add origin git@github.com:DandesClasses/myjlc-repo-2.git
```

12)Push to remote first time

```
git push -u origin jlcmaster
```

13)Write new File , commit and push

```
echo "I am Hello.java" >> Hello.java
git status
git add Hello.java
git commit -m "master commit-2"
git status
git push
```

Developer -1 Tasks: Work on Feature Branch

14)Clone the Remote Repository

```
mkdir developer-1
cd developer-1
git clone git@github.com:DandesClasses/myjlc-repo-2.git
```

15)Create the Branch to Work for New Feature 1

```
git checkout -b my-new-feature-1
```

16)Write your Code and Commit

```
echo "I am New-Feature-1A.java" >> New-Feature-1A.java
git add New-Feature-1A.java
git commit -m "feature commit-1"
git status
```

17)Push to Remote Origin

```
git push --set-upstream origin my-new-feature-1
```

18)Write your Code Again and Commit

```
echo "I am New-Feature-1B.java" >> New-Feature-1B.java
git add New-Feature-1B.java
git commit -m "feature commit-2"
git status
```

19)Push to Remote Origin

```
git push
```

20)Now See the Feature Branch commits

```
git log --oneline

3c48842 feature commit-2
c14bb3c feature commit-1
6110d90 master commit-2
c227334 master commit-1
```

21)Create the **Pull Request** .

Create the **Pull Request** - for merging 2 commits of my-new-feature-1with jlcmaster

Do this GitHub.com

Architect Tasks: Setup Remote repositoty

22)Write new File , commit and push

```
echo "I am Hai.java" >> Hai.java
git add Hai.java
git commit -m "master commit-3"
git status


git push
```

- Architect or Owner will look into PR.
- Architect will look into the Commits of Developer- 1and will try to merge into jlcmaster

jlcmaster has 3 commits	feature branch has 4 commits
e213359 master commit-3	3c48842 feature commit-2
6110d90 master commit-2	c14bb3c feature commit-1
c227334 master commit-1	6110d90 master commit-2
	c227334 master commit-1

- After Merge commit, Pull Request will be Closed.
- After Merge commit, Following **6 commits** will be there in **jlcmaster**

4ae3508Merge pull request #1 from DandesClasses/my-new-feature-1

e213359 master commit-3

3c48842 feature commit-2

c14bb3c feature commit-1

6110d90 master commit-2

c227334 master commit-1

29.B. Rebase and Merge

Architect Tasks: Setup Remote repositoty

1) Create Empty Private Repository in GitHub with name **myjlc-repo-3** without any files
2) Do the following in your local Machine

```
cd architect
mkdir myjlc-repo-3
cd myjlc-repo-3
echo "# myjlc-repo-3" >> README.md
git init
git status
git add README.md
git commit -m "master commit-1"
git status
```

3) Change the Branch name to **jlcmaster**

```
git branch -M jlcmaster
```

4) Add the Remote Origin

```
git remote add origin git@github.com:DandesClasses/myjlc-repo-3.git
```

5) Push to remote first time

```
git push -u origin jlcmaster
```

6) Write new File , commit and push

```
echo "I am Hello.java" >> Hello.java
git status
git add Hello.java
```

```
git commit -m "master commit-2"
git status
git push
```

Developer -1 Tasks: Work on Feature Branch

7) Clone the Remote Repository

```
mkdir developer-1
cd developer-1
git clone git@github.com:DandesClasses/myjlc-repo-3.git
```

8) Create the Branch to Work for New Feature 1

```
git checkout -b my-new-feature-1
```

9) Write your Code and Commit

```
echo "I am New-Feature-1A.java" >> New-Feature-1A.java
git add New-Feature-1A.java
git commit -m "feature commit-1"
git status
```

10)Push to Remote Origin

```
git push --set-upstream origin my-new-feature-1
```

11)Write your Code Again and Commit

```
echo "I am New-Feature-1B.java" >> New-Feature-1B.java
git add New-Feature-1B.java
git commit -m "feature commit-2"
git status
```

12)Push to Remote Origin

```
git push
```

13)Now See the Feature Branch commits

```
git log --oneline


3c48842 feature commit-2

c14bb3c feature commit-1

6110d90 master commit-2

c227334 master commit-1
```

14)Create the **Pull Request** .

Create the **Pull Request** - for merging 2 commits of my-new-feature-1with jlcmaster

Do this GitHub.com

Architect Tasks: Setup Remote repositoty

15)Write new File , commit and push

```
echo "I am Hai.java" >> Hai.java

git add Hai.java

git commit -m "master commit-3"

git status


git push
```

- Architect or Owner will look into PR.
- Architect will look into the Commits of Developer- 1and will try to merge into jlcmaster using using Rebase option

jlcmaster has 3 commits	**feature branch has 4 commits**
e213359 master commit-3	3c48842 feature commit-2
6110d90 master commit-2	c14bb3c feature commit-1
c227334 master commit-1	6110d90 master commit-2
	c227334 master commit-1

- **After Rebase and Merge**, Pull Request will be Closed.
- After **Rebase and Merge**, Following **5commits** will be there in **jlcmaster**

f7f4854 feature commit-2

8a65d0c feature commit-1

34059bd master commit-3

d7340ef master commit-2

86ad808 master commit-1

29.C. Squash and Merge

Architect Tasks: Setup Remote repositoty

1) Create Empty Private Repository in GitHub with name **myjlc-repo-4** without any files
2) Do the following in your local Machine

```
cd architect
mkdir myjlc-repo-4
cd myjlc-repo-4
echo "# myjlc-repo-4" >> README.md
git init
git status
git add README.md
git commit -m "master commit-1"
```

```
git status
```

3) Change the Branch name to **jlcmaster**

```
git branch -M jlcmaster
```

4) Add the Remote Origin

```
git remote add origin git@github.com:DandesClasses/myjlc-repo-4.git
```

5) Push to remote first time

```
git push -u origin jlcmaster
```

6) Write new File , commit and push

```
echo "I am Hello.java" >> Hello.java
git status
git add Hello.java
git commit -m "master commit-2"
git status
git push
```

Developer -1 Tasks: Work on Feature Branch

7) Clone the Remote Repository

```
mkdir developer-1
cd developer-1
git clone git@github.com:DandesClasses/myjlc-repo-4.git
```

8) Create the Branch to Work for New Feature 1

```
git checkout -b my-new-feature-1
```

9) Write your Code and Commit

```
echo "I am New-Feature-1A.java" >> New-Feature-1A.java
git add New-Feature-1A.java
git commit -m "feature commit-1"
git status
```

10)Push to Remote Origin

```
git push --set-upstream origin my-new-feature-1
```

11)Write your Code Again and Commit

```
echo "I am New-Feature-1B.java" >> New-Feature-1B.java
git add New-Feature-1B.java
git commit -m "feature commit-2"
git status
```

12)Push to Remote Origin

```
git push
```

13)Now See the Feature Branch commits

```
git log --oneline
3c48842 feature commit-2
c14bb3c feature commit-1
6110d90 master commit-2
c227334 master commit-1
```

14)Create the **Pull Request** .

Create the **Pull Request** - for merging 2 commits of my-new-feature-1with jlcmaster

Do this GitHub.com

Architect Tasks: Setup Remote repositoty

15)Write new File , commit and push

```
echo "I am Hai.java" >> Hai.java
git add Hai.java
git commit -m "master commit-3"
git status
git push
```

- Architect or Owner will look into PR.
- Architect will look into the Commits of Developer- 1and will try to merge into jlcmaster using using **Squash option**

jlcmaster has 3 commits	**feature branch has 4 commits**
e213359 master commit-3	3c48842 feature commit-2
6110d90 master commit-2	c14bb3c feature commit-1
c227334 master commit-1	6110d90 master commit-2
	c227334 master commit-1

- **After Squash and Merge**, Pull Request will be Closed.
- After **Squash and Merge**, Following **4 commits** will be there in **jlcmaster**

ada44e8My new feature 1 with 2 commits here

ecda360 master commit-3

6e1153d master commit-2

6bd8a92 master commit-1

30.Fork the Repository

- A fork is a copy of a repository.
- Forking a repository allows you to freely experiment with changes without affecting the original project.
- Forking in GitHub is the process of creating a copy of a complete repository from One user GitHub Account to another account

Forking Workflow

1) Create a Fork

- Simply click on the "fork" button of the repository page on GitHub.

2) Clone your Fork

- clone command creates a local git repository from your remote fork on GitHub.

```
git clone git@github.com:DandesClasses/myjlc-repo-4.git
```

3) Modify the Code

- Modify the code and commit them to your local clone using the git commit command.

4) Push your Changes

- Use the git push command to upload your changes to your remote fork on GitHub.

5) Create a Pull Request

- On the GitHub page of your remote fork, click the "pull request" button.
- Wait for the owner to merge or comment your changes and be proud when it is merged