Git Workflow

1. **Centralized Workflow**
    - One repository and multiple developers
    - give everyone on your team push access
    - the repository only holds stable code
    - each developer has a master branch which holds the stable code and several develop branches

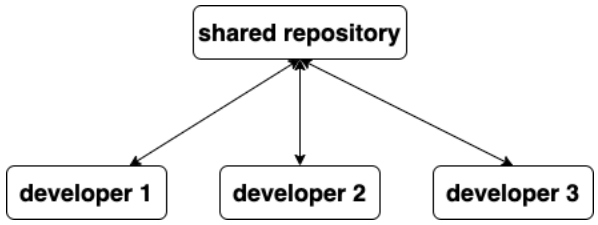

Figure 1. The paradigms of the centralized workflow

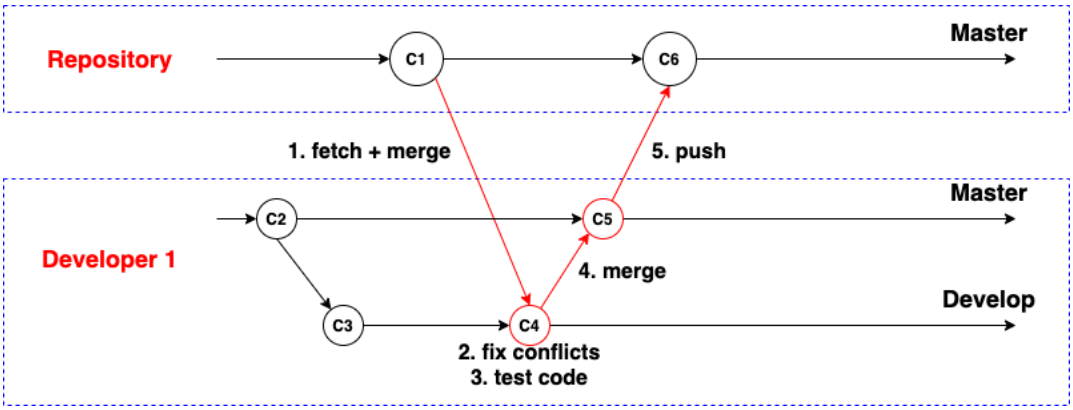

Figure 2. The commands for pushing a commit

Commands

1. fetch + merge
    a. git fetch origin, fetch the commit to the develop branch in the local
    b. git merge origin/master, merge the remote master branch to the local develop branch
2. fix conflicts
    a. if there are any conflicts, fix the conflict and then commit
3. test the code to make sure everything works well
4. merge
    a. git checkout master, switch to the local master branch
    b. git merge develop, merge the local develop branch to the local master branch
    c. if there are any conflicts, fix them and then commit
5. push
    a. git push origin HEAD, push the local master to the remote master

2. **Integration-Manager Workflow**
    - One repository branch saving the stable code, can be accessed by the manager
    - One repository branch for each developer, and each developer has the push access to their own repository branch
    - Each developer has fetch access from the branch for the stable code
    - The manager has the fetch access from the developers' branches
    - Each developer has a master branch which holds the stable code and several develop branches

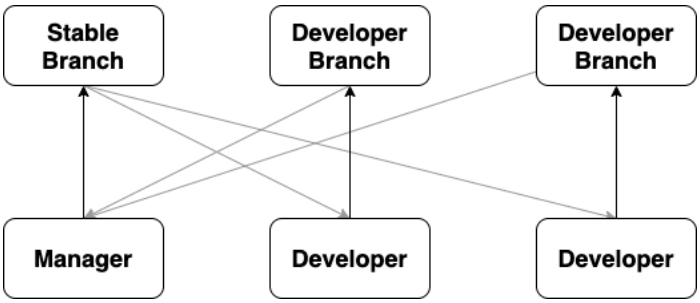

Figure 3. The paradigms of the Integration-Manager Workflow

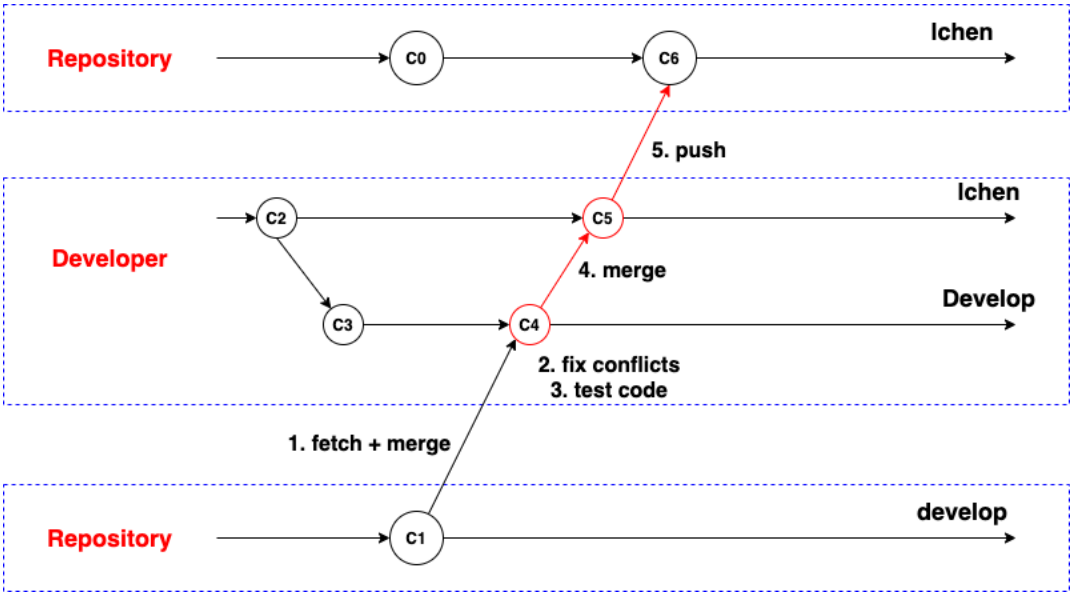

Figure 4. The commands of each developer

Developer Commands

0. rename branch if not done (suppose the developer branch is called lchen)
    a. git branch -m master lchen, change the local master branch to the corresponding branch name on the remote repository, avoid pushing to other developers' branch
1. fetch + merge (suggest not using pull)
    c. git fetch origin, fetch the commit to the **develop** branch in the local
    d. git merge origin/develop, merge the stable code from the remote **develop** branch to the local develop branch

2. fix conflicts
    b. if there are any conflicts, fix the conflict and then commit
3. test the code to make sure everything works well
4. merge
    d. git checkout lchen, switch to the local branch holding the stable code
    e. git merge develop, merge the local develop branch to the local lchen branch
    f. if there are any conflicts, fix them and then commit
6. push
    a. git push origin HEAD, push the local lchen branch to the remote lchen branch

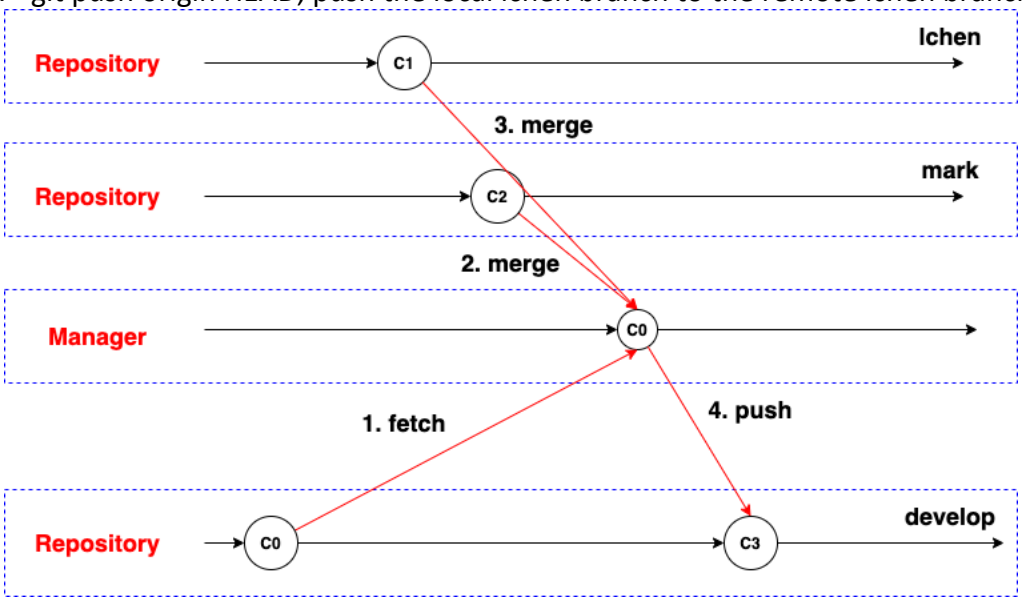

Figure 5. The commands of the manager

Manager Commands

0. merge request
    a. developer sends a merge request to the manager
1. fetch
    a. git clone url, fetch the code to the local
    b. git checkout -b develop origin/develop, create a local branch
2, 3. merge
    a. git merge origin/lchen, fix conflicts if exist
    b. git merge origin/mark, fix conflicts if exist
    c. test the code to make sure everything works well
4. push
    a. git push origin HEAD, push to the remote develop branch

3. **Dictator and Lieutenants Workflow**
    - Each developer creates a stable master branch
    - Each Lieutenant merges the master branches of the developers in his/her team
    - The dictator merges the master branches of the lieutenants, then push to the blessed repository

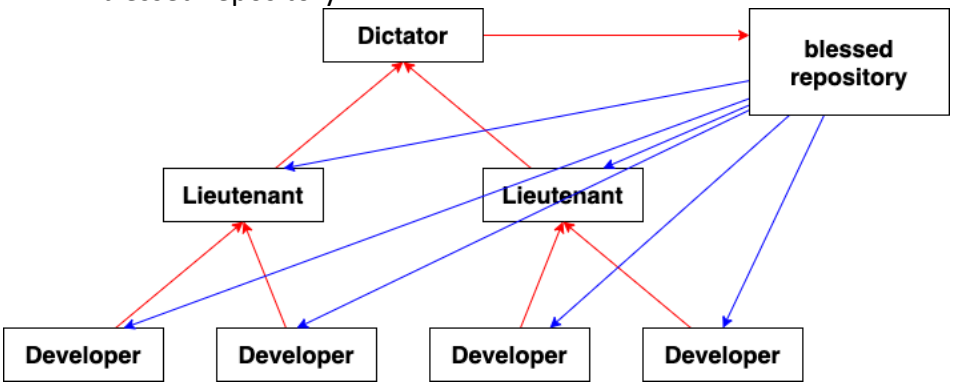

Figure 6. The paradigms of the Dictator and Lieutenants Workflow

4. Reference

https://git-scm.com/book/en/v2/Distributed-Git-Distributed-Workflows, Pro Git, 2nd Edition.