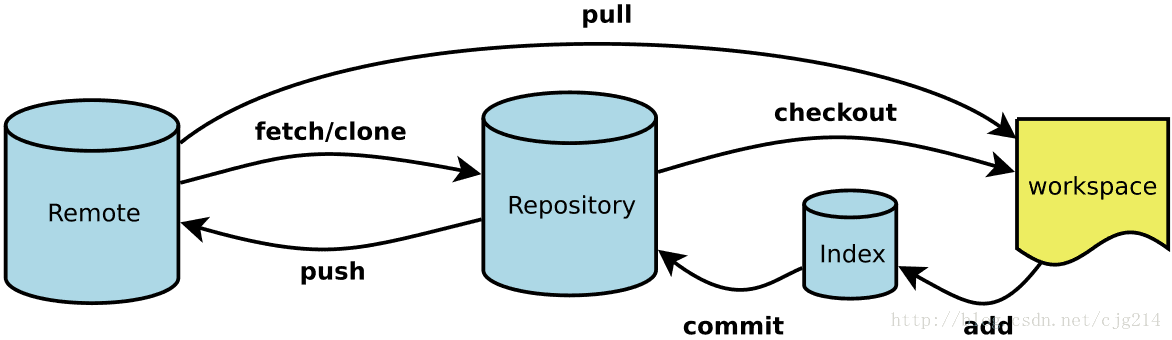
**Git Process for Multi-person Collaborative Development**



Git Command

When multiple people work together, the main (master) branch and dev branch are usually on GitHub at the beginning of the project, and the ‘main (master)’ branch is the main branch, which is directly associated with the final released program, and each developer cannot directly push the program To the main branch, it can only be pushed to the dev branch. After testing without any problems, it will be finally pushed to the main branch by a special person. Each developer needs to create their own branch based on the dev branch.

# For example:

The project team members are Zongyang and Simon, and Zongyang is the project leader.

## Step1: creates a repo

The project leader creates a repo called gitTest on GitHub and add Simon.

Graphical user interface, text, application, email

Description automatically generated

## Step2: create two branches.

Init project，create two branches named ‘main’ and ’dev’

$ git init

Create a text file a.txt, and add text ‘zongyang init’;

$ git add .

$ git commit -m "init"

$ git remote add origin https://github.com/zongyanggong/gitTest.git

$ git push origin master

$ git checkout -b dev

$ git push --set-upstream origin dev

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Then, there are two branches on GitHub, master, and dev.

Graphical user interface, text, application, email, Teams

Description automatically generated

A picture containing Teams

Description automatically generated

## **Step3：**clones the project.

### 3.1 Project member Simon clones the project.

$ git clone https://github.com/zongyanggong/gitTest.git

Check local repository:

$ git branch

### 3.2 At this moment, there is only one branch: master. Simon needs to fetch ‘dev’ branch。

$ git fetch origin dev:dev

Meaning: fetch remote ‘dev’ branch to local repository.

### 3.3 Create local developing branch：

$ git checkout -b simon\_test

Edit a.txt file and add word “Simon created”.

$ git add .

$ git commit -m “Simon created”

$ git push origin simon\_test

Text

Description automatically generated

Now, simon\_test branch was added on GitHub and content in a.txt file was changed.

A picture containing table

Description automatically generated

But a.txt in dev branch did not change.

A picture containing application

Description automatically generated

### 3.4 Update dev branch:

$ git checkout dev

$ git merge simon\_test

$ git push origin dev

Text

Description automatically generated

Then the a.txt in dev branch on GitHub was changed。

Background pattern

Description automatically generated

## Step4：daily work flow (Take Simon for example)

### 4.1 Fetch latest content in dev before work

For example, a.txt file has been added one line by Zongyang “add by zongyang”. Now a.txt file in dev branch has already been updated, but a.txt file in simon\_test branch is not the latest version.

Table

Description automatically generated with medium confidence

Table

Description automatically generated with low confidence

$ git checkout dev

$ git pull origin dev

Now if dev branch has new version, Simon should checkout to his local branch and update it.

$ git checkout simon\_test

$ git merge dev

Then, his local branch’s a.txt file has updated to the same version as dev。

$ git push origin simon\_test

The a.txt file in simon\_test branch on GitHub has been updated.

Text

Description automatically generated

A picture containing table

Description automatically generated

### 4.2 Push code to your own branch simon\_test

You can directly push the content of your own branch simon\_test at any time during the work process.

### 4.3 Push your work back to the dev branch.

When get off work or a function has been solved, you need to push your work to the dev branch.

For example, add a line "simon add a=1" to the a.txt fileA picture containing rectangle

Description automatically generated

$ git add .

$ git commit -m “simon add a=1”

$ git push origin simon\_test

Text

Description automatically generated

At this time, the a.txt file under the simon\_test branch on GitHub is updated, but the a.txt file under the dev branch has not been updated yet.Graphical user interface, application, Teams

Description automatically generated

Application

Description automatically generated with low confidence

If Zongyang has updated the content of the a.txt file currently, add a line "add by zongyang a=2".Graphical user interface, application

Description automatically generated

Currently, the content under the dev branch on GitHub is inconsistent with the content of Simon's local dev branch. Simon locally switches to the dev branch first,

$ git checkout dev

The content of a.txt under the dev branch is:

Graphical user interface, text, application

Description automatically generated

Therefore, every time Simon needs to update the content of the dev branch, he first needs to pull the content of the dev branch on GitHub, to ensure that the local dev branch file is up to date before merging.

$ git pull origin dev

Then open the local a.txt file, which is consistent with the a.txt file on GitHub.Shape, rectangle

Description automatically generated

Then merge the files under the simon\_test branch into the dev branch:

$ git merge simon\_test

At this time, a conflict will be prompted, and the conflict needs to be resolved manually. Open the local a.txt file:

Text, shape, rectangle

Description automatically generated

After manually resolving conflicts, the file is as follows:A picture containing graphical user interface

Description automatically generated

At this point, you can add and commit the dev branch and push it to GitHub.

$ git add .

$ git commit -m “simon add a=1”

$ git push origin dev

Text

Description automatically generated

Application

Description automatically generated with low confidence

But at this time, the content of the simon\_test branch on Simon’s local and GitHub is the content before the merger:Application

Description automatically generated with medium confidence

Switch to the simon\_test branch:

$ git checkout simon\_test

Currently, the content of the local a.txt file is as follows:Graphical user interface, text

Description automatically generated with medium confidence

$ git merge dev

Now open the local a.txt file, the content has changed.Graphical user interface, application

Description automatically generated

$ git push origin simon\_test

Currently, the content under the simon\_test branch on GitHub is updated.Graphical user interface, application

Description automatically generated

Benefits of such a process:

1. There is almost no conflict.

2. Never pollute the dev branch. Because every time the local merger is completed and the conflict is cleared, the push will go to the remote end. Then when someone updates the local dev branch and then merges, even if there is a conflict, it will only be a conflict generated by the latest code.

Finally, let's clarify our thinking again.

1. During official development, each person only needs to have two local branches. One is called dev, and the other is your own branch.

2. Everyone can directly push code to their own branch. But when pushing the dev branch. The latest remote dev branch must be pulled first. Then merge with the local branch, and push after clearing the conflict. Currently, add and commit are required.

3. After submitting the dev branch, update the respective branches on GitHub. At this time, only merge and push are needed, and add and commit are not required.