
Git Collaboration

Centralized workflow using
push/pull

BCIT COMP 1800



AGENDA

1. Basic collaboration

- GitHub
- Collaboration workflow
- Conflict Resolution

2. Advanced collaboration

- Branch, checkout, merge
- Fork, pull-request
- Workflows

Review: GIT basics

Command-line program

GUI interfaces available (optional)

Basic functions:

- Check out (git clone, git pull,)
- Check in (git add, git commit, git push ...)
- Check status (git log, git status, git diff, git blame ...)
- Check remote (git remote show, git remote show origin, git remote add origin url,...)

Today: collaboration with git

GitHub is a cloud platform for git repositories

- Central point for sharing code (social collaborative coding)
- Distributed environment
- Microsoft (2018); there are other options

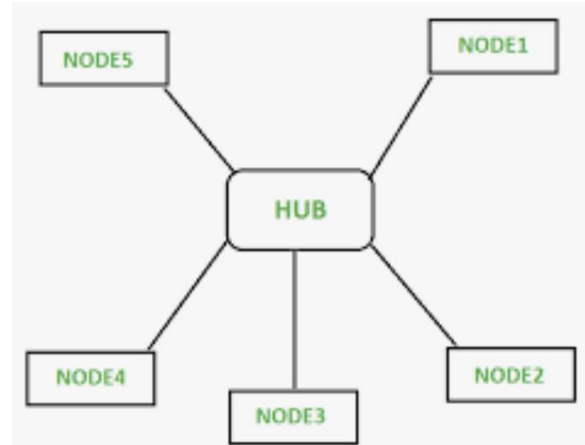
Big Questions:

- How do we make it easy to contribute? To maintain?
- How do we co-ordinate with a team of developers?

Git Workflows

Terminology:

- “Node” (contributor’s location)
- “Hub” (central location)



Many possible ways of using Git <https://www.atlassian.com/git/tutorials/comparing-workflows>

- 1.The “Fork / Pull-request” Workflow (open source)
- 2.The “Centralized” Workflow (this week!)
- 3.The “Feature Branch” Workflow
- 4.The “Gitflow” Workflow

Workflows

A workflow is an agreement:

- made by the team
- how everyone will use git and GitHub
- communication protocol.



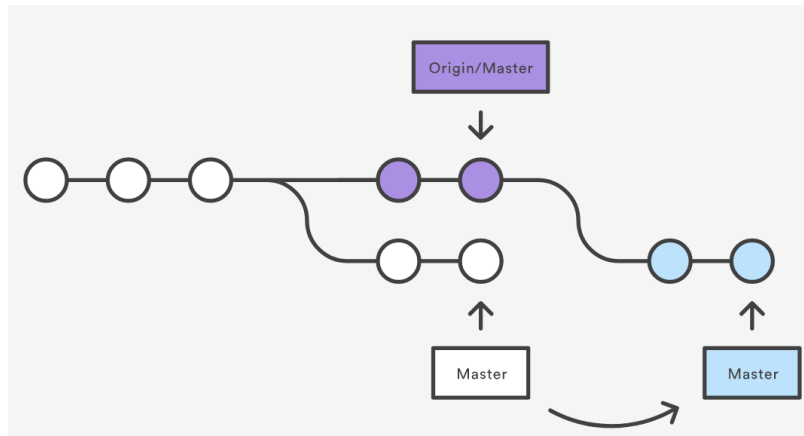
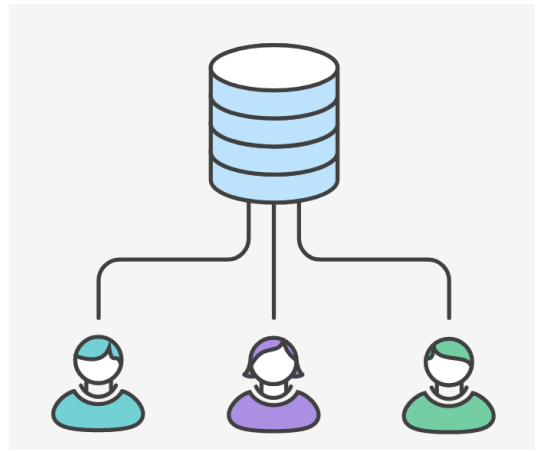
Git Centralized Workflow

<https://www.atlassian.com/git/tutorials/comparing-workflows#centralized-workflow>

- Everyone can push, pull
- If you can't push, try this:

`git pull --rebase origin master` OR

`git pull` (to get the latest)



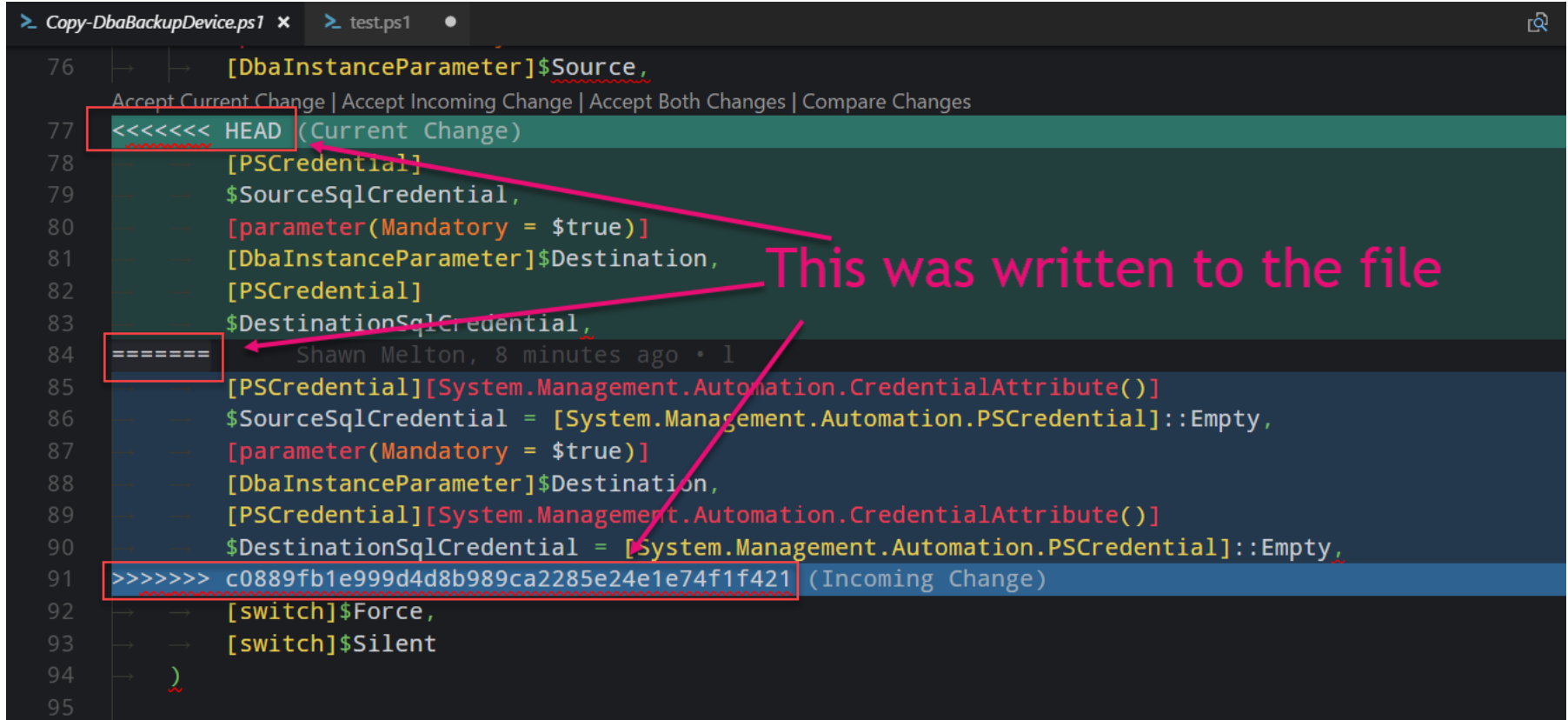
Pull may cause a “Merge Conflict”

<https://help.github.com/en/articles/resolving-a-merge-conflict-using-the-command-line>

What if local version of file is different from remote version?

- Git says: *“Oh the files look different! Let me merge the files for you. I will keep lines from both versions. I won’t delete anything. Promise.”*
- Developer says: *“Gee thanks! I will open the merged file on my computer using an editor, and decide what to keep.”*

Open merged file (using text editor)



The screenshot shows a PowerShell script editor with two tabs: 'Copy-DbBackupDevice.ps1' and 'test.ps1'. The script content is as follows:

```
76 [DbInstanceParameter]$Source,  
    Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes  
77 <<<<<<< HEAD (Current Change)  
78 [PSCredential]  
79 $SourceSqlCredential,  
80 [parameter(Mandatory = $true)]  
81 [DbInstanceParameter]$Destination,  
82 [PSCredential]  
83 $DestinationSqlCredential,  
84 ===== Shawn Melton, 8 minutes ago • 1  
85 [PSCredential][System.Management.Automation.CredentialAttribute()]  
86 $SourceSqlCredential = [System.Management.Automation.PSCredential]::Empty,  
87 [parameter(Mandatory = $true)]  
88 [DbInstanceParameter]$Destination,  
89 [PSCredential][System.Management.Automation.CredentialAttribute()]  
90 $DestinationSqlCredential = [System.Management.Automation.PSCredential]::Empty,  
91 >>>>>>> c0889fb1e999d4d8b989ca2285e24e1e74f1f421 (Incoming Change)  
92 [switch]$Force,  
93 [switch]$Silent  
94 )  
95
```

Annotations in the image include:

- A red box around line 77: '<<<<<<< HEAD (Current Change)'.
- A red box around line 84: '===== Shawn Melton, 8 minutes ago • 1'.
- A red box around line 91: '>>>>>>> c0889fb1e999d4d8b989ca2285e24e1e74f1f421 (Incoming Change)'.
- A pink text overlay 'This was written to the file' with arrows pointing to the three red boxes.



AGENDA

1. Basic collaboration

- GitHub
- Push and Pull
- Conflict Resolution

2. Advanced collaboration

- Branch, checkout, merge
- Fork, pull-request
- Workflows

Git commands: branch, checkout

`git push <remote> <branch>` (push a branch)

`git branch abc def` (create some new branches)

`git checkout abc` (checkout a branch)

`git checkout def`

`git branch` (list all branches)

`git branch -d xyz` (deletes xyz)

NOTE: a branch is not available to others unless you push it to remote.

NOTE: a shortcut for checkout and create a new branch in one step:

`git checkout -b my_new_branch` (creates and checks out the new branch)

Git commands: merge

To join two or more development histories together.
It is tracked, and treated, like a commit in history.

`git checkout -b abc` (this will create, and checkout a branch)

OR

`git checkout abc` (checkout abc as your active branch)

`git diff <source_branch> <target_branch>`

`merge def` (merge def into abc)

`merge def ghi` (merge both def and ghi into abc)

GitHub feature: the pull-request

You are ... *requesting* another developer (e.g., repo gatekeeper) to *pull* a branch from your branch/repo into their branch/repo.

- Need 4 pieces of info:
Source repo, Source branch, Dest repo, Dest branch
Keep clear in your mind: 2 distinct branches, 2 distinct repos.
- Forces the team to dialogue/discuss proposal
- Results in “reject” (*close pull-request*), or “approve” (*merge*)

GitHub feature: the fork command

Makes a complete copy of...

Someone's existing repo on gitHub onto your gitHub

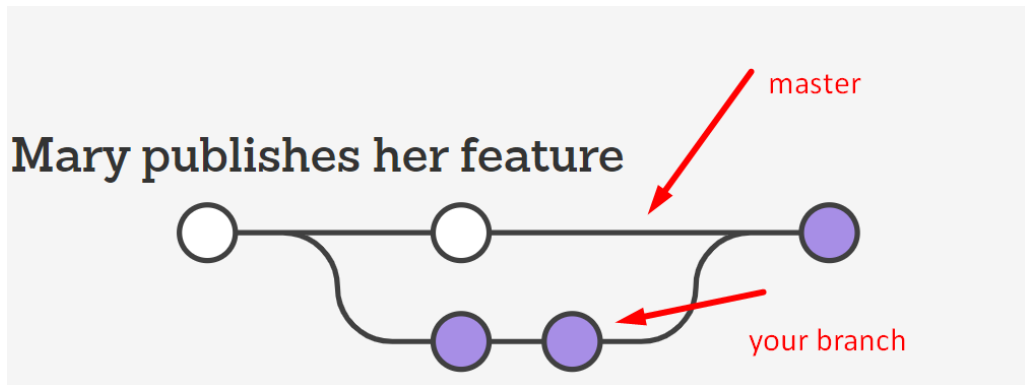
Purpose:

1. Propose changes to someone else's project (open source)
Fork, clone/make changes, submit a "pull-request" to the owner
2. Use someone's code as starting point for your own idea
Need license file to indicate how your code was obtained, or to be shared

The “feature branch” workflow

<https://www.atlassian.com/git/tutorials/comparing-workflows>

- One master branch
- Everyone creates a feature branching from master
- Combine with master:
 - merge locally to master, then push master, OR
 - push branch, and pull-request into master

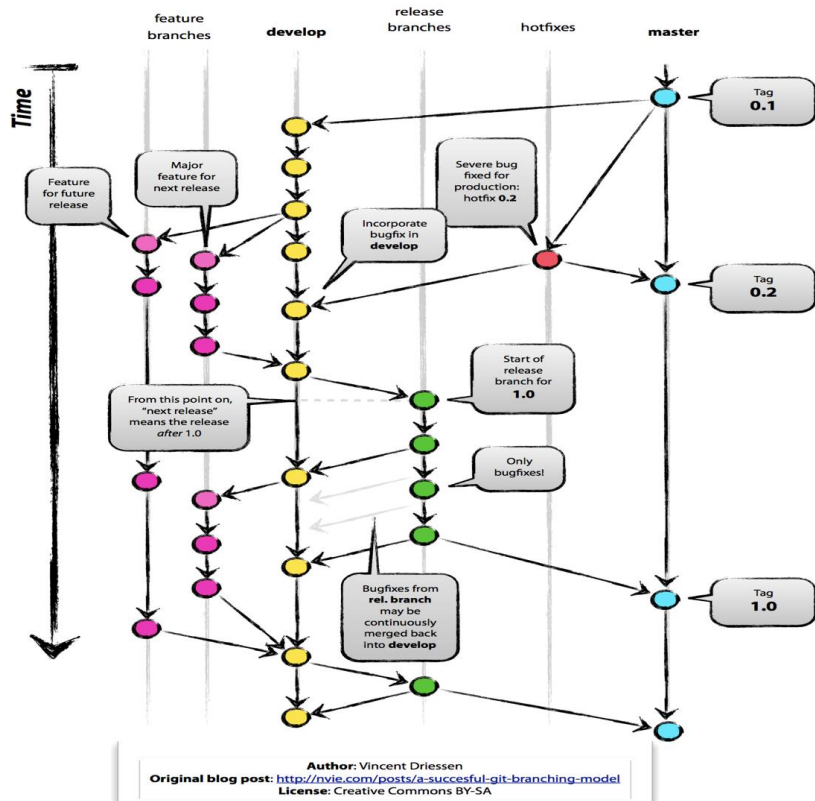


The “gitflow” workflow

<http://nvie.com/posts/a-successful-git-branching-model/>

Supporting branches:

- Release branch
- Hot-fix branch



Great references

- <https://git-scm.com/book/en/v2>
- <https://www.atlassian.com/git/tutorials/comparing-workflows>