Git Workflows

Git 工作流程

Jean Hu - 11 September 2017

Fork and Pull Request

Fork 和 Pull Request 皆不是 Git 原生的操作模式，而是 Git 服務網站衍伸出的新功能。

- Fork
  - 將整個 Git 檔案庫分出另一個版本。
  - Clone 不會記來源檔案庫， Fork 會保留來源的檔案庫，以便合併回原來的檔案庫。
  - 若參與一個沒有寫入權限的專案，可以 Fork 一份這個專案的副本到用戶的帳號下。
  - 透過這方式，開源專案就不需要把所有貢獻者加入來讓他們擁有寫入的權限。
  - 通常會有以下流程：
    1. 貢獻者 Fork 專案，對 Fork 出來的專案推送變更。
    2. 發出 Pull Request，把這些變更貢獻回原本的專案裡。
    3. Pull Request 開立一個能夠作程式碼審閱的討論串，供擁有者能和貢獻者討論。
    4. 擁有者將變更合併回原始專案。

- Pull Request
  - 當建立一個 Pull Request 時，來源專案的擁有者會收到有人建議專案變動的通知。
  - 專案的擁有者可以閱覽建議的變更，做以下決定：
    - 留下評論
      - 當維護者留下評論，建立這個 Pull Request 的人及所有關注這個 Respository 的人都會收到通知。
      - 貢獻者就可以依照評論得知需要做哪些調整才能讓擁有者接受這個變更。
      - 如果是在已發出的 Pull Request 中再加入新的提交並不會觸發通知，需要再留下一個評論以通知專案擁有者已完成所要求的修改。
    - 拒絕變更
    - 接受變更並且合併回檔案庫
      - 若檔案發生衝突，需將目標分支合併到進行 pull request 的分支：
        1. 將來源檔案庫新增為遠端
           ```
           $ git remote add <來源檔案庫簡稱> <來源檔案庫>
           ```
        2. 從這個遠端擷取最新的內容
           ```
           $ git fetch <來源檔案庫簡稱>
           ```
        3. 把主要分支合併進本地端分支
           ```
           $ git merge <來源檔案庫簡稱>/<分支名稱>
           ```
        4. 修正產生的衝突
        5. 再推送回同一個遠端分支
           ```
           $ git add <變更檔案>
           $ git commit
           $ git push
           ```
      - 做完上述步驟，Pull Request 會自動更新並檢查是否能乾淨的合併。

Centralized Workflow

- 適合工作團隊
  - 剛從 SVN 轉到 Git 的團隊，可直接採用如同 SVN 一樣的工作模式來開發專案。
  - 團隊人數少。
  - 更動沒有那麼頻繁。
  - 團隊人員彼此溝通方便。
  - 專案不需要同時維護多個版本。

- 分支
  - 一個長期分支：主分支（master）

- 工作方式
  1. 從遠端檔案庫 clone 專案至本地檔案庫
     ```
     $ git clone
     ```

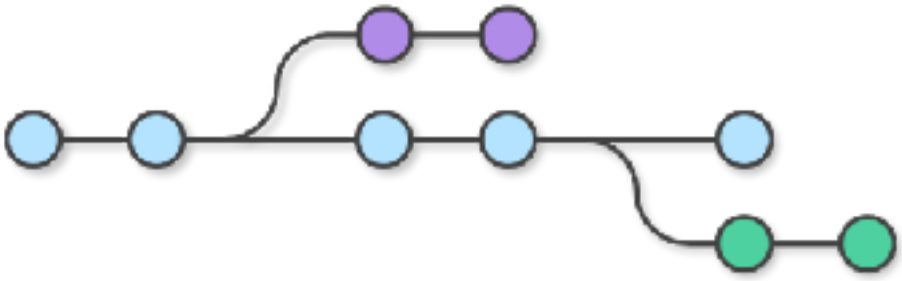
  2. 在本地檔案庫開發功能及提交更新
     ```
     $ git add
     $ git commit
     ```

  3. 同步遠端檔案庫至本地檔案庫，並且使用 rebase 的方式合併
     ```
     $ git fetch
     $ git rebase
     ```

     或是
     ```
     $ git pull –rebase
     ```

  4. 將主分支推送至遠端
     ```
     $ git push
     ```

Feature Branch Workflow



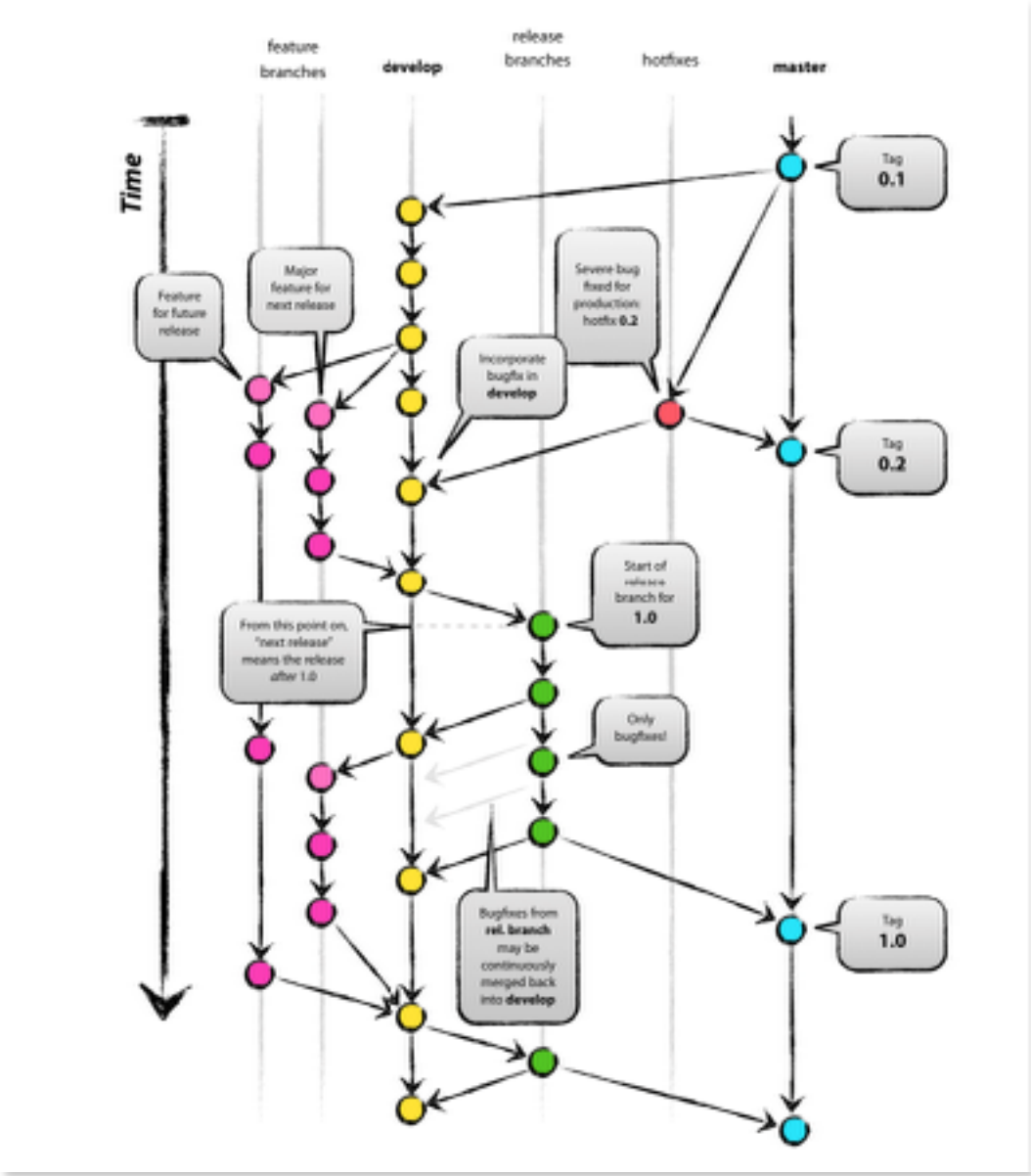
- 適合工作團隊
  - 專案週期短、更新頻繁。
  - 專案不需要同時維護多個版本。
  - 開發團隊固定且有一定規模。
  - 有多個新功能跟問題需並行開發。
  - 代碼審查要求較高。

- 分支
  - 一個長期分支：主分支（master）
  - 一種短期分支：功能分支（feature）

- 工作方式
  1. 從遠端檔案庫 clone 專案至本地檔案庫
     ```
     $ git clone
     ```
  2. 每次開發新功能或修復 bug 都會新建立一個 feature branch
     ```
     $ git checkout -b <分支名稱>
     ```
  3. 在本地檔案庫開發功能及提交更新
     ```
     $ git add
     $ git commit
     ```
  4. 將 feature branch 推送至遠端檔案庫
     ```
     $ git push
     ```
  5. 發送 Pull Request 請求專案維護者合併到 master branch
  6. 結束 Pull Request，將 feature branch 合併至 master branch ，發佈新功能到遠端檔案庫

  - 功能開發應該在另外一個分支，不應該影響 master 分支。
  - 有效應用 Pull Request，在開發階段就能接收反饋。

Gitflow Workflow



- 適合工作團隊
  - 專案規模較大。
  - 開發團隊固定且有一定規模。
  - 有並行開發需求。
  - 有專門的檔案代碼庫維護員。
  - 團隊對於 Feature、Release、Hot fix 有統一定義標準。

- 分支
  - 兩個長期分支（歷史分支）：主分支（master）、開發分支（develop）
  - 三種短期分支：功能分支（feature）、發布分支（release）、維護分支（hotfix）

- 工作方式
    1. 建立 Historical Branches
        - master 分支保存官方正式發佈歷史。
        - develop 分支作爲功能集成的分支，衍生出各個 feature 分支。

    2. 每當開發一個新功能，從 develop 分支切出一個 feature 分支
        - 當 feature 分支功能開發完成後，會合併回 develop 分支。
        - feature 分支不與 master 分支直接交互。

    3. 當 develop 有足夠功能可以發布時，從 develop 分支切出一個 release 分支
        - release 分支主要功能為清理釋放、使用者測試、Bug 修復和更新文檔。
        - 從切出時間點開始之後，新的功能不能再加到這個分支上。
        - 一旦準備好上架，將 release 分支合併至 master 分支並標記一個版本號 Tag。

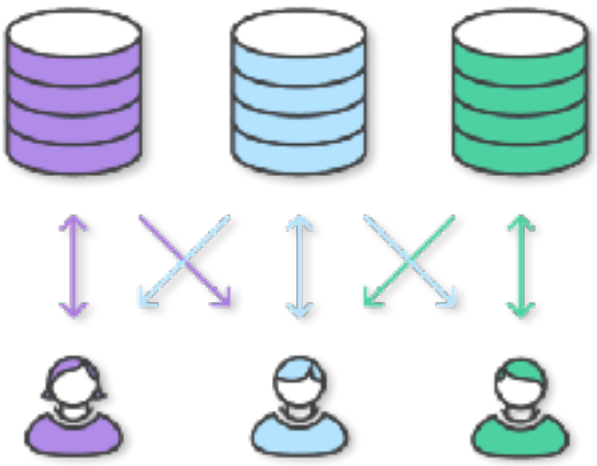
        ```
        $ git tag -a <標籤名稱> -m <標籤資訊> master
        $ git push origin master --tags
        ```

        另外， release 分支做的修改也要合併回 develop 分支。

    4. 若發布產品出現 Bug 需修復，從 master 分支切出一個 hotfix 分支
        - 快速對已發佈產品做修復 Bug 或微調功能。
        - 從 master 分支直接衍生出來。
        - 一旦完成修復，將 hotfix 分支合併回 master 分支並標記一個版本號 Tag，hotfix 分支也要合併至develop 分支。

- 2-4 會同步進行且有循環週期。
- 以含義及目的性來看，feature 分支加上 develop 分支其實就是 Feature Branch Workflow。

Forking Workflow



- 適合工作團隊
  - 常用於開源專案。
  - 開發者需要 Fork 出一個衍伸的檔案庫。
  - 開發團隊不固定。
  - 分散式的 workflow。
  - 為大型、自發性團隊提供安全的協作流程。

- 分支
  - 與來源遠端檔案庫所用的 workflow 切出來的分支相同。

- 工作方式
  1. 貢獻者 Fork 官方遠端檔案庫來創建它的拷貝，然後存放在個人檔案庫中
     `$ git clone`
  2. 貢獻者需要時時同步官方遠端檔案庫確保代碼同步
  3. 當貢獻者準備好發佈新功能時，推送至個人檔案庫
     `$ git push`
  4. 從個人檔案庫發送 Pull Request 至官方檔案庫請求合併
  5. 維護者審查代碼，若同意則將更動合併至官方檔案庫

  - Forking Workflow 與上述討論的工作流很不同，不是多個開發者共享一個遠端檔案庫，而是每位開發者擁有自己的檔案庫。也就是說每位貢獻者都有兩個檔案庫，個人檔案庫和遠端共享檔案庫。
  - 開發者可以不用直接參與到專案內，若想要貢獻代碼，只需要 Fork 一份目標檔案庫，做完修改後，再提交 Pull Request 給來源檔案庫。
  - 開源專案無需開放貢獻者寫入權限，也可以審查代碼。

參考資料

- Pro Git 2nd Edition
- A successful Git branching model
- Smart branching with SourceTree and Git-flow
- Comparing Workflows
- 大话 Git 工作流
- Git工作流指南