Cách dùng git làm việc nhóm

Bắt đầu dự án:

```
git clone <link repository>
```

Quy trình push:
Sẽ có file `.gitignore` để kiểm tra xem các file hay folder nào sẽ được push hay là không được push

Để tham gia vào một repository của người khác sẽ có hai trường hợp:
Trường hợp 1 : Trong dự án local hiện tại không có git của một repository nào thì dùng lệnh sau:

```
git remote add origin <link repository>
```

Trường hợp 2 : Trong dự án local hiện tại đã có git của một repository thì dùng lệnh sau:

```
git remote remove origin
git remote add origin <link repository>
```

```
git add .
git commit -m <message>
```

*Chú ý : cách commit. Sẽ có issue được giao cho từng người trên phần issues của github và cách commit là:

```
git commit -m "#[number issue] - [name] - task"
```

*Demo phần issue trong github

```
git push
```

Git branch:
Giờ nếu đã tham gia vào repository rồi thì để thấy tất cả các nhánh thì:

```
git fetch
```

Tạo nhánh - đổi nhánh - shorthand:

```
git branch <tên nhánh>
git checkout <tên nhánh>
git checkout -b <tên nhánh>
```

Tạo nhánh con từ nhánh chính:

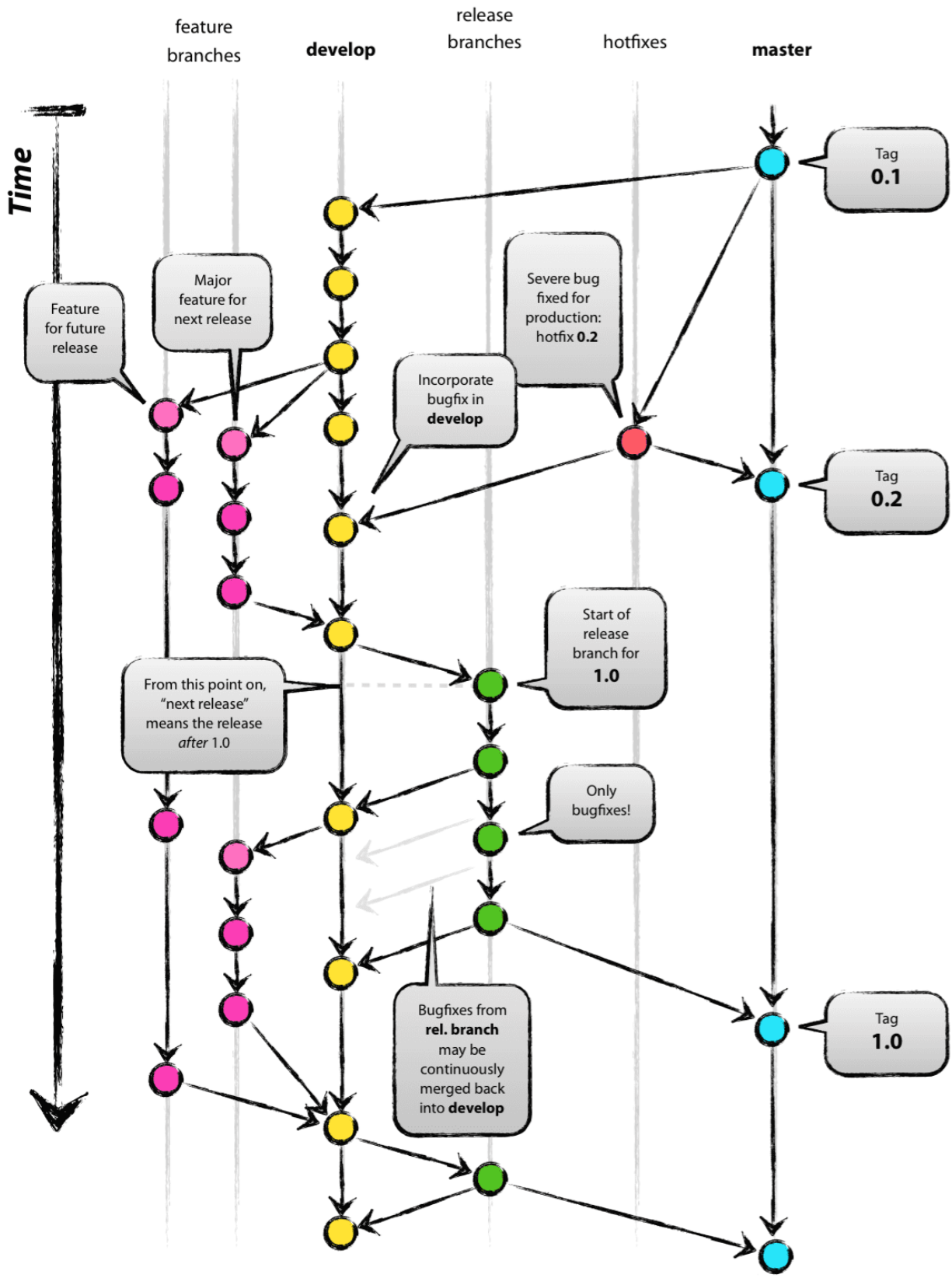
```
git checkout -b <tên nhánh con> <tên nhánh chính>
```

Xem git tree:

```
 git log --oneline --graph --color --all --decorate
```

Git merge:

```
git merge <tên nhánh hiện tại> <tên nhánh merge>
```



Với mỗi lần làm xong một feature thì cần xóa cái nhánh đó đi:

Nếu nhánh đó đã được push lên repo remote rồi

```
git push origin --delete <tên nhánh>
```

Nếu nhánh đó còn ở local

```
git branch -d <tên nhánh>
```

*Giải thích sơ đồ, giải thích nhánh hotfixes : Người dùng feedback mình sửa liền
*Nhớ muốn merge vào main thì phải pull request

Thống nhất cấu trúc folder:

- App
  - Utils (Everything is shared between components or features like function, hook, etc …)
  - Features
  - Components (Always have this folder to contain every components can be reused in whole app)
  - Api (API request declaration)
  - Home
    - Comopnents
    - Features
    - … (etc)
  - Admin
    - …
  - User
    - …
  - User-Guest
    - …
- Public
- Reudx

Code convention

- Đặt tên biến:
  - Sử dụng cú pháp "camelCase" (chữ cái đầu tiên của từ đầu tiên viết thường, chữ cái đầu tiên của các từ tiếp theo viết hoa) cho biến và hàm.
  - Sử dụng tên biến có ý nghĩa, mô tả rõ mục đích của biến.
  - Ví dụ: fullName, numberOfItems, calculateTotal.
- Đặt tên hằng số:
  - Sử dụng chữ hoa và dấu gạch dưới (_) để phân tách các từ.
  - Ví dụ: MAX_SIZE, DEFAULT_VALUE, PI.
- Đặt tên lớp:
  - Sử dụng cú pháp "PascalCase" (chữ cái đầu tiên của từ đầu tiên và chữ cái đầu tiên của các từ tiếp theo viết hoa) cho tên lớp.
  - Tên lớp nên là danh từ hoặc cụm danh từ.
  - Ví dụ: Customer, OrderProcessor, UserManager.

- ➔ Đặt tên phương thức:
  - ◆ Sử dụng cú pháp "camelCase" cho tên phương thức.
  - ◆ Tên phương thức nên miêu tả hành động được thực hiện bởi phương thức.
  - ◆ Ví dụ: calculateTotalPrice(), getUserInfo(), addItemToCart().
- ➔ Đặt tên tập tin:
  - ◆ Sử dụng chữ thường và dấu gạch dưới (_) để phân tách các từ.
  - ◆ Tên tập tin nên phản ánh mục đích và nội dung của tập tin.
  - ◆ Ví dụ: user_model.py, database_utils.java.

Cách đặt tên HTML class

Sử dụng chữ thường:

```
<div class="main-content"></div>
<div class="sidebar-nav"></div>
<div class="header-section"></div>
```

Sử dụng cú pháp "kebab-case":

```
<div class="main-content"></div>
<div class="sidebar-nav"></div>
<div class="header-section"></div>
```

Đặt tên có ý nghĩa:

```
<div class="main-content"></div>
<div class="contact-info"></div>
<div class="product-list"></div>
```

Sử dụng tên class liên quan đến kiểu giao diện:

```
<button class="button-primary"></button>
<div class="container-fluid"></div>
<div class="card-featured"></div>
```

Tránh đặt tên class trùng lặp:

```
<div class="card"></div>
<div class="card-featured"></div>
```

Sử dụng tên class ngắn gọn:

```
<div class="header"></div>
<div class="nav"></div>
<div class="promo"></div>
```

Nếu có thành phần con:

```
<div class="card">
  <h2 class="card-title">Tiêu đề thẻ</h2>
  <p class="card-description">Mô tả về thẻ</p>
</div>
```

Cách comment:

Quy tắc chung : Viết bằng tiếng Anh , đúng chính tả rõ ràng ngắn gọn dễ hiểu
Đối với biến:
Tên người đọc , chức năng của biến
// maximum length for user input
Const MAX_LENGTH_TEXT = 1000
Đối với hàm:
Các hàm trong class cũng được thể hiện một cách tương tự, ngoại trừ có phần giải thích thêm cho các thông số truyền vào.

```
/**
 * This method is used to add two integers. This is
 * a the simplest form of a class method, just to
 * show the usage of various javadoc Tags.
 * @param firstNumber This is the first paramter to addNum method
 * @param secondNumber This is the second parameter to addNum method
 * @return int This returns sum of firstNumber and secondNumber.
 */
public int addNum(int firstNumber,int secondNumber){
   return firstNumber+secondNumber;
}
```

Đối với các dòng code bình thường:
Ghi chú càng rõ ràng, dễ hiểu càng tốt cho các dòng code
Các dòng xử lí lặp, rẽ nhánh, giải quyết vấn đề... đều phải được giải thích rõ ràng. Hãy xem phiên bản hàm add được trình bày ở trên, với các lời giải thích rõ ràng:

```
public int addNum(int firstNumber,int secondNumber){
   // If the first parameter is 0, return 0.
   if(firstNumber==0){
      return 0;
   }else{
      // If the second parameter is 0, return 1.
      if(secondNumber==0)
        return 1;
      else
        return firstNumber+secondNumber;
   }
}
```