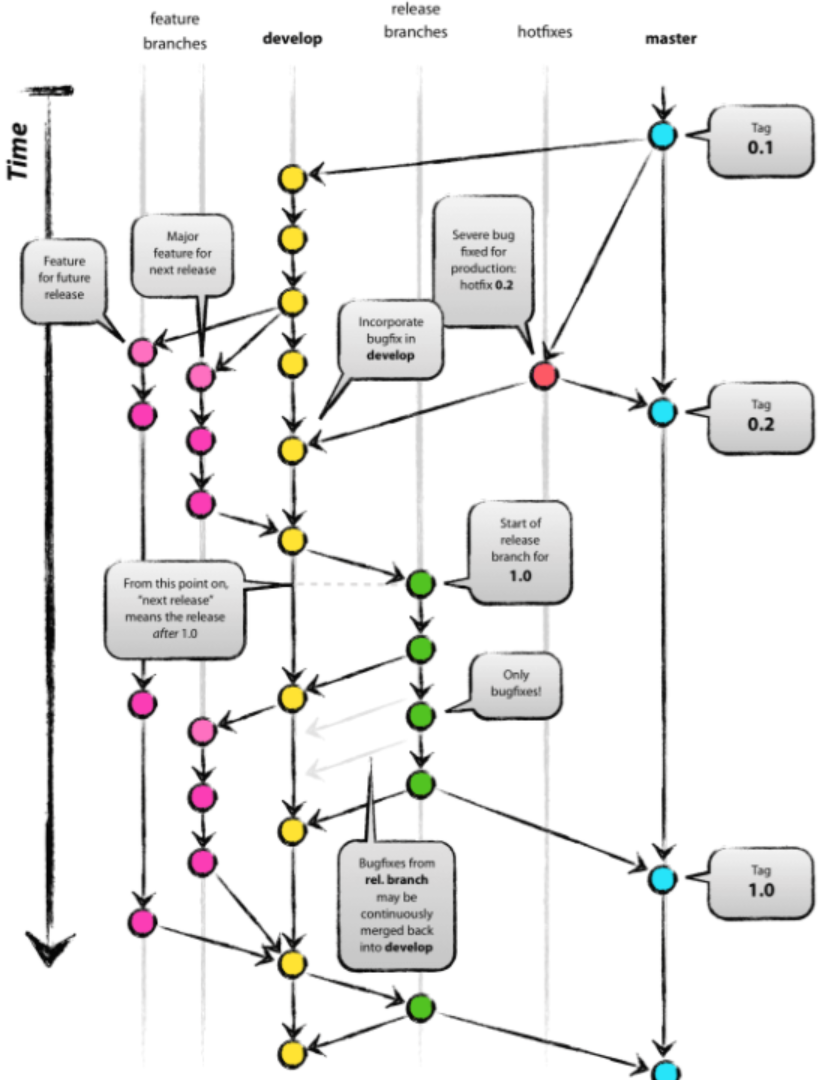
* 브렌치 전략 : Git-flow 방식

**1. branch 종류**

* master : 기준이 되는 branch로 제품을 배포하는 기준점이 되는 브랜치 입니다. 즉, 배포 가능한 상태만을 관리한다.
* develop : 개발 브랜치로 개발자들이 이 브랜치를 기준으로 각자 작업한 기능들을 합(Merge)칩니다.
* feature : 단위 기능을 개발하는 브랜치로 기능 개발이 완료되면 develop 브랜치에 합칩니다.
* release : 배포를 위해 master 브랜치로 보내기 전에 먼저 QA(품질검사)를 하기위한 브랜치 입니다.
* hotfix : master 브랜치로 배포를 했는데 버그가 생겼을 떄 긴급 수정하는 브랜치 입니다.

**2. gitflow**  


**3. 브랜치 명칭 정의**  
\*Git 브랜치명은 브랜치 용도에 따라 master, develop, feature, release, hotfix 브랜치별로 네이밍 규칙을 따른다.

* feature, release, hotfix 브랜치는 아래와 같이 접두어를 붙여준다. feature는 기능명을 표현하는 단어로 단어(알파벳 소문자)와 구분자(-)를 사용하고 release, hotfix에는 버전명을 사용한다.
* 브랜치별로 다음과 같은 규칙을 사용한다.
* Production branch name  
  : master
* Development branch name  
  : develop
* Feature 브랜치 접두어  
  : feature/
* Release 브랜치 접두어  
  : release/
* Hotfix 브랜치 접두어  
  : hotfix/

**\*샘플**

* Feature branch  
  : feature/this-is-awesome

**4. 사용 예시**

1) 일단 master 브랜치에서 시작을 합니다.

2) 동일한 브랜치를 develop에도 생성을 합니다. 개발자들은 이 develop 브랜치에서 개발을 진행합니다.

3) 개발을 진행하다가 회원가입, 장바구니 등의 기능 구현이 필요할 경우 A개발자는 develop 브랜치에서 feature 브랜치를 하나 생성해서 회원가입 기능을 구현하고 B개발자도 develop 브랜치에서 feature 브랜치를 하나 생성해서 장바구니 기능을 구현합니다.

4) 완료된 feature 브랜치는 검토를 거쳐 다시 develop 브랜치에 합칩니다.(Merge)

5) 이제 모든 기능이 완료되면 develop 브랜치를 release 브랜치로 만듭니다. 그리고 QA(품질검사)를 하면서 보완점을 보완하고 버그를 픽스합니다.

6) 모든 것이 완료되면 이제 release 브랜치를 develop 브랜치와 master 브랜치로 보냅니다. master 브랜치에서 버전추가를 위해 태그를 하나 생성하고 배포를 합니다.  
배포를 했는데 미처 발견하지 못한 버그가 있을 경우 hotfixes 브랜치를 만들어 긴급 수정 후 태그를 생성하고 바로 수정 배포를 합니다.

**[참조 URL]**

* <https://woowabros.github.io/experience/2017/10/30/baemin-mobile-git-branch-strategy.html>
* <https://uxgjs.tistory.com/183>-<https://gmlwjd9405.github.io/2018/05/11/types-of-git-branch.html>