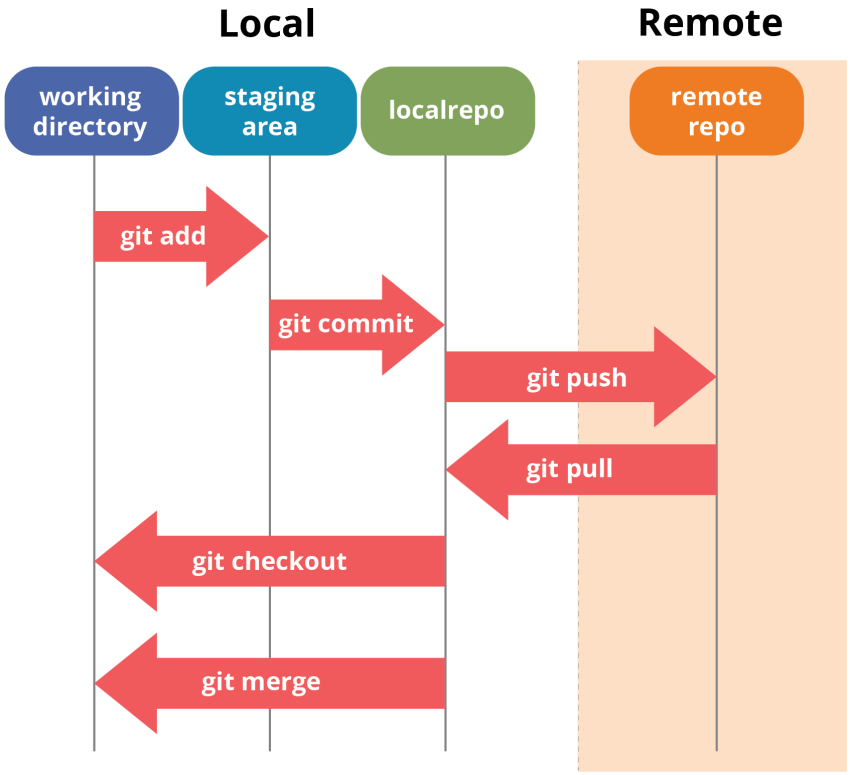
[시스템 학습] Git 활용법 및 실습

> 진행: 최광호 컨설턴트
>
> 날짜: 2021.01.07

- 목차
  1. Git 개요
  2. Git branch
  3. Git branch 전략 및 GitFlow
  4. Git 사용 리뷰
  5. Git wrap up
  6. Git branching 실습
  7. Q&A

1. Git 개요

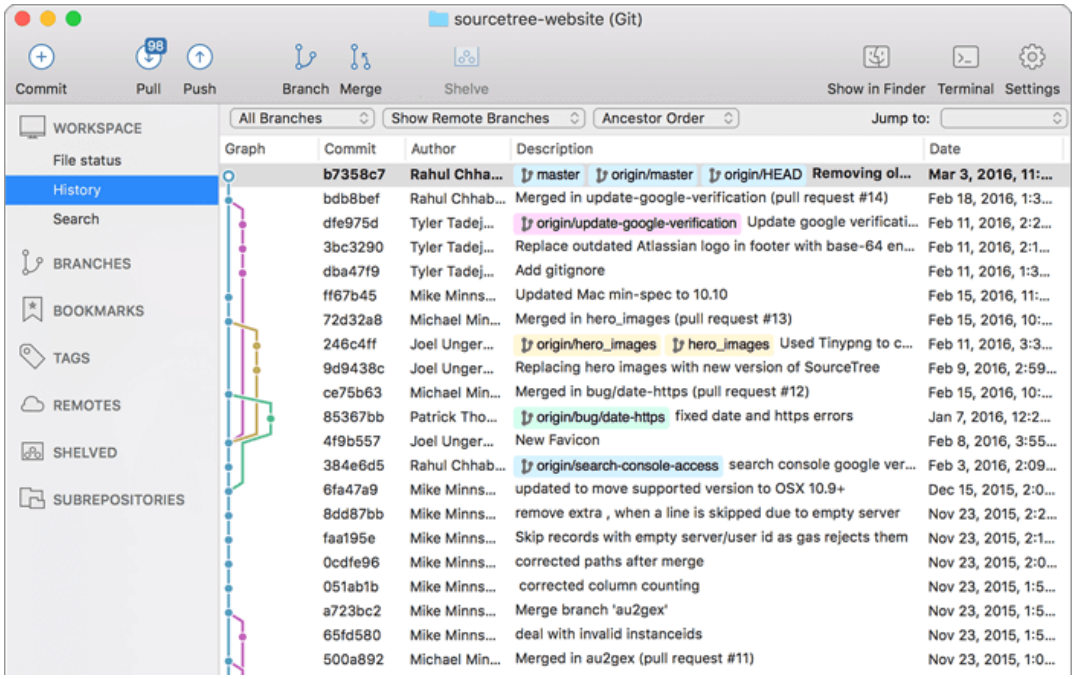
- 기업에서 주니어 개발자를 채용할 때 자격요건으로 git 사용 경험을 명시해놓는 경우가 많다.
  - rocketpunch에서 채용 공고들 중 `git`이라는 단어가 들어간 공고들은 약 25%로, 가장 많이 언급되는 키워드 중 하나다.
- 최종, 최종2, 리얼_최종, 최종3333과 같은 방식으로 파일 관리한 경험이 있을 것이다. 하지만 개발자들은 **Git**을 통해 코드를 관리한다!
- Git 그냥 쓰면 되는거 아닌가요?
  - Git은 협업을 위해 쓰이기 때문에 원활한 협업을 위해 지켜야하는 약속과 컨벤션들이 있다.
- Git 구조



- Git 명령어
  - google에 `git cheat sheet`이라고 검색하면 git 명령어들이 잘 정리된 자료들을 많이 찾아볼 수 있다.
  - 직접 타이핑해보는 연습을 통해 명령어들을 익히는 것이 좋다.
- Gitlab cheat sheet

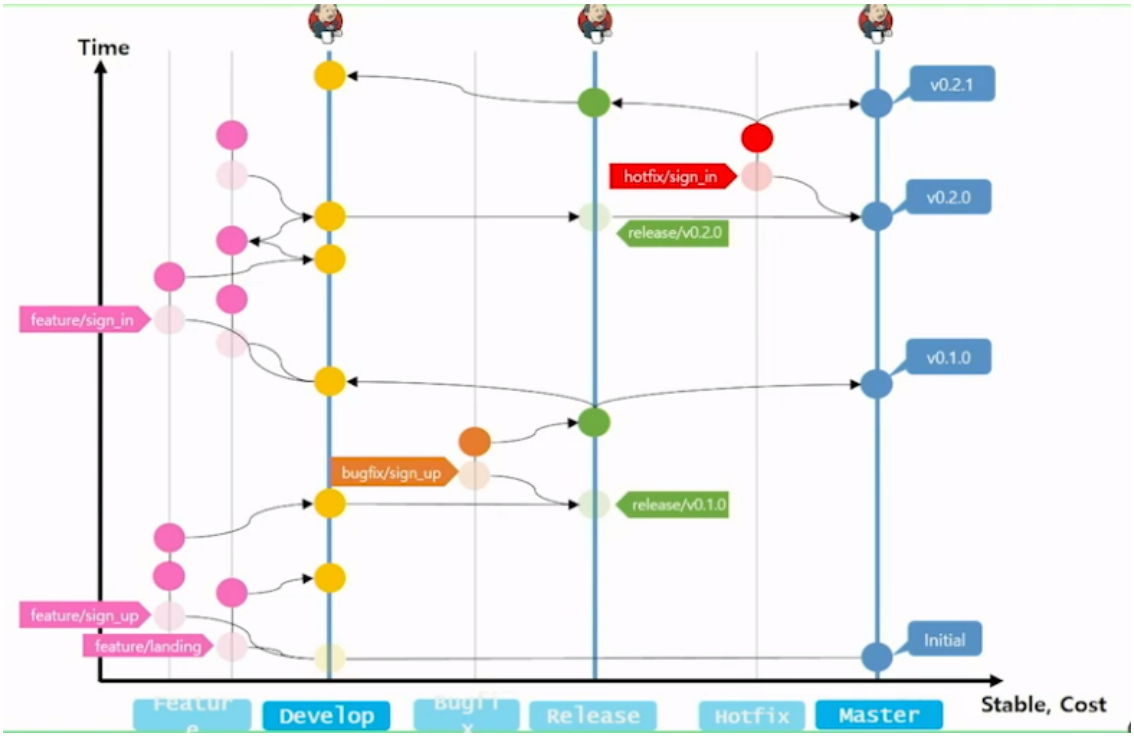
2. Git branch

- Git에서 제일 중요한 단위이자 버전 관리의 핵심은 `commit` 이다.
- `branch`는 코드를 관리하기 위해서 분기되는 흐름을 만들어 한 저장소에서 여러명이 동시에 협업이 가능하도록 한다. branch는 각자 이름을 가지고 있으며, 브랜치 이름이 commit에 달리면 이를 이용해서 merge와 같은 작업을 쉽게 할 수 있다.
- Git source tree (commit tree)
  - 프로젝트 소스 코드가 어떠한 변화를 거쳤는지 볼 수 있으며, git flow를 보면서 로컬 repo에 한해서 이전 버전으로 돌리거나 이것저것 변경사항을 적용해보며 실험을 해볼 수 있게 해준다.



3. Git branch 전략 및 GitFlow

- GitFlow란?
  - 업계에서 보편화된 git branch 전략 중 하나로 git을 사용하는 best practice 중 하나이다.



- x축은 cost로, 오른쪽으로 갈 수록 문제가 생기면 비용이 많이 드는 것이며, y축은 시간의 순서를 나타낸다.
- 개발자는 주로 `develop`에서 작업을 하다가 출시를 준비 시작하면 `release` 브랜치로 옮겨 테스터, QA, 기획자 등이 배포된 서비스를 사용해보면서 품질 보증 활동을 한다. 보통 이때는 `staging server`라는 연습용 서버에 서비스를 올려서 테스트를 진행한다.
- `develop`과 `master`는 기본 브랜치들로 고정되어있지만 나머지 브랜치들은 임시로 만들어졌다가 목적을 달성하면 소멸되는 서포팅 브랜치다.
- `develop`에서 개발된 것은 알파 테스트용, `release`에서 개발된 것은 베타 테스트용, `master`에는 정식 출시용 서비스가 있다고 보면된다. 그래서 각 브랜치에서 `jenkins`와 같은 CI/CD 툴을 통해 각각 서버를 운영한다.

- Git flow 시작하기

```
$ git flow init
Initialized empty Git repository in
C:/Users/multicampus/projects/test/git-flow-test/.git/
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] v
```

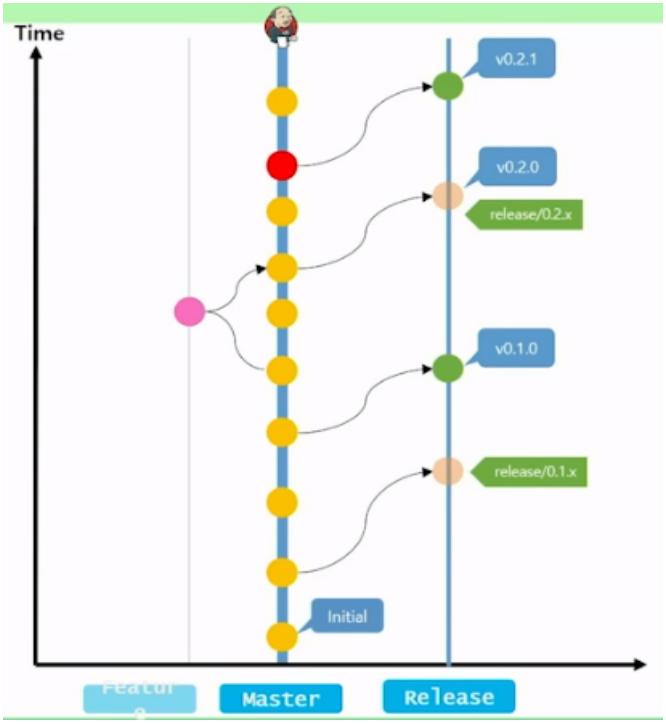
- branch name for production releases(제품을 출시하기 위한 브랜치 이름) 설정
- branch name for next release development(다음 출시를 위해 준비하는 브랜치 이름) 설정
- 기타 등등 네이밍 컨벤션을 설정

```
multicampus@SSAFY MINGW64 ~/projects/test/git-flow-test (develop
$ git branch
* develop
  master

$ git flow feature start sign_up
Switched to a new branch 'feature/sign_up'

Summary of actions:
- A new branch 'feature/sign_up' was created, based on 'develop'
- You are now on branch 'feature/sign_up'

Now, start committing on your feature. When done, use:
     git flow feature finish sign_up
```

```
# 작업 시작
git flow feature start <브랜치 이름>

# 작업이 끝나면
git flow feature finish <브랜치 이름>
```

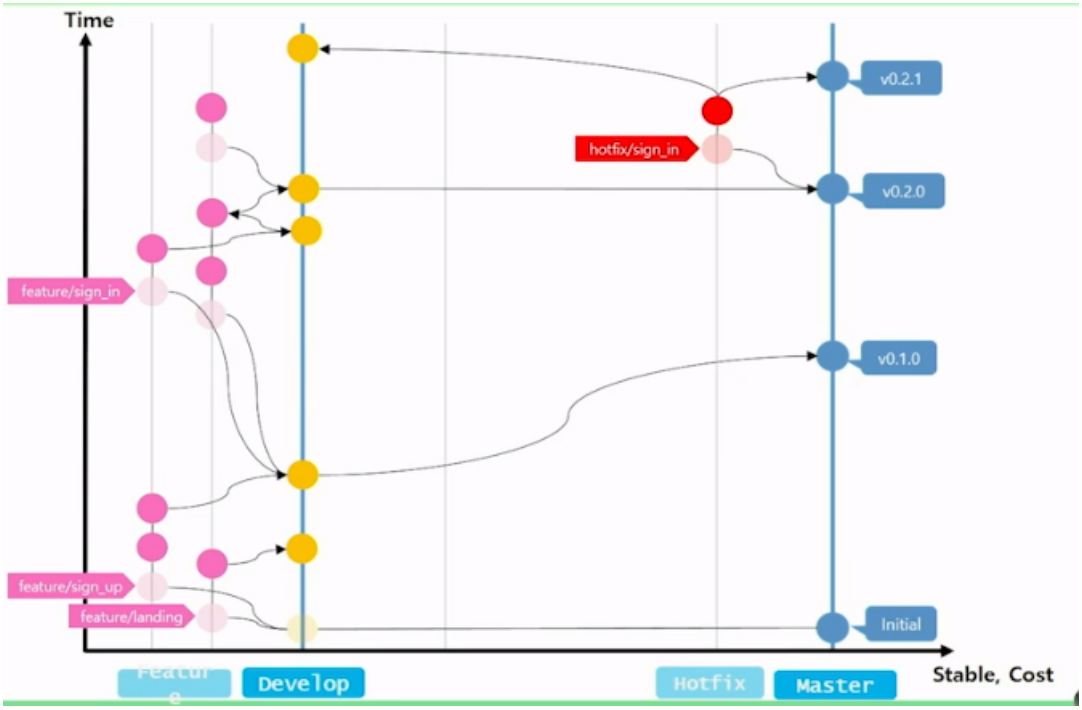
  - **하지만 SSAFY에서 프로젝트 진행 시에는 git flow 명령어를 사용하지 않는다**. 왜냐하면 git flow 명령어를 사용하면 자동으로 머지되기 때문이며, 우리는 **코드 리뷰를 위해 merge request를 보낼 예정**이다.

- 그 외 git branch 전략
  - Trunk based 전략



    - `master`에서 모든 작업이 이루어진다. master는 언제나 배포될 수 있는 상태이며, 브랜치를 딸 수도있지만 아주 잠깐 사용했다가 다시 master로 돌아오는 흐름을 가진다.
    - 이러한 전략은 성숙도가 있는 시니어 개발자가 많은 팀이 활용 가능하다. 왜냐하면 이렇게 flow를 깔끔하게 유지하기 힘들기 때문이다.

- **SSAFY 2학기 프로젝트 진행 시 권장하는 git flow 전략**



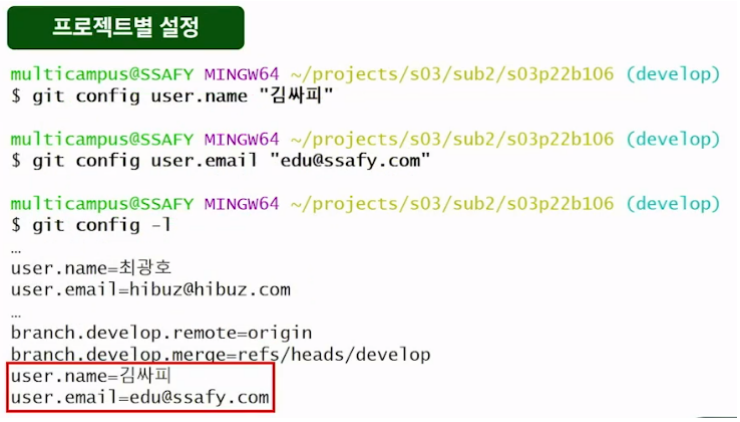
- `develop`과 `master`를 두고 개발하며, 상세 기능 개발 시에 그에 맞는 `feature` 브랜치를 파서 개발한다.
- 우리는 운영 서버는 없고 개발 서버만 있지만 그래도 스프린트가 끝날 때마다 master로 머지해서 마치 실제 출시하는 상황을 시뮬레이팅한다.
- 버전 태깅도 해보면 좋다.
- 필요에 따라 `hotfix`를 사용하는 것도 좋다.

4. Git 사용 리뷰

- Git commit 사전 작업

전역(global) 설정

```
multicampus@SSAFY MINGW64 ~/projects
$ git config --global user.name "김싸피"

multicampus@SSAFY MINGW64 ~/projects
$ git config --global user.email "edu@ssafy.com"

multicampus@SSAFY MINGW64 ~/projects
$ git config -l
...
credential.helper=manager
user.name=김싸피
user.email=edu@ssafy.com
```



  - 사전 config 세팅을 통해 gitlab 커밋 내역에 이름이 알맞게 남도록 한다.
- git push -f origin develop

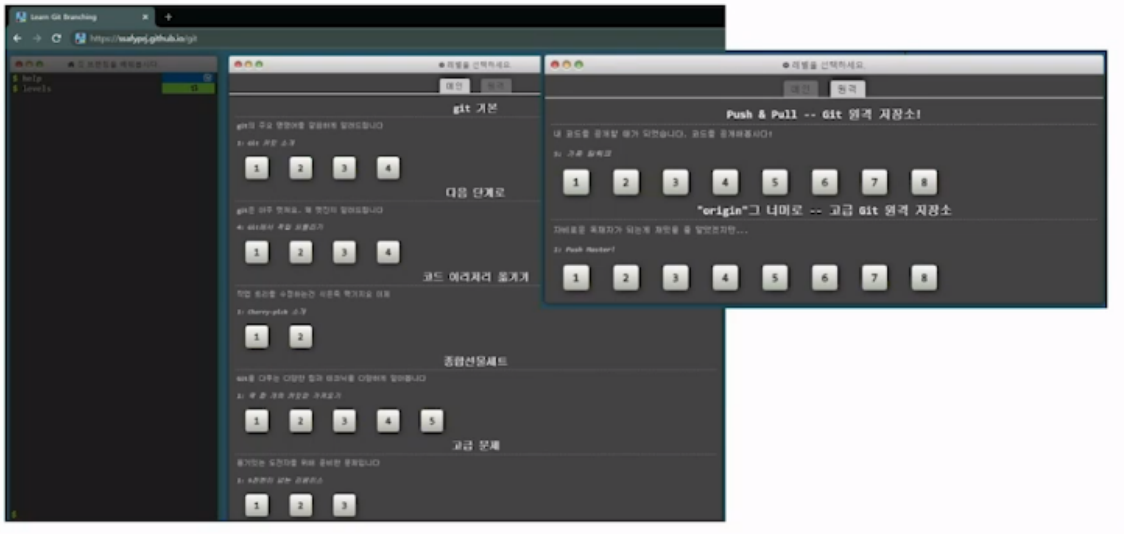
  - `git push -f origin develop`을 이용해서 강제로 커밋 내역을 푸쉬하는 것은 지양한다! 강제로 푸쉬하면 오른쪽과 같이 다른 사람의 커밋 내역이 사라져버리거나 커밋 내역이 꼬여버릴 수도 있다.
- .gitignore 관리
  - gitignore.io을 통해 .gitignore 파일 내용을 자동 생성하는 기능을 활용한다.

5. Git wrap up

- 개발에 방해되지 않도록 git 기본 개념/명령어를 잘 익히기
- git branch를 사용한 코드관리의 흐름(flow) 이해
- 회사/팀별 상황에 맞는 git branch 전략 선택
- 팀 별 합의된 방식에 맞춰 일관된 git 사용 노력 필요
- 궁금하거나 자주 마주하는 케이스를 위한 연습 필요

6. Git branching 실습

- ssafyprj.github.io/git에서 연습해보기



- 심화 학습
  - git with d3: 명령어를 시각화해주는 프로그램
  - git 문서: 더 심층적으로 공부해보고 싶은 명령어가 있으면 여기를 참고할 것

7. Q&A

1. 버전 태그 남기는 일은 팀에서 버전 관리 자율로하나요? 아니면 암묵적인 룰이있나요?
   - Semantic versioning이라는 컨벤션이 있어 메이저, 마이너, 패치에 따라 버저닝 숫자를 올립니다.