

git, bitbucket

11번가 2021년 개발 신입 역량육성과정

(Developer's Beginning Course)

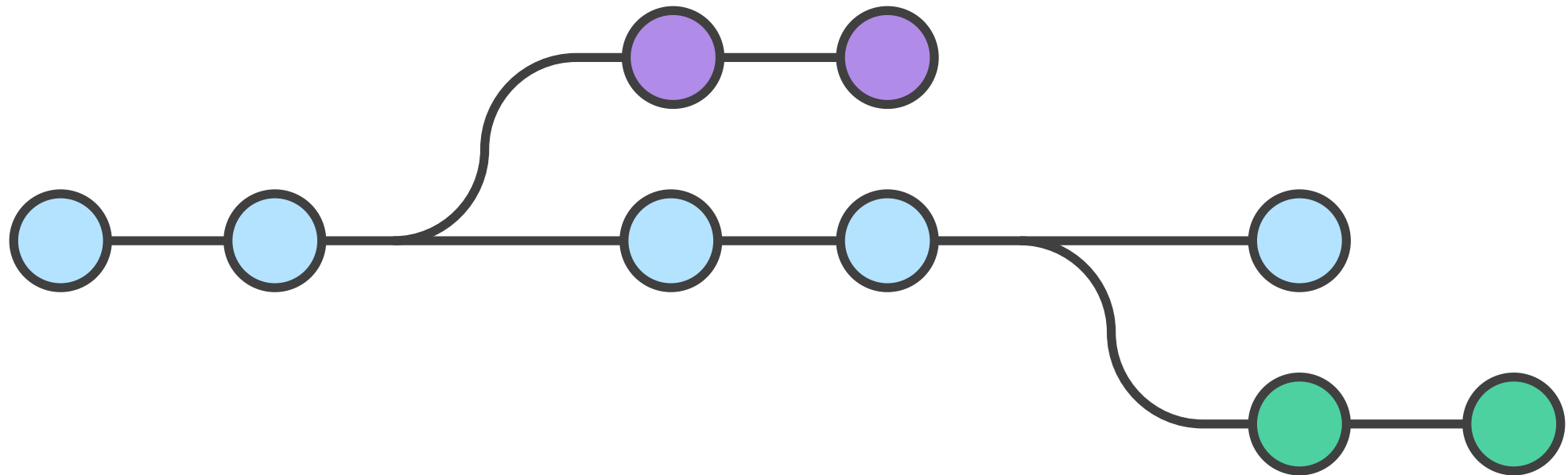
Previously..

- git의 원활한 사용을 위한 CLI shell, vim command
- 개발문서 작성을 위한 Markdown 작성법
- 원활한 git 사용 및 개발을 위한 환경 구성
- git의 구성요소와 process 실습
- commit convention 습관화
- repository 필수 요소의 작성

Today, We will learn about..

- Branch 사용법
- Branch 전략: git flow
- 상황별 되돌리기
- trello로 스크럼보드 관리하기
- 협업 하는 법: Forking Workflow

Branch



Branch

- 분기점을 생성하여 독립적으로 코드를 변경할 수 있도록 도와주는 모델

master

```
print('hello' + ' ' + 'world')
```

develop

```
words = ['world', 'hello']  
print(' '.join(words[:-1]))
```

Branch(1)

Show available local branch

```
$ git branch
```

Show available remote branch

```
$ git branch -r
```

Show available All branch

```
$ git branch -a
```

Branch(2)

Create branch

```
$ git branch stem
```

Checkout branch

```
$ git checkout stem
```

Create & Checkout branch

```
$ git checkout -b new-stem
```

make changes inside [readme.md](#)

```
$ git commit -a -m 'edit readme.md'
```

```
$ git checkout master
```

merge branch

```
$ git merge stem
```

Branch(3)

delete branch

```
$ git branch -D stem
```

push with specified remote branch

```
$ git push origin stem
```

see the difference between two branches

```
$ git diff master stem
```


Practice(1)

- Spiderman.md를 생성하고 다음의 정보를 배역을 맡은 배우별로 브랜치를 생성하여 이를 시각화 하세요.
- 완결된 브랜치는 master 브랜치로 merge 해야 합니다.
- 각 commit은 개봉연도 순서대로 존재해야 합니다.
- AndrewGarfield 브랜치는 master의 첫 commit 에서 TobeyMaguire 브랜치와 함께 시작해야 합니다.(리부트이므로..)
- Format

```
# {Movie Name}  
- Year:  
- Name:
```

- TobeyMaguire branch

```
# Spider-Man 1
- Year: 2002
- Name: Peter Benjamin Parker

# Spider-Man 2
- Year: 2004
- Name: Peter Benjamin Parker

# Spider-Man 3
- Year: 2007
- Name: Peter Benjamin Parker
```

- AndrewGarfield branch

```
# Amazing Spider-Man 1
- Year: 2012
- Name: Peter Benjamin Parker

# Amazing Spider-Man 2
- Year: 2014
- Name: Peter Benjamin Parker
```

- Tom Holland branch

```
# Captain America: Civil War
- Year: 2016
- Name: Peter Benjamin Parker

# Spider-Man: Home Coming
- Year: 2017
- Name: Peter Benjamin Parker

# Avengers: Infinity war
- Year: 2018
- Name: Peter Benjamin Parker

# Avengers: Endgame
- Year: 2019
- Name: Peter Benjamin Parker

# Spider-Man: Far From Home
- Year: 2019
- Name: Peter Benjamin Parker
```

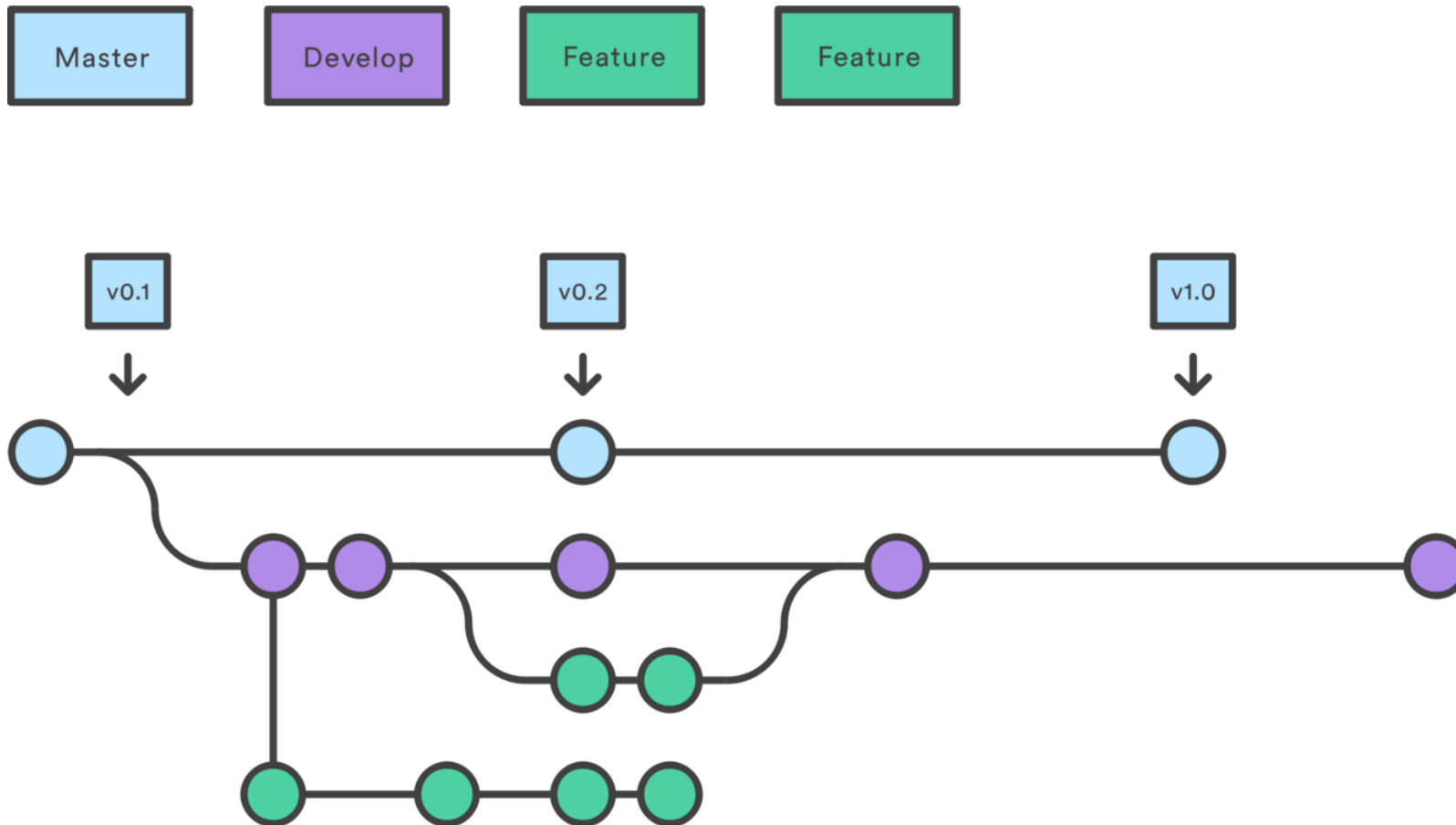
Additional Practice

- Venom branch
- Into the Spider-verse branch(Miles Morales)

branching models

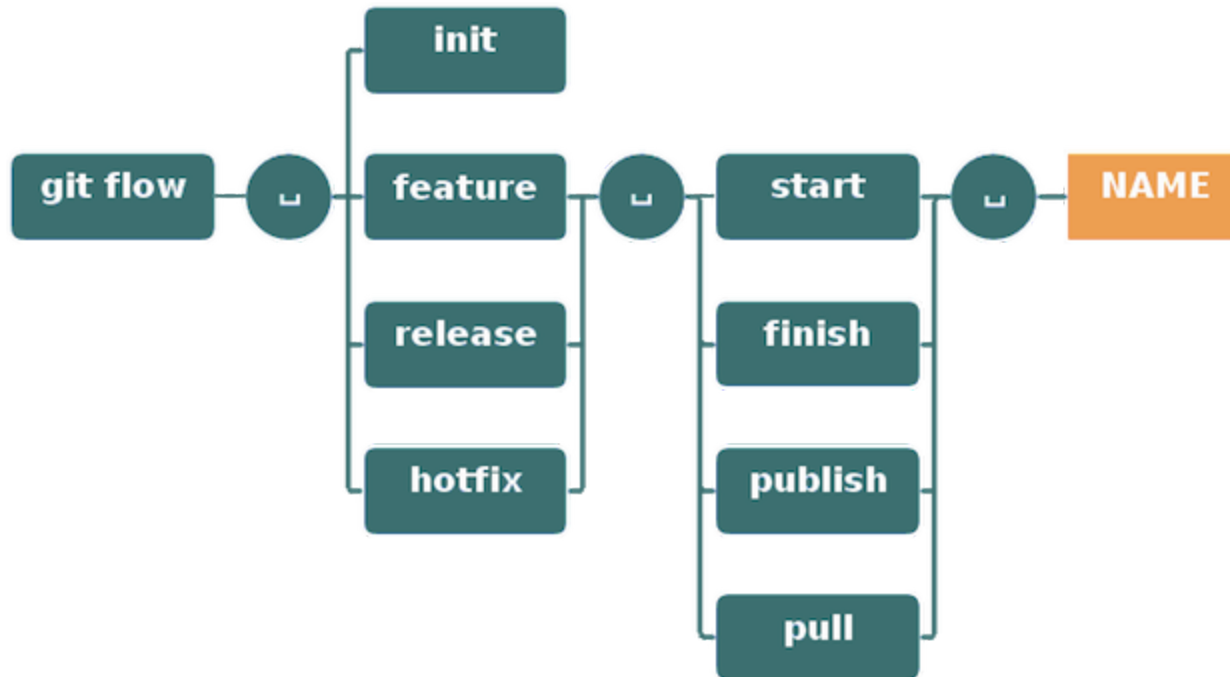
- git flow
 - (hotfix)- `master` -(release)- `develop` - feature
 - pros: 가장 많이 적용, 각 단계가 명확히 구분
 - cons: 복잡..
- github flow
 - `master` - feature
 - pros: 브랜치 모델 단순화, `master` 의 모든 커밋은 deployable
 - cons: CI 의존성 높음. 누구 하나라도 실수했다간..(pull request로 방지)
- gitlab flow
 - `production` - `pre-production` - `master` - feature
 - pros: deploy, issue에 대한 대응이 가능하도록 보완
 - cons: git flow와 반대 (`master` -develop, `production` -master)

git flow strategy



use git flow easily!

[Link](#)



Practice(2)

- git flow 전략을 활용하여 어제 작성한 introduce.md를 index.html에 재작성하세요.

Requirements

- develop 브랜치에서 다음 릴리즈를 위한 개발이 끝나야 합니다.
- head, body 등 section별 작업은 각각의 브랜치에서 작업되어야 합니다.
- css, js 작업 또한 각 브랜치를 소유합니다.(선택)
- [Semantic Web Elements](#)를 적극 활용하세요.

Revert Everything!

Rename

- Worst

```
$ mv server.py main.py -> deleted, new file
```

- Best

```
$ git mv server.py main.py -> renamed
```

파일의 history를 남기기 위해서는 삭제 후 생성이 아닌 이름바꾸기로 추적

Undoing

```
$ git checkout -- . or $ git checkout -- {filename}
```

Unstaging

```
$ git reset HEAD {filename}
```

Unstaging and Remove

```
$ git rm -f {filename}
```

Edit latest commit

```
$ git commit --amend
```

Edit prior commit

```
$ git rebase -i <commit>
```

abort rebase

```
$ git rebase --abort
```

Complete rebase

```
$ git rebase --continue
```

Reset Commit

Worst case: Reset

ex) 직전 3개의 commit을 삭제한 후, remote에 강제 push

```
$ git reset --hard HEAD~3  
$ git push -f origin <branch>
```

- 협업 시 다른 cloned repo에 존재하던 commit log로 인해 파일이 살아나거나, 과거 이력이 깔끔히 사라져 commit log tracking이 힘들어짐.
- solution: 잘못된 이력도 commit으로 박제하고 수정한 이력을 남기자!

Best case: Revert

ex) 현재 HEAD에서 직전의 3개의 commit을 순서대로 거슬러 올라가 해당 내역에 대해 commit, push 수행

```
$ git revert --no-commit HEAD~3
```

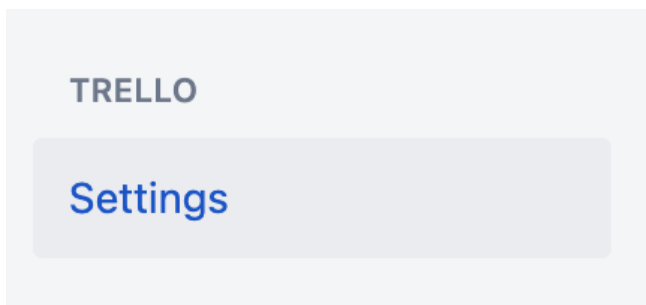
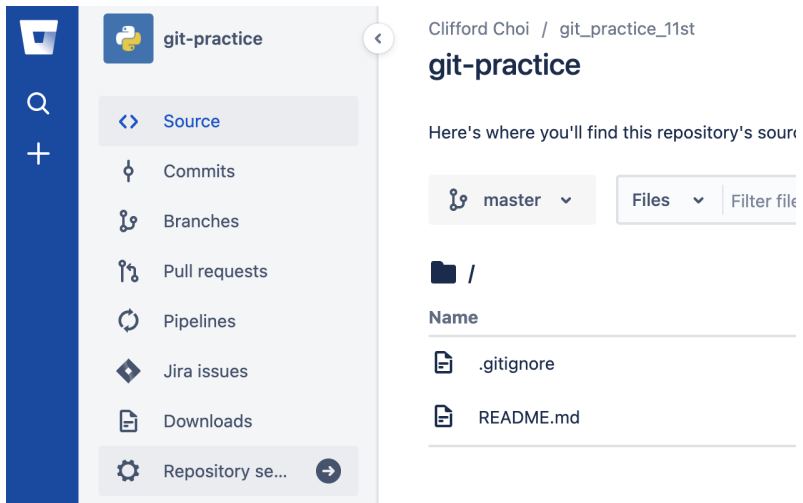
```
$ git commit
```

```
$ git push origin <branch>
```


- 잘못하기 전 과거로 돌아가 최신을 유지하면서 되돌렸다는 이력을 commit으로 남겨 모든 팀원이 이 사항을 공유하고 주지시킬 수 있음.

Integrate Trello boards in Bitbucket

Repository Setting - TRELLO Settings



Enable Trello

 **Jira Software**

Get more out of Bitbucket with Jira Software [Learn more](#)

×

Trello settings



Work more collaboratively and get more done with Trello in Bitbucket

Enable Trello

☐

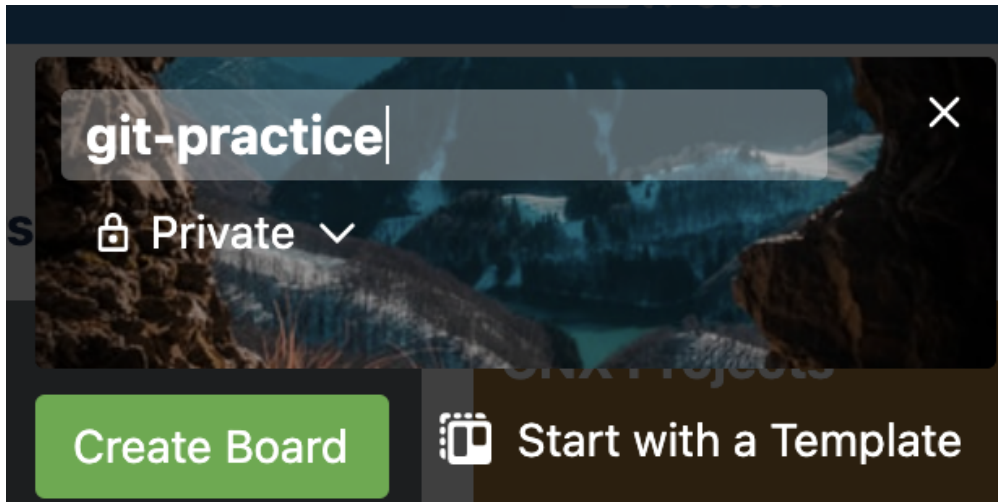
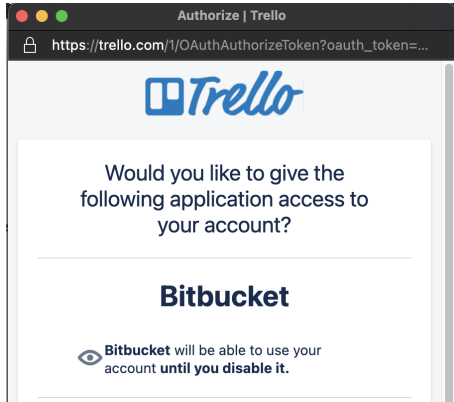
Work more collaboratively and get more done with Trello in Bitbucket

Enable Trello

☒

Connect

Sign in Atlassian Cloud, Create board



Link board

Trello settings



Work more collaboratively and get more done with Trello in Bitbucket

Enable Trello

Connected to [Trello](#) as [Clifford Choi](#) · [Disconnect](#)

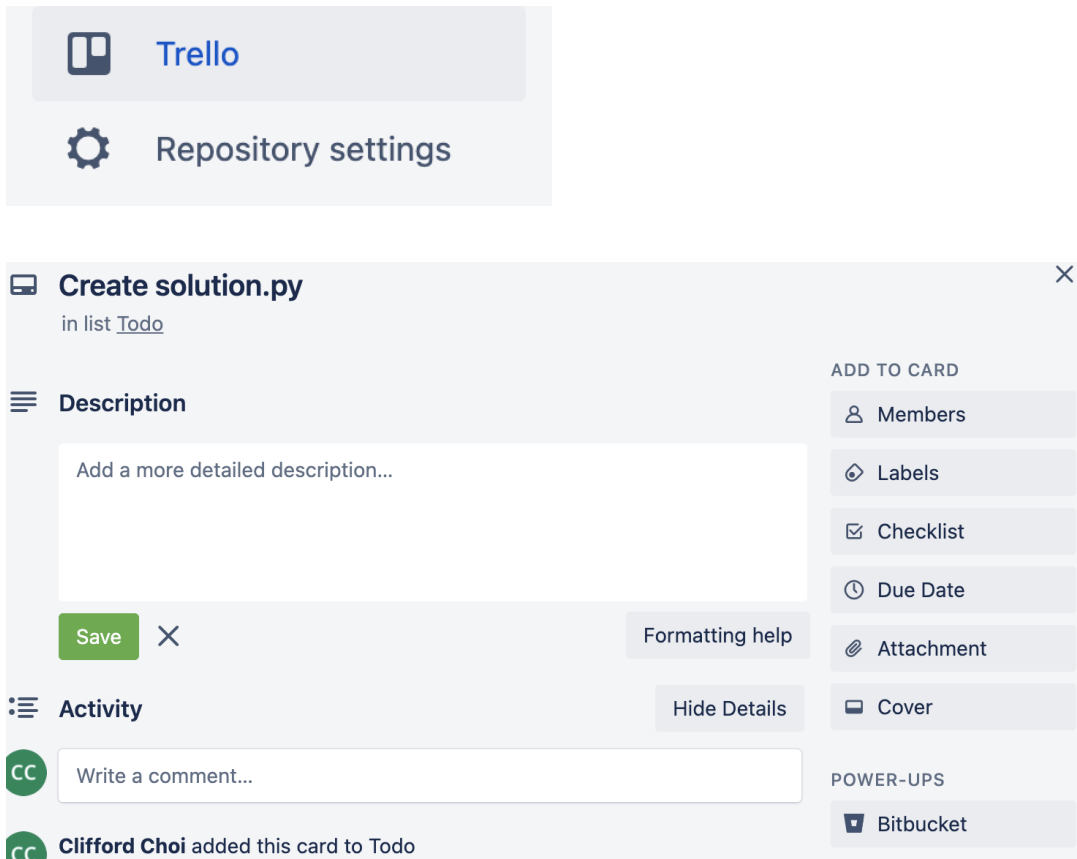
Link board

Linked boards

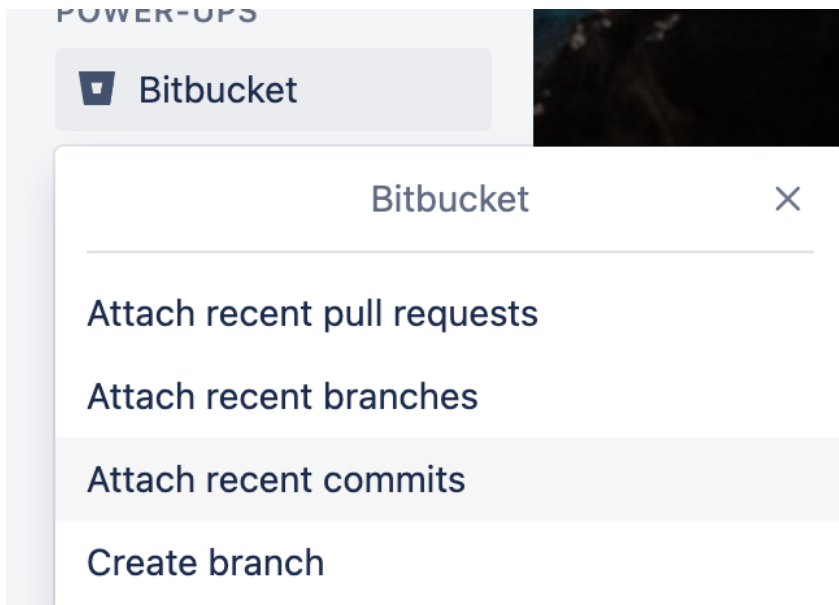
Name	Linked by	Self-invoke permissions		
git-practice	Clifford Choi	WRITE		Unlink DEFAULT BOARD

**Self-invoke permissions correspond to the repo access permissions, and let you control which Bitbucket users can add themselves to a linked board.*

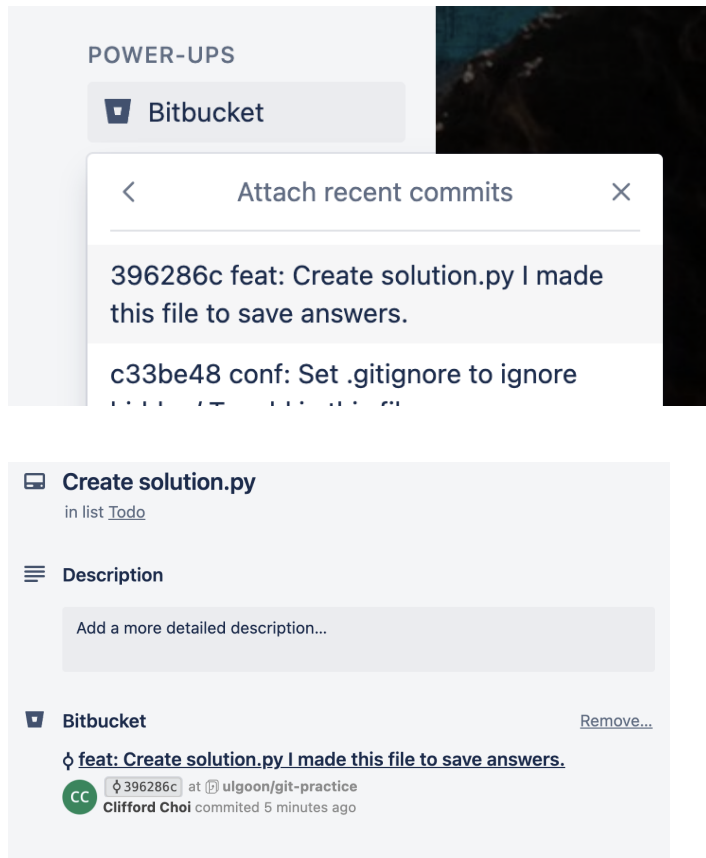
Open Trello, Select Bitbucket



Attatch recent commits



Attatch complete



Collaborate with your teammates

Forking workflow

PM: Create a new repository

Create a new repository

[Import repository](#)

Workspace

kingwang zzang

Project name *

collabo

Repository name *

fork-tutorial

Access level

☐ Private repository

Uncheck to make this repository public. Public repositories typically contain open-source code and can be viewed by anyone.

Include a README?

Yes, with a tutorial (for begin...)

Include .gitignore?

Yes (recommended)

Advanced settings

Description

Language

Python

Create repository

Cancel

PM: Clone, create and push develop branch

```
vagrant@ubuntu-bionic:~/Documents/dev/fork-tutorial$ git flow init

Which branch should be used for bringing forth production releases?
- master
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? []
Hooks and filters directory? [/home/vagrant/Documents/dev/fork-tutorial/.git/hooks]
vagrant@ubuntu-bionic:~/Documents/dev/fork-tutorial$ git branch
* develop
master
```

PM: Check develop branch, Add dev1

kingwang zzang / collabo

fork-tutorial

Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, [add a description to y](#)

develop

Files

Filter files

/

Name	Size	Last commit	Message
.gitignore	624 B	5 minutes ago	Initial commit
README.md	2.56 KB	5 minutes ago	Initial commit
index.html	63 B	56 seconds ago	feat: html, head, body update on index.html

Users

Add a user by their name or email address

Read

Add

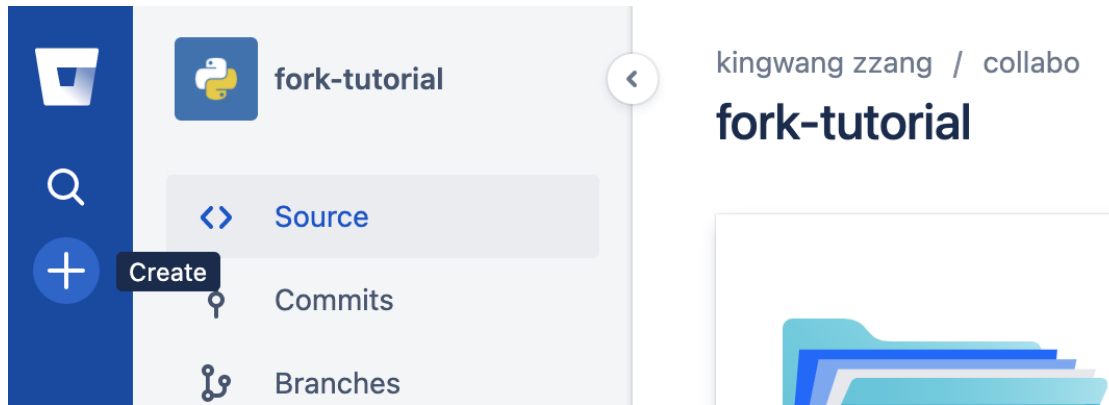
ulgoon89@gmail.com (Resend invitation)

READ

WRITE

ADMIN

dev1: Click +(Create), fork



GET TO WORK

 **Clone** this repository

 **Compare** branches or tags

 **Fork** this repository

dev1: Fork complete

Fork kingwangzzang1234 / fork-tutorial

Workspace

Clifford Choi

Project *

git_practice_11st

Name *

fork-tutorial

Access level

☐ Private repository


> Advanced settings

Fork repository

Cancel

Clifford Choi / git_practice_11st

fork-tutorial



Take the next steps for this repository and its freshly added files

Copy and connect the repository locally so that you can push updates you n repository URL on the command line:

```
git clone https://ulgoon@bitbucket.org/ulgoon/fork-tutorial.git
```

You can also clone in [Sourcetree](#) or [VS Code](#) to avoid the command line. [Le](#)

Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, [ad](#)

dev1: Clone, git flow init

```
fork-tutorial on ✎ master  
→ git branch  
★ master
```

```
fork-tutorial on ✎ master  
→ git flow init  
  
Which branch should be used for bringing forth production releases?  
- master  
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? []  
Hooks and filters directory? [/Users/ulgoon/Documents/dev/fastcampus/fork-tutorial/.git/hooks]  
  
fork-tutorial on ✎ develop took 4s  
→ ls  
README.md  index.html
```

dev1: Create feature branch, Do some work, finish feature branch

```
fork-tutorial on ↳ develop
→ git flow feature start head-init
Switched to a new branch 'feature/head-init'

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

Now, start committing on your feature. When done, use:

    git flow feature finish head-init
```

```
fork-tutorial on ↳ feature/head-init
→ git flow feature finish head-init
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.
Updating 27072c0..7623cff
Fast-forward
 index.html | 13 ++++++++--
 1 file changed, 11 insertions(+), 2 deletions(-)
Deleted branch feature/head-init (was 7623cff).

Summary of actions:
- The feature branch 'feature/head-init' was merged into 'develop'
- Feature branch 'feature/head-init' has been locally deleted
- You are now on branch 'develop'
```


dev1: Push to dev1:develop, Create a pull request

```
fork-tutorial on ȳ develop [ȳ]
→ git push origin develop
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 481 bytes | 240.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote:
remote: Create pull request for develop:
remote:   https://bitbucket.org/ulgoon/fork-tutorial/pull-requests/new?source=develop&t=1
remote:
To https://bitbucket.org/ulgoon/fork-tutorial.git
   27072c0..7623cff  develop -> develop
```

Source

Commits

Branches

Pull requests

Pipelines

Jira issues

Downloads

Repository settings

Create a pull request

ulgoon / fork-tutorial
Created 5 minutes ago, updated 24 seconds ago
develop

kingwangzzang1234/fork-tutorial
develop

Title* feat: Add meta tags

Description

Aa B I ...

- set viewport
- set charset to utf-8
- create title tag
- update body tags

Feedback ?

dev1: Done!

kingwang zzang / collabo / fork-tutorial

Pull requests

Search pull requests



Open



Author



Target branch

Summary



feat: Add meta tags

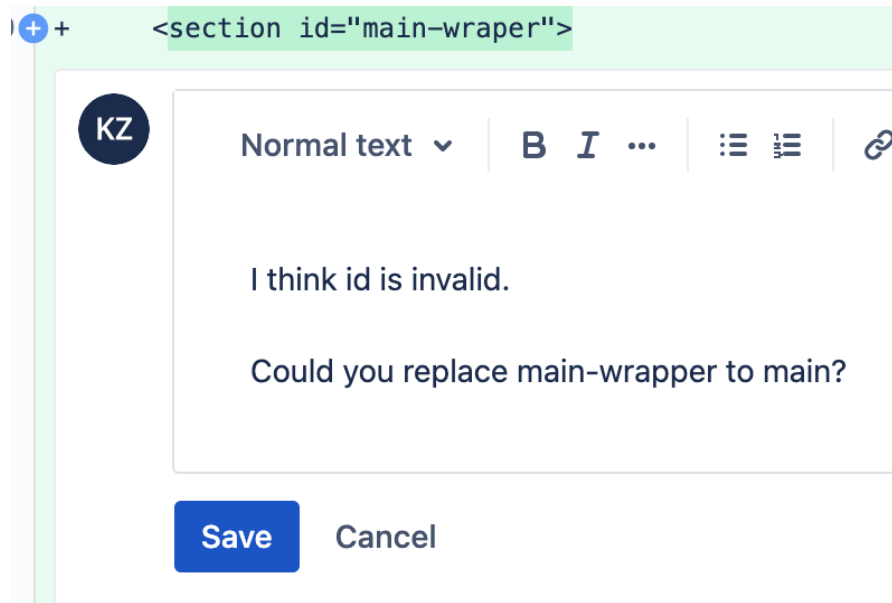
ulgoon/fork-tutorial:develop



develop

Clifford Choi - #1, created 20 seconds ago, updated 20 seconds ago

PM: Review dev1's code



dev1: Push another commit to dev1:develop while pull request is open

2 commits



Clifford Choi

8f471c1

fix: replace section id main-wr... 1 minute ago

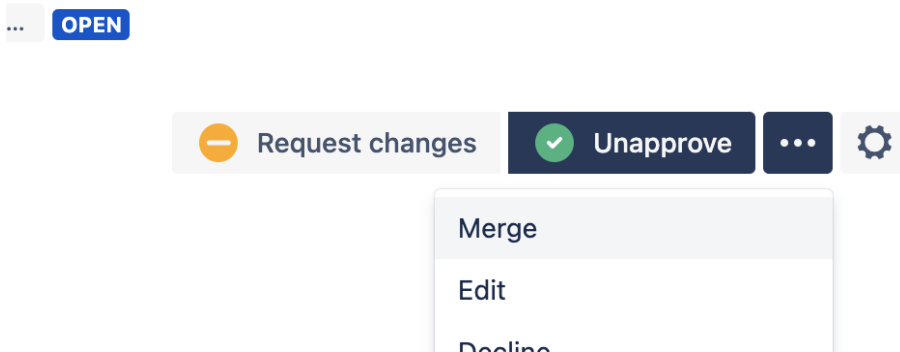
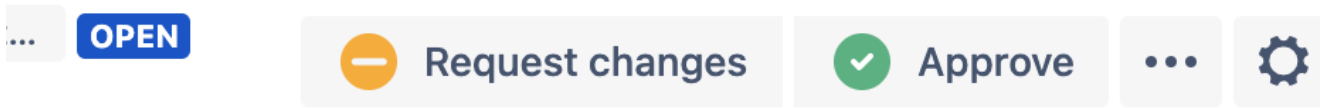


Clifford Choi

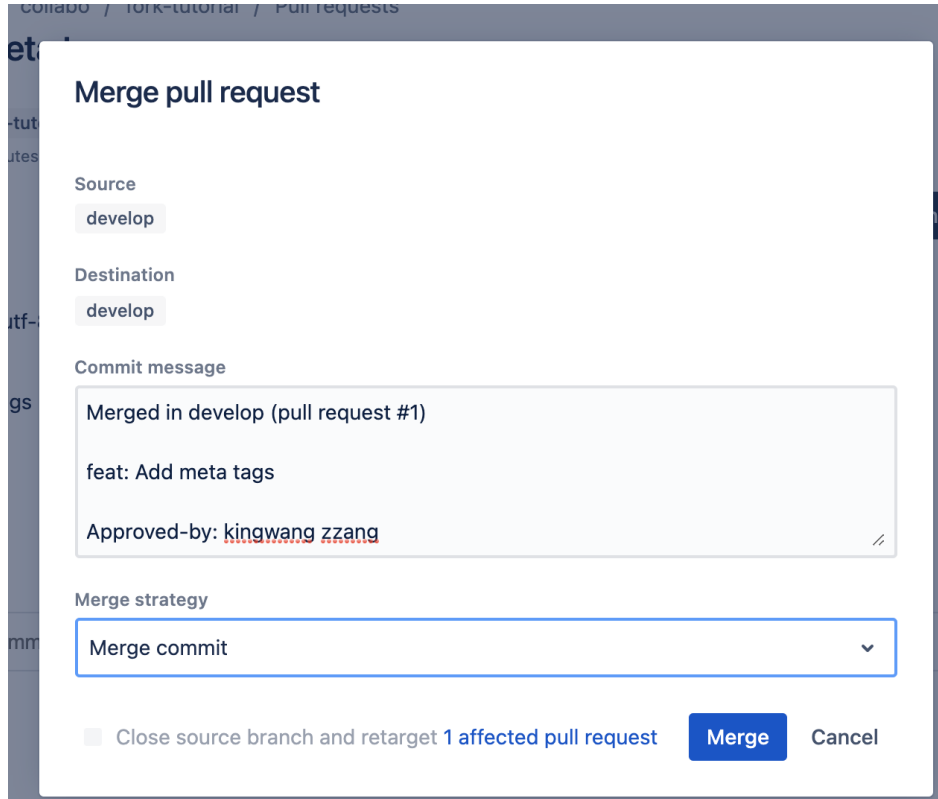
7623cff

feat: Add meta tags set viewp... 6 minutes ago

PM: Approve pull request, Merge



PM: Merge!!



The image shows a 'Merge pull request' dialog box from GitHub. The 'Source' and 'Destination' branches are both set to 'develop'. The 'Commit message' field contains the text: 'Merged in develop (pull request #1)', 'feat: Add meta tags', and 'Approved-by: kingwang zzang'. The 'Merge strategy' is set to 'Merge commit'. At the bottom, there is a checkbox for 'Close source branch and retarget 1 affected pull request', a blue 'Merge' button, and a 'Cancel' button.

colabo / fork-tutorial / Pull requests

Merge pull request

Source
develop

Destination
develop

Commit message

Merged in develop (pull request #1)

feat: Add meta tags

Approved-by: kingwang zzang

Merge strategy
Merge commit

☐ Close source branch and retarget 1 affected pull request **Merge** Cancel

dev2,dev3,devn, .. : Update develop branch

```
$ git remote add pmorigin {PM's repo addr}
```

```
$ git pull pmorigin develop
```

Practice(3)

- 수업 중 제공되는 저의 repository 주소에 접근하여 fork 한 뒤, 하고 싶은 말을 [Chat.md](#) 에 pull request 하세요.

Practice(4)

- 3인이 팀이 되어 프로젝트 수행
- 아래의 과제 중 하나를 수행할 것
 - i. [피보나치킨](#) 클론(치킨과 인원 수에 따라 적절한 맥주의 용량도 출력)
 - ii. [socket.io](#)를 이용한 웹소켓 단체 채팅 서비스(authentication 필수)
 - iii. [What's my value?](#) 게임(11번가 베스트 중 [가전/디지털](#) 스크래핑 후 진행)
- Requirements
 - 타겟 플랫폼, 언어나 Framework는 팀 내 협의 후 결정
 - 서비스 기획 -> backlog 작성(trello) -> 개발 -> 평가 순으로 진행

Wrap it up!

- branch를 활용한 개발은 코드의 독립성을 보장한다.
- branching strategy 중 git flow 전략을 잘 활용하면 깔끔한 코드관리와 CI/CD에서도 도움이 될 수 있다.
- 모든 잘못된 작업은 push 하기 전까지 아무도 모르게 되돌릴 수 있다.
- Atlassian의 도구들을 잘 활용하면, 현재 프로젝트의 진행상황을 깔끔하게 관리할 수 있다.
- git flow 전략과 forking workflow를 활용하면 비동기적 분산형 저장소의 장점을 잘 살릴 수 있다.