

Hello World



GitHub

version 2018.35

Intro

이 문서는 **GitHub Guides**에서 제공하는 [Hello World](#)을 번역 및 정리한 것입니다. Hello World는 깃허브(**GitHub**)를 처음 사용하는 입문자를 위한 튜토리얼입니다.

Hello World 프로젝트는 컴퓨터 프로그래밍에서 오래된 전통입니다. 새로운 것을 시작할 때에 간단한 실습이 될 수 있지요. 이제 깃허브를 알아보시다!

학습할 항목

- 리포지터리(repository)를 생성하고 사용합니다.
- 새로운 브랜치(branch)를 시작하고 관리합니다.
- 파일을 수정하고 커밋(commit)을 이용하여 깃허브에 푸쉬(push)합니다.
- 풀 요청(pull request)을 생성하고 머지(merge)합니다.

깃허브란?

깃허브는 버전 관리와 협업을 위한 코드 호스팅 플랫폼입니다. 깃허브는 여러 사람들이 언제 어디서나 함께 일할 수 있는 환경을 제공합니다.

호스팅(hosting): 서버 컴퓨터의 전체 또는 일정 공간을 이용할 수 있도록 임대해 주는 서비스

우리는 리포지터리, 브랜치, 커밋, 풀 요청과 같은 깃허브의 필수 요소들을 튜토리얼을 통해 학습할 수 있습니다. Hello World 리포지터리를 생성하고 풀 요청의 동작 흐름을 배워보겠습니다.

준비물

인터넷이 연결된 상태에서 [깃허브 계정](#)만 있으시면됩니다. 코드를 어떻게 작성하는지, 커맨드 명령어를 어떻게 사용하는지 심지어 깃(Git)을 어떻게 설치하는지조차 알 필요가 없습니다.

Step 1. 리포지터리 생성하기

일반적으로 하나의 리포지터리는 하나의 프로젝트를 관리합니다. 리포지터리는 디렉토리, 파일, 이미지, 동영상, 스프레드시트, 데이터 셋 등을 포함할 수 있습니다. 깃허브에서는 **README** 파일 또는 프로젝트에 대한 설명이 기술된 파일을 리포지터리에 포함할 것을 권고하고 있습니다. 또한 깃허브에서는 새로운 프로젝트를 생성하는 동시에 README 파일을 만들 수 있는 기능을 제공합니다. 뿐만 아니라 라이선스 파일과 같은 파일도 가능합니다.

깃허브의 리포지터리는 아이디어, 자원을 저장하거나 다른 사람들과 공유하고 토론하는 장소가 될 수 있습니다.

새로운 리포지터리 생성

1. 브라우저 상단 오른쪽에 위치한 **+** 버튼을 클릭합니다. 그리고 **New repository**를 선택합니다.
2. 리포지터리 이름을 `hello-world` 로 기재합니다.
3. 리포지터리를 설명할 수 있는 내용을 Description란에 기재합니다.
4. **Initialize this repository with a README**를 선택합니다.
5. **Create repository**를 클릭합니다.

Owner

CheolhoJeon ▾

/


Repository name

hello-world ✓


Great repository names are short and memorable. Need inspiration? How about **silver-bassoon**.

Description (optional)

GitHub 입문을 위한 Hello World 리포지터리

☒  **Public**

Anyone can see this repository. You choose who can commit.


☐  **Private**

You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾ 

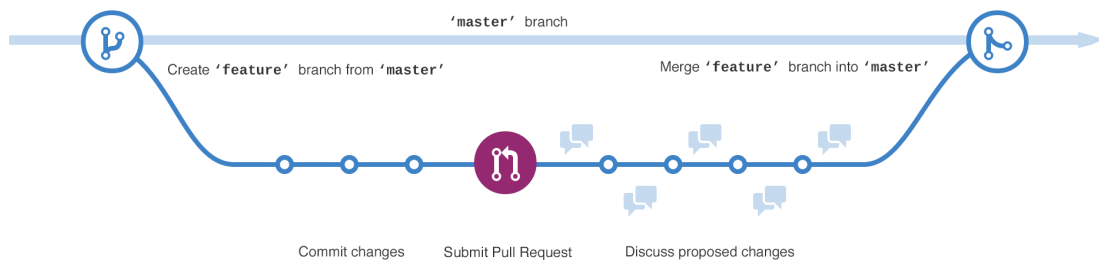
Create repository

Step 2. 브랜치 생성하기

브랜치(branch)는 동시에 여러 버전에서 작업할 수 있도록 해줍니다.

기본적으로 리포지터리는 하나의 `master` 브랜치를 가집니다. 이 브랜치는 최종 브랜치라고 생각하시면 됩니다. 다른 브랜치들은 `master` 브랜치에 커밋하기 전에 수정 하기 위해 사용됩니다.

`master` 브랜치에서 새로운 브랜치를 생성하게 되면, 새로운 브랜치는 그 시점에서 `master` 브랜치의 복사본이 됩니다. 만약 우리가 생성한 브랜치에서 작업을 하는동안 다른 누군가가 `master` 브랜치를 수정 했다면, 우리는 수정사항을 적용하기 위해 풀(pull) 해야할 것입니다.

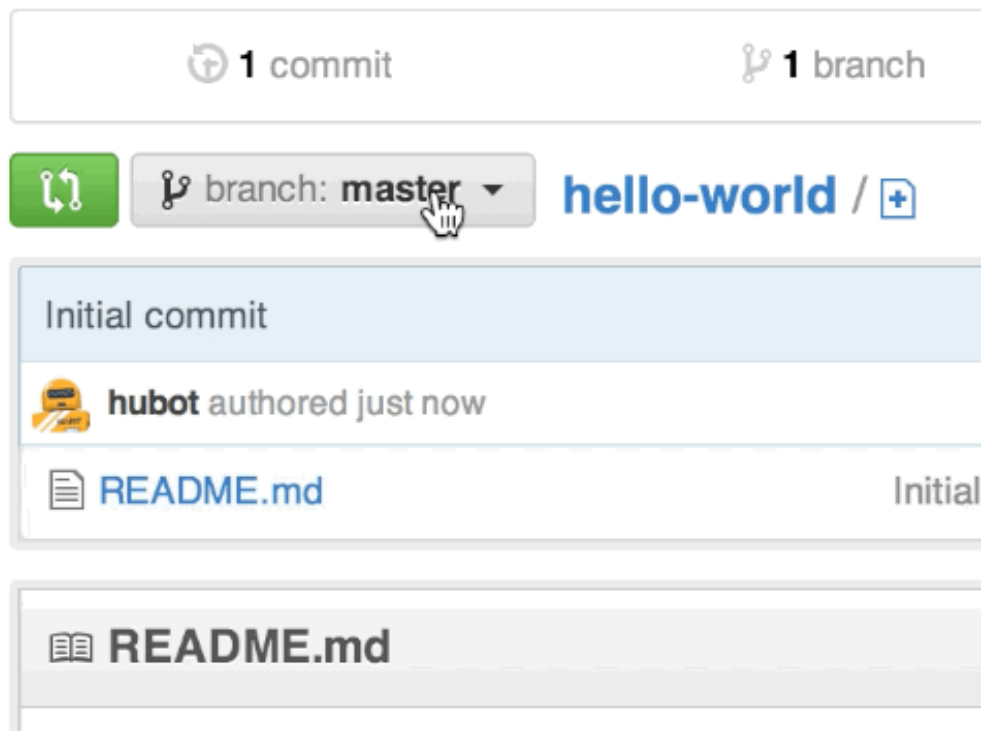


깃허브의 개발자, 작가, 디자이너들은 버그 수정 또는 각각의 작업을 위해 `master` 브랜치와 분리된 브랜치를 사용합니다. 이들은 수정작업이 모두 완료되면, `master` 브랜치에 머지합니다.

새로운 브랜치 생성

1. `hello-world` 리포지터리로 이동합니다.
2. 파일 리스트 위에 보이는 **branch: master**를 클릭합니다.
3. `readme-edits` 라는 새로운 브랜치 이름을 기입란에 기입합니다.
4. **Create branch**를 선택하거나 "Enter"를 칩니다.

Just another repository — Edit



이제 우리는 `master` 브랜치, `readme-edits` 브랜치를 가졌습니다. 이 브랜치들은 똑같아 보이지만 앞으로는 아닐 것입니다. 다음으로 새로운 브랜치에서 수정 작업을 해보겠습니다.

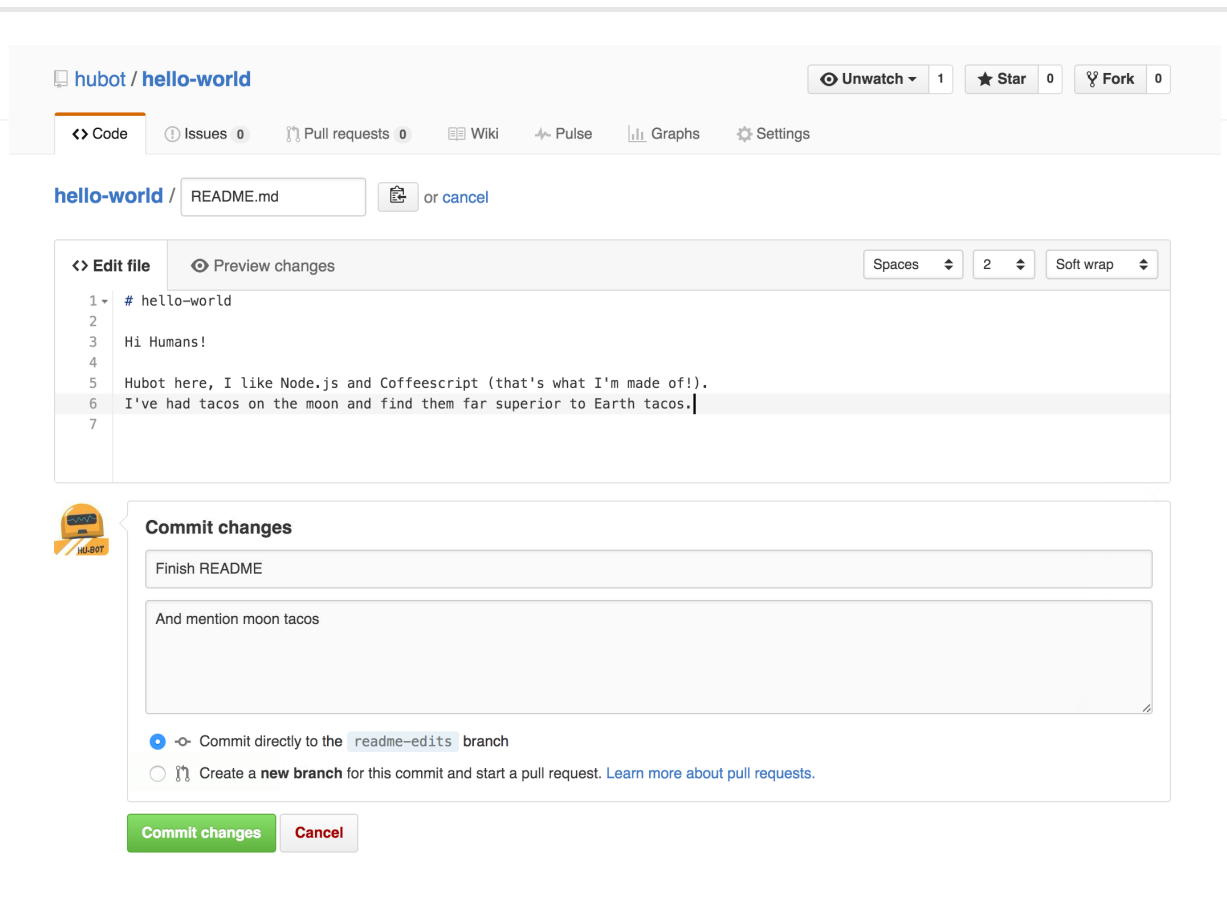
Step 3. 변경 사항을 만들고 커밋하기

`master` 브랜치의 복사본인 `readme-edits` 브랜치의 코드를 확인할 수 있습니다. 이제 몇가지 사항을 수정해 보겠습니다.

깃허브에서는 저장된 변경 사항을 **커밋(commit)**이라고 부릅니다. 각각의 커밋은 변경 사항이 만들어진 이유에 대해 설명해주는 메시지를 가집니다. **커밋 메시지(commit message)**는 변경 사항에 대한 기록이며, 다른 프로젝트 참여자가 당신이 왜 수정을 하였는지 이해하는데에 도움을 줍니다.

변경 사항 생성 및 커밋

1. `README.md` 파일을 클릭합니다.
2. 파일을 수정하기 위해 뷰어 위에 있는 연필모양 아이콘을 클릭합니다.
3. 에디터에서 내용을 수정하거나 추가합니다.
4. 변경 사항에 대해 설명해주는 커밋 메시지를 작성합니다.
5. **Commit changes** 버튼을 클릭합니다.



위의 동작을 모두 마치면 `readme-edits` 브랜치의 REAME 파일만 수정됩니다. 때문에 `readme-edits` 브랜치는 이제 `master` 브랜치와 다른 내용을 가지게 됩니다.

Step 4. 풀 요청하기

풀 요청은 협업을 하기 위한 중요 기능 중 하나입니다. 풀 요청을 했을 때, 우리의 변경사항을 제안할 수 있고 다른 개발자들에게 리뷰를 부탁할 수도 있습니다. 마지막으로 해당 브랜치에서 풀 하거나 머지할 수 있습니다. 풀 요청은 두 브랜치 사이의 차이점을 보여줍니다. 수정, 추가, 삭제된 사항들은 초록색이나 빨간색으로 보여지게 됩니다.

커밋을 만들면 풀 요청을 할 수 있게됩니다. 이 후에 우리는 다른 협업자들과 함께 토의할 수 있을 것입니다.

풀 요청 메시지에서 깃허브의 [멘션 시스템](#)을 사용하여 특정 사람 또는 팀원들에게 피드백을 요청할 수 있습니다.

※ 깃허브의 풀 요청(pull request)은 깃랩의 머지 요청(merge request)과 동일합니다.

[Reference](#)

풀 요청 생성

Step	Screenshot
------	------------

Pull Request 탭을
클릭한 후에, **New
pull request** 버튼
을 클릭합니다.

**Example
Comparisons** 박스
에서 `readme-
edits` 를 클릭합니
다.

변경 사항을 살펴보고
제출하려는 내용이 맞
는지 확인합니다.

제출하려는 변경 사항
이 만족스럽다면
**Create Pull
Request** 버튼을 클
릭합니다.

orid

Unwatch 1

Star 0

Fork 0

0

Pull requests 0

Wiki

Pulse

Graphs

Settings

hello-world / README.md

Find fileCopy path

DME

710d852 3 minutes ago

EXAMPLE COMPARISONS

readme-edits

4 minutes ago

1 commit

1 file changed

Commits on Oct 27, 2014

hubot

Finish README ...

Showing 1 changed file with 1 addition and 1 deletion.

2 README.md

...	...	@@ -1,4 +1,4 @@
1	1	hello-world
2	2	=====
3	3	
4		-Just another repository
	4	+Hubot here, I like Node.js and Coffee them far superior to Earth tacos.

base: master


compare: readme-ed

Create pull request

Discuss and review the

풀 요청 제목을 쓰고
변경 사항에 대한 간략
한 설명을 작성합니다.

base: master ▾ ... compare: readme-edits ▾



Readme edits

Write Preview


Content for non-telepathic human.

작성이 끝나면 **Create pull request** 버튼을 클릭합니다.


Step 5. 풀 요청 머지하기


마지막 단계인 풀 요청 머지는 `readme-edits` 브랜치를 `master` 브랜치로 병합함으로써 변경 사항을 적용시킵니다.

1. 변경 사항을 `master` 로 병합하기 위해 **Merge pull request** 버튼을 클릭합니다.
2. **Confirm merge** 버튼을 클릭합니다.
3. 변경 사항이 적용되었으므로 `readme-edits` 브랜치를 **Delete branch** 버튼을 클릭함으로써 함께 삭제합니다.




✓ **This branch has no conflicts with the base branch**
Merging can be performed automatically.

 **Merge pull request** You can also [open this in GitHub Desktop](#) or view [command line instructions](#).



Pull request successfully merged and closed
You're all set—the `readme-edits` branch can be safely deleted.

 **Delete branch**

마무리

연구실 내에서 GitLab을 활용한 경험을 가지고 작성하였습니다. GitHub는 처음 접하기 때문에 간단한 튜토리얼을 학습해보았습니다. 문서를 작성하면서 풀 요청에 대한 의문이 생겼지만 구글링을 통해 깃랩의 머지 요청과 같다는 것 또한 학습할 수 있었습니다.