GitHub 소개

고세윤

서울대학교 통계학과

2016년 4월 18일

Section 1

GitHub 소개

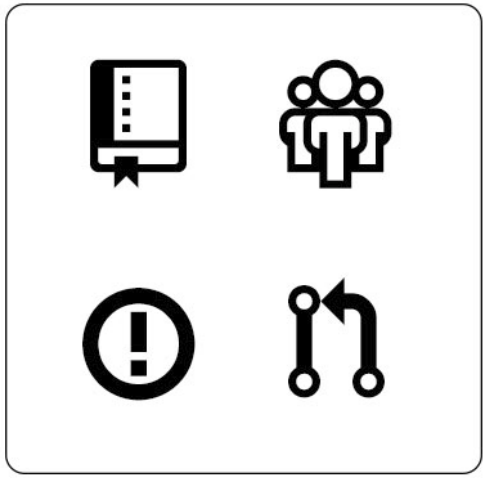
GitHub이란?

분산 버전 관리 시스템(DVCS) "Git" 위에서 만들어진 협업 플랫폼

GitHub이란? (2)

- Git 프로젝트를 호스팅하는 기능 뿐만 아니라, 협업을 위한 여러 기능을 갖추고 있음
  - Issues
    - 기능과 버그에 관한 논의
  - Pull Requests
    - 프로젝트에 추가/변경하고자 하는 기능에 대한 논의와 리뷰

GitHub



GitHub이란? (3)

"프로그래머의 페이스북 같은 존재"

- 많은 오픈 소스 패키지들이 GitHub을 사용하고 있음
  - 프로그래밍 언어: Scala, Julia
    - 2016년 1월 1일: Python이 GitHub로 이주하기로 결정
  - 과학계산: NumPy, SciPy
  - 기계학습: scikit-learn, Apache Mahout
  - 딥러닝: Theano, TensorFlow, Caffe, Keras, Neon
  - 수업: 많은 coursera, edx 등 오픈 코스 강의들
    - 2015년 2학기부터: 전산통계 및 실험, 통계계산에서 사용 중

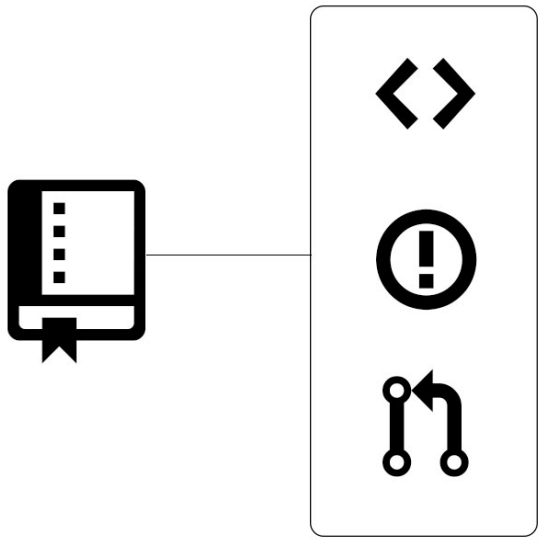
GitHub의 주요 기능

- 이슈
- 풀 리퀘스트
- 커밋 역사 확인
- 그래프
- 변경 내역 확인
- 이메일 알림
- 멘션
- 간단한 웹사이트 호스팅
  - `snu-stat.github.io` (2015년 전산통계 및 실험)

저장소 둘러보기

https://github.com/snu-stat/tutorial-github

- 저장소(repository)
  - 프로젝트가 담겨 있는 디렉토리 또는 저장 공간. 줄여서 repo라고도 한다.
  - 저장소는 개별 컴퓨터의 폴더 안에 있을 수도 있고 GitHub나 다른 온라인 호스트의 공간에 있을 수도 있다.
  - 숨김 파일의 형태로 Git 정보가 저장되어 있다.
  - 코드 파일, 텍스트 파일, 이미지 파일 등등을 저장할 수 있다.



저장소 둘러보기 (2)

https://github.com/snu-stat/tutorial-github

- Code
  - repo에 있는 파일들을 확인할 수 있다.
  - 프로젝트, 문서, 다른 중요한 파일 등등
  - 프로젝트의 root라고도 한다.
  - 변경 사항은 Git의 버전 관리에 의해 추적된다.
- README.md
  - 각 저장소에 기본적으로 포함하는 것을 권장함.
  - GitHub에서는 이 파일의 내용을 repository 아래에 표시한다.
  - 프로젝트를 설명하고 프로젝트 내의 도움이 되는 정보들을 가리켜야 함.

저장소 둘러보기 (3)

이슈

- `Issues`
  - 버그 추적과 기능 요청에 사용
  - 특정 팀 멤버들에게 할당될 수 있으며 활발한 논의와 협업을 위해 만들어짐

저장소 둘러보기 (4)

풀 리퀘스트

- `Pull Request`
  - 작성자가 저장소에 주고자 하는 파일 추가, 변경, 삭제 등의 변화를 나타냄
  - Issue를 해결하기 위해 사용

Section 2

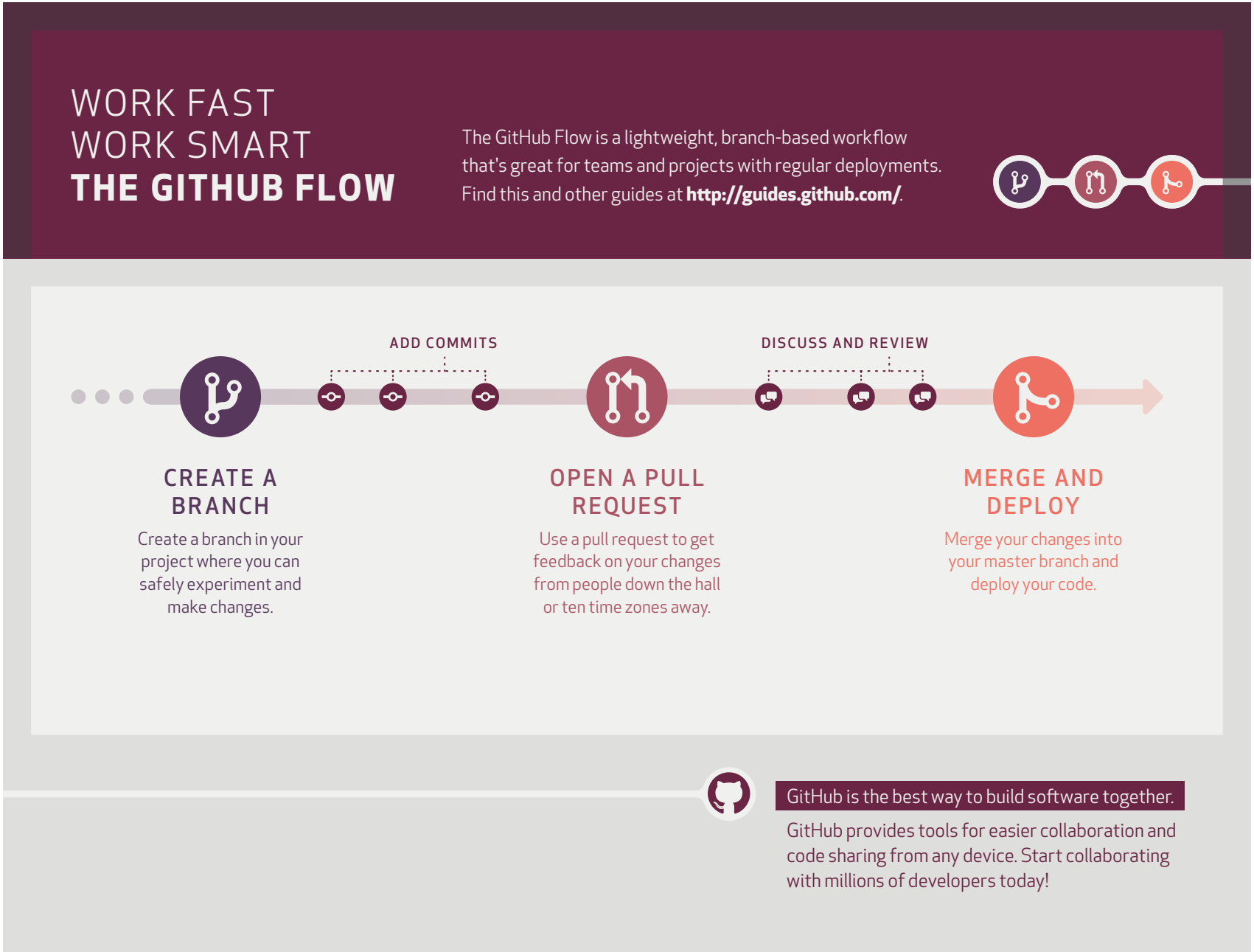
GitHub 작업의 흐름

GitHub 작업의 흐름

새로운 아이디어를 프로젝트를 망칠 걱정 없이 안전하게 시험하기 위한 과정

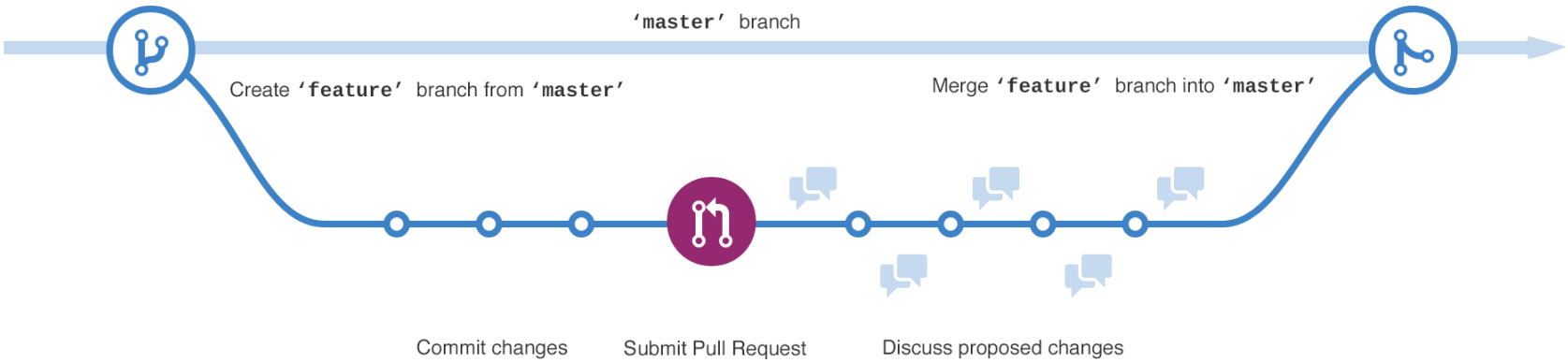
- 브랜치 만들기
- 커밋
- 풀 리퀘스트
- 병합

GitHub 작업의 흐름 (2)



브랜치 만들기

- 브랜치(branch): 병행 개발을 위한 분기. GitHub에 있는 모든 것은 각각의 브랜치에 속하게 된다.
  - `master`: 관례적으로 모든 작업의 기준이 되는 브랜치.
  - feature branch: 기능을 추가하기 위해 `master`에서 분기된 브랜치
- (시연)



파일 만들기

- (시연)
- 커밋(commit)
  - 저장소의 “스냅샷”
  - 프로젝트를 이전의 어떤 상태로든 재평가하거나 복원할 수 있는 체크포인트를 만든다.

풀 리퀘스트 (pull request)

- 프로젝트 파일의 변경을 제안하기 위해 사용
- 이슈를 해결하기 위한 행위
- 풀 리퀘스트는 병합되기 전까지 '진행중인 작업'으로 간주되며, 커밋을 추가할 수 있음
- (시연)

병합 (merge)

- `master` 브랜치에 변경된 내용을 병합할 준비가 끝났을 때
- 병합하고자 하는 기능 브랜치의 내용과 역사를 `master` 브랜치의 내용과 역사에 추가함
- 병합이 완료된 브랜치는 삭제할 수 있음
- 그룹에서 누가 풀 리퀘스트를 병합할 지에 대해서는 내부적으로 약속을 해 두어야...
- (시연)

Section 3

GitHub Desktop

GitHub Desktop

https://desktop.github.com/

- git 명령어를 커맨드라인에서 사용하지 않고도 손쉽게 로컬 컴퓨터에서 GitHub 작업을 할 수 있게 해 주는 도구
- 단순화된 환경
  - 커밋 되지 않은 파일을 자동으로 스테이징
  - pull과 push을 묶어 `sync`라고 하며, `sync` 버튼으로 remote와의 상호작용을 할 수 있음
  - pull request를 프로그램 내에서 직접 생성 가능

작업 과정

- 설정하기
- 저장소 복제하기
- 브랜치 선택하기
- 로컬 파일 편집하기
- 커밋
- 동기화
- (필요할 경우) 풀 리퀘스트

GitHub Desktop 설정하기

- 아이디
- 비밀번호
- 이름
- 이메일

저장소 복제(clone)하기

- 복제 (clone)
  - 저장소의 복제본을 로컬 컴퓨터에 저장하는 과정
  - 역사를 포함한 그 저장소의 모든 것을 복제함 (DVCS의 이점)

브랜치 선택하기

로컬 파일 편집하기

- 복제된 저장소가 있는 공간에서 파일을 원하는 대로 편집

2단계(two-stage) 커밋

- 스테이징
  - 커밋될 예정인 파일의 내용을 준비
  - 쇼핑몰에서 물건을 구매할 때 먼저 장바구니에 넣는 것과 유사
- 커밋
  - 저장소의 "스냅샷"
  - 프로젝트를 이전의 어떤 상태로든 재평가하거나 복원할 수 있는 체크포인트를 만든다.

변경된 내용 동기화(Synchronize)하기

- 컴퓨터 내에서 변경한 내용을 GitHub 서버에 업데이트

풀 리퀘스트

정리

- GitHub는 협업을 위한 버전 관리 플랫폼
- 작업 과정
  - 브랜치 만들기
  - 편집하기
  - 커밋
  - 풀 리퀘스트
  - 병합
- GitHub Desktop을 통해 로컬 컴퓨터에서 작업한 내용을 GitHub에 올릴 수 있음

Q & A

감사합니다!