## 2일차에서 할 이야기

- 프로젝트 준비 과정 중 개발과 백엔드 팀에 가까운 부분
  - GitHub 세팅
  - 작업 방식에 대한 의사결정
  - 기술적인 의사결정
- 팀장 정도가 할 일

## 이번 파트에서 할 이야기

- GitHub organization
  - 보편적인 practice
- Git/GitHub : 어떤 대안이 있을지

## GitHub setup

- 한명이 맡아서 해주는 것이 좋다
  - 의사결정이 필요할 때 팀원들을 부르기
- 그래도 경험해보는 것이 좋으니 직접 해보고 나중에 삭제하자

# GitHub setup

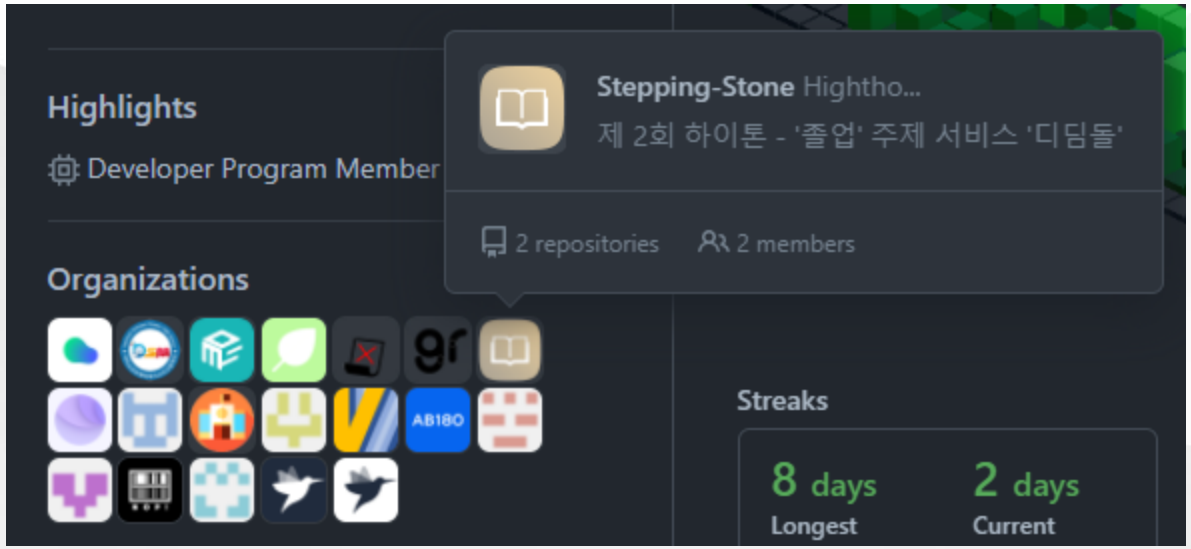
## repository 만들기 〉 어디에 만들까

- personal repository vs organization을 만들기
- 추천: 후자
    i. ownership을 개인에 주는 것보다 organization에 두고 모이는 게 좋음 (Credibility)
    ii. 프로젝트 하나에 딸린 repository가 더 많아질 수 있음
    iii. organization은 팀 관리하라고 준비해둔 기능이 많다
        - team에 사람들을 묶고, repository 권한을 team 단위로 조절
        - 조직 전체에서 진행하는 roadmap을 Project로 등록하고 issue에서 링크
            - → Kanban 보드도 지원

# GitHub setup

## repository 만들기 〉 어디에 만들까

- 개인적으로 : 프로젝트마다 organization을 만들곤 했음



# GitHub setup

## organization 만들기

1. GitHub 우상단 `+`
2. New Organization
3. Create a free organization
4. Set up your organization
    - Organization account name : url로 사용되므로 예쁘게 짓자
    - This organization belongs to : My personal account
5. Add organization members : 지금 초대해도 되고, 나중에 해도 됨
6. Welcome to GitHub : 화면 쭉 내려서 submit하면 다 skip 됨

# GitHub setup

## organization 만들기 〉 free organization?

- organization plan은 총 3가지
  - `Free`, `Team` ($4 per user/month), `Enterprise` ($21 per user/month)
- Enterprise plan은 웬만한 조직들에게 필요 없음
- Team plan의 기능은 쓸만한 것이 많다
  - Protected branches
  - Multiple reviewers in pull requests
  - Required reviewers
  - Scheduled reminders
  - Automatic code review assignment
  - → private 레포에서 지원 가능하다는 것 뿐임 (free plan도 public 레포에서는 지원)

# GitHub setup

## organization 만들기 〉 세팅 〉 1. Owner 설정

- organization에서 사람은 Owner/Member로 구분됨
  - Owner는 뭐든 다 할 수 있음 (Team 설정, 멤버 추방, 레포 삭제 등)
  - Member는 설정을 통해 권한 제어 가능
- organization 만든 사람이 기본적으로 Owner
- 관리자 권한을 가져야 할 사람(e.g. 프론트엔드 팀장)의 role을 Owner로 변경
- 방법
  i. People
     - → **멤버별 톱니바퀴 아이콘 〉 change role**
  ii. 애초에 유저 초대할 때 role을 설정할 수 있음

# GitHub setup

## organization 만들기 〉 세팅 〉 2. Member privileges

- member의 권한 제어
  - repository 단위로 어떤 행동을 허용하고 제한할지
- repository에 clone, pull만 가능하게 하거나, push도 가능하게 하거나 등..
  - 악의적인 팀원의 트롤링 방지
  - 실수 방지

## GitHub setup

### organization 만들기 〉 세팅 〉 2. Member privileges

- **Settings 〉 Member privileges 〉 Base permissions**
  - 권한은 repository마다 추가적으로 설정 가능함
  - base permission : 미설정된 repository에 어떤 권한을 가질지
  - No permission, Read, Write, Admin
  - 자세한 설명은 GitHub 참조

→ 보통 Read로 두고 repository 단위로 별도 권한을 부여

# GitHub setup

## organization 만들기 〉 세팅 〉 2. Member privileges 〉 Admin repository permissions

- Repository visibility change
- Repository deletion and transfer
- Issue deletion
- Repository discussions
- Admin repository permissions는 다 뺏어두는 것이 관리하기 편함
    - base permission이 admin이 아니더라도 repository에서 admin이 될 수 있음

# GitHub setup

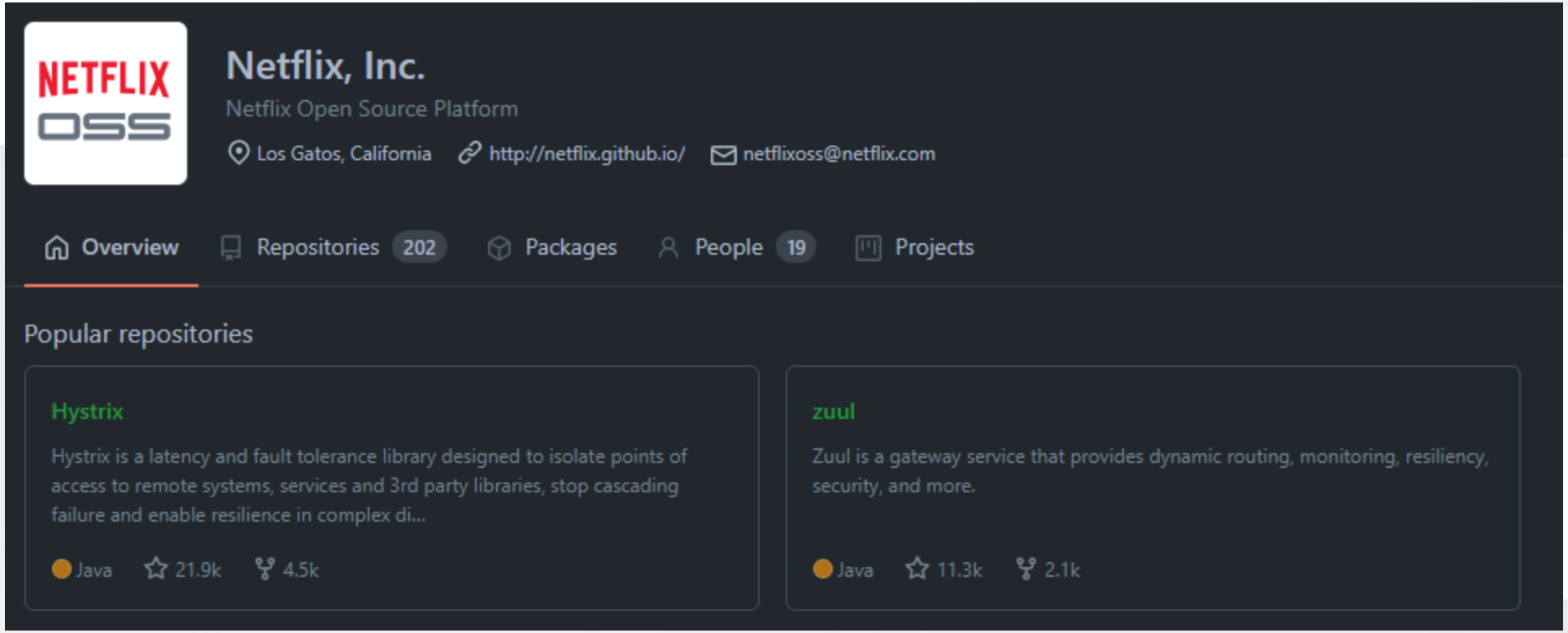
## organization 만들기 〉 세팅 〉 3. Team 설정

- 팀에 사람이 5명, repository는 10개
  - 팀원이 새로 들어온다면?
- **Teams 〉 New Team**
- 백엔드 team 만들고 member를 추가하자
- sub team을 만들어서 세분화하기도 함
  - backend 팀 하위에 api, data platform 팀을 나누는 식
- 사례 : member base permission을 `No permission` 으로 두고 team에 넣어서 권한을 주는 조직

# GitHub setup

## organization 만들기 〉 세팅 〉 4. Profile 설정

- **Settings 〉 Profile**
  - display name, description, 아이콘 등을 설정할 수 있어서 꾸미기에 이용



# GitHub setup

## repository 만들기

1. repository 생성 및 clone
2. spring project init
3. gitignore 세팅
    - 추천 : gitignore.io
        - `macOS` 를 추가하자 : `.DS_Store`
    - tracked file들은 안 지워짐
        - `git rm -r --cached .`
        - 알아볼 것 : git life status lifecycle
4. 모두가 납득하는 boilerplate가 있다면 미리 추가해둬도 좋다
5. 간단한 README.md 작업 (파일만 만드는 수준도 괜찮다)
6. Settings 〉 Manage access 〉 Invite teams or people
    - Repository permission levels for an organization

## 항상 대안을 생각하기

- 요즘은 Git/GitHub을 쓰는 것이 당연하지만
- Git 대신
  - Mercurial, Subversion(SVN) 등
  - CVCS(Centralized Version Control System) vs DVCS(Distributed Version Control System)
  - Git - 버전 관리란?
  - For home projects, can Mercurial or Git (or other DVCS) provide more advantages over Subversion?
- GitHub 대신
  - BitBucket, GitLab 등
    - BitBucket : private repository 지원, JIRA 연동이 탁월하나 서버가 느리고 버그가 많음
    - GitLab : GitHub에게 없던 자체 CI/CI 지원. GitHub Actions가 생기고 점유율 하락

# Insight

## Cloud vendor들이 제공하는 git 호스팅

- AWS CodeCommit
- Google Cloud Source Repositories 등
- upside
  - 소스코드 관리부터 배포까지 private network에서 처리할 수 있다
  - 비용 절감
- downside
  - CodeCommit에 익숙한 개발자가 세상에 있기나 할까..?
  - GitHub에 비해 기능적으로 아직 많이 부족함

## 정리

- 팀 프로젝트는 personal repository보단 organization을 만들고 시작하자
- Owner role을 적절히 주어서 분담
- member base permission은 Member role을 가진 사람에게 부여되는 repository별 기본 권한
  - 기본값은 Read. 기본값 그대로 Read 혹은 No permission으로 설정하고 따로 권한을 부여하는 것을 추천
  - admin repository permissions도 신경써주면 좋다
    - repository를 생성한 사람은 자동으로 해당 repository에 Admin 권한으로 추가되기 때문
- repository에 collaborator 추가는 유저 단위보다는 team을 만들어서 team 단위로 추가하는 것이 좋다
  - sub team을 이용하면 더 디테일하게 관리가 가능하다
- repository 생성 후 setup하면서 gitignore.io 이용하면 좋다
- Git/GitHub도 대안이 있긴 하다