협업 / DevOps

임완섭(email@wanseob.com)

강의자료 링크

https://docs.google.com/presentation/d/e/2PACX-1vQZf-kBASduq5u6Qjba1DTBTVTSPzMfOn-OpCrZJqpu9KqgCzWH3yKDirizRzPl4bAqXkGhQrZb12cu/pub?start=false&loop=false&delayms=3000

실습 진행 계획

1. Git inside

- Git 동작 원리 파악
- Git 고수준 명령어 실습

2. 협업/DevOps

- 협업을 위한 규칙
- 협업과 DevOps 실습

2. 협업 / DevOps

협업 = Convention

1. Code Convention
2. Semantic Versioning
3. Branch strategies
4. Conventional Commits
5. Governance

Code Convention

가독성을 위해 기여자들간 코드 스타일을 통일하는 것이 필요

(go의 경우에는 표준 코드 스타일이 내장되어 있음)

https://github.com/google/styleguide

Semantic Versioning

프로그램의 버전 관리 방법

MAJOR.MINOR.PATCH

Major: Breaking changes
Minor: Feature updates
Patch: Small bug fixes

alpha stage: (동작은 되는 단계)
0.1.0: Feature A updated
0.1.1: Minor bug fixes
0.2.0: Feature B updated.
beta stage: (배포를 준비하는 단계)
1.0.0-rc0: Release candidate 0
1.0.0-rc1: Release candidate 1
released stage: (배포 단계)
1.0.0: V1 release
1.0.1: Hotfix: security bug fix
...
2.0.0: V2 Release(can be incompatible with V1)

Branch strategy: Git flow vs Github flow

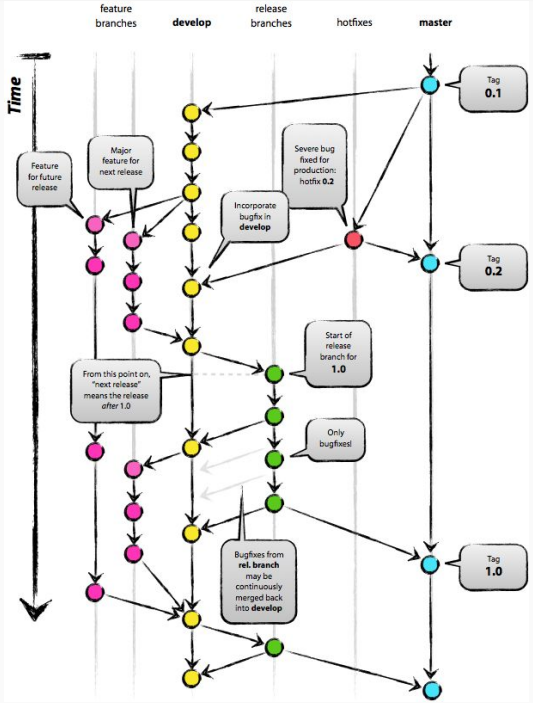
Git-flow:

https://nvie.com/posts/a-successful-git-branching-model/

Github-flow:

https://guides.github.com/introduction/flow/

Branch strategy: Git flow vs Github flow



Conventional Commits

https://www.conventionalcommits.org/

Governance

1. Maintainer : 프로젝트의 관리자
2. Committer : 특정 브랜치에 커밋 권한이 있는 기여자
3. Contributor : Pull request 등을 통해 기여를 진행한 모든 사람

실습과정

1. Python 라이브러리를 협업과 DevOps를 사용해 배포하기
    a. Travis 설정을 통한 자동배포 설정
        i. 테스트
        ii. 도큐멘테이션
        iii. pypi 배포
    b. 메인테이너 선정
    c. Committer 선정
    d. Project manager를 사용한 Issue 등록
    e. 브랜치 관리
    f. 컨트리뷰터의 풀리퀘스트
    g. 코드 리뷰
    h. 풀리퀘스트 머지 & 라이브러리 완성

실습

오늘의 미션:
파이썬 수학 라이브러리 한글화

만약 윈도우즈라면.

- 개발할 때에는 Ubuntu 운영체제 사용하는 것을 권장합니다.
- 윈도우즈 사용시에는 개발자모드를 켜서 Bash활성화하세요.
- 다음 글에 설정법이 자세하게 나와 있습니다. http://sanghaklee.tistory.com/39 글을 참조해주시고 Step3부터 보시면 됩니다.

git과 python 그리고 virtualenv 준비하기

```
sudo apt update

sudo apt install git

sudo apt install python3

sudo apt install python3-pip

pip3 install virtualenv
```

Github 가입 및 organization 접속

https://github.com/git-training-190328

저장소 fork

https://github.com/wanseob/math-kor

저장소 클론

```
cd /mnt/c/Users/…./

git clone https://github.com/{your-id}/math-kor

cd math-kor
```

파이썬 환경 준비

```
cd /path/to/math-kor

python -m virtualenv .venv

source .venv/bin/activate

(.venv) ~@~: pip install -r requirements
```

Test & Dev

```
(.venv) ~@~: pytest
```

협업 1. 버전 계획

1. 버전별로 넣고자 하는 기능을 정리
2. 프로젝트 보드 활용

협업 2. issue 정리하기

버전 계획에 있는 것은

1. help wanted

그렇지 않은 것은

2. new feature

협업 3. pull request 만들기

1. 포크해간 레포지토리를 본인 로컬 컴퓨터에 클론한다
2. 작업 진행
3. 포크 레포지토리에 작업 결과물을 업로드한다
4. 메인 레포지토리(upstream)에게 본인이 업로드한 작업 결과물을 가져가 달라고 요청한다(pull request).

협업 4. 코드 리뷰하기

CODE_OF_CONDUCT.md

오픈소스 활동에 행동규약을 정한다.

https://www.ubuntu.com/community/code-of-conduct

협업 5. 자동 배포하기

1. travis-ci와 같은 CI/CD(Continuous Integration & Continuous Deployement) 도구를 사용한다.
    a. 커밋이 발생했을 경우, 자동으로 테스트 수행
    b. 태그된 커밋이 발생했을 경우에는 테스트 이후 배포 수행
2. 개발 결과에 따라 자동으로 pypi에 배포된 파이썬 패키지를 확인한다.

협업 6. 문서화하기

```
mkdocs gh-deploy
```

Github Org Team

https://github.com/git-training-190328

참고: Repository service provider 제공기능

1. 쉬운 코드 Exploring UI 제공
2. Pull request (Merge request) 기능 제공
3. 코드 리뷰 기능 제공
4. 프로젝트 관리 기능 제공
5. 다양한 응용 프로그램
6. 도큐멘테이션용 무료 정적 웹 호스팅

참고: 서비스별 비교

Github	Gitlab	Bitbucket
가장 많은 프로젝트 호스팅 가장 많은 응용프로그램 지원 직접 호스팅하기 위해서는 기업용 Enterprise버전 필요	오픈소스로 이루어져 있으므로 직접 호스팅할 수 있음	직접 호스팅 가능 Atlassian사의 프로젝트 관리용 제품들과 완전한 호환