



구성 관리

Python 기본 사항

발표자 이름

날짜

학습 내용

강의 핵심 내용

학습 내용:

- 프로젝트 인프라를 정의합니다.
- 프로젝트가 성공을 거두는 데 프로젝트 인프라가 중요한 이유를 설명합니다.
- 소프트웨어 구성 관리의 목적과 기능을 정의합니다.






프로젝트 인프라

프로젝트 인프라

프로젝트 인프라는 프로젝트가 구성되는 방식입니다. 건축가는 다리의 인프라를 구성하고 소프트웨어 개발자는 코드의 인프라를 구성합니다.

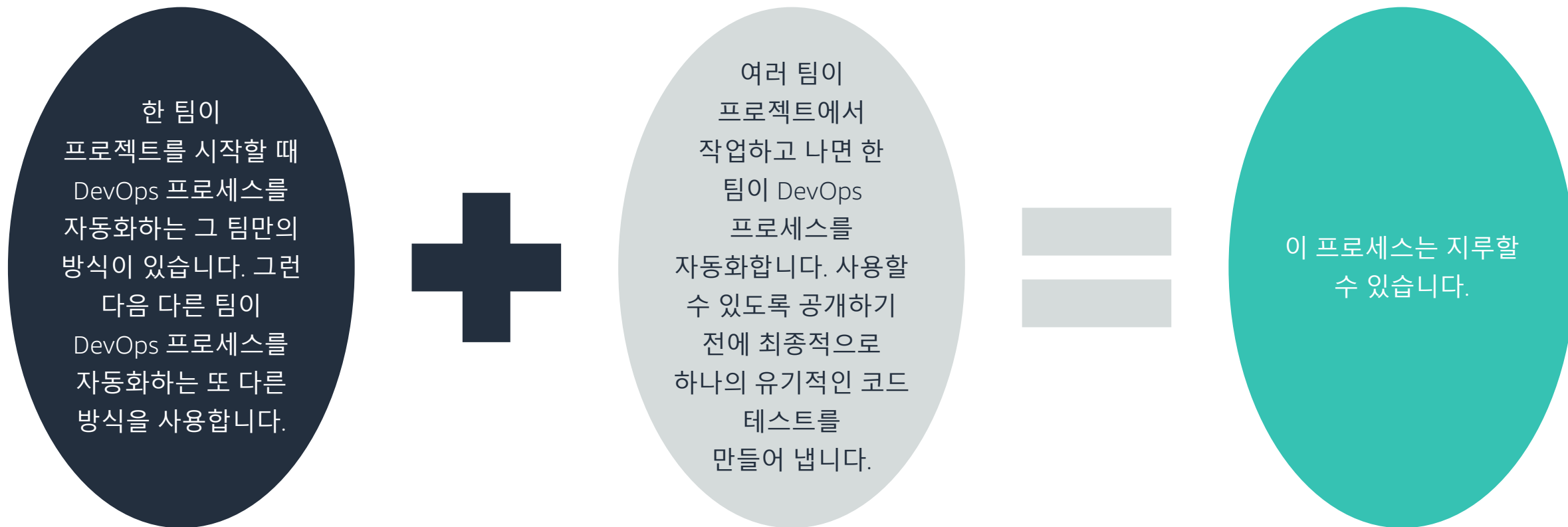


프로젝트 인프라가 잘
구성되면 여러 팀이
하나의 프로젝트에서
같은 방식으로 협업할
수 있습니다.

프로젝트 인프라가 잘
구성되지 않으면 여러
팀이 협업할 때 다른
팀이 무슨 일을 하고
있는지 고려하지 않고
작업하게 됩니다.

전통적인 프로젝트 인프라

전통적인 프로젝트 인프라는 여러 팀이 함께 코드를 개발할 때 효율성이 떨어집니다.



코드 정리

잘 정리된 코드란 어떤 의미입니까?

대부분의 회사에는 직원이 따라야 하는 특정한 코딩 스타일이 있습니다. 이 스타일에는 코드에 있는 변수 이름을 지정하는 방식과 코드 블록에 들여써야 하는 공백의 수가 포함됩니다.

이런 관행은 회사마다 다르다는 점이 중요합니다. 스타일의 세부 사항보다는 스타일을 혼동하지 않는 것이 중요합니다. 이 생기지 않도록 직원이 이 스타일을 따라야 합니다.

여러 팀이 하나의 프로젝트에 참여하는 경우 테스트 세트가 둘 이상이면 안 됩니다. 테스트 세트에는 논리 테스트와 코드 컴파일이 포함됩니다.

도구와 템플릿

여러 팀이 유기적인 인프라를 구축하는 데 많은 도구를 사용할 수 있습니다.

스타일 확인

- **pylint**와 같은 유틸리티는 코드 블록의 들여쓰기가 올바른지 확인하고 형식이 올바르지 않은 코드 블록을 수정하기 위해 실행할 수 있습니다.

논리 확인

- **pytest**와 같은 유틸리티는 코드를 변경해도 여전히 요구 사항을 충족하는지 확인하기 위해 사용할 수 있습니다.



소프트웨어 구성 관리

구성 관리란?

- 코드를 개발하는 과정에서 버전을 추적합니다.
- 개발자가 프로젝트의 서로 다른 부분에서 독립적으로 작업한 후 변경 사항을 프로젝트에 다시 병합할 수 있습니다.
- 버전 관리 소프트웨어(예: Git)는 어떤 코드가 변경되었는지, 누가 변경했는지를 추적합니다.
- 오류가 발생하면 구성 관리를 통해 제대로 작동하는 이전 버전으로 빠르게 롤백할 수 있습니다.



[Jason Long](#)의 Git 로고는 [Creative Commons Attribution 3.0 Unported License](#)에 따라 라이선스를 부여받았습니다.

구성 관리의 작동 방식

- 개발자가 AWS CodeCommit 또는 GitHub 같은 리포지토리에서 코드를 **체크아웃**합니다.
- 코드 작성을 완료하면 개발자가 변경 사항을 리포지토리에 업로드합니다.
- 새로운 코드가 모든 테스트를 통과하면 메인 프로젝트에 **병합**할 수 있습니다.
- 코드를 체크인하고 체크아웃하는 과정은 다음을 통해 수행할 수 있습니다.
 - 명령줄에서 **Git**를 실행합니다.
 - 통합 개발 환경(IDE)에 내장된 PyCharm과 같은 도구를 사용합니다.
- **pylint**와 **pytest** 등의 도구를 실행하는 것도 체크인 과정의 일부입니다.



예제 별 구성 관리

리포지토리(리포) 구성 방식에 따라 다른 단계가 필요할 수도 있으므로, 다음과 같은 간단한 예제에서는 구성 관리용 Git를 사용합니다.

원격 리포의 사본을 가져옵니다. `$git @<examplerepo.org>:<username>/<sourcecode>.git`

로컬에 변경 사항을 커밋합니다. `$git -commit -m "Message about the changes."`

변경 사항을 리포에 다시 푸시합니다. `$git push`

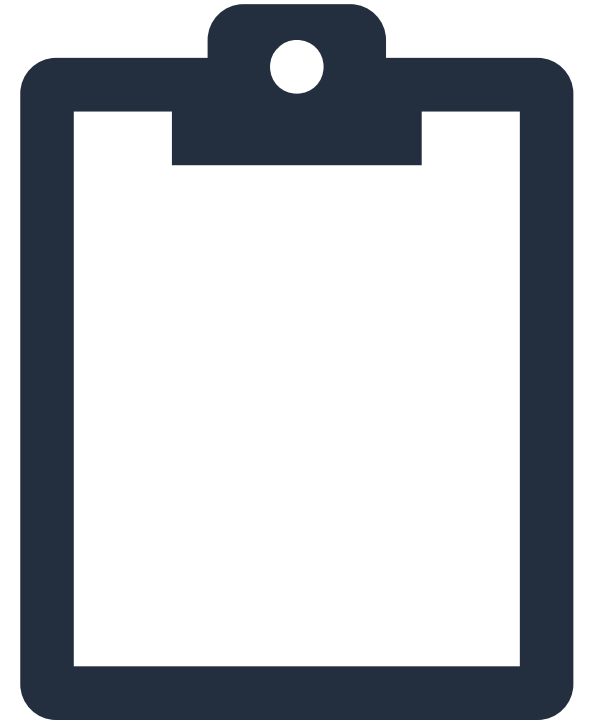
구성 관리 버저닝

- 개발자가 코드를 업데이트할 때 소프트웨어의 새 버전을 배포할 책임이 있는 릴리스 관리자가 변경 사항을 모니터링할 수 있습니다.
- 모든 테스트와 기능을 검증한 후에 릴리스 관리자가 리포지토리의 내용에 따라 소프트웨어의 새로운 디스트리뷰션을 만듭니다.
- 이제 릴리스 관리자가 소프트웨어를 **버저닝**하여 문제 발생 시 고객의 문제를 관리할 수 있습니다(여기에서 **이전 버전으로의 롤백**이 중요해짐).
- 대부분의 경우 숫자 값(예: **버전 3.2.1**) 형태로 버전을 지정합니다.



구성 관리 회계

- 팀 리드와 관리자가 프로젝트의 상태를 물어볼 때가 있습니다.
- 구성 관리가 되어 있으면 팀은 프로젝트의 리포지토리에서 체크인과 체크아웃 및 기타 활동을 검사하여 현재 진행하는 일을 신속 보고할 수 있습니다.



구성 관리 보안

- 리포지토리에 대한 액세스 권한을 부여해야 하므로, 권한 없는 사람이 소스 코드에 액세스할 수 없도록 합니다.
- 액세스가 기록되어 다음을 알 수 있습니다.
 - 코드를 체크아웃/체크인한 사람
 - 체크아웃/체크인이 진행된 시기
 - 커밋된 변경 사항



요점



- 프로젝트 인프라는 프로젝트가 목표를 달성하도록 하는 데 중요한 규율입니다.
- 또한 프로젝트 인프라를 통해 Python 코드가 올바른 스타일로 작성되고 예상대로 기능하도록 합니다.
- **pylint**는 코드의 스타일을 확인하고 **pytest**는 코드가 예상대로 작동하는지 테스트합니다.
- 소프트웨어 구성 관리에는 프로젝트에 사용되는 코드를 관리하기 위해 코드 리포지토리를 사용하는 일이 포함됩니다.
- 코드 리포지토리를 사용할 수 있도록 도구가 IDE에 내장되어 있으며, 명령줄 도구인 **Git**도 사용할 수 있습니다.