

# SW 개발·테스트 중심 DevOps 환경구축 및 활용 (Python)

---

강사 정보

(주)시네틱스 대표 한동준  
[handongjoon@gmail.com](mailto:handongjoon@gmail.com)  
[dongjoon.han@synetics.kr](mailto:dongjoon.han@synetics.kr)

# 교육 소개

---

## □ 목적

- DevOps 운영에 필수인 개발(Development) 환경의 자동화를 위하여 오픈소스 도구를 활용한 ALM(Application Lifecycle Management) 환경 구축 및 운영 역량을 확보

## □ 내용

- DevOps에서 ALM의 필요성에 구성에 대한 이해
- 주요 오픈소스 ALM 도구 이해
  - 단독으로 사용하는 방법
  - Jenkins, VSCode 에서 사용하는 방법

## □ 교육에서 강조하는 부분

- 도구 설치, 도구 간 연동 방법
- Jenkins 중심의 활용 방법

## □ 교육 시간 내에 상대적으로 적게 다루는 부분

- Python 언어 자체에 대한 부분 (문법)
- 국내에 이미 책과 교육이 활성화 되어 있는 도구의 기본 사용 방법: Git

# 강의 순서

---

1. DevOps 와 오픈소스 개발 환경 자동화의 이해
2. 버전 관리: Git
3. 지속 통합: Jenkins
4. Python Formatter: Black
5. 단위 테스트와 커버리지 분석: Unittest & Coverage
6. 룰 기반 소스코드 분석: SonarQube

# 시간표

구분	시간	내용
1일차	08:30 ~ 09:00	교육 접수 및 안내
	09:00 ~ 12:00	<ul style="list-style-type: none"><li>○ DevOps와 오픈소스 개발 환경 자동화의 이해</li><li>○ 서버 환경 구성빌드 관리</li><li>○ Git을 이용한 버전 관리</li></ul>
	12:00 ~ 13:00	점심식사
	13:00 ~ 18:00	<ul style="list-style-type: none"><li>○ Jenkins를 이용한 지속적 통합</li><li>○ Black을 활용한 소스코드 스타일 관리</li></ul>
2일차	09:00 ~ 12:00	<ul style="list-style-type: none"><li>○ SonarQube를 이용한 소스코드 정적 분석</li></ul>
	12:00 ~ 13:00	점심식사
	13:00 ~ 18:00	<ul style="list-style-type: none"><li>○ Python 단위 테스트</li><li>○ 테스트 커버리지 분석</li><li>○ 기타 소스코드 품질 향상을 위한 오픈소스</li></ul>

# 개발자 PC 구성

---

## □ H/W

- 실습 PC 사양

## □ S/W

- Python 3.10 (최신 버전), PIP
- VSCode (최신 버전)
- Git 최신 버전

# 서버 구성 (1대)

## □ Azure (공통)

- Windows Server 사용

## □ S/W

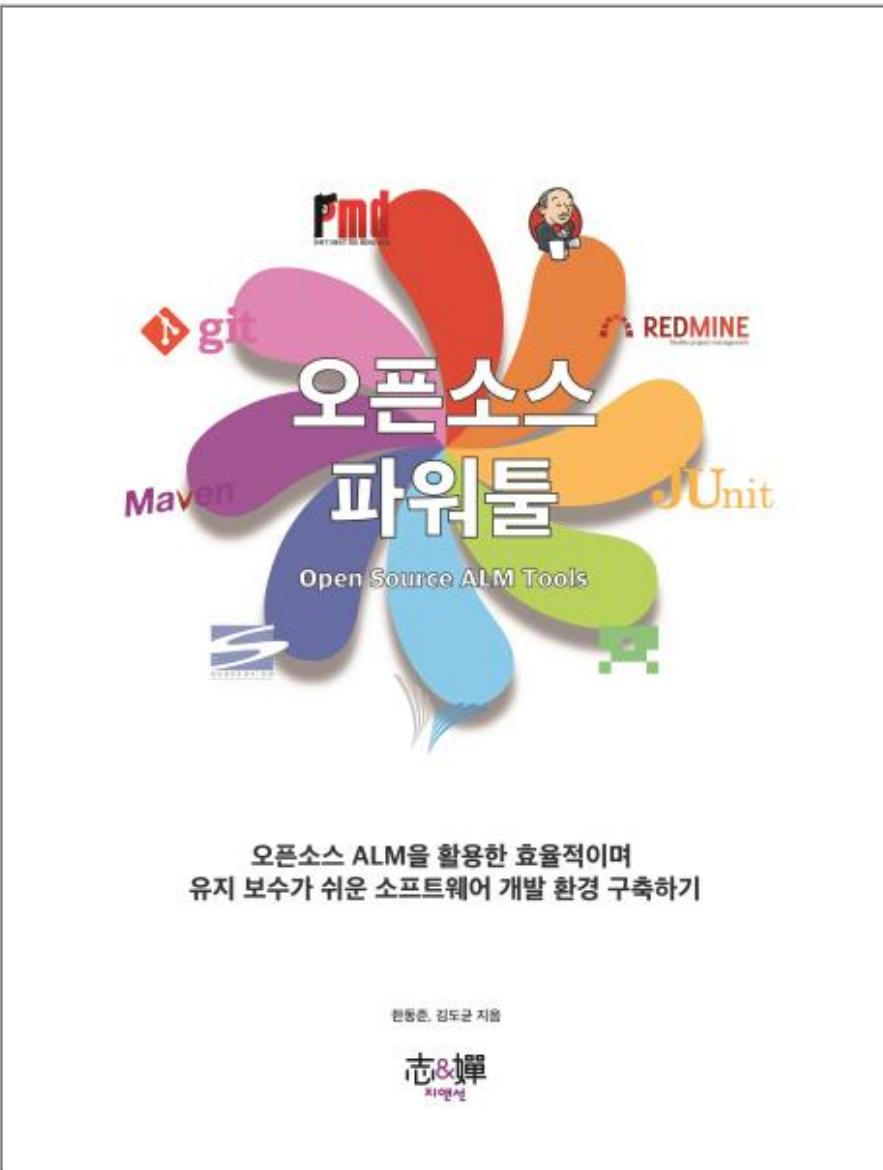
- Git
- Jenkins
- Python

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with various service icons like App Services, SQL Database, and Storage accounts. The main area is the 'Dashboard' which includes four cards: 'Resource Group' (Resource monitor), 'Web Front End' (CPU Percentage and Memory Percentage), 'Database' (DTU percentage and Database size percentage), and 'Processes' (All runs). Below these cards is a 'Cognitive Services' section with a DTU quota chart. At the bottom, there are summary metrics: 27 active VMs, 11 sessions, 623 page views, and 277 active users. Above the dashboard, there's a large promotional banner with the text 'Create your Azure free account today' and 'Get started with 12 months of free services'. It features two buttons: 'Start free >' and 'Or buy now >'. The URL 'www.microsoft.com/azuresdksetup' is visible at the bottom right of the banner.

# 오픈소스 파워툴 소개

---

# 오픈소스 파워툴 (지앤선, 2017.05)



## [참고] 오픈소스 파워툴 책 목차

장	도구명	용도	교육 포함
1	ALM	전체 구성	V
2	Maven	Java 빌드	V
3	Subversion	버전 관리	
4	Git	버전 관리	V
5	SCMManager	버전 관리 통합 서버	
6	Jenkins	지속 통합	V
7	JUnit	단위 테스트	V
8	Cobertura	테스트 커버리지	V
9	PMD	룰기반 정적분석	V
10	JavaNCSS	Java 매트릭	V
11	JDepend	패키지 의존성 분석	V
12	Redmine	이슈 관리	
13	Mylyn	이슈 관리 클라이언트	
14	시나리오	다 같이 해보기	V
15	NSIS	Windows 배포판 생성	

1

# DevOps 와 오픈소스 개발 환경 자동화의 이해

---

# 일반적인 소프트웨어 개발과 유지보수



개발 단계



개발 후반부 통합 빌드 및 배포 / 테스트 횟수만큼 배포

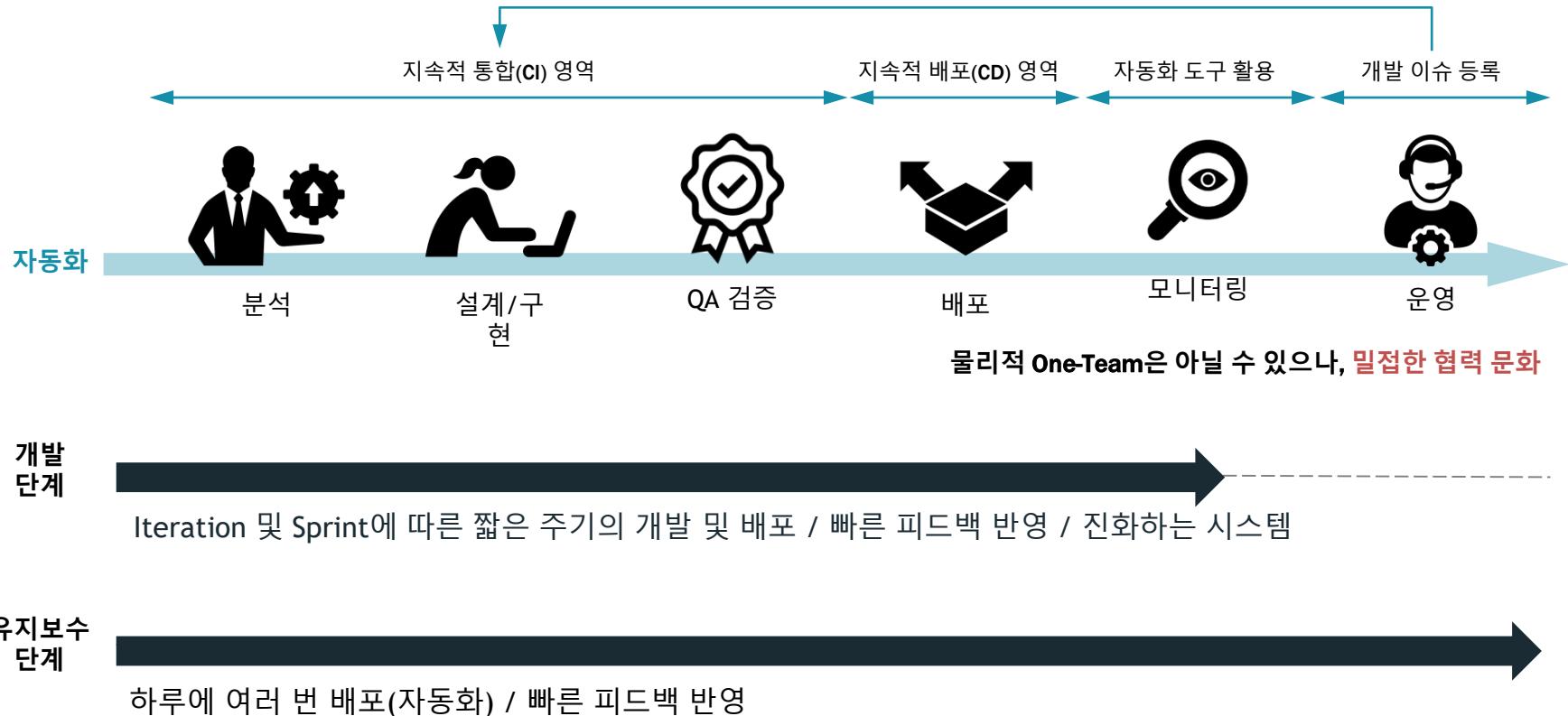
유지보수 단계



1달에 1~2회 정기 배포 / 문제 발생 시 긴급 배포

느린 피드백 및 반영

# DevOps 소프트웨어 개발과 유지보수



# 품질과 안정성이 확보된 소프트웨어를 더 빠르게 제공

- 소프트웨어 개발/운영 담당자의 원활한 의사 소통
- 개발/운영 프로세스의 단순화 및 자동화 (휴먼 에러 최소화)
- 지속적인 소프트웨어 품질 및 안정성 확보

## 자동화된 지속적 통합(CI) / 지속적 배포(CD) 환경 구축

## 지속적 통합(CI: Continuous Integration)

SW 통합 오류를 개발 초기부터 예방하는 것

코딩 시작한 날

- 통합 빌드, 정적 분석, 테스트 커버리지, 함수 복잡도

- 문제 발생 시 당일 조치

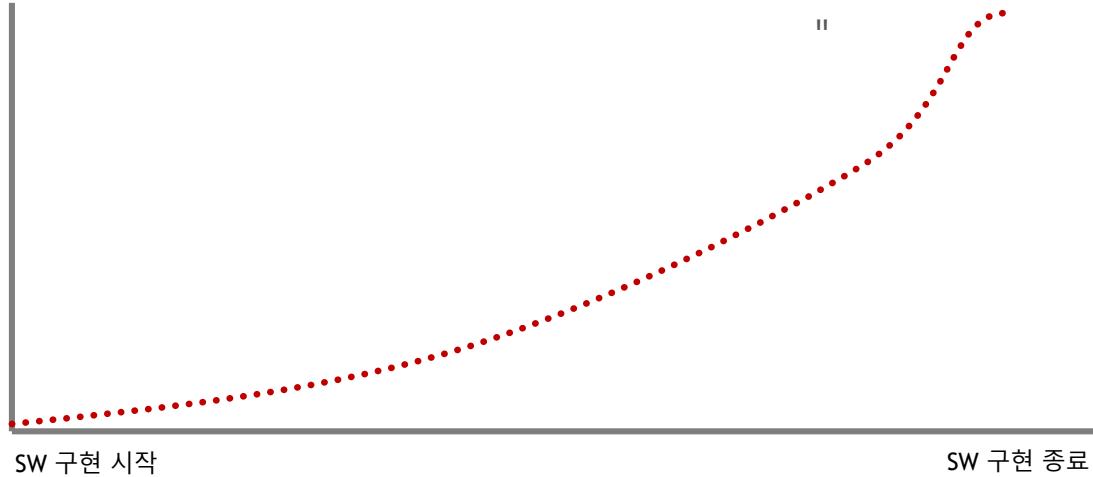
모든 SW 개발에 적용 가능한 정의

## “소프트웨어 구현 후반부에”

### 소스코드/소프트웨어 문제들

- 빌드 오류 수
- 정적분석 위반사항 수
- 보안 룰셋 위반사항 수
- 테스트 커버리지 불만족 비율
- 함수, 모듈 크기 위반율
- 복잡도 위반율

“소스코드가 품질 요구를  
**너무** 많이 위반했는데요...

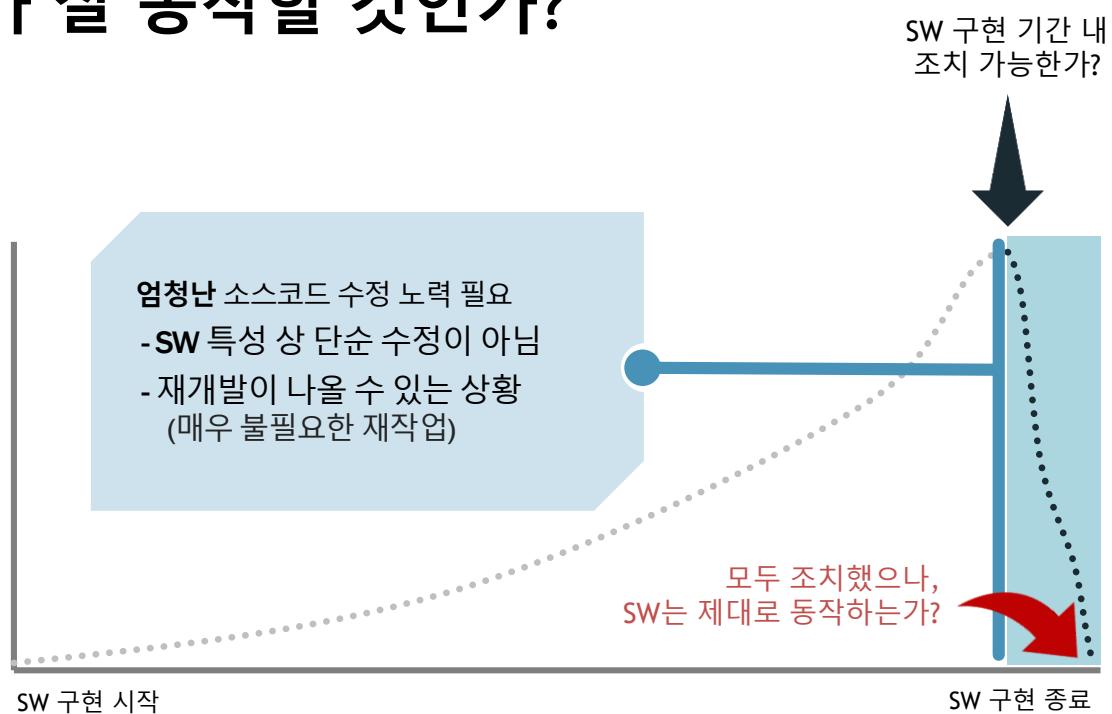


이 상황에서 품질을 만족해야 한다면

# SW 구현 종료까지 조치 가능한가? 조치가 끝나면 SW가 잘 동작할 것인가?

# 소스코드/소프트웨어 문제들

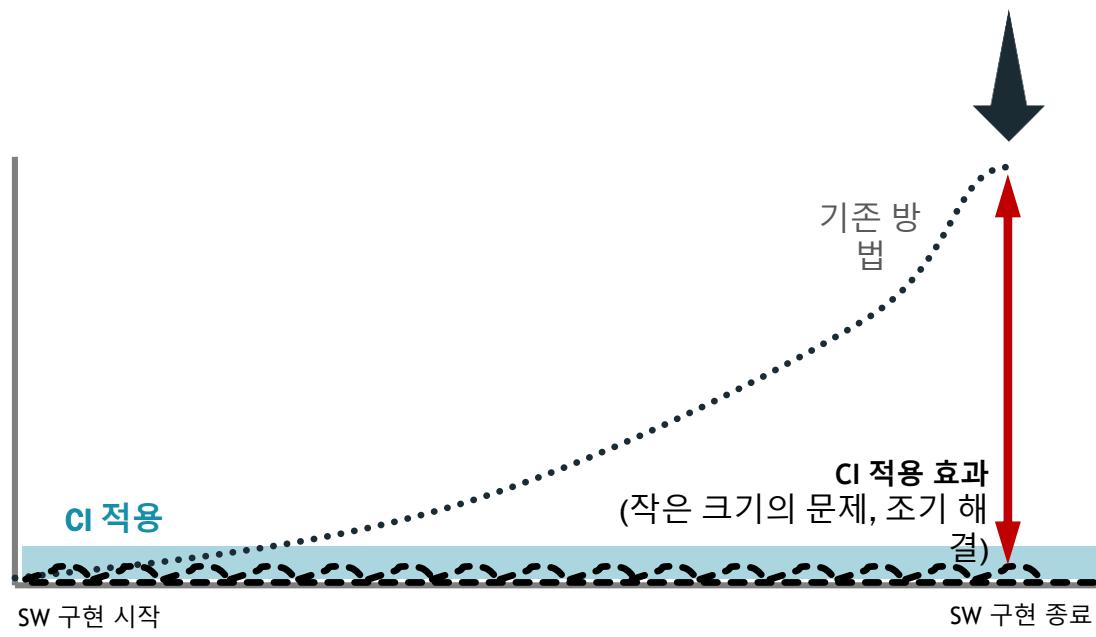
- 빌드 오류 수
  - 정적분석 위반사항 수
  - 보안 룰셋 위반사항 수
  - 테스트 커버리지 불만족 비율
  - 함수, 모듈 크기 위반율
  - 복잡도 위반율



### SW 구현 시작부터 지속적 통합(CI)을 적용, 도구가 알아서 검사/보고하여 품질과 안정성 조기 확보

#### 소스코드/소프트웨어 문제들

- 빌드 오류 수
- 정적분석 위반사항 수
- 보안 룰셋 위반사항 수
- 테스트 커버리지 불만족 비율
- 함수, 모듈 크기 위반율
- 복잡도 위반율



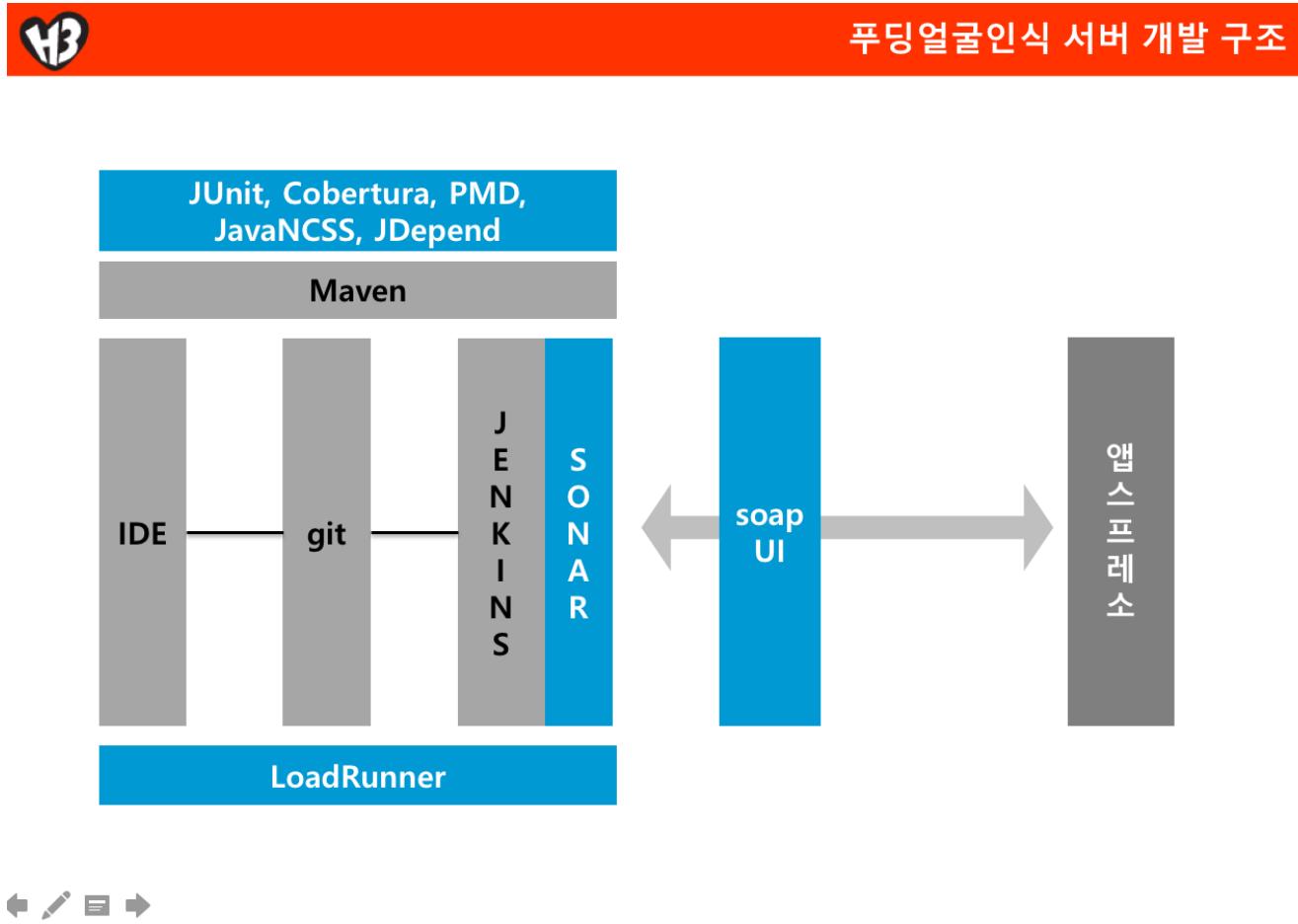
## 지속적 통합(CI)과 지속적 배포(CD)의 장점

---

- 소스코드/소프트웨어 빌드 및 품질 문제점을 빠르게 식별하고 조치
- 빌드, 품질 검사, 문제점 보고, 배포 자동화로 휴먼 에러 및 투입 자원 감소
- 빌드 환경의 형상을 소프트웨어 폐기까지 유지
- 오픈소스를 기반으로 구축 가능 및 다양한 구축 사례 공유

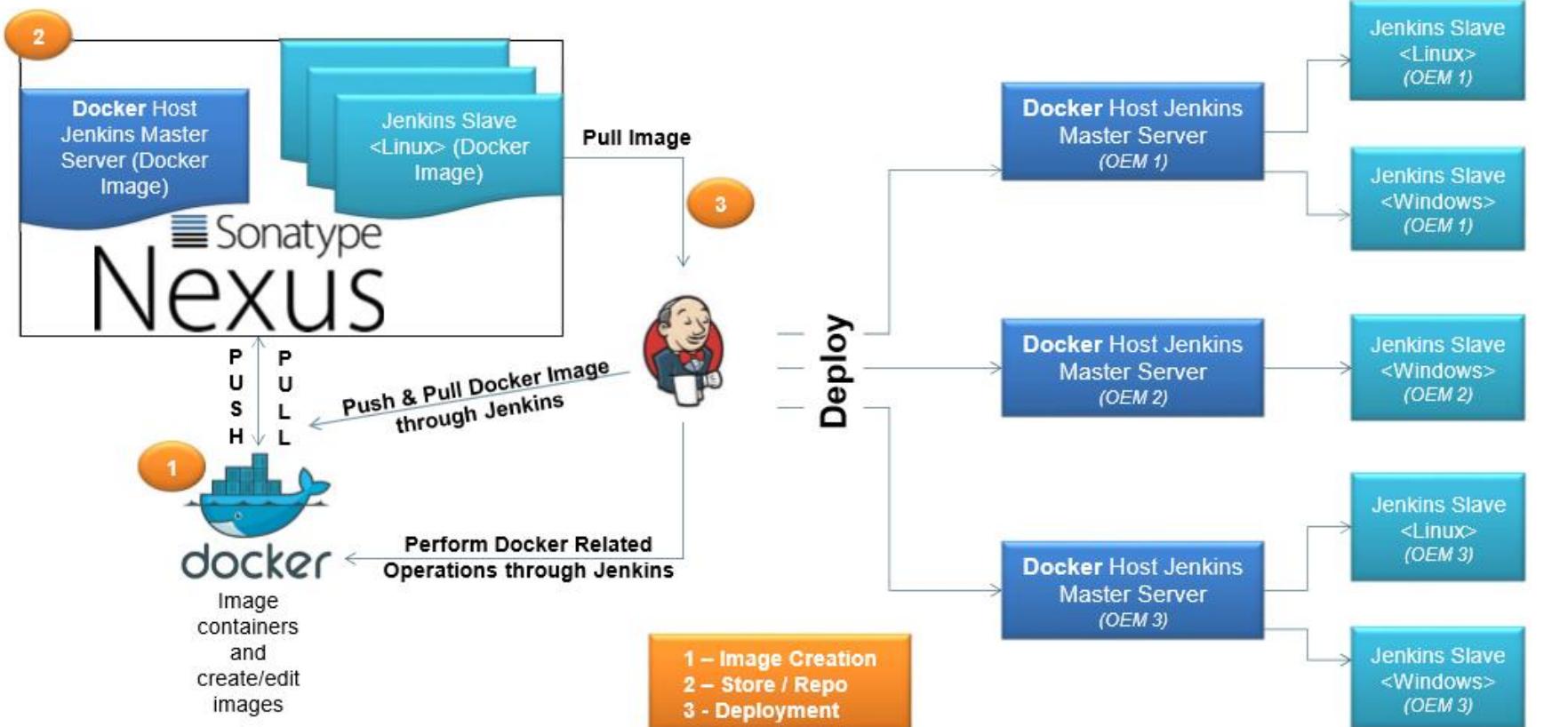
# 이제는 많이 하는 소프트웨어 개발 활동

## 2012 H3 컨퍼런스 발표 – 이미 ALM 사용사례 발표

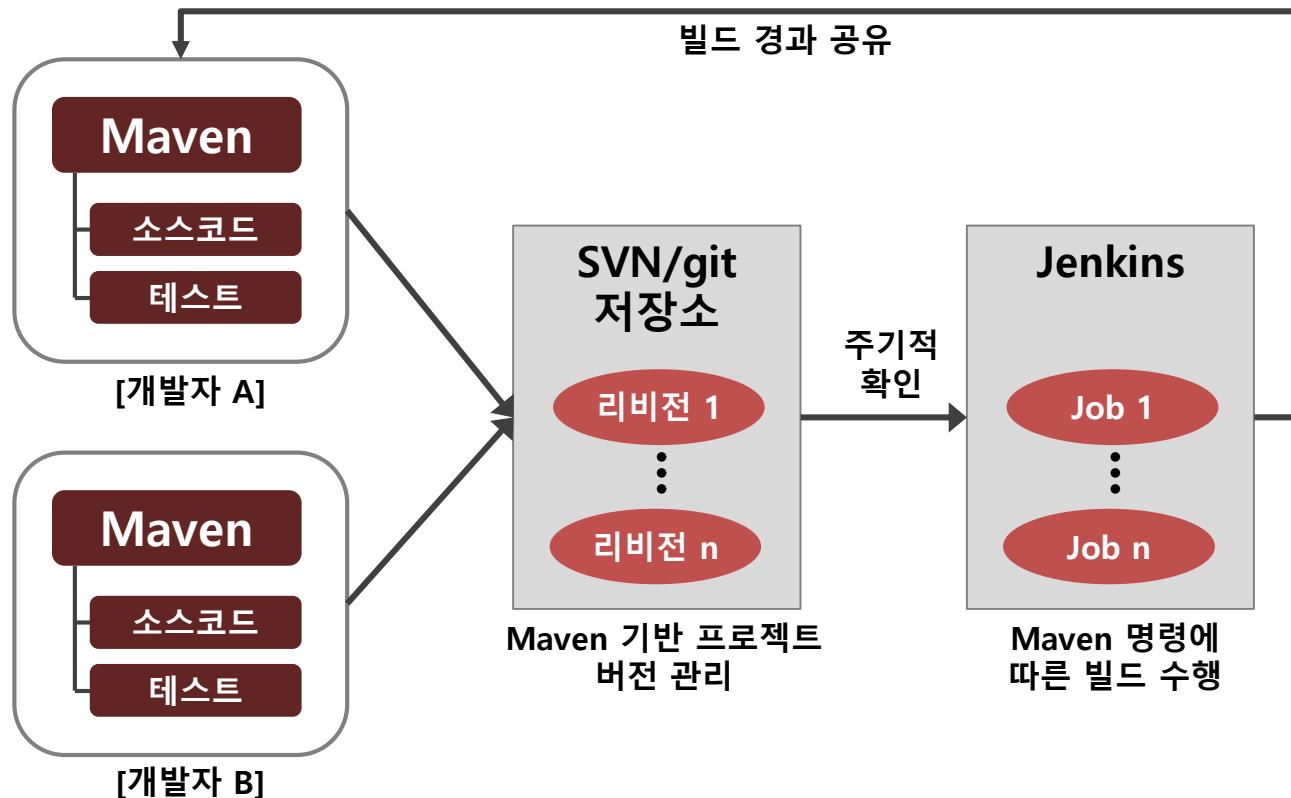


# 심지어 자동차 SW 개발에도 적용

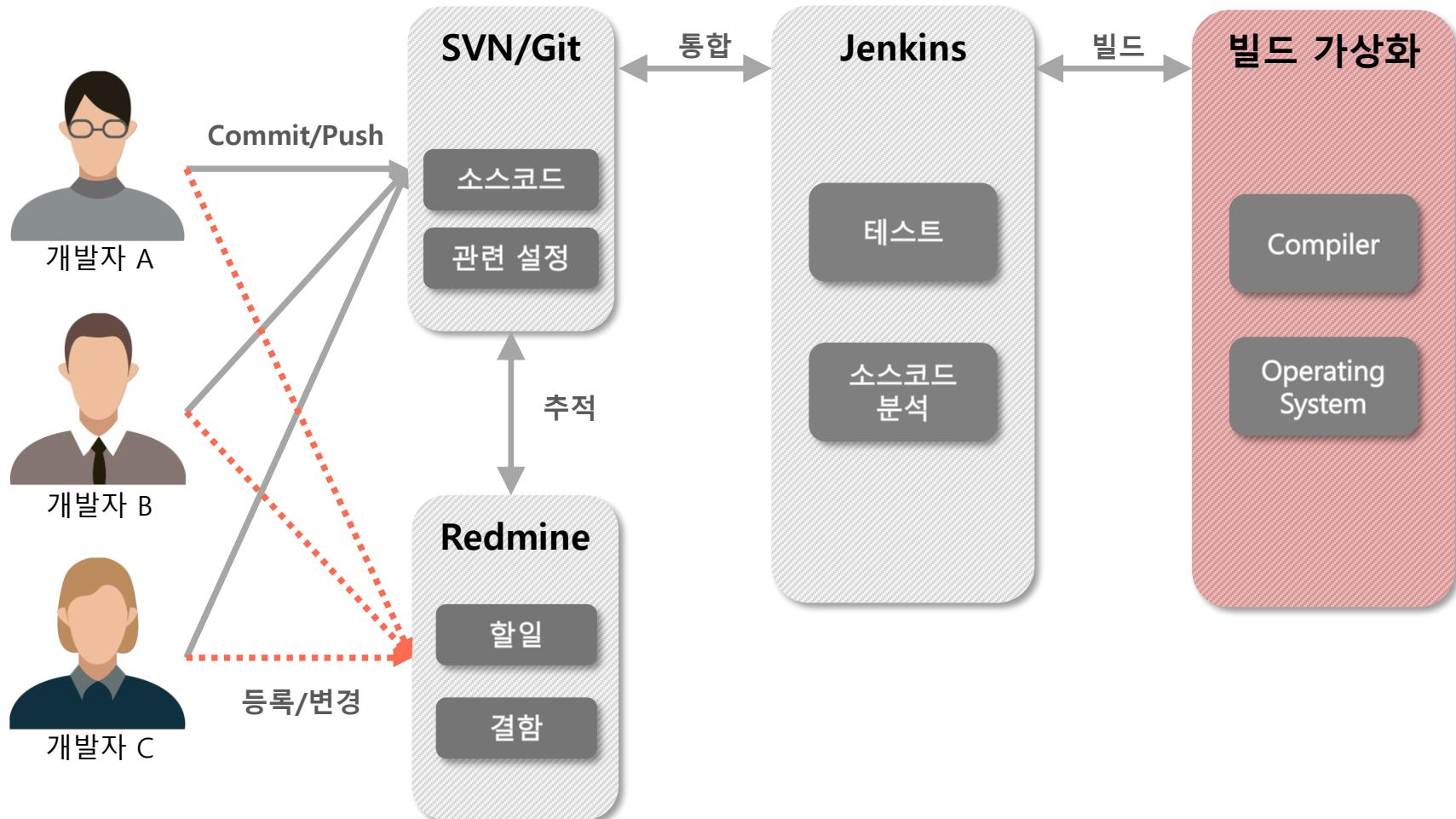
## Docker Deployment Architecture



# 일반적인 오픈소스 기반 ALM 구성 (Java 기반)



# 보다 확대한 구성



# 서버 설정



## 가입 후 Azure에 서버 추가

---

### □ Azure에 Windows VM 추가

- 네트워크 보안에서 80, 8080 인바운드 허용

### □ 설치 필요 SW

- Java (Jenkins 실행)
- Python
- Git

# 개발 PC 설정



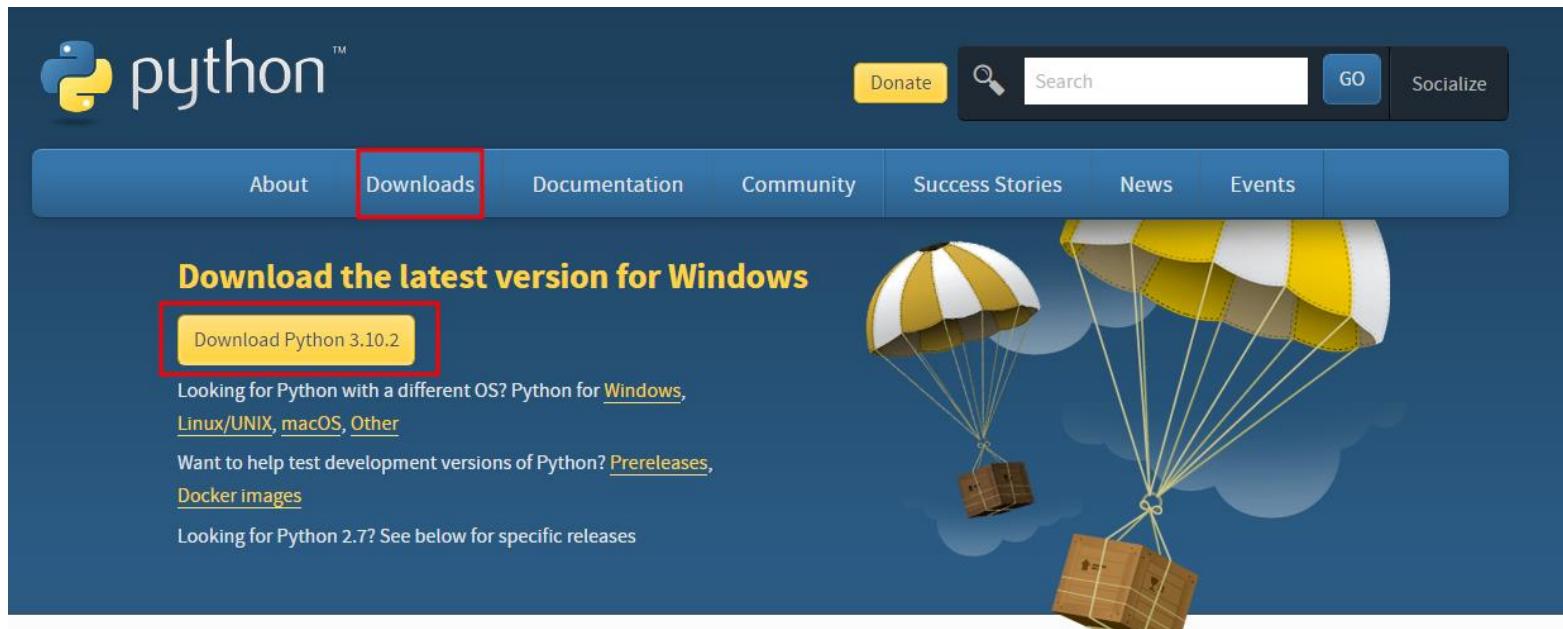
# Python 설치

## □ Windows 환경에서 주의할 점

- 터미널(명령 프롬프트)에서 Python을 입력하는 경우, Python이 없으면 Windows Store로 이동하여 설치 요구
  - 이렇게 설치하는 경우, 환경 설정이 완전하게 진행되지 않음
- Python 홈페이지에서 Windows 버전 다운로드하여 설치 권장

## □ 설치 버전

- 3.10.X

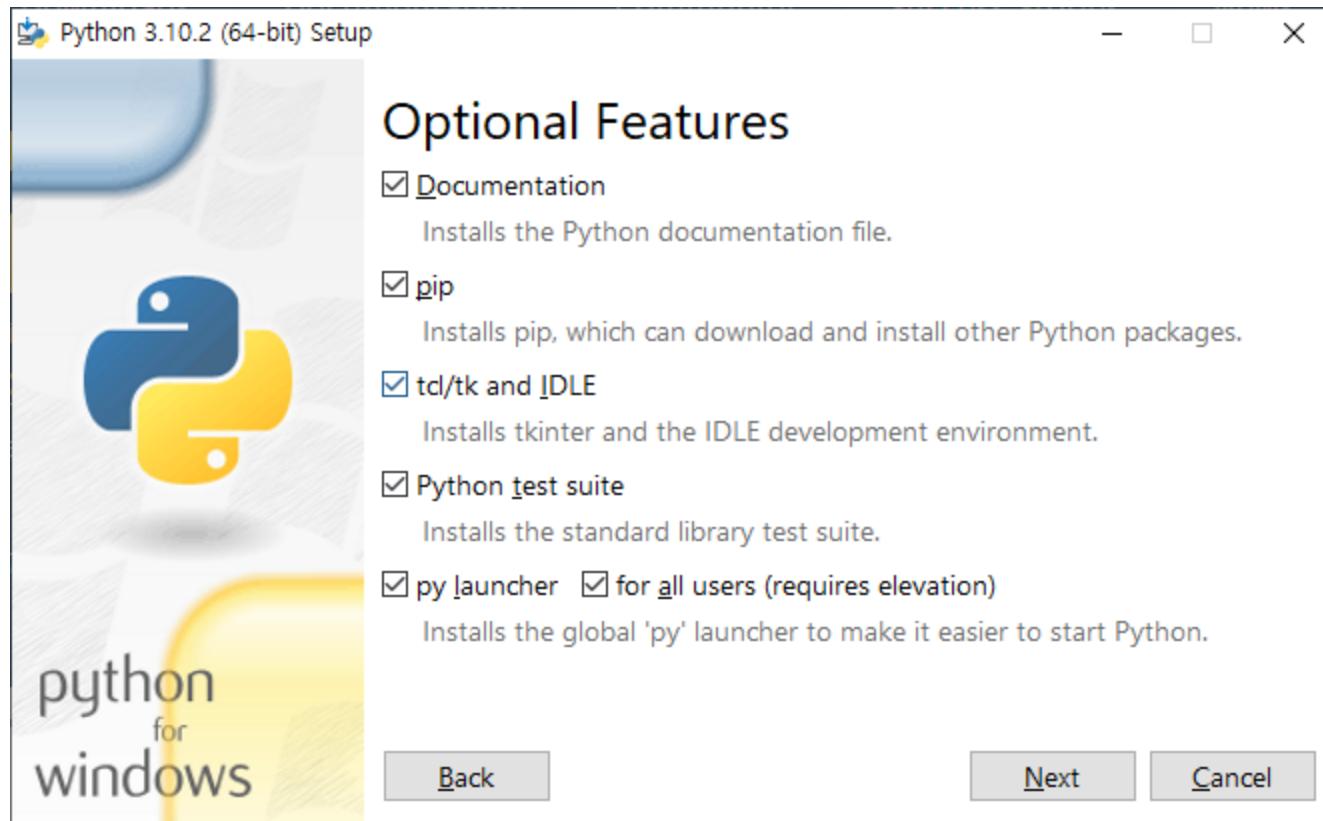


# Python 설치(계속)

- Add Python.. 선택
- Customize Installation 클릭



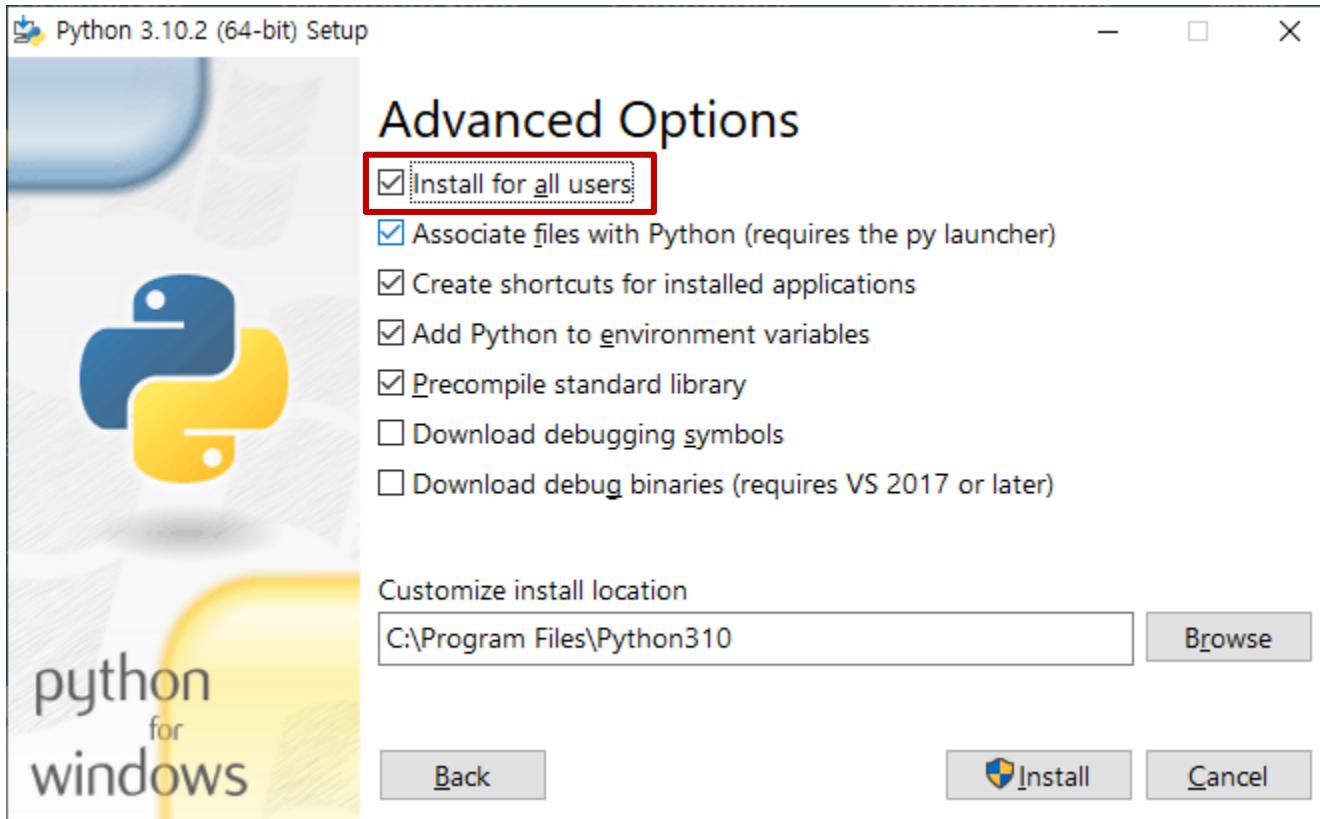
# Python 설치(계속)



# Python 설치(계속)

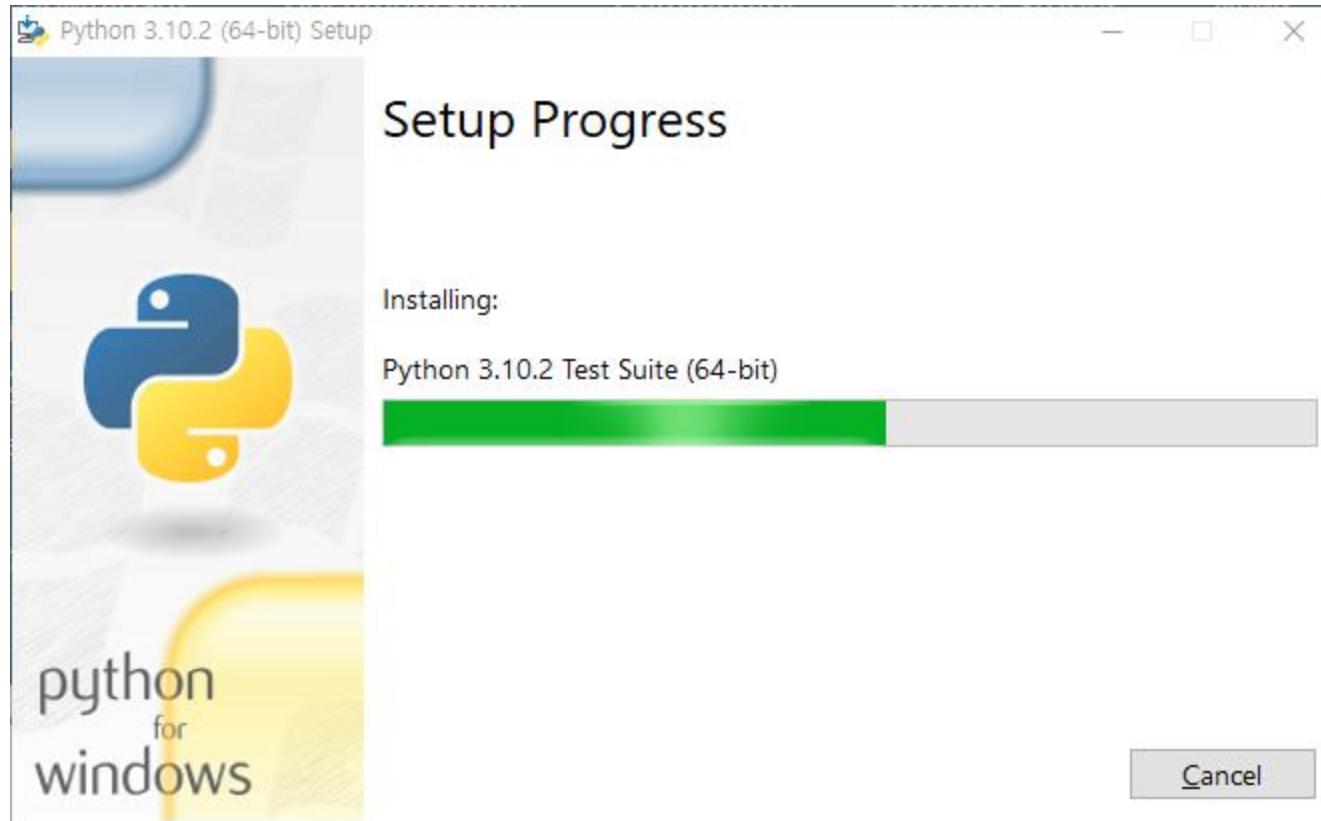
□ Install for all Users 선택

□ Install 클릭



# Python 설치(계속)

## □ 설치 진행



# Python 설치(계속)

## □ 설치 확인

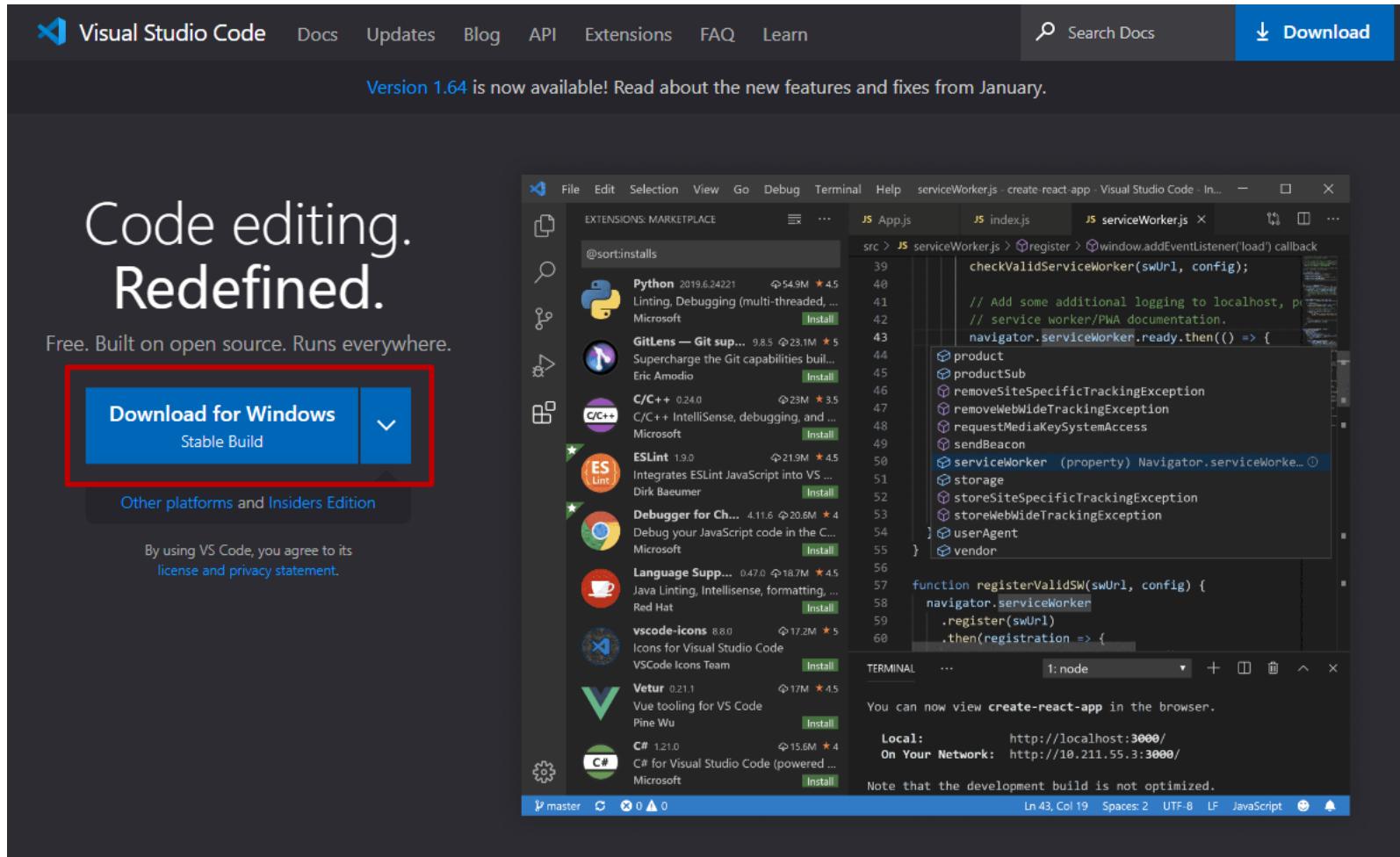
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19044.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\devops>python --version
Python 3.10.2

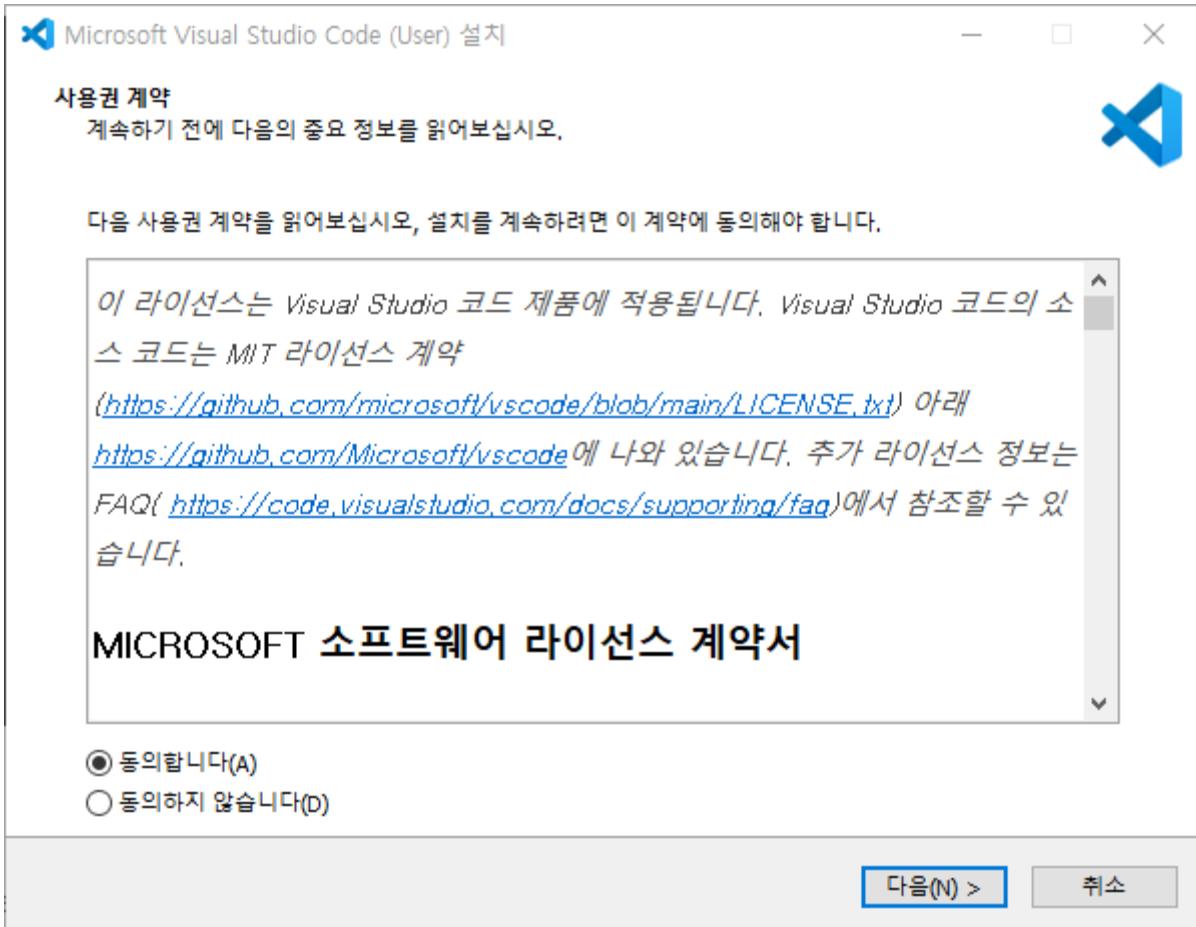
C:\Users\devops>
```

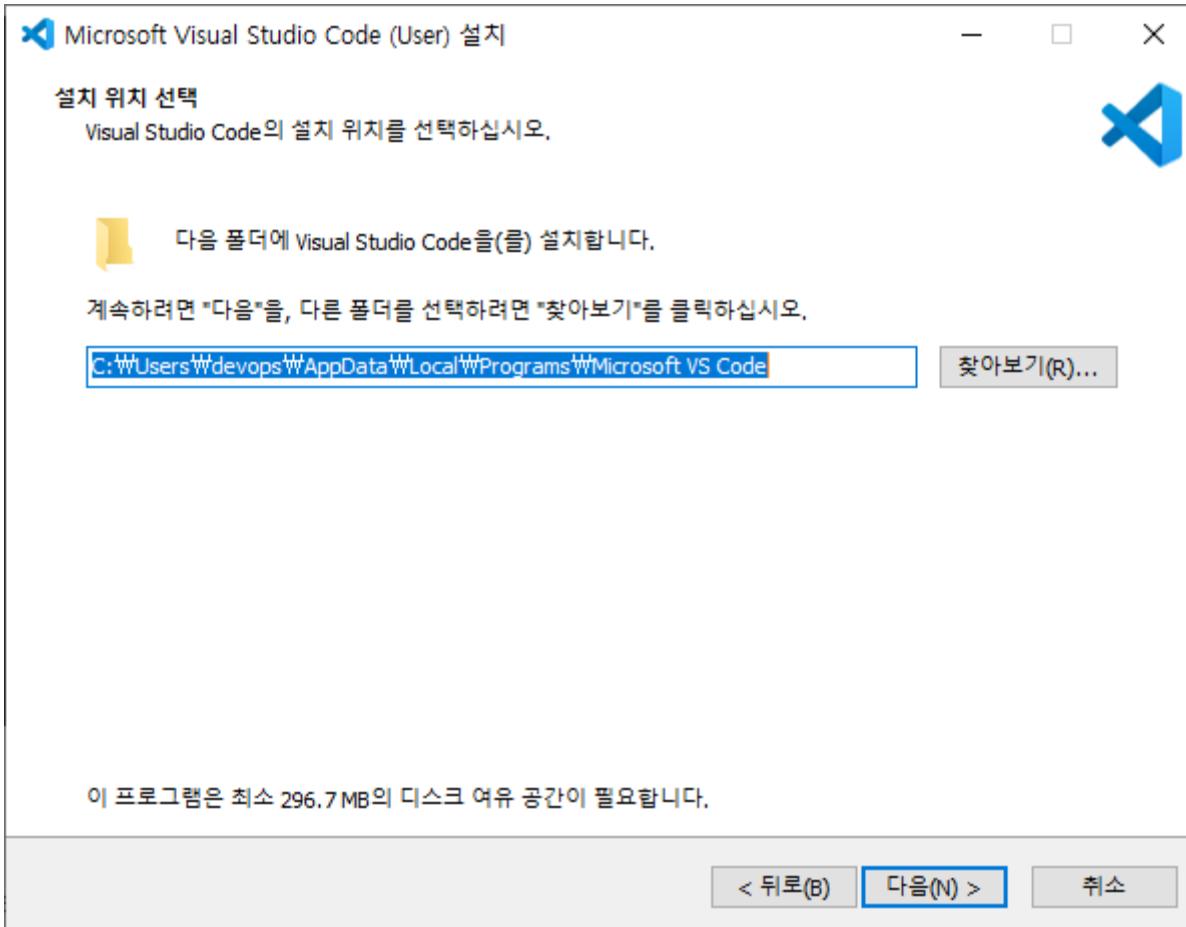
# VSCode 설치

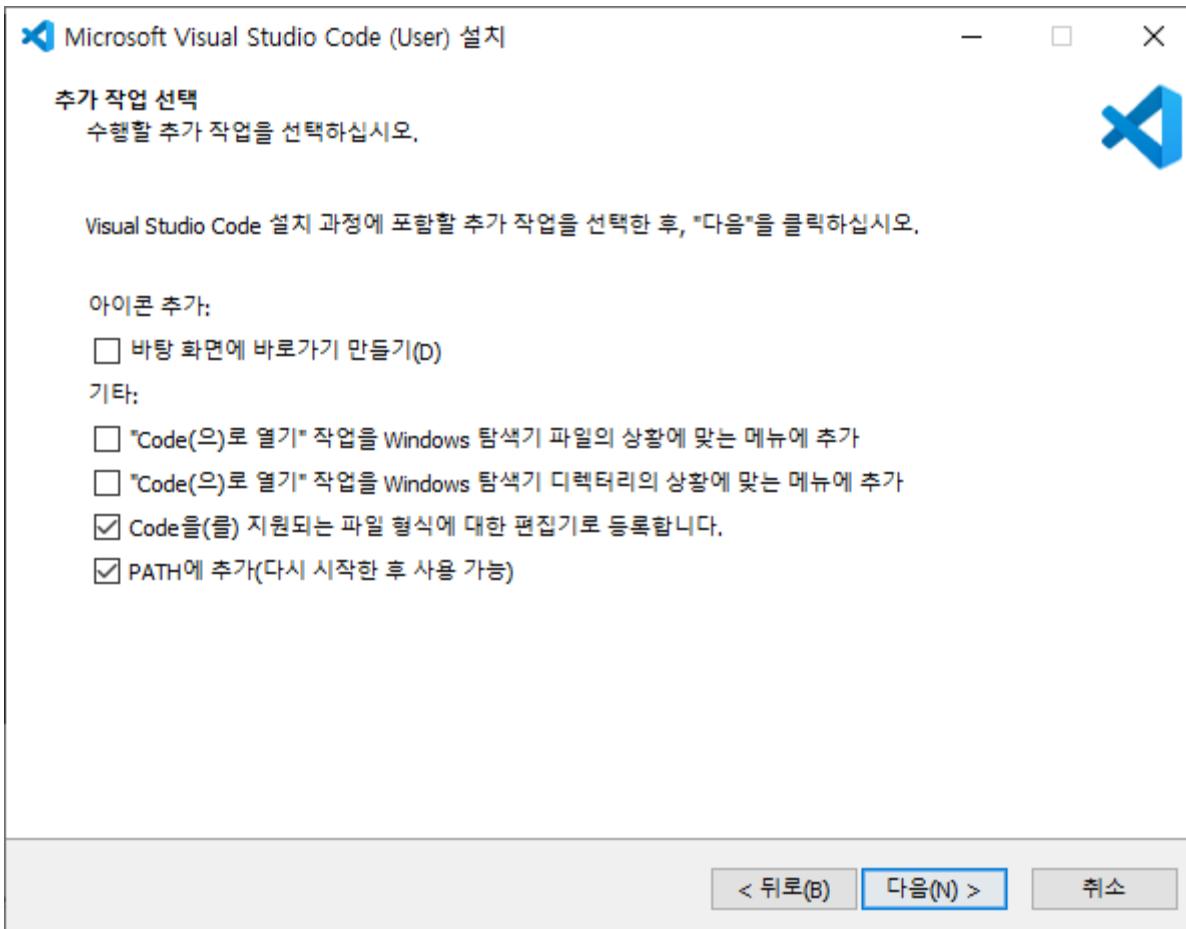
## □ 공식 웹 사이트에서 최신 버전 다운로드



The screenshot shows the official Visual Studio Code website. At the top, there's a navigation bar with links to Docs, Updates, Blog, API, Extensions, FAQ, and Learn. To the right of the navigation is a search bar labeled "Search Docs" and a blue "Download" button. Below the navigation, a message says "Version 1.64 is now available! Read about the new features and fixes from January." The main content area features a large banner with the text "Code editing. Redefined." and "Free. Built on open source. Runs everywhere." Below the banner is a prominent blue "Download for Windows" button, which is highlighted with a red rectangular box. To the right of the download button is a dropdown menu showing "Stable Build". Further down, there are links for "Other platforms" and "Insiders Edition". A note below the download buttons states: "By using VS Code, you agree to its license and privacy statement." On the right side of the screenshot, a screenshot of the VS Code interface is displayed. It shows multiple tabs for "App.js", "index.js", and "serviceWorker.js". The code editor has several lines of JavaScript code, and the bottom status bar indicates the file is "master" and has 0 changes. The terminal tab shows the command "node" and some network information. The status bar also shows "Ln 43, Col 19" and "JavaScript".

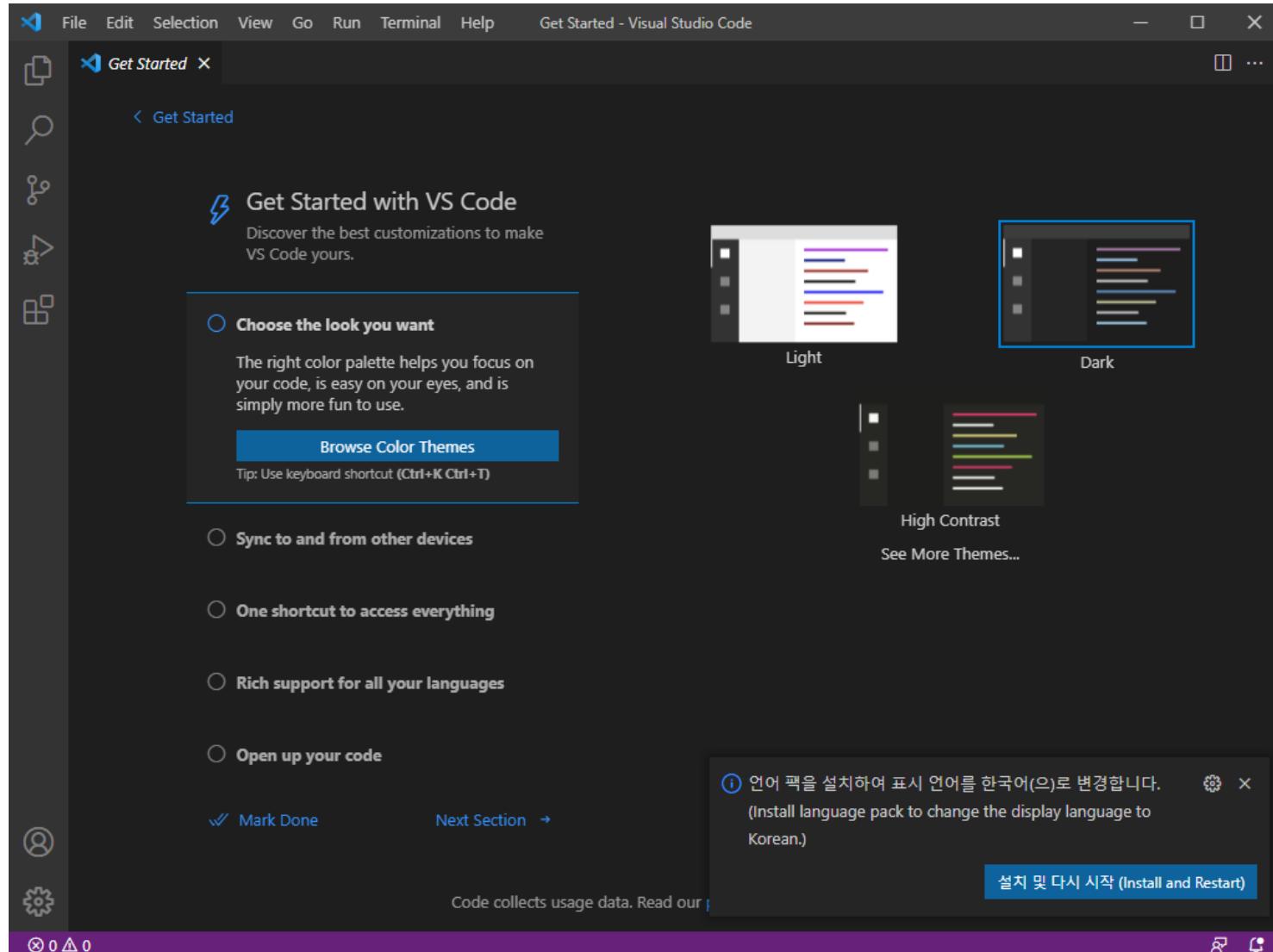






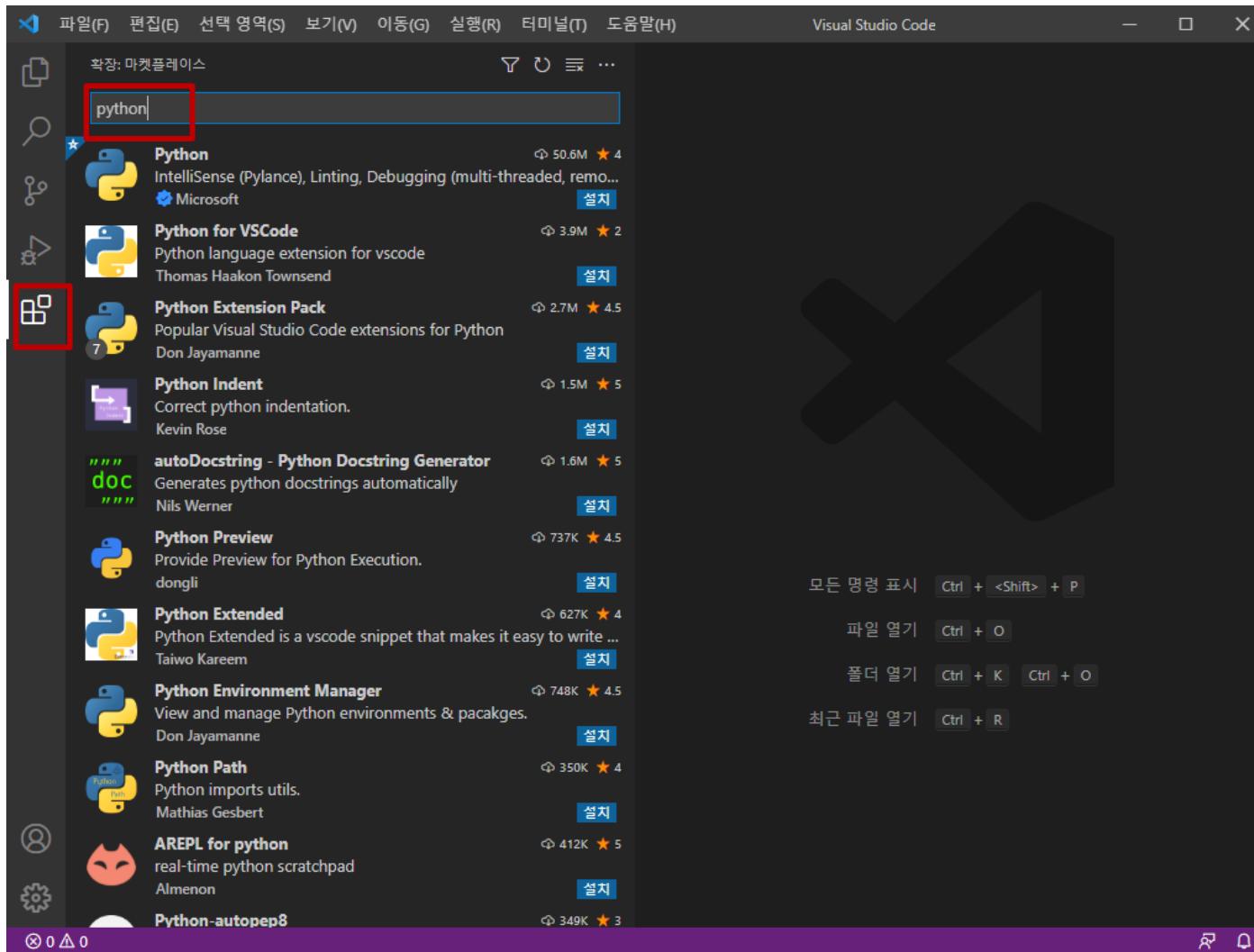
# VSCODE 첫 실행 화면

## □ 필요 시 언어팩 설치 후 재시작



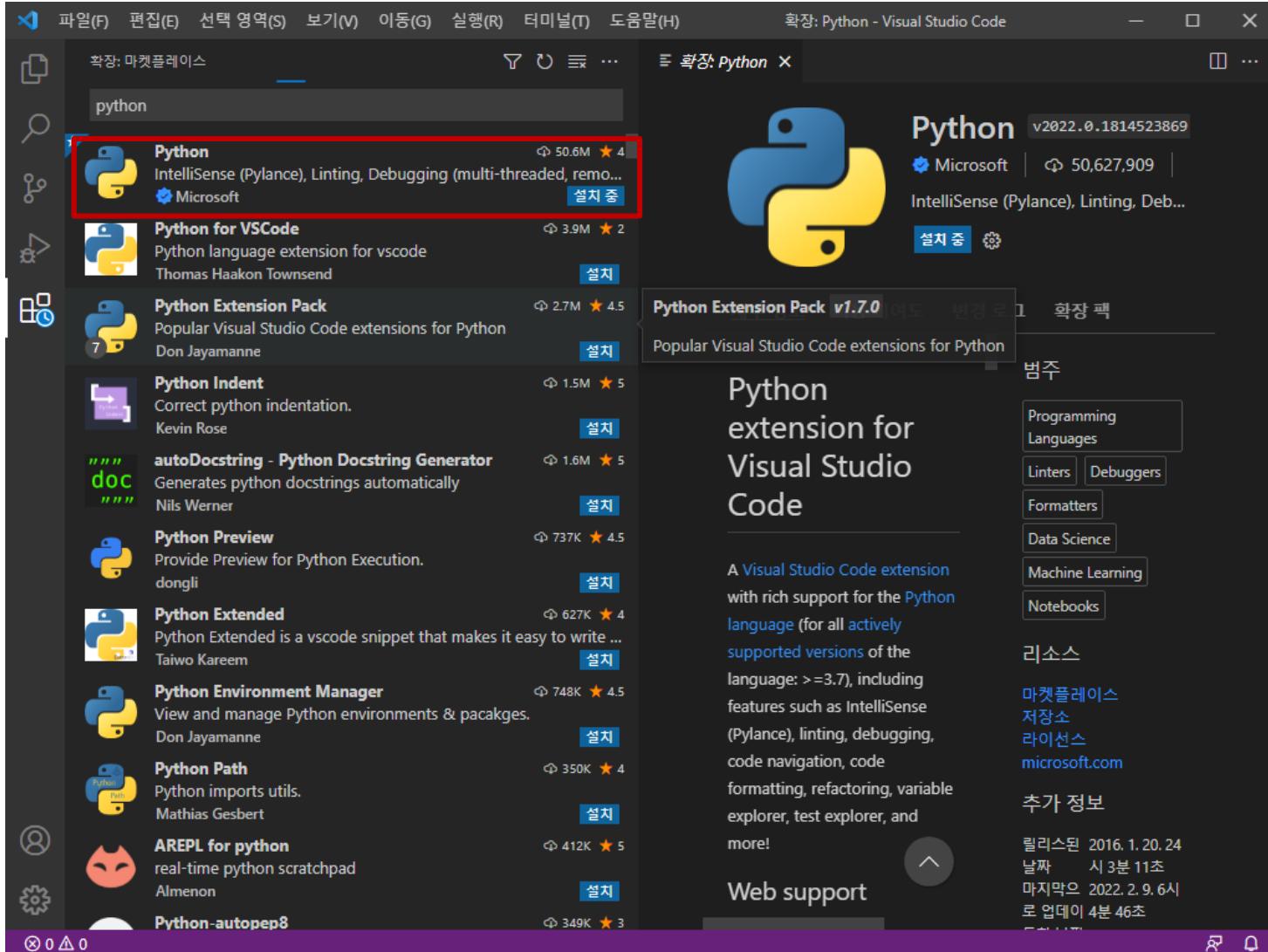
# Python 확장 검색

## □ 확장에서 Python 검색



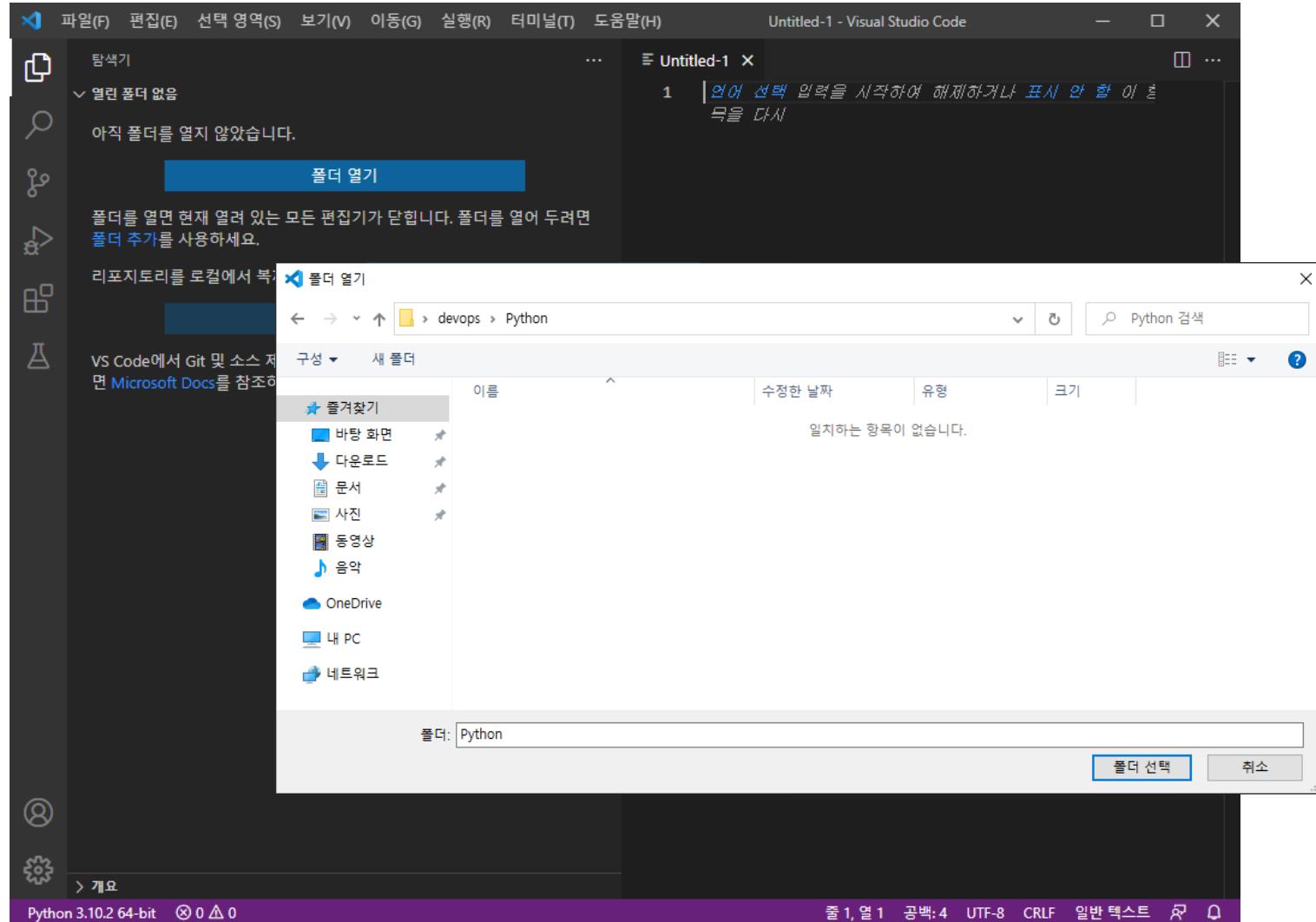
# Python 확장 설치

## □ Python 확장 설치 (목록 제일 처음 표시)



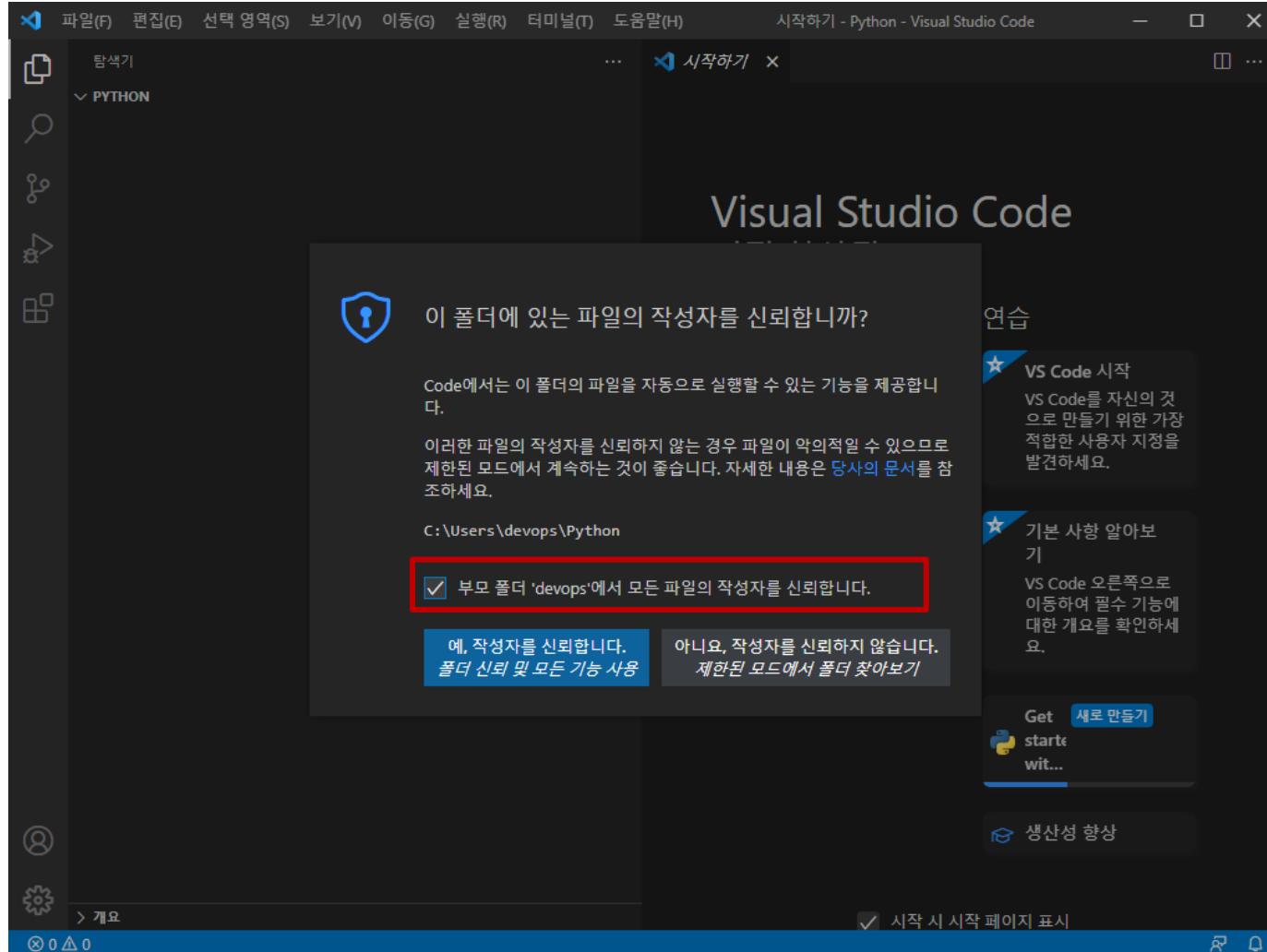
# 소스코드 폴더 지정

## □ 폴더 열기를 선택해서 적절한 위치 지정 (예제는 기본 폴더에 DevOps 생성)



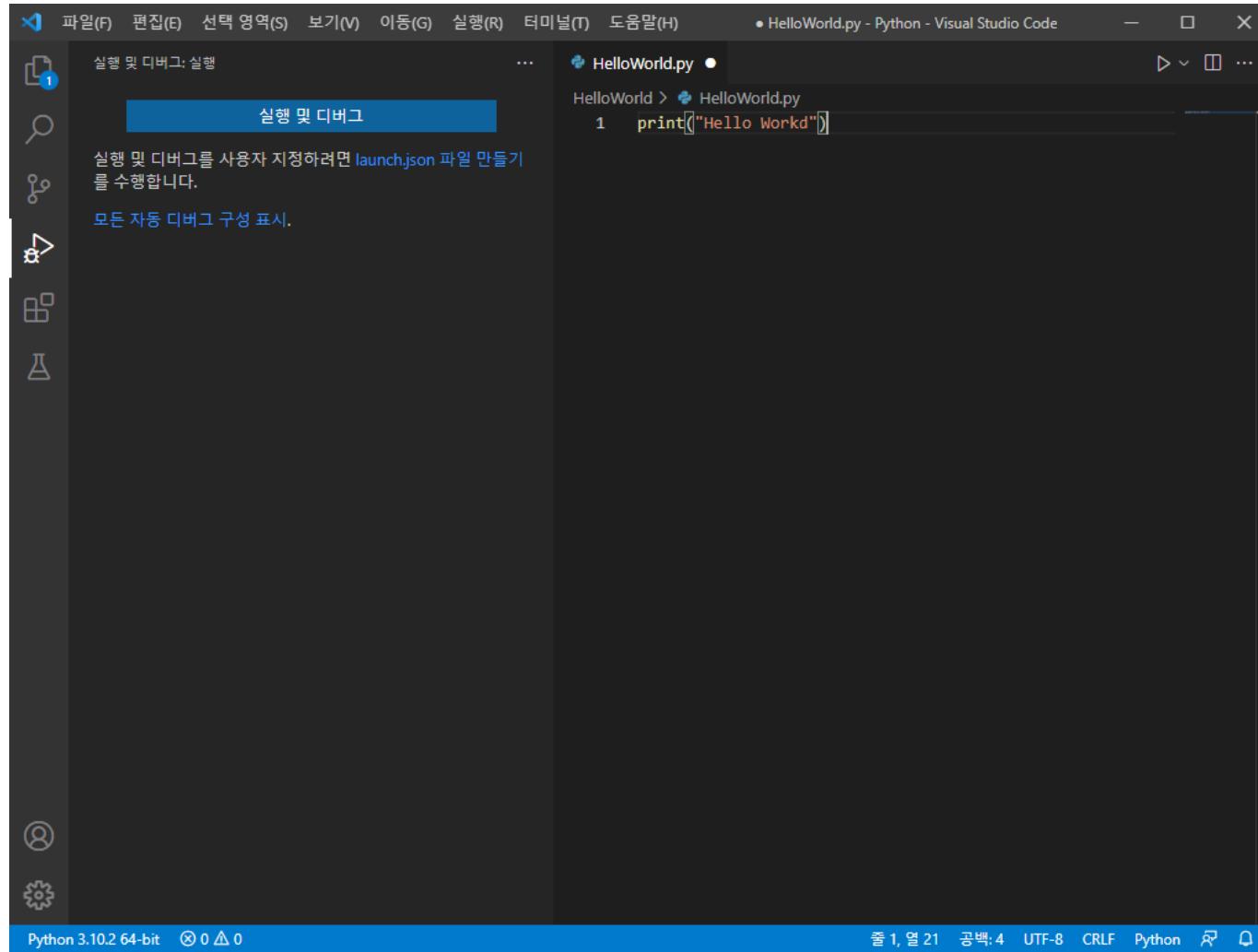
# 파일 작성자 신뢰

## □ 하위 폴더/파일까지 신뢰 선택



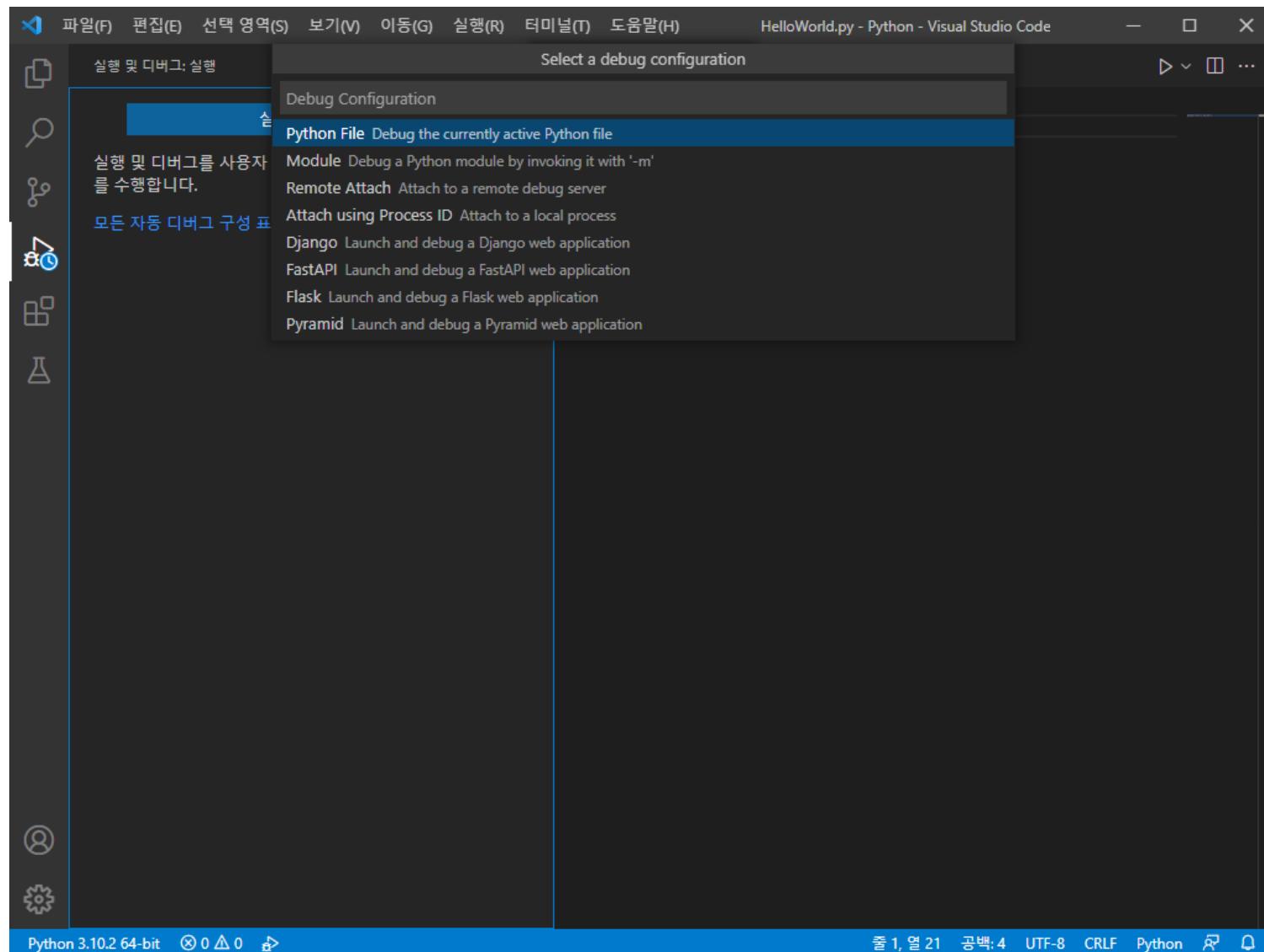
# Hello World!

## □ print("Hello World") 입력 후, “실행 및 디버그” 실행



The screenshot shows the Visual Studio Code interface with the Python extension installed. The left sidebar has icons for file, search, and other development tools. The main editor area displays a single line of code: `print("Hello World")`. The status bar at the bottom indicates "Python 3.10.2 64-bit" and "Python". The title bar shows "HelloWorld.py - Python - Visual Studio Code". A tooltip in the center says: "실행 및 디버그를 사용자 지정하려면 launch.json 파일 만들기 를 수행합니다." (Perform this operation to create a launch.json file and customize execution and debugging). The status bar also shows "줄 1, 열 21" and "UTF-8 CRLF".

# Python 파일 설정



# Hello World 실행 완료

The screenshot shows a Visual Studio Code interface with the following details:

- File Menu:** 파일(F), 편집(E), 선택 영역(S), 보기(V), 이동(G), 실행(R), 터미널(T), 도움말(H)
- Title Bar:** HelloWorld.py - Python - Visual Studio Code
- Left Sidebar:** Includes icons for file operations (New, Open, Save, Find, Replace, Copy, Paste, Delete, Find in Files, Find in Workspace, Find in Project, Find in Editor, Find in Editor (Search), Find in Editor (Replace), Find in Editor (Search and Replace), Find in Editor (Replace All), Find in Editor (Replace All)), and a gear icon.
- Center Area:**
  - Run and Debug View:** Shows "HelloWorld.py" and its code: `print("Hello World")`. A button labeled "실행 및 디버그" (Run and Debug) is highlighted in blue.
  - Message Box:** A message states: "실행 및 디버그를 사용자 지정하려면 launch.json 파일 만들기" (Create a launch.json file to customize execution and debugging) and "모든 자동 디버그 구성 표시" (Show all automatic debug configurations).
- Bottom Bar:** Includes tabs for 문제 (Issues), 출력 (Output), 디버그 콘솔 (Debug Console), and 터미널 (Terminal). The 터미널 tab is selected.
- Terminal View:** Displays a Windows PowerShell session:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/powershell

PS C:\Users\devops\Python> & 'C:\Program Files\Python310\python.exe' 'c:\Users\devops\.vscode\extensions\ms-python.python-2022.0.1814523869\pythonFiles\lib\python\debugpy\launcher' '50395' '--' 'c:\Users\devops\Python\HelloWorld\HelloWorld.py'
Hello World
PS C:\Users\devops\Python>
```
- Status Bar:** Shows Python 3.10.2 64-bit, 총 1, 열 18, 공백 4, UTF-8, CRLF, Python, and icons for file operations.

## 2      Git

---

# 도구 소개

---

## □ 도구 개요

- 버전 관리의 대표 도구

## □ 목적 : 커뮤니케이션

- 표준: 산출물의 무결성 확보
- 소스코드 버전 관리
  - 마음의 평화: 실수해도 이전 버전으로 빠르게 되돌리기
- 왜 이렇게 수정하였지 확인
- 과거의 내가 얼마나 멋진 개발자였는지 확인

## □ 일반적인 사용 방법

- 별도의 저장소 운영 서버 사용
  - Git: GitHub, Bitbucket, Gitlab, **Giblit**
  - 통합 저장소: SCMManger
- 클라이언트는 Core나 별도의 확장 도구 사용
  - Eclipse 플러그인: 기본 포함
  - Tortoise 시리즈
  - Sourcetree

# Git 개요

---

## □ 역사

- 2005년 리누스 토발즈를 중심으로 개발
- 현재 GitHub에서 운영

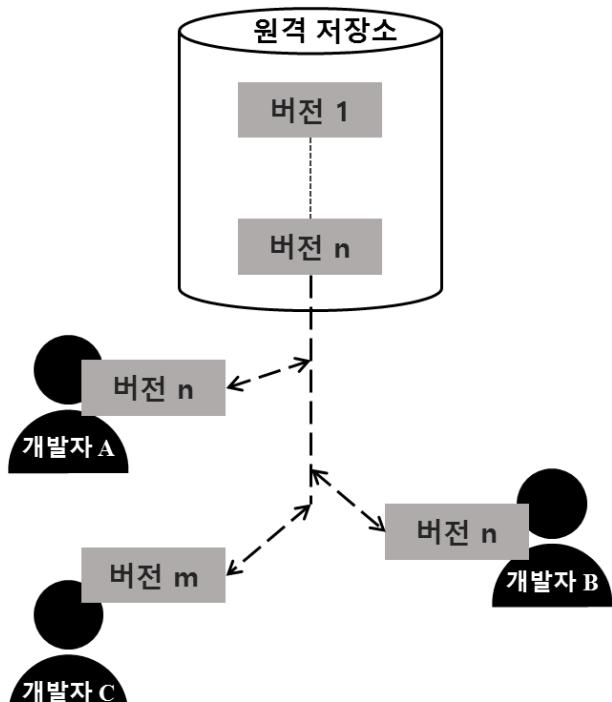
## □ 사용 추세

- 리눅스, Git을 포함하여 주요 오픈소스는 Git(GitHub) 사용
- 회사의 신규 프로젝트는 Git을 많이 사용하는 추세

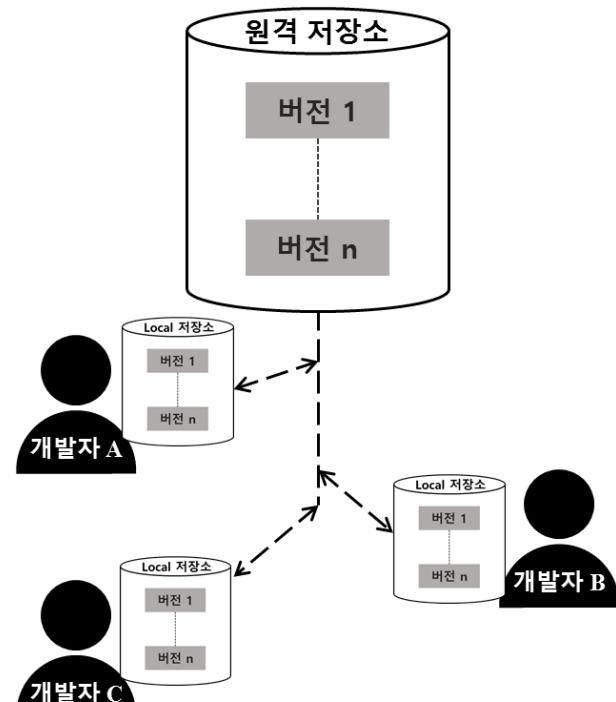
# 버전 관리 방식

## □ 분산 시스템 기반

- 원격 저장소와 동일한 로컬 저장소를 각 개발자가 유지
  - SVN은 최종 리비전만 각 개발자가 유지



Subversion



Git

# 기본 용어

---

영문	설명
Init	기존 폴더에 저장소를 생성
Clone	원격 저장소를 내 로컬에 복사
Push	내 저장소의 변경사항을 원격 저장소에 반영
Pull	원격 저장소의 변경사항을 내 저장소에 반영
Add	수정된 파일을 Staging Area에 등록하여 Staged 상태 만들
Check-out	작업할 브랜치를 지정
Commit	Staged 상태의 파일을 로컬 저장소에 반영
Remote Repository	공유를 위해 사용하는 저장소, 서버의 역할

# Git의 Branch

---

## □ 브랜치(Branch)

- 원본의 복사본
- 대부분의 VCS(Version Control System)에서 지원
- 용도
  - 특정 기능을 원본에 영향을 주지 않고 개발/테스트 하기 위해
  - 베이스라인을 설정하기 위해

## □ Subversion 브랜치 사용의 어려움

- 브랜치 생성 시, 원본을 특정 폴더에 복사함 (용량 \* 2)
  - 예) Visual C++의 3기가 소스코드의 단계 별/릴리즈 별 브랜치 10회 = 3기가 \* 10

## □ Git 브랜치의 용이함

- Git은 변경 사항을 저장하지 않고, 변경을 스냅샷 포인터로 관리
- 서버 자체를 복제하기 때문에, 포인터를 이용한 브랜치 생성 및 이동 가능
  - 예) Visual C++의 3기가 소스코드의 단계 별/릴리즈 별 브랜치 10회 = 40byte \* 10

## □ Git를 보다 잘 사용하고 싶다면, 브랜치를 어떻게 사용하는가에 좌우

## 브랜치 전략

---

### □ 업계에서 3개의 브랜치 전략을 추천

1. Subversion 처럼 사용하는 방식
2. GitFlow를 따르는 방식
3. Pull Request(GitHubFlow)를 따르는 방식

## 1) Subversion 처럼 사용하는 방식

---

### □ 방법

- Master 브랜치(기본 브랜치)를 공유하는 방법

### □ 장점

- Subversion 사용자가 쉽게 Git에 적용 가능

### □ 단점

- 브랜치를 사용하지 않음

### □ 추천하는 조직

- 기존 Subversion 등 중앙 집중식 도구에서 Git으로 넘어오는 조직
- 급격한 변화가 어려운 조직

## 2) GitFlow를 따르는 방식

---

### □ 방법

- GitFlow 도구를 설치하고, GitFlow의 프로세스를 준수
  - Git의 확장으로, Git의 브랜치 사용을 쉽게 적용할 수 있도록 확장 도구로 제공

### □ 장점

- 정형화된 브랜치 전략의 사용
- 참고할 수 있는 자료가 많음

### □ 단점

- 임베디드 SW보다 릴리즈가 잦은 웹 서비스 분야에 보다 적합
  - 임베디드 SW에도 적용이 안되는 것은 아님
  - K사의 사례

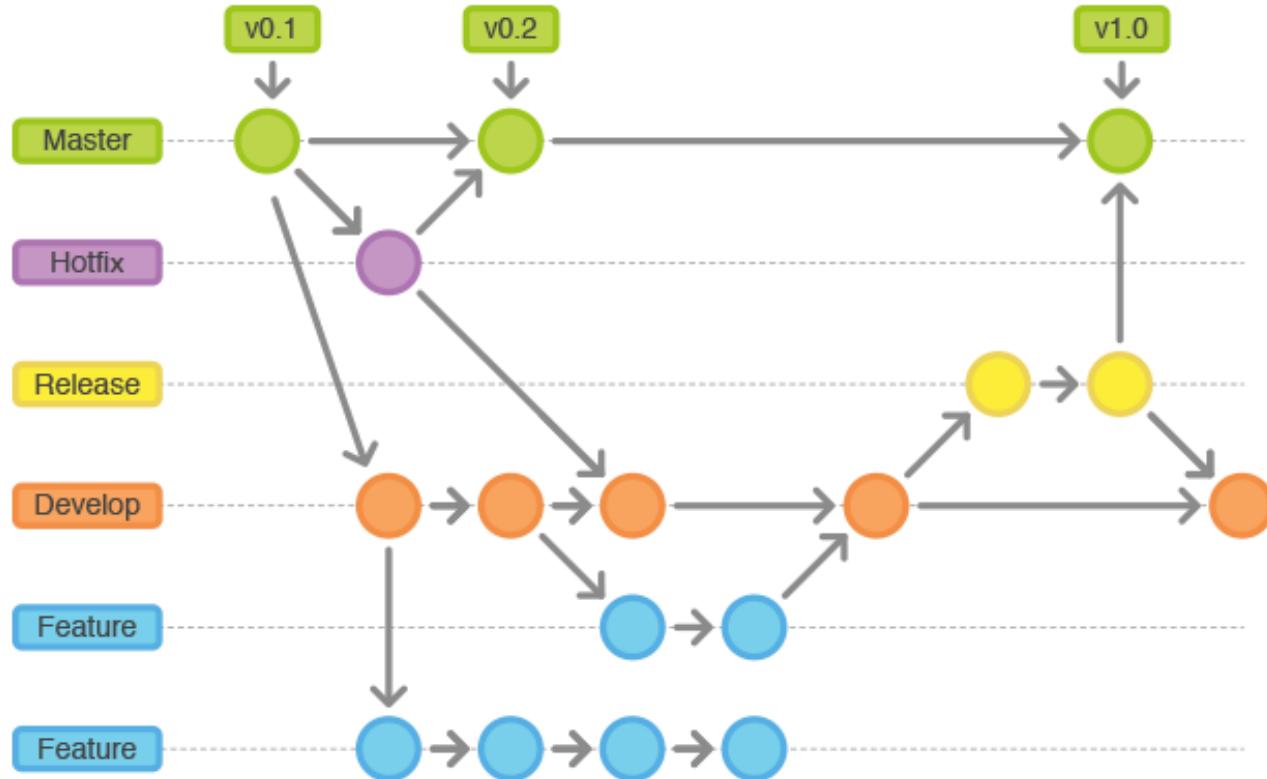
### □ 추천하는 조직

- 브랜치를 처음 적용하는 조직
- 중규모(협업 인원이 20명) 이하인 경우

## 2) GitFlow를 따르는 방식 - GitFlow

### ❑ Vincent Driessen가 개발한 브랜치 전략 및 지원 도구

- Sourcetree와 같은 Git 도구에서 기본 지원
- Master: 운영 환경과 동일한 소스코드가 있는 브랜치



### 3) Pull Request(GitHubFlow)를 따르는 방식

---

#### □ 방법

- 개발자가 저장소를 Fork(복제) 또는 Branch하여 수정하고, 변경 사항의 적용을 요청(Pull Request)
  - GitHub에서 오픈소스에 Contribute 하는 대표적인 방식

#### □ 장점

- 관리자의 검토 후, 개발자의 소스코드 반영 여부를 결정 가능

#### □ 단점

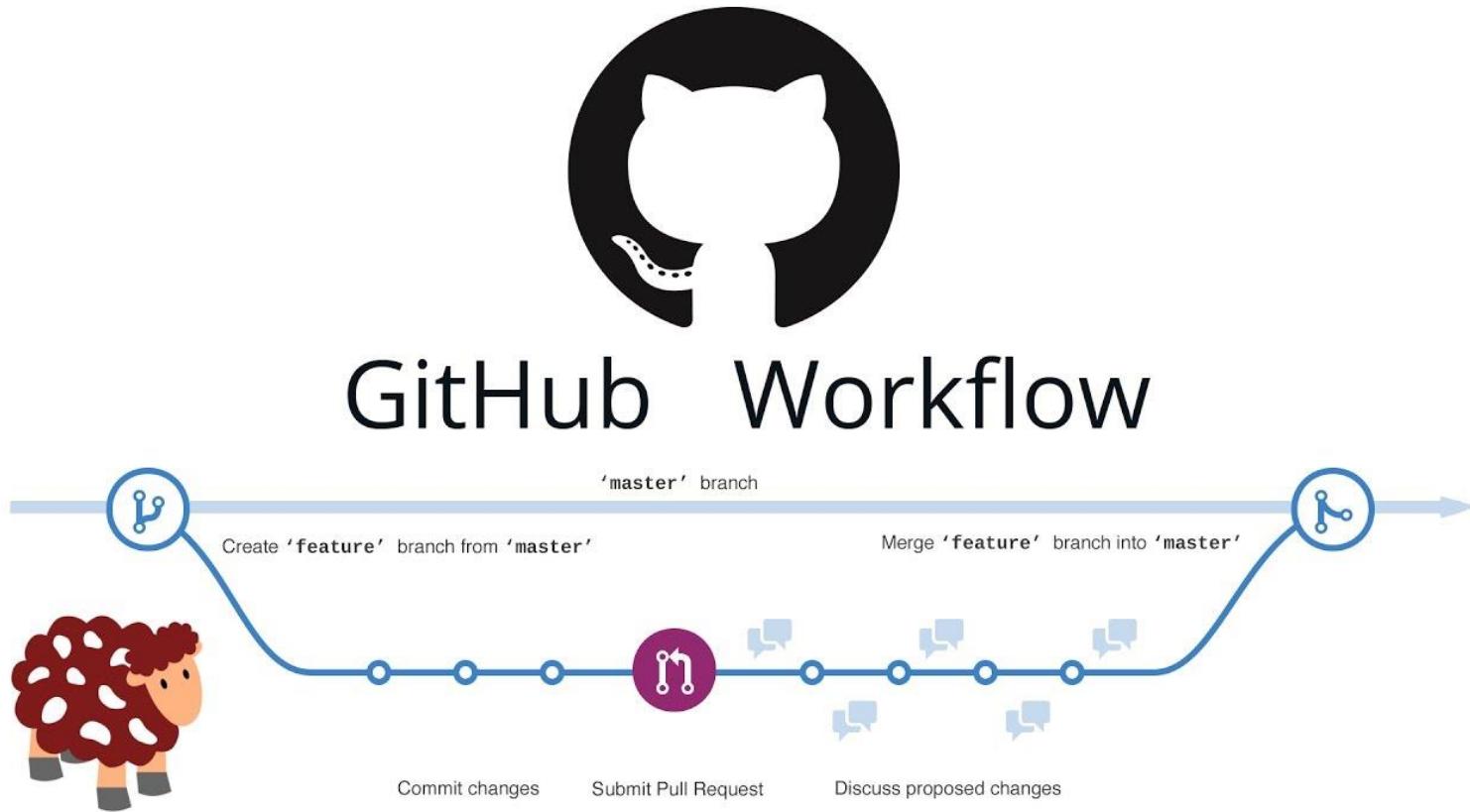
- Pull Request를 처리하기 위한 관리자 필요
- 원활한 사용을 위해서는 Pull Request를 지원하는 서버 도구 사용 필요

#### □ 추천하는 조직

- 협력 업체와 동일한 저장소로 공유하는 경우
- 대규모 개발인 경우

### 3) Pull Request(GitHubFlow)를 따르는 방식

- GitHub에서 운영되는 대부분의 오픈소스 개발에 적용



# [서버 설정] GitHub을 원격 저장소로 이용하기

## □ <https://github.com/>

- 회원가입하면, public 저장소 무제한 생성 가능
  - 다른 사람에게도 소스코드가 보이지만, 커밋하는 사람을 지정할 수 있음

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \* Repository name \*

 DongJoonHan /

Great repository names are short and memorable. Need inspiration? How about [didactic-octo-happiness?](#)

Description (optional)

 Public Anyone on the internet can see this repository. You choose who can commit.

 Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license A license tells others what they can and can't do with your code. [Learn more.](#)

[Create repository](#)

# [저장소 생성] 실습을 위한 저장소 생성

## □ 저장소 명과 “README” 파일 추가

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner \*  DongJoonHan \*

Repository name \*  

Great repository names are short and memorable. Need inspiration? How about [didactic-octo-happiness](#)?

Description (optional)

 Public  
Anyone on the internet can see this repository. You choose who can commit.

 Private  
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file  
This is where you can write a long description for your project. [Learn more](#).

Add .gitignore  
Choose which files not to track from a list of templates. [Learn more](#).

Choose a license  
A license tells others what they can and can't do with your code. [Learn more](#).

This will set  main as the default branch. Change the default name in your [settings](#).

**Create repository**

# 저장소 생성 확인

The screenshot shows a GitHub repository page for 'DongJoonHan / DevOps\_Test'. The repository is public and contains one branch ('main') and one commit ('Initial commit'). The commit was made by 'DongJoonHan' and has a timestamp of 'now'. The commit message is 'Initial commit' and it is associated with the file 'README.md'. The repository has 0 tags. On the right side, there is an 'About' section with a note: 'No description, website, or topics provided.' It also lists 'Readme', '0 stars', '1 watching', and '0 forks'. Below that is a 'Releases' section stating 'No releases published' and 'Create a new release'. At the bottom is a 'Packages' section stating 'No packages published' and 'Publish your first package'.

DongJoonHan / DevOps\_Test (Public)

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

DongJoonHan Initial commit 920876c now 1 commit

README.md Initial commit now

README.md

**DevOps\_Test**

About

No description, website, or topics provided.

Readme 0 stars 1 watching 0 forks

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

# Git 설치

## □ Git 웹사이트에서 최신 버전 다운로드

The screenshot shows the official Git website at [git-scm.com](https://git-scm.com). The header features the Git logo and the tagline "git --distributed-even-if-your-workflow-isn't". A search bar is located in the top right corner. Below the header, there are two main sections of text. The first section describes Git as a free and open source distributed version control system designed for efficiency. The second section highlights its ease of learning and superior performance compared to other SCM tools like Subversion, CVS, Perforce, and ClearCase, mentioning features such as cheap local branching, convenient staging areas, and multiple workflows. To the right of the text, there is a 3D diagram illustrating the distributed nature of Git, showing multiple repositories connected by bidirectional arrows. At the bottom of the page, there are four main navigation links: "About", "Documentation", "Downloads", and "Community", each accompanied by a circular icon. On the right side, a large monitor displays the "Latest source Release" information, specifically version 2.35.1 from January 29, 2022, with a red box highlighting the "Download for Windows" button.

**About**  
The advantages of Git compared to other source control systems.

**Documentation**  
Command reference pages, Pro Git book content, videos and other material.

**Downloads**  
GUI clients and binary releases for all major platforms.

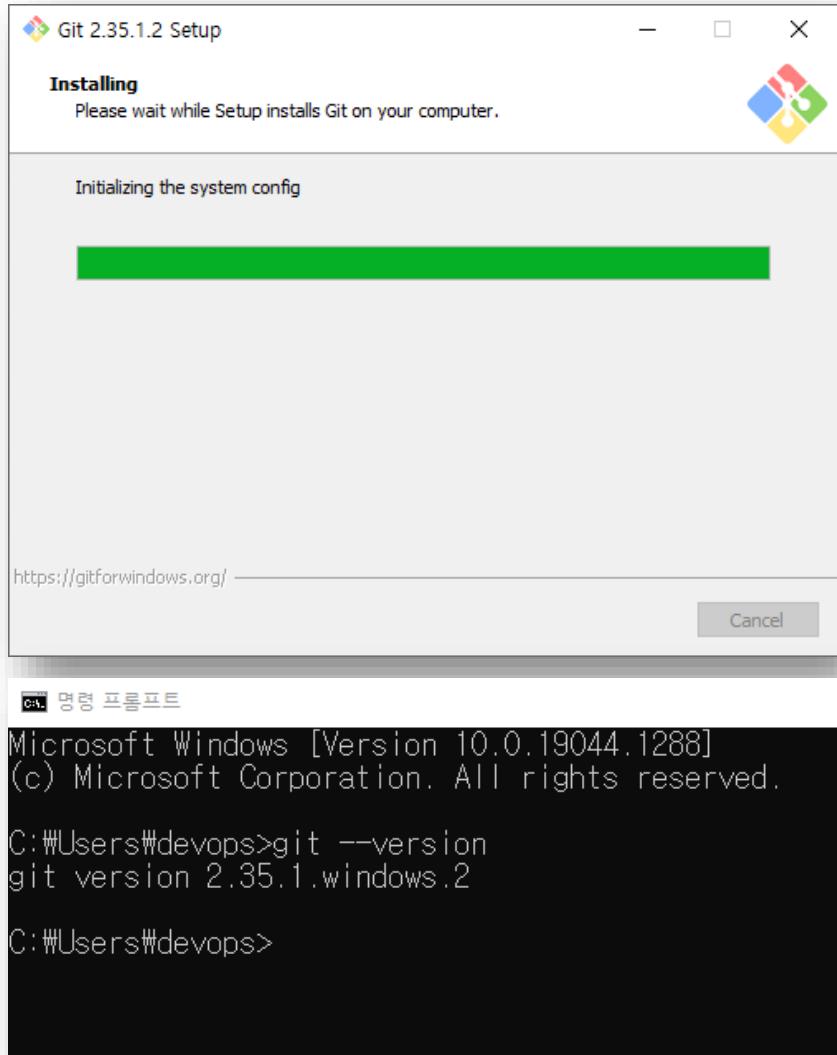
**Community**  
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release  
**2.35.1**  
[Release Notes \(2022-01-29\)](#)

[Download for Windows](#)

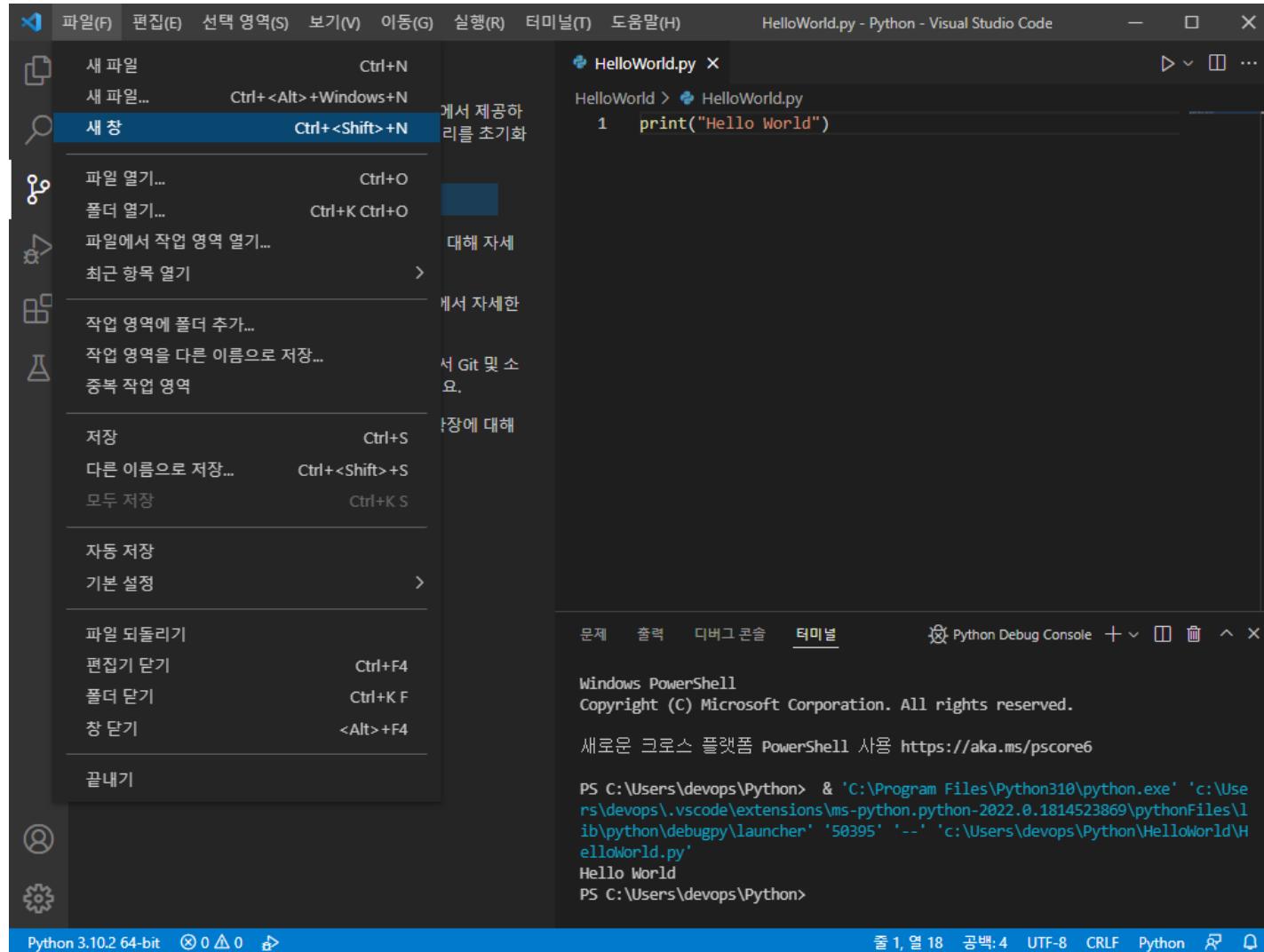
# Git 설치

## □ 기본 설정으로 설치 후 확인

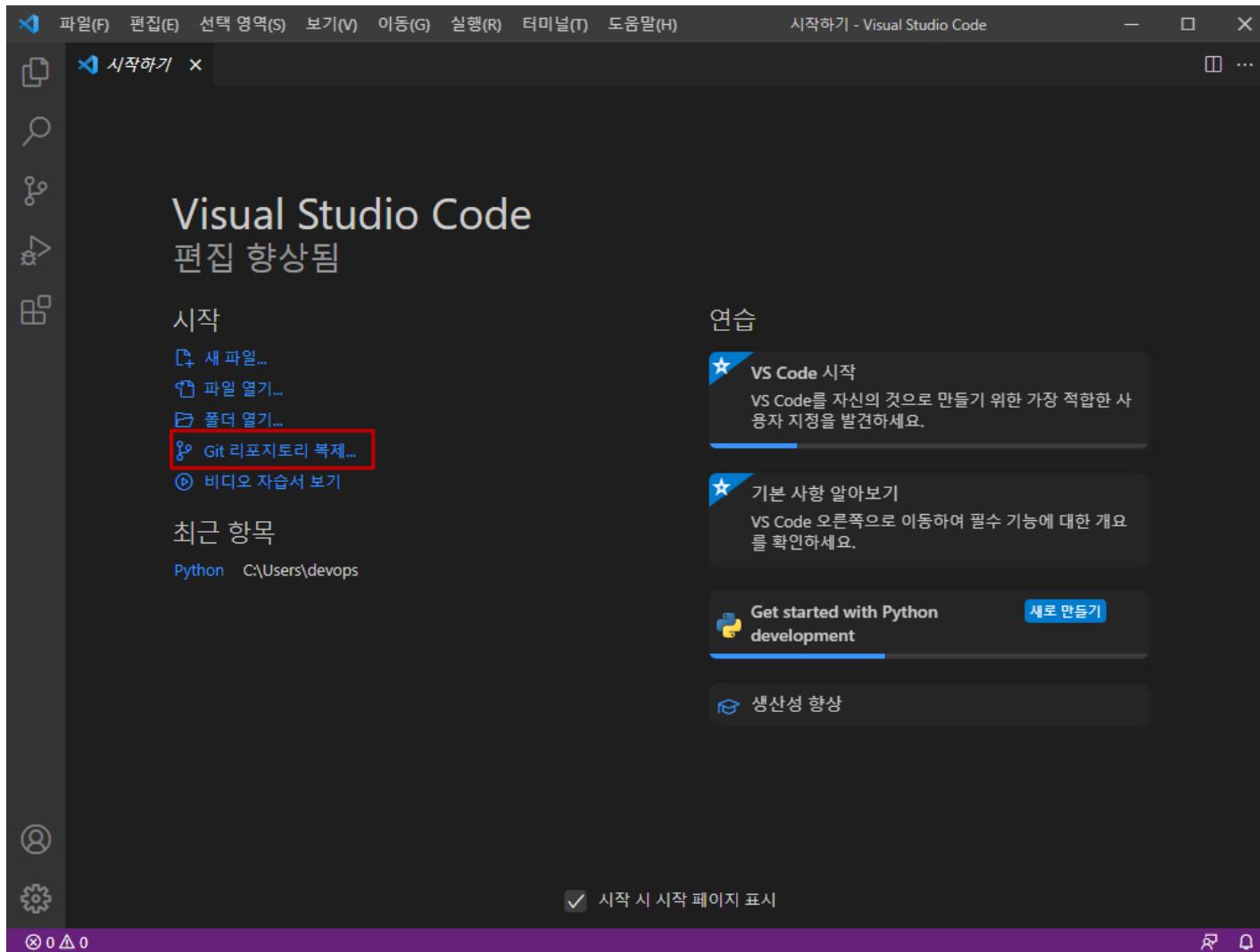


# VSCode 에서 Clone

## □ 파일 – 새창으로 VSCode 새로 열기

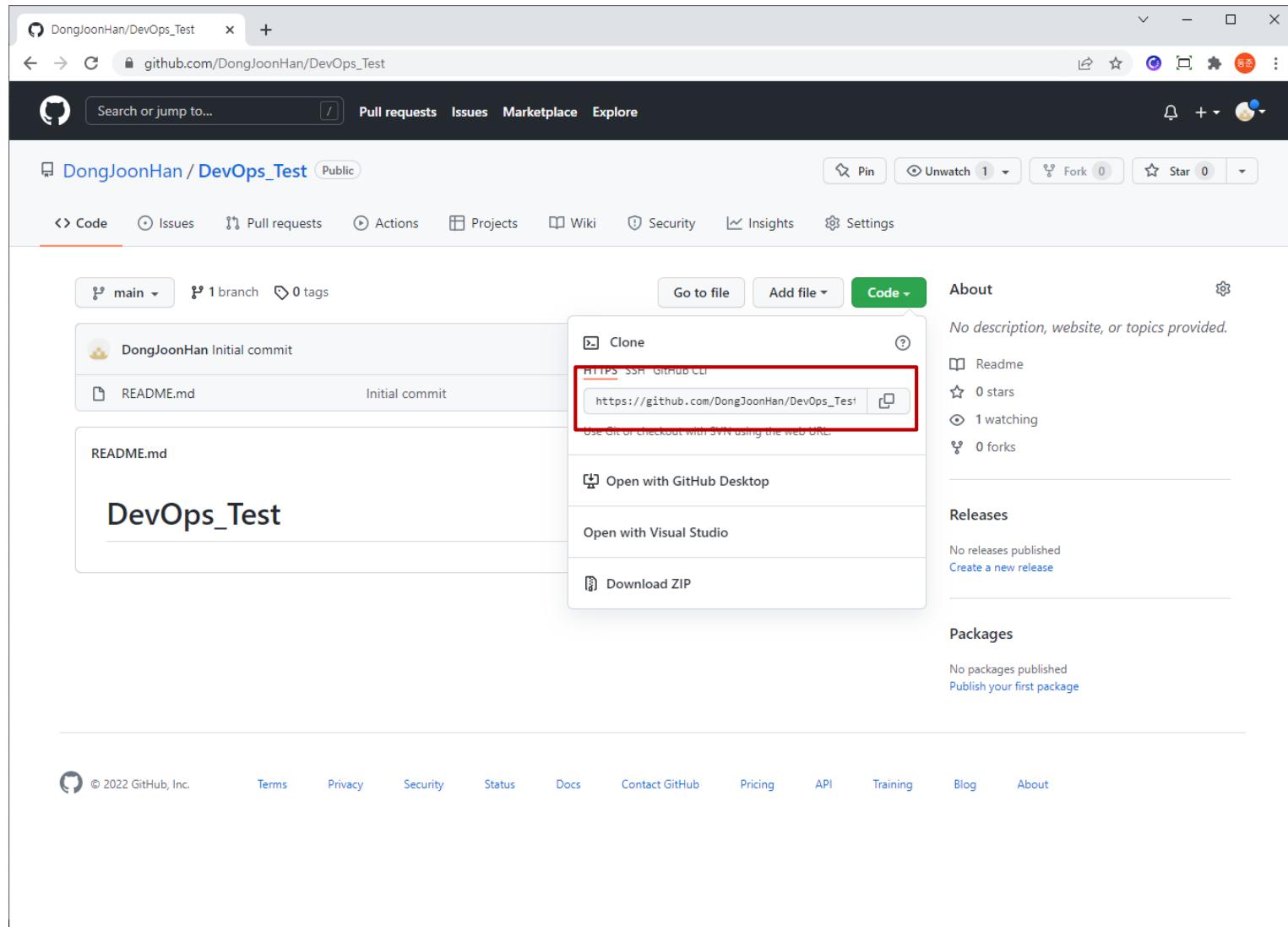


# 리포지토리 복제(Clone)



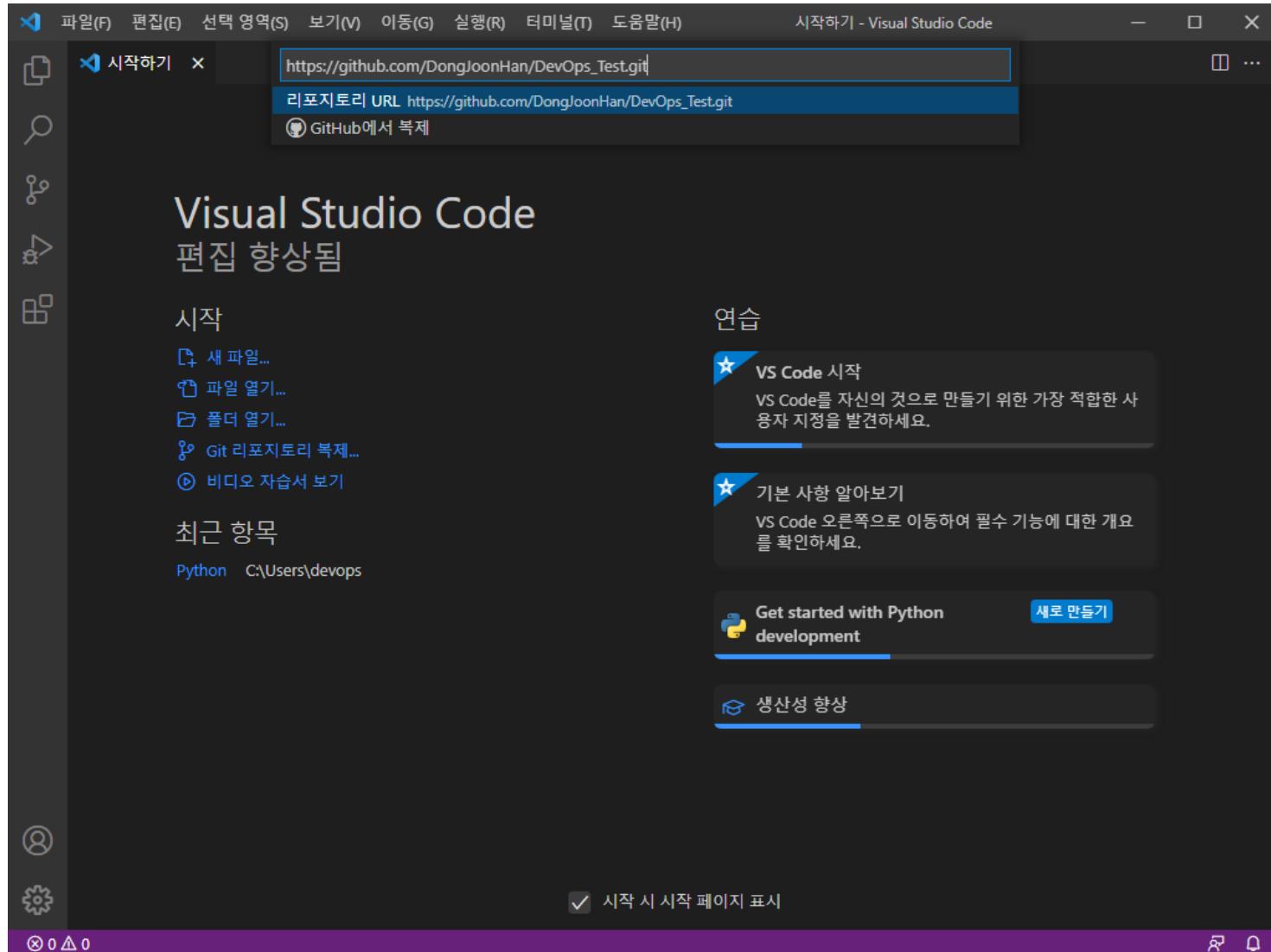
# 복제할 리포지토리 정보 확인

## 저장소의 초록색 Clone 버튼

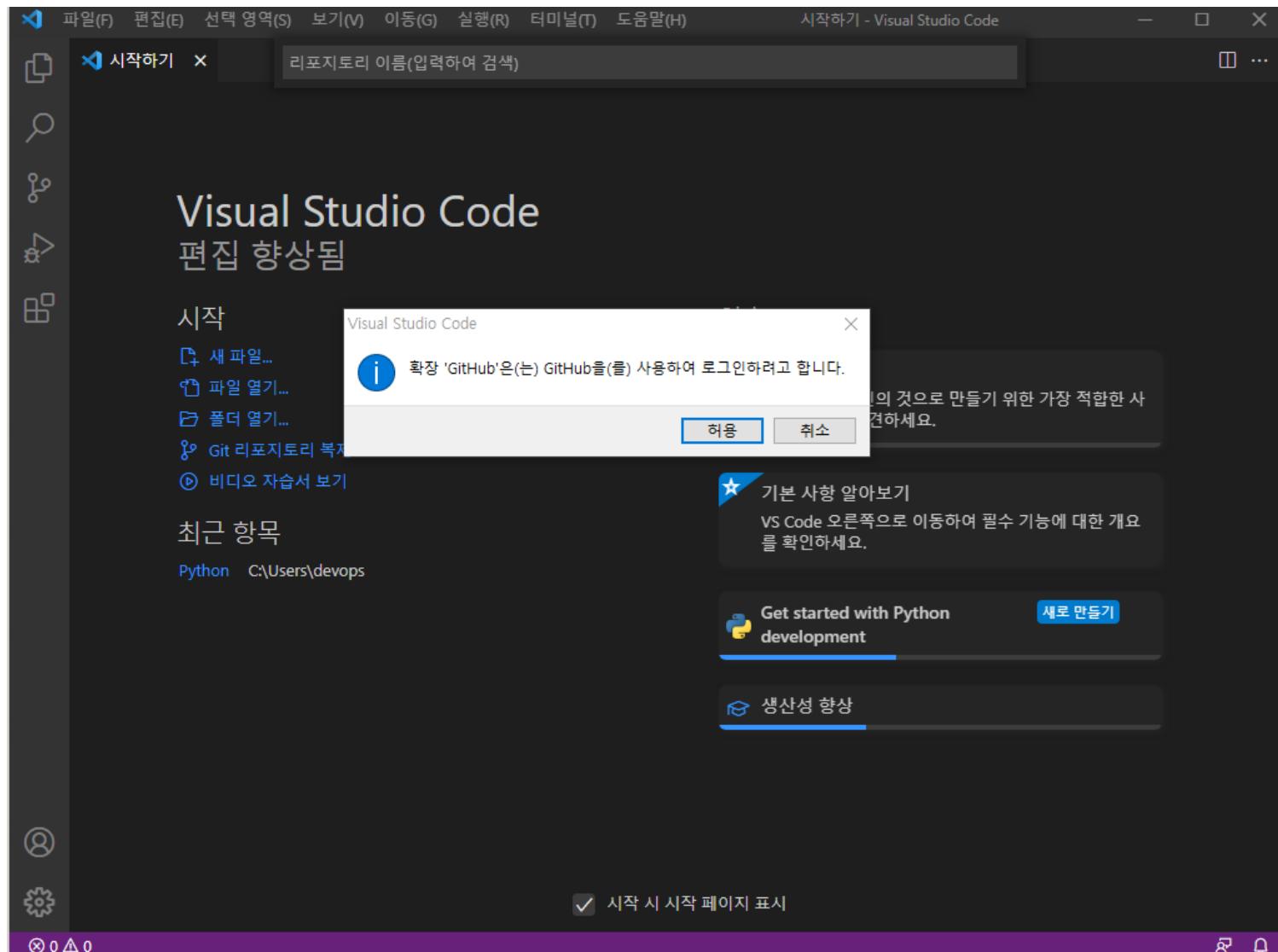


# 리포지토리 주소 입력

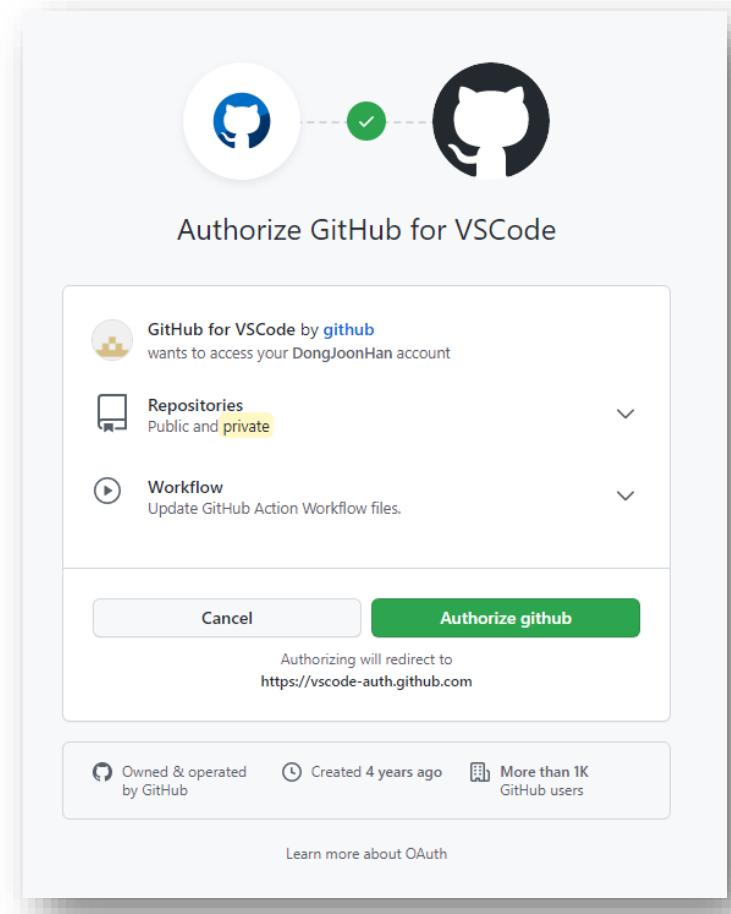
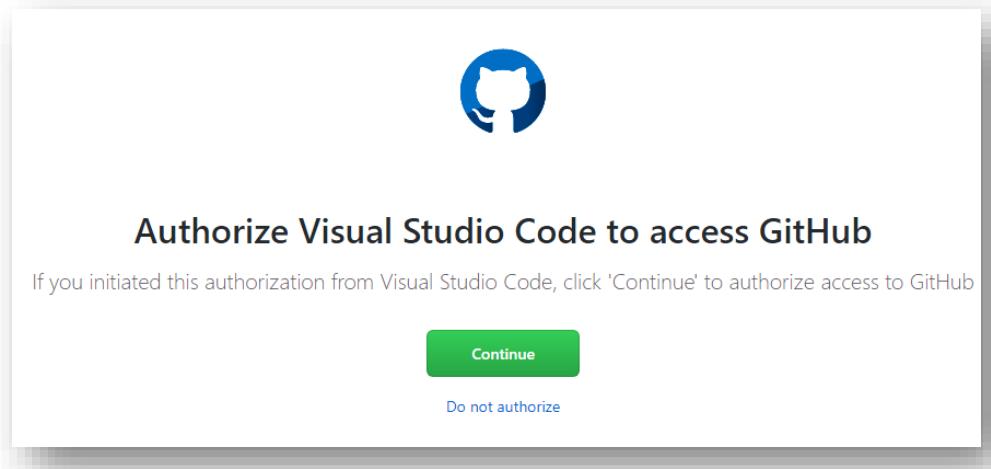
## □ 주소 입력 후 "GitHub에서 복제 선택"



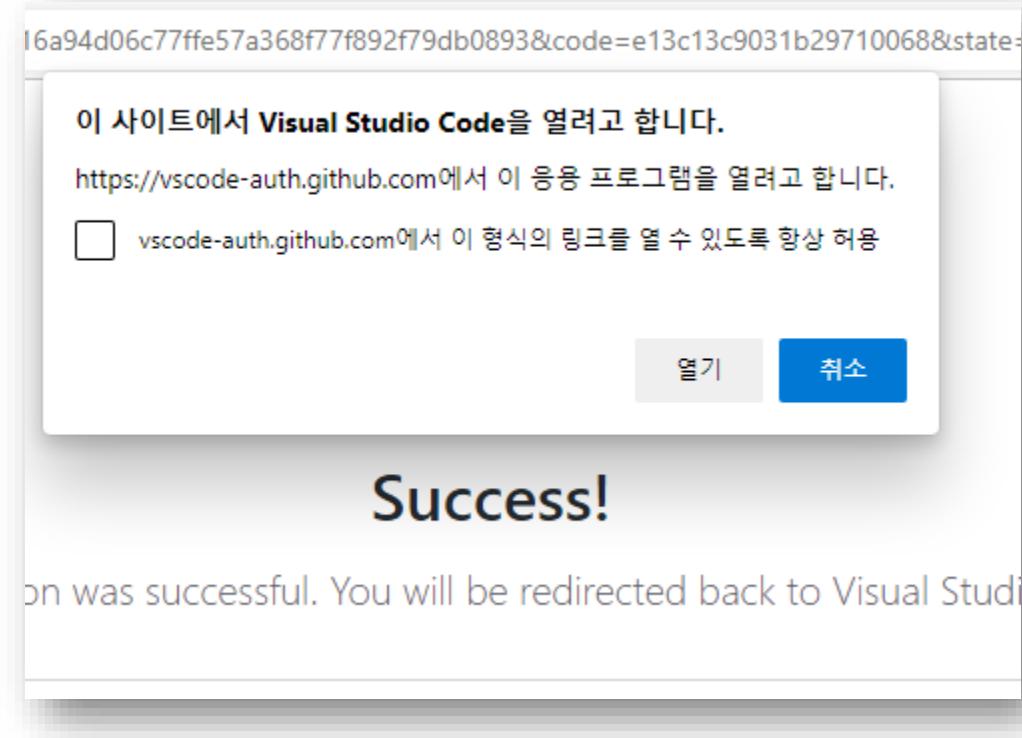
# GitHub 로그인 처리



# GitHub 로그인 처리

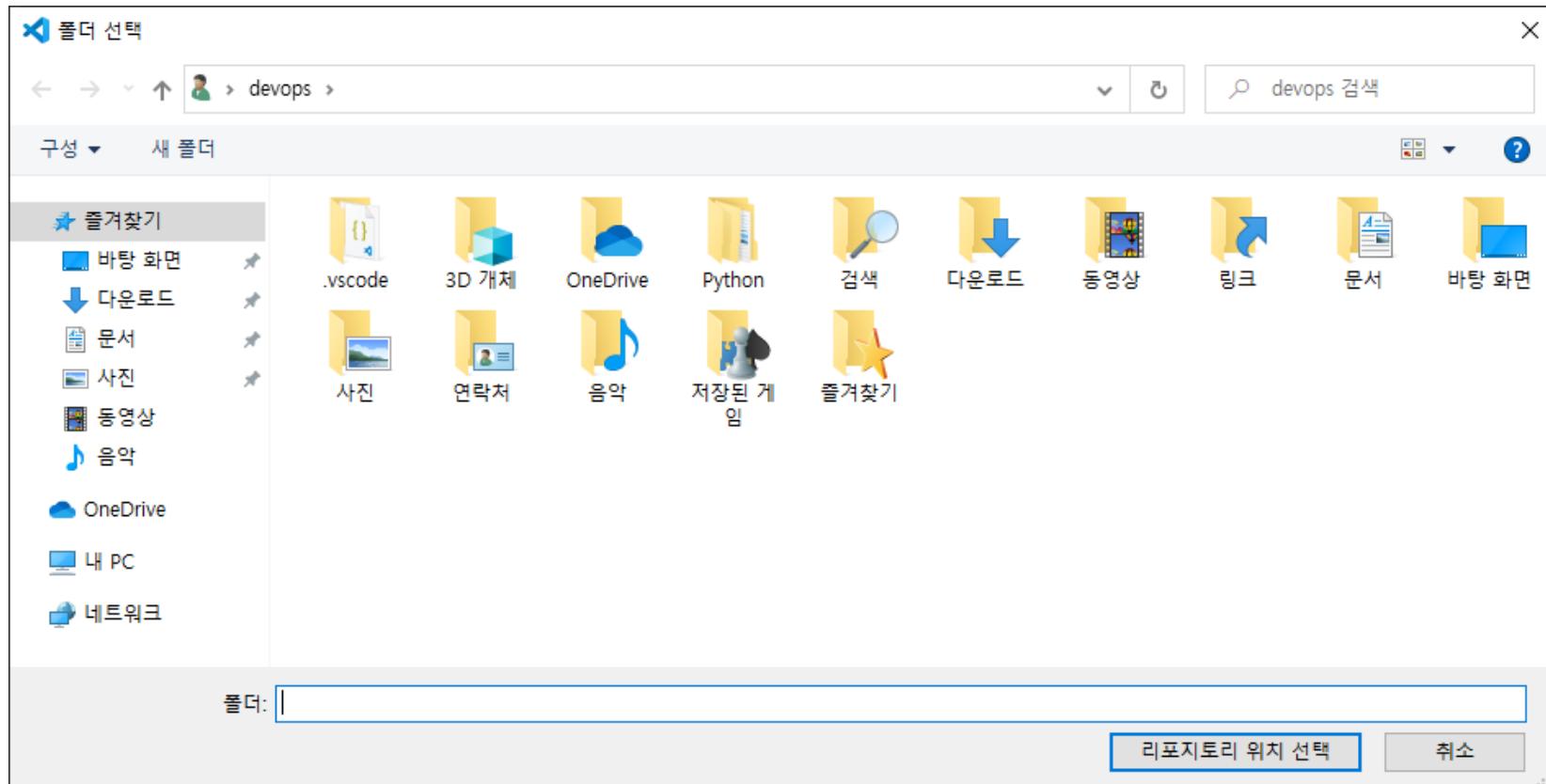


## GitHub 로그인 처리



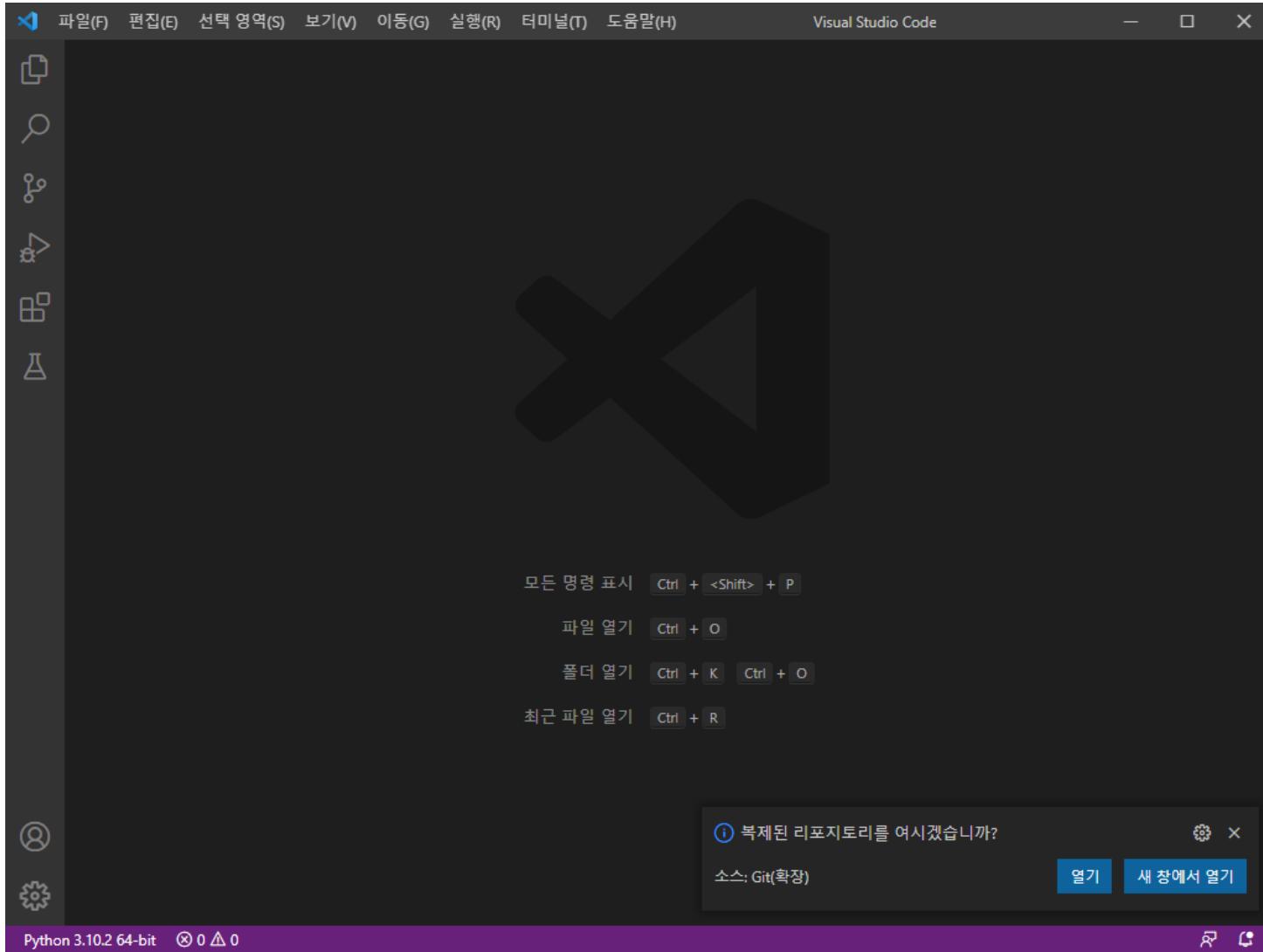
# 리포지토리 복제 시도

- 다시 한 번 복제를 클릭하면 폴더 선택 창 출력
- 기본 폴더 선택



# 복제된 리포지토리 열기

## □ 우측 하단의 "열기" 클릭



# 리포지토리 복제 확인

## □ GitHub에서 생성한 저장소가 복제 되었는지 확인

The screenshot shows a dark-themed instance of Visual Studio Code. The left sidebar displays a file tree with a folder named 'DEVOPS\_TEST' containing a file named 'README.md'. The main editor area shows the content of 'README.md': '# DevOps\_Test'. The status bar at the bottom indicates the file is 'main' and has 0 changes. The bottom right corner shows standard file status icons.

```
1 # DevOps_Test
```

# README 최초 수정하기

## □ 수정 후 저장하면, 소스제어에 <1> 표시

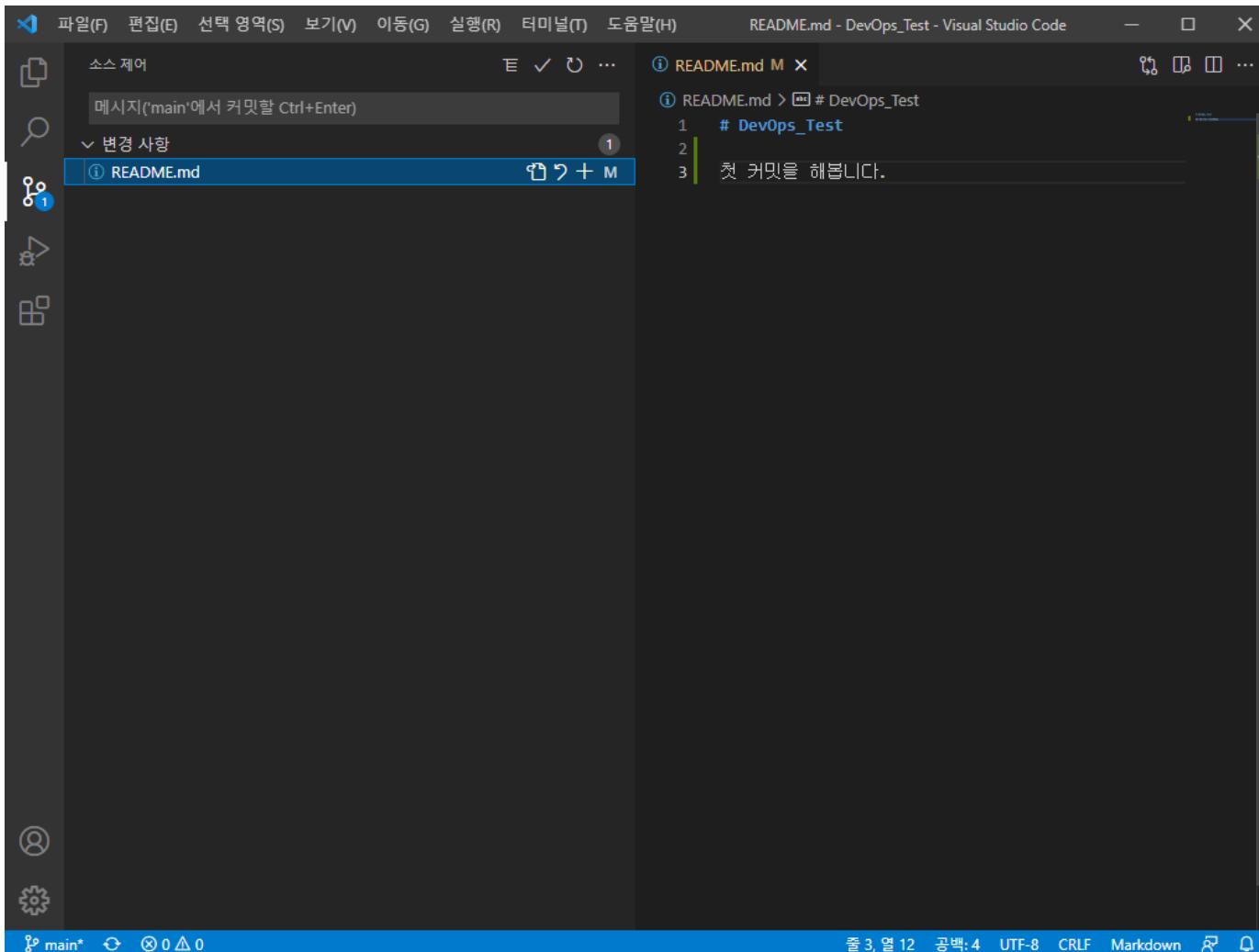
The screenshot shows the Visual Studio Code interface with a dark theme. On the left is the Explorer sidebar, which displays a folder named 'DEVOPS\_TEST' containing a file 'README.md'. A small blue badge with the number '1' is visible next to the folder icon. The main editor area shows the content of 'README.md':

```
① README.md M X
① README.md > ② # DevOps_Test
1 # DevOps_Test
2
3 첫 커밋을 해봅니다.
```

The status bar at the bottom indicates the file is 'main\*' and shows other details like encoding (UTF-8), line endings (CRLF), and the current line (줄 3, 열 12). The status bar also includes icons for saving, closing, and other file operations.

# 변경 파일 Add

□ + 버튼을 눌러 변경파일 Add (메뉴명은 스테이징)



# 스테이징 변경 사항 커밋

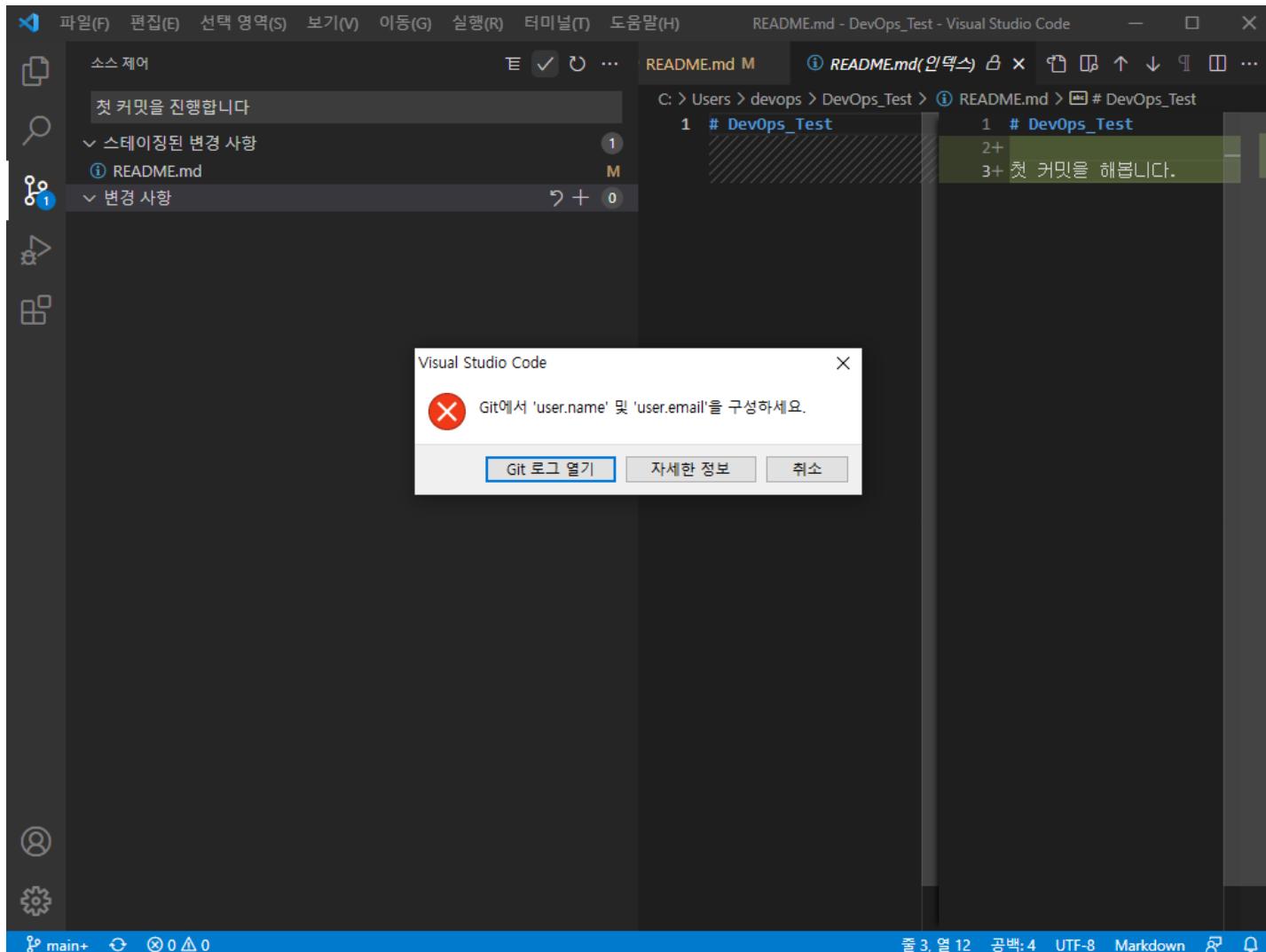
- 커밋은 변경 지점을 만드는 것
- 커밋 메시지 작성 후, V 버튼 클릭

The screenshot shows the Visual Studio Code interface with a dark theme. A terminal window is open, titled 'README.md - DevOps\_Test - Visual Studio Code'. The command line shows the path: 'C:\Users\devops\DevOps\_Test> README.md > # DevOps\_Test'. The user has typed the first line of a commit message: '1 # DevOps\_Test'. Below it, the message body starts with '2+ 첫 커밋을 해봅니다.' (First commit). The status bar at the bottom indicates the file is main+, has 0 changes, and is in Markdown mode.

```
1 # DevOps_Test
2+
3+ 첫 커밋을 해봅니다.
```

# 커밋을 위한 필수 정보 입력 (최초 1회)

## □ E-mail과 작성자 이름



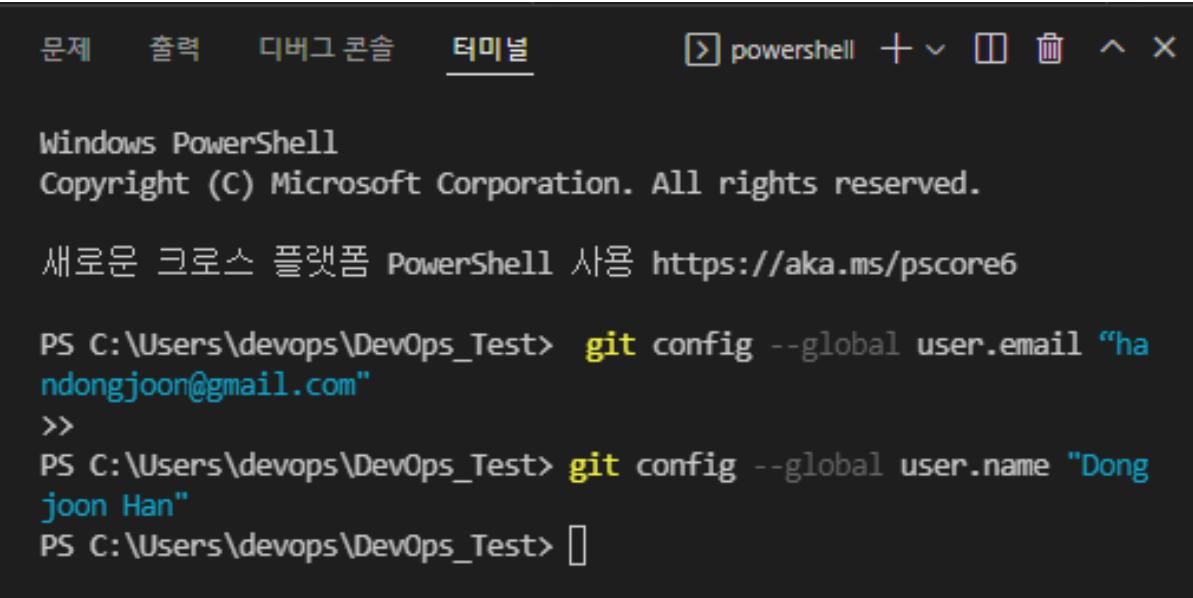
# 필수 정보 입력

## □ Git Log 확인

```
git config --global user.email "you@example.com"  
git config --global user.name "Your Name"
```

## □ 본인 정보 터미널에 입력 (한 줄씩)

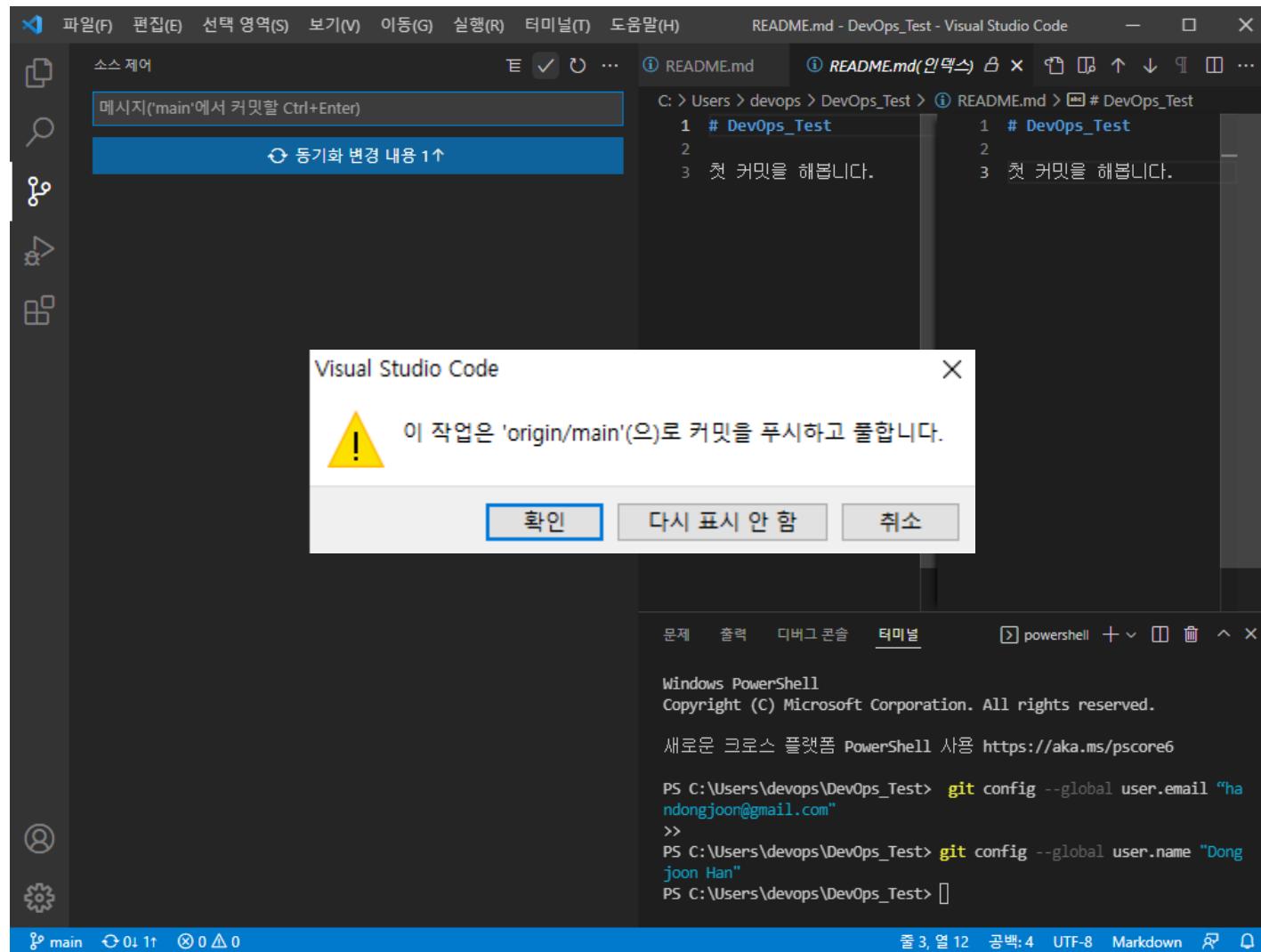
```
git config --global user.email "handongjoon@gmail.com"  
git config --global user.name "Dongjoon Han"
```



The screenshot shows a Windows PowerShell window titled 'powershell'. The title bar also includes tabs for '문제', '출력', '디버그 콘솔', and '터미널'. The main area displays the following text:

```
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6  
  
PS C:\Users\devops\DevOps_Test> git config --global user.email "ha  
ndongjoon@gmail.com"  
>>  
PS C:\Users\devops\DevOps_Test> git config --global user.name "Dong  
joon Han"  
PS C:\Users\devops\DevOps_Test> []
```

# 커밋 다시 푸시 시도



# GitHub에서 반영 확인

The screenshot shows a GitHub repository page for 'DongJoonHan / DevOps\_Test'. The repository is public and contains 1 branch and 0 tags. The main file, README.md, has two commits from 'DongJoonHan' with the message '첫 커밋을 진행합니다'. The latest commit was made 3 minutes ago by 'aa1ff5f'. The repository has 0 stars, 1 watching, and 0 forks. The 'About' section notes 'No description, website, or topics provided.' The 'Releases' section indicates 'No releases published' and 'Create a new release'. The 'Packages' section shows 'No packages published' and 'Publish your first package'. The 'Code' tab is selected, showing the file content:

```
DevOps_Test
첫 커밋을 해봅니다.
```

# 3 Jenkins

---

## 지속 통합(CI: Continuous Integration)

---

- ⑤ 팀 구성원들이 자신이 한 일을 **자주 통합**하는 소프트웨어 개발 실천 방법
- ⑥ 각 통합은 자동화된 빌드를 검증하여 **최대한 빨리 통합 오류**를 탐지
- ⑦ 하루에 여러 번 통합 빌드를 수행하는 것

- 마틴 파울러

소프트웨어 **통합 오류를 개발 초기부터 예방**하는 것

소프트웨어 통합을 위해 현존하는 가장 훌륭한 전략

---

통합 지옥

Integration HELL



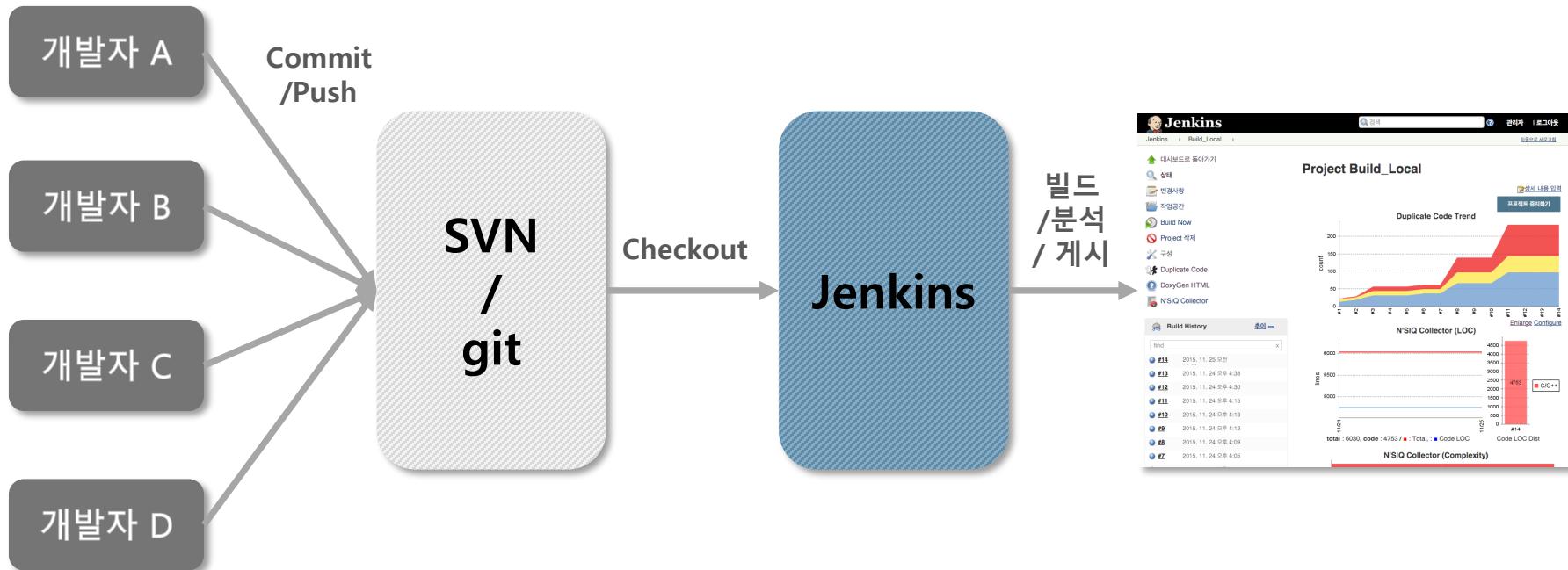
# Jenkins

<http://www.jenkins-ci.org>

- ⑤ 웹 기반 오픈소스 CI 도구
- ⑤ 점유율 약 70%
- ⑤ 1,300 개의 플러그인

# 지속적 통합 - 동작 방식

- ① 버전관리 도구에서 최신 리비전을 체크아웃 받아
- ② 주어진 명령대로 빌드하여
- ③ 결과를 게시/전달함



# 도구 모음

Jenkins      검색      관리자 | 로그아웃

Jenkins > Build\_Local >

[자동으로 새로고침](#)

[대시보드로 돌아가기](#)

[상태](#)

[변경사항](#)

[작업공간](#)

[Build Now](#)

[Project 삭제](#)

[구성](#)

[Duplicate Code](#)

[Doxygen HTML](#)

[N'SIQ Collector](#)

## Project Build\_Local

[상세 내용 입력](#)

[프로젝트 증지하기](#)

### Duplicate Code Trend

count

#1 #2 #3 #4 #5 #6 #7 #8 #9 #10 #11 #12 #13 #14

[Enlarge](#) [Configure](#)

### N'SIQ Collector (LOC)

lines

total : 6030, code : 4753 / ■ : Total, □ : Code LOC

11/24 11/25

4753

C/C++

Code LOC Dist

### N'SIQ Collector (Complexity)

Code LOC Dist

Build History

추이

find

번호	날짜
#14	2015. 11. 25 오전
#13	2015. 11. 24 오후 4:38
#12	2015. 11. 24 오후 4:30
#11	2015. 11. 24 오후 4:15
#10	2015. 11. 24 오후 4:13
#9	2015. 11. 24 오후 4:12
#8	2015. 11. 24 오후 4:09
#7	2015. 11. 24 오후 4:05

## 설치 사전 조건 – Java 설치

### □ 1안) <https://www.java.com/ko/>에서 Java 다운로드 후 설치

- 회사 정책에 따라, OpenJDK를 설치해야 할 수 있음

### □ 2안) <https://github.com/ojdkbuild/ojdkbuild>에서 OpenJDK 다운로드 후 설치

- Java-11 버전 다운로드 후 적절한 위치에 압축 해제

☰ README.md

Azure Pipelines succeeded

### Downloads for Windows x86\_64

- 1.8.0\_322-1 (LTS, supported until May 2026, announcement)
  - [java-1.8.0-openjdk-1.8.0.322-1.b06.ojdkbuild.windows.x86\\_64.zip \(sha256\)](#)
  - [java-1.8.0-openjdk-1.8.0.322-1.b06.ojdkbuild.windows.x86\\_64.msi \(sha256\)](#)
- 11.0.14.9-1 (LTS, supported until October 2024, announcement)
  - [java-11-openjdk-11.0.14.9-1.windows.ojdkbuild.x86\\_64.zip \(sha256\)](#) (highlighted with a red box)
  - [java-11-openjdk-11.0.14.9-1.windows.ojdkbuild.x86\\_64.msi \(sha256\)](#)
- 17.0.2.0.8-1 (LTS, announcement)
  - [java-17-openjdk-17.0.2.0.8-1.win.x86\\_64.zip \(sha256\)](#)
  - [java-17-openjdk-17.0.2.0.8-1.win.x86\\_64.msi \(sha256\)](#)

# 설치 사전 조건 – Java 설정

## □ 설정

- 시스템 환경변수에 'JAVA\_HOME' 추가: JDK 압축 해제 폴더(JRE 아님!!)
- 시스템 환경변수의 path에 JDK bin 폴더 추가: JDK 압축 해제 폴더의 bin 폴더
- 예제에서는 "C:\DevTools\OpenJDK11" 활용

## □ 시스템 환경 변수 진입

- 내컴퓨터 -> 고급 시스템 설정 -> 환경 변수

## □ Java 설치 확인

- 반드시 새로운 명령 프롬프트 창을 열고 실행

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19044.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\devops>java -version
openjdk version "11.0.14" 2022-01-18 LTS
OpenJDK Runtime Environment 18.9 (build 11.0.14+9-LTS)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.14+9-LTS, mixed mode)

C:\Users\devops>
```

# [설치]

## □ 다양한 설치 방식 지원

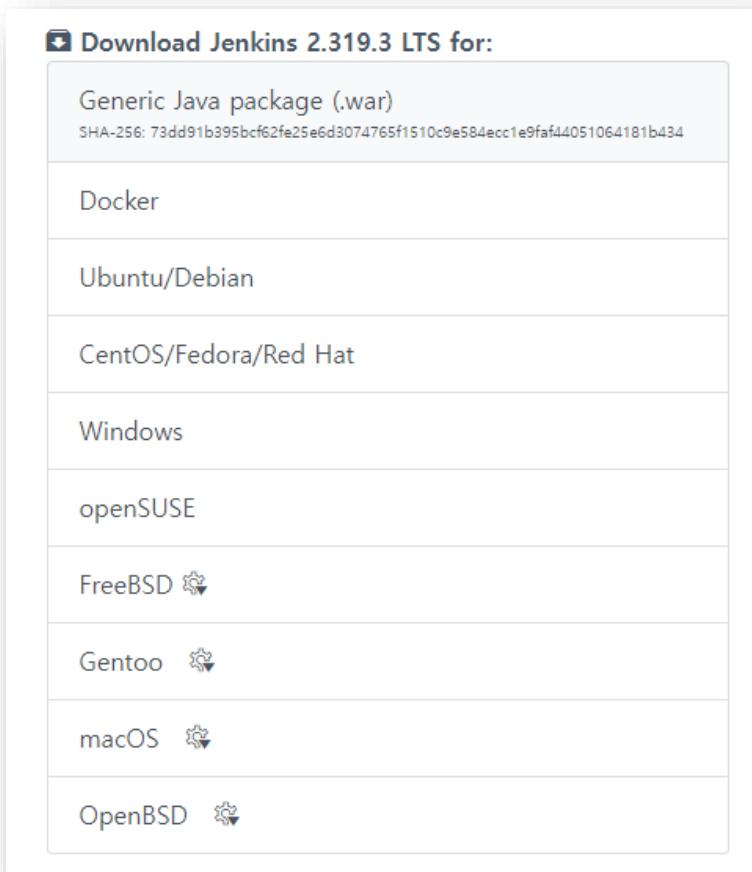
- WAS에서 실행하는 WAR 파일 제공
- 단독 실행(standalone)이 가능한 패키지 제공
- Docker 이미지 및 클라우드 배포 제공

## □ 다운로드

- <https://Jenkins.io/download/>

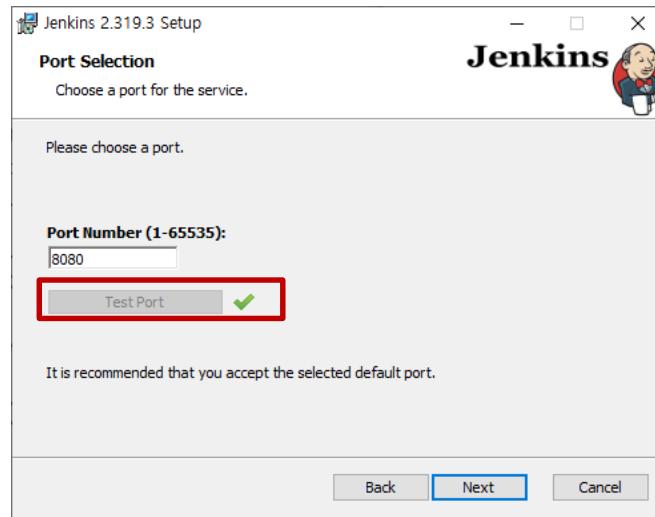
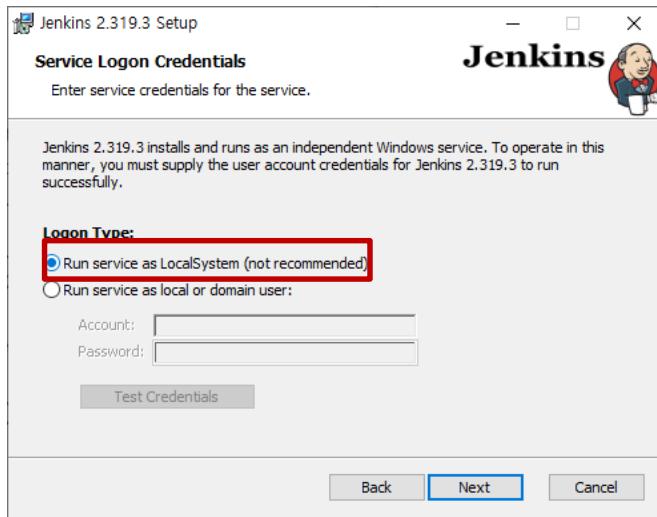
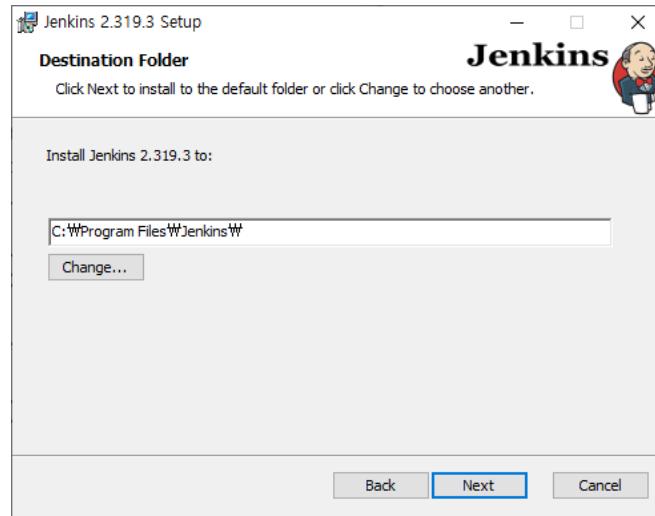
## □ 이번 교육에서는 Windows 설치 버전 활용

- 설치 후, 서비스 형태로 실행

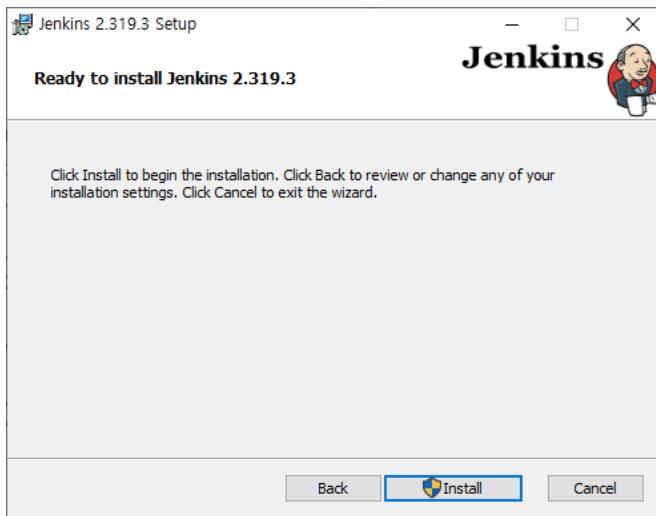
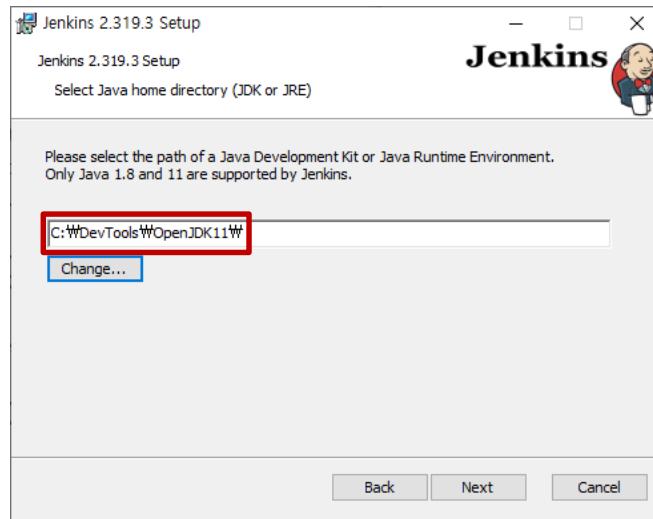
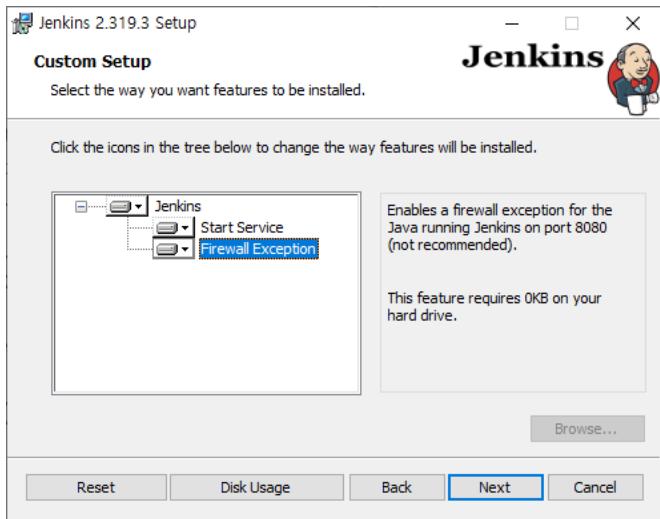


# [설치] 실행 파일로 설치

## □ 순서에 따라 진행



## [설치] 실행 파일로 설치



# Jenkins Home 설정

## □ Jenkins Home이란?

- Jenkins가 소스코드를 Clone, 빌드, 분석, 테스트하는 위치
- Jenkins의 Workspace 역할

## □ 설정 변경 방법

- 기본 설정은 매우 복잡한 경로 하위에 위치
- 빌드가 잘 되는지, 문제 발생 시 해결의 경우 접근 필요
- 별도의 폴더를 만들고 해당 폴더 지정 권고

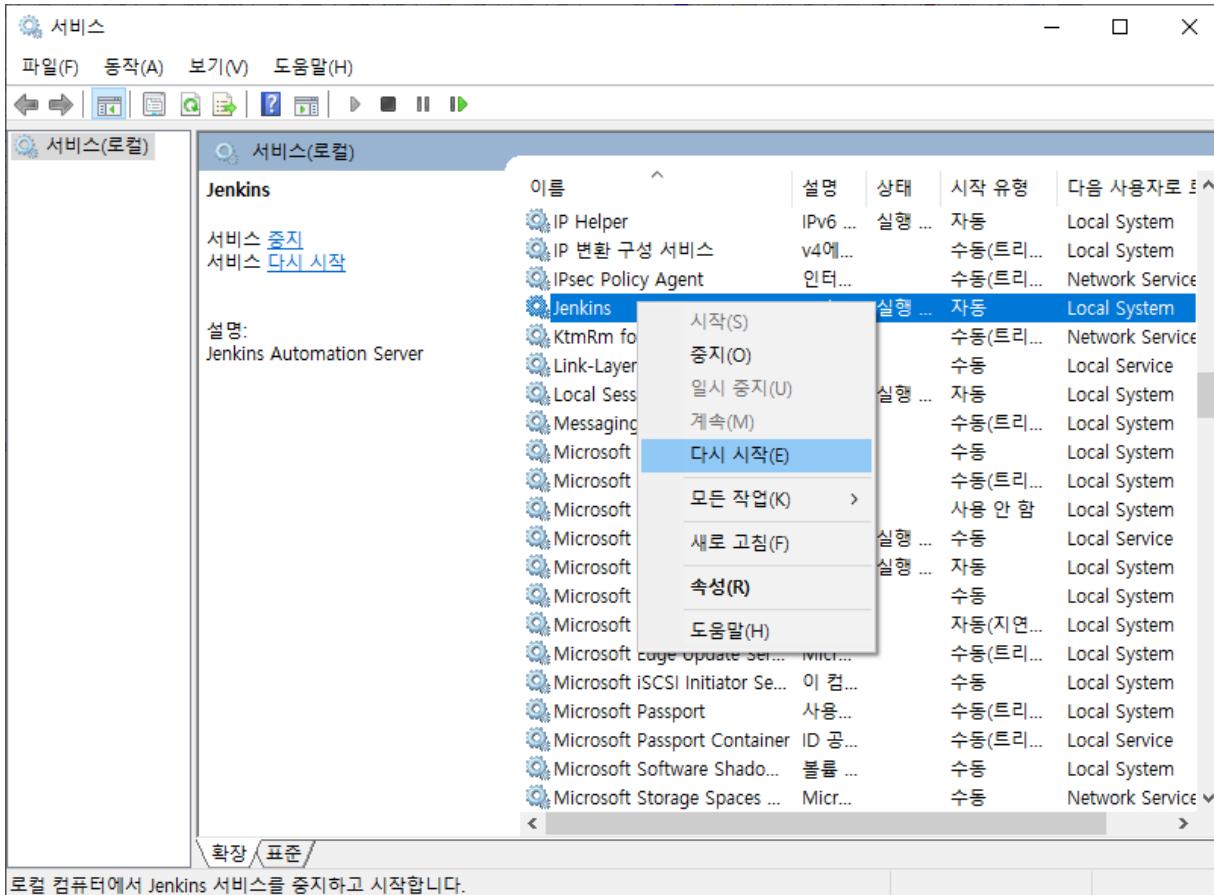
## □ 설정 변경 방법

- C:\Program Files\Jenkins\jenkins.xml 의 설정 변경
- 약 34줄의 <env name="JENKINS\_HOME" value="%ProgramData%\Jenkins\jenkins"/> 변경
  - value="C:\JenkinsWork" 등과 같이 적절한 위치 지정
- 저장 후, 서비스에서 Jenkins 다시 시작

```
<service>
  <id>jenkins</id>
  <name>Jenkins</name>
  <description>This service runs Jenkins automation server.</description>
  <env name="JENKINS_HOME" value="C:\JenkinsWork"/>
```

# Jenkins 서비스 재시작 방법

- Windows 관리도구 -> 서비스 실행
- Jenkins에서 오른쪽 클릭 -> 다시 시작



## [설정] Initial Password 입력

- Localhost:8080 접속
- 설치 위치에서 확인 가능

Getting Started

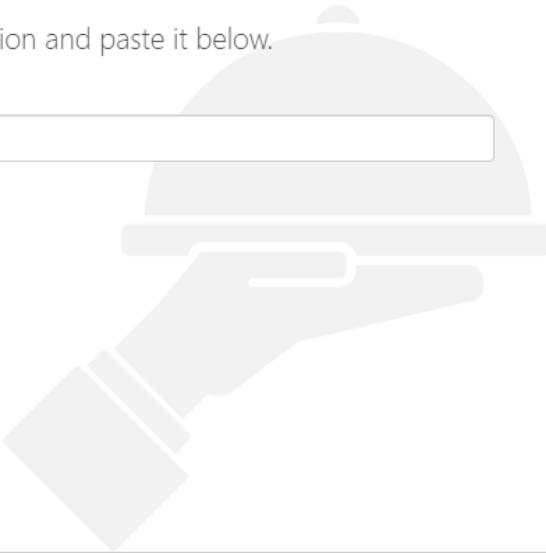
# Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

C:\JenkinsWork\secrets\initialAdminPassword

Please copy the password from either location and paste it below.

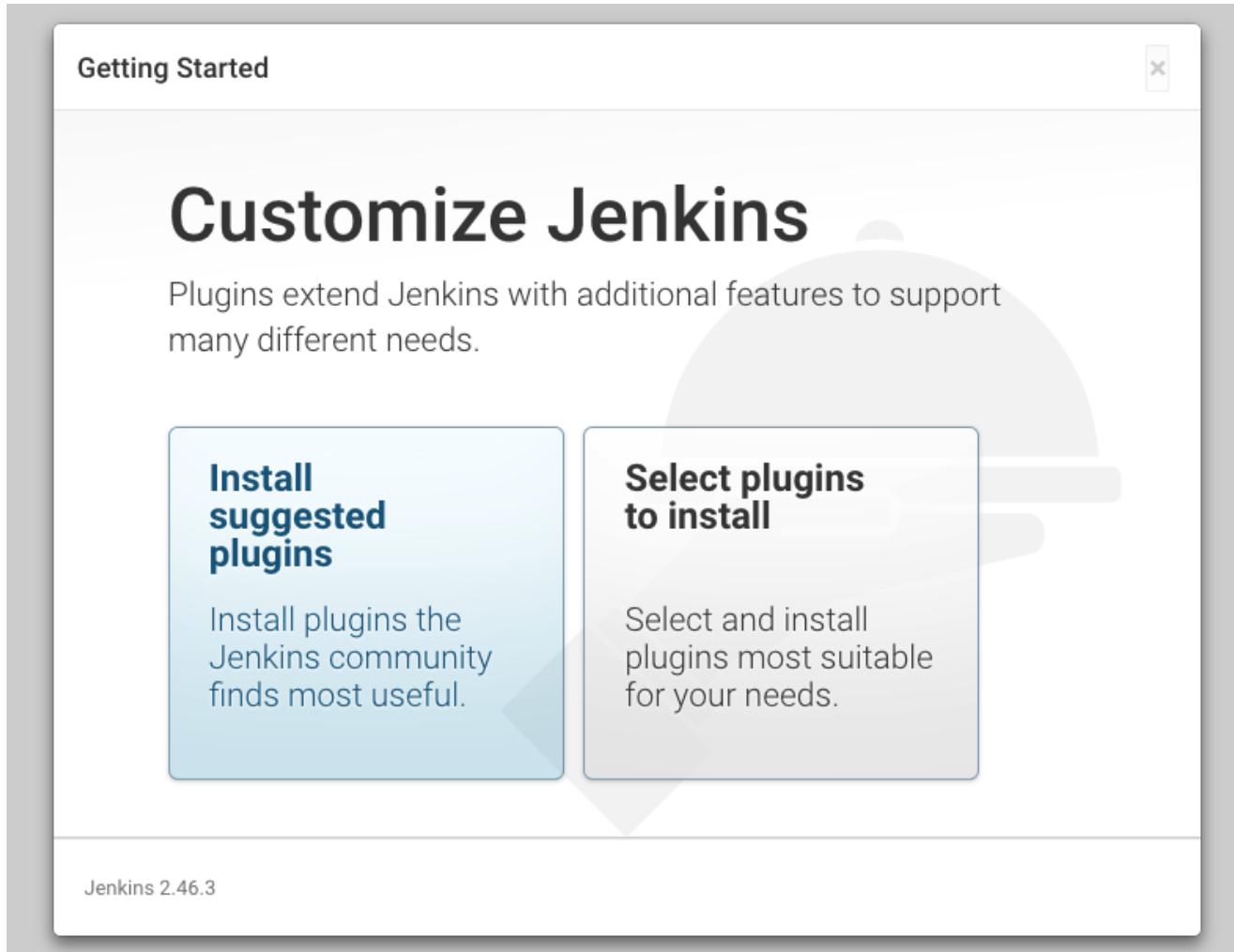
Administrator password



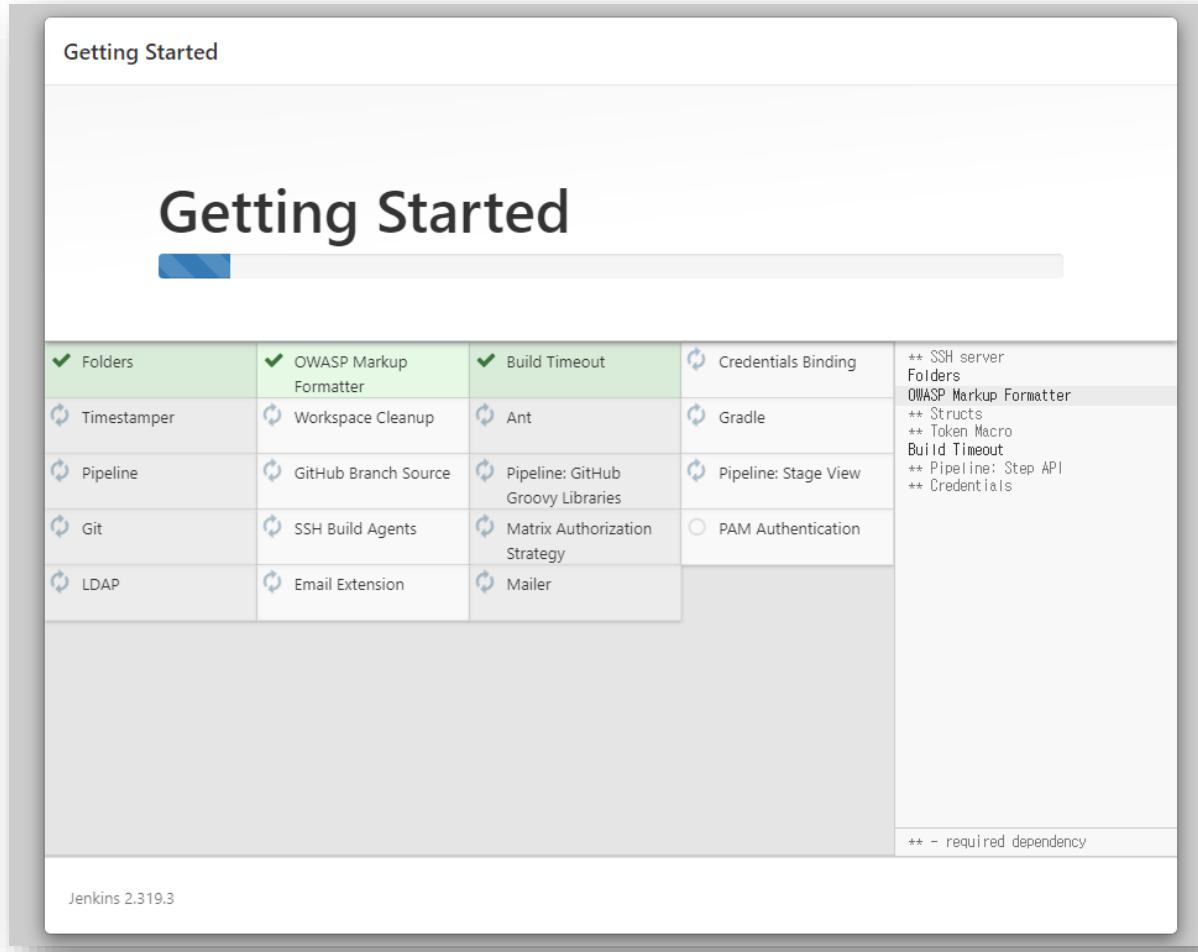
Continue

## [설정] 기본 설치 플러그인

- 제안 플러그인 설치 후, 필요 플러그인 언제든지 설치 가능



## [설정] 플러그인 설치 화면



## [설정] 최초 관리자 생성

Getting Started

# Create First Admin User

계정명:

암호:

암호 확인:

이름:

이메일 주소:

Jenkins 2.319.3

Skip and continue as admin

Save and Continue

## [설정] 접근 경로 설정

### □ Jenkins 관리에서 다시 설정

The screenshot shows the Jenkins 'Instance Configuration' page under the 'Getting Started' section. The title 'Instance Configuration' is prominently displayed. A 'Jenkins URL:' field contains the value 'http://localhost:8080/'. Below the field, explanatory text states: 'The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD\_URL environment variable provided to build steps.' Another note below says: 'The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.' At the bottom, it says 'Jenkins 2.319.3' and provides 'Not now' and 'Save and Finish' buttons.

Getting Started

# Instance Configuration

Jenkins URL: http://localhost:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD\_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.319.3

Not now

Save and Finish

# [설정] 초기화면

The screenshot shows the Jenkins Dashboard. At the top, there is a navigation bar with the Jenkins logo, a search bar, and user information (admin, 1 job). Below the navigation bar, the main content area has a title "Jenkins에 오신 것을 환영합니다." (Welcome to Jenkins) and a subtitle explaining the purpose of the page: "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." There is a "Create a job" button with a right-pointing arrow. On the left side, there is a sidebar with various links: "새로운 Item", "사람", "빌드 기록", "Jenkins 관리", "My Views", "Lockable Resources", and "New View". Under "빌드 대기 목록", it says "빌드 대기 항목이 없습니다.". Under "빌드 실행 상태", it shows "1 대기 중" and "2 대기 중". At the bottom right, there are links for "REST API" and "Jenkins 2.319.3".

## [설정] Jenkins 관리

#	메뉴	설명
1	시스템 설정	Jenkins 실행에 대한 전반적인 환경을 설정한다. 초기 화면에 보여질 시스템 메시지, 전역 변수, Maven, JDK 설정 등이 포함된다.
2	Configure Global Security	Jenkins 전반적인 보안 항목을 설정한다. 기본 값은 보안을 사용하지 않는 것이다. LDAP 사용 여부, ID 사용 여부 및 권한이 포함된다.
3	플러그인 관리	약 1000개 이상의 플러그인을 설치하고 업데이트 할 수 있다.
4	노드 관리	빌드 하기 위한 Master(Jenkins가 설치된 서버)와 Slave(빌드만 전용으로 하는 머신)를 설정한다.
5	사용자 관리	Jenkins 사용자를 설정하는 기능으로, 'Configure Global Security'의 설정 결과에 따라 이 메뉴가 표시된다. 최초 설치 후에는 표시되지 않는다.

# [설정] 시스템 기본 설정

---

## □ 홈 디렉토리

- 빌드 Job을 설정하고 빌드할 때 소스 코드 저장소에서 체크아웃한 파일이 저장되고, 빌드되는 로컬 디렉터리의 위치
- 특정 위치로(예를 들면 D:\ 등) 설정하고 싶다면, Jenkins 설치 전에 설정 필요
  - web.xml의 JENKINS\_HOME 변경

## □ # of executors

- Master에서 동시에 빌드하는 Job의 개수

## □ SCM checkout retry count

- 소스 코드 저장소에서 체크아웃을 실패했을 때 재시도 하는 횟수
- 기본값인 0은 체크아웃이 실패해도 빌드가 실패한 것으로 간주

## □ Help make Jenkins better by sending anonymous usage statistics and crash reports to the Jenkins project.

- Jenkins 개선을 위한 정보 보내기
- 오픈소스 프로젝트에 기여하는 하나의 방법

## [설정] Jenkins Location 설정

### □ Jenkins URL

- Jenkins 접속 주소. 반드시 IP나 URL로 변경 필요

### □ System Admin e-mail address

- Jenkins에서 사용자에게 알림 메일을 보낼 때 사용할 관리자의 e-mail 주소

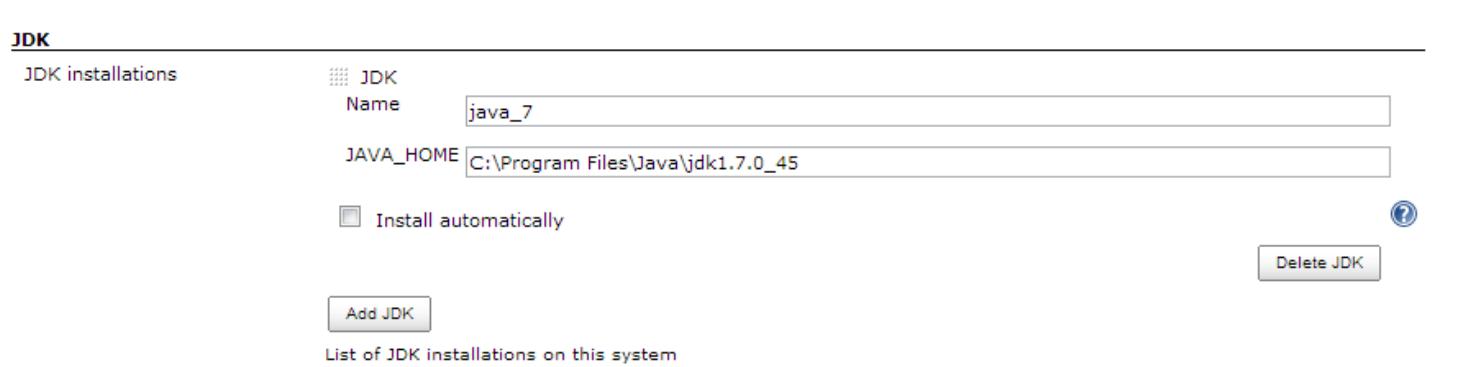
#### Jenkins Location

Jenkins URL	<input type="text" value="http://localhost:8080/jenkins/"/> <span>?</span>
<span style="color: yellow;">⚠ Please set a valid host name, instead of localhost</span>	
System Admin e-mail address	<input type="text" value="address not configured yet &lt;nobody@nowhere&gt;"/> <span>?</span>

## [설정] 빌드 도구 설정 (Java의 경우)

### □ JDK 설정

- 'Add JDK'를 클릭해 JDK 추가
- 자동으로 특정 버전을 추가하거나 JDK 설치 후 JAVA\_HOME의 위치를 지정



### □ Maven 설정

- JDK와 동일한 방식으로 추가

## [설정] Configure Global Security (권한) 설정

- 권한 없어도 접근 가능한 수준부터, 프로젝트 별 권한까지 설정 가능
- 사용 권한을 설정하기 위해서는 'Enable security'를 선택

Enable security ?

TCP port for JNLP slave agents  Fixed :   Random  Disable ?

Disable remember me  ?

Access Control

**Security Realm**

- Delegate to servlet container ?
- Jenkins' own user database ?
- LDAP ?

**Authorization**

- Anyone can do anything ?
- Legacy mode ?
- Logged-in users can do anything ?
- Matrix-based security ?
- Project-based Matrix Authorization Strategy ?

# [설정] Configure Global Security (권한) 설정

---

## □ Security Realm

- 권한이 어디에 위치하는지 지정하는 것으로 Jenkins 내부의 DB에 저장할 수도 있고, 다른 곳의 권한을 사용할 수도 있음
  - Delegate to servlet container: 컨테이너, 즉 Tomcat과 같은 WAS의 권한을 위임받아 사용
  - Jenkins' own user database: Jenkins의 내부 데이터 베이스에 아이디 정보 저장
  - LDAP: 사용자 계정 서버와 연결하여 사용
- 사용자 계정 서버가 없는 경우는 두 번째 선택값인 Jenkins 내부의 DB를 사용

# [설정] Configure Global Security (권한) 설정

## □ Authorization

- Anyone can do anything: 로그인 여부와 상관없이, Jenkins에 접속하는 누구나 Jenkins의 설정을 포함한 모든 기능을 사용할 수 있게 한다. 기본값이지만, 기본 설정 후 보안을 위해 반드시 다른 항목으로 변경해야 한다.
- Legacy mode: Tomcat과 같은 WAS의 admin 권한을 이용하여, admin만 모든 기능을 사용할 수 있고 그 외에는 읽기 전용 권한을 부여한다.
- Logged-in users can do anything: 1번 항목과 비슷하나, 로그인한 사용자가 모든 기능을 사용할 수 있게 한다.
- Matrix-based security: 역할이나 사용자에 따라 권한을 설정한다. 기본값으로 'Anonymous'가 포함되나, 다음 그림과 같이 'Add' 버튼을 이용해서 사용자나 그룹을 추가할 수 있다.

### Authorization

- Anyone can do anything
- Legacy mode
- Logged-in users can do anything
- Matrix-based security

User/group	Overall						Credentials						Slave					
	Administer	Configure	UpdateCenter	ReadRunScripts	UploadPlugins	Create	Delete	ManageDomains	UpdateView	Build	ConfigureConnect	CreateSlave	DeleteSlave	DisconnectSlave	ConnectSlave	DisconnectSlave		
Anonymous	<input type="checkbox"/>																	
admin	<input checked="" type="checkbox"/>																	

User/group to add:

Add

- Project-based Matrix Authorization Strategy

# [설정] Configure Global Security (권한) 설정

## □ Authorization (계속)

- Project-based Matrix Authorization Strategy: Matrix-based security와 유사하나, 각 Job 설정에서 다시 한 번 Job에 대한 권한을 설정할 수 있다. 예를 들어, A 프로젝트에서 빌드하는 Job과 B 프로젝트에서 빌드하는 Job이 다르고, A와 B 프로젝트 팀원은 다른 팀의 Job은 볼 수 없도록 설정할 때 사용한다. 첫 번째 그림과 같이 사용자나 그룹 별 권한을 설정하고, 두 번째 그림과 같이 Job 설정 화면에서 권한을 다시 한 번 설정할 수 있다.

Authorization

- Anyone can do anything
- Legacy mode
- Logged-in users can do anything
- Matrix-based security
- Project-based Matrix Authorization Strategy

User/group	Overall								Credentials			
	Administer	Configure	UpdateCenter	Read	RunScripts	UploadPlugins	Create	Delete	ManageDomains	UpdateView	View	
admin	<input checked="" type="checkbox"/>											
Anonymous	<input type="checkbox"/>											

User/group to add:

Enable project-based security

Block inheritance of global authorization matrix

User/group	Credentials				Job				Run	SCM				
	Create	Delete	ManageDomains	UpdateView	Build	Cancel	Configure	Delete	Discover	Read	WorkSpace	Delete	Update	Tag
Anonymous	<input type="checkbox"/>													
admin	<input checked="" type="checkbox"/>													

User/group to add:

# [설정] Configure Global Security (권한) 설정

---

## □ 주의할 점

- Jenkins를 설치한 후 'Jenkins's own user database'을 이용하여 사용 권한을 처음 설정할 때에는, 다음의 순서로 진행하자. 설치 후에는 Jenkins 사용자 DB에 따로 사용자가 없기 때문에, 만약 'Anyone can do anything' 외에 다른 항목으로 설정한다면 아이디가 없기 때문에 로그인을 못해서 아무 작업도 못할 수 있다.

1. 'Security Realm'에서 'Jenkins' own user database'를 선택한다
2. 'Authorization'에서 'Anyone can do anything'를 선택하고 적용한다.
3. 'Jenkins 관리'의 'Manage Users'에서 'admin' 또는 원하는 관리자 계정을 생성한다.
4. 다시 'Configure Global Security' 메뉴로 돌아와서, 'Authorization' 항목의 다른 권한 설정을 선택한다.
5. 단, Matrix-based security 등을 선택 시에는, 3번에서 생성한 관리자 계정을 'Add'로 추가하고 필요한 관리자 권한을 부여한다.

## [설정] 플러그인 관리

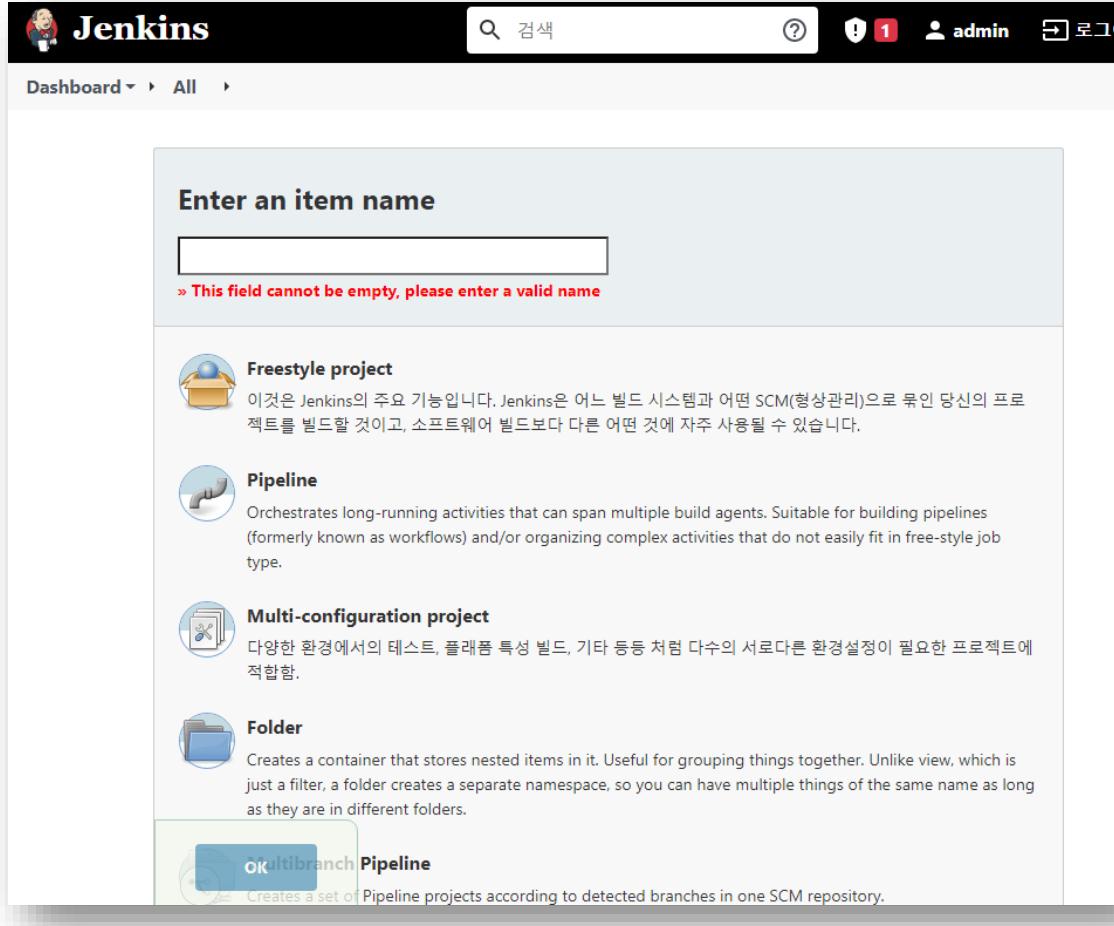
- 플러그인의 추가/삭제/업데이트 설정
- 플러그인 추가 후, Jenkins 재시작이 필요할 수 있음

The screenshot shows the Jenkins Manage Plugins interface. At the top, there are tabs: '업데이트된 플러그인 목록' (Updated Plugins List), '설치 가능' (Available), '설치된 플러그인 목록' (Installed Plugins List), and '고급' (Advanced). Below these tabs, there is a table with columns: '설치' (Install), '이름 ↓' (Name ↓), '버전' (Version), and '설치됨' (Installed). A single plugin is listed: 'SCM API Plugin'. The '설치' column has a checkbox next to it. The '이름' column shows 'SCM API Plugin'. The '버전' column shows '1.1'. The '설치됨' column shows '1.0'. Below the table, there is a note: 'This plugin provides a new enhanced API for interacting with SCM systems.' At the bottom left, there is a button labeled '지금 다운로드하고 재시작 후 설치하기' (Download now and restart after installation). In the center, there is a message: 'Update information obtained: 5 sec ago'. At the bottom right, there is a button labeled '지금 확인' (Check now).

# [Job 설정] Job 추가

## □ 메인화면의 '새로운 Item' 클릭

- Python 프로젝트의 경우, Freestyle 로 설정



## [Job 설정] 설명 설정

### □ Job에 대해 알리고 싶은 내용을 작성

The screenshot shows the 'General' tab selected in a Jenkins job configuration. Below it, there is a section titled '설명' (Description) with a text area containing the placeholder text '설명을 작성합니다.' (Write a description). At the bottom of this section, there are two links: '[Plain text]' and '[미리보기]' (Preview).

General	소스 코드 관리	빌드 유발	빌드 환경	Build	빌드 후 조치
---------	----------	-------	-------	-------	---------

설명

설명을 작성합니다.

[Plain text] [미리보기]

## [Job 설정] 기본 설정

- 오래된 빌드 삭제 ?
- 이 빌드는 매개변수가 있습니다 ?
- 빌드 안함 (프로젝트가 다시 빌드를 할 때까지 새로운 빌드가 실행되지 않습니다.) ?
- 필요한 경우 concurrent 빌드 실행 ?

설정 항목	설명
오래된 빌드 삭제	이 항목을 설정하지 않으면 Jenkins는 빌드 결과를 계속 유지한다. 그러나 하드디스크 용량 등을 고려했을 때, 지난 빌드 결과는 삭제하는 것이 필요할 수도 있다. 이 항목을 선택하면 특정 기간이나 빌드 수만큼 빌드 결과를 보관할 수 있다.
이 빌드는 매개변수가 있습니다.	빌드 시 매개변수를 정의하고 사용할 수 있다. 매개변수는 단순히 Text도 가능하지만, Subversion의 tag 목록도 가능하다.
빌드 안함	이 항목을 선택하면 빌드가 수행되지 않는다. 모든 빌드 설정은 유지하면서, 잠시 빌드를 사용하지 않을 때 활용한다.
필요한 경우 concurrent 빌드 실행	Jenkins는 하나의 Job을 빌드 중 또 다시 해당 빌드를 시작하면 Queue에 쌓이게 된다. 그러나 Queue에는 1개까지만 유지되고, 나머지 빌드는 무시된다. 이 항목을 선택하면 빌드 머신이 가능한 경우 동시 빌드를 수행한다.

## [Job 설정] 소스코드 관리

### □ 빌드를 위해 소스코드 저장소에서 소스코드를 가져오는 설정

소스 코드 관리

None  
CVS  
CVS Projectset  
Git  
 Subversion

Modules Repository URL  ⓘ  
**Repository URL is required.**

Local module directory (optional)  ⓘ

Repository depth  ⓘ

Ignore externals  ⓘ

Add more locations...

Check-out Strategy  ⓘ  
Use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain when a new build starts.

Repository browser  ⓘ

고급...

# [Job 설정] 소스코드 관리

## □ 입력 정보

- 저장소 URL: 권한이 필요한 저장소라면, 계정 정보 추가 입력
- Branch: 작업을 수행할 브랜치 선택

**소스 코드 관리**

None ?

Git ? ?

**Repositories**

Repository URL ?  
https://github.com/DongJoonHan/DevOps\_Test.git

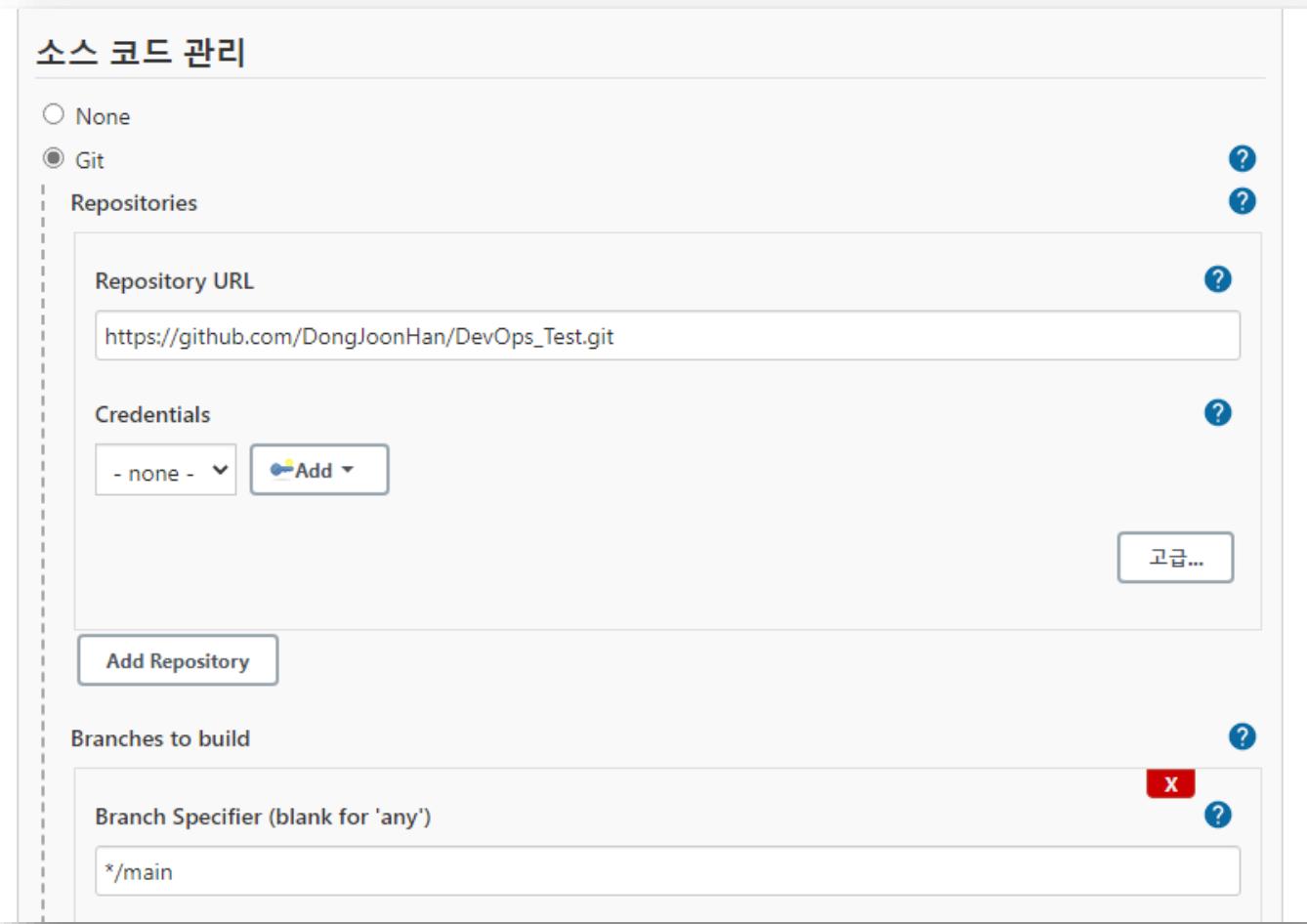
Credentials ?  
- none - Add

고급...

Add Repository

**Branches to build** ?

Branch Specifier (blank for 'any') X ?  
\*/main



## [Job 설정] 빌드 유발

### □ 빌드를 언제 수행할 것인지 지정

#### 빌드 유발

- Build after other projects are built
- Build periodically
- Poll SCM



항목	설명
Build after other projects are built	다른 Job의 빌드가 완료되면 이 빌드를 시작한다. 예를 들어 Library란 Job의 빌드가 완료된 후 이 빌드를 시작하려면, 'Projects to watch' 항목에 Job 이름인 Library를 입력하면 된다.
Build periodically	정해진 일정에 따라 빌드를 진행한다. 일정 지정 형식은 cron 표기법과 유사하다.
Poll SCM	정해진 일정에 따라 소스 코드 저장소에서 변경 사항이 있는지 확인한다. 변경이 있을 경우 빌드를 진행한다. 일정 지정 형식은 cron 표기법과 유사하다.

## [Job 설정] 빌드 유발

---

### □ Build periodically과 Poll SCM의 일정 지정은 cron 표기법을 사용

- TAB이나 공백으로 구분된 5자리
  - 분 시 일 월 요일순
- 예) 매일 새벽 5시에 빌드를 진행
  - 0 5 \* \* \*
- 예) 15분 단위로 소스 코드 저장소의 변경 사항을 확인
  - H/15 \* \* \* \*
- cron 표기법 참고
  - <https://en.wikipedia.org/wiki/Cron>

## [Job 설정] Maven 프로젝트

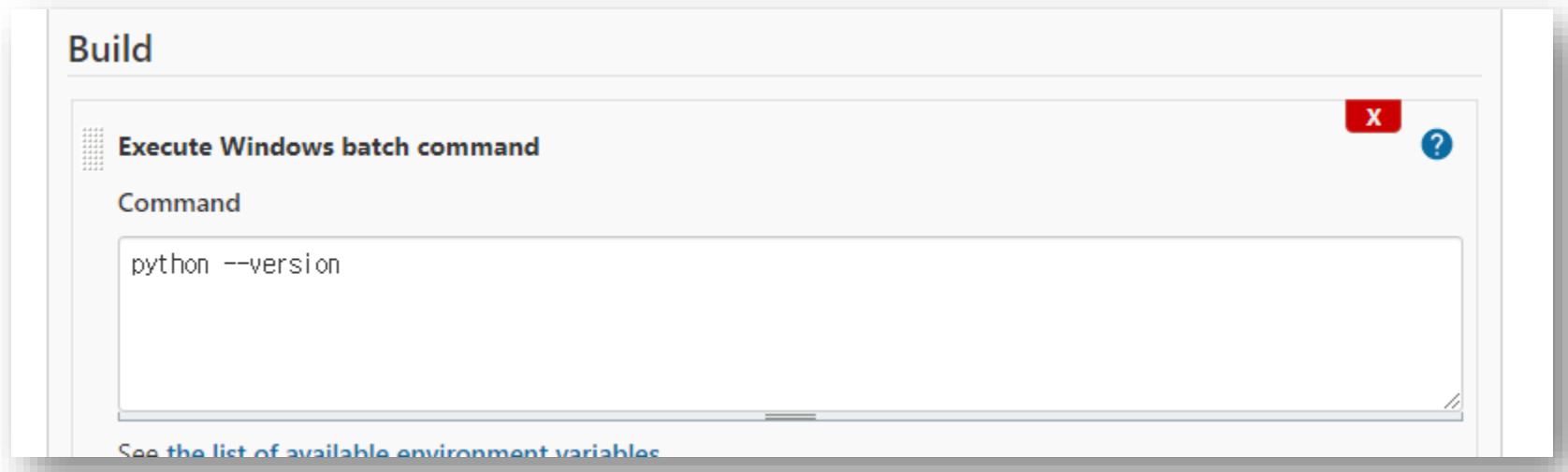
### □ Build에서 'Execute Windows batch command' 선택

- Windows에서 '명령 프롬프트' 창을 하나씩 실행하는 기능
- 이 Build 항목을 여러 개 사용할 수 있음
  - 마치 명령 프롬프트를 실행했다 종료하고, 새로 실행하고...

### □ 실행 명령 입력

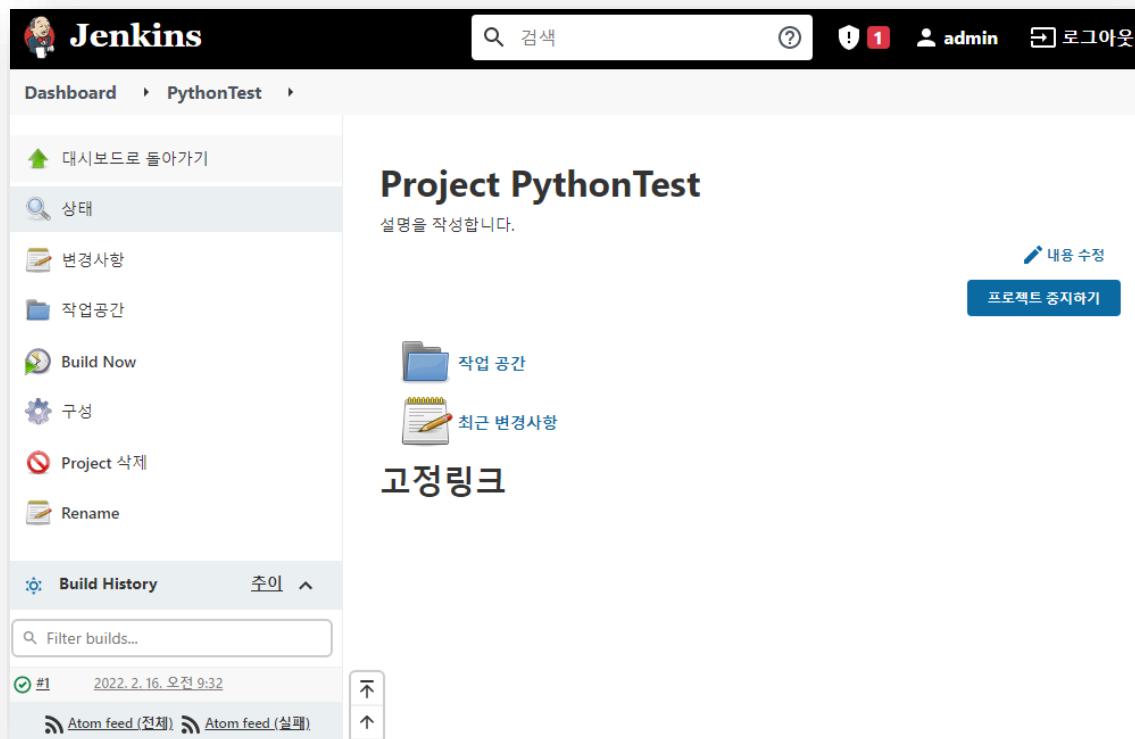
- 명령 프롬프트에 입력하는 것과 동일하게 입력
- 예제에서는 파이썬 버전 확인

### □ 완료 후 저장



## [Job 실행] 빌드 결과 확인

- Jenkins 메인화면에서 각 Job을 클릭하면 Job 화면으로 이동
- Job이 정상적으로 빌드되었는지 결과만 확인하려면, 왼쪽 하단의 Build History를 확인
  - #1, #2 순으로 빌드가 표시
  - 파란색 공일 경우 빌드 성공
  - 빨간색 공일 경우 빌드 실패



## [Job 실행] 빌드 실패 원인 확인

### □ 각 빌드 #를 클릭하고, Console Output 확인

The screenshot shows the Jenkins dashboard for a 'PythonTest' job. The build number is #1, and it was started on 2022. 2. 16. 오전... by admin. The status is 'No changes.' and the build time is 51 sec 전에 업데이드 됨. The build was triggered via a GitHub push. The revision is aa1ff5f8be1c42a3d056fc8750b2ed5fea747b2 and the repository is https://github.com/DongJoonHan/DevOps\_Test.git. The build log indicates a successful execution of Python --version.

### 콘솔 출력

```
Started by user admin
Running as SYSTEM
Building in workspace C:\Jenkins\Work\workspace\PythonTest
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/DongJoonHan/DevOps_Test.git
> git.exe init C:\Jenkins\Work\workspace\PythonTest # timeout=10
Fetching upstream changes from https://github.com/DongJoonHan/DevOps_Test.git
> git.exe --version # timeout=10
> git --version # 'git version 2.35.1.windows.2'
> git.exe fetch --tags --force --progress -- https://github.com/DongJoonHan/DevOps_Test.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe config remote.origin.url https://github.com/DongJoonHan/DevOps_Test.git # timeout=10
> git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10
Checking out Revision aa1ff5f8be1c42a3d056fc8750b2ed5fea747b2 (refs/remotes/origin/main)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f aa1ff5f8be1c42a3d056fc8750b2ed5fea747b2 # timeout=10
Commit message: "첫 커밋을 진행합니다"
First time build. Skipping changelog.
[PythonTest] $ cmd /c call C:\Windows\TEMP\jenkins5141540997017777496.bat

C:\Jenkins\Work\workspace\PythonTest>python --version
Python 3.10.2

C:\Jenkins\Work\workspace\PythonTest>exit 0
Finished: SUCCESS
```

# 4 Python Black

---

# Black

---

## □ 도구 개요

- 코드 스타일을 통일시켜주는 코드 포맷팅 도구(Code Formatter)

## □ 필요성

- 협업을 위한 약속
- 읽기 좋은 코드가 좋은 코드
- 스타일 기준에 대한 고민 줄이기

## □ 사용 방법

- Command-Line에서 문제 파일 찾고 Black이 수정하기
- (권고) IDE에서 파일 저장할 때마다 Black이 수정하기

# Black 설치

---

## □ PIP를 이용한 설치

- pip install black

```
PS C:\Users\devops\DevOps_Test> pip install black
Defaulting to user installation because normal site-packages is not
writeable
Collecting black
  Downloading black-22.1.0-cp310-cp310-win_amd64.whl (1.1 MB)
    |██████████| 1.1 MB 1.3 MB/s
Collecting mypy-extensions>=0.4.3
  Downloading mypy_extensions-0.4.3-py2.py3-none-any.whl (4.5 kB)
Collecting black-formatter
  Downloading black_formatter-0.1.0-py3-none-any.whl (1.1 kB)
```

## □ 설치 확인

- black

```
PS C:\Users\djhan\Prj> black
Usage: black [OPTIONS] SRC ...

One of 'SRC' or 'code' is required.
```

# Command-Line으로 사용하기

## □ 포맷팅 수정할 파일이 있는지 확인하기

- black --check [파일명] or [폴더명]
- 예) black --check .
  - 현재 폴더에 포맷팅 수정할 파일이 있는지 확인

```
PS C:\Users\djhan\Prj> black --check .
would reformat apitemp.py
would reformat calc.py
would reformat black_example.py
would reformat testSum.py
would reformat testSum2.py
```

## □ 포맷팅 수정하기

- black [파일명] or [폴더명]
- 예) black example.py

```
PS C:\Users\djhan\Prj> black testSum2.py
reformatted testSum2.py

All done! ✨ 🎉 ✨
1 file reformatted.
```

```
if very_long_variable_name is not None and \
    very_long_variable_name.field > 0 or \
    very_long_variable_name.is_debug:
    z = 'hello ' + 'world'
else:
    world = 'world'
    a = 'hello {}'.format(world)
    f = rf'hello {world}'
```



```
if (
    very_long_variable_name is not None
    and very_long_variable_name.field > 0
    or very_long_variable_name.is_debug
):
    z = "hello " + "world"
else:
    world = "world"
    a = "hello {}".format(world)
    f = rf"hello {world}"
```

# VSCode에서 사용하기

## □ VSCode에서 포매터로 지정하기

- Black을 기본 포맷터로 실행하도록 VSCode 기본 설정에 추가

- .vscode/settings.json 수정
- Ctrl + Shift + P로 명령 팔레트 이동
- 다음 항목 추가

```
"python.formatting.provider": "black", # VSCode의 기본 포맷터 대신 Black을 사용하게 해주는 설정  
"editor.formatOnSave": true # 코드를 저장할 때마다 자동으로 포맷팅 해주는 설정
```

## □ 파일 저장하기

- calc.py 파일의 저장 전 후의 차이

```
PS C:\Users\djhan\Prj> black --check .  
would reformat apitemp.py  
would reformat calc.py  
would reformat testSum.py  
  
Oh no! 💥 ❤️ 💥  
3 files would be reformatted, 2 files would be left unchanged.  
PS C:\Users\djhan\Prj> black --check .  
would reformat apitemp.py  
would reformat testSum.py  
  
Oh no! 💥 ❤️ 💥  
2 files would be reformatted, 3 files would be left unchanged.
```

## Black의 포맷팅 항목

---

### □ 별도 자료 제공

## 5 단위 테스트와 커버리지

---

# Python의 단위 테스트

---

## □ 단위 테스트

- 보통 메소드/함수를 대상으로 함
- “이런 값을 입력하고 실행하면 아마 이런 출력값이 나올텐데, 정말 예상대로 출력되었는가?” 확인
- 리팩토링의 기본 조건
- 일반적으로 단위 테스트를 도와주는 단위 테스트 프레임워크 활용

## □ Python의 단위 테스트 프레임워크

- unittest: Python에 기본으로 포함된 단위 테스트 프레임워크
- Pytest: 별도의 설치가 필요하며, unittest를 포함하여 테스트 기능을 확장한 프레임워크

# unittest의 기본 사용 방법

## □ 테스트 작성 기본

- unittest를 import
- Test Class를 만들고 unittest.TestCase 상속
- 관례적으로, test\_ 를 붙인 테스트 함수 작성

## □ Assertion

- 함수의 매개변수(parameter)와 반환값(return value)을 이용하여 테스트
- 특정 함수에 매개변수를 지정하고 실행하였을 때,  
반환값이 개발자가 예상한 값과 같은지 확인하는 방법

Assertion 함수	내용
assertEqual(a, b)	기대값과 결과값이 동일한지 비교
assertNotEqual(a, b)	기대값과 결과값이 동일하지 않은지 비교
assertTrue(x)	기대값이 true 인지 비교
assertFalse(x)	기대값이 false 인지 비교

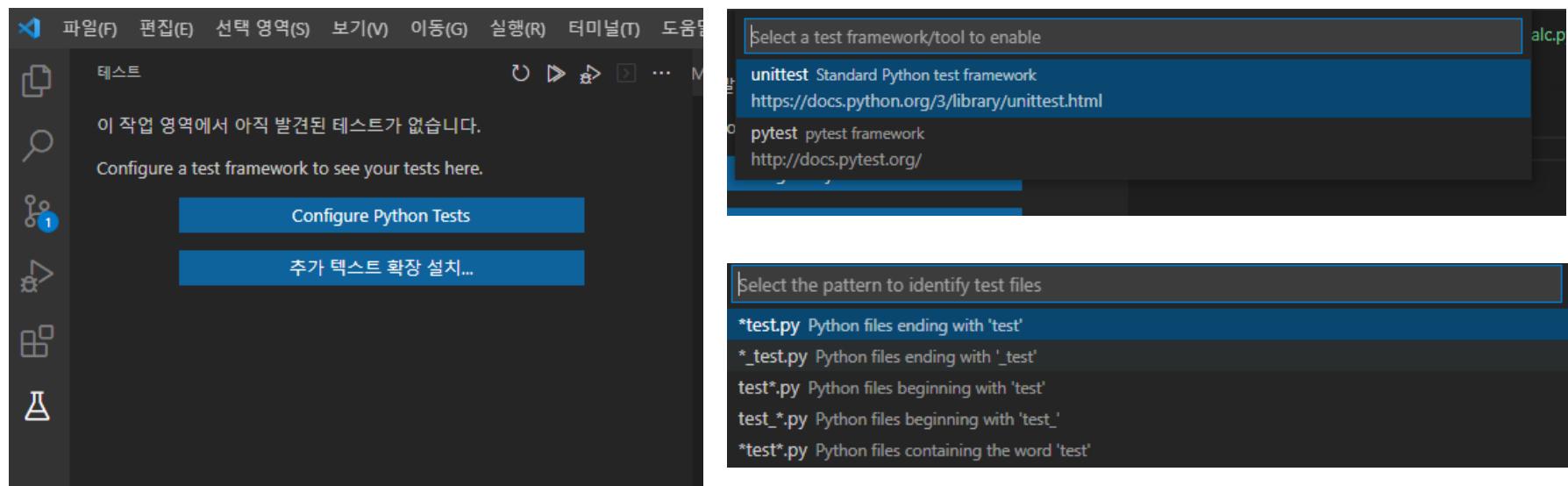
# 테스트 실행 방법

## □ 명령줄 실행

- python -m unittest // 전체 테스트 실행
- python -m 모듈명 // 해당 모듈만 테스트 실행
  - 예) python -m testSum

## □ VSCode에서 실행 설정

- 좌측 메뉴에서 "테스트" 선택
- Configure Python Test 선택 후, unittest 선택
- 테스트 파일 이름 유형에, test\_\*.py 선택



# unittest 예제

더하기 함수

```
def sum(a, b):  
    return a + b
```

테스트 클래스

```
1 import unittest  
2 import calc  
3  
4  
5 class sumtest(unittest.TestCase):  
6     def test_Sum(self):  
7         result = calc.sum(3, 5)  
8         self.assertEqual(8, result)
```

테스트 결과

```
PS C:\Users\djhan\Prj> python -m unittest testSum  
C:\Users\djhan\Prj\testSum.py:8: DeprecationWarning: Please use assertEquals instead.  
    self.assertEqual(8, result)  
.  
-----  
Ran 1 test in 0.002s
```

## 기능이 추가된 계산기 클래스

---

```
def sum(a, b):
```

```
    return a + b
```

```
def minus(a, b):
```

```
    return a - b
```

## 기능이 추가된 계산기 테스트 클래스

---

```
import unittest  
  
import calc  
  
  
  
  
  
class sumtest(unittest.TestCase):  
  
    def testSum(self):  
  
        result = calc.sum(3, 5)  
  
        self.assertEqual(8, result)  
  
  
  
  
  
    def testMinus(self):  
  
        result = calc.minus(5, 3)  
  
        self.assertEqual(2, result)
```

# Mock을 활용하기

---

## □ Mock이란?

- 테스트의 의존성을 줄이기 위한 가짜 객체를 테스트에 활용하는 방법
- 테스트 의존성?
  - 단위 테스트에서 DB에 항상 접속해야 한다면
  - DB에서 외부 API를 항상 불러야 한다면
- 특정 함수가 호출될 때, 특정 값으로 지정하는 기능
  - 실제 반환 값이 얼마이든, 내가 지정한 값으로 고정
  - 예) 외부 API에서 불러온 값이 무조건 2로 고정

## □ Python의 Mock 기능

- Unittest의 MagicMock 사용 가능
- 테스트 함수에서 @patch Annotation을 사용해서 지정
  - 예1) @patch("calc.api", return\_value=2) // calc의 api가 호출되면 반환값을 2로 고정해라
  - 예2) @patch("calc.api") // calc의 api가 Mock 대상이다

```
def testsum(self, mock): // mock이 포함된 함수 선언  
    mock.return_value = 2 // calc의 api가 호출되면 반환값을 2로 고정해라
```

## 외부 API 연동이 추가된 계산기 클래스

---

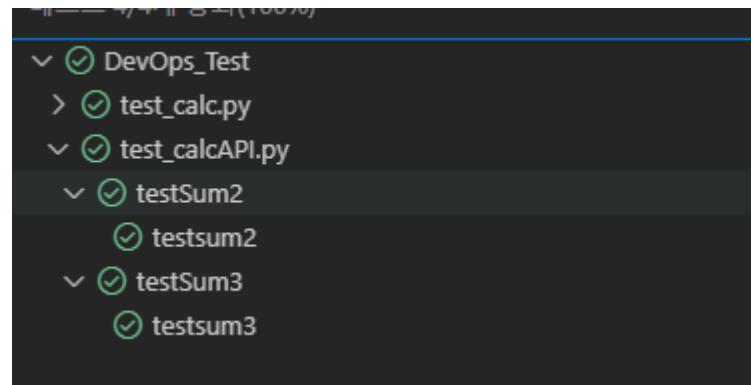
```
def sum(a, b):  
    return a + b  
  
def minus(a, b):  
    return a - b  
  
def sumWithApi(a): #api 호출  
    return a + api()  
  
def api(): # api  
    return 3
```

## 외부 API 연동을 Mock 처리한 테스트 클래스

```
import unittest
from unittest.mock import MagicMock, patch
import calc

class testSum2(unittest.TestCase):
    @patch("calc.api", return_value=2)
    def testsum2(self, mock):
        result = calc.api()
        self.assertEqual(result, 2)

class testSum3(unittest.TestCase):
    @patch("calc.api")
    def testsum3(self, mock):
        mock.return_value = 2
        result = calc.sumWithApi(5)
        self.assertEqual(result, 7)
```



## Coverage

---

### □ 소스코드의 얼마만큼 테스트 했는가?

- 라인(Line) 커버리지 (Statement 커버리지 라고도 함)
- 브랜치(Branch) 커버리지 (Decision 커버리지 라고도 함)

### □ 왜 소스코드 테스트 커버리지가 중요한가?

- 소스코드 품질을 위한 테스트 목표 (QA가 닥달하기 딱 좋은 지표)
  - 안전성 분야에서는 100%가 기준
- 테스트가 부족한 부분은 확인
  - 불필요한 코드인가?
  - 테스트가 충분한가?

### □ Python의 경우, 기본적으로 라인 커버리지만 측정

# Line 커버리지

---

## □ Line(Statement) 커버리지 산출 공식

- 테스트한 구문 / 전체 구문

※ 구문(statement): ';'로 끝나는 명령 라인

## □ 의미

- 프로그래밍 언어의 기본 구성 단위에 대한 검증
- 개발자가 구현한 전체 구문 및 블록에 대하여 최소한의 검증 수행
- 낮은 단계의 커버리지이나 시스템이 복잡할 수록 달성하기 어려움

# Line 커버리지

## □ 예제

```
#include <stdio.h>

void api1(int i){

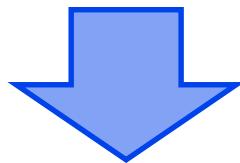
    printf("...\\n");
    if(i<0){
        printf("...\\n");
        printf("...\\n");
    }
}
```

### Statement Coverage

Q1) i를 0으로 테스트 한 경우:

Q2) i를 -1로 테스트 한 경우:

기혼자이고  
나이가 35세를 초과하면  
100만원의 소득 공제를 받음



Condition      Condition  
if(married && (age > 35))  
                      ----- Decision  
then amount = amount + 100;

# Branch(Decision) 커버리지

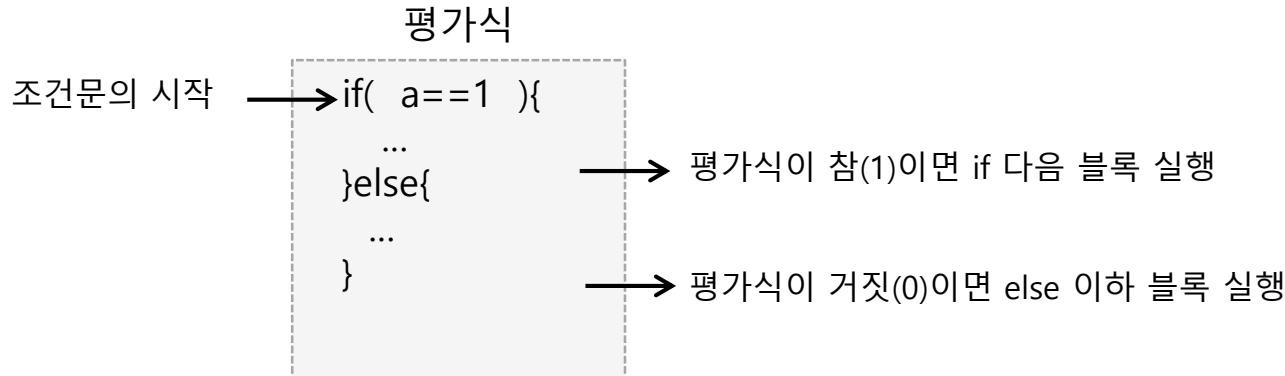
## ✓ Decision Coverage 산출 공식

- 테스트한 Decision 수 / 전체 Decision 수
- Decision Coverage 100% 달성하기 위한 최대 테스트 케이스 수
  - 전체 조건문 수 \* 2

## ✓ 의미

- 결정의 참/거짓을 한 번씩 테스트 해보는 것

## ✓ 코드 분석 팁(Code Reading Tip)



- 각 조건문은 2개의 분기를 가진다.

## Branch(Decision) 커버리지

```
void util(int a){  
    if( a>0 ){  
        .....  
    }  
  
    if( a==1 ){  
        ...  
    }else{  
        ...  
    }  
}
```

Q1) a를 0으로 테스트 한 경우:

Q2) a를 1으로 테스트 한 경우:

Q3) Decision Coverage 100%를 달성하기 위한 최소개의 테스트 케이스는?

# 도구 설치 및 실행

---

## □ pip install coverage

- 단, 설치 후 path에 pip의 패키지 등록 위치를 path에 추가하는 것을 권고
- 일반적으로 다음과 같음
  - C:\Users\사용자계정\AppData\Roaming\Python\Python310\Scripts

## □ 커버리지 측정 실행

- 측정

```
coverage run -m unittest discover
```

```
PS C:\Users\devops\DevOps_Test> coverage run -m unittest discover
C:\Users\devops\DevOps Test\test calc.py:12: DeprecationWarning: Pl
    self.assertEquals(2, result)
..C:\Users\devops\DevOps_Test\test_calcAPI.py:11: DeprecationWarnin
g: Please use assertEquals instead.
    self.assertEquals(result, 2)
...
-----
---
Ran 4 tests in 0.031s
```

# 커버리지 측정 확장

## □ 커버리지 결과 보기(터미널)

- coverage report

```
PS C:\Users\devops\DevOps_Test> coverage report
Name           Stmts   Miss  Cover
-----  
calc.py          8      1    88%  
test_calc.py     9      0    100%  
test_calcAPI.py 14      0    100%  
-----  
TOTAL           31      1    97%
```

## □ 커버리지 결과 보기(html)

- coverage html
- 결과 저장 폴더가 생성되고, html 파일이 생성

Coverage report: 97%

Module	statements	missing	excluded	coverage
calc.py	8	1	0	88%
test_calc.py	9	0	0	100%
test_calcAPI.py	14	0	0	100%
Total	31	1	0	97%

coverage.py v6.3.1, created at 2022-02-17 06:33 +0900

## Jenkins에서 결과 보기

---

### □ Jenkins에서 결과를 확인하기 위해 다음이 필요

- coverage 결과를 xml로 출력하기
  - coverage xml -o coverage.xml
  - coverage.xml 로 결과를 출력하라는 명령
- 빌드 후 조치 추가해서, coverage 표현 지정

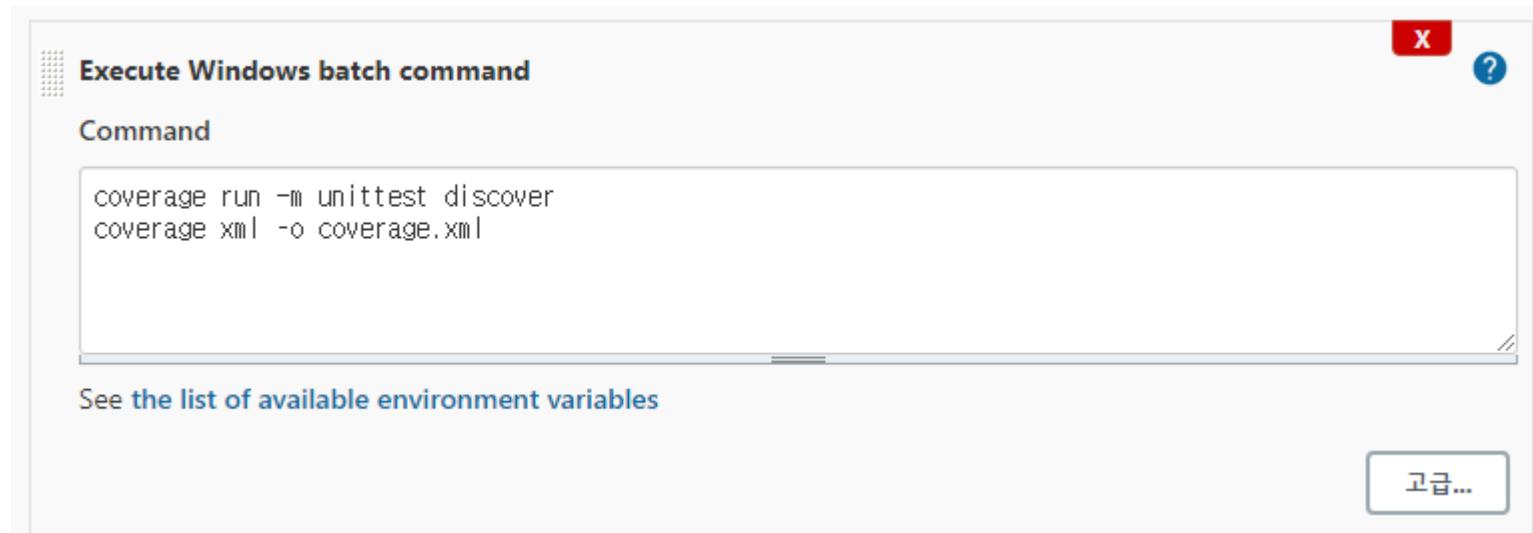
# Coverage 측정 및 표현 설정 - 1

## □ Jenkins 서버의 Coverage 실행 설정

- Jenkins 서버도 coverage 설치 및 path 설정

## □ Jenkins Job에서 Coverage 실행 설정

- coverage run -m unittest discover
- coverage xml -o coverage.xml

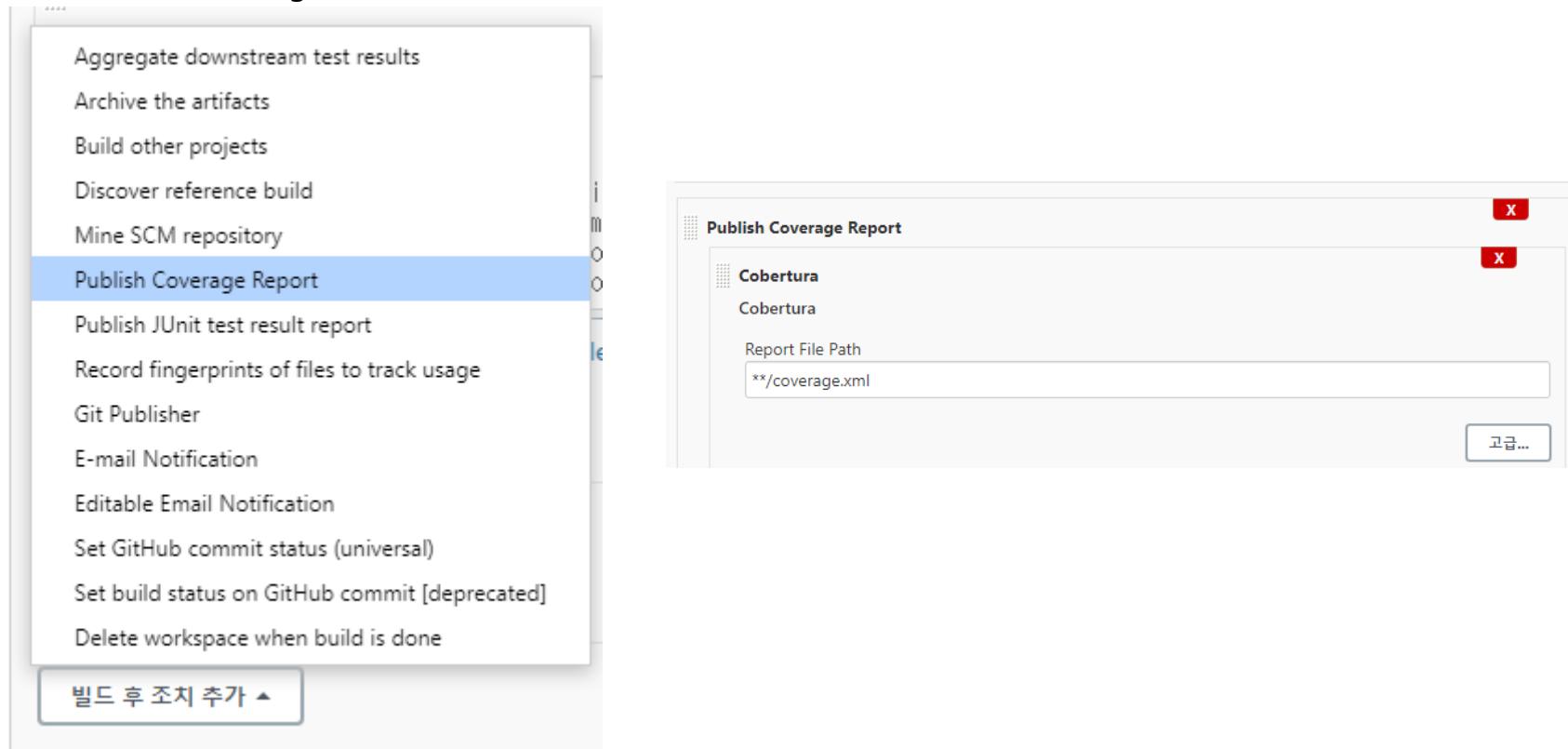


\* 단, 처음 실행 시 pip install coverage 필요

## Coverage 측정 및 표현 설정 - 2

### □ Jenkins Job에서 커버리지 표현 설정

- 빌드 후 조치 추가에서, Publish coverage report 선택
- Add를 클릭해서, Cobertura 선택
- Publish coverage report 에서 앞서 설정한 xml 파일명 지정
  - \*\*/coverage.xml



## Project PythonTest

설명을 작성합니다.

내용 수정

프로젝트 중지하기



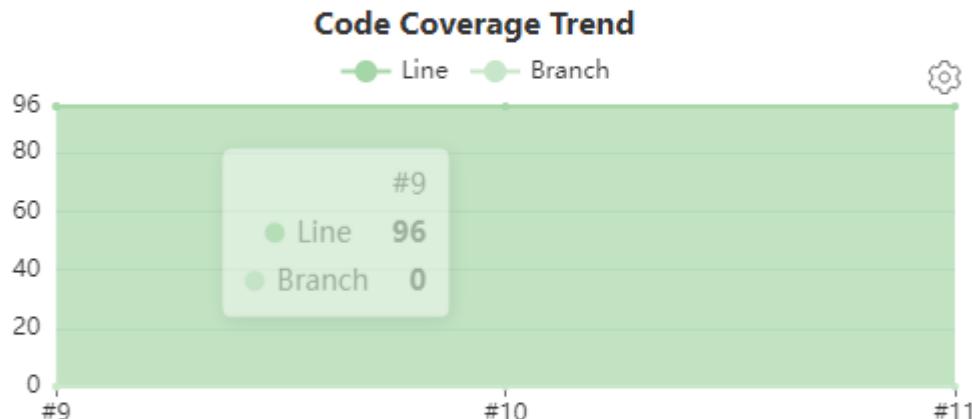
작업 공간



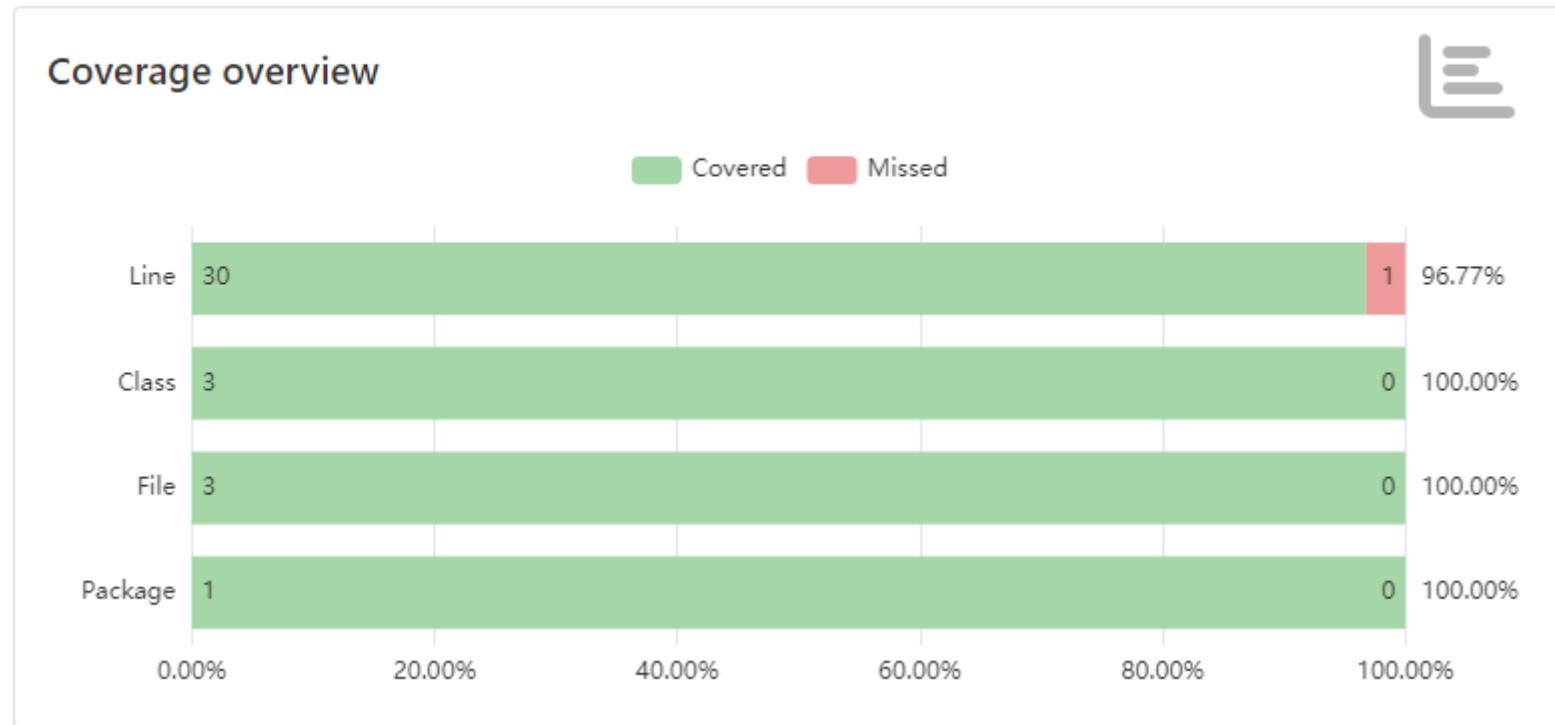
최근 변경사항

### 고정링크

- Last build, (#11), 1 min 13 sec 전
- Last stable build, (#11), 1 min 13 sec 전
- Last successful build, (#11), 1 min 13 sec 전
- Last failed build, (#6), 28 min 전
- Last unsuccessful build, (#6), 28 min 전
- Last completed build, (#11), 1 min 13 sec 전



### Coverage of 'cobertura: coverage.xml'



## Jenkins에서 커버리지 확인 - 파일별

### Coverage details

Package Overview File Coverage

Show 10 entries Search:

Package	File	Line Coverage	Branch Coverage
.	calc.py	87.50%	n/a
.	test_calc.py	100.00%	n/a
.	test_calcAPI.py	100.00%	n/a

Showing 1 to 3 of 3 entries

## 6 SonarQube

---

# 정적 분석 개요

## □ 정의

- 실제 실행 없이 컴퓨터 소프트웨어, 특히 소스코드를 분석하는 것 (위키피디아)

## □ (개발자를 위한) 사용 시기

- 컴파일 전에 소스코드 품질 검토
  - 소스코드 품질이란? 복잡도, 라인수, 복사-붙여넣기 등

## □ 사용 목적

- 소스코드의 잠재적인 품질 문제(낮은 품질) 발견
  - 네트워크 자원 누수, 높은 복잡도, 추천하지 않는 패턴 등
  - 네트워크 자원 누수 예) 네트워크 소켓의 open()는 있으나, 명시적인 close()가 없는 경우
- 결함의 조기 발견
  - 메모리 누수, 버퍼 오버플로우 등
- 소스코드 표준 준수 확인
  - 코딩 컨벤션(명명규칙), 보안

```
// 임시 저장값 불러오기
SharedPreferences prefs = getSharedPreferences("temp", MODE_
String notice = prefs.getString("notice_temp", "");
String[] weeks = { "일", "월", "화", "수", "목", "금", "토" };
// 날짜 불러오기
Calendar m = Calendar.getInstance();

year = m.get(Calendar.YEAR);
month = m.get(Calendar.MONTH) + 1;
day = m.get(Calendar.DATE);

// 카드 인터페이스를 불러온다.
// initCardView
mCardView = (CardUI) findViewById(R.id.cardsview);
mCardView.setSwipeable(true);
// add AndroidViews Cards

// 공지사항을 불러온다.
if (notice.matches("")) {
} else {
```



File	Line	Priority	Type	Category
AddSubject.java:5	5	Normal	UnusedImports	Import Statements
AddSubject.java:163	163	Normal	EmptyStatementNotInLoop	Empty Code
AddTodo.java:4	4	Normal	UnusedImports	Import Statements
AddTodo.java:13	13	Normal	UnusedImports	Import Statements
Avoid unused imports such as				
AlertUI.java:12	12	Normal	UnusedLocalVariable	Unused Code
ChangeSubject.java:5	5	Normal	UnusedImports	Import Statements
ChangeSubject.java:195	195	Normal	EmptyStatementNotInLoop	Empty Code

분석 결과

# [예제] MISRA 룰의 Good/Bad Case

## MISRA\_15\_03 [Contents]

### switch 문의 마지막 절이 default 절이어야 함

마지막 절에 `default`가 오는 것은 방어적 프로그래밍을 위함이다. `default` 절에서는 적절한 행동을 취하거나 아무 행동을 하지 않으면 적절한 주석을 달아야 한다.

#### [0090] switch문에 default 가 존재 검사

**BAD**

```
switch ( defaults )
{
    case 3U:
    {
        use_uint16 ( defaults );
        break;
    }
    case 4U:
    {
        use_uint16 ( defaults );
        break;
    }
    /* Not compliant. No default clause. */
}
```

**GOOD**

```
switch ( defaults )
{
    case 3U:
    {
        use_uint16 ( defaults );
        break;
    }
    case 4U:
    {
        use_uint16 ( defaults );
        break;
    }
    default:
    {
        use_uint16 ( defaults );
        break;
    }
    /* Compliant */
}
```

SW 품질 확보를 위한

# 가성비

더 적은 노력과  
일찍 수정해야 적은 비용

소프트웨어 결함은

구현 단계에서 가장 많이 유입

소프트웨어 개발은  
노동집약적 산업

# 제품 품질 메트릭

---

## □ 규모 관련

- 라인 수(LOC: Line of Code)
- 주석 제외 라인 수
- 주석 비율
- 함수 별 라인 수

## □ 복잡도 관련

- 순환 복잡도(Cyclomatic Complexity): 함수의 제어 흐름이 얼마나 복잡한지 측정. 분기문 +1
- 인지 복잡도(Cognitive Complexity): 함수가 얼마나 중첩문이 많은지 측정. Nesting Depth 라고도 표현

## □ 테스트 관련

- 기능 커버리지
- 함수 커버리지
- 문장/분기/MCDC 커버리지

## □ 룰 기반 정적 분석 관련

- 정적 분석 룰 위반 수

# 제품 품질 메트릭

---

## □ 결함 관련

- 소스코드 라인 대비 결함 비율
- 유형 별 결함 비율
- 원인 별 결함 비율

## □ 의존성 관련

- 함수 별 호출하는 건수
- 함수 별 호출되는 건수
- 변경 영향 비율(Stability): 한 함수가 변경되면 코드의 몇 %가 영향받는가?
- 상호 참조 비율: 상호 호출하는 파일이 있는가? (변경에 대한 영향도 파악 목적)

## □ 출시 관련

- 출시 후 결함 수

# 룰 기반(Rule based) 정적 분석 소개

---

## □ 개요

- 사전에 정해진 룰 가이드라인을 소스코드가 만족하는지 분석하는 도구
  - 룰 가이드라인은 검증 도구가 해석할 수 있는 룰과 이에 대한 가이드로 구성되며, 룰셋은 룰을 카테고리화 한 것
    - 예제 참고

## □ 대표 룰 가이드라인(코딩 표준)

- MISRA: Motor Industry Software Reliability Association에서 개발한 안전성, 호환성, 신뢰성을 위한 C, C++ 룰
- CERT: SEI CERT(Computer Emergency Response Team)의 SW 개발보안을 위한 C, C++, Java 룰
- 행정자치부 개발보안(시큐어 코딩): 행정기관의 SW 개발보안을 위한 JAVA, C, Android-Java 룰
- 기타 각 도구/언어/도메인/회사에 따른 룰 가이드라인이 존재
  - 예) 구 Sun 룰 / SonarQube 룰 / PMD 룰 / 방사청 룰

# PMD 를 적용 사례

## ◦ PMD 제공 전체 룰

#	Rule Set	Rule 명	선정후보 (QA)	선정후보 (개발자)	룰반영	Sonar 반영	한글 Rule 설명	
48	Design (java)	UnnecessaryLocalBeforeReturn	○		○	○	불필요한 지역 변수 생성은 피해야 함	Avoid the creation o
50	Design (java)	UncommentedEmptyMethod	○		○	○	주석 없는 빈 메소드는 주석 필요	Uncommented Empt
51	Design (java)	UncommentedEmptyConstructor	○		○	○	주석 없는 빈 생성자는 주석 필요	Uncommented Empt
75	Basic (java)	ForLoopShouldBeWhileLoop		○	○	○	for를 while로 간략화 할 수 있는 경우,	Some for loops can
78	Basic (java)	ReturnFromFinallyBlock	○		○	○	finally 블록에서 반환 하는것은 피해야	Avoid returning fro
79	Basic (java)	UnconditionalIfStatement	○		○	○	항상 true 이거나 false인 조건에서는 i	Do not use "if" state
80	Basic (java)	BooleanInstantiation		○	○	○	Boolean 객체는 인스턴스화를 지양함.	Avoid instantiating B
88	Basic (java)	AvoidUsingOctalValues	○		○	○	integer는 0으로 시작하면 안됨. 8진수	Integer literals shou
89	Basic (java)	AvoidUsingHardCodedIP	○		○	○	IP주소를 코드에 하드코딩 하면 안됨	Application with har
101	Strict Exceptions (java)	AvoidThrowingRawExceptionType	○		○	○	가공되지 않은 Exception을 throw하는	Avoid throwing cert
102	Strict Exceptions (java)	AvoidThrowingNullPointerException	○		○	○	NullPointerException을 throw하는 것	Avoid throwing Null
113	Android (java)	DoNotHardCodeSDCard		○	○		"/sdcard"를 사용하는 대신 Environment.get	Use Environment.get
116	Java Logging (java)	SystemPrintln	○		○	○	System.out.println은 보통 디버그 목적	References to Syste
120	Controversial (java)	OnlyOneReturn		○	○	○	메소드는 1개의 return만을 사용해야	A method should ha
121	Controversial (java)	AssignmentInOperand	○		○	○	피연산자내에 할당문이 사용됨. 해당	Avoid assignments i
138	Controversial (java)	AvoidLiteralsInIfCondition		○	○	○	조건문에서 하드 코딩된 literal의 사용	Avoid using hard-co
144	Type Resolution (java)	LooseCoupling		○	○	○	implementation 타입(예. HashSet) 사	Avoid using implem
148	Empty Code (java)	EmptyCatchBlock	○		○	○	빈 catch 블록은 피해야 함	Empty Catch Block f
149	Empty Code (java)	EmptyIfStmt	○		○	○	빈 if 문장은 피해야 함	Empty If Statement f
150	Empty Code (java)	EmptyWhileStmt	○		○	○	빈 while 문장은 피해야 함	Empty While Stateme
151	Empty Code (java)	EmptyTryBlock	○		○	○	빈 try 블록은 피해야 함	Avoid empty try blo
152	Empty Code (java)	EmptyFinallyBlock	○		○	○	빈 finally 블록은 피해야 함	Empty finally blocks
153	Empty Code (java)	EmptySwitchStatements	○		○	○	빈 switch 문장은 피해야 함	Empty switch statem
154	Empty Code (java)	EmptySynchronizedBlock		○	○	○	빈 Synchronized 블록은 피해야 함	Empty synchronized
155	Empty Code (java)	EmptyStatementNotInLoop	○		○	○	for나 while 내의 빈 문장(또는 ;만 있는	An empty statement
156	Empty Code (java)	EmptyInitializer		○	○		빈 초기화는 피해야 함	Empty initializers se
158	Empty Code (java)	EmptyStaticInitializer		○	○	○	빈 static 초기화는 피해야 함	An empty static initi
159	String and StringBuffer (jav	AvoidDuplicateLiterals		○	○	○	반복되는 Stirng literal은 constant로	Code containing du

# 룰 기반 정적 분석 도구

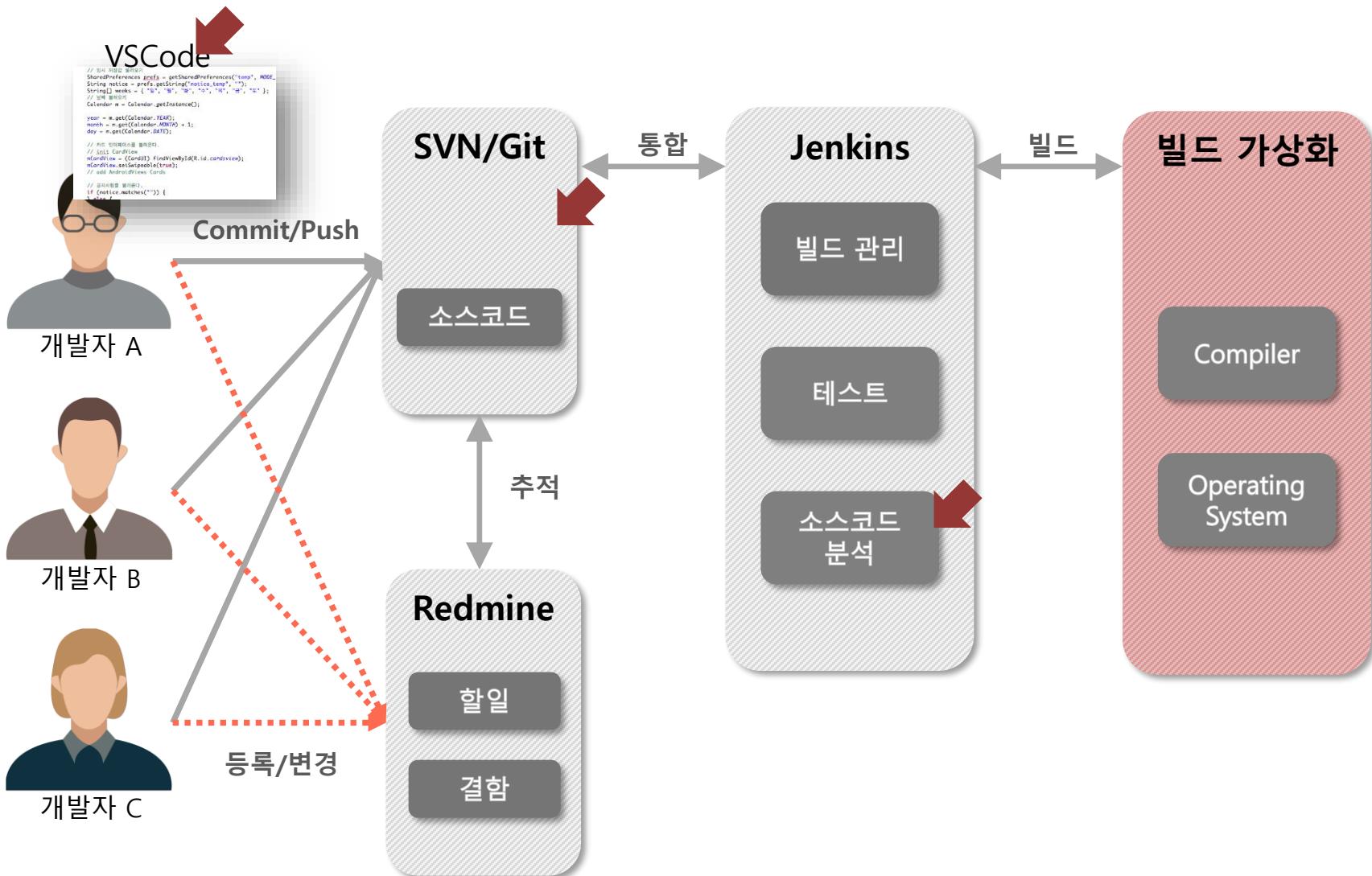
## □ 대표 도구

언어	국산	외산	오픈소스	특징
C/C++	Sparrow(파수) CodeInspector(Suresoft) Resort(Soft4Soft)	QAC/QAF(PRQA) Coverity(Synopsys) Polyspace-Bug Finder(MathWorks) Klockwork(Rogue Wave)	CppCheck	<ul style="list-style-type: none"><li>국산 도구는 행자부 개발 보안 검증</li><li>ISO 26262 인증 (오픈소스 제외)</li></ul>
Java	Sparrow(파수) CodeInspector(Suresoft) Resort(Soft4Soft)	QAC/QAF(PRQA) Coverity(Synopsys)	PMD FindBug	<ul style="list-style-type: none"><li>국산 도구는 행자부 개발 보안 검증</li><li>보안의 경우 제외하고, 오픈소스 위주 사용</li></ul>

## □ 도구 선택 시 확인 사항

- 분석 대상 언어(C/C++ or JAVA)
- 1종, 2종 오류 발생율
  - 1종: 결함인데 못잡는 것
  - 2종: 결함이 아닌데 결함이라 하는 것

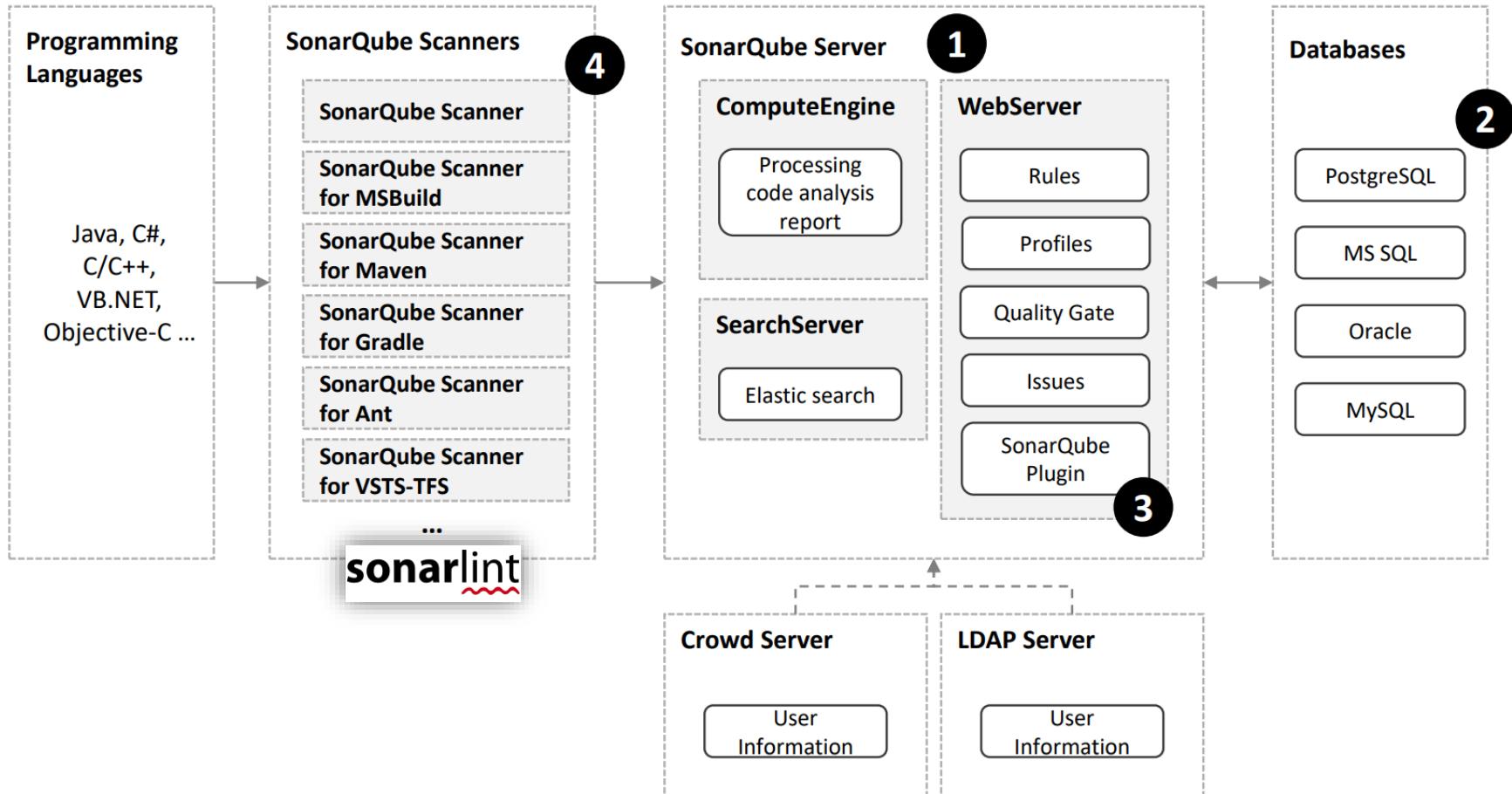
# [참고] ALM에서 정적 분석 도구 역할



# SonarQube 소개

## □ 개요

- 27개 이상 언어의 버그, 취약점, 코드악취 발견을 위한 정적 분석 및 웹을 통한 결과 게시를 지원
- Server, DB, Scanner, Plugin의 4개 서브 시스템으로 구성



# Sonarlint(Sonarscanner IDE Plugin)

---

## □ 개요

- IDE에서 플러그인 형식으로 SonarQube의 정적 분석을 실행할 수 있는 Plugin

\*참고: 개발자가 IDE에서 정적 분석을 실행하고 결과를 조치하는 것이 가장 효율적임

## □ 공식 지원 IDE

- Eclipse
- IntelliJ IDEA 계열
  - IntelliJ
  - PyCharm
  - PHPStorm
  - Android Studio
- Visual Studio
  - 2015
  - 2017
- VS Code

## □ 설치방법

- Plugin 설치 메뉴에서 “sonarlint” 검색

## SonarQube 룰셋

SonarQube는 Java, Object C, Python을 포함한 27개 이상의 개발 언어를 지원하며, 버그, 코드악취, 취약점으로 코딩 룰을 분류함



**버그:** 실행 중 기대하지 않은 행동을 할 수 있는 코드나 잠재 버그



**코드악취:** 유지보수성을 저하시키는 기술 부채를 의미



**취약점:** 설계와 다른 방향으로 프로그램을 실행시킬 수 있는 약점

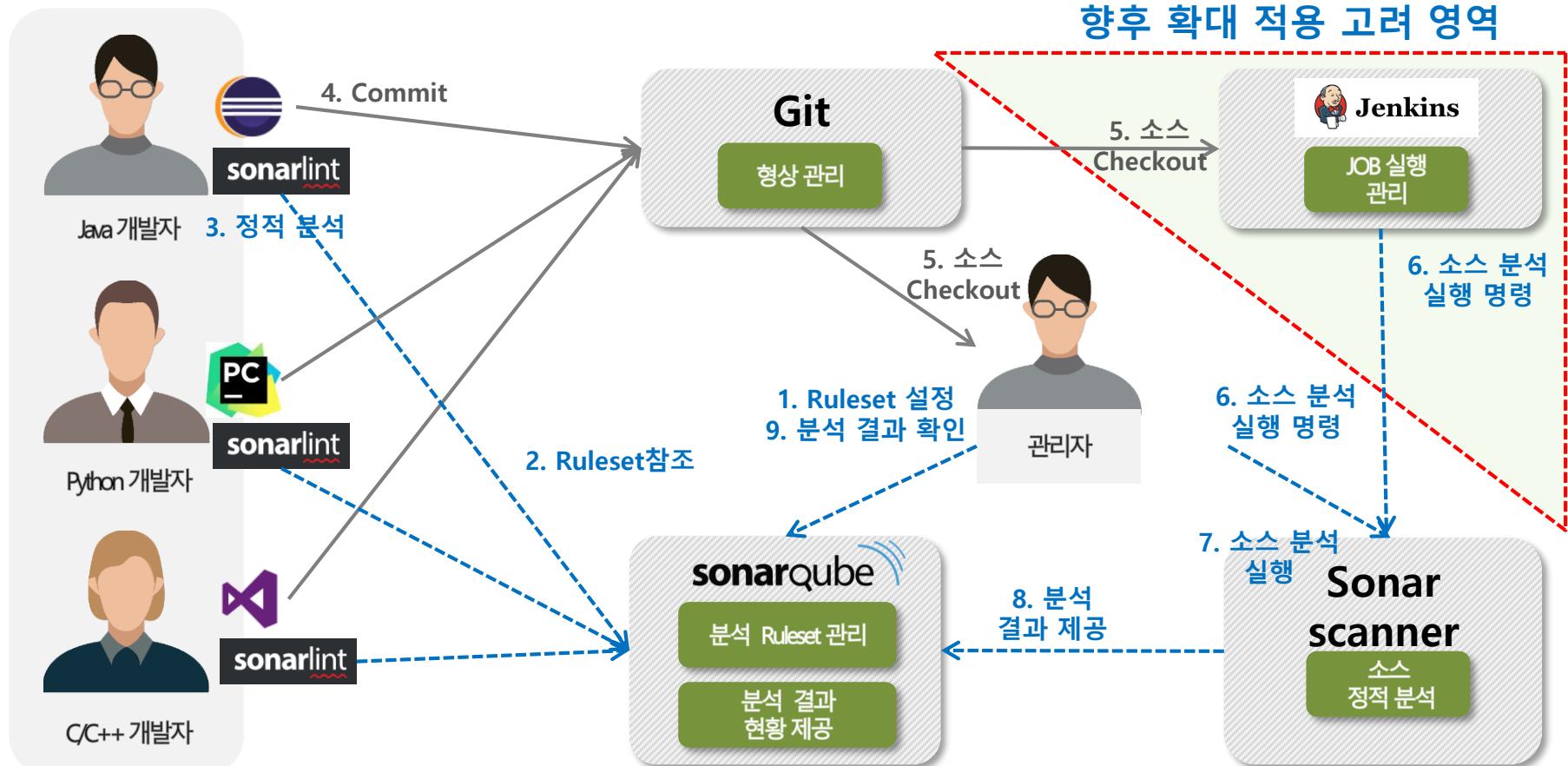
구분	Java	C/C++	C#	Python	Objective C
버그	118	73	74	13	47
코드악취	332	253	256	40	177
취약점	48	2	21	1	2
Security Hotspot	30	0	22	0	0
<b>총합</b>	<b>528</b>	<b>328</b>	<b>373</b>	<b>54</b>	<b>226</b>

[룰셋 구분]

# SonarQube 동작 구성

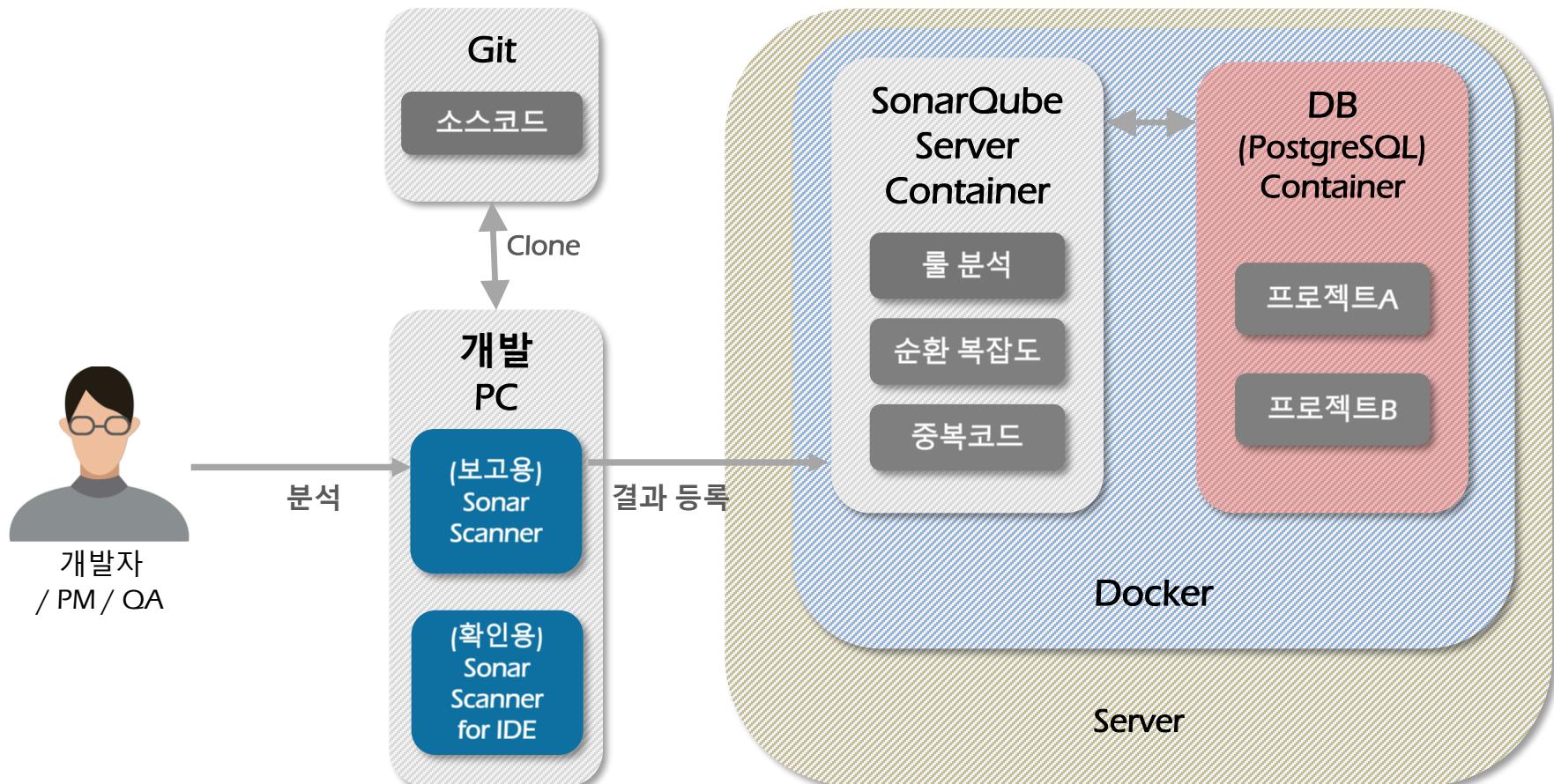
개발자는 IDE 플러그인, 관리자는 Sonar scanner를 사용하여 소스 정적 분석 수행

개발자는 IDE에서 바로 오류 조치, 관리자는 웹을 통하여(SonarQube) 분석 결과확인



# Docker를 이용한 SonarQube 구성도

제공된 리눅스 서버에 컨테이너 가상화 환경(Docker)을 구성하고,  
SonarQube와 DB를 컨테이너로 생성하여 운영



# 설치를 위한 다양한 방법

---

## □ 기본 설치 프로그램 활용

- 다운로드: <https://www.sonarqube.org/downloads/>
- 실행 방법
  - Bin 폴더의 OS 별 실행
- 단점
  - Java 11 요구 / H2 DB 사용 여부

## □ Docker 활용

- 다운로드
  - <https://github.com/SonarSource/docker-sonarqube/blob/master/example-compose-files/sq-with-postgres/docker-compose.yml>
- 실행 방법
  - docker-compose up
- 단점
  - Docker가 설치된 8GB 이상의 머신 필요

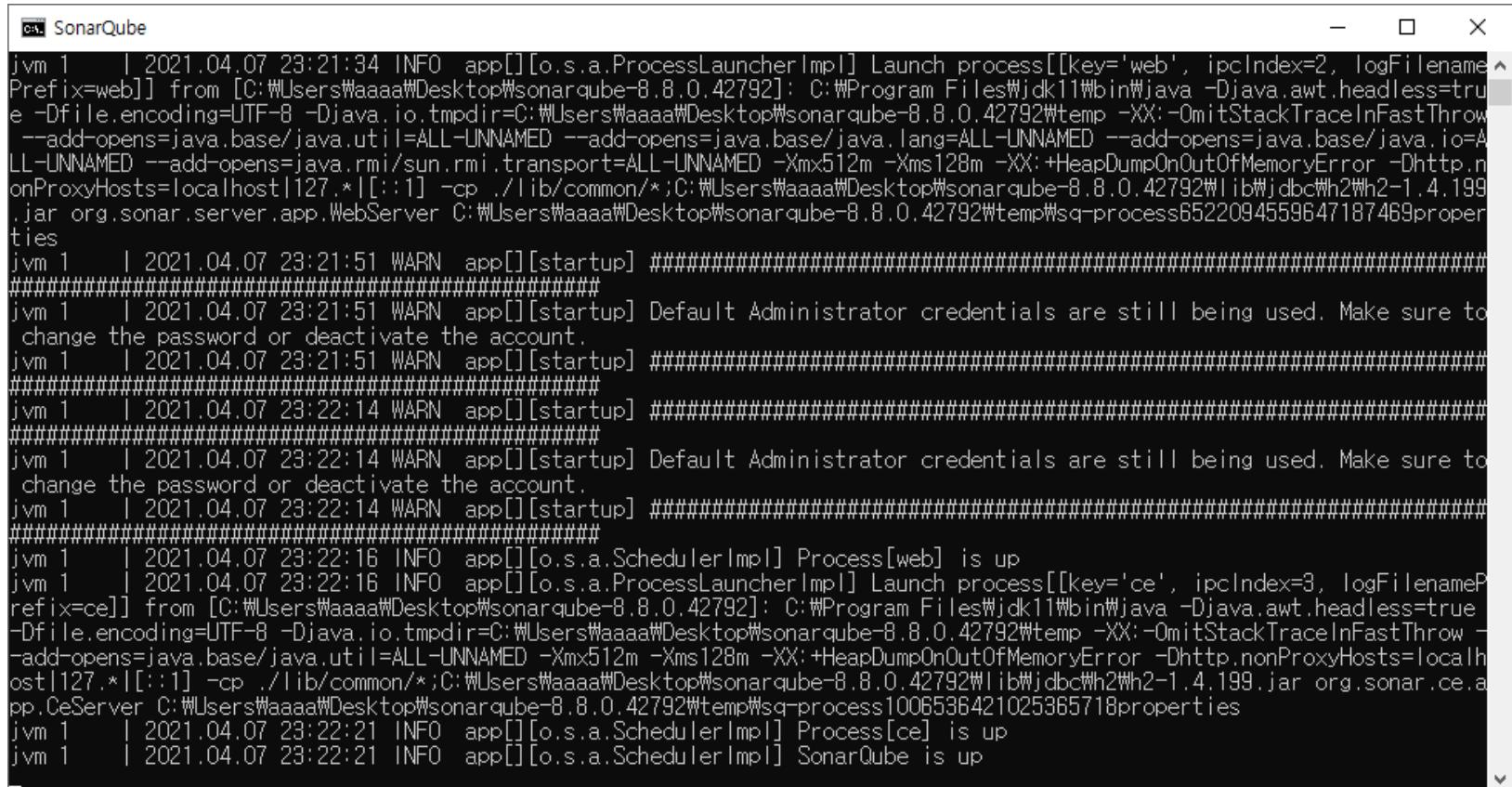
## □ 가상 머신(Virtual Box) 활용

- 다운로드: <https://bitnami.com/stack/sonarqube/virtual-machine> (Bitnami 패키지)
- 실행 방법
  - Virtual Box에서 OVA Import
- 단점
  - 운영 환경에서 사용 가능한지 검토 필요

# SonarQube 실행

## □ bin\windows-x86-64 폴더에 실행 파일 위치

- StartSonar 더블클릭하여 실행



The screenshot shows a Windows Task Manager window with a single process named 'SonarQube'. The window title bar says 'SonarQube'. The main area displays a log of Java process output. The log includes several INFO and WARN messages from the SonarQube application's startup process. Key messages include:

- INFO app[] [o.s.a.ProcessLauncherImpl] Launch process[[key='web', ipcIndex=2, logFilenamePrefix=web]] from [C:\Users\aaaaa\Desktop\sonarqube-8.8.0.42792]: C:\Program Files\jdk11\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=C:\Users\aaaaa\Desktop\sonarqube-8.8.0.42792\temp -XX:-OmitStackTraceInFastThrow --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED -Xmx512m -Xms128m -XX:+HeapDumpOnOutOfMemoryError -Dhttp.nonProxyHosts=localhost|127.\*|[::1] -cp ./lib/common/\*;C:\Users\aaaaa\Desktop\sonarqube-8.8.0.42792\lib\jdbch2-1.4.199.jar org.sonar.server.app.WebServer C:\Users\aaaaa\Desktop\sonarqube-8.8.0.42792\temp\sq-process6522094559647187469properties
- WARN app[] [startup] #######  
######\_ Default Administrator credentials are still being used. Make sure to change the password or deactivate the account.
- WARN app[] [startup] #######  
######\_ Default Administrator credentials are still being used. Make sure to change the password or deactivate the account.
- INFO app[] [o.s.a.SchedulerImpl] Process[web] is up
- INFO app[] [o.s.a.ProcessLauncherImpl] Launch process[[key='ce', ipcIndex=3, logFilenamePrefix=ce]] from [C:\Users\aaaaa\Desktop\sonarqube-8.8.0.42792]: C:\Program Files\jdk11\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=C:\Users\aaaaa\Desktop\sonarqube-8.8.0.42792\temp -XX:-OmitStackTraceInFastThrow --add-opens=java.base/java.util=ALL-UNNAMED -Xmx512m -Xms128m -XX:+HeapDumpOnOutOfMemoryError -Dhttp.nonProxyHosts=localhost|127.\*|[::1] -cp ./lib/common/\*;C:\Users\aaaaa\Desktop\sonarqube-8.8.0.42792\lib\jdbch2-1.4.199.jar org.sonar.ce.app.CeServer C:\Users\aaaaa\Desktop\sonarqube-8.8.0.42792\temp\sq-process1006536421025365718properties
- INFO app[] [o.s.a.SchedulerImpl] Process[ce] is up
- INFO app[] [o.s.a.SchedulerImpl] SonarQube is up

# JAVA가 제대로 설치되지 않은 경우

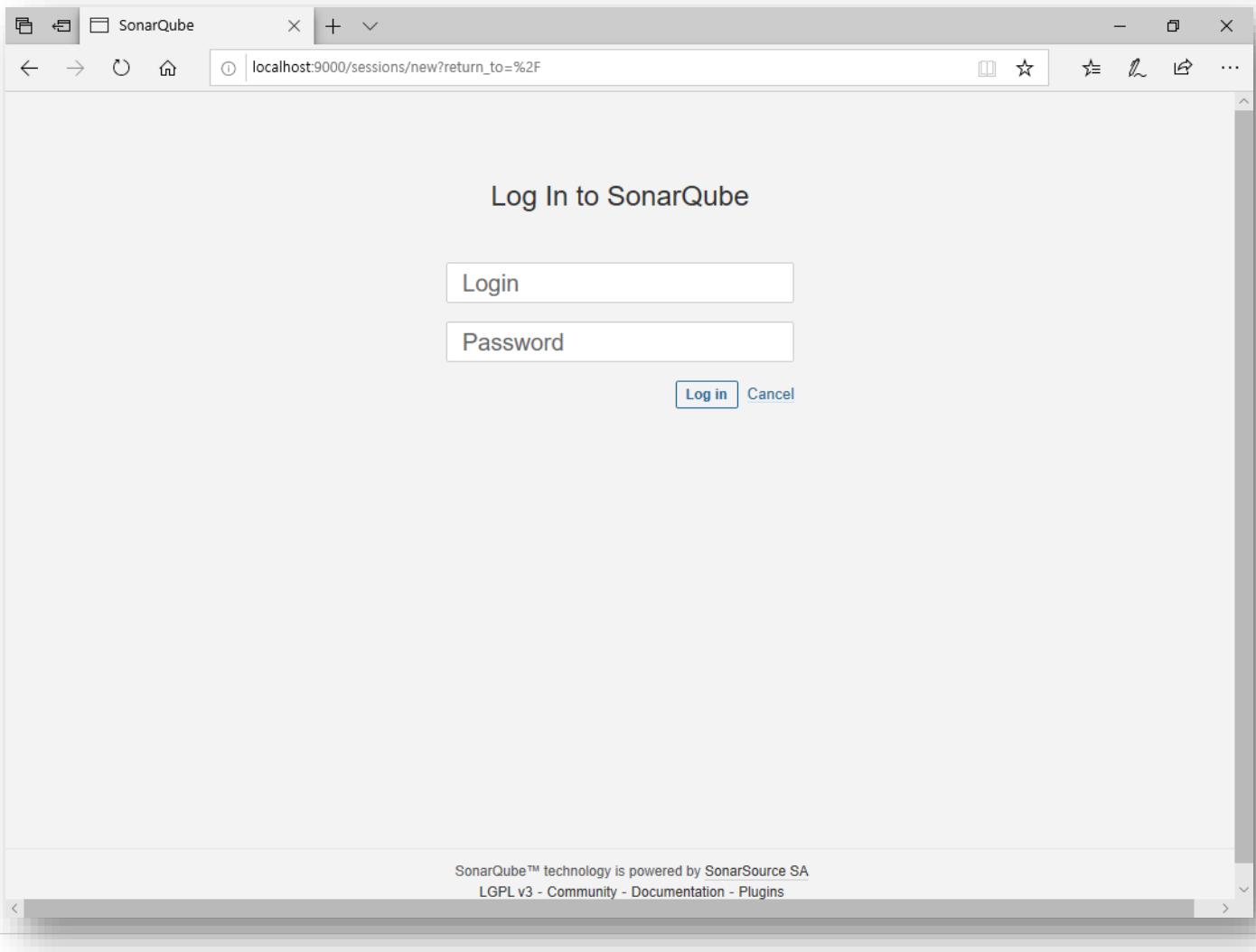
## □ 에러 발생

```
SonarQube
wrapper  --> Wrapper Started as Console
wrapper  Launching a JVM...
jvm 1   Wrapper (Version 3.2.3) http://wrapper.tanukisoftware.org
jvm 1   Copyright 1999-2006 Tanuki Software, Inc. All Rights Reserved.
jvm 1
jvm 1
jvm 1   WrapperSimpleApp: Encountered an error running main: java.lang.IllegalStateException: SonarQube requires Java
11 to run
jvm 1   java.lang.IllegalStateException: SonarQube requires Java_11 to run
jvm 1       at com.google.common.base.Preconditions.checkState(Preconditions.java:508)
jvm 1       at org.sonar.application.App.checkJavaVersion(App.java:93)
jvm 1       at org.sonar.application.App.start(App.java:56)
jvm 1       at org.sonar.application.App.main(App.java:87)
jvm 1       at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
jvm 1       at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
jvm 1       at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
jvm 1       at java.lang.reflect.Method.invoke(Unknown Source)
jvm 1       at org.tanukisoftware.wrapper.WrapperSimpleApp.run(WrapperSimpleApp.java:240)
jvm 1       at java.lang.Thread.run(Unknown Source)
wrapper  <-- Wrapper Stopped
계속하려면 아무 키나 누르십시오 . . . ■
```

## 초기 화면

### □ 접속은 IP:9000

- <http://localhost:9000>
- 초기 계정: admin / admin



## 초기 비밀번호 변경

Update your password

This account should not use the default password.

Enter a new password

All fields marked with \* are required

**Old Password \***

**New Password \***

**Confirm Password \***

**Update**

# 한국어 언어팩 설치

## □ Administration -> Marketplace에서 Korean Pack 선택

The screenshot shows the SonarQube Administration interface with the 'Marketplace' tab selected. A red box highlights the 'Korean Pack' entry in the list.

Category	Version	Description	Actions
Java I18n Rules / EXTERNAL ANALYSERS	0.1.0	Initial Release ... Code checks to find internationalization (i18n) in Java	<a href="#">Homepage</a> <a href="#">Install</a>
Jazz RTC / INTEGRATION	1.2	Compatibility with SQ 7.7+ ... Provides SCM Jazz RTC integration	<a href="#">Homepage</a> <a href="#">Issue Tracker</a> <a href="#">Install</a> Licensed under GNU LGPL 3
Korean Pack / LOCALIZATION	1.7.0	Support for SonarQube 7.0 ... Language Pack for Korean	<a href="#">Homepage</a> <a href="#">Issue Tracker</a> <a href="#">Install</a> Licensed under GNU LGPL 3 Developed by <a href="#">creatinnov</a>
L10n :: Spanish Pack / LOCALIZATION	1.14	Support for SonarQube 6.7+ ... Language Pack for Spanish	<a href="#">Homepage</a> <a href="#">Issue Tracker</a> <a href="#">Install</a> Licensed under GNU LGPL 3 Developed by <a href="#">excentia</a>
Mercurial / INTEGRATION	1.1.2	Compatible with SonarQube 6.7 LTS ... Provides SCM Mercurial integration	<a href="#">Homepage</a> <a href="#">Issue Tracker</a> <a href="#">Install</a> Licensed under GNU LGPL 3 Developed by <a href="#">SonarSource</a>
Mulesoft / COVERAGE	0.5 (BUILD NULL)	Initial release, includes basic support for analyzing Mulesoft test coverage reports. ... Coverage analyzer for Mulesoft Workflows	<a href="#">Homepage</a> <a href="#">Issue Tracker</a> <a href="#">Install</a> Licensed under GNU LGPL 3 Developed by <a href="#">SonarSource</a>

# 한국어 언어팩 적용 모습

## □ 브라우저 설정에 따라 언어 표시

The screenshot shows the SonarQube web interface with Korean localization applied. The top navigation bar includes links for '프로젝트' (Project), '이슈' (Issues), '코딩 규칙' (Coding Rules), '품질 프로파일' (Quality Profile), '품질 게이트' (Quality Gate), and '환경설정' (Environment Settings). The search bar at the top right contains the placeholder '프로젝트, 서브 프로젝트 및 파일'. The main content area displays various quality metrics and project analysis options, all in Korean.

관점: 전체 상태 정렬 기준: 이름

프로젝트 이름 혹은 키로 검색할 수 있습니다.

Once you analyze some projects, they will show up here.

Here is how you can analyse new projects

Create new project

0 projects

Filters

Quality Gate

- Success: 0
- Failed: 0

신뢰성 (Bugs)

- A: 0
- B: 0
- C: 0
- D: 0
- E: 0

보안성 (취약점)

- A: 0
- B: 0
- C: 0
- D: 0
- E: 0

유지보수성 (코드 악취)

- A: 0
- B: 0
- C: 0
- D: 0
- E: 0

커버리지

- ≥ 80%: 0

SonarQube™ technology is powered by SonarSource SA  
Community Edition - Version 8.0 (build 29455) - LGPL v3 - Community - Documentation - Get Support - Plugins - Web API - About

# 사용자 생성

## □ 환경설정 -> 시큐리티 -> 사용자 -> 사용자 생성

The screenshot shows the SonarQube web interface with the '환경설정' (Environment) tab selected in the top navigation bar. A modal dialog box titled '사용자 생성' (User Creation) is open in the center. The dialog contains fields for '로그인\*' (Login\*), '이름\*' (Name\*), 'Email', and '비밀번호\*' (Password\*). Below these fields is a section for 'SCM Accounts' with a '추가' (Add) button. A note states that login and email accounts are automatically linked to SCM accounts. At the bottom of the dialog are '생성' (Create) and '취소' (Cancel) buttons. The background shows a list of users, including one named 'admin'.

# 룰 설정의 필요성

---

## □ 조직/프로젝트 특성에 맞는 룰 선정 필요

- Java 기준, SonarQube는 500여개 룰이 있으며, 이 중 300여개를 기본 검사함
- **과연, 우리에게 모두 필요한 룰인가?**
  - 모든 룰은 유용하나, 우리의 현 특성에 딱 맞지는 않음

## □ 룰 선정 시 고려할 항목

- 처음 시작하는 조직인가?
- 반드시 지켜야 하는 룰이 있는가?
  - 보안을 위한 CERT, OWASP 적용
  - 계약 상 발주사의 요청
  - MISRA 등 표준 요건 만족 필요
- 유지보수를 고려해야 하는가?
- QA가 품질 목표(위반 목표)를 제시하는가?

# Quality Profile – SonarQube 룰셋

---

## □ 개요

- 우리말: 품질 프로파일
- SonarQube의 룰셋(룰의 모음) 설정
- 언어 별 여러 룰셋을 지정하고, 해당 프로젝트에서 사용할 수 있음
- 특정 룰셋을 지정하지 않을 경우, 기본값 룰셋 적용

## □ SonarQube의 기본 룰셋

- 각 언어 별 Sonar Way가 기본 룰셋
- Java의 경우, 약 350개의 룰을 기본 지정

## □ 우리만의 룰셋을 지정하기 위한 Tip

- Sonar Way를 복사
- 불필요한 룰을 비활성화 처리
- 우리 룰셋을 기본값(Default) 처리

# SonarQube 품질 프로파일 목록

The screenshot shows the SonarQube interface for managing quality profiles. The top navigation bar includes links for 'sonarqube', '프로젝트', '이슈', '코딩 규칙', '품질 프로파일', '품질 게이트', '환경설정', and '더보기'. The '품질 프로파일' link is highlighted with a red dashed box. The main content area displays five sections, each representing a different programming language or technology:

- Java, 1 프로파일**: Projects: 기본값 (391), Updated: 2개월 전, Used: Never. Options: gear icon.
- JavaScript, 2 프로파일**: Projects: 기본값 (101), Updated: 2개월 전, Used: Never. Options: gear icon.  
Sonar way Recommended: Projects: 기본값 (0), Rules: 141, Updated: 2개월 전, Used: Never. Options: gear icon.
- JSP, 1 프로파일**: Projects: 기본값 (0), Updated: 2개월 전, Used: Never. Options: gear icon.
- Kotlin, 1 프로파일**: Projects: 기본값 (31), Updated: 2개월 전, Used: Never. Options: gear icon.
- PHP, 3 프로파일**: Drupal: Projects: 0, Rules: 21, Updated: 2개월 전, Used: Never. Options: gear icon.  
PSR-2: Projects: 0, Rules: 20, Updated: 2개월 전, Used: Never. Options: gear icon.  
Sonar way: Projects: 기본값 (107), Rules: 21, Updated: 2개월 전, Used: Never. Options: gear icon.

# 품질 프로파일 복사

## □ Java - Sonar Way의 복사

The screenshot shows the SonarQube interface for managing profiles. At the top, there are two tabs: 'Java, 1 프로파일' and 'JavaScript, 2 프로파일'. Below these are sections for 'Projects', 'Rules', 'Updated', and 'Used'. A dropdown menu is open next to the 'Used' section, showing options: '비교' (Compare), '복사' (Copy), and 'Extend'. The '복사' option is highlighted.



The dialog box has the following content:

다음 프로파일을 복사합니다: **Sonar way**

새 이름\*

**복사** **취소**

# 복사한 품질 프로파일 Main 화면

The screenshot shows the SonarQube Quality Profile main page for the 'Synetics Way' profile. The page has a dark header with tabs for '프로젝트', '이슈', '코딩 규칙', '품질 프로파일' (which is selected), '품질 게이트', '환경설정', and '더보기'. The title bar shows the URL: 104.46.217.69/profiles/show?language=java&name=Synetics+Way.

**품질 프로파일 / Java**

**Synetics Way**

업데이트됨: 34초 전 사용됨: Never

**규칙 활성화**

규칙	활성화	비활성화
Total	391	190
Bugs	113	12
Vulnerabilities	36	8
Code Smells	212	168
Security Hotspots	30	2

[더 많은 규칙 활성화](#)

**프로파일 상속**

Synetics Way 391개의 규칙 활성화 0개의 오버라이딩 된 규칙

[부모 프로파일 변경](#)

**프로젝트**

이 프로파일을 명시적으로 할당한 프로젝트가 없습니다.

[프로젝트 변경](#)

**권한**

글로벌 "Manage Quality Profile" 권한을 가진 사용자들은 이 quality profile를 관리할 수 있습니다.

[더 많은 사용자에게 권한 부여](#)

SonarQube™ technology is powered by SonarSource SA  
Community Edition - Version 7.9.1 (build 27448) - LGPL v3 - Community - Documentation - Get Support - Plugins - Web API - About

# 불필요 룰 비활성화

## □ Code Smell의 Minor 를 비활성화

- “대규모 변경”을 통해, 선택된 룰 전체를 한 번에 비활성화 함

Junit4 [Jenkins] 코딩 규칙

sonarqube 프로젝트 이슈 코딩 규칙 품질 프로파일 품질 게이트 환경 설정 더보기

Filters 모든 필터 초기화 대규모 변경

규칙으로 검색... 다음에서 활성화... 다음에서 비활성화... 다음에서 비활성화 Synetics Way

Language

タ입 coding\_rules.facets.languagescoding\_ru

Bug 17  
Vulnerability 7  
Code Smell 81  
Security Hotspot 4

Tag  
Repository

Default Severity Minor 초기화  
Blocker 15 Critical 34  
Major 80 Minor 81  
Info 2

Status  
Security Category  
Available Since  
Template  
Quality Profile Synetics... 초기화

Inheritance  
Activation Severity

or "@Nullable" should not be used on primitive types Java Code Smell 비활성화

"@Deprecated" code should not be used Java Code Smell cert, cwe, obsolete 비활성화

"catch" clauses should do more than rethrow Java Code Smell clumsy, error-handling, finding, unused 비활성화

"close()" calls should not be redundant Java Code Smell redundant 비활성화

"Collections.EMPTY\_LIST", "EMPTY\_MAP", and "EMPTY\_SET" should not be used Java Code Smell obsolete, pitfall 비활성화

"equals(Object obj)" should be overridden along with the "compareTo(T obj)" method Java Code Smell 비활성화

"finalize" should not set fields to "null" Java Code Smell clumsy, performance 비활성화

"indexOf" checks should use a start position Java Code Smell confusing 비활성화

"private" methods called only by inner classes should be moved to those classes Java Code Smell confusing 비활성화

"read(byte[],int,int)" should be overridden Java Code Smell performance 비활성화

"StandardCharsets" constants should be preferred Java Code Smell clumsy, java7 비활성화

"Stream" call chains should be simplified when possible Java Code Smell clumsy 비활성화

"switch" statements should have at least 3 "case" clauses Java Code Smell bad-practice 비활성화

"ThreadLocal.withInitial" should be preferred Java Code Smell java8 비활성화

# 불필요 룰 비활성화와 기본값 설정

## □ 비활성화 확인



The screenshot shows the SonarQube interface with two sections: Java (2 profiles) and JavaScript (2 profiles). In the Java section, the "Synetics Way" profile is selected. A context menu is open over the "기본값" button for this profile, listing options: "더 많은 규칙 활성화", "백업하기", "비교", "복사", "Extend", "Rename", "기본값 설정" (selected), and "삭제".

프로젝트	Rules	Updated	Used
Sonar way (Built-in)	기본값 391	2개월 전	Never
Synetics Way	0 310	9초 전	Never

프로젝트	Rules	Updated
Sonar way (Built-in)	기본값 101	2개월 전
Sonar way Recommended (Built-in)	0 141	2개월 전

# SonarScanner 다운로드

## □ SonarScanner란?

- SonarQube의 기본 분석기
- 빌드를 Hook 형식으로 분석하는 C/C++/C# 등을 제외한 다른 언어에 적용
  - PHP, JS, Python, TS, Ruby 등
- OS 별 실행파일 제공

## □ 동작 방식

- SonarScanner에서 분석을 실행
- 결과를 SonarQube 서버에 지정된 프로젝트로 전송

### 2 프로젝트 분석 실행하기

프로젝트는 주로 어떤 언어로 구현되어 있습니까?

[Java](#) [C# 혹은 VB.NET](#) [기타 \(JS, Python, PHP, ...\)](#)

운영체제는 무엇을 사용하십니까?

[Linux](#) [Windows](#) [macOS](#)

Windows 용 SonarQube Scanner를 다운로드하고 압축을 풁니다

And add the `bin` directory to the `%PATH%` environment variable

[다운로드](#)

# 분석 실행

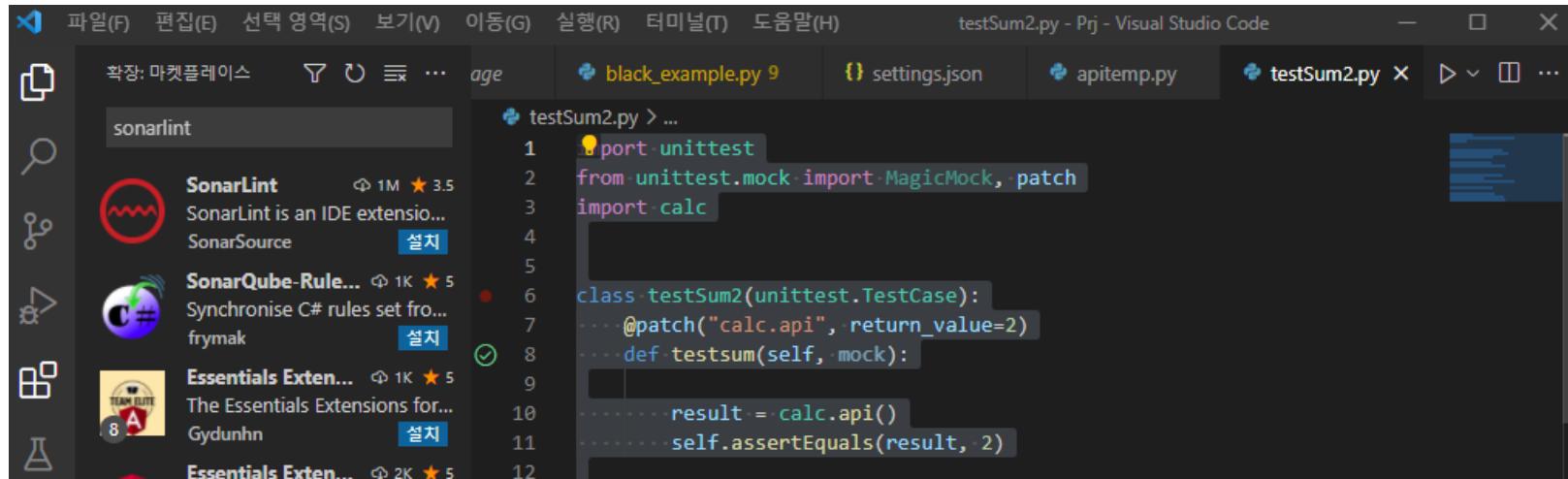
---

## □ 분석 실행 명령 예제

- 프로젝트 root 폴더에서 실행
- 예제: sonar-scanner.bat -D"sonar.projectKey=JSProject" -D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -D"sonar.login=1345f1196ab3ae5d70d5aa0a46b0c851dc349951"

# VSCode에서 사용

## □ 확장에서 “sonarlint” 검색 후 설치



# VSCode에서 사용

## □ 코드 내 문제점을 표시 및 조치 가이드

The screenshot shows the Visual Studio Code interface with the following details:

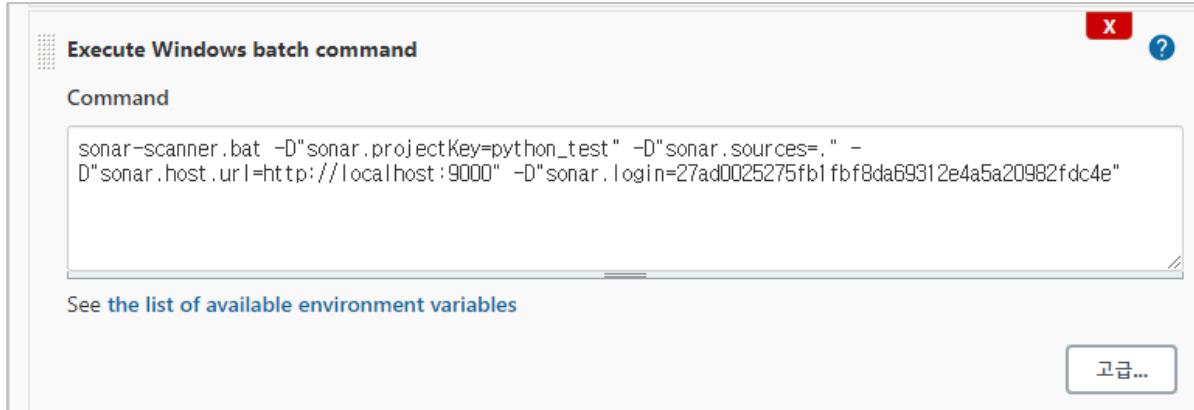
- File Explorer:** Shows the project structure under "DEVOPS\_TEST". The file "test\_calcAPI.py" is selected.
- Code Editor:** Displays Python test code for "calc.py".

```
1 import unittest
2 from unittest.mock import MagicMock, patch
3 import calc
4
5
6 class testSum2(unittest.TestCase):
7     @patch("calc.api", return_value=2)
8     def testsum2(self, mock):
9
10         result = calc.api()
11         self.assertEqual(result, 2)
12
13
14 class testSum3(unittest.TestCase):
15     @patch("calc.api")
16     def testsum3(self, mock):
17
18         mock.return_value = 2
19         result = calc.sumWithApi(5)
20         self.assertEqual(result, 7)
```
- Problems View:** Shows 2 errors for "test\_calcAPI.py":
  - Rename class "testSum2" to match the regul... sonarlint(python:S101) [6, 7]
  - Rename class "testSum3" to match the regul... sonarlint(python:S101) [14, 7]
- Bottom Status Bar:** Shows "main\*" file, Python 3.10.2 64-bit, and other status information.

# Jenkins에서 실행

## □ Execute Windows Batch Command 를 추가

## □ 앞서 살펴본 sonar-scanner 실행 명령 추가



## □ SonarQube 분석 실행 화면

The screenshot shows the Jenkins log output for a SonarQube analysis. The log starts with:

```
Writing XML report to coverage.xml
```

Then it shows the command executed:

```
C:\Jenkins\Work\workspace\PythonTest>exit 0  
[PythonTest] $ cmd /c call C:\Windows\TEMP\jenkins3356154065900391504.bat
```

Following that, the SonarScanner logs are displayed:

```
C:\Jenkins\Work\workspace\PythonTest>C:\DevTools\sonar-scanner\bin\sonar-scanner.bat -D"sonar.projectKey=python_test" -D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -D"sonar.login=27ad0025275fb1fbf8da69312e4a5a20982fdc4e"  
INFO: Scanner configuration file: C:\DevTools\sonar-scanner\bin..\conf\sonar-scanner.properties  
INFO: Project root configuration file: NONE  
INFO: SonarScanner 4.6.2.2472  
INFO: Java 11.0.11 AdoptOpenJDK (64-bit)  
INFO: Windows 10 10.0 amd64  
INFO: User cache: C:\Windows\system32\config\systemprofile\.sonar\cache  
INFO: Scanner configuration file: C:\DevTools\sonar-scanner\bin..\conf\sonar-scanner.properties  
INFO: Project root configuration file: NONE  
INFO: Analyzing on SonarQube server 9.3.0  
INFO: Default locale: "ko_KR", source code encoding: "x-windows-949" (analysis is platform dependent)  
INFO: Load global settings
```

# 기본 분석 실행 결과

The screenshot shows the SonarQube interface for the project 'python\_test' in the 'master' branch. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and a search bar. A message indicates 'Last analysis had 2 warnings' from February 17, 2022, at 7:42 AM.

The main content area displays the Quality Gate Status as 'Passed' with the message 'All conditions passed.' Below this, the 'MEASURES' section is shown under the 'Overall Code' tab, with tabs for 'New Code' and 'Overall Code'. The measures include:

- Bugs: 0, Reliability: A
- Vulnerabilities: 0, Security: A
- Security Hotspots: 0, Security Review: A
- Debt: 20min, Code Smells: 3, Maintainability: A
- Coverage: 0.0% (29 lines to cover), Unit Tests: -
- Duplications: 0.0% (89 lines), Duplicated Blocks: 0

**End of Document**

---

**Any Question?**