

한 걸음 앞선 개발자가 지금 꼭 알아야 할 **클로드 코드**

한 걸음 앞선 개발자가

지금 꼭 알아야 할

MASTERING
CLAUDE
CODE

조훈, 정찬훈 지음

클로드

코드

CLAUDE
CODE

실무에서 검증된 개발 방식 그대로,

매일 1시간 4주

Claude Code 에이전트 실전 훈련!

```
# 설치(macOS, Windows)
# CLAUDE.md 설정
# MCP 연동/다양한 활용 전략
# 클로드 워크플로 전략
# 설계 → 부트스트래핑 →
  테스트 → 개선 → 명세
# 클로드 코드의 효율을 극대화
  하는 다양한 방법
```

갈벗

깃허브: https://github.com/sysnet4admin/_Book_Claude-Code

MCP: 깃허브를 위한 설정 (로컬)

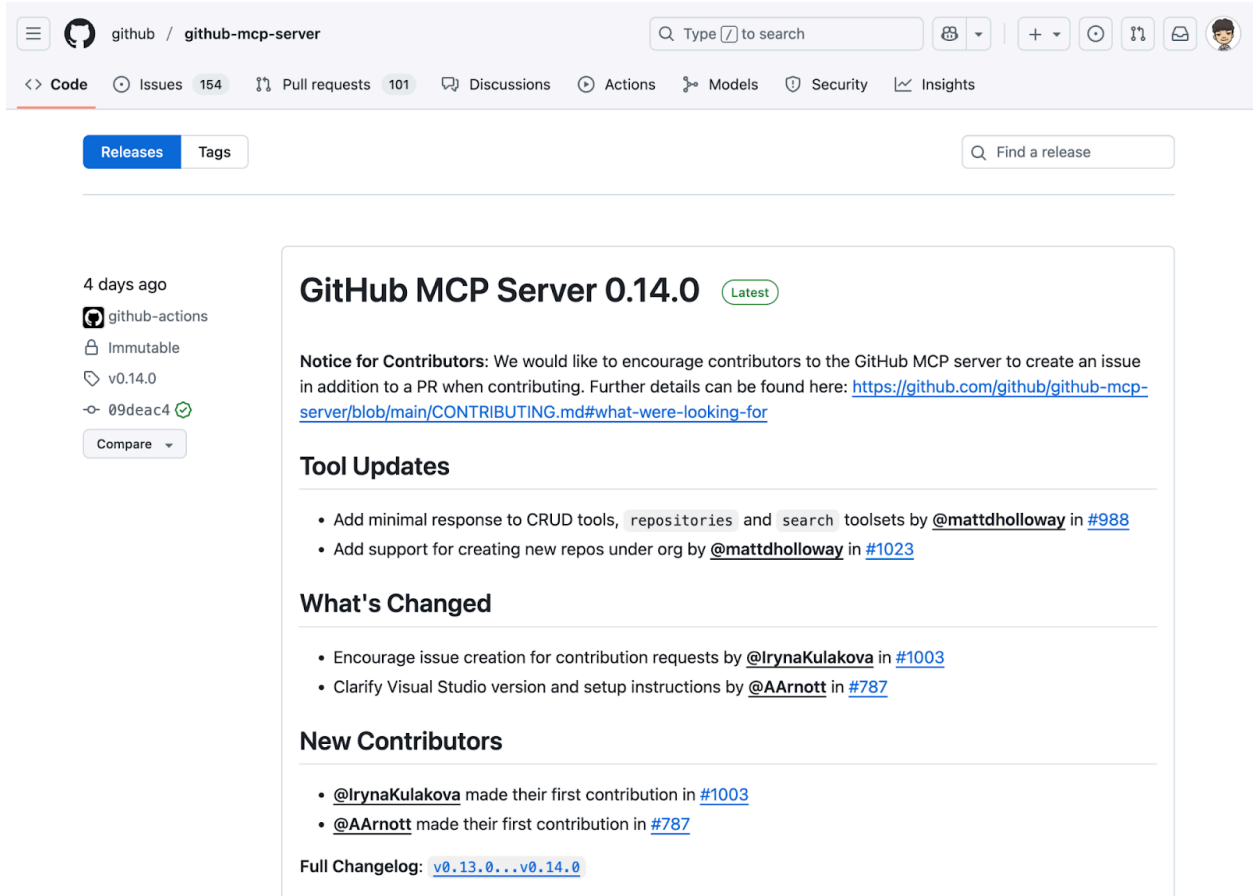
깃허브 MCP를 사용하는 대표적인 방법은 로컬에 구현하는 방법과 원격지 서버를 이용하는 방법입니다. 완벽한 구성법은 존재하지 않으며, 각 방법은 가용성과 보안 등에 뚜렷한 장단점이 있으므로 현재 구성에 적합한 방법을 선택하는 것이 좋습니다.

| 구분 | 로컬 MCP 구성 | 원격 MCP 구성 |
|-------|-------------------------|------------------------|
| 응답 속도 | ✅ 매우 빠름 (네트워크 지연 없음) | ❌ 네트워크 지연으로 상대적으로 느림 |
| 보안성 | ✅ 매우 높음 (데이터 외부 유출 없음) | ❌ 상대적으로 낮음 (외부 전송 위험) |
| 비용 | ✅ 초기 구축 후 운영비 거의 없음 | ❌ 지속적인 클라우드/네트워크 비용 |
| 확장성 | ❌ 로컬 하드웨어에 제한됨 | ✅ 거의 무제한 확장 가능 |
| 접근성 | ❌ 특정 기기에서만 접근 가능 | ✅ 인터넷 연결만 있으면 어디서든 접근 |
| 협업 | ❌ 동시 접근 어려움 | ✅ 다수 사용자 동시 접근 용이 |
| 안정성 | ✅ 네트워크 상태에 무관 | ❌ 인터넷 연결 필수 |
| 유지보수 | ❌ 직접 관리 필요 (업데이트, 백업 등) | ✅ 자동 관리 (업데이트, 백업 자동화) |
| 보안 | ✅ 상대적으로 안전한 데이터 보안 | ❌ 서비스 제공업체에 데이터가 노출됨 |
| 처리 능력 | ❌ 로컬 시스템 성능에 제한 | ✅ 고성능 클라우드 리소스 활용 |
| 가용성 | ❌ 로컬 시스템 장애 시 중단 | ✅ 높은 가용성 (이중화, 백업 시스템) |
| 초기 구축 | ❌ 복잡한 설정과 구축 필요 | ✅ 상대적으로 간단한 설정 |

[표 1] MCP를 로컬에 구현하는 것과 원격지에 구현하는 것에 대한 비교

로컬에서 사용하는 방법은 다시 크게 도커(Docker)를 이용하는 방법과 서비스 업체에서 제공하는 방법을 이용하는 것으로 나눌 수 있습니다. 도커(Docker)의 경우는 책에서 충분히 설명했으니 여기서 다루지 않고, 다른 방법인 서비스 업체에서 제공하는 방법에 대한 사전 설정을 여기서 다루겠습니다.

깃허브의 경우 사용자의 편의를 위해 직접 사용할 수 있는 바이너리 파일을 제공하며, 플랫폼 별로 내려받을 수 있습니다.



[그림 1] 깃허브 MCP 릴리스 페이지 (<https://github.com/github/github-mcp-server/releases>)

이 문서에서는 깃허브에서 제공하는 MCP를 위한 바이너리 파일을 통한 연결 방법을 설명합니다.

1. 현재 설정되어 있는 mcp가 있는지 **claude mcp list** 명령으로 확인합니다.

<Terminal박스>

```
$ claude mcp list
```

```
No MCP servers configured. Use `claude mcp add` to add a server.
```

</Terminal박스>

2. 연동을 위해서 깃허브 토큰이 필요합니다. 이는 책에서 도커를 통해서 진행하는 과정에서 개인용 접근 토큰(Personal Access Token)을 발행해서 사용하므로, 여기서는 단순히 해당 부분을 확인만 하겠습니다.

<Terminal박스>

```
$ export | grep TOKEN
```

```
GITHUB_TOKEN=ghp_Ng4TuDQUs2Xqb2ukRRta7mdM7TP8Ay3u3wac # 노출되면 안되는 정보
```

</Terminal박스>

<노트> 생성된 깃허브 토큰이 정상적인지 확인하려면?

다음의 curl 명령을 실행해서 응답이 온다면 정상적인 상태입니다.

<Terminal박스>

```
$ curl -H "Authorization: token $GITHUB_TOKEN" https://api.github.com/user
```

```
{  
  "login": "sysnet4admin",
```






















[생략]

</Terminal박스>

</노트>

3. 각 호스트 플랫폼에 맞는 릴리즈 버전을 내려받기 위해 깃허브 MCP 릴리스 페이지를 접속하고 파일들을 확인합니다. 그리고 플랫폼에 맞는 주소를 클립보드에 복사합니다.

▼ Assets 11

| | | | | |
|--|----------------------|---|-----------|------------|
|  github-mcp-server_0.14.0_checksums.txt | sha256:01d8f38d40... |  | 824 Bytes | 4 days ago |
|  github-mcp-server_Darwin_arm64.tar.gz | sha256:3059ebf574... |  | 4.3 MB | 4 days ago |
|  github-mcp-server_Darwin_x86_64.tar.gz | sha256:324df20ad0... |  | 4.57 MB | 4 days ago |
|  github-mcp-server_Linux_arm64.tar.gz | sha256:1c069a457b... |  | 4.11 MB | 4 days ago |
|  github-mcp-server_Linux_i386.tar.gz | sha256:61bf612c0d... |  | 4.19 MB | 4 days ago |
|  github-mcp-server_Linux_x86_64.tar.gz | sha256:5c05347bca... |  | 4.47 MB | 4 days ago |
|  github-mcp-server_Windows_arm64.zip | sha256:95dfbe2360... |  | 4.17 MB | 4 days ago |
|  github-mcp-server_Windows_i386.zip | sha256:49ee325c51... |  | 4.38 MB | 4 days ago |
|  github-mcp-server_Windows_x86_64.zip | sha256:e2078e683b... |  | 4.6 MB | 4 days ago |
|  Source code (zip) | | | | 4 days ago |
|  Source code (tar.gz) | | | | 4 days ago |
|  Release attestation (json) | | | | 4 days ago |

  9  3  6  4  7 18 people reacted

[그림 2] 깃허브 MCP 릴리스 페이지에 있는 각 플랫폼 별 파일

4. curl 명령이 복사한 주소를 넣어서 플랫폼에 맞는 릴리스 파일을 내려 받습니다.

<Terminal박스>

```
$ curl -LO https://github.com/github/github-mcp-server/releases/download/v0.14.0/github-mcp-server_Darwin_arm64.tar.gz
```

| % Total | % Received | % Xferd | Average Speed | Time | Time | Time | Current |
|---------|------------|---------|-------------------------------------|------|------|-------|---------|
| | | | Dload Upload Total Spent Left Speed | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 100 | 4403k | 100 | 4403k | 0 | 0 | 17.2M | 0 |

</Terminal박스>

5. 압축 파일을 해제한 후에 **github-mcp-server** 바이너리에 실행 권한을 줍니다.

<Terminal박스>

```
$ tar -xvzf github-mcp-server_Darwin_arm64.tar.gz
```

x LICENSE
x README.md
x github-mcp-server

```
$ chmod +x github-mcp-server
```

</Terminal박스>

6. 셸 어느 위치에서도 바이너리를 호출할 수 있도록 **/usr/local/bin/**로 옮깁니다. 옮긴 후에 동작 확인을 위해 **github-mcp-server -v**을 입력하고 아래와 비슷한지 확인합니다.

<Terminal박스>

```
$ sudo mv github-mcp-server /usr/local/bin/
```

Password: <암호 입력>

```
$ github-mcp-server -v
```

GitHub MCP Server
Version: 0.14.0
Commit: 09deac45d4f0bf00d8d78d41334267a594f92935
Build Date: 2025-09-03T13:34:46Z

</Terminal박스>

7. 로컬 바이너리 파일을 이용해서 깃허브 MCP에 연결할 준비는 모두 완료하였습니다. **claude mcp add** 명령을 이용해서 추가합니다. (현재 부분은 책에도 포함되어 있습니다.)

<Terminal박스>

```
$ claude mcp add github -s user -e  
GITHUB_PERSONAL_ACCESS_TOKEN=${GITHUB_TOKEN} -- github-mcp-server stdio  
Added stdio MCP server github with command: github-mcp-server stdio to user config  
File modified: /Users/hj/.claude.json
```

</Terminal박스>

8. 추가된 깃허브 MCP 서버를 확인하고, 책처럼 실행해 봅니다.

<Terminal박스>

```
$ claude mcp list  
Checking MCP server health...  
  
github: github-mcp-server stdio - ✓ Connected
```

```
$ claude "현재 열려 있는 PR을 분석해줘"
```

```
┌  
| ✧ Welcome to Claude Code! |  
| |  
| /help for help, /status for your current setup |  
| |  
| cwd: /Users/hj/_Book_Claude-Code/week2/Fri |  
└
```

Tips for getting started:

Run /init to create a CLAUDE.md file with instructions for Claude
Use Claude to help with file analysis, editing, bash commands and git
Be as specific as you would with another engineer for the best results

✓ Run /terminal-setup to set up terminal integration

> 현재 열려 있는 PR을 분석해줘

</Terminal박스>

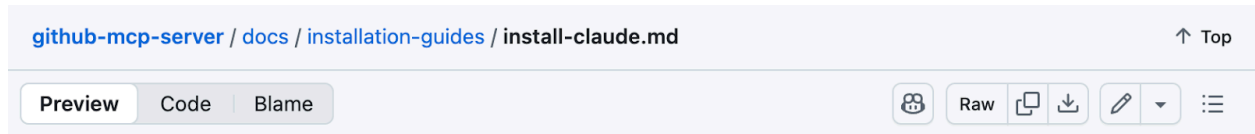
MCP: 깃허브를 위한 설정 (원격지)

책에서는 다루지 않았지만, 원격지 설정도 간단히 언급하면 좋을 것 같아서 추가 설명합니다.

대부분의 SaaS형 서비스 업체는 MCP 연결을 위한 정보를 제공합니다.

따라서 이렇게 구성하는 경우 더 편리할 수 있습니다. 하지만 이미 로컬과 원격지에 대한 비교를 통해 배운 것처럼 정보가 외부로 나간다는 것에서 문제가 될 수 있으므로 선택적으로 구성해서 사용해야 합니다.

또한 원격지 설정은 매우 자주 변경될 수 있는 정보이므로, 문제 발생시 서비스 제공 업체의 공식 정보를 확인하고 수정해야 합니다.



Remote Server Setup (Streamable HTTP)

1. Run the following command in the Claude Code CLI

```
ttp github https://api.githubcopilot.com/mcp -H "Authorization: Bearer YOUR_GITHUB_PAT"
```

With an environment variable:

```
thubcopilot.com/mcp -H "Authorization: Bearer $(grep GITHUB_PAT .env | cut -d '=' -f2)"
```

2. Restart Claude Code
3. Run `claude mcp list` to see if the GitHub server is configured

[그림 3] 깃허브 MCP 서버에 원격지 연결하기 위한 정보를 [깃허브 mcp 서버 페이지](#)에서 제공함

1. 현재 설정되어 있는 github 라는 다른 이름으로 구성하기 위해 **github-mcp-remote**라는 이름으로 변경하고 다음과 같이 입력합니다.

<Terminal박스>

```
$ claude mcp add github-mcp-remote --transport http https://api.githubcopilot.com/mcp -s user -H "Authorization: Bearer ${GITHUB_TOKEN}"
```

Added HTTP MCP server github-mcp-remote with URL: https://api.githubcopilot.com/mcp to user config

Headers: {

```
"Authorization": "Bearer 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"
```

}

File modified: /Users/hj/.claude.json

</Terminal박스>

2. 추가된 원격지 깃허브 MCP 서버를 확인합니다.

<Terminal박스>

```
$ claude mcp list
```

```
Checking MCP server health...
```

```
github: github-mcp-server stdio - ✓ Connected
```

```
github-mcp-remote: https://api.githubcopilot.com/mcp (HTTP) - ✓ Connected
```

</Terminal박스>

<노트> 여러 개의 MCP 서버가 있는 경우 클로드 코드는 어디에 물어보나요?

클로드 코드는 자동으로 최적의 MCP 서버에 물어봅니다. 좀 더 지정하고자 한다면, 질의하는 문장에 MCP 서버의 이름을 넣어서 지정하면 해당 MCP에 물어볼 가능성을 높일 수 있습니다.

<Terminal박스>

```
$ claude "***github-mcp-remote** MCP 서버로 현재 열려 있는 PR을 분석해줘"
```

</Terminal박스>

</노트>