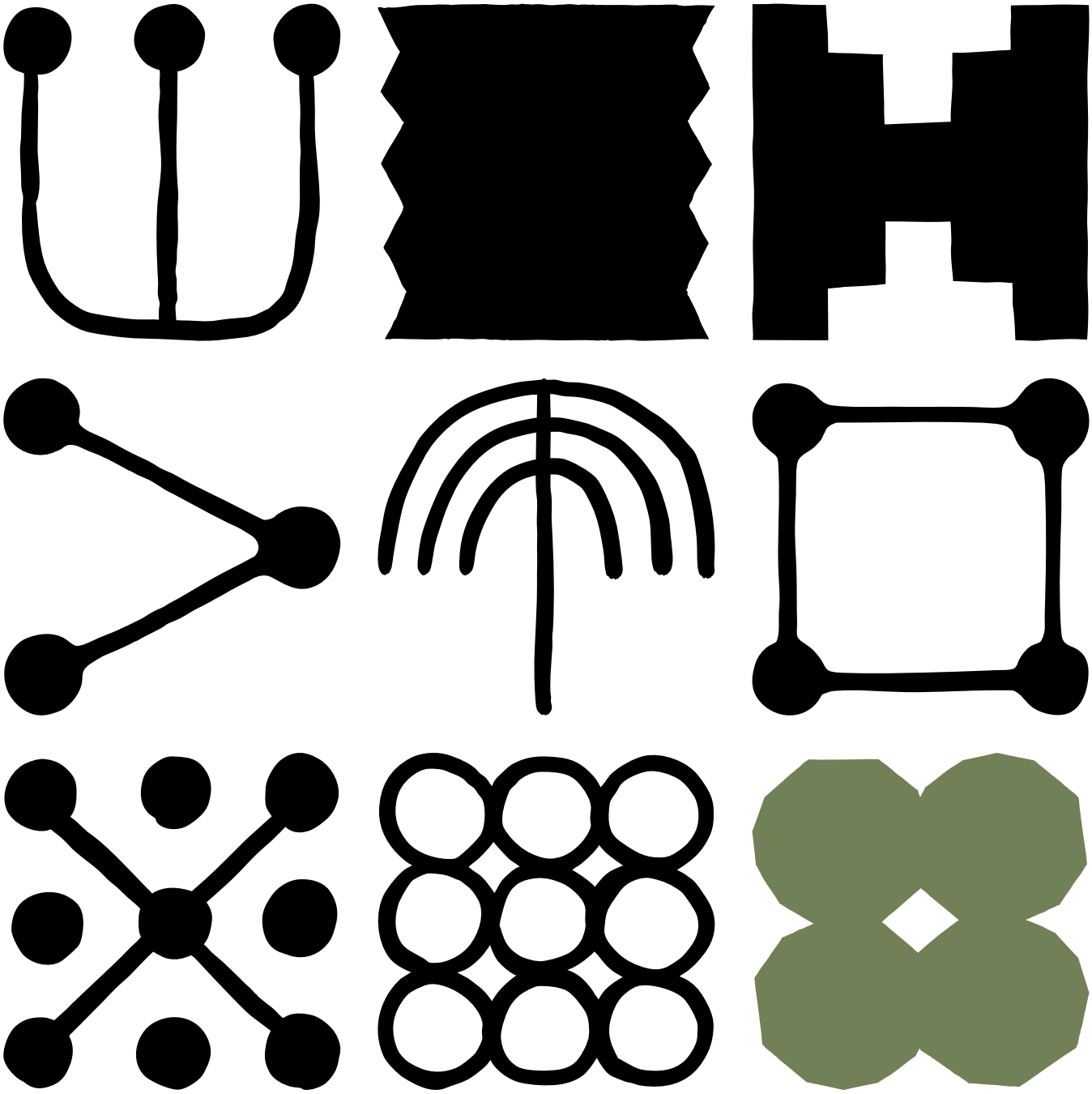


권한 프롬프트를 넘어서: Claude Code를 더욱 안전하고 자율적으로 만들기

게시됨2025년 10월 20일

Claude Code의 새로운 샌드박싱 기능인 bash 도구와 웹상의 Claude Code는 파일시스템과 네트워크 격리라는 두 가지 경계를 통해 권한 프롬프트를 줄이고 사용자 안전성을 높입니다.

카테고리Claude Code, Claude는 당신과 함께 코드를 작성하고, 테스트하고, 디버깅하며, 코드베이스를 탐색하고, 여러 파일을 편집하고, 작업을 검증하기 위해 명령을 실행합니다. Claude에게 코드베이스와 파일에 대한 이러한 광범위한 접근 권한을 부여하는 것은 특히 프롬프트 인젝션의 경우 위험을 초래할 수 있습니다.

이를 해결하기 위해 샌드박싱을 기반으로 구축된 Claude Code에 두 가지 새로운 기능을 도입했습니다. 이 두 기능 모두 개발자들에게 더 안전한 작업 환경을 제공하는 동시에 Claude가 더 자율적으로 작동하고 권한 요청을 줄일 수 있도록 설계되었습니다. 내부 사용에서 샌드박싱이 권한 요청을 84% 안전하게 줄인다는 것을 확인했습니다. Claude가 자유롭게 작업할 수 있는 명확한 경계를 정의함으로써 보안과 자율성을 모두 향상시킵니다.

Claude Code에서 사용자 보안 유지

Claude Code는 권한 기반 모델로 작동합니다: 기본적으로 읽기 전용이며, 이는 수정을 하거나 명령을 실행하기 전에 권한을 요청한다는 의미입니다. 여기에는 몇 가지 예외가 있습니다: echo나 cat과 같은 안전한 명령은 자동으로 허용하지만, 대부분의 작업은 여전히 명시적인 승인이 필요합니다.

지속적으로 "승인"을 클릭하는 것은 개발 주기를 늦추고 '승인 피로'를 유발할 수 있으며, 이로 인해 사용자가 승인하는 내용에 주의를 기울이지 않게 되어 결과적으로 개발을 덜 안전하게 만들 수 있습니다.

이를 해결하기 위해 Claude Code용 샌드박싱을 출시했습니다.

샌드박싱: 더 안전하고 자율적인 접근 방식

샌드박싱은 Claude가 각 작업에 대해 허가를 요청하는 대신, 미리 정의된 경계 내에서 더 자유롭게 작업할 수 있는 환경을 만듭니다. 샌드박싱이 활성화되면 허가 요청이 대폭 줄어들고 안전성이 향상됩니다.

우리의 샌드박싱 접근 방식은 운영 체제 수준의 기능을 기반으로 구축되어 두 가지 경계를 제공합니다:

1. **파일시스템 격리**, 클로드가 특정 디렉토리에만 접근하거나 수정할 수 있도록 보장합니다. 이는 프롬프트 주입된 클로드가 민감한 시스템 파일을 수정하는 것을 방지하는 데 특히 중요합니다.
2. **네트워크 격리**, 클로드가 승인된 서버에만 연결할 수 있도록 보장합니다. 이는 프롬프트 주입된 클로드가 민감한 정보를 유출하거나 악성 소프트웨어를 다운로드하는 것을 방지합니다.

효과적인 샌드박싱에는 다음이 모두 필요하다는 점을 주목할 가치가 있습니다 *둘 다* 파일시스템과 네트워크 격리입니다. 네트워크 격리가 없다면, 손상된 에이전트가 SSH 키와 같은 민감한 파일을 유출할 수 있고, 파일시스템 격리가 없다면, 손상된 에이전트가 샌드박스를 쉽게 탈출하여 네트워크 접근 권한을 얻을 수 있습니다. 이 두 기술을 모두 사용함으로써 Claude Code 사용자에게 더 안전하고 빠른 에이전트 경험을 제공할 수 있습니다.

Claude Code의 두 가지 새로운 샌드박싱 기능

샌드박스 bash 도구: 권한 프롬프트 없는 안전한 bash 실행

우리는 새로운 샌드박스 런타임을 소개합니다**베타 버전으로 연구 프리뷰로 제공되는**이 런타임을 통해 컨테이너를 생성하고 관리하는 오버헤드 없이 에이전트가 액세스할 수 있는 디렉토리와 네트워크 호스트를 정확히 정의할 수 있습니다. 이는 임의의 프로세스, 에이전트 및 MCP 서버를 샌드박스화하는 데 사용할 수 있습니다. 또한 **오픈 소스 연구 프리뷰로도 제공됩니다**.

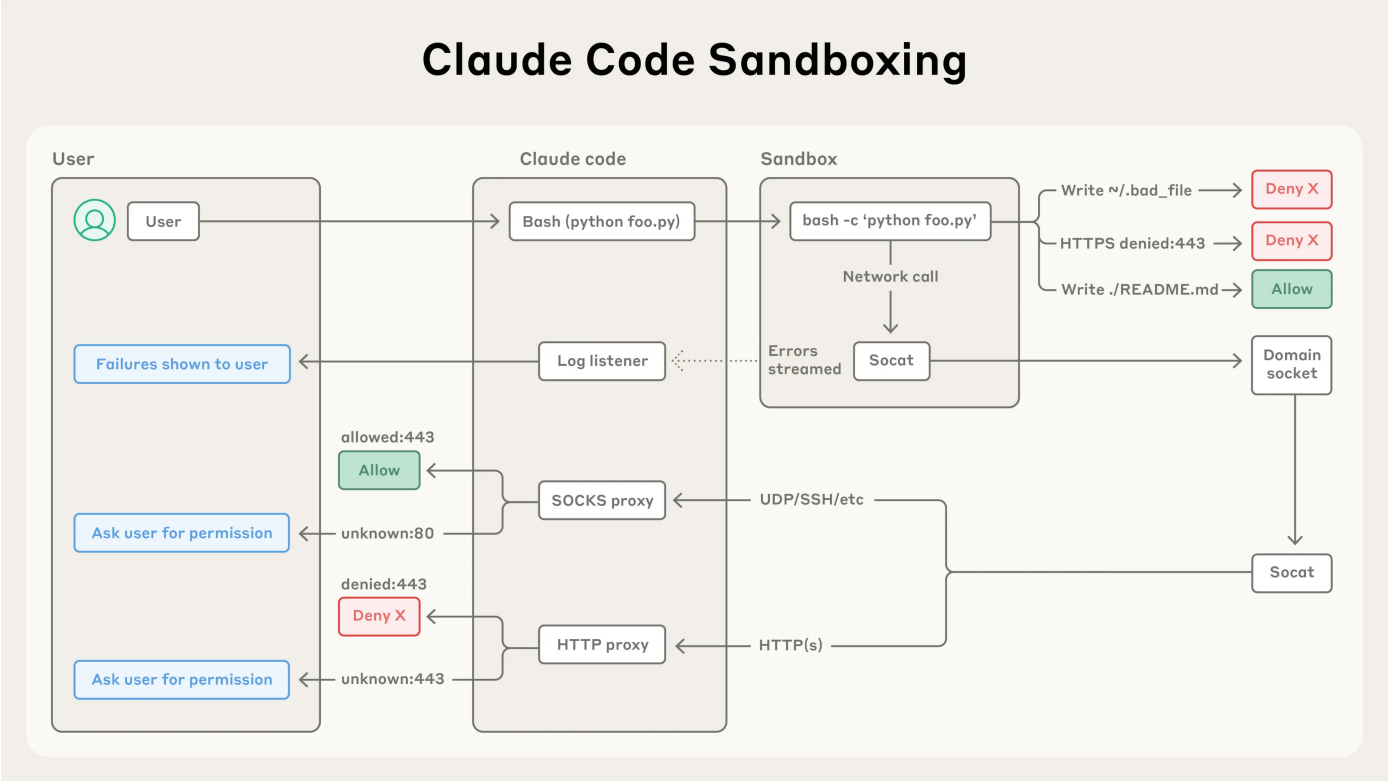
Claude Code에서는 이 런타임을 사용하여 bash 도구를 샌드박스화하며, 이를 통해 Claude가 사용자가 설정한 정의된 제한

내에서 명령을 실행할 수 있습니다. 안전한 샌드박스 내에서 Claude는 권한 요청 없이 더 자율적이고 안전하게 명령을 실행할 수 있습니다. Claude가 샌드박스 *외부의* 무언가에 접근하려고 시도하면 즉시 알림을 받게 되며, 이를 허용할지 여부를 선택할 수 있습니다.

우리는 이를 다음과 같은 OS 수준 프리미티브 위에 구축했습니다:Linux bubblewrap그리고 MacOS seatbelt을 사용하여 OS 수준에서 이러한 제한사항을 강제합니다. 이는 Claude Code의 직접적인 상호작용뿐만 아니라 명령어에 의해 생성되는 모든 스크립트, 프로그램 또는 하위 프로세스도 포함합니다. 위에서 설명한 바와 같이, 이 샌드박스는 다음 두 가지를 모두 강제합니다:

1. **파일시스템 격리,** 현재 작업 디렉토리에 대한 읽기 및 쓰기 액세스는 허용하되, 그 외부의 파일 수정은 차단합니다.
2. **네트워크 격리,** 샌드박스 외부에서 실행되는 프록시 서버에 연결된 유닉스 도메인 소켓을 통해서만 인터넷 액세스를 허용합니다. 이 프록시 서버는 프로세스가 연결할 수 있는 도메인에 대한 제한을 적용하고, 새로 요청된 도메인에 대한 사용자 확인을 처리합니다. 그리고 보안을 더욱 강화하고 싶다면, 나가는 트래픽에 임의의 규칙을 적용하도록 이 프록시를 사용자 정의하는 것도 지원합니다.

두 구성 요소 모두 설정 가능합니다: 특정 파일 경로나 도메인을 허용하거나 차단하도록 쉽게 선택할 수 있습니다.



클릭하여 재시도

클릭하여 재시도

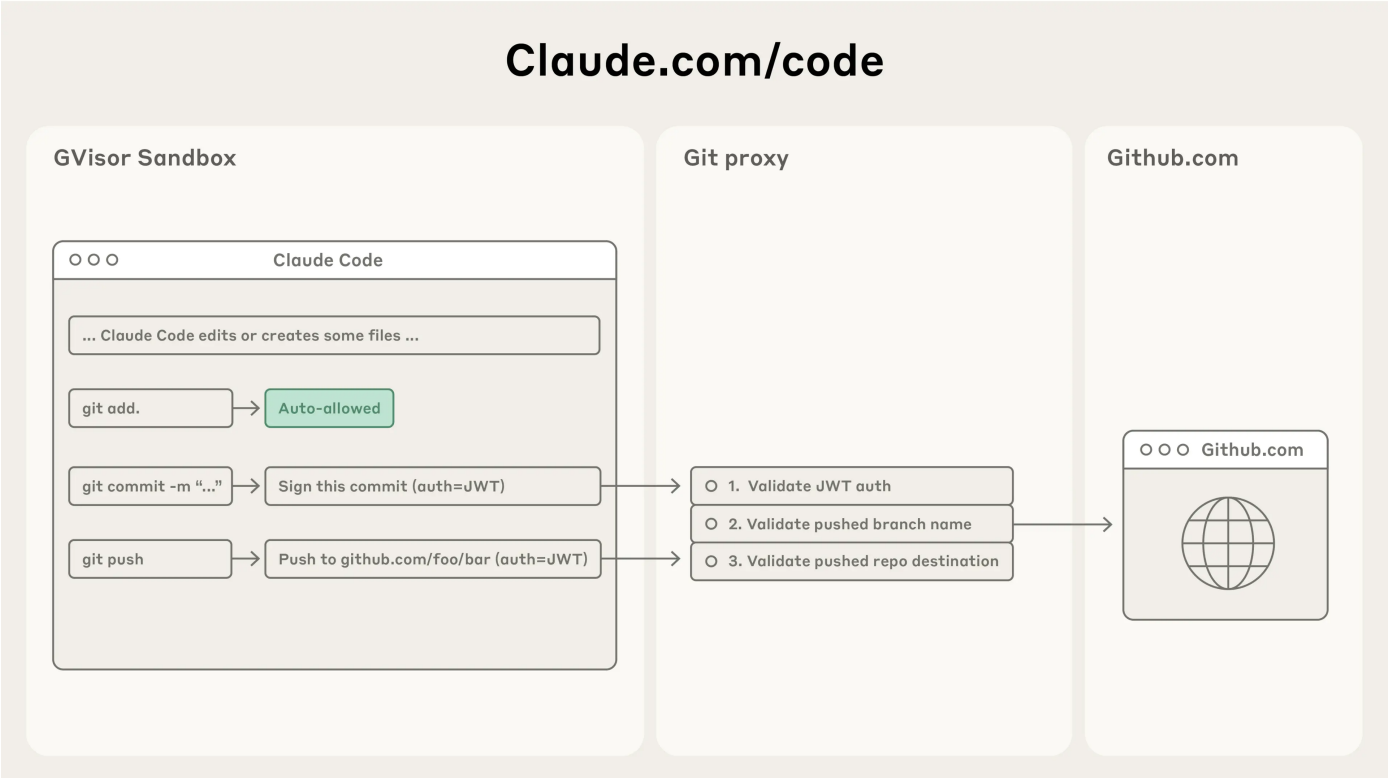
이 기능을 시작하려면 Claude Code에서 /sandbox를 실행하고 확인해보세요더 자세한 기술적 세부사항우리의 보안 모델에 대해.

다른 팀들이 더 안전한 에이전트를 구축할 수 있도록 돕기 위해, 우리는오픈 소스화했습니다이 기능을. 우리는 다른 사람들이 자신들의 에이전트의 보안 태세를 강화하기 위해 자신들의 에이전트에 이 기술을 채택하는 것을 고려해야 한다고 믿습니다.

웹에서의 Claude Code: 클라우드에서 Claude Code를 안전하게 실행

오늘, 우리는 또한 출시합니다웹에서의 Claude Code사용자가 클라우드의 격리된 샌드박스에서 Claude Code를 실행할 수 있도록 합니다. 웹의 Claude Code는 각 Claude Code 세션을 격리된 샌드박스에서 실행하여 안전하고 보안이 유지되는 방식으로 서버에 완전히 액세스할 수 있습니다. 우리는 민감한 자격 증명(git 자격 증명이나 서명 키 등)이 Claude Code와 함께 샌드박스 내부에 절대 들어가지 않도록 이 샌드박스를 설계했습니다. 이렇게 하면 샌드박스에서 실행되는 코드가 손상되더라도 사용자는 추가적인 피해로부터 안전하게 보호됩니다.

웹상의 Claude Code는 모든 git 상호작용을 투명하게 처리하는 맞춤형 프록시 서비스를 사용합니다. 샌드박스 내부에서 git 클라이언트는 맞춤 제작된 범위 지정 자격 증명으로 이 서비스에 인증합니다. 프록시는 이 자격 증명과 git 상호작용의 내용을 검증한 후(예: 구성된 브랜치에만 푸시하는지 확인), GitHub에 요청을 보내기 전에 적절한 인증 토큰을 첨부합니다.



Claude Code의 Git 통합은 인증 토큰, 브랜치 이름, 저장소 대상을 검증하는 보안 프록시를 통해 명령을 라우팅하여 무단 푸시를 방지하면서 안전한 버전 관리 워크플로우를 가능하게 합니다.

시작하기

새로운 샌드박스 bash 도구와 웹의 Claude Code는 엔지니어링 작업에 Claude를 사용하는 개발자들에게 보안과 생산성 모두에서 상당한 개선을 제공합니다.

이러한 도구를 시작하려면:

1. Claude에서 `/sandbox` 를 실행하고 확인해보세요저희 문서에서 이 샌드박스를 구성하는 방법을 확인하세요.
2. 웹에서 Claude Code를 사용해보려면 claude.com/code로 이동하세요.

또는, 자체 에이전트를 구축하는 경우 다음을 확인하세요오픈소스 샌드박싱 코드, 그리고 이를 여러분의 작업에 통합하는 것을 고려해 보세요. 여러분이 무엇을 만들어낼지 기대하고 있습니다.

Claude Code에 대해 더 자세히 알아보려면 저희출시 블로그 포스트.

감사의 말

David Dworken과 Oliver Weller-Davies가 작성한 기사이며, Catherine Wu, Molly Vorwerck, Alex Isken, Kier Bradwell, Kevin Garcia가 기여했습니다

Beyond permission prompts: making Claude Code more secure and autonomous

Published Oct 20, 2025

Claude Code's new sandboxing features, a bash tool and Claude Code on the web, reduce permission prompts and increase user safety by enabling two boundaries: filesystem and network isolation.

In Claude Code, Claude writes, tests, and debugs code alongside you, navigating your codebase, editing multiple files, and running commands to verify its work. Giving Claude this much access to your codebase and files can introduce risks, especially in the case of prompt injection.

To help address this, we've introduced two new features in Claude Code built on top of sandboxing, both of which are designed to provide a more secure place for developers to work, while also allowing Claude to run more autonomously and with fewer permission prompts. In our internal usage, we've found that sandboxing safely reduces permission prompts by 84%. By defining set boundaries within which Claude can work freely, they increase security and agency.

Keeping users secure on Claude Code

Claude Code runs on a permission-based model: by default, it's read-only, which means it asks for permission before making modifications or running any commands. There are some exceptions to this: we auto-allow safe commands like echo or cat, but most operations still need explicit approval.

Constantly clicking "approve" slows down development cycles and can lead to 'approval fatigue', where users might not pay close attention to what they're approving, and in turn making development less safe.

To address this, we launched sandboxing for Claude Code.

Sandboxing: a safer and more autonomous approach

Sandboxing creates pre-defined boundaries within which Claude can work more freely, instead of asking for permission for each action. With sandboxing enabled, you get drastically fewer permission prompts and increased safety.

Our approach to sandboxing is built on top of operating system-level features to enable two boundaries:

1. **Filesystem isolation**, which ensures that Claude can only access or modify specific directories. This is particularly important in preventing a prompt-injected Claude from modifying sensitive system files.
2. **Network isolation**, which ensures that Claude can only connect to approved servers. This prevents a prompt-injected Claude from leaking sensitive information or downloading malware.

It is worth noting that effective sandboxing requires *both* filesystem and network isolation. Without network isolation, a compromised agent could exfiltrate sensitive files like SSH keys; without filesystem isolation, a compromised agent could easily escape the sandbox and gain network access. It's by using both techniques that we can provide a safer and faster agentic experience for Claude Code users.

Two new sandboxing features in Claude Code

Sandboxed bash tool: safe bash execution without permission prompts

We're introducing a new sandbox runtime, available in beta as a research preview, that lets you define exactly which directories and network hosts your agent can access, without the overhead of spinning up and managing a container. This can be used to sandbox arbitrary processes, agents and MCP servers. It is also available as an open source research preview.

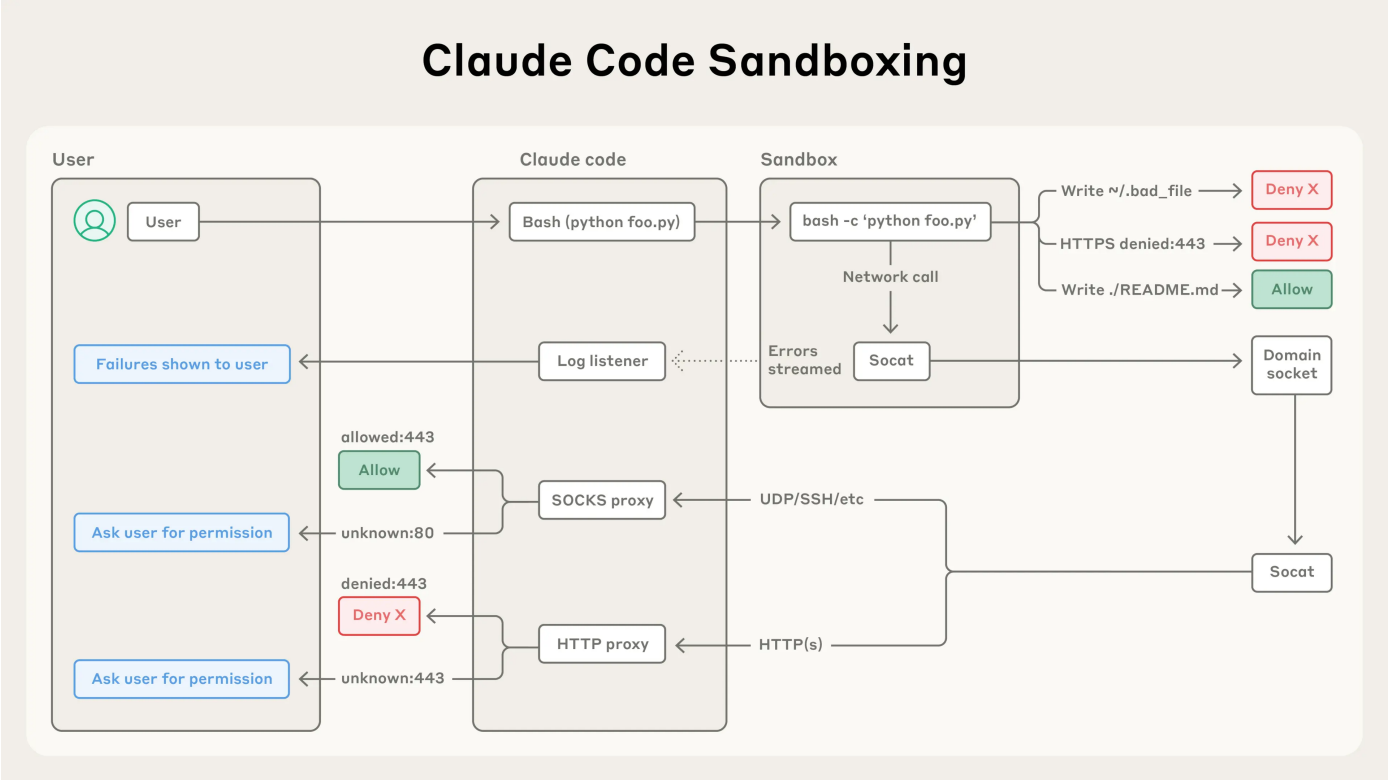
In Claude Code, we use this runtime to sandbox the bash tool, which allows Claude to run commands within the defined limits you set. Inside the safe sandbox, Claude can run more autonomously and safely execute commands without permission prompts. If Claude tries to access something *outside* of the sandbox, you'll be notified immediately, and can choose whether or not to allow it.

We've built this on top of OS level primitives such as Linux bubblewrap and MacOS seatbelt to enforce these restrictions at the OS level. They cover not just Claude Code's direct interactions, but also any scripts, programs, or subprocesses that are spawned by the command.As described above, this sandbox enforces both:

1. **Filesystem isolation,** by allowing read and write access to the current working directory, but blocking the modification of any files outside of it.
2. **Network isolation,** by only allowing internet access through a unix domain socket connected to a proxy server running outside the sandbox. This proxy server enforces restrictions on the domains

that a process can connect to, and handles user confirmation for newly requested domains. And if you'd like further-increased security, we also support customizing this proxy to enforce arbitrary rules on outgoing traffic.

Both components are configurable: you can easily choose to allow or disallow specific file paths or domains.



Claude Code's sandboxing architecture isolates code execution with filesystem and network controls, automatically allowing safe operations, blocking malicious ones, and asking permission only when needed.

Sandboxing ensures that even a successful prompt injection is fully isolated, and cannot impact overall user security. This way, a compromised Claude Code can't steal your SSH keys, or phone home to an attacker's server.

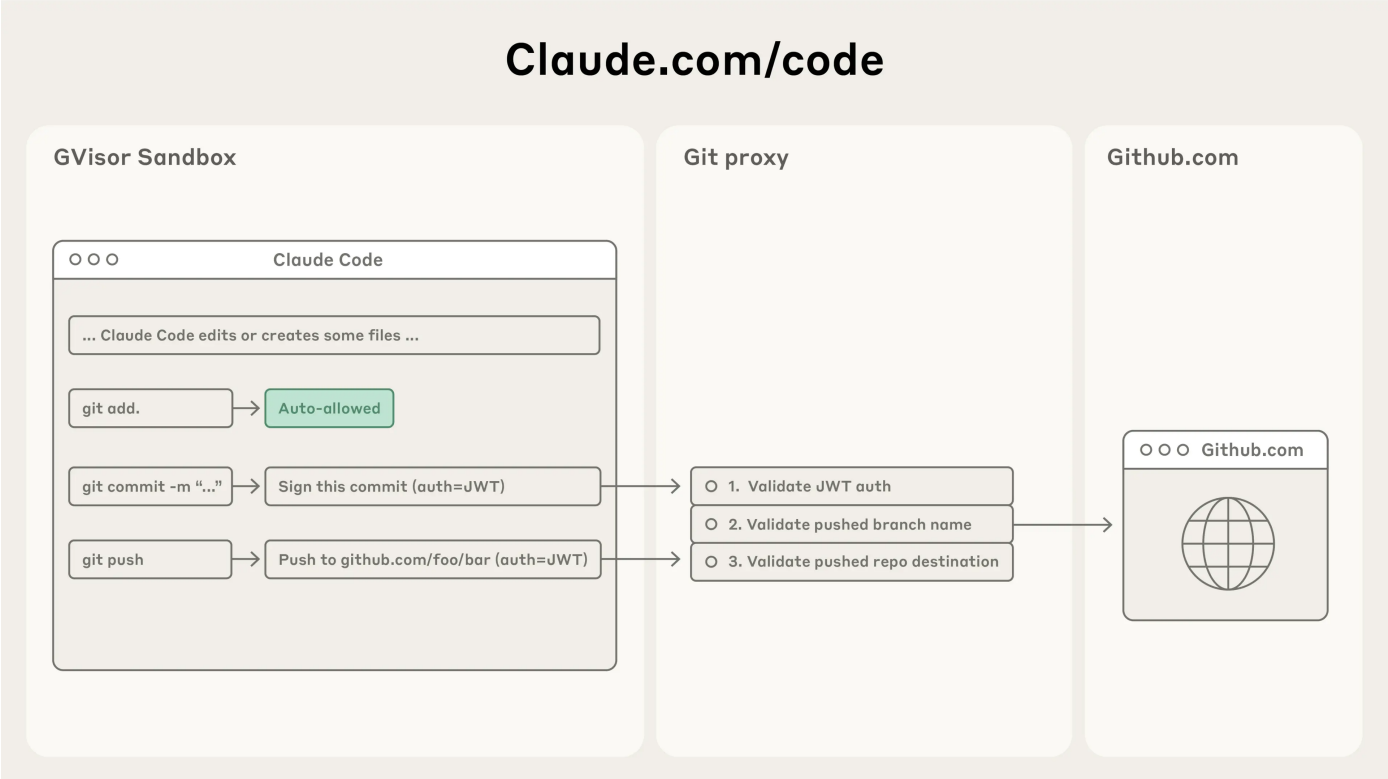
To get started with this feature, run /sandbox in Claude Code and check out more technical details about our security model.

To make it easier for other teams to build safer agents, we have open sourced this feature. We believe that others should consider adopting this technology for their own agents in order to enhance the security posture of their agents.

Claude Code on the web: running Claude Code securely in the cloud

Today, we're also releasing Claude Code on the web enabling users to run Claude Code in an isolated sandbox in the cloud. Claude Code on the web executes each Claude Code session in an isolated sandbox where it has full access to its server in a safe and secure way. We've designed this sandbox to ensure that sensitive credentials (such as git credentials or signing keys) are never inside the sandbox with Claude Code. This way, even if the code running in the sandbox is compromised, the user is kept safe from further harm.

Claude Code on the web uses a custom proxy service that transparently handles all git interactions. Inside the sandbox, the git client authenticates to this service with a custom-built scoped credential. The proxy verifies this credential and the contents of the git interaction (e.g. ensuring it is only pushing to the configured branch), then attaches the right authentication token before sending the request to GitHub.



Claude Code's Git integration routes commands through a secure proxy that validates authentication tokens, branch names, and repository destinations—allowing safe version control workflows while preventing unauthorized pushes.

Getting started

Our new sandboxed bash tool and Claude Code on the web offer substantial improvements in both security and productivity for developers using Claude for their engineering work.

To get started with these tools:

1. Run `/sandbox` in Claude and check out our docs on how to configure this sandbox.
2. Go to claude.com/code to try out Claude Code on the web.

Or, if you're building your own agents, check out our open-sourced sandboxing code, and consider integrating it into your work. We look forward to seeing what you build.

To learn more about Claude Code on the web, check out our launch blog post.

Acknowledgements

Article written by David Dworken and Oliver Weller-Davies, with contributions from Catherine Wu, Molly Vorwerck, Alex Isken, Kier Bradwell, and Kevin Garcia