On Permission Fatigue

I use Kiro at work and Claude Code on weekends. I also occasionally tinker with Cursor, Codex, Pi, and others. The interfaces differ but these tools are all wrappers around models that are getting better at an astounding speed. It is clear that we are not going back to writing code the way we did in the recent past. It is also clear that babysitting these tools with `Do you want <Tool> to do <X>?` is already getting in the way. As Claude Code's current documentation puts it:

> *By default, Claude Code requests permission for actions that might modify your system: file writes, Bash commands, MCP tools, etc. This is safe but tedious. After the tenth approval you're not really reviewing anymore, you're just clicking through.*
>
> *– https://code.claude.com/docs/en/best-practices#configure-permissions*

We cannot address *safe but tedious* with `--dangerously-skip-permissions`. Agents are tenacious and even mild prodding can push them to seek unintended goals in creative ways. If they have access to sensitive data, can run arbitrary code, and can access remote endpoints, we have a problem. This page shares how I use Apple Containers and Claude Code sandbox to contain this lethal trifecta in my weekend projects.

My Claude Sandbox

I built `claude-sandbox` to automate this setup. It requires macOS Tahoe on Apple Silicon with the `container` CLI installed. You can install it with Cargo:

```
cargo install --git https://github.com/aldrin/claude-sandbox.git
```

The README has the details, but the workflow is straightforward.

I go into a project I need AI assistance with and run `claude-sandbox init` to scaffold a `.claude-sandbox/` folder with the container configuration. Then I run `claude-sandbox build` to build the container image and `claude-sandbox run` to start a Claude Code session and work as usual. The `build` step installs the latest Claude Code into the image and configures its sandbox settings. The only use for Claude on the host-side is running `claude auth login` to populate the OAuth token. At runtime, `claude-sandbox run` reads it from the host keychain and passes it to the Claude Code instance inside the container.

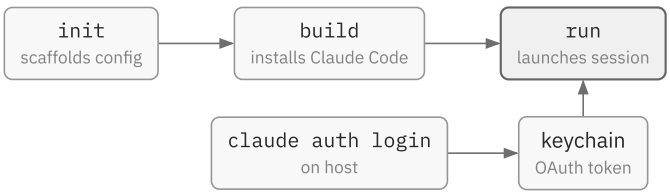

Figure 1: The claude-sandbox workflow.

To limit the data the agent can access, `claude-sandbox run` launches Claude Code inside an Apple Container. Each container runs in its own lightweight VM using Virtualization.framework on Apple Silicon, so isolation happens at the VM boundary rather than through kernel namespaces. The container

mounts only the project directory I started it in. The agent has no path to any other file on my machine. I pick what to share by picking where to run the tool.

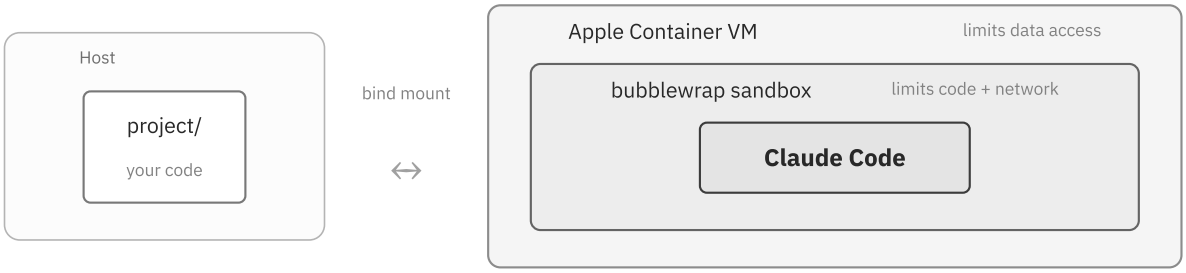

Figure 2: Isolation layers at runtime.

To limit the impact of arbitrary code, `claude-sandbox` configures Claude Code's Bash sandbox inside the container. With `autoAllowBashIfSandboxed` on, Bash commands run without approval prompts. The `acceptEdits` permission mode lets the agent write files freely within the mounted directory. Under the hood, bubblewrap restricts Bash commands to the mounted project directory and blocks network access to domains not on the allowlist. Every Bash command and its child processes inherit these restrictions, so if the agent goes off track, the damage stays contained.

To limit connections to arbitrary endpoints, all network traffic routes through local proxies. Anthropic's engineering post describes the design: HTTP, HTTPS, and SOCKS traffic go through proxy servers on localhost. DNS resolution and direct TCP connections to external hosts are blocked. When the agent tries to reach a domain not on the allowlist, the request goes to Claude Code's permission system and I see a prompt.

While the agent does its thing in the sandbox, the project directory is a bind mount, so changes appear on my host filesystem in real time. I keep my own editor and git workflow. When the session ends, the container goes away, but the work remains in my git workspace to review, refine, and publish. This page was written in one such session.