Claude Code

- Quick start
- Authentication
- Configuration
  - Remote control
  - Default startup command
- Agents view
- Base image
- Use a local model

The following instructions apply to local sandboxes. For cloud authentication and usage, see Authenticate cloud agents and Use cloud sandboxes.

Official documentation: Claude Code

Quick start

Launch Claude Code in a sandbox by pointing it at a project directory:

```
$ sbx run claude ~/my-project
```

To start Claude with a specific prompt in the current directory:

```
$ sbx run --name my-sandbox claude -- "Add error handling to the login function"
```

Everything after `--` is passed directly to Claude Code. You can also pipe in a prompt from a file with `-- "$(cat prompt.txt)"`.

To create a mountless sandbox, use `sbx create` without a workspace path, then attach by name.

Authentication

For the default Anthropic models, Claude Code requires either an Anthropic API key or a Claude subscription. For other models, see Use a local model.

API key: Store your key using stored secrets:

```
$ sbx secret set anthropic
```

Claude subscription: If no API key is set, use the `/login` command inside Claude Code to authenticate via OAuth.

Configuration

Sandboxes don't pick up user-level configuration from your host, such as `~/.claude`. Only project-level configuration in the working directory is available inside the sandbox. See Why doesn't the sandbox use my user-level agent configuration? for workarounds.

Remote control

To use Claude Code's `/remote-control` command inside a sandbox, turn on `claude.remoteControl`:

```
$ sbx settings set claude.remoteControl true
```

Default startup command

Without extra args, the sandbox runs:

```
claude --dangerously-skip-permissions
```

Arguments after `--` are added after the default flags when the first one is itself a flag (begins with `-`), so `--dangerously-skip-permissions` is preserved:

```
$ sbx run --name <sandbox-name> -- -c   # runs claude --dangerously-skip-permissions -c
```

When the first argument is a bare word, such as the `agents` subcommand, it replaces the defaults instead.

See the Claude Code CLI reference for available options.

Agents view

Claude Code's agents view starts background sessions that run tasks in parallel. Pair it with clone mode to keep their changes inside the sandbox:

```
$ sbx run --clone claude . -- agents
```

This invocation replaces the default startup command, so it doesn't include `--dangerously-skip-permissions` and you can't switch to bypass-permissions mode inside the sandbox. To work around this, either use Claude Code's auto mode or pass the flag explicitly:

```
$ sbx run --clone claude . -- --dangerously-skip-permissions agents
```

For simultaneous tasks, each background session needs a separate worktree or clone. Branches alone don't isolate edits in one working directory. How Claude Code manages working directories depends on the task, configuration, and project instructions. The `--clone` flag doesn't control this behavior. Ask Claude Code to create a separate worktree for each task inside the sandbox.

To review a branch created by a session, fetch the `sandbox-<sandbox-name>` remote from the host:

```
$ git fetch sandbox-<sandbox-name>
$ git diff main..sandbox-<sandbox-name>/<branch>
```

See Git workflows for clone-mode details.

Base image

The sandbox uses `docker/sandbox-templates:claude-code`. See Base images to build your own image on top of this base.

Use a local model

For local models, hosted providers, and custom inference endpoints, see Use local and hosted models.

□□□ #2
□ □□□ □□□ 2026-10-31 18:57:37 UTC
□ □□□ □□□ 2026-10-31 18:57:38 UTC