

Run it on a home server

This guide puts Secretbase on a computer that stays at home (the server), and lets you publish to it from your laptop, wherever you are.

Quick start

- 1 Install Secretbase on the server with the Linux command from the download page (see the Setup guide).
- 2 On the server, run `secretbase frontdoor setup` and open the sign-in link it prints on your phone.
- 3 On the server, run `secretbase tokens create --name laptop --scope admin` and copy the token.
- 4 On the laptop, run `curl -fsSL https://secretbase.me/download/install.sh | sh -s -- --cli`.
- 5 On the laptop, run `secretbase remote add home <address>` and paste the token when it asks.
- 6 Publish from the laptop: `secretbase publish ./my-site --to home`.

What you get

- One computer at home, always on, that runs your apps. It needs Linux, but no screen.
- A laptop you carry anywhere. From it you publish a folder or a git link with one command.
- An access token connects the two. With `--scope admin`, anyone with the token can publish, change and delete apps on the server, so treat it like a password. You can revoke it any time.

Set up the server

Install Secretbase on the server as the Setup guide says, at its keyboard or over SSH. When it asks "Open Secretbase from any browser, on your phone or another computer?", yes also lets you open the dashboard from your laptop's browser.

A server without a screen can't open the Tailscale sign-in page itself, so `secretbase frontdoor setup` prints a line "Continue in your browser: <link>". Open the link on your phone and sign in with your personal Tailscale account. If a second link appears, open it too and click **Allow**. The server then says "Ready: your apps can go online."

Make a token

On the server, run `secretbase tokens create --name laptop --scope admin` as yourself, without sudo. It shows the token once and the line to run on your laptop:

```
Access token "laptop" (shown only now: copy it somewhere safe):
```

```
sbt_...
```

```
Scope admin, expires <date>.
```

```
On the other computer, run this and paste the token when it asks:
```

```
secretbase remote add home <address>
```

`--scope admin` lets the laptop do everything in this guide. A token lasts 90 days, or for good with `--expires never`. Without `--scope`, a token gets `deploy`, made for AI tools (see the guide "Let an AI run your server"). Tokens made before version 0.17.0 work as admin, with no expiry.

Keep the token in a password manager. `secretbase tokens` lists your tokens (never the tokens themselves); `secretbase tokens revoke laptop` stops one at once. Make one token per laptop or tool, up to 20. The **Access tokens** card on the dashboard's **AI tools** page does the same.

Set up your laptop

The laptop only needs the secretbase command. On Linux, run `curl -fsSL https://secretbase.me/download/install.sh | sh -s -- --cli`: no password, and only secretbase goes into `~/.local/bin`.

Then run `secretbase remote add home <address>` and paste the token when it asks (never on the command line). It says "Added home". `home` is just a name: add more servers under other names. `secretbase remote list` shows them.

Publish from your laptop

```
secretbase publish ./my-site --to home
```

It sends the folder, publishes it on the server and ends with "Live at" and the app's address. A folder, a page (`.html`) or a `.zip` works, up to 20,000 files and 2 GB; `.git` and `node_modules` are never sent. Publish the same folder again to update the app.

Everything else works the same way, with `--to home`:

- `secretbase status --to home`: every app on the server, with its state and address.
- `secretbase logs my-site --to home`: the app's last log lines.
- `secretbase offline my-site --to home` and `secretbase online my-site --to home`.
- `secretbase delete my-site --to home`: removes the app and its data. Its backups stay.

Publish from a git link

The server can fetch your code from GitHub, GitLab or another git service itself:

```
secretbase publish https://github.com/you/app --to home
```

The app is named after the repository. `--ref <branch or tag>` publishes another branch or tag; `--name <name>` gives it another name. After that the server publishes each new commit by itself (**Update on push**, checked every minute); turn it off on the server with `secretbase updates my-app off`. On the dashboard, **From a git link** does the same.

Only `https://` links work. Submodules aren't fetched, Git LFS files arrive as small pointer files, and a fetch stops after 10 minutes.

A **private repository** needs an access token from your git service (on GitHub: a fine-grained token with read-only access to that repository's contents). Run `secretbase git-token add github.com --to home` and paste it when it asks. `secretbase git-token list --to home` and `secretbase git-token remove github.com --to home` manage them.

Connect your AI tool on the laptop

`secretbase tokens create` also prints a line for Claude Code, `claude mcp add --transport http ...`. Make a token for it without `--scope admin` (or with `--app <app>` for one app only), run its line on the laptop, and Claude Code runs the server from anywhere: it asks you first before it deletes an app or makes a new one public. The guide "Let an AI run your server" shows what it can do.

That connection publishes git links. To let Claude Code publish a folder from the laptop, run once:

```
claude mcp add --scope user secretbase -- secretbase mcp --remote home
```

Then ask it to publish your project: it publishes to the server. Other AI tools can use `secretbase mcp --remote home` the same way.

Claude on the web, desktop or phone connects by signing in instead: see "Claude on the web, desktop or phone" in the guide "Let an AI run your server".

With curl

The server's address answers the same API under `/v1`, with the token in the `Authorization` header (never in the address). Run `read -rs TOKEN`, paste the token, and set `SERVER=<address>`. Then:

```

curl -H "Authorization: Bearer $TOKEN" $SERVER/v1/projects
tar -cf my-site.tar my-site
curl -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/x-tar" \
  --data-binary @my-site.tar $SERVER/v1/uploads
curl -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d '{"upload_id": "<upload_id>", "on_exists": "update"}' $SERVER/v1/projects
curl -X POST -H "Authorization: Bearer $TOKEN" $SERVER/v1/projects/<id>/publish

```

The first lists your apps. The upload's answer has an `upload_id`; adding it answers with the app's `id`; publishing answers with a `job_id`. `GET $SERVER/v1/jobs/<job_id>` shows the job's `state`, and `GET $SERVER/v1/projects/<id>` the app's `state` and `address`. Send `{"git_url": "https://github.com/you/app"}` to `/v1/projects` to add a repository instead. The same API sets keys (`PUT .../keys/<NAME>`), undoes (`POST .../undo`), takes an app offline (`POST .../offline`) and online (`POST .../online`), and deletes it (`DELETE $SERVER/v1/projects/<id>`).

Good to know

- Tailscale gives a computer three public addresses. The server's online address, used by tokens and the online dashboard, takes one, which leaves two for apps. For more, run `secretbase tailscale-key` on the server and paste a reusable key from Tailscale's admin console (Settings → Keys → Generate auth key, Reusable on, Ephemeral off, Tags off): every new app then gets an address of its own. Or put apps on your own domain (see the Setup guide).
- Deleting an app frees its address; taking it offline doesn't.
- The online dashboard (`secretbase dashboard on` on the server) makes and revokes tokens and sets keys, but it can't make a Full control token, add git tokens, sign in to Tailscale or Cloudflare, approve moving a domain, change backup places, show the recovery key or bring an app back. Do those on the server.
- An admin token can see apps, publish, undo, read logs, take apps offline and online, delete them, set keys, manage git tokens, and back up and restore. Anything else answers 403 and is done on the server. A `read` or `deploy` token can do less (see the guide "Let an AI run your server").

If something goes wrong

- "Make access tokens as yourself, not with sudo": run `secretbase tokens create` without sudo.
- "This computer's online address is going up": run `secretbase tokens` a minute later; it ends with "They work at" and the address.
- It says `~/local/bin` "isn't on your PATH yet": add the line it shows to `~/profile`, then open a new terminal.
- A publish asks to accept the notice: run `secretbase notice` on the server once.
- "Secretbase couldn't read this repository": check the link, or add a git token (above).

- An app is published but has no public address (HB-FRONTDOOR-017) : all addresses are taken. Delete an app, add a Tailscale key, or use your own domain.
- **401** "This access token is wrong or was revoked": make a new one. **429** "Too many wrong tries from your address": wait 15 minutes. **403** "This can only be done on the computer itself.": do it on the server.
- **403** "This token's scope is < scope > ; this needs < needed > ": make an admin token. **401** "This access token expired on < date > ": make a new one.

Still stuck? Send us the step number, the exact message, and what `secretbase status` shows on the server.