1Password CLI Secrets from the shell

START HERE

`op signin`
Sign in

`op item get <item>`
Show an item

`op whoami`
Who am I?

`op read op://<vault>/<item>/<field>`
Print one secret

`op vault list`
See vaults

`op run --env-file=<file> -- <cmd>`
Run with secrets

`op item list`
See items

`op inject -i <tpl> -o <out>`
Fill a template

Reference op://vault/item/field → **`op run` / `inject`** → **1Password CLI** signed-in account → **resolves** → **Secret value** env var or file

CONCEPTS

Keep secrets out of files: store op:// references instead, and let op run or op inject resolve them at the last moment.

With the desktop app integration on (Settings > Developer), op unlocks via Touch ID or system auth and needs no manual signin.

Use --account <shorthand> or OP_ACCOUNT when you have several accounts; use --format json to script output.

Command arguments can be seen by other processes. Pass sensitive values via --template or stdin, not as assignment arguments.

SIGN IN

`op signin`
Sign in (prompts only if needed)

`eval $(op signin --account <shorthand>)`
Sign in, set session in this shell
Needed only without the desktop app integration.

`op whoami`
Show the signed-in account

`op signout`
Sign out

`op account list`
List configured accounts

`op account add --address <host>`
Add an account

`op signin --raw`
Print only the session token

`op update`
Check for a new CLI version

`op completion <shell>`
Generate shell completions

VAULTS

`op vault list`
List readable vaults

`op vault get <vault>`
Show a vault

`op vault create <name>`
Create a vault

`op vault edit <vault> --name <new>`
Rename or edit a vault

VAULTS (cont.)

`op vault delete <vault>`
Delete a vault

`op vault user list <vault>`
Who can access this vault

`op vault user grant --vault <v> --user <u> --permissions <p>`
Grant a user access

ITEMS

`op item list`
List items

`op item list --vault <vault>`
List items in one vault

`op item list --categories Login`
Filter by category

`op item list --tags <tag>`
Filter by tag

`op item get <item>`
Show item details

`op item get <item> --fields label=password`
Print one field

`op item get <item> --otp`
Current one-time password

`op item get <item> --format json`
Full item as JSON

`op item get <item> --share-link`
Get a share link

`op item create --category login --title <title>`
Create a login item

`op item create ... --generate-password=20,letters,digits`
Create with a generated password

`op item create ... <field>=<value>`
Set a field on create
Section fields use <section>.<field>=<value>.

`op item create --template <file.json>`
Create from a JSON template

`op item edit <item> <field>=<value>`
Change a field

`op item edit <item> --generate-password`
Regenerate the password

ITEMS (cont.)

`op item edit <item> '<section>.<field>[delete]'`
Delete a custom field

`op item edit <item> --tags <t1>,<t2>`
Set tags

`op item delete <item>`
Delete an item
Stays in Recently Deleted for 30 days.

`op item delete <item> --archive`
Archive instead of delete

`op item move <item> --current-vault <a> --destination-vault <b>`
Move to another vault
Creates a copy with a new item ID.

`op item share <item> --emails <addr> --view-once`
Share by link

`op item template list`
List item categories

`op item template get <category>`
Get a category's JSON template

SECRET REFERENCES

`op read op://<vault>/<item>/<field>`
Print a secret

`op read op://<vault>/<item>/<section>/<field>`
Read a field in a section

`op read "op://<vault>/<item>/one-time password?attribute=otp"`
Read the current OTP

`op read "op://<vault>/<item>/private key?ssh-format=openssh"`
SSH key in OpenSSH format

`op read -o <file> op://<vault>/<item>/<field>`
Write a secret to a file
File mode defaults to 0600.

`op read -n op://<vault>/<item>/<field>`
No trailing newline

References use names or IDs. Quote them if a name contains spaces.

RUN AND INJECT

`op run -- <cmd>`
Run with env var references resolved

RUN AND INJECT (cont.)

`op run --env-file=<file> -- <cmd>`
Resolve references from a .env file

`op run --no-masking -- <cmd>`
Show secrets in command output
Output is masked by default.

`op inject -i <tpl> -o <out>`
Fill a template into a file

`op inject -i <tpl> -f -o <out>`
Overwrite without asking

`cat <tpl> | op inject`
Fill a template from stdin

Templates use {{ op://vault/item/field }}. Env vars inside references, like op://$APP_ENV/db/password, switch vaults per environment.

Delete resolved output files when done; they hold real secrets.

DOCUMENTS

`op document list`
List documents

`op document create <file> --title <title>`
Upload a file

`op document get <doc> -o <file>`
Download a document
Will not overwrite an existing file.

RECIPES

Run an app with its secrets

```
# .env
DB_URL=op://<vault>/<item>/url

op run --env-file=.env -- ./start.sh
```

Render a config file

```
op inject -i config.yml.tpl -o config.yml
# remove config.yml when done
```

Log in to Docker from 1Password

```
op read op://<vault>/docker/password \
  | docker login -u <user> --password-stdin
```

Copy an item to another vault

```
op item get <item> --vault <a> --format json \
  | op item create --vault <b> -
```