Kit authoring patterns

- Keep runtime access with the tool
- Leave build tools out of the mixin
- Run setup after combining kits
- Seed storage after it is mounted
  - Configure the volume
- Publish fixed components with configurable options

{{< summary-bar feature_name="Docker Sandboxes sbx" >}}

When you package a tool as a kit, aim to make it work wherever you add it. These patterns show how to include the access it needs, prepare it for each sandbox, and give your team useful options without asking them to assemble the environment themselves.

Keep runtime access with the tool

Keep a tool's network rules and credential request in the mixin that installs it. For example, a Claude Code mixin can package the executable, allow access to the Anthropic API, and request the API key. Those requirements follow the tool when you add it to another workload.

The user still needs to supply the credential and approve access. See Build a tool mixin for a complete example.

Leave build tools out of the mixin

Build a tool in one Dockerfile stage, then copy the executable into an empty stage. Each sandbox gets the tool without also getting its compiler and source code:

```
# syntax=docker/sandbox-kit:3
schemaVersion: "3"
kind: mixin
```

```
provides: ["gojq@0.12.17"]

build: |
  FROM golang:1.25 AS build
  RUN CGO_ENABLED=0 go install github.com/itchyny/gojq/cmd/gojq@v0.12.17

  FROM scratch
  COPY --from=build /go/bin/gojq /usr/local/bin/gojq
```

`CGO_ENABLED=0` builds gojq without a dependency on the workload's C libraries. `FROM scratch` starts the final stage empty, so it contains only the binary you copy. The `provides` entry tells other kits which gojq version this mixin installs.

Run setup after combining kits

Some setup needs tools or files from the workload. Package what you can in the mixin, then use an install hook to finish setup once all the kits' files are in place. For example, bring an internal certificate authority (CA) certificate in the mixin and register it in the workload's trust store:

```
# syntax=docker/sandbox-kit:3
schemaVersion: "3"
kind: mixin

build: |
  FROM scratch
  COPY internal-ca.crt /usr/local/share/ca-certificates/team-internal-ca.crt

capabilities:
  - type: com.docker.sandbox/lifecycle@1
    config:
      install:
        - command: update-ca-certificates
          user: "0"
```

Save your PEM-encoded CA as `internal-ca.crt` beside the descriptor and use the mixin with a workload that supplies `update-ca-certificates`, such as Docker's shell kit. The hook updates the workload's trust store as root, after all kit files are present and before the agent launches.

```
$ sbx run docker.io/docker/sbx-kit-shell:1.0.0 --kit ./internal-ca
```

Seed storage after it is mounted

Mounting storage at a path hides any files the image already has there. To give a tool some initial data, keep that data elsewhere in the image and copy it to the mounted directory in an install hook. Check whether the destination file exists first so you preserve any changes the user has made:

```
# syntax=docker/sandbox-kit:3
schemaVersion: "3"
kind: mixin

build: |
  FROM scratch
  COPY defaults.json /usr/share/company-cli/defaults.json

capabilities:
  - type: com.docker.sandbox/volume@1
    config:
      path: /home/agent/.company-cli
      size: 1g
  - type: com.docker.sandbox/lifecycle@1
    config:
      install:
        - command: |
            set -eu
            if [ ! -e /home/agent/.company-cli/config.json ]; then
              chown 1000:1000 /home/agent/.company-cli
              install -o 1000 -g 1000 -m 0644 /usr/share/company-cli/defaults.json
/home/agent/.company-cli/config.json
            fi
          user: "0"
```

Save your tool's initial configuration as `defaults.json` beside the descriptor. Use a workload with `chown` and `install`, such as Docker's shell kit. The hook makes the directory and configuration file writable by the agent. If the volume already has a configuration file, the hook leaves its contents intact.

Configure the volume

In local sandboxes, persistent volumes (`volume@1` with `tmpfs: false`) keep data across restarts, kit updates, and sandbox removal. Docker Sandboxes reattaches a volume when the sandbox name, kit source, and volume path match. Changing a volume path gives it separate storage, so keep paths stable across kit releases when you want users to retain their data.

For published kits, changing a tag or digest retains the same kit source. For Git sources, the repository and kit subdirectory must stay the same; for local sources, the source directory must stay the same. Tmpfs contents are discarded when the sandbox stops.

This example allocates `1g` of space. If you omit `size`, `sbx` allocates `512m`.

Use an install hook when your tool needs specific ownership or permissions, as this example does. Make initialization preserve existing files and permissions when the volume is reused. See the upstream volume definition for storage options.

To copy initial files into a mounted workspace instead, use `WORKSPACE_DIR` as the destination and declare it in the hook's environment.

Publish fixed components with configurable options

Choose compatible agent and tool versions for your team, then publish them as a set. Expose arguments for settings users can change without replacing those components, such as a model or linter mode. Keep version choices fixed in the published images.

For the argument syntax and constraints, see Configure component arguments.

□□□ #2
□ □□□ □□□ 2026-10-31 19:00:09 UTC
□ □□□ □□□ 2026-10-31 19:00:10 UTC