Docker Sandboxes release notes

- 0.50.0
  - Highlights
  - What's new
    - Security
    - Kits
    - Sandbox lifecycle and workspaces
    - Authentication and agents
    - Networking and policy
    - Cloud sandboxes
    - CLI and diagnostics
    - Experimental Runtime backend
- 0.49.2
  - Highlights
  - Security updates
  - Recommended actions
- 0.49.1
  - Highlights
  - What's new
    - Kits
    - Sandbox lifecycle and workspaces
    - Networking and cloud sandboxes
    - Authentication and MCP
    - CLI, settings, and diagnostics
    - Templates and installation
- 0.48.0
  - Highlights

- Breaking changes
  - CLI flags, output, and automation
    - Update command flags
    - Update structured-output parsers
    - Update non-interactive workflows
    - Terminal output and streams
  - What's new
    - Security
    - SSH and desktop integration
    - Kits
    - Sandbox lifecycle and workspaces
    - Cloud sandboxes
    - Authentication, agents, and MCP
    - Networking and policy
    - CLI and diagnostics
- Earlier releases

This page lists changes in recent stable releases of Docker Sandboxes. For the full release history, including pre-releases and downloads, see the Docker Sandboxes releases on GitHub.

0.50.0

{{< release-date date="2026-10-26" >}}

GitHub release

Highlights

Docker Sandboxes v0.50.0 adds configurable kit host mounts, preserves persistent kit volumes across updates and sandbox recreation, and adds cloud sandbox inspection. It also strengthens host-execution, credential, and workspace-copy protections.

What's new

Security

- `sbx cp` rejects symbolic links that escape a sandbox workspace when the workspace is reached through an alternate host path, including a symlink or junction, macOS `/private/tmp`, or a Linux bind mount of a workspace subdirectory.
- Host Git commands no longer run programs configured by a sandbox through the workspace repository's `.git` directory.
- Browser, clipboard, and self-update helpers run only from trusted system locations, preventing sandbox-writable directories on `PATH` from supplying those programs.
- Local source kits are reused and composed only when their digests match recorded local builds. Each local source kit rebuilds once after upgrading.
- Git kit references reject ambiguous paths, non-ASCII hostnames, and SSH transport syntax. Use IDNA ASCII labels for internationalized hostnames. HTTPS source restrictions are checked against the parsed URL host.
- MCP OAuth requests block cross-origin credential redirects. Rejected silent refreshes preserve saved credentials without opening a browser.
- Browser-open logs omit URL query strings and fragments, keeping OAuth parameters and tokens in those components out of daemon logs and diagnostic bundles.
- Windows clients verify that the daemon's named pipes belong to the same Windows user. Before using the updated CLI, older elevated daemons may need to be stopped with the previous CLI or Task Manager.
- Environment MCP commands require approval to execute on the host, remain scoped to their sandbox, and are disabled by `--skip-host-commands`. Approval notices explain that executable paths are pinned, but file contents are not. Keep executables and scripts outside writable sandbox mounts.
- Gordon clears **Approve all** and **Always allow** grants when a conversation resumes, requiring approval before running tools again.

Kits

- When creating a sandbox, use `--mount HOST:SANDBOX_PATH` with `sbx create` or `sbx run` to select a host directory for a kit-declared host mount. The sandbox target must exactly match the `path` of a `host-mount@1` capability declared by a kit in the resolved kit set. Environment files support the same bindings through `mounts`, mapping sandbox target paths to host directories. These options cannot add undeclared mount targets.
- Persistent kit volumes retain data across kit updates and sandbox removal and recreation with the same name. This applies to sandboxes created with this version; existing sandbox records retain their previous storage behavior.
- Creating a sandbox from a v3 kit set without a workload kit reports a clear explanation instead of "the kit set is not coherent."

Sandbox lifecycle and workspaces

- Host Git identity is copied into sandboxes for repositories that use the reftable ref format.

- Sandboxes created with `--clone` preserve kit-provided workspace files and report clone initialization failures.
- Canceling clone remote configuration preserves the cancellation error even if stopping Git also fails.

Authentication and agents

- Kiro IDE AWS IAM Identity Center sign-in callbacks reach the sandbox even when the callback port hasn't been published.
- Claude sandboxes using subscription OAuth are provisioned without the API-key helper.

Networking and policy

- HTTP network rules support query-parameter restrictions. Repeat `--query name=value` to require multiple parameters.

Cloud sandboxes

- `sbx --cloud inspect <id|name>` shows details for a single cloud sandbox, including published port URLs. Use `--format json` or `--format yaml` for scripting.
- Running `sbx --cloud run --name <name>` without an agent suggests the correct form: `sbx --cloud run <agent> --name <name>`.

CLI and diagnostics

- `sbx exec` returns exit status 125 when it cannot run the command, such as when the sandbox doesn't exist.
- On macOS and Windows, `sbx diagnose` and daemon logs warn when `SSL_CERT_FILE` or `SSL_CERT_DIR` replaces the system trust store and can cause HTTPS certificate verification failures.

Experimental Runtime backend

- Workloads have network access.
- Startup stops waiting and cleans up when a workload exits or fails before becoming ready.

0.49.2

{{< release-date date="2026-10-21" >}}

GitHub release

Highlights

Docker Sandboxes 0.49.2 updates Go and bundled dependencies as part of Docker's response to the eMudhra certificate authority compromise.

Security updates

- Updated Go to 1.27.3.
- Updated bundled llmman to 0.1.571.

Recommended actions

Updating the `sbx` CLI alone does not update existing sandboxes. Docker has also rebuilt the built-in template images with updated CA certificates.

Follow one of the documented mitigation paths:

- Recreate sandboxes with refreshed images, using `--pull always`. Save any files and unpushed commits before removing a sandbox.
- Update CA certificates in existing sandboxes, then restart applications. Tools that bundle their own CA certificates may require separate updates.

See the security advisory for commands and guidance for custom templates.

0.49.1

{{< release-date date="2026-10-20" >}}

GitHub release

Highlights

Docker Sandboxes v0.49.1 improves kit compatibility, proxy support for cloud connections, and sandbox shutdown reliability.

> [!NOTE] v0.49.0 was skipped. These release notes cover all changes since v0.48.0.

What's new

Kits

- `sbx kit validate` and `sbx kit inspect` accept `--offline` to check a local directory or ZIP kit without starting the daemon or signing in.
- Creating sandboxes from local or Git v3 kits honors build-time `--kit-arg` values, including scoped overrides.
- Sandboxes combining multiple agent kits, such as Claude and Codex, start successfully and receive guidance in each agent's context file.
- On Windows, sandboxes can combine workload and mixin kits, including kits loaded through Windows junctions.
- Kits declaring multiple OAuth credentials authenticate every declared service.
- `sbx kit sign` warns when Git would change the signed file contents and invalidate signatures for `git+` references.
- Reading a kit directory rejects unsupported entries, such as named pipes and device nodes, instead of hanging. `sbx kit pack` rejects symlinks that would produce an archive Sandboxes cannot load.

Sandbox lifecycle and workspaces

- SSH sessions start in the sandbox workspace, matching `sbx exec`.
- SSH-agent access granted by a kit continues working in running sandboxes after the daemon restarts.
- In newly created local sandboxes, Git preserves the host's ignore decisions for paths already present in the primary workspace. This applies to primary workspaces mounted at a Git repository root.
- Sandbox startup waits for Docker to become ready without launching competing Docker daemons.
- Daemon shutdown and sandbox reset no longer hang indefinitely when Docker is unresponsive or a kit registry operation stalls.
- Sandbox reset handles concurrent operations more reliably, avoids false cleanup warnings, and identifies sandboxes it could not fully remove.
- Port changes fail cleanly during sandbox creation or deletion, avoiding stale host bindings.
- SSH connection attempts fail cleanly when the daemon is shutting down.
- Cursor sandboxes no longer show an unexpected Workspace Trust prompt for workspace paths containing characters such as `.` or `_`.

Networking and cloud sandboxes

- Explicit CIDR deny rules take precedence over matching domain allow rules when Sandboxes resolves the destination address.
- Cloud commands, `sbx move`, the TUI, and cloud SSH honor configured proxies, including system and PAC proxies. Set `proxy.sshDirect` to `true` to connect cloud SSH directly on networks that permit direct SSH connections.
- `sbx mcp add` completes OAuth discovery on networks that require an HTTP proxy, without needing `--skip-ssrf-check`.
- Fixed uploads stalling while waiting for HTTP 100 Continue through intercepted HTTPS connections.
- Fixed a Windows system-proxy watcher hang that could prevent the daemon from stopping.
- Image pulls from `public.ecr.aws` work under the `balanced` network policy.
- `sbx policy refresh` immediately refreshes policy, for example after an AI Governance change, instead of waiting for the next polling interval. Existing connections remain open.
- `sbx policy ls` continues showing `default-deny-all` after a kit adds network rules.
- Cloud connection and authentication errors include guidance for resolving the problem.

Authentication and MCP

- Custom secrets using `OPENAI_API_KEY` or `ANTHROPIC_API_KEY` appear in the sandbox as expected.
- `sbx secret set-custom` warns before replacing a different host or environment-variable mapping under the same placeholder.
- `sbx secret rm` returns an error when no secret matches `--registry`, `--placeholder`, `--host`, or `--env-name`.
- MCP gateway code-mode tools work on Windows hosts.

CLI, settings, and diagnostics

- `sbx policy ls`, `sbx settings ls`, and `sbx policy approval ls` accept `-q` to print only rule IDs, setting keys, or approval IDs.
- Configure daemon log size and retention with `sbx settings set` using `daemon.log.maxSizeMB`, `daemon.log.maxBackups`, and `daemon.log.maxAgeDays`.
- The `browser.enabled` setting controls whether sandboxes can open the host browser.
- `sbx env` reports unknown keys and invalid values with their source file, line, column, and field. Invalid values in a base file must be fixed even if a later file overrides them; this also applies to `~/.sbxenv.yaml` and `sbx env rm`.
- `sbx settings set` rejects incorrectly structured values for `kit.allowedSources` and `kit.trustedSigners` before saving them.
- `sbx cp` clearly reports a missing destination directory without extra tar errors.
- `sbx diagnose` reports running CLI and shim processes, warns when global network policy is not initialized, and detects a ZFS setting that can make image builds unusually slow.

Templates and installation

- Full templates now include a C/C++ toolchain for compiling native code inside the sandbox.
- The Linux tarball installer works for regular users on systems with AppArmor.

0.48.0

{{< release-date date="2026-10-13" >}}

GitHub release

Highlights

Docker Sandboxes v0.48.0 updates the CLI across all commands, with consistent terminal output and JSON/YAML formats. This changes flags, structured output, and prompts used by existing scripts. Review the migration details below before upgrading automated workflows.

The release also adds managed Codex and Claude Desktop SSH targets on macOS and Windows, additional v3 kit capabilities, and support for adding kits with startup commands to existing sandboxes.

Breaking changes

CLI flags, output, and automation

The CLI refactor changes both interactive use and scripting interfaces. Existing `--json` parsers may need updates even if their command lines still work. Scripts that parse human-readable output must switch to `--format json` or `--format yaml`.

Update command flags

Command	Previous spelling	Replacement or behavior
`sbx run`, `sbx env run`	`--detached`	Use `--detach` or `-d`. `--detached` is rejected.
`sbx logout`	`--yes`, `-y`	Use `--force` or `-f`.
`sbx env plan`, `sbx env create`, `sbx env run`	`--auto-approve`, `-y`	Use `--force` or `-f`.
`sbx create`	`--quiet`, `-q`	Use `--no-progress`. This suppresses progress, but still prints the creation summary.

Command	Previous spelling	Replacement or behavior
`sbx template save`	`-d` for the description	Use `--description`.
`sbx diagnose`	`--output`, `-o`	Use `--format table`, `--format json`, `--format yaml`, or `--format github-issue`.
`sbx secret ls`, `sbx secret rm`	`--env`	Use `--env-name`.
`sbx policy log`	`--quiet`, `-q`	Removed; there is no replacement.

`--format`, `--json`, and `-q` are mutually exclusive on commands that use the shared output flags. Even `--json --format json` is rejected. Unsupported format values fail during flag parsing instead of falling back to human-readable output.

`sbx settings ls` is the primary spelling; `sbx settings list` remains an alias. `sbx kit builder history rm` keeps `--yes` / `-y` because its `--force` flag is forwarded to Buildx. The deprecated `--global` and `mcp add --skip_auth` flags remain accepted but hidden, with a deprecation warning on stderr; use `--skip-auth` instead of `--skip_auth`.

Update structured-output parsers

`--json` is an alias of `--format json` and emits the same document. JSON output structures have changed for the commands listed in the following table. Output uses `snake_case` field names; YAML is derived from the same representation.

Command	Output change
`sbx ls`	Returns the sandbox array directly instead of `{"sandboxes": [...]}`. Fields use `snake_case`, `created_at` and `last_used_at` use RFC 3339 timestamps, and rows include `detached`.
`sbx template ls`	Returns the image array directly instead of `{"images": [...]}`, for local and cloud sandboxes.
`sbx mcp ls` locally	Returns the server array directly instead of `{"gateway": {...}, "servers": [...]}`. Each server row carries `gateway`.
`sbx policy log`	Local and cloud output use `{updated_at, allowed: [...], blocked: [...]}`. Entries contain `sandbox_id`, `host`, `proxy_type`, `rule`, `reason`, `count`, `first_seen`, and `last_seen`. `--limit` applies. This replaces local `allowed_hosts` / `blocked_hosts` and the cloud camelCase document.
`sbx --cloud policy ls`	Field names change from camelCase to `snake_case`.
`sbx ports`	Cloud rows use `sandbox_port` instead of `port`, alongside `url`. Local rows use `host_ip`, `host_port`, `sandbox_port`, and `protocol`.

Command	Output change
`sbx ports --publish` / `--unpublish` locally	Structured formats emit result rows with `action: published` or `action: unpublished` instead of human-readable lines. Warnings go to stderr.
Cloud `volume create`, `volume ls`, and `volume inspect`	Fields such as `ID`, `Name`, and `CreatedAt` become `id`, `name`, and `created_at`. `volume inspect` defaults to a human-readable details view; request `--format json` for JSON.
`sbx rm`, `sbx stop`, `sbx template rm`, `sbx volume rm`	Structured formats return an array of results with `action`, `resource`, `name`, and optional `id` and `error`. Actions are lowercase: `removed`, `stopped`, `removing`, `stopping`, or `failed`. Missing and failed targets are included.

For example, a script that extracts sandbox names from `sbx ls --json` must change its filter from `.sandboxes[].name` to `.[].name`.

Local and cloud output follow the same conventions. MCP list output still represents different resources: local registrations versus server usage across cloud sandboxes, so those rows differ beyond their shared `name` and `type` fields.

`sbx prune --dry-run` accepts `--format json` and `--format yaml`. Cloud `ttl --format yaml` emits YAML instead of JSON, and `sbx diagnose` supports YAML. `sbx events --json` remains an NDJSON stream, and `sbx mcp auth --format text|json` retains its command-specific formats.

Update non-interactive workflows

- Confirmation prompts require terminal input. Piping an answer, such as `echo y | sbx secret set …`, no longer confirms the operation. Use `--force` / `-f` where the command provides it. Commands that require confirmation fail with exit status 1 when stdin is not a terminal.
- Declining a confirmation returns a non-zero exit status consistently, including for removal, reset, and prune operations.
- Read-only invocations and runs with `--force` do not start an interactive sign-in flow. Sign in before running automation that requires authentication. Read-only examples include `ports SANDBOX`, `ttl SANDBOX`, `prune --dry-run`, and `diagnose`; their modifying or upload variants are not read-only.
- Without a terminal, `sbx diagnose --upload` saves the diagnostic bundle but skips uploading it and exits with status 0. A successful exit does not mean that an upload occurred.
- Without a terminal, `sbx create` and `sbx run` fail if the workspace directory does not exist and print a `mkdir -p` hint. Create the directory before invoking the command.
- Ctrl-D at a selection or free-text prompt cancels instead of accepting the default. Enter still accepts the default.

Terminal output and streams

Human-readable output changes across all commands: headings, status marks, indentation, progress, and result messages use a consistent format. `sbx ls` includes state marks and a creation-time column; `sbx ls` and `sbx inspect` identify detached sandboxes.

Results go to stdout. Errors, warnings, corrective hints, daemon-start progress, image pulls, and kit-build progress go to stderr. Update scripts that depend on exact message text or capture only one stream.

What's new

Security

- Fixed sign-in enforcement ignoring `allowedOrgs` on macOS.
- `sbx cp` blocks local paths that escape a sandbox workspace through symbolic links.
- Cloud `sbx cp` downloads and `sbx kit pack` preserve restrictive destination permissions without temporarily exposing the replacement file. They account for permissions at completion and do not widen them. On Unix, cloud `sbx cp` refuses to replace files owned by another user; on Windows, permission preservation covers the read-only attribute. Special permission bits are retained where the kernel permits. This cloud-copy change does not affect local `sbx cp`.
- `sbx kit add` checks the filesystem mount policy before recreating a sandbox and refuses the operation if its workspace or a saved mount is denied, matching `sbx start`.
- Fixed the daemon crashing on a malformed request sent through an intercepted HTTPS connection.

SSH and desktop integration

- On macOS and Windows, `sbx setup ssh --auto-create` registers Docker Sandbox connections in Codex and Claude Desktop. Each connection creates a dedicated sandbox on first use and reuses it on subsequent connections. These sandboxes have no host workspace mounted. Clone or copy your project into the sandbox to work on it.
- Desktop SSH setup warns about missing authentication and provides local sign-in guidance for Codex and Claude. Repeating `sbx setup ssh` recognizes existing registrations even while the applications are running.
- Closing a Cursor or VS Code window connected over SSH no longer stops the daemon and other running sandboxes.
- SSH connections no longer fail because the CLI's initial feature lookup timed out before starting the daemon.
- SSH-agent forwarding supports Windows OpenSSH and 1Password agents when `SSH_AUTH_SOCK` is not set.

Kits

- V3 kits support capability groups, `kit.env` references, bundled agent skills, and the `ssh-agent@1` and `git-identity@1` capabilities. Selected optional groups retain their network rules, lifecycle files, and context; unsupported optional groups are skipped. Bundled skills can use directories outside the agent's home.
- `sbx kit add` accepts kits with startup commands. Those commands run after the kit is added and on each subsequent sandbox start.
- Sandbox containers expose the provisioned agent's name through `SBX_AGENT` for kit hooks and scripts.
- `sbx kit validate` rejects kits whose credentials route the same host, identifying the conflicting services. The daemon logs the collision during sandbox creation.
- Sandbox creation supports kits whose resolved lock exceeds Docker's container-label size limit.
- `sbx kit pack` excludes its output from the archive, prevents writes into the kit's `files/` directory, and handles Windows 8.3 output paths without renaming the destination or including a previous archive. The output path must end in `.zip` and cannot be a named pipe, socket, or device node.
- An unresolvable kit-file path no longer causes `sbx kit push` to fail.
- Kits that declare an agent-context directory now have their agent instructions file written in that directory instead of beside the workspace, and large workload and mixin kit combinations no longer fail to create with "failed to run sandbox container".

Sandbox lifecycle and workspaces

- Creating a sandbox from a local snapshot saved with this version avoids registry requests by default, including when using the daemon API.
- `sbx exec`, `sbx run`, `sbx cp`, and SSH report an unresponsive guest with stop-and-start guidance instead of hanging. `sbx ls` marks the sandbox as not responding.
- Commands against an already-running sandbox reuse guest setup instead of repeating it on every `exec`, `run`, or `cp` invocation. Setup runs once per container start or daemon restart.
- `sbx reset` no longer leaves stray sandbox resources or stalls when another sandbox operation is in progress.
- Sandbox startup identifies files still in use from a previous run instead of reporting a generic failure.
- `sbx exec` accepts empty command arguments.
- `sbx cp` preserves dot-prefixed paths and restrictive directory permissions. On Windows, downloads can replace a read-only destination instead of failing with “Access is denied”.
- `sbx run` and `sbx create` support workspace directories whose names start with a dot.

Cloud sandboxes

- `sbx move --to cloud` and `sbx --cloud env create` accept `--ttl` and `--on-timeout restart`, extending always-on support beyond `sbx --cloud create`.
- Fixed `on_timeout` response decoding errors when using `--on-timeout`.

Authentication, agents, and MCP

- Sandbox-scoped custom secrets can override global secrets that use the same environment variable name.
- Fixed Docker Hub sign-outs on macOS when the host sleeps or wakes during login-token refresh.
- The built-in Codex kit registers the MCP gateway with supported settings, removing ignored-setting warnings and sending the gateway its Authorization header. Recreate existing Codex sandboxes with `sbx rm` and `sbx create` to receive the change.
- `sbx mcp unload NAME --sandbox SANDBOX` detaches a loaded MCP server without restarting the sandbox's agent.
- MCP connections and discovery probes identify as `docker-sbx`, and backend recovery respects rate limits to avoid reconnect storms.
- `sbx run --model --provider` uses the API key stored with `sbx secret set <provider>` without exposing it to the sandbox. `sbx run --variant` selects a model thinking variant.
- Fixed `sbx run --model` failing with “401 Unauthorized” when pulling from Docker Hub after a browser-based `docker login`.

Networking and policy

- The proxy handles trailing-dot hostnames and mixed-case `host.docker.internal` consistently, fixing refused requests and missing credential injection.
- Sandbox image pulls retry temporary registry failures reported as HTTP 5xx errors, including when the original error type is lost across API or progress-stream boundaries.
- Fixed WebSocket connections through the proxy occasionally losing the first bytes sent by the client.
- `sbx policy ls SANDBOX` returns an error when the sandbox does not exist.
- `sbx policy inspect` accepts policy names that match one of their own rule names, such as `default-deny-all`.
- Removing a kit-managed network rule by resource explains that the kit manages it instead of asking for a rule ID.

CLI and diagnostics

- Shell completion scripts generate without starting telemetry, reducing generation time.
- `sbx diagnose` checks for `mkfs.ext4`.

Earlier releases

For older versions, see the Docker Sandboxes releases on GitHub.

□□□ #2
□ □□□ □□□ 2026-10-31 19:02:07 UTC
□ □□□ □□□ 2026-10-31 19:02:09 UTC