

CommitScope 2.4.2 | User Guide

01 / One complete review

Know the finish line before opening a terminal

A completed corporate run means: the employee reviewed an exact clean Git commit under an approved external policy; Semgrep, Gitleaks, Trivy, Claude Hunter and a fresh Claude Verifier all completed; the protected run was handed unchanged to an assigned reviewer; that person ran verify-review and recorded a separate decision. Hunter and Verifier are mandatory. A scanner-only demo, scan, ai, or GitHub Action is partial evidence.

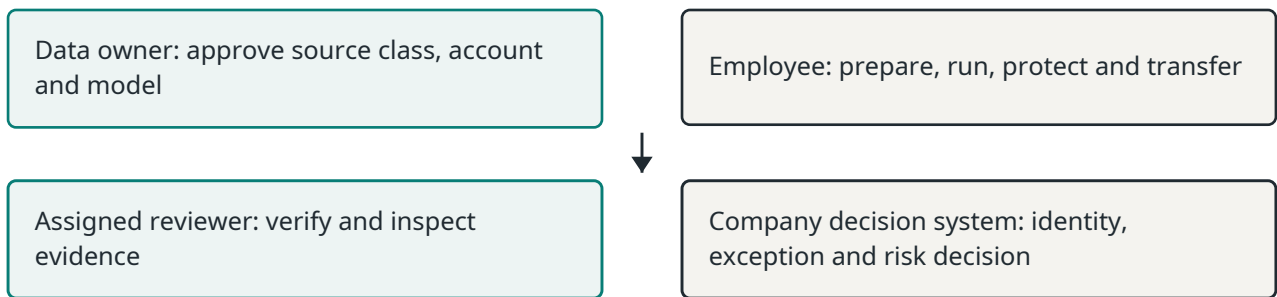


Figure 1. Roles and handoff; tools never become the risk owner.

READY_FOR_HUMAN_REVIEW is a handoff state, not a merge approval or a claim that the application is secure. A real repository needs its own source-transfer approval and human review; this book uses synthetic material only.

02 / Install and diagnose

Install the pinned PyPI package on a supported host

Objective: start from reviewed v2.4.2, not an arbitrary moving branch. CommitScope supports macOS and glibc Linux on x86_64/ARM64 with Python 3.11-3.14; native Windows is outside its scope. Use WSL2 with a supported Linux distribution. Have Git, pipx/Python, certificate roots and approved outbound HTTPS. [G1]

```
pipx install commitscope==2.4.2
commitscope --version
commitscope preflight
commitscope bootstrap
commitscope doctor
```

Expected: version 2.4.2, host prerequisites accepted, pinned Semgrep/Gitleaks/Trivy installed in CommitScope state, and doctor confirming scanner presence and version. Bootstrap downloads scanner executables and installs Semgrep dependencies; the Trivy database is retrieved by scan/demo, not bootstrap. Bootstrap does not execute or install the target application. Tool and DB details are recorded in run evidence.

Recovery: if preflight rejects OS, architecture or Python, move to a supported host. If bootstrap or doctor fails, repair network, certificates, free space or scanner state and rerun diagnostics. Do not call a failed doctor a completed review or bypass a mandatory scanner.

03 / Claude Code account and consent

Use the intended personal or company account

Objective: make the future Hunter and Verifier requests from the approved unprivileged OS user and account. Install official Claude Code 2.1.259 or newer under that user. The local wrapper pins 2.1.278 for new installs but does not upgrade an existing CLI; update older versions via Anthropic's documented channel. Claude Code's own platform support is broader than CommitScope's. [G2]

```
claude --version
claude auth login
claude auth status
```

Expected: a supported CLI version and a valid login for the account the organization authorized. Confirm which credential and privacy terms actually apply. Personal and commercial data-use and retention policies differ; a personal subscription is not automatically permission to upload company source. The corporate review has --auth account only, rejects ambient API/provider overrides, and has no API fallback. [G3] [G4]

Recovery: if login, quota, model access or policy approval is missing, stop before review. Sign in again as the intended user, resolve conflicting ambient credentials, obtain owner authorization, or choose the approved exact model ID. Do not paste a token into the guide or switch providers silently.

04 / Prepare a synthetic target

Create a clean full-commit training repository

Objective: practice with the bundled synthetic IDOR example, not company code. The commands below assume /protected is an approved, user-owned 0700 parent outside the target repository; replace it with an approved absolute path on your host. Obtain the v2.4.2 source solely to access its fixture and example policy. Never use a live repository until its owner permits source transfer.

```
git clone --branch v2.4.2 --depth 1 \
  https://github.com/akarazhev/commitscope.git \
  /protected/commitscope-training-source
install -d -m 700 /protected/training-target
install -d -m 700 /protected/reviews
cp /protected/commitscope-training-source/examples/ai-acceptance/idor/vulnerable/app.py \
  /protected/training-target/app.py
install -m 600 /protected/commitscope-training-source/examples/review-policy.json \
  /protected/review-policy.json

git -C /protected/training-target init -q
git -C /protected/training-target add app.py
git -C /protected/training-target \
  -c user.name=Training -c user.email=training@example.invalid \
  commit -m "Synthetic IDOR baseline"
git -C /protected/training-target status --short
git -C /protected/training-target rev-parse HEAD
```

Expected: an empty status output and one full lower-case 40-character commit ID. Inspect and approve the copied policy outside the target: scope, threats, invariant AUTH-001, threshold and upload allowance. The synthetic fixture has no third-party dependency manifest, so its live review uses an explicit --allow-empty-sca reason after inspection. On a real target, do not use that exception to conceal a missing lockfile.

Recovery: if status is not empty, commit or remove every change, including untracked files. If policy contains recognizable credential forms such as Bearer/Basic authorization values or URLs with user information, it is rejected before scans. Screening of source packets is heuristic and cannot prove that unknown secrets are absent. Fix the source or policy and seek new approval before upload.

05 / Practice, then consent-gated review

Know which command produces which evidence

Objective: distinguish a scanner exercise from a full local review. First run the scanner-only demo in a new directory; it makes no Claude request and is not corporate acceptance. Its scanner results depend on current databases. Then, only after explicit owner approval for this synthetic source, run the consent-gated review using the full SHA printed above and an approved exact model ID.

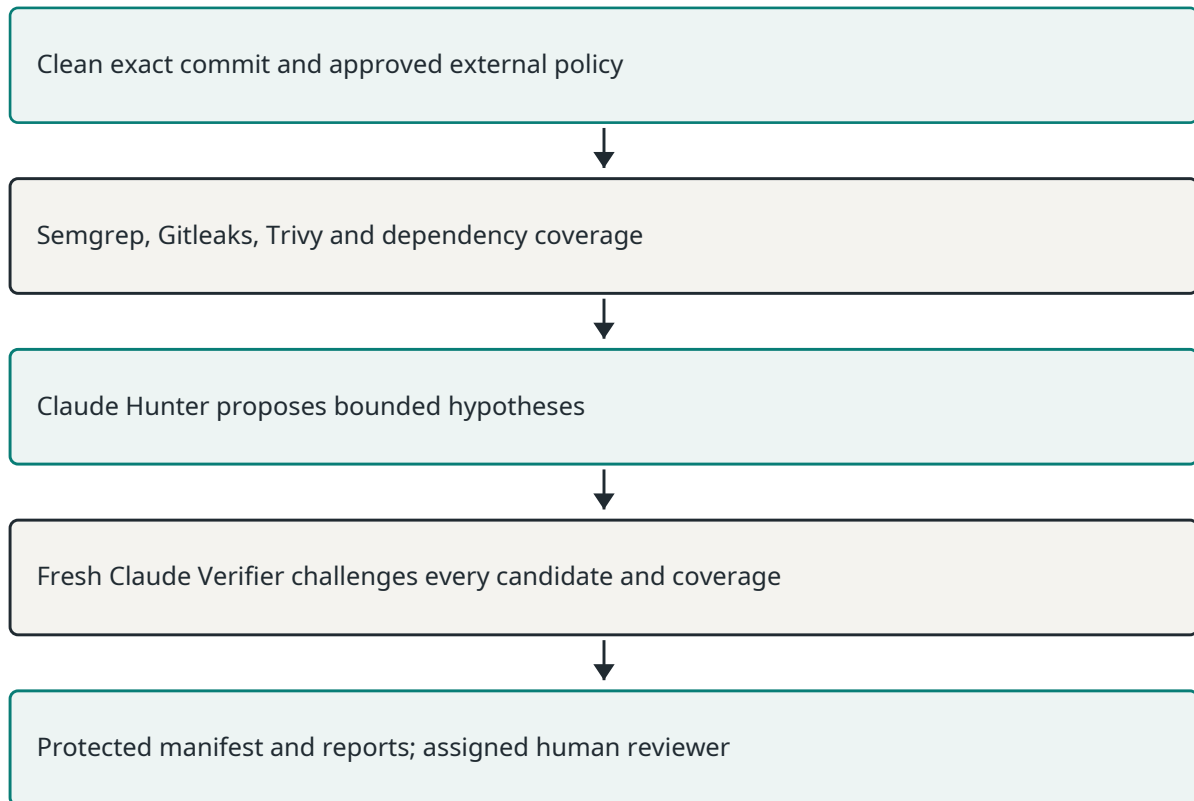


Figure 2. One full run: commit, scanners, independent model stages, evidence, human handoff.

```
commitscope demo --out /protected/reviews/scanner-demo
```

Expected: DEMO_PASSED when all synthetic scanner checks match. SCANNERS_VERIFIED_AI_NOT_RUN is not completed corporate acceptance. A demo failure means investigate tool or DB drift, not edit the expected result to green. The demo's output is separate from the review run.

```
commitscope review \
--repo /protected/training-target \
--ref 0123456789abcdef0123456789abcdef01234567 \
--policy /protected/review-policy.json \
--out /protected/reviews/run-id \
--auth account --allow-code-upload \
--model APPROVED_EXACT_MODEL_ID \
--allow-empty-sca "Synthetic stdlib-only fixture; reviewed imports and metadata"
```

This is a command template, not an observed live run. Replace the example SHA with the actual full ID from g04, APPROVED_EXACT_MODEL_ID with the authorized exact ID, and run-id with a new name. --allow-code-upload explicitly permits the screened source packet to reach Anthropic under the chosen account's terms. No Claude request is made by reading or building this guide. [G4]

Expected: one of the three states in g06 and a protected run directory. Recovery: an authentication, quota, model, timeout, malformed response or required-stage failure is INCOMPLETE. Preserve that directory, fix the cause, and start a new run; do not reuse the output path or fall back to an API key.

06 / Read states and findings

Translate program exit into a reviewer action

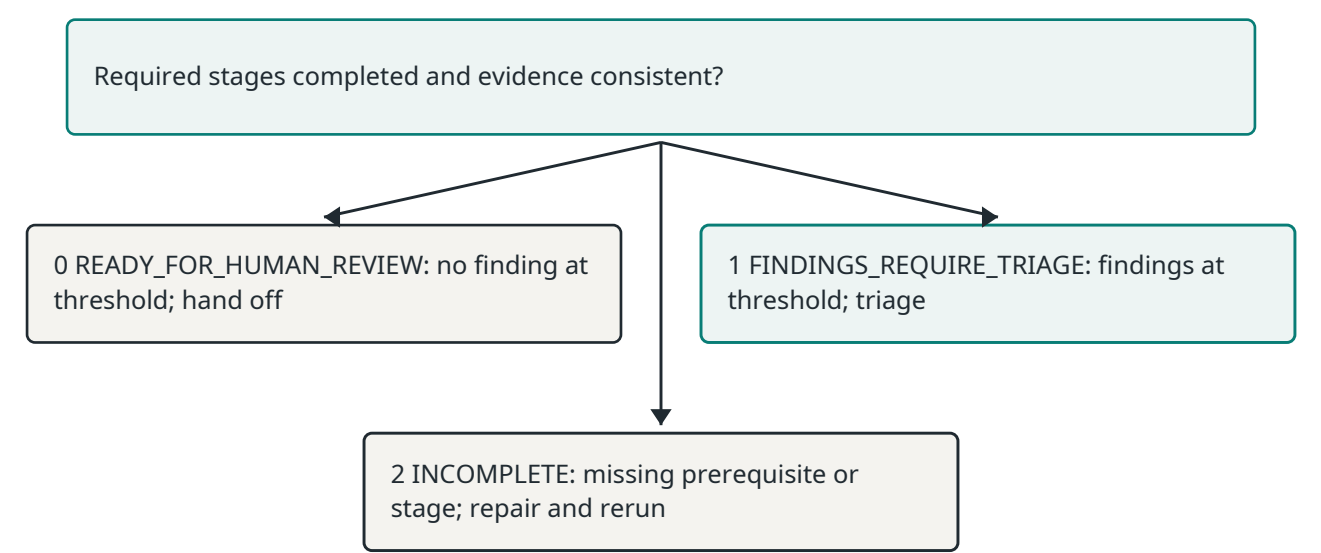


Figure 3. All three machine states lead to preservation; only complete runs can be triaged.

Table 1. Status, evidence meaning, and next action

Exit / status	Meaning	Next action
0 / READY_FOR_HUMAN_REVIEW	Required stages complete; no normalized finding at policy threshold	Reviewer still inspects and decides
1 / FINDINGS_REQUIRE_TRIAGE	Required stages complete; findings reach threshold	Reviewer validates, prioritizes, or rejects
2 / INCOMPLETE	Prerequisite, coverage or consistency failed	Keep evidence, repair, run again

Read the top-level report.md first, then report.json for normalized finding fields and report.sarif for compatible tools. Follow each finding to evidence/ and the exact code location; inspect the strongest counterargument and omissions. Raw scanner output, source packets, model envelopes and diagnostics under private/ are confidential. A synthetic IDOR candidate is an example of a hypothesis, not proof of live model performance.

Illustrative synthetic record, not observed model output: app.py:9-11 drops requesting_tenant and returns ORDERS[order_id]. Invariant AUTH-001 says a caller may access only its tenant's data. Evidence to test: tenant-a can request order-2 belonging to tenant-b. Counterargument: an upstream route might enforce ownership, so inspect the actual call chain before confirming. Human decision: the assigned reviewer records confirm, reject or needs-more-evidence separately in the protected decision system.

If a finding lacks a reachable path, violated invariant or credible evidence, record the uncertainty; do not silently delete it. If coverage is absent, classify the run INCOMPLETE. READY_FOR_HUMAN_REVIEW never authorizes a merge.

07 / Protect, transfer, verify

Give the reviewer the unchanged run

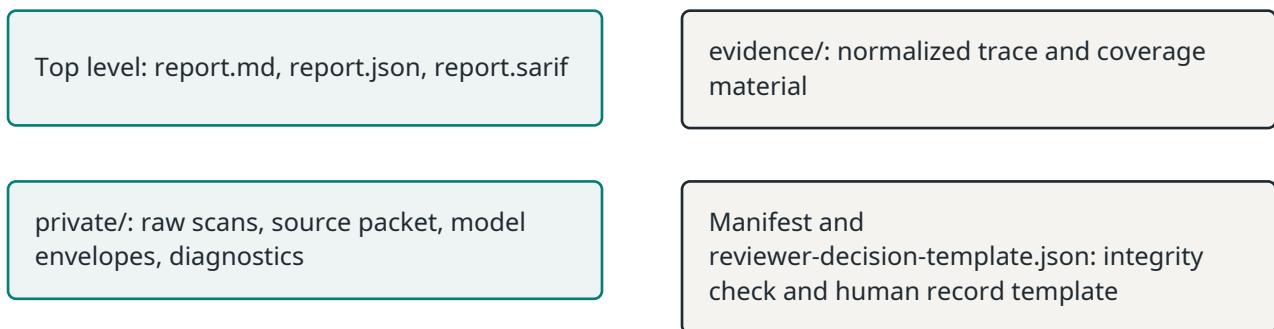


Figure 4. Run directory anatomy; the entire directory is confidential.

Objective: preserve the original 0700 directory and 0600 files, then send that unchanged directory through the company's approved protected channel to the named reviewer. Never commit or publish private/. Normalized reports and evidence/ may be shared only under company rules. The reviewer must obtain and verify the entire original run, not just a Markdown excerpt.

```
commitscope verify-review --run /protected/reviews/run-id
```

Expected: verify-review checks layout, file types and permissions, hashes, commit/snapshot and policy/resource identity, scanner and Hunter/Verifier coverage, and decision consistency. A valid complete run returns its recorded 0 or 1 class; changed, missing or incomplete evidence returns 2. The reviewer inspects evidence and limitations, then completes a copy of reviewer-decision-template.json in the protected company decision system with the verified manifest hash.

The manifest is unsigned. verify-review establishes internal consistency, not authorship or immutable custody. Human identity, exception approval, risk acceptance and storage audit are external controls. Recovery: if verification fails, retain the original, investigate the transfer and obtain a fresh unchanged run; never edit the sealed evidence.

08 / Fix and rerun

A new commit is a new review

Objective: see how remediation changes evidence without overwriting history. In training, copy the bundled fixed IDOR app.py into the same synthetic target, commit it, and record the new full SHA. For company code, apply the approved fix and functional/security tests instead. Use a fresh output directory and repeat all scanners, Hunter, Verifier, verification and human triage.

```
cp /protected/commitscope-training-source/examples/ai-acceptance/idor/fixed/app.py \
/protected/training-target/app.py
git -C /protected/training-target add app.py
git -C /protected/training-target \
-c user.name=Training -c user.email=training@example.invalid \
commit -m "Synthetic ownership fix"
git -C /protected/training-target rev-parse HEAD
```

Expected: a different full commit ID. Run the g05 review command again with this new SHA and /protected/reviews/run-fixed as --out, then verify and hand off. Keep run-id and run-fixed unchanged. Compare normalized findings, coverage and policy; disappearance alone is not proof of remediation. Have the reviewer confirm the ownership invariant in authorized tests and record a new decision.

Recovery: if the new tree is dirty, commit or clean it before review. If the run is INCOMPLETE, repair and start another fresh directory. If a finding remains or changes, triage it rather than assuming the fix succeeded.

09 / Troubleshooting and reference

Diagnose the failed boundary, not just the exit code

Table 2. Observable problem, likely boundary, next action

Observation	Likely cause	Action
preflight fails	Host, Python, Git or path restriction	Use supported host; inspect exact diagnostic
doctor fails	Pinned scanner missing or wrong version	Repair bootstrap and rerun doctor
review rejects policy	Credential-like data or invalid external policy	Remove material; obtain approval again
review rejects repository	Dirty tree, abbreviated SHA or unsafe path	Clean tree; use full SHA and safe paths
Claude stage INCOMPLETE	Login, quota, exact model, timeout or malformed result	Resolve account/stage; new run
verify-review returns 2	Changed, missing, unsafe or incomplete evidence	Preserve original; investigate transfer
Finding disappears	Changed code or detection variability	Test invariant and re-review exact commit

```
commitscope preflight
commitscope bootstrap
commitscope doctor
commitscope demo --out /protected/reviews/scanner-demo
commitscope review --help
commitscope verify-review --run /protected/reviews/run-id
```

A command returning 0 is not automatically a security decision. The GitHub Action and compatibility scan/ai commands are scanner or partial diagnostics; the assigned reviewer is the gate for a real repository. Check this guide against the installed v2.4.2 CLI and revalidate model and scanner behavior over time. [G1] [G2] [G5]

Sources and verification date

- [G1] CommitScope 2.4.2 on PyPI. <https://pypi.org/project/commitscope/2.4.2/> (2026-09-24)
- [G2] Claude Code advanced setup and installation. <https://code.claude.com/docs/en/setup> (2026-09-23)
- [G3] Claude Code authentication. <https://code.claude.com/docs/en/iam> (2026-09-23)
- [G4] Claude Code data usage. <https://code.claude.com/docs/en/data-usage> (2026-09-23)
- [G5] Claude Code security and untrusted content. <https://code.claude.com/docs/en/security> (2026-09-23)