

Suhail

Client setup guide

Connect your first app. Understand the results. Check your fixes.

Practical steps, expected results and troubleshooting for your own Suhail workspace.

Contents

01 Start here	3
02 Find the right screen	4
03 Account, sign-in and recovery	5
04 Connect GitHub and iOS source	6
05 Connect Cloudflare resources	7
06 Scans, progress, coverage and inventory	8
07 Triage findings and verify fixes	9
08 Repair dependencies and exposed credentials	10
09 AutoFix and Claude-assisted repairs	11
10 Review pull requests	12
11 Publish a GitHub check and set merge policy	13
12 Team review access	14
13 Check a running application	15
14 Running-app results, stopping and retests	16
15 Run scoped security tests	17
16 Security-test evidence and retesting	18
17 Notifications, monitoring and extended reviews	19
18 Plans, usage, cancellation and data	20
19 Troubleshooting by symptom	21
20 Get help and give feedback	22

01 | Start here

Suhail helps small studios review selected security checks across GitHub source, Cloudflare resources and configured app routes. This guide walks you from a new workspace to a checked fix, with optional workflows after the essentials. You use the website; you do not install a server or keep your computer running for scheduled checks.

Before you start: have your account access, permission to connect the repository, its actual branch name and a recent pushed commit. Add Cloudflare and running-app checks after your first repository result. A normal modern desktop browser is easiest for initial setup; everyday review also works on mobile.

The first-session goal: connect one app, inspect one completed check, understand any coverage gaps and confirm your notification destination. Optional repair, PR and running-app controls can wait until you need them.

Your copy of this guide: download it from the setup screen or Help & history, or read the online version at <https://suhail.dataloup.io/setup-guide.html>. Saving your workspace queues one setup email to its account address. That operational email is separate from monitoring-alert preferences. It contains public guide links, not credentials. Delivery status appears on the setup screen; downloading always remains available.

What a result means: a check covers its recorded files, routes, revision and time. Zero findings with incomplete coverage does not establish that an app is secure. Follow the named gaps and next action on your own result.

Your first useful session

1. Open <https://suhail.dataloup.io> and use your existing sign-in route. For a linked account, use **Sign in with Dataloup**. Confirm the workspace before connecting anything.
2. Complete **Set up your workspace**: name, first repository, first check, coverage acknowledgement and notification preferences.
3. In **Apps**, open the intended app. Confirm repository and branch, then inspect **What was checked** and the recorded commit.
4. Open **Findings**. Review known-exploited dependency matches and high/critical findings first, alongside coverage gaps.
5. For a valid issue, fix the source or preview a supported repair. Review project CI and merge yourself. Request verification or allow the next enabled scan to verify the changed commit.
6. In **Settings > Connections & alerts > Notifications**, test the delivery channel you intend to use. A queued message is not a received message.

Success: one recognisable app, the correct source revision, intelligible coverage and a working alert destination. If the first run is incomplete, use its specific reason rather than repeatedly reconnecting the app.

02 | Find the right screen

The sidebar has five main destinations: Overview, Apps, Findings, Monitoring and Settings. Help & history contains the setup guide, contact and feedback links, and specialist reference pages. On a small screen, open **Menu**. Page labels below follow this guide edition; permissions determine which actions are available.

Destination	Use it for
Overview	Workspace attention, current status and the next useful action.
Apps	Repository-connected apps, progress, settings, coverage and connections.
Findings	Review, false positives, repair proposals, exports and fix verification.
Pull requests	Manual reviews, automatic reviews, sharing and optional GitHub checks.
Team	Invitations, shared workspace selection, roles and removal.
Settings > Connections & alerts	GitHub and Cloudflare resources; email and Telegram delivery.
Monitoring	Incidents, background-check health, exports and installed-release detail.
Security review	Optional recent history, public assets, provider alerts and account posture.
Running app	Verified origin, public checks and dedicated-account read checks.
Security tests	Manually authorised session, CORS, cache and record-boundary probes.
Pack Money tests	Existing dedicated synthetic Pack Money suites; separate from generic testing.
Releases & checks	Retained local release/check records; not proof of current cloud protection.
Profiles / Scope & access	Capability descriptions, permission scope and limitations.
Activity	Recorded events and review history.
Account & security	Native credentials, passkeys, recovery, sessions and Dataloup linking.
Product & usage	Current product access, shared allowance and period state where exposed.

App, repository and product are related but different. A connected Suhail app selects a repository and branch. A commercial product can include several repositories within its allowance. Cloudflare connections attach deployed resources to that app. A Together subscription additionally pairs matching product identities across Suhail and NosyPup. A matching name alone does not prove a confirmed commercial mapping.

Use an app's overview and contextual links to stay with the intended repository. In shared workspaces, confirm the selected workspace before reading or changing findings. Never create a replacement workspace to recover a missing original one.

If a control is missing: first check whether you are the owner, a team reviewer or a named report reader. Then check app state, supported connection type, allowance and required permissions. An absent button does not necessarily mean the build failed.

03 | Account, sign-in and recovery

Prerequisites: existing authorised access and the correct production sign-in address. The public Suhail login currently says new customers join by invitation. Do not infer open registration from the presence of a central billing page.

Use the correct identity

1. For existing native Suhail access, use the password/passkey controls on its login screen. For an already linked account, select **Sign in with Dataloup** or open Suhail from the central account product card.
2. Check the workspace name and existing apps after sign-in. Test and production identities may open different workspaces.
3. For an existing native account that needs linking, open **Account & security**, find the Dataloup account section and follow its link-and-confirm flow. Review the displayed workspace and app mappings before confirming.
4. If mapping is incomplete or a conflict appears, preserve the existing account. Refresh and resolve the named link error; do not overwrite another account's mapping.

Success: the same intended Suhail records are accessible through the linked route. Linking is not a subscription purchase and does not grant NosyPup general access to security findings.

Native Suhail passkeys

1. In **Account & security > Add your first passkey**, give it a recognisable name and confirm your current password.
2. Review the passkey-only option and recovery-code confirmation, then choose **Enable passkey sign-in**. Finish the device/password-manager prompt.
3. Save the recovery codes securely. Enabling or changing these controls signs out browsers; sign in again and check the intended method works.
4. To add another passkey, verify with an existing passkey or unused recovery code first. Keep a working fallback before removing a device.

Native Suhail passkeys and central Dataloup passkeys belong to their respective sign-in services. A QR prompt can mean the browser is offering another device, not that a credential was lost. Check the website stored with the passkey. Do not delete a working passkey merely to dismiss a prompt.

Recover or end access

Use **Forgot password?** on the native login if needed. Source-defined native reset links are single-use and expire after 30 minutes; enabled second-factor requirements remain. Recovery codes work once. Replacing the set invalidates the old set. Removing the last native passkey returns native login to password-only and deletes those recovery codes.

Sign out everywhere ends current browser sessions. Routine scheduled checks continue, but security/account changes can invalidate delegated PR automation; inspect its status and re-enable deliberately if required. Unlinking Dataloup ends linked sessions without deleting product records. It does not cancel billing.

If recovery fails: contact support with sign-in domain, workspace, time and error. Never send a password, passkey QR, reset link or recovery code.

04 | Connect GitHub and iOS source

Prerequisites: source committed and pushed to GitHub, authority to install/approve the app and the intended branch. Suhail cannot inspect changes that exist only on your Mac.

1. Open **Apps > Connect an app** or **Connect with GitHub** during onboarding.
2. Choose **Continue with GitHub**. Select only the repositories needed for this workspace in the GitHub installation.
3. Back in Suhail, choose the account/organisation and repository. The selection session lasts ten minutes; restart it if expired.
4. Under **App in Suhail**, choose an existing app when reconnecting it so its history stays attached. Confirm app name and branch.
5. Select **Connect and queue first check**. The expected response is a connected app and queued first check, normally picked up from the next scheduler minute subject to capacity.
6. Use **View scan progress**, then inspect the recorded repository, branch, commit and coverage. A connection alone is not protection evidence.

If a repository is absent: check the installation's selected repositories, organisation approval and pagination. Search filters only the currently loaded repository page. Do not broaden access to all repositories merely to make one appear.

Token alternative

Apps > Add an app also accepts a fine-grained, repository-restricted token with a recorded expiry. Use this only where appropriate. Paste it into the product's secure field, not chat or email. Token connections can support ordinary scans and eligible patch downloads; the current hosted draft-PR and focused PR-review workflows require the GitHub App.

Connect an Apple app from Xcode

Keep your project's Swift source, supported configuration files and relevant dependency lockfiles in its repository. Commit and push through your normal Xcode/Git workflow, then connect that repository and the actual working branch in Suhail. Check that the next recorded commit equals the pushed revision. If source or a lockfile is ignored, a local build cannot make it visible to the scanner.

Swift checks are bounded source and dependency checks. They do not build an IPA, inspect the signed shipping app, prove Keychain behaviour on a device, test all iOS networking or replace App Store privacy review. The added storage rules examine selected explicit Keychain accessibility and no-file-protection patterns; ambiguous syntax remains a gap.

Rename without reconnecting: open the app's **Edit app name** control and save the display name. Repository identity, connected services and history remain linked. Use the logo controls in the app settings for recognisable supported app branding.

05 | Connect Cloudflare resources

Prerequisites: an app in Suhail, the correct Cloudflare account ID, exact Worker or Pages project name and a suitable read-only token. Connecting source and connecting deployment metadata are separate steps.

1. Open **Settings > Connections & alerts**, select the app, then **Cloudflare resources > Add a Cloudflare resource**.
2. Select **Worker** or **Pages project**. Use the resource name, not its URL, in the name field.
3. In Cloudflare's API token settings, create a token limited to the relevant account/resource where supported. Follow **How to create this token** in Suhail for the current role names.
4. For Workers, the guide identifies **Workers > Content Read-Only**, with older screens showing **Workers Scripts > Read**. Pages uses its own read permission. D1 metadata needs additional **D1 Read**; container observations need **Containers Read**.
5. Enter the account ID, resource name, token and its actual expiry in Suhail. If the token has no expiry, select that explicitly. The recorded date does not change the token's life at Cloudflare.
6. Select **Validate and save Cloudflare**. Open the resource details and use **Check access now** to inspect access and metadata.

Expected result: the selected resource is present, recent metadata is timestamped and unavailable permissions are visible as gaps. A Worker connection can show settings, public addresses, deployment changes and bound D1 recovery metadata. Pages shows available project/deployment information. These observations do not inspect customer rows, KV values or secret values.

Add each Worker or Pages project separately, including a separate payment Worker if it belongs to the app. The source UI supports up to ten Cloudflare resources per app. Source code living in another repository needs its own repository connection and correct product mapping; connecting its Worker does not scan that source.

If validation fails: check Worker versus Pages selection, account ID, exact name, token scope and expiry. A token restricted to your laptop IP will not work for requests sent from the hosted service. Do not remove an intentional network restriction without choosing an appropriate approved access route.

Pause: use **Pause cloud checks** on the selected resource. To replace credentials or resume, use **Replace connection or resume checks**, validate and save. Revoke a compromised token at Cloudflare; pausing a Suhail connection is not provider revocation.

D1 distinction: observing a Time Travel bookmark proves metadata availability only. The accepted 104 drill restored a disposable synthetic database and removed it. It did not restore a customer production database. Recovery procedures are in chapter 19.

06 | Scans, progress, coverage and inventory

Open **Apps**, select the app and its checks view. **Repository scan** shows the current pass; **What was checked** shows the evidence. **Check now** requests work; **Pause checks / Resume checks** controls the app. A saved request continues while the browser is closed.

Choose the checks

1. Review **Scan settings**. Included source checks use the connected revision. Optional package lookups need consent because supported package names and versions go to advisory services.
2. If using workflow observations, approve GitHub **Actions: Read**, then select the actual workflows for that app. Suhail observes their results; it does not run your project tests itself.
3. Save the displayed settings and request a check if an immediate result is needed. Do not repeatedly press the button during a continuing pass.
4. When it finishes, inspect source commit, finish time, checked/skipped counts, individual groups and gap reasons. Open **Scope, skipped files & limits** for details.

Schedules: ordinary automatic repository checks follow a daily cadence subject to capacity. PR automation has separate polling behaviour. Automatic work shares up to 70% of the monthly run allowance; requested capacity remains protected. Use the page's actual next-due and status information rather than treating cadence as a guaranteed delivery time.

Coverage groups are not interchangeable. Basic source checks, isolated deeper analysis, dependency advisories, workflow evidence, Workers/request handling and Swift storage each have their own proof. A passing count in one group cannot fill a gap in another. In particular, an older result with 0 deeper files checked remains that result after a routing fix is installed. Only a fresh scan can establish new deeper-engine execution.

Incomplete or stale: retain the last finished pass as historical evidence. Fix the named access, parser, size, provider or time limit before another attempt where possible. Changed source may need a fresh pass; partial coverage cannot silently resolve old findings.

Dependency inventory

Open the app's inventory section, review collection time and commit, and use its search/licence filters. Export ordinary JSON or CycloneDX where offered. The inventory is bounded to 500 unique package versions, 12 file references and 384 KiB, with omissions visible. Advisory lookup failures can coexist with a collected inventory.

CycloneDX is version 1.6 and marked incomplete. It is not a full deployed bill of materials, dependency graph or licence compliance opinion. npm licence strings are declarations, not independently reviewed licence texts. A development dependency is not automatically low risk, and package presence does not prove reachable production use.

07 | Triage findings and verify fixes

Prerequisites: a completed or partial result you can identify by app, commit and time. Open **Findings**, choose the app and use **Open findings**. Expand **Severity, type & grouping** when useful.

1. Begin with known-exploited dependency matches, then critical/high findings. Review coverage gaps alongside the queue. A missing exploited-catalogue match does not establish safety.
2. Open an issue and read its evidence, affected path/version, rule and priority explanation. Grouping combines presentation, not the identities of individual occurrences.
3. Decide whether the issue is applicable, needs investigation or is a false positive. A private repository does not make its deployed endpoints private.
4. For an applicable issue, repair the actual source, run project tests, commit and push to the connected branch.
5. Use **Verify pushed fixes** for up to 200 selected findings from one app, or allow the next enabled scan. Confirm the verification inspected the commit with your change.
6. Check **Resolved findings** and the resolution evidence. A matching completed check or verified removal can resolve source findings; a queued run, read receipt or merged PR cannot do so alone.

False-positive decisions

Open **Review as a false positive**, explain why the check does not apply and save. Reasons must be specific and must not contain credentials or source excerpts. For a shared reason, select eligible findings on the page and preview the bulk review. Use one reason only if it applies to every selected finding.

To reverse a decision, choose **Reviewed false positives**, open the finding and select **Reopen this finding** with a reason. Changed evidence can reopen a prior decision automatically. This records a review decision; it does not fix source or establish that the app is safe.

External-agent review

1. In **Findings**, choose the app and **Prepare agent review**. Export the selected findings or the app's open findings; include history only when useful.
2. Read the disclosure before sharing. The findings bundle contains private app names and paths. Share it only with your authorised coding agent or reviewer.
3. Copy the supplied instructions and ask the agent to preserve IDs, revisions, commits and review tokens. The agent proposes decisions; it does not approve them for you.
4. Upload the returned proposal file. Review ready and rejected rows. Each proposal starts unselected; explicitly select the decisions you accept and confirm them. The server checks the selected proposals again before applying them.
5. For valid issues, fix the source and use **Verify pushed fixes**. A false-positive review is not a source repair.

Choose the right download: the agent findings bundle is for proposing finding reviews. **Download saved evidence** is for reviewing recorded checks, including saved running-app/security-test results where available; it starts no tests. An evidence archive cannot be uploaded as an agent-review proposal. If a proposal is stale, export current findings and review again.

08 | Repair dependencies and exposed credentials

Dependency workflow

1. Open the dependency finding and compare the recorded installed version with the advisory's affected and fixed-version evidence.
2. Treat a listed fixed version as advisory information, not a compatibility recommendation. If fixed-version information is absent or incomplete, keep that uncertainty.
3. Choose **Preview a fix** where offered. Approve the stated public registry/advisory metadata lookup and review the exact patch.
4. For an unsupported graph or major/minor upgrade, use the project's package manager and tests in your usual development environment, then push the actual fix and rescan.

The built-in deterministic scope includes bounded npm v3 lockfile repairs for supported direct dependencies and a unique root-hoisted transitive leaf. Candidates stay within the same major/minor patch line.

Ambiguous paths, duplicates, incompatible constraints, hooks, private sources and child-dependency changes require manual work. A transitive repair changes the lockfile, not an unrelated root declaration.

Simple exact public PyPI pins and selected TLS/configuration fixes are also supported. A requirement edit is not a fully resolved transitive environment. No package installation or customer test suite executes inside the preview.

Possible credential workflow

1. Confirm privately whether the detected pattern represents a real credential and which service uses it. Provider-specific guidance is inferred from the rule; Suhail has not tested the credential with its issuer.
2. In the issuer's own controls, create and configure a replacement, or retire the use. Keep replacement values in an appropriate server-side secret store, never an iOS or browser bundle.
3. Revoke the old credential at GitHub, Stripe, Cloudflare or the relevant issuer. If exposure is active, prioritise containment over preserving uninterrupted use.
4. Remove the old value from source, commit and push. Review history/exposure and connected systems as appropriate; deleting the current line does not revoke the credential or erase old copies.
5. Record the four steps in the finding's remediation checklist and save. Request source verification when ready.

Success: provider revocation/replacement is confirmed by you, and the matching source scan confirms removal where covered. The saved checklist is your attestation and never resolves the finding by itself. A source scan cannot prove revocation at the issuer.

If the checklist conflicts: refresh the current finding. Its saved state is tied to the exact evidence; a new commit or concurrent edit can make the old form stale. Do not copy prior completion onto different evidence without reviewing it.

09 | AutoFix and Claude-assisted repairs

Prerequisites: an eligible open finding at the exact connected branch-head commit, authority over the repository and sufficient allowance/capacity. A generated patch is a proposal.

Deterministic repair

Open the finding, choose **Preview a fix**, and review every changed line and preparation check. Download the patch if you want to apply it locally. For the hosted route, approve the displayed patch to create a branch and draft pull request. The installation needs optional **Contents: Read and write** and **Pull requests: Read and write**. Ordinary scans continue using read-only access.

The operator requests the optional GitHub App permissions; each installation owner approves them in GitHub. Do not substitute a broad write token. Creating a branch/PR can trigger that repository's own workflows, so review the repository's CI policy before approval.

Claude source repair

1. On a supported JavaScript/TypeScript request-source finding, prepare the context. Preparation retrieves the exact file but makes no model call.
2. Review the complete file and provider disclosure. The eligible context is one file, up to 16,000 UTF-8 bytes and 300 lines. Known secret indicators can block preparation, but your review is still required.
3. Explicitly approve sharing that context with Anthropic and request generation. The source pins a Claude model; the exact provider/model shown in the consent is authoritative for that attempt.
4. Inspect the returned patch. Up to three bounded replacements and syntax/consistency checks do not prove behaviour or security correctness.
5. Download it or separately approve the draft-PR action. Run project CI, review and merge as a human. Then verify the actual updated branch through a fresh scan.

The supported AI path covers seven selected request-source rules, not arbitrary findings, complete repositories or autonomous agent work. No model tools, repository commands, package installs or automatic retries execute. Source contexts expire for use after 15 minutes but remain encrypted in their saved row until replaced; that expiry is not deletion.

Usage and availability: the service applies generation limits and may temporarily refuse another attempt. An uncertain attempt can retain its reserved allowance. Read the error before trying again; a failed or interrupted response is not permission to generate repeatedly.

If stopped or uncertain: refresh the saved operation before another request. A changed branch needs a new scan/context. A branch may exist after a later provider failure; inspect its recorded identity before retrying. Never force-push over an externally changed branch to resume a proposal.

Success: reviewed change, actual project CI, human merge and matching post-merge evidence. No appropriate real finding is required to be invented merely to demonstrate AI.

10 | Review pull requests

Prerequisites: enabled GitHub App connection, PR read permission and a supported open PR targeting the connected branch. Fork PRs are outside the current workflow.

Manual review

1. Open the app, then **Checks > Pull requests**, select the app/repository and the intended PR.
2. Request its source review. Read the saved report's head, base, merge-base, timestamp and coverage.
3. Compare baseline signals with new/changed signals. These are source-review observations, not confirmed exploitable vulnerabilities.
4. Use **Check latest PR status** before relying on an older snapshot. That refresh reads metadata; it does not rerun the source checks.

The review covers selected credential, JavaScript/TypeScript and Swift storage rules. It does not run the deeper scanner container, advisory lookups, project tests or Claude. Bounds include 20 changed files, 128 KiB per file, 1 MiB total, 65 provider requests, 35 seconds and 100 signals. Oversize, binary, symlink, unsupported or unavailable files remain gaps.

Automatic reviews

Enable **Automatic reviews** once per repository. The scheduler polls eligible metadata about every 15 minutes, with backlog delays possible. It considers the 20 most recently updated open PRs targeting the selected branch, skipping drafts and forks. Truncation is disclosed; older PRs may need a manual review.

Refresh status to inspect last lookup, queue, next eligible check and saved result. Normal browser sign-out does not stop deliberately enabled automation; use its pause control. Changes in access or security context may require re-enabling. Complete reviews use the existing automatic allowance, with a six-review-per-workspace/hour limit. Incomplete reports remain visible without unlimited automatic retries.

Notifications and sharing

New automatic results containing signals or coverage gaps appear under **Notifications > PR reviews**. Open the exact saved review. Reading acknowledges that version only; a later result becomes unread again. This feature has no historical backfill or PR email/Telegram delivery.

To share one snapshot, enter up to two named existing Suhail sign-in emails in the report's sharing controls. Save, copy the link and send it yourself. Suhail sends no invitation email for this workflow. Recipients get read access to that report, not permission to scan, repair or administer the workspace. Remove a named reader to revoke subsequent reads; a previously viewed or saved copy cannot be recalled.

Success: the saved result matches the intended current commits and the recipient sees only the permitted report. A PR report does not clear main-branch findings.

11 | Publish a GitHub check and set merge policy

Prerequisites: a working automatic review on a supported PR. Publishing the result and requiring it for merge are two separate owner decisions.

1. In **Pull requests > Bring the review to GitHub**, open **One-time GitHub setup & merge rules**.
2. The GitHub App owner requests repository **Checks: Read and write**. The installation owner approves that change in GitHub. No branch-administration permission is needed.
3. Return to Suhail, choose the threshold, tick the summary-publication consent and select **Enable GitHub check**.
4. Let the scheduler publish a saved automatic result. **Refresh check status** reads state; **View GitHub check** opens the exact result.
5. Confirm head/base, conclusion and the signed-in saved-report link before considering branch rules.

Policy	What a passing result establishes
New high or critical signals	Complete selected coverage and no new/changed high or critical signal.
All new or changed signals	Complete selected coverage and no new/changed signal at any severity.
Action required	Gaps, uncertain/stale comparison or another state requiring attention; not a pass.

Existing baseline signals remain visible but do not alone fail the new-signal threshold. The GitHub summary publishes counts, commits and a report link, not source text, secret values or teammate addresses. Publishing a saved result consumes no extra review run.

Optional required status check

After a real result is verified, a repository administrator can configure **Suhail PR review** in GitHub's branch rules, select the Suhail App as expected source and require the branch to be up to date. Keep normal CI and human review. Suhail does not change these rules for you.

Do not require the check on uncovered branches, fork flows or merge queues. Unsupported PRs can otherwise wait for a check that never arrives. Provider permissions and available branch controls depend on the repository configuration.

Disable safely: Disable GitHub check stops future publication. Existing checks remain on GitHub. If it is a required check, an administrator must separately remove or change that requirement. Pausing reviews is not removal of the branch rule.

If GitHub is unavailable: an older result may remain visible. Inspect Suhail's publication error and the exact commit. A lost create acknowledgement is reconciled using the saved identity; do not reset or recreate checks blindly. Confirm a real GitHub result before relying on it.

12 | Team review access

Prerequisites: an owner workspace and an existing enabled, verified Suhail account for the recipient. Team access is workspace-wide, covering current and future apps. It is not per-app delegation.

1. Open **Settings > Team & invitations > Manage your team > People**.
2. Enter the exact Dataloup sign-in email used by the recipient's Suhail account. Choose the least-privileged role that fits the work.
3. Create the invitation, copy the workspace link and send it yourself. This release does not send an invitation email.
4. The recipient signs in as that identity and accepts from their own **Team** page. Pending invitations expire after seven days.
5. Have them select the shared workspace and verify the intended saved views. Their personal workspace remains separate.

Role	Allowed work
Owner	All workspace roles and operational controls.
Admin	Manage members/read-only viewers and review findings; cannot grant admin or alter owner/admin authority.
Member	Review permitted findings and saved app/PR evidence.
Read-only	Read permitted saved evidence; no finding decisions.

Scans, connections, automation, source-sharing consent, repairs and billing remain owner controls. Team readers do not automatically gain provider permissions or full runtime-test setup/report management.

Change or remove access: use People to change the member's role or revoke membership. The server checks live authority on requests; open result views recheck while visible and on tab return. Removal retains review and membership history. Existing downloaded evidence is outside that revocation mechanism.

Success: the person can see the intended workspace and can perform only their role's actions. A hosted acceptance check should also show that revocation prevents a subsequent read and clears the visible shared view.

If an invitation fails: check exact email, verified/enabled account, expiry and the inviter's current role. An invitation created under authority that was later revoked may no longer be valid. Do not create a second identity just to sidestep an account-link issue.

Check your plan and workspace access before adding teammates. The two named readers per PR snapshot are a separate sharing limit.

13 | Check a running application

Prerequisites: authority over the target, a connected app and one final public HTTPS origin. Use a test workspace with synthetic records for private checks. Review that chosen GET endpoints have no side effects.

Public setup

1. Open **Running app** at <https://suhail.dataloup.io/application-checks> and choose the app.
2. Enter the final origin and a direct public path, such as /. Avoid redirects, custom ports, IP literals, credentials and query strings.
3. Confirm control and choose **Save public setup**. Leave daily checks off until the first result is understood.
4. Download the displayed verification file and publish it at **/.well-known/suhail-verification.txt** on that exact origin. It must return 200, text/plain and the exact content. Prefer no-store caching; do not route it through authentication or a SPA fallback.
5. Select **Verify domain**, then **Run selected checks**. Refresh for the queued/running/saved state.
6. Read failed and incomplete checks first; export the result JSON if needed.

Public checks observe HTTPS reachability, selected response headers and Secure on observed cookies. They do not enumerate TLS ciphers, prove CSP correctness or execute a browser. A public-only result leaves sign-in and tenant/role coverage untested.

Add dedicated-account checks

Use the guided account and route setup on **Apps > your app > Checks > Running app**:

1. Add one to three dedicated test accounts with recognisable labels. Enter a short-lived Bearer session or a named cookie using the matching authentication choice.
2. Set each account's read-only identity endpoint and expected user, tenant and role fields. Choose the appropriate value types; the actual response must match.
3. Add one to three read-only private routes. Identify the account allowed to read each route, the JSON pointer to a unique synthetic marker, and the accounts that must be denied, including anonymous access.
4. Preview the setup. The preview omits credentials; inspect the origin, accounts, routes and expected boundaries. Editing inputs invalidates that preview.
5. Confirm authority, dedicated identities and the read-only scope, then save. The guided form retains inputs when validation fails so you can correct the named field.

A role-boundary test needs different roles in the same tenant. A tenant-boundary test needs identities in different tenants. Create these identities and synthetic records in your own application first; Suhail does not create them.

Advanced setup: JSON sign-in and explicit record-ID substitution use the downloadable templates. They are not part of the simplified Bearer/cookie form. Keep completed templates private.

Advanced JSON sign-in must return one named cookie with Path /; its flat request body supports up to eight string fields. Browser/SSO/MFA/CSRF handshakes and token-returning login JSON are outside scope. Imported sessions do not test the original login. Suhail does not log out of the target automatically.

Each private route needs an authorised actor, expected JSON pointer and unique synthetic marker, plus explicit denied identities including anonymous. Positive controls must return HTTP 200 JSON with the expected marker. A valid denial is 401, 403 or 404 without it. Redirects, expired identities and ambiguous responses remain incomplete.

Saving replaces the previous setup. Keep completed templates private; sample routes are not created by Suhail.

14 | Running-app results, stopping and retests

Selected checks passed means the selected observable criteria passed. **Results need attention** means a selected rule failed. **Some checks are incomplete** means the evidence was not established. None of these expands coverage to unselected routes or every tenant.

1. Open the saved result and confirm origin, scope and time.
2. For a failure, inspect the named route and identity labels. A synthetic private marker returned to a denied identity creates a finding even in an error response.
3. Fix the actual access-control or configuration issue in the target, deploy it and rerun the same scope.
4. Confirm the matching passing check resolves that finding. A source scan, changed scope or unavailable control cannot clear it.

Daily use: enable the separate daily option only after setup is verified. Its first due time is about 24 hours after saving; request the initial run manually. Daily work uses the automatic allowance and existing scheduler. Queueing is not completion.

Pause / stop: invalidates active scope and prevents subsequent requests once observed. A request already in flight may finish. Resume does not immediately replay an interrupted login. An expired lease becomes interrupted and waits for a new request or next eligible daily run.

Remove setup and stored test credentials: stops scheduling and removes the encrypted profile/proof from active setup. Saved reports and findings remain. Revoke target-issued sessions in the target application if needed. Removing the public proof alone is less direct than using Stop because caches may delay observation.

Limit	Running app
Target	One verified public HTTPS origin
Identities / routes	Up to three of each
Requests / duration	48 requests and 25 seconds per run
Response	Four seconds and 64 KiB per target response
Controls	DNS, proof, sign-in and identity checks count in the budget
Retention	Newest 20 reports per app; older finding records can remain

If incomplete: reduce the selected scope, renew dedicated sessions, correct identity/marker pointers, remove redirects or resolve provider/network failure. Do not weaken the target's real access controls to make a test pass. A positive control must exercise the application's real data/authorisation path, not a test-only bypass that always returns the expected answer.

The setup owner manages full configuration and reports. Team review retains its permitted finding evidence. Run your own authorised scope before relying on it for your app.

15 | Run scoped security tests

Prerequisites: verified, enabled **Running app** setup with dedicated private test identities and routes. Public-only setup is insufficient. These tests are bounded HTTP probes, not a comprehensive penetration test.

1. Open the app, then **Checks > Access tests** (the page title is **Security tests**) at <https://suhail.dataloup.io/security-tests> and choose the app.
2. Select the existing private routes and probes. Changed-session rejection changes one token/cookie byte. Cookie CORS checks use two untrusted Origins. Cache checks inspect selected private-response caching behaviour.
3. Review limits: defaults are 64 requests, 45 seconds and at least 500 ms between request starts. Controls count; reduce selections if their estimate does not fit.
4. Confirm ownership, dedicated identities, synthetic data and GET routes without side effects, then save scope. Saving starts no target request.
5. Review **Allowed tests**, authorise this particular run and choose **Run selected tests**.
6. Refresh status and read the saved scope/run receipt. A queued authorisation expires after one hour without execution if it cannot start. A requested run has a five-minute cooldown.

CORS probes require cookie-based authentication or the supported JSON-login cookie. A Bearer session alone cannot establish a browser-cookie exposure. Cache observations distinguish explicit shared-cache permission, private/no-store protection and incomplete policy; a signed-out read does not by itself prove a CDN caused a leak.

Explicit record-ID substitution

Download the account-aware scope template. Supply two different configured actors, two known synthetic private record IDs, their distinct expected markers and the single permitted query parameter on the read endpoint. Both owners' positive controls must succeed before and after substitution. There is no ID enumeration.

At most two record pairs and three existing routes fit within the shared budget. IDs allow bounded ASCII letters, digits, underscores and hyphens, up to 80 characters. Use only records intended to be private to their owners; intentionally shared records are inappropriate denial fixtures.

Importing creates no accounts, records or target routes. Keep the original private scope file if you need to change it. The simple route-selection form replaces previously imported object tests; review that warning before saving.

Success: a saved report for the exact approved scope, with positive controls, selected outcomes and explicit gaps. Complete runs consume one existing app-behaviour run allowance; incomplete runs release their reservation. Access tests require per-run authorisation; they are not scheduled daily. Running-app checks and Access tests for the same app cannot overlap.

16 | Security-test evidence and retesting

Read the scope fingerprint, configuration revisions, approval/run timestamps and safe request trace. Traces contain method, path, query names, account labels, purpose and status/reason, but omit secret headers, query values, expected identity/record values and response bodies. Keep private information out of labels and paths anyway.

1. Open the finding's saved report and identify the exact failed case.
2. Correct and deploy the target behaviour through your normal development process.
3. Request a linked retest with the same scope. Review its fresh controls and outcomes.
4. Only a passing matching case resolves that finding. Changed scope, missing evidence or an interrupted run leaves prior findings open.

Use **Stop / pause** to invalidate approval and block subsequent requests once observed. An in-flight request may finish. **Resume scope** permits a newly authorised run; it never replays the old run automatically. Changes to the underlying Running app setup invalidate 112 approval, so review and save the scope again before authorising.

Remove this test scope stops Access tests work and deletes its encrypted record-ID configuration. It retains reports/findings. The reused test credentials belong to Running app and must be removed there separately. Revoke issued sessions at the target where needed.

Limit	Access tests
Target	One verified public HTTPS origin
Request budget	Maximum 64, including proof/DNS/identity controls
Time budget	Maximum 45 seconds; four seconds per response
Response / pacing	64 KiB per response; minimum spacing selectable from 500 to 2,000 ms
Scope	Three accounts, three routes, two explicit record pairs
History	Newest 20 reports; latest 30 displayed scope/run audit actions
Audit retention	Existing 90-day rule

The maximum request count may not fit the time budget if responses are slow. Limits are per app/run, not a shared hostname-wide limiter across multiple configured apps. A timeout is incomplete evidence, never a pass.

There is no crawling, arbitrary script execution, injection suite, brute force, exploit chain, shell, write/payment/cancellation test or denial-of-service test. No compliance certification is produced. Keep the scope narrow and explain it when sharing a report.

17 | Notifications, monitoring and extended reviews

Email and Telegram

1. Open **Settings > Connections & alerts > Notifications**. Review the account email and automatic-alert preference; save intentional changes.
2. Use **Send test email** and confirm receipt in your inbox. Provider acceptance alone is not proof of receipt.
3. For Telegram, select **Connect Telegram**, open the private link and press Start in Telegram.
4. Return, select **Refresh Telegram**, verify the displayed identity and **Confirm this chat**. Never confirm an unfamiliar chat.
5. Use **Send Telegram test**; if needed, use **Test both channels** and confirm both receipts.

When enabled, routine monitoring problems go to Telegram after connection; important incidents also use email. Delivery is normally handled on the separate five-minute monitoring run. Tests can succeed while automatic alerts are off, so check both configuration and receipt. Uncertain sends are not blindly repeated. Disconnecting Telegram restores the workspace's existing email policy.

Open **Monitoring** for incidents, recent background checks, freshness, delivery status and evidence exports. A queued or running check is not recovery; a fresh successful result is needed. Both hosted workers still depend on Cloudflare, so an independent external monitor matters.

Security review

Open **Security review**, choose the app and select recent history, GitHub security alerts, GitHub posture and/or Cloudflare account posture. Optional selected public assets use the workers.dev/pages.dev address of an enabled connection, up to five paths and 256 KiB per asset. This older asset sampler does not cover custom domains or crawl the site.

Confirm the read-only scope, **Save review choices**, then **Collect review**. Review each section and **Download review JSON**. Collection can take up to three minutes; leave five minutes between requests. History is a sample of up to five commits and five eligible changed files per commit.

Use **Access needed** only for a selected unavailable check. GitHub alerts, repository administration and organisation membership have separate read scopes. Cloudflare account posture needs Account Settings read. A provider may not expose personal MFA or effective bypass details; unknown remains unknown. These reviews do not change provider settings, clear source findings or automatically send new notifications.

Other specialist views: Pack Money tests apply only to their dedicated supported fixtures and configured test integration. Use their displayed setup and saved-run instructions; they are not generic app coverage. Releases & checks and imported local projects are historical records until explicitly matched. Profiles describe capabilities; the latest app result describes actual coverage.

18 | Plans, usage, cancellation and data

The following monthly USD plan allowances describe the current Suhail catalogue. Use the availability and terms shown in your Dataloup account for your actual purchase. Do not infer active access from a price card alone.

Suhail plan	USD/month	Products	Repositories	Runs	Linux minutes
Protect	29	1	3	150	250
Multi-product	59	3	9	450	750
Studio	129	10	30	1,500	2,500

All capacities are workspace pools, not new allowances for each product. Automatic checks can use up to 70% of monthly runs: 105, 315 and 1,050 respectively. Requested capacity is preserved. Linux time is measured separately. A pending reservation, incomplete attempt and completed billed run are different usage states.

Together Pup/Pack/Studio remain US\$49/89/189 for one/three/ten matching product identities, including the corresponding Suhail and NosyPup pools. They do not multiply capacity or silently replace an unavailable plan. NosyPup remains independently live.

Manage an existing subscription

1. Open the shared Dataloup **Plans & billing** page from your account.
2. Confirm the product family, workspace and current plan before changing anything. Resume an existing saved purchase instead of creating duplicates.
3. For a tier change, read its preview, new monthly amount and effective renewal date. Approved terms specify no immediate charge or proration.
4. For cancellation, cancel renewal and check the recorded paid-period end. Paid access continues to that date.
5. Return to Suhail and review **Product & usage** after the authoritative billing update. If access is unavailable or unverified, refresh and contact support; do not buy again to repair a missing receipt.

If you cannot start or resume setup from your account, use Contact us before making another purchase. Your account should identify the intended workspace, plan and order.

Data: downgrades retain saved data and restrict over-limit activity; there are no automatic overage charges. Cancelling renewal, revoking integration access and deleting data are separate actions. Evidence exports are not a full personal-data export or restorable backup. For access/correction/deletion, contact Info@dataloup.io with the workspace and scope. Ask support about the scope and timing of a deletion request.

19 | Troubleshooting by symptom

Symptom	Next action and evidence of recovery
Wrong or empty workspace	Confirm production domain and selected identity/workspace. Use the existing account mapping; do not recreate data. Recovery is the original apps/history visible.
Reconnect or expired access	Check selected repository/resource, token expiry and required scope. Reconnect the same app and verify a fresh successful check.
Old commit after a fix	Confirm push to the connected branch; inspect the latest finished pass rather than only the newest attempt.
Zero deeper files	Read engine/gateway/coverage reasons. A fresh completed deeper result is needed to establish execution.
Partial scan	Open skipped-file/limit details. Fix the concrete cause; retain unresolved findings for uncovered files.
Fix preview unavailable	Check eligibility, exact commit and supported dependency graph. Use a manual repair for unsupported cases.
AI unavailable or capped	Check provider state, context expiry and workspace/global cap. A new approval is needed after uncertain generation; do not bypass spend controls.
PR result looks stale	Check latest metadata and head/base; request/await a matching review. A green older commit is not current proof.
Required GitHub check missing	Check supported branch/PR type, permission, opt-in, queue and publication error. Do not weaken branch policy just to hide the error.
Team invite not received	Team invitations send no email in this release. Send the copied link and have the exact existing identity accept in Team.
Domain proof fails	Check exact HTTPS origin, 200 plain text, exact content, no redirect/auth/SPA fallback and caching.
Runtime control incomplete	Renew test sessions and verify identity pointers, expected marker and actual GET response. Reduce scope if budgets expire.
Access tests will not start	Check verified private Running app setup, saved scope, per-run consent, cooldown, queue expiry and conflicting app run.
Alert test succeeds but incidents don't arrive	Check automatic-alert preference and routing; test delivery does not enable automatic alerts.
Export too large	Select one app or use supported CSV. Keep data private and preserve the recorded scope.
Billing unavailable/unverified	Resume the existing order, inspect its state and contact support. Do not buy a second subscription to repair access.

For any uncertain save or provider operation, refresh and inspect its saved state before repeating it. Do not clear journals, widen access, reset a workspace or mark a finding false positive merely to remove an error.

20 | Get help and give feedback

Contact us: open Help & history > Contact us, or <https://suhail.dataloup.io/contact.html>. You can use this page without signing in, including when an account problem prevents access. Select Contact us / ask for help, enter your email and describe the issue.

Feedback: use Help & history > Send feedback, or select Share feedback on the same form. Tell us what you were trying to do, where you got stuck and what outcome would help. You do not need to diagnose the technical cause.

What to include

1. Use your account email if the request concerns a workspace. Include the app's display name only if it is safe to share.
2. Identify the screen, approximate time and timezone, action taken and the exact displayed error reference.
3. Say what you expected and what actually happened. For stale results, include the non-secret commit ID or report ID when relevant.
4. Keep passwords, API keys, session cookies, reset links, recovery codes, private source and customer records out of the message. The form has no attachments.
5. Send the message and retain the displayed reference. A saved/queued confirmation means submission succeeded; it does not claim a human has read it or that email delivery has completed.

If a connection fails or the form cannot confirm submission, your text remains in the form. Retry without editing after a transient failure to reuse its submission identity. If the form reports too many attempts, wait or email Info@dataloup.io. If an error persists, send the reference to that address. No response-time guarantee is implied.

Keep your setup healthy

Return after a meaningful code or deployment change. Confirm the recorded source commit, last successful checks, gaps, credential expiry and notification destination. Review a proposed fix before merging it and inspect a fresh result afterwards. Remove unused connections and revoke credentials at their issuer when they are no longer needed.

This guide describes the client controls in edition 116, including the guided workflows introduced in 115. Visible actions depend on your access, app scope and service configuration. The online guide and your own recorded result are the best places to check the applicable workflow.