Installation Guide

This document takes a Salesforce org from "nothing" to "OrgGuardian installed, access granted, and its first scan complete". Read §1 first: it says how a version reaches your org and where to install it first.

Logical next step: Permissions and Personas guide, then the Configuration Guide.

---

1. Distribution — versions and installation links

The version installed in your org can be read in Setup → Installed Packages, in the "Version Number" column. Through the user interface you never enter a package version ID: the **installation link** you receive — from the publisher, or through the AppExchange listing — already carries the target version (§3).

Install in a sandbox first. Validate the package in a sandbox of your org before any production installation, with your own edition, settings and users (§2.1).

An in-place upgrade, with history preserved, exists only from a released version of the package. From any other version, installing a later version first requires uninstalling the installed one, which deletes the package's data (§7).

> ⚠ **Uninstalling a managed package destroys its custom objects and all their data**: your findings, your attestations and all your trend history disappear with it. Read the Uninstall Guide **before** you need it, not after.

---

2. Prerequisites

2.1 Salesforce editions

OrgGuardian is supported on Enterprise, Unlimited, Performance and Developer Editions. Professional and Group Editions are not supported, for two reasons that lie in the product itself:

- the trend history is written to **7 Big Objects** (§4.3), and the product has no fallback storage;
- the `OS_SelfOrg` self-callout queries your own org's API, and access to it is not included as standard in these editions. Nine features depend on it: Health Check parity, Event Monitoring, the limits supplement, silent failure detection, the storage breakdown, the certificate probe, the org hardening audit, LWC and Aura linting and the Agentforce agent posture (the Configuration Guide §3, “Wiring the `OS_SelfOrg` self-callout”).

The package does not read your org's edition: nothing blocks an installation on an unsupported edition; the support matrix excludes it.

The package requires no optional platform feature. It declares no dependency on a feature such as multi-currency or Event Monitoring, nor on an add-on licence, and the product never checks which edition your org runs. A setup source that does not exist in your org is skipped without a finding; a read that your org refuses is skipped too, without failing the scan. For the main setup sources, `AsyncApexJob` included, the refusal is reported as a finding; some other reads degrade without one (the Glossary §7, "Graceful degradation"; the Troubleshooting Guide §4).

Validate in a sandbox first. As with any managed package, install OrgGuardian in a sandbox of your org and check it there — with your edition, your settings and your users — before installing it in production (§1). The technical constraints of §2.2 apply whatever the edition.

2.2 Technical constraints

Constraint	Value
Source API version	**67.0**
Namespace	`orgguardian`
User interface	**Lightning Experience only**
Form factor	**Desktop only** — no mobile app and no notification on mobile either (below)
Package type	**Managed 2GP**

An org that has not enabled Lightning Experience will therefore not see the app. A mobile user will not see it either: this is by design. Nor will its **notifications** reach a phone: the shipped notification type `OrgGuardian_Alert` is declared for the desktop channel only (Setup → **Custom Notifications**), so finding alerts and remediation reminders ring only in the Lightning Experience bell (the Operations Guide §5); no

outbound channel is open at installation. This guide does not cover whether an org administrator can turn the mobile channel back on in Notification Delivery Settings for a type shipped by a managed package.

> **The same requirement applies to each user, not only to the org.** A user with the *Minimum Access — Salesforce* profile who is assigned only the `OrgGuardian_Viewer` set **cannot open the app**. They get the platform page "You can't access Lightning Experience — Lightning Experience is enabled for your organization, but not yet for you." This is expected: the *Viewer*, *Analyst* and *Auditor* sets **deliberately grant no user privilege**, because which of your users may use Lightning Experience is your decision, not the package's.
>
> **What the administrator must do**: grant these users the platform privilege **"Lightning Experience User"** (profile, or an in-house permission set). On most standard profiles it is already present; it is not on *Minimum Access*.
>
> Check it before opening a ticket: in **Setup → Users**, open the user — **Lightning Experience User** must be enabled in the system permissions of their profile or of one of their assigned permission sets. Or query it, with the user's Id in place of `005...` (at least one row means the privilege is held):
>
> ```
> SELECT PermissionSet.Label, PermissionSet.Profile.Name, PermissionSet.IsOwnedByProfile
> FROM PermissionSetAssignment
> WHERE AssigneeId = '005...'
>   AND PermissionSet.PermissionsLightningExperienceUser = true
> ```

2.3 What the user who installs and configures needs

- **Installing**: your org's package installation rights.
- **Configuring afterwards**: twenty of the 23 panes of the **Settings** screen are editors that write Custom Metadata records through the Apex Metadata API, and the platform requires the user who saves to hold the permission **"Modify Metadata Through Metadata API Functions"**, which System Administrators have (**"Customize Application"** on its own is not enough); the **Heroku option** and **Secrets and consents** panes, and the feature mute (the **Muting** section of the **Subscription** tab), write protected custom settings, which need **"Customize Application"**. Two assistants also switch features on through the same Custom Metadata deployment as the editors, so they need "Modify Metadata Through Metadata API Functions" as well: on a paid plan (or during the plan's trial), the activation assistant of the **Heroku option** pane switches on `Heroku_Tier`, and the **Tooling API access** pane switches on, one at a time, the features that depend on the `OS_SelfOrg` self-callout. OrgGuardian does not bypass these checks and does not grant either privilege to itself: the platform decides, never the package.
- **The custom permission does not replace that privilege.** The shipped custom permission **`OrgGuardian_Manage_Features`**, carried by the **`OrgGuardian_Admin`** permission set, gets a user past OrgGuardian's own check before a save, not past the platform's, and `OrgGuardian_Admin` grants neither privilege. On its own, it therefore does not let a user save Custom Metadata records or protected settings from these editors (the Configuration Guide §2, "Who is allowed to save").

2.4 What is optional and is NOT a prerequisite

- **Shield / Event Monitoring**: not required. When the Event Monitoring licence is present, an optional feature enriches the screens — its feature switch is **OFF by default**, and it goes through the `OS_SelfOrg` self-callout (the Configuration Guide §3, “Wiring the `OS_SelfOrg` self-callout”). The actual coverage without Event Monitoring is described in the Operations Guide §8, which names the detectors that run without it and those it adds; what the product does regarding encryption is stated in the same section, "With or without Shield".
- **Heroku option** (external collector): disabled by default, fail-closed, and it requires **you** to declare a remote site (the package ships none). You request the activation from the product, and the publisher confirms it.
- **Named Credentials** (Slack, Teams, multi-org, connectivity probes): to be created by you, case by case, only for the features you enable.

---

3. Installing

3.1 From the command line

```
sf package install --package <version-id> --target-org <your-alias> \
  --security-type AdminsOnly --wait 30 --no-prompt
```

- `--package` accepts the **version ID** supplied by the publisher (it starts with `04t`), or the local alias that you give it in your own `sfdx-project.json`. You do not have to build it: it is provided to you with the installation link.
- `--security-type AdminsOnly` grants access to administrators only — the equivalent of "Install for Admins Only" (§3.2). It is also the command's default.
- `--wait 30` waits up to 30 minutes for the installation to finish; without it, the command returns before the end, and `sf package install report` gives the status of the request.
- `--no-prompt` skips the confirmation prompt. That prompt concerns remote site and CSP trusted site settings, and the package ships neither; read what the package installs (§4) before you use it all the same.

3.2 From the user interface

The installation URL of a package version is the platform's standard page, `/packaging/installPackage.apexp?p0=<version-id>` on the target login domain (`https://test.salesforce.com/…` for a sandbox). **This link is handed to you already filled in** — by the publisher, or through the AppExchange listing: you have no ID to enter.

Choosing who is granted access — the page asks to whom access should be granted. Choose **"Install for Admins Only"**. OrgGuardian distributes its access through **permission sets** (§5.1), not through profiles: granting access to all users at installation brings nothing and needlessly widens the surface. This option gives the **System Administrator** profile the "OrgGuardian: manage features" permission and visibility of the administration tabs: the administrator who installs can therefore open Enablement and Settings even before assigning themselves a permission set. Do not use "Install for All Users".

---

4. What the package writes in your org

4.1 At installation

The post-install script **does not bootstrap the product — and this is deliberate**. It writes **a single installation trace** and nothing else: one row in the List custom setting `OrgGuardian_Install__c` (fields `Version__c`, `Installed_At__c`, `Is_Upgrade__c`, `Previous_Version__c`), plus, on a best-effort basis, a `System_Log__c` log row.

The reason: a post-install script runs as a package system user **with neither a profile nor a permission set**. Under that identity, the bootstrap would produce **no scheduled job**, **no** `Scan_Run__c`, **no** `Finding__c`, **no** `Org_Metric_Current__c` — and **no** `System_Log__c` to report the failure. The product therefore leaves the bootstrap to the first authorised user.

Practical consequence: after `sf package install`, the org does nothing yet. The actual bootstrap takes place the **first time a real, authorised user opens a screen** (§5.2), or through an explicit call (§5.3).

4.2 During operation — the "metadata" notice

When an administrator saves a configuration from a no-code editor, the package writes records of **its own Custom Metadata Types in the `orgguardian` namespace**, through the platform's Apex Metadata API — never through DML, never outside those screens.

The exact scope is 26 types. Twenty-five are written from the **Settings** screen; the fleet policies (`Fleet_Policy__mdt`) are written by the fleet policy editor of the **Fleet** screen. The editors of the Settings screen, with the type or types each one writes, are listed in the Configuration Guide §2. The publisher's **Metadata API notice**, which lists the 26, is available on request.

Of the **37 shipped Custom Metadata Types** (all listed in the Glossary §5), the **11** that are not on that list have no editor in the product, and the package never writes to them.

What the package NEVER touches: no standard metadata, no type created by you or by another package, no business data, no code, no permission.

4.3 What the package writes in normal operation

Its own objects — and, only after an administrator's action, a Salesforce Files document (when they import a compliance mapping of their own) and Chatter posts (once the Chatter alert channel is enabled). The exhaustive list, scheduled jobs included, is in the Security FAQ §2. The main objects, with their shipped labels (in English; the package ships no translation of them):

Object	Shipped label	Content
`Finding__c`	Finding	A detected problem, with its severity, its category and its target
`Finding_Activity__c`	Finding Activity	The handling trail of a finding
`Org_Metric_Current__c`	Current Org Metric	The last known value of a measurement
`Scan_Run__c`	Scan Run	The marker of each scan

Object	Shipped label	Content
`Insight__c`	Insight	A synthesis derived from findings and metrics
`Incident__c`	Incident	A grouping of correlated findings
`Attestation__c`	Attestation	A compliance sign-off by an auditor
`System_Log__c`	System Log	The package's internal log
`Org_Health_Alert__e`	Org Health Alert	The platform event emitted with each alert

Trend history goes into the **7 Big Objects** — `Org_Limit_Snapshot__b`, `Config_Snapshot__b`, `Compliance_Snapshot__b`, `Em_Metric_Snapshot__b`, `Endpoint_Health_Snapshot__b`, `Login_Baseline__b`, `Prediction_Outcome__b`. They matter again at uninstall time: see the Uninstall Guide §1.3.

A retention purge — the scheduled job `OrgGuardian Data Retention` — removes rows that are too old according to configurable windows; it targets `Finding__c`, `Incident__c`, `Org_Metric_Current__c`, `System_Log__c`, `Finding_Activity__c`, `Scan_Run__c`, `Callout_Log__c`, `Insight__c`, `Posture_Report__c`, `Config_Dependency__c`, `Fleet_Finding__c` and the 7 Big Objects.

---

5. After installation

5.1 Assigning access

The package **assigns itself no permission set**. Until you assign something, nobody sees anything.

Five permission sets are shipped:

Permission set	Shipped label
`OrgGuardian_Viewer`	OrgGuardian: read-only access
`OrgGuardian_Analyst`	OrgGuardian: finding handling
`OrgGuardian_Auditor`	OrgGuardian: audit and compliance
`OrgGuardian_Admin`	OrgGuardian: technical administration
`OrgGuardian_Financials`	OrgGuardian: financial amounts

Four **permission set groups** combine them into ready-to-use personas:

Group (persona)	Shipped label	Shipped composition
`OrgGuardian_Persona_AdminSF`	OrgGuardian: Salesforce admin persona	Viewer + Analyst + Admin
`OrgGuardian_Persona_RSSI`	OrgGuardian: CISO persona	Viewer + Auditor + Financials
`OrgGuardian_Persona_DSI`	OrgGuardian: IT department persona	Viewer + Financials
`OrgGuardian_Persona_Direction`	OrgGuardian: executive leadership persona	Viewer + Financials

Three API names keep French terms: `RSSI` stands for CISO, `DSI` for IT department and `Direction` for executive leadership. The labels shown in Setup are in English.

Procedure: Setup → Users → **Permission Set Groups** → **OrgGuardian: Salesforce admin persona** (`OrgGuardian_Persona_AdminSF`) → **Manage Assignments** → add yourself.

The detail of who sees what is in Permissions and Personas guide.

5.1 bis Giving the scan account its read access

Scheduled scans run as the user who armed them: the owner of the "OrgGuardian Hourly Scan" job (Setup → **Scheduled Jobs**, "Submitted By" column), that is, the first authorised user who opens the home screen (§5.2). "Run scan now" runs as the user who clicks. This **scan account** needs four system permissions, all **read-only**:

Permission (Salesforce label)	API name	What is missing without it
View Setup and Configuration	`ViewSetup`	Configuration reads (permission sets and their assignments, setup audit trail, scheduled jobs) are refused.
View Roles and Role Hierarchy	`ViewRoles`	The role hierarchy is not read; Salesforce does not grant View Setup and Configuration without it.
View All Users	`ViewAllUsers`	If user sharing is restricted, the population read is truncated **without any error**.
View All Profiles	`ViewAllProfiles`	If profile filtering is on, another account's profile comes back empty **without any error**: privileged and integration accounts are no longer recognised.

The package cannot grant them: a packaged permission set does not carry user permissions, and none of the nine permission sets and groups of the package grants any of these four permissions.

- **The scan account is a system administrator**: its profile already holds them, nothing to do.
- **Otherwise**: Setup → **Permission Sets** → **New**, no licence; under **System Permissions**, tick the four permissions of the table; save, then **Manage Assignments** → add the scan account. If your org does not offer View All Profiles, tick the other three. Do it **before** the home screen is opened for the first time (§5.2): an account without View Setup and Configuration cannot read scheduled jobs, and the bootstrap reports it without being able to check what it armed.

Checking. The **Give the scan account read access to setup** step of the setup tour, on the home screen, reads the rights of the scan account, not yours. It is **Done** when the account holds all four; otherwise it names the account and what it lacks. It says it cannot be verified when your own account cannot read scheduled jobs or the scan account's assignments: that is the case without View Setup and Configuration, even when the scan account is yours.

While a permission is missing, OrgGuardian says so instead of concluding on an incomplete read:

- a **High** read notice, Security category, names the permission (`OrgGuardian:ScanRights:ViewAllProfiles`, for example); it closes by itself, without a verification date, at the first scan that finds the right;
- the Security score is reported as **partial**, and the org health can no longer show as healthy: it is reported as **partial** where it would be green;

- the controls that depend on the permission are **not evaluated**: neither credited with a pass nor closed as fixed; a finding they still raise stays open and counts as a failure;
- alert recipients chosen by role or by permission set may be missed.

Other, optional rights enrich the scans without being required: "Monitor Login History", "Manage MFA in API", "Customize Application". What each one opens, and what else it allows: Permissions and Personas guide § 4.

5.2 Bootstrapping — opening the app

App Launcher → **OrgGuardian** → **Home** tab. On this first opening by an authorised user, the product schedules its recurring jobs (listed in the Operations Guide §1) and launches its reference scan. **There is nothing to schedule by hand.**

The first findings usually appear within a few minutes; meanwhile a banner, **OrgGuardian is starting up**, shows that the initial scan is in progress.

On an org without a paid plan, this first opening also starts the **30-day trial of the full plan**, once per installation: a trial that has started, whether it has ended or not, never starts again. Starting it writes a protected setting, so the user who opens the screen must also hold "Customize Application" (§2.3), as a System Administrator does. The trial starts only while the product bootstraps: once its scheduled jobs and first scan are in place, opening a screen no longer starts it. If you want the trial, the first user to open an OrgGuardian screen must hold "Customize Application".

5.3 Forcing the bootstrap if necessary

From the interface first: on the **Home** tab, the **Setup tour** offers **Run scan now** (step **Run your first scan**) and **Schedule now** (step **Schedule recurring scans**).

The package also provides two administrator commands, to run as anonymous Apex — for example with `sf apex run --file <file> --target-org <your-alias>`:

```
// Restores all scheduled jobs (safe to repeat); returns the number of
// scheduled jobs in the org afterwards, or null if it cannot be read.
System.debug(orgguardian.OrgGuardianAdmin.scheduleMonitoringJobs());

// Queues an immediate scan; returns "OK: …", "REFUSED: …" or "ERROR: …".
System.debug(orgguardian.OrgGuardianAdmin.runScanNow(false));
```

Both commands check two conditions before doing anything, and the refusal is explicit:

- **authorisation** — the caller must hold `OrgGuardian_Manage_Features` **or** one of the platform privileges of §2.3, "Customize Application" or "Modify Metadata Through Metadata API Functions";
- **capability** — the caller must be able to create and edit `Finding__c` and `Org_Metric_Current__c`, which is what a scan writes. A scheduled job runs forever under the identity of whoever scheduled it: without this second check, an authorised holder without these object rights would set up an hourly scan running forever and never writing anything. The `OrgGuardian_Admin` permission set grants both.

On refusal, **nothing is queued or scheduled**: the scan command returns `REFUSED: …` with the reason, and the scheduling command stops with an access-denied error instead of returning a count, so that a refusal cannot be mistaken for an unreadable count.

Publishing to the Heroku collector now. A third command serves the Heroku option: it does what the **Publish now** button of the **Heroku option** panel does, so you can automate it. Unlike the button, it asks for no confirmation.

```
// Queues ONE publish of the aggregated health summary. Returns "OK: ENQUEUED",
// or "REFUSED: " followed by a code, in which case nothing is queued.
String outcome = orgguardian.OrgGuardianAdmin.publishNow();
System.debug(outcome.startsWith('OK:') ? 'Publish queued' : outcome);
```

- **Authorisation**: the same as above. A refusal stops with an access-denied error (`OrgGuardianAdmin.AccessDeniedException`) and nothing is published.
- **Read access**: the publish runs under your identity and reads `Finding__c` and `Org_Metric_Current__c`. If you cannot read these objects, the command returns `REFUSED: NO_READ_ACCESS` and queues nothing: otherwise the summary would leave without its counts and replace the previous one. A field you cannot read is left out of the summary, never published as zero. The `OrgGuardian_Admin` permission set grants this read access. The capability check does not apply: the command schedules nothing, and a publish writes no finding and no metric.
- **The option's own checks**: the command bypasses none of them. `REFUSED: NOT_READY`: the Heroku option is not open for your org (it needs a subscription or a trial in progress), consent has not been given, the collector address is missing, or another condition of the panel prevents the publish. `REFUSED: NO_SIGNING_KEY`: no signing key is saved. `REFUSED: PAIR_UNBOUND`: the saved key and viewer token were not issued by the collector now configured (another collector issued them, or no issuing collector is recorded), so the publish could not be signed for it; run the activation for this collector to receive its own pair.
- **`OK: ENQUEUED`** means a publish is queued, not that it has arrived. Its outcome shows a few seconds later in the same panel (**Last publish**).
- **Contract**: test for success with the `OK:` prefix. Any other return value starts with `REFUSED:` and means nothing was queued. Codes are not translated, and a published code is never renamed. A later version may add codes: treat a `REFUSED:` code you do not know as a refusal.

5.4 Enabling what you need

The differentiating features are shipped **disabled** — their feature switch is off by default — and are enabled without code, feature by feature or in bulk, from the **Administration** hub (**Administration ▸ Settings ▸ Feature activation**), which stores your choices in `Feature_Gate__mdt`. The **Settings** (`OrgGuardian_Config`) and **Enablement** (`OrgGuardian_Enablement`) tabs are not pinned in the app's navigation, but remain reachable by deep link.

No outbound call is made by default. The 22 outbound features — alert channels (5, reading Slack reactions included), exports and integrations (4), external probes (7), self-callouts (6) — are all off at installation and are switched on one by one: a bulk action never switches one on.

Details in the Configuration Guide. A separate publisher notice on enabling the optional features (Event Monitoring, multi-org, SIEM, ITSM, Heroku option) is available on request.

---

6. Checking that the installation succeeded

Six checks, from the quickest to the most conclusive.

1. The package is there.

```
sf package installed list --target-org <your-alias>
```

You should see the `orgguardian` namespace, the package name and the installed version number. Equivalent in the user interface: Setup → **Installed Packages**.

2. The installation trace exists. Setup → Custom Settings → **OrgGuardian Installation** (`OrgGuardian_Install__c`) → **Manage**. The row carries `Version__c` and `Installed_At__c`.

3. The app is visible. App Launcher → **OrgGuardian**. If it does not appear: no permission set is assigned to you (§5.1), or Lightning Experience is not enabled for the org or for you (§2.2).

4. The scheduled jobs are in place. Setup → **Scheduled Jobs**. The OrgGuardian jobs have a name prefixed with `OrgGuardian` — this is exactly the criterion by which the uninstall script tries to cancel them; delete them yourself before uninstalling (the Uninstall Guide §4.2). **Zero jobs after an authorised user has opened the Home tab is a symptom**, not a normal state: see the Troubleshooting Guide.

5. The scan has actually written something. This is the only check that proves the chain works end to end. Open the **Findings** tab — or query:

```
sf data query --target-org <your-alias> \
  --query "SELECT COUNT() FROM orgguardian__Scan_Run__c"
sf data query --target-org <your-alias> \
  --query "SELECT COUNT() FROM orgguardian__Finding__c"
```

Reading the result:

- **0** `Scan_Run__c` → the org has never scanned. The bootstrap has not taken place: go back to §5.2, then §5.3.
- some `Scan_Run__c` but **0** `Finding__c` → the org has scanned without being able to write anything. This is the symptom of the identity without rights described in §5.3: the job belongs to a user who cannot persist what the scan produces.
- some `Scan_Run__c` **and** some `Finding__c` → the chain is complete.

6. The scan account has its read access. The **Give the scan account read access to setup** step of the setup tour is **Done**, and no `OrgGuardian:ScanRights:` notice is open in the **Findings** tab — or query:

```
sf data query --target-org <your-alias> \
  --query "SELECT orgguardian__Dedupe_Key__c FROM orgguardian__Finding__c WHERE
orgguardian__Dedupe_Key__c LIKE 'OrgGuardian:ScanRights:%' AND orgguardian__Status__c = 'Open'"
```

No row: the scan read with its four rights. One row per missing permission: go back to §5.1 bis, then run a scan again. A row ending in `Unverified` means the scan could not read its own rights; it closes at the first scan that can.

---

7. Upgrading

An in-place upgrade, with history preserved, exists only from a released version of the package. From any other version, installing a later version, released or not, first requires uninstalling the installed one, which deletes the package's data. Validate any new version in a sandbox first (§1). After an upgrade, the installed version can be read in Setup → **Installed Packages**, and the installation trace (§6, check 2) carries `Is_Upgrade__c` and `Previous_Version__c`.

Uninstalling is a different operation, which destroys the package's data: the Uninstall Guide.