Table of Contents

1 Overview

2 Getting started and help

3 Laptop client

- 3.1 Install and choose repos
- 3.2 Client reference

4 Self-hosting

- 4.1 Overview and prerequisites
- 4.2 Network
- 4.3 Install the server
- 4.4 Configuration
- 4.5 Model and telemetry
- 4.6 Back up, restore and uninstall
- 4.7 Upgrade and roll back
- 4.8 Troubleshooting
- 4.9 Connect your identity provider
- 4.10 JumpCloud: before the install
- 4.11 Onboard on Auth0
- 4.12 Licence

5 Changelog

1 Winner AI

Team metrics for AI coding agents. Laptops push their coding-agent sessions into a Postgres schema belonging to their organisation, and managers browse them in a dashboard behind single sign-on.

Winner AI is built for a team, not one person. It gives the organisation a roster, enrolls each laptop with a database login of its own, keeps one organisation's sessions apart from another's, and signs people in through the organisation's own identity provider: any OpenID Connect provider, Auth0 by default. Over those sessions it gives managers the Dashboard, to browse, search and read every session, and the pages built on them: Home, Prompt Coaching, Skills and Benchmarks. One installation serves one organisation.

It is **self-hosted**. An organisation runs the stack on a host it controls, and its laptops enroll against that control plane. There is no central service in the middle, and sessions live only on the laptops and in the database on that host. What can leave is what the organisation chooses to send: the jobs behind the product pages, the model check the operator runs in the jobs service, and the control plane for Skills' Generate call a model endpoint, which defaults to one running beside the stack under the `local-llm` profile (a team can set its own on Settings), and the dashboard's generated insights call a second one. Pointing either at a hosted provider sends that provider what Model and telemetry lists, call by call. When the operator sets an error-report DSN, the control plane and the jobs service also send error reports and timings there, and, with the telemetry secret set and a sweep interval of a minute or more, the jobs sweep sends cron check-ins, all without text or raw ids, as the same page lists. When the operator sets them, usage counts also go to an analytics host, with each control-plane page's page views and its own script errors: event names and counts under hashed ids, never text (Model and telemetry).

1.1 What it does

Every enrolled laptop pushes its coding-agent sessions to the organisation's database, and managers read them in a browser. The pages they get:

- **Dashboard.** Every session of the organisation in one place: browse and search them, read transcripts, star, pin, rename and trash them, and generate insights. Curation is shared across the organisation.
- **Home.** The team's 30-day spend, and, for each feature that is on, its figure: skills generated, patterns found and what could be saved, prompts flagged. Then credential exposures, active sessions and output tokens; the team's sessions over 7, 30 or 90 days and by agent; each member's sessions over the last 30 days, last activity and push status; and recent sessions.
- **Admin.** The roster: a manager revokes laptops. With Auth0 Management credentials a manager also adds people; without them, a manager removes and restores people.
- **Prompt Coaching.** Judges the prompts people typed against a fixed rubric (Verification, Planning, Context, Model fit), shows each finding with a clearer rewrite where it has one, and gives each person a few points to improve.
- **Skills.** Finds workflows the team repeats across sessions and names them. A manager turns one into a `SKILL.md` that loads in Claude Code, Codex and Cursor, edits and approves it, and only an approved version downloads.
- **Benchmarks.** Ranks the prompts the team repeats by what they cost and how often they run, and replays recurring self-contained automation tasks on cheaper models to show whether the answer holds and what it would save. Live

traffic is never rerouted.

- **Settings.** Every value the organisation runs with and the layer that set it, plus the team's own model endpoint and key; the configuration reference lists them.

Prompt Coaching, Skills and Benchmarks are off until the operator lists them in `FLEETVIEW_FEATURES`; until then each shows as Soon. A team can switch a listed one off on Settings, and it shows as Soon again. While the licence is lapsed the three leave the sidebar (Licence).

1.1.1 Who it is for

- **Managers** sign in through the browser. Every page above shows the whole organisation, the dashboard included, only to a manager: each checks that the person signed in is a manager of the organisation, and a manager page refuses anyone else.
- **Members** install the `winner` client, sign in once with `winner login`, and choose which of their repos it may send in the list login shows, and later with `winner repos pick`, `winner repos allow` and `winner repos block` or on their own web page at `/repos` (Choose which repos send); the sessions of the chosen repos push in the background. The hourly start-commit report keeps to the same choice: it names a recent session's id, start commit and branch only in a chosen repo. They read their own sessions on their laptop in the session collector (`agentsview`). On the web they see their own Home, their own laptops, their own coaching findings, and the Skills and Benchmarks drawn from their own prompts; every manager page refuses them.
- **The operator** runs the server: installs it, upgrades it to a new release, sets `.env`, and keeps its backups (Install the server, Upgrade and roll back, Back up, restore and uninstall, Troubleshooting).

1.2 Where to start

You are	Read
Installing Winner AI for your organisation, from choosing the server's name to backups	Start here: the install in order
An engineer whose team already runs Winner AI	Laptop client guide
Working out whether it fits, or what an installation needs	Overview and prerequisites, and What each feature sends to the model
Standing up the server	Network, then Install the server
Keeping a bundle install's history safe, or removing it	Back up, restore and uninstall
Installing a later release	Upgrade and roll back

You are	Read
Running a deployment and something is wrong	Troubleshooting
Just signed, or something broke and you need us	Getting started and help
Connecting your identity provider, and adding and removing people	Connect your identity provider
The JumpCloud administrator, before the install	JumpCloud: before the install
Adding an Auth0 organisation's people	Onboard on Auth0
Installing or renewing the licence file	Licence

1.3 How it fits together

Data plane. Each laptop's collector runs as `agentsview pg push --watch` under the background push `winner` installs, and sends the laptop's sessions straight to the organisation's schema. The server runs the Dashboard as `agentsview pg serve` over the same schema.

Control plane. Winner AI's own service: sign-in, the roster, device enrollment, and setting up the organisation's database schema and roles. It is never in the path a session takes, so it can be down while laptops keep pushing.

Identity. The organisation's people sign in through its identity provider, and a group or an e-mail list decides who is a manager; on Auth0 with Management credentials, the organisation, its managers and its members live in Auth0 Organizations. A proxy in front of every page, the control plane's and the dashboard's, makes signing in mandatory, and the control plane tells the edge whether a manager belongs to the one organisation the installation serves.

Isolation. One Postgres schema per organisation, plus one database role per enrolled device. A device's credentials reach that organisation's schema and nothing else.

How the data moves draws the connections between them.

1.4 What changed

The changelog is where each release from 0.4.0 on records what it changed, and whether it asks anything of the people running it.

2 Getting started and getting help

This page is for the people on your side who run Winner AI: your operator, who installs and runs the server; your first Manager, who signs in first and adds the team; and your developers, who install the client on their laptops. It lays out the install in order, says what Winner AI does along the way, where to write when something breaks, how urgent we will treat it, and when to expect an answer.

The severity levels, response targets and roles below are proposed and not yet confirmed. Until the product owner confirms them, they are not a commitment.

2.1 Start here: the install in order

Each step names who does it, what it needs first and the page that covers it. Steps 1a and 1b run in parallel once step 0 is done, and so does 1c, except that on Amazon Bedrock it needs the instance role from 1a. The steps follow the eight steps of Install the server, which is the authority on their order.

Step	What	Who	Needs first	Page
0	Choose the server's DNS name, `<control-plane-host>`, and its base URL, `https://<control-plane-host>`, and the name laptops dial for the database, `PUBLIC_PG_HOST`, which may be the same name. Every later step uses them, the identity provider's callback address included.	Your operator	Nothing	Network, Before you install
1a	Prepare the network, the DNS zone, the VPN where laptops reach a private server, and the server, with an instance role where one is needed: the `dns` certificate mode, Amazon Bedrock signed with the instance role, or secrets kept in AWS Secrets Manager.	Your AWS or network administrator	Step 0	Network, Overview and prerequisites

Step	What	Who	Needs first	Page
1b	Set up the identity provider's application for the sign-in gate, and the Managers' group.	Your identity provider's administrator	Step 0	Connect your identity provider; for JumpCloud, JumpCloud: before the install; for Auth0, Onboard on Auth0, whose A manager adds the first Manager
1c	Get access to the model: on Amazon Bedrock, model access in the AWS account and the instance role's permission to call it. Nothing to do with the server's own AI runner.	Your AWS administrator	Step 0; on Bedrock, the instance role from 1a	Model and telemetry, Amazon Bedrock
2	Install the server: install steps 1 to 5, and step 6's checks of health, the services and the database's certificate.	Your operator	1a and 1b; the bundle and its key's fingerprint from Winner AI (below)	Install the server
3	Sign in as a Manager (install step 6), then upload the licence (step 7's first item).	Your first Manager	Step 2; the licence file from Winner AI (below)	Licence; Check it when sign-in fails
4	Point the stack at the model and set the Benchmarks Routes, run the model check, list the features in `.env` and recreate the services that read it (the rest of install step 7).	Your operator	Step 3, and 1c for a hosted model	Install the server, step 7

Step	What	Who	Needs first	Page
5	Install the client, sign in and choose the repos that send (install step 6 for the first laptop, step 8 for the rest). For Skills, each laptop also needs Ollama with `ollama pull nomic-embed-text`.	Each developer	Step 3, so that a Manager has signed in first; the developer given the application at the identity provider, and on Auth0 added to the organisation	Install and choose repos; Joining; on Auth0, People; Embeddings
6	Schedule backups and copy them off the server, rehearse a restore once, and schedule the certificate-expiry check (install step 8).	Your operator	Step 2	Back up, restore and uninstall, Rehearse a restore, Certificate renewals

Onboarding is complete when the first developer's laptop shows a push on the Admin roster: a session count or a push time, not "awaiting first push" alone. Check the installation on any installation, Checks on Auth0, and Day to day when it does not appear. From then on, write to support@getwinner.com.

2.1.1 What Winner AI does

What	Who	For your step
Hand over the self-hosting bundle. The release public key is printed in the install guide; confirm its fingerprint with your Winner AI contact by a different channel from the bundle.	Winner AI	2: Install the server, whose step 2 checks the bundle's signature against the key
Issue the licence file and send it to your customer contact, before the grace period below runs out.	Winner AI	3: Licence

Until a licence is uploaded, Managers see a banner on Winner AI's pages, and anyone approving a laptop's sign-in sees it on that page, counting down 30 days of grace from the deployment's first start, while Coaching, Skills and Benchmarks keep working; then it says "The 30 days of grace are over." and those three switch off until a licence that brings them back is uploaded (Licence says which licence restores them). The install uploads the licence before the features go on (step 3, then step 4), but nothing refuses to go on without it: if the file has not arrived, step 4 and the steps after it work inside that period.

2.2 Where to write

Write to support@getwinner.com.

A person reads the address, and each message becomes a case in our tracker. Nothing in the product sends to it or depends on it.

Put in the message:

- what you cannot do;
- since when;
- how many people or laptops are affected;
- the release you run.

Attach what applies:

- **From a laptop:** the file `winner diagnostics` writes, as Collecting diagnostics describes.
- **From the server:** the file the diagnostics command writes, as Collecting a diagnostics file describes: `sudo ./scripts/diagnostics-server.sh`, in the folder of the release that runs now, which writes the file into your home directory.

Read each file before you send it, and attach it yourself. Both commands only write the file; they send nothing.

Before you write, Troubleshooting lists what to check first.

2.3 Severity levels and response targets (proposed)

Working hours (proposed) are Monday to Friday, 09:00 to 17:00 in the time zone your contract names, except public holidays there. The targets count working hours only. A first answer means a person has read the case, set its level and replied; it is not a fix. The level follows what you cannot do. We may change it as we learn more, and we tell you when we do. You can ask for a higher level.

Level	What you cannot do	First answer	Updates	Status
1 Stopped	Nobody can sign in, or no laptop can send sessions, or the control plane does not answer.	Within 2 working hours	Every 4 working hours, until it is resolved or a workaround is in place	Proposed

Level	What you cannot do	First answer	Updates	Status
2 A page fails	One product page your installation has turned on fails for everyone (Home, Dashboard, Admin, Settings, and Prompt Coaching, Skills or Benchmarks once switched on), or a group of laptops cannot enrol or send, with no workaround.	Within 1 working day	Every working day	Proposed
3 One person or laptop	One person cannot sign in, or one laptop cannot enrol or send, or a fault has a workaround.	Within 2 working days	Every 5 working days	Proposed
4 A question	How to do something, or a request.	Within 3 working days	With the answer	Proposed

A page shown as Soon is not a fault: Prompt Coaching, Skills and Benchmarks show as Soon until your installation switches them on, and again if a Manager switches one off on Settings.

2.4 Who answers (proposed)

Role	Ours or yours	Does
Support lead	Ours	Reads the address first, at least once in every working hour (proposed), so a Level 1 case gets its first answer within its target; opens the case in our tracker; sets the level; sends the first answer; and is the one voice to you through the case: updates, the fix, closing.
Support engineer	Ours	The engineer on the support rota who owns the part concerned: the laptop client for a laptop file, the server and deployment for a server file or sign-in. The support lead hands the case to them. They write the fix or workaround; their updates reach you through the support lead.
Product owner	Ours	Confirms or changes a level you dispute, owns the response targets, and covers for the support lead when the lead is away.
Second engineer on the rota	Ours	Covers for the support engineer.

Role	Ours or yours	Does
Customer contact	Yours	The contact named in the contract who writes to us, usually your operator or your first Manager.

Proposed: the address always has at least two readers, so no case waits on one person. Who holds each role on our side is recorded in our tracker, not on this page.

3 Laptop client

3.1 Laptop client guide

The `winner` client sends a developer's coding-agent sessions to their team's Winner AI deployment, from the repos they choose and no others. This page takes a developer from nothing to a laptop that pushes, on macOS or on Ubuntu or Debian: install, sign in, choose the repos, and what to expect afterwards. Client reference holds the detail behind each step.

3.1.1 Supported systems

System	Versions	Architectures
macOS	13 (Ventura) or later	Apple Silicon and Intel (one universal binary)
Ubuntu	20.04 or later	amd64 and arm64
Debian	11 or later; 12 on amd64 is the one validated	amd64 and arm64

On Linux the installer refuses, before downloading anything, a machine whose C library is older than glibc 2.28 (the client's session collector, `agentsview`, needs it, and its project publishes no musl or static build), so Ubuntu 18.04 and Alpine are refused. It also needs `ssh-keygen` from OpenSSH 8.1 or later (the `openssh-client` package; `ssh -V` shows the version) to check the release signature. Requirements has the rest.

`winner login` and `winner repos` list the laptop's repos with git 2.36 or newer (`git --version` shows yours). The stock git of Ubuntu 20.04 and 22.04 and Debian 11 is older, and a Mac without the Command Line Tools has none. Those systems stay supported, so the installer warns and installs anyway; until git is updated, the first login on a newly enrolled laptop with coding sessions in a project folder says, for example, "Could not list this laptop's repos (git 2.25.1 found; listing repos needs git 2.36 or newer), so none are chosen and nothing is sent." Requirements says how to get git 2.36 or newer on each system and what to run after.

Windows is not supported in this release: the installer stops with "Windows builds are not published yet."

3.1.2 Install on macOS

Open Terminal and run the installer your team's control plane serves. Replace `<control-plane-host>` with the hostname your team's control plane runs at; whoever deployed it can give you the line with it filled in. Left as written the shell reads `<` and `>` as redirection and installs nothing.

```
curl -fsSL --proto '=https' --proto-redir '=https' https://<control-plane-host>/install.sh | bash
```

The two `--proto` flags keep the fetch on https from end to end: `-L` follows redirects, and without them a redirect to plain http would pipe unverified bytes straight into `bash`. A trial of the server serves plain http on loopback, which those flags refuse; for one, the line is `curl -fsS --noproxy localhost http://localhost:4180/install.sh | bash`, run on the trial machine itself. Any other laptop first needs the tunnel in A trial from the bundle, with port 4180 and the trial's `POSTGRES_PORT` free on it (not held by a stack of its own).

It needs no `sudo` and no GitHub account. It installs `winner`, and the session collector (`agentsview`) at the version the control plane runs when the laptop has none, into `~/.local/bin`. It checks `winner` against checksums signed with the Winner AI release key, and the collector against its project's own checksums, which no one signs (Requirements). It records the control plane in `~/.agentsview/fleetview-control-url`. If that folder is not on your `PATH` it prints the `export PATH=...` line to run; it ends by printing the `winner login` line to run next.

Run it in Terminal, not by downloading `winner` with a browser. A file a browser downloads is quarantined, and macOS refuses to run it; `curl` sets no such flag. If you did download it that way, clear the flag:

```
xattr -d com.apple.quarantine ~/.local/bin/winner
```

Running the installer again on a laptop that is already enrolled is safe: it replaces the binary and keeps the enrolment, the repo choice and the background push. Its "Next step" line can be ignored there.

What you see: the unsigned build

The macOS binary of this release is not signed with a Developer ID; that comes in a later release. Gatekeeper does not stop it, because the installer's download carries no quarantine flag. What you notice is the name: after `winner login`, macOS shows a "Background Items Added" notification named after the binary, "winner", rather than Winner AI.

The login installs three background items, listed under System Settings > General > Login Items > Allow in the Background:

Item (launch agent label)	What it does
`com.getwinner.push`	The background push: sends the sessions of your chosen repos as they change
`com.getwinner.update`	The daily update: runs `winner update` at login and every day at noon
`dev.winner.reporter`	The hourly reporter: tells the control plane the names this laptop pushes under, and the start commit of recent sessions in your chosen repos; sends the name and state of each repo and applies the choices made on the web page

Switching an item off there stops that job; switching off the push stops sending. On the current release `winner login` does not ask macOS for access to Desktop or Documents: the binary carries no signature macOS checks, so sessions in those folders are recorded without a start commit. A signed build changes this. It asks once, for the reporter: Folder access on macOS says what that reads. It asks only when a recent session ran in one of them, through a fourth, one-shot launch agent, `dev.winner.folder-access`, that login installs, runs and removes within the two minutes it waits for your answer; it is not one of the standing items. Whether macOS also announces it with a "Background Items Added" notification of its own has not been checked.

For IT: allowing the items with an MDM

The approved install path is the one above: each developer runs the curl installer, which installs to `~/.local/bin` with no `sudo`. An MDM can then keep the three items on. A Managed Login Items policy (the `com.apple.servicemanagement` payload; JumpCloud offers it for macOS 13 and later) with three **Label** rules, one per label in the table above, approves them, and once applied users cannot switch them off. The one-shot `dev.winner.folder-access` job needs no rule: login removes it as soon as the folder-access question is answered or times out. A single Label Prefix rule cannot cover all three, because the reporter's label starts with `dev.winner.` and the other two with `com.getwinner.`.

Check also that no endpoint security or binary allow-listing tool blocks unsigned executables in users' home folders.

Untested: that a Label rule approves an unsigned item has not been tried on a JumpCloud-enrolled Mac. Until it has, check it on one Mac before rolling the policy out.

3.1.3 Install on Ubuntu or Debian

Run the same installer line as on macOS (Install on macOS), in a terminal, with no `sudo`. It installs the static `winner` binary for your architecture, and the collector when the machine has none, into `~/.local/bin`, under the same checksum checks as on macOS. On a default Ubuntu a new login shell puts `~/.local/bin` on `PATH` once it exists, but the shell you installed from does not: run the `export PATH=...` line the installer printed, or open a new shell, before `winner login`.

Login installs a systemd user unit and two user timers in `~/.config/systemd/user/` (or under `XDG_CONFIG_HOME` when you set it): `winner-push.service` (the background push), `winner-update.timer` (the daily update) and `winner-reporter.timer` (the hourly reporter).

Machines with no login session

The push and both timers belong to your login session, which a laptop gives them. A machine you only reach over SSH starts no session at boot, so they wait until you log in unless you let your user manager linger. That needs no `sudo` on Ubuntu:

```
loginctl enable-linger "$USER"
```

Name the user: a bare `loginctl enable-linger` with no login session fails with "No such device or address".

A Linux machine that does not run systemd (WSL without `systemd=true`, most containers) gets no background push. Login warns that the push was not installed, and prints crontab lines for the daily update and the hourly reporter for you to add with `crontab -e`.

3.1.4 Sign in

```
winner login
```

`--control-url https://<control-plane-host>` is optional on a laptop the installer set up: login uses the flag, then `FLEETVIEW_CONTROL_URL`, then the control plane the installer recorded. Against a trial of the server it is `--control-url http://localhost:4180`.

Login first checks this laptop's collector version against the control plane's (The collector version ceiling). It then prints a code and the full address of the control plane's approval page. Open that address in any browser, on this laptop or on another machine, sign in with your work account and approve the code. Nothing listens on the laptop for the answer.

On a control plane that serves its own sign-in, the laptop talks only to the control plane; your browser is what visits your company's identity provider. A control plane without its own sign-in is the exception (What the laptop sends, and where).

Login then installs the background push and the two scheduled jobs, and, on a laptop with no repo choice yet, asks which repos to send (next section). `winner login --no-service` enrolls without installing any of them.

3.1.5 Choose which repos send

A laptop sends the sessions of the repos its developer chose, and nothing is chosen for you. A laptop that already has a choice keeps it, and login says "Repo selection kept: ..." without asking. Otherwise, at a terminal, login lists the repos this laptop's coding sessions ran in, with **none ticked**:

- numbers and ranges tick and untick rows: `1 3 5-7`;
- `a` ticks all the repos shown except shared or separated names, `n` none of them;
- `/text` shows only the repos matching `text`, `/` shows all again;
- Enter goes on to one last question, what happens to a repo found later (`1` ask, `2` block; Enter keeps the one shown), and the choice is saved after it; `q` quits without saving, and nothing is sent. Ending the input at either question quits the same way.

A login with no terminal chooses nothing and says so: "No terminal to ask on, so no repos are chosen and nothing is sent. Choose them with: winner repos pick". On a fresh machine where no coding agent has run yet there is nothing to list: "agentsview lists no repos on this laptop yet; nothing is sent. Choose them later with: winner repos pick". Use your coding agent once in a repo, then run `winner repos pick`. Where the collector or git cannot list the repos, login says "Could not list this laptop's repos (...), so none are chosen and nothing is sent. Choose them with: winner repos pick", with the reason in the brackets. Where a saved choice cannot be read, login says so, then "Choose your repos again with: winner repos pick".

The list comes from the collector, which the background push starts, so `winner login --no-service` usually cannot ask: unless a collector you started yourself is running, it ends with "agentsview is not running yet, so no repos are chosen and nothing is sent. Once it runs, choose them with: winner repos pick".

To see or change the choice later:

```
winner repos                    # every repo, grouped by what happens to it
winner repos list web           # only those whose name, path or remote holds "web"
winner repos pick               # the same list as at login, from your current choice
winner repos allow web-app      # send it, earlier sessions included
winner repos allow --all        # send every repo listed, except shared or separated names
winner repos block web-app      # stop sending its new sessions; those already sent stay
winner repos block --all        # stop sending any repo's new sessions
winner repos separate api       # give each checkout behind a shared name its own name
winner repos mode ask           # or block: what happens to a repo found later
winner repos apply              # re-apply ~/.agentsview/fleetview-repos.json after a hand edit
winner repos help               # the commands, then what is sent and what cannot be withdrawn
```

`winner repos list` reads from the collector, so it needs the collector running, as it is while the background push runs.

The first time it starts, the collector reads every coding session already on the laptop before it answers, which takes a while on a long history; the background push waits for that and pushes once it ends. Until then `winner repos` says agentsview is still starting and how far it has got, as the collector reports it (for example "initial sync: 300/321 sessions (93%) · 5120 messages"); run it again once it finishes.

`winner repos allow --all` leaves out a name that several checkouts share and a name that was separated, and prints "Skipped ..." for each, saying why; allow those by name.

A repo first seen after you chose is handled by the mode:

- `ask`, the default: "New repos wait for you until you allow them; winner repos lists them as waiting."
- `block`: "New repos are blocked and nothing asks you about them; winner repos still lists them."

What the client says of the repos you do not choose is: "Sessions from repos you don't choose stay on this laptop, unless one of them carries the name of a repo you chose." A Cursor session is named after its folder, not its repository root, so a folder that reads as a chosen repo's name is sent under it.

What was sent stays. Blocking a repo, or unticking it, stops its new sessions only: "Sessions already sent stay with your team." Nothing the client does deletes what was sent, and deleting sent sessions is not in this release. Allowing a repo works the other way: it sends that repo's earlier sessions too, not only new ones.

A repo's name is recorded from its folder name: the basename of the git root, with every `-` read as `_` and case kept, so `web-app` and `web_app` are one repo. When two checkouts share a name, allowing it would send both, so the list and the picker never allow it and name the checkouts. `winner repos separate <repo>`, or `winner repos allow <repo>` at a terminal, asks for a new name for each checkout, suggesting `owner/repo` from its remote, before anything is sent; the new name is what your team sees. A shared name that was already sent cannot be separated: what was sent stays merged under it.

No `winner` command renames a repo already sent: the sessions sent stay under that name, because a rename would make the next push delete them from your team's server. `winner repos rename <repo>` only says so and changes nothing. A rename in the collector's own interface is not covered; see limit 3 below. A separation that stops partway keeps a folder rule under the name you chose for a checkout once that name was sent, or when the record of sent repos cannot be read, and says which and why: removing it would move that folder's sessions out of the name as surely as making one, so its new sessions go under that name. Blocking a repo already sent always works, and what was sent stays.

The collector's search index reads every session on the laptop, chosen or not, and sends its text to the embeddings endpoint its config names. By default that is the laptop's own `localhost:11434`, so nothing leaves the laptop, and no vector of a repo you did not choose reaches your team's database. If the endpoint is set to another host, the text of unchosen repos goes there too: the picker and `winner repos help` then name that host. The index cannot be switched off for one repo, only for the whole laptop.

The choice is kept on the laptop in `~/.agentsview/fleetview-repos.json`, and only the laptop writes it; the names of its repos, unchosen ones included, do leave it (Choices made on the web page). A reinstall that removes `~/.agentsview` removes it, and nothing is sent until repos are chosen again.

A laptop enrolled before repo selection existed sent every repo. Its next `winner login`, or its first `winner update` after the client that added selection is installed (the daily update runs it), allows every repo the collector lists then and records them as chosen, so it keeps sending what it sent and repos found from then on wait for you. If that cannot be done safely, the laptop keeps sending every repo, says why, and checks again at the next update or login; until then `winner repos` changes nothing and says so. The reasons: `agentsview` is not on `PATH`; it has not finished a push since it was installed or updated, or no push log shows one; its names cannot all be read, or its answers cannot be used as they are; or it holds sessions under a name the list would leave out. `winner update` tries only on a run that changed nothing: a run that replaced `winner` or the collector, or was due to, has not pushed with the new version yet.

A laptop with no background push, one enrolled with `--no-service`, is checked only after a `winner login` without `--no-service`.

Choices made on the web page

The web page is **Your repos**, at `https://<control-plane-host>/repos`, linked from a Member's Home page on the server.

On a laptop holding a laptop pass, with the background push installed, the hourly reporter also sends the control plane the name of every repo the collector lists here, chosen or not, each with its state: allowed, blocked, waiting for you, or shared by several checkouts. Nothing else about a repo leaves: no remote, no file name, and no path beyond what the name itself holds, which for Cursor is built from the folder's path. The control plane answers with the choices you made on the web page, and the laptop applies each one itself, exactly as `winner repos allow` or `winner repos block` would, stopping the push, writing the list and starting it again; the control plane never writes the list. A choice takes effect at the reporter's next run, which is within the hour on macOS and on Linux (`winner-reporter.timer`), later while the laptop sleeps or while the collector is not running here, and the page shows it as pending until then.

A choice is refused, with its reason sent back, for a name several checkouts share, a name that was separated, a name this laptop does not have, or a shared name this laptop has already sent. The reporter opens no session folder and runs no git to list them, so each folder a repo's sessions ran in counts as a checkout of its own: a repo whose sessions also ran in a subfolder or a linked worktree shows as shared on the page and is refused there, and `winner repos allow` is the way to allow it.

Each choice is applied once: a change you make with `winner repos` after the laptop took a choice wins over it, and the ids of the choices applied are kept in `~/.agentsview/fleetview-repo-choices.json` until the control plane stops sending them. A choice carries no time, so a change you made before the laptop took it does not win: an allow clicked on the page and then blocked with `winner repos block` before the reporter's next run is still applied, and that repo's earlier sessions are sent, which cannot be withdrawn. Blocking it on the page replaces an allow still pending. A push stopped with `winner push stop` stays stopped and takes the new list at `winner push start`. A control plane that cannot be reached, or answers anything but the list of choices, changes nothing here, and a choice the list could not be written for, or that a push still running on the old list would not follow, because it did not stop or the collector's own push service could not be removed, stays pending; the next run tries again. Until a run, or a `winner repos` change, sees every push on the new list, `~/.agentsview/fleetview-repo-choices-old-list` marks the laptop, and each run stops and restarts the push to try again. `winner repos apply` says what held it: a push that did not stop (`winner push status`), or the collector's own push service, which goes once `agentsview` is on `PATH` or after `agentsview pg service uninstall`.

A laptop enrolled with `--no-service`, one still on a sign-in token from before the laptop pass, or a Linux machine without systemd collects nothing from the page: use `winner repos` there. Nor does a laptop enrolled before repo selection that still sends every repo: it sends no list until `winner update` or `winner login` has kept the repos it sends, and the page's choices wait until then.

3.1.6 Known limits of this release

None of these is fixed in this release.

1. **Cursor spend is missing from laptops.** No Cursor usage rows leave a laptop whose push is filtered, and every laptop with repo selection filters, so the Dashboard has no Cursor spend from laptops.
2. **A later checkout under an allowed name is sent.** A second clone of an allowed repo, a linked worktree made later, or a Cursor folder that reads as an allowed name, is sent under that name without asking.
3. **A rename in the collector's own interface deletes stored sessions.** Renaming a project in the collector's own interface (the agentsview app or its local web page) makes the next push delete the rows stored under the old name on your team's server.
4. **The laptop's database login can delete stored rows.** Its credentials are in `~/.agentsview/config.toml`, and with them plain SQL can delete that laptop's rows on the server. Backups are the answer (Back up, restore and uninstall); narrowing it is being looked at for a later release.

Not yet tested, and so a known limit: a long time off the VPN followed by the catch-up.

3.1.7 What the laptop sends, and where

A laptop contacts its own deployment and nothing else of Winner AI's:

- **the control plane, over https:** the installer and the `winner` binary, the configuration `winner login` reads, the sign-in and enrolment, the hourly report (with the name and state of each repo, Choices made on the web page), and the report of a failed `winner login` or `winner update` (When `winner` fails);
- **the deployment's Postgres, on 5432,** over TLS the laptop verifies against the system trust store (`sslmode=verify-full`): every push of session data goes there.

On a control plane that serves its own sign-in, the laptop never contacts the identity provider; your browser does, while you sign in. On a control plane without its own sign-in (its configuration names an Auth0 tenant instead), the laptop itself does: `winner login` runs Auth0's device flow from the laptop, and the hourly reporter refreshes its token at the Auth0 tenant. So does the reporter of a laptop whose last `winner login` signed in that way, until a `winner login` against its own sign-in. The embeddings endpoint stays on the laptop (`localhost:11434`) unless its config names another host (above).

Two kinds of traffic carry no session data:

- github.com, and the `githubusercontent.com` host it redirects release downloads to, for the collector's own release (`agentsview`, unmodified), which the installer and `winner update` download;
- raw.githubusercontent.com and openrouter.ai, where the collector refreshes its model prices. The collector offers no switch for that refresh.

The collector's anonymous usage ping and its check for a newer version of itself are switched off: every service `winner` installs sets `AGENTSVIEW_TELEMETRY_ENABLED=0` and `AGENTSVIEW_DISABLE_UPDATE_CHECK=1`, and the config enrolment writes sets `disable_update_check = true`. A daemon you start yourself afterwards, from the agentsview desktop app or a shell, carries your own environment instead.

On a private deployment

When your deployment answers only on a private network, the laptop pushes only while it is on the VPN. Off it, pushes fail and the push keeps retrying; sessions wait on the laptop and catch up about a minute after the network returns, none lost. A network change in the middle of a push, such as joining the VPN, can hold the next push back for about two minutes, until the server's keepalive settings drop the old connection; `winner push status` shows nothing wrong meanwhile. `winner login`, `winner update` and the hourly report also wait for the VPN. Home shows a laptop Stale after 24 hours without a push, so during a pilot connect to the VPN at least once a day. A full-tunnel VPN works too, where the private network's own internet access reaches the identity provider and the hosts above: sign-in and updates then go through it. A Mac on a full-tunnel AWS Client VPN has been tested and pushed every session over it. A laptop off the VPN has the detail.

3.1.8 Day to day

```
winner push status      # running or stopped, repeated push failures, and when it last wrote to
its log
winner push logs -f     # follow what it pushes (-n N for the last N lines; 50 by default)
winner push stop        # stops it, across reboots too, and on macOS any running collector
winner push start       # resumes
```

During an outage the push service itself keeps running, so status still says running, and adds that "the last N pushes failed" with the newest failure's reason once two or more pushes in a row have failed. No warning is not proof that pushes land: status reads only the failures the collector records.

A stop holds across reboots, but the next `winner login` re-enables the push: run `winner push stop` again after a re-login if you want it off.

If you signed in before your coding agent had ever run on the laptop, your first sessions can arrive up to 15 minutes late, until the push restarts (a reboot, an update that changes a binary, or `winner push stop` and `start`). After that a session arrives within about 30 seconds of changing.

The client updates itself daily at noon: on macOS also at each login, and a noon the Mac slept through becomes one run when it wakes; on Linux a noon the machine was off for becomes one run at its next start. `winner update` runs the same thing by hand, bringing `winner` and the collector to the versions the control plane runs, in either direction. It restarts the push only when it replaced the collector, replaced the collector's own push service, or rewrote a push service set up before the collector's opt-outs:

```
winner update --check   # report what would change; install nothing
winner update           # converge now
winner update --force   # also replace a winner the installer did not place, or an agentsview
whose version cannot be read
```

`winner update` and Running on its own cover its rules and logs. For support, `winner diagnostics` writes one file to read and attach (Collecting diagnostics).

3.1.9 Uninstall

Uninstalling deletes nothing already sent: what the laptop pushed stays with your team.

On macOS (these steps have not been run end to end on a Mac yet):

```
winner push uninstall
launchctl bootout gui/$(id -u) ~/Library/LaunchAgents/com.getwinner.update.plist
launchctl bootout gui/$(id -u) ~/Library/LaunchAgents/dev.winner.reporter.plist
rm ~/Library/LaunchAgents/com.getwinner.update.plist
~/Library/LaunchAgents/dev.winner.reporter.plist
rm ~/.local/bin/winner
```

On Ubuntu or Debian:

```
winner push uninstall
systemctl --user disable --now winner-update.timer winner-reporter.timer
units="${XDG_CONFIG_HOME:-$HOME/.config}/systemd/user"
rm "$units/winner-update.timer" "$units/winner-update.service" \
   "$units/winner-reporter.timer" "$units/winner-reporter.service"
systemctl --user daemon-reload
rm ~/.local/bin/winner
```

If `FLEETVIEW_BIN_DIR` named another folder at install, remove `winner` from there instead of `~/.local/bin`. On a machine without systemd, remove the crontab lines login printed with `crontab -e` instead of the timers. `winner push uninstall` removes only the background push, and keeps its logs.

None of this ends the laptop's access on the server: its database login and its laptop pass stay valid there. Ask a manager to press **Revoke** beside the laptop on Admin, which ends both.

`~/.agentsview/` holds your local session history beside Winner AI's files and the laptop's database login (`~/.agentsview/config.toml`) and pass, so removing it removes those credentials from the laptop; removing it, and the collector (`agentsview`) itself, is your choice.

3.2 Client reference

The detail behind the `winner` client: what each part does, and the rules it follows. Installing it, signing in and choosing repos is Laptop client guide; what a laptop sends for the web page, and how long a choice made there can take, is Choices made on the web page.

3.2.1 Folder access on macOS

macOS guards Desktop and Documents: a background program that opens either makes the system ask you first. The hourly reporter never opens one on its own. Instead, when some of your sessions from the last seven days ran in Desktop or Documents, `winner login` asks once, while you are at the terminal:

1. It says which of the two folders it is about and what will be read: the git history file (the reflog) of the repositories there, and nothing else.
2. macOS shows its own dialog asking whether "winner" may access the folder, one dialog per folder. Login waits up to two minutes for your answer.
3. **Allow** lets the hourly reporter record start commits for sessions in that folder. **Don't Allow** leaves those sessions recorded without a start commit. Nothing else changes either way.

If no recent session ran in either folder, nothing is asked. If you do not answer within two minutes, nothing is recorded for that folder, it stays unread, and the next `winner login` asks again. A Desktop or Documents that is a link to somewhere else, or is missing, is not asked about; login says so in one line. Inside an allowed folder, a session directory that is a link leading out of the folder is not read. Your answer is kept in `~/.agentsview/fleetview-folder-access.json`, and the hourly reporter reads only that file to decide; it never tries a folder to find out.

Before it reads a reflog, the reporter works out where the session's repository really is, and then reads one file of it: the HEAD reflog in the repository's `.git` directory. To find it, it checks that the repository's `HEAD` and `logs` are there, that the session folder is not itself a git directory, and in a linked worktree it also reads the `.git` file and the `gitdir` and `commondir` files it leads to. Nothing else in the repository is opened. It runs no git, so it needs none installed (`winner login` and `winner repos` do: git 2.36 or newer, see Requirements). This is so on every system the reporter runs on. A session is recorded without a start commit when its folder is reached through a link into a folder you did not allow, when its `.git` is a link or a file that is not a linked worktree's own (one made by `git worktree add` is read as before), when it is inside a git submodule or inside a `.git` directory, or when its repository keeps no reflog file: one with the reflog switched off, or one in git's reftable format.

To change the answer, open System Settings > Privacy & Security > Files and Folders > winner, switch the folder on or off, then run `winner login` again so the new answer is recorded. For a folder you have already answered, `winner login` reads the system's current answer again only when a session from the last seven days ran there; it does so without asking again. It tells you when the answer changed, and a Don't Allow that still holds is printed again at every login, with how to change it. If macOS has lost your answer it asks again, and login says so while it waits.

Between two logins the hourly reporter trusts the recorded answer. If macOS loses an Allow you gave in that time, the reporter's next read of that folder makes macOS ask again with nobody at the terminal: the one case in which the background job asks. A folder you switched off in that time is refused by macOS, and its sessions are recorded without a start commit.

The answer belongs to the `winner` program macOS showed in the dialog, at the place it is installed. After `winner` is replaced by a build macOS treats as a different program, or moved, the folders are left unread until the next `winner login`. A `winner` with no code signature macOS checks is never asked. The macOS binary of the current release is one: macOS never asks about Desktop or Documents, `winner login` says so when your sessions ran there, and sessions in those folders are recorded without a start commit. A signed build changes this.

Downloads, iCloud and other cloud folders, and external volumes are never opened and never asked about; sessions there are recorded without a start commit.

3.2.2 The collector version ceiling

Before the device flow, `winner login` checks the version of this laptop's session collector (`agentsview`) against the control plane's, which is the ceiling: a laptop ahead of the control plane would fail to push. Newer is refused - on a Mac or Linux, enrollment stops and prints, with the versions filled in:

```
This laptop's agentsview (0.45.0) is newer than the control plane's (0.44.0).
Sessions would fail to push: the control plane migrates its own schema,
and laptops must not run ahead of it.
Bring this laptop down to agentsview 0.44.0, then run `winner login` again:
    winner update --control-url 'https://<control-plane-host>'
```

An older laptop only warns and enrolls anyway:

```
WARNING: this laptop's agentsview (0.42.0) is older than the control plane's (0.44.0).
Enrollment can continue; bring this laptop to that version when convenient:
    winner update
```

On a Mac or Linux both messages point at `winner update`, which converges the collector to the control plane's version in either direction. The refusal spells out the control plane it checked against, because it exits before that login records one: a bare `winner update` there would resolve whatever the laptop had recorded before - another control plane, or nothing at all. The warning needs no flag, since that login goes on to enroll and record the URL.

`winner update` installs macOS and Linux builds only, so on any other platform both messages link the collector's own release page for the control plane's version instead. That link is that version's release page, not the latest release, which can be past the ceiling: a refused laptop would reinstall and be refused again.

On a Mac or Linux, against a control plane that names no `winner` version, the advice is the same: `winner update` converges the collector against any control plane that names a collector version, which is the only kind this check compares against. Only the `winner` half is left out, because there is no version to converge it to (an untagged build names none), so both messages add:

```
This control plane names no winner version (every deployment before 0.5.0,
and any untagged build), so that run converges agentsview only and
leaves the winner binary alone.
```

A control plane that names an *older* `winner` gets no such line: the command converges that binary too, downwards.

A control plane that reports no collector version skips the check, with a warning. An already-enrolled laptop keeps pushing with the collector (and `winner`) it has until `winner update` (below) runs on a Mac or Linux, or someone installs `agentsview` by hand elsewhere. On a Mac or Linux, `winner login` without `--no-service` installs the daily check that runs it unattended (below), so nobody has to run it there.

3.2.3 `winner update`

`winner update` converges this laptop's `winner` and its session collector to the versions its control plane runs, in either direction: a laptop ahead is brought back down, since a newer collector is denied on every push. Each binary is independent: one the control plane names no version for is left alone, so a control plane that publishes no `winner` version converges the collector only. It finds the control plane the way `winner login` does, so an enrolled laptop needs no flags.

```
winner update            # converge each binary the control plane names a version for
winner update --check    # report what would change; install nothing
winner update --force    # also replace a `winner` the installer did not place, or an
`agentsview` whose version cannot be read
```

`winner` comes from the control plane's own download route and the collector from its own project's GitHub releases, unmodified; both are checked against the `SHA256SUMS` published beside them before anything installed is touched. For `winner` that manifest counts only once `SHA256SUMS.sig` beside it verifies against the Winner AI release key compiled into the running binary, so an update whose manifest is unsigned, or signed by another key, is refused and the installed binary stays. The collector is checked against its project's own `SHA256SUMS` alone: we cannot sign another project's manifest. A `winner` the installer did not place is left alone without `--force`, since that binary was chosen on purpose; `--force` also reinstalls a collector whose version cannot be read. A binary in a directory this user cannot write is skipped: the other still updates, and the run exits non-zero naming the skipped binary and why. A failed collector download or install likewise leaves `winner` to update, and the run exits non-zero with the failure. Run `winner update --check` first: it reports the same skips without installing anything, though it cannot foresee a download that fails.

Running on its own

On macOS, `winner login` installs the launch agent `com.getwinner.update`, which runs `winner update` with no flags when `winner login` installs it and at every macOS sign-in after that (a reboot included, since the person signs in again), then every day at noon; a noon the laptop slept through becomes one run when it wakes. It follows whichever control plane the last `winner login` recorded; a login that could not record its control plane removes the job rather than leave it following an older one. The rules above hold unchanged: a `winner` the installer did not place is left alone, while the collector still converges. A run that cannot reach the control plane fails before it touches anything; the push keeps running on the binaries it has, and the next day's run tries again.

Every run appends to `~/Library/Logs/winner-update.log`.

```
launchctl print gui/$(id -u)/com.getwinner.update    # loaded, schedule, last exit
tail ~/Library/Logs/winner-update.log                # what the last runs did

# stop it, across reboots too
launchctl bootout gui/$(id -u) ~/Library/LaunchAgents/com.getwinner.update.plist
launchctl disable gui/$(id -u)/com.getwinner.update

# arm it again
launchctl enable gui/$(id -u)/com.getwinner.update
winner login
```

Re-running `winner login` replaces the job rather than adding a second one. A job you disabled needs the `enable` first, or login warns that automatic updates were not installed.

On Linux, `winner login` installs two systemd user timers instead. `winner-update.timer` runs `winner update` when login installs it and every day at noon; it is persistent, so a noon the laptop was off or asleep for becomes one run at its next start. It follows the recorded control plane exactly as the launch agent does. `winner-reporter.timer` runs `winner report` when login installs it and hourly after that; once the machine restarts, its first run comes an hour after your session starts the timer. Their runs append to `~/.agentsview/winner-update.log` and `~/.agentsview/winner-reporter.log`.

```
systemctl --user list-timers 'winner-*'            # both timers and their next runs
systemctl --user status winner-update.timer        # enabled, active, last trigger
tail ~/.agentsview/winner-update.log               # what the last runs did

# stop one, across reboots too
systemctl --user disable --now winner-update.timer
```

As with the push, the next `winner login` enables both timers again. They belong to your login session like the push unit, so a machine with no login session needs the `loginctl enable-linger` step in Machines with no login session. A Linux machine that does not run systemd (WSL without `systemd=true`, most containers) gets neither: login says so and prints the crontab lines that would do the same, with the absolute path to `winner` and your PATH baked in, for you to add with `crontab -e`.

3.2.4 When `winner` fails

`winner login` prints this notice once the laptop is enrolled:

> If `winner` fails after this laptop is enrolled, it sends a short report to your organisation's Winner AI server: the command, the step, the HTTP status, the kind of error, the versions, and your operating system and processor type. The report never contains prompts, code, repository names, file paths, emails or the text of an error. If your organisation's operator has switched telemetry on, the server passes those fields on to Winner AI's error reports and, under a scrambled laptop id, its usage counts. `winner` sends the report nowhere else.

Only `winner login` and `winner update` report. The step is one of `enroll`, `write_config` and `install_service` for a login, and `fetch`, `verify`, `swap` and `converge` for an update. The report also names the laptop's handle, its `machine_name`, which the control plane uses only to find your own enrolled laptop and never passes on. What the server does with it is in Model and telemetry.

A failure stays on screen only, and nothing is sent:

- while a login reaches the control plane and signs in, at every login;
- when a login's enrolment fails and the laptop holds no earlier enrolment with that control plane to report as, as at a first login;
- when the control plane cannot be reached directly, as from a laptop that reaches it only through a proxy: `winner` never sends the report through a proxy or to any other host;
- on a laptop that last signed in with Auth0 rather than the control plane's own login, for `winner update`, because getting a token would contact Auth0;
- on any system other than macOS or Linux, on any processor other than amd64 or arm64, and from a build of `winner` that carries no release version.

The report never changes how the command ends: the error is printed first, and the command exits 1 whether or not the report arrived. A failed install of the background push is reported too, and that login still succeeds. The report follows no redirect and waits at most ten seconds.

3.2.5 Collecting diagnostics

`winner diagnostics` gathers what support needs about a laptop's push into one file, which you read and then attach to an e-mail yourself:

```
winner diagnostics                  # ~/winner-diagnostics-<UTC time>.txt
winner diagnostics -o ~/diag.txt    # a file you name; an existing one is never overwritten
```

The file holds the `winner` and collector (`agentsview`) versions, the operating system and architecture, the push service's state and whether its service file exists, the address of the control plane the laptop enrolled against (scheme and host only), the time and first line of the newest push failure, and the last 50 lines of the push log. It is written even when the push service is not installed, and says so.

It is meant to hold no password, token, e-mail address, repository name or path: every log line passes Winner AI's redaction rule, which removes passwords, tokens and e-mail addresses, and then a stricter pass that replaces paths outside Winner AI's and the collector's own folders and a few binary folders with `[path]`, the machine and database role names with `[machine]` and `[role]`, the organisation's schema with `[org]`, and the laptop's address in a refused connection with `[host]`. A web address keeps only its scheme and host. Your home folder shows as `~`. The command never opens the reporter's token file or the collector's config, which hold the sign-in token and the database password. The push log is written by the collector, though, and a pattern cannot prove every line of it clean, so read the file before you send it.

Without `-o` the file goes in your home folder rather than the current one, which may be a checkout it could be committed from; a second run in the same second adds a number to the name.

The command contacts nothing: it reads local files, asks `launchctl` or `systemctl` about the push service, and runs `agentsview version` (the collector's own version command).

3.2.6 Signing and the background item's name

macOS names a background item, in the "Background Items Added" notification and in the Login Items pane, after the Developer ID that signed the program the launch agent runs. That is why the launch agent runs `winner`: the name shown is whoever signed `winner`. Until the binary is signed, macOS shows the binary's own name instead of Winner AI.

Signing is not what lets the binary run. macOS applies the `com.apple.quarantine` attribute from *applications that opt in* (browsers, Mail, AirDrop), not from `curl` in a terminal, so a binary fetched by the installer carries no quarantine flag and Gatekeeper never challenges it.

Getting Winner AI's name onto the item needs the binary signed with Winner AI's Developer ID and notarized, which comes in a later release.

3.2.7 Requirements

The installer, the `install.sh` your server serves, needs what follows; the list says where it refuses and where it only warns.

- macOS 13 (Ventura) or later, Apple Silicon or Intel. The floor is the toolchain's: the client is built with Go 1.27, which links macOS binaries for 13.0 and up. The installer checks the operating system, never the macOS version. It checks that `ssh-keygen` is there on both systems, and on Linux also the C library and the `ssh-keygen` version, below. The universal binary lands in `~/.local/bin` (`FLEETVIEW_BIN_DIR` overrides it) with no `sudo`; the installer says so if that directory is missing from `PATH`. Nothing is installed unless the download matches a checksum: the `SHA256SUMS` entry served with it, or `FLEETVIEW_AGENT_SHA256` when a mirror supplies the digest instead. A served `SHA256SUMS` is trusted only once `SHA256SUMS.sig` beside it verifies against the Winner AI release key embedded in the script, with the `ssh-keygen` every supported macOS ships (`-Y verify` needs OpenSSH 8.1; macOS 13 carries 9.0). A missing or bad signature, or no `ssh-keygen`, installs nothing. With `FLEETVIEW_AGENT_SHA256` set no signature is fetched or checked, and the script says so: the digest is yours, not the mirror's. The installer then runs the binary and fails if it will not start, but it has already been moved into place by then, so a failure there leaves it on disk to remove by hand.
- The collector, `agentsview`, when it is not on `PATH` or already in that directory, is installed at the version the control plane reports, from its project's GitHub release (`github.com` and the `githubusercontent.com` host it redirects release downloads to must be reachable), into the same directory as `winner`, with no `sudo`, and only if it matches that release's `SHA256SUMS` and reports that version when run. A control plane that reports no version, one reached over plain `http` to a host that is not loopback, or a run with no `FLEETVIEW_CONTROL_URL` to ask, fails and names the release page instead of falling back to the latest release. The control plane is recorded for `winner login` before

this step, so a failure here still leaves `winner` ready. A collector already installed is left alone; the version check at `winner login` still catches one at the wrong version.

- Linux: Ubuntu 20.04 or later, or Debian 11 or later, on amd64 (`uname -m` prints `x86_64`) or arm64 (`aarch64`). The same command installs the static `winner-linux-<arch>` binary into `~/.local/bin` with no `sudo` , under the same signature and checksum rules as on macOS, and the collector from its project's `linux_<arch>` tarball at the control plane's version. Two floors come from what the install needs, and each is refused, with the reason, before anything is downloaded. The collector's Linux builds need glibc 2.28 or newer and its project publishes no musl or static build, so Ubuntu 18.04 and Alpine are refused. The signature check needs `ssh-keygen` from OpenSSH 8.1 or later (`ssh -V` ; the `openssh-client` package), so a machine without it, or with an older one such as Debian 10's 7.9, is refused too, unless `FLEETVIEW_AGENT_URL` and `FLEETVIEW_AGENT_SHA256` supply the digest and no signature is checked. The checksum uses `sha256sum` , or `shasum` where that is missing. Everything after enrollment is the same, the background push and `winner update` and its daily timer included.
- git 2.36 or newer, for `winner login` and `winner repos` , which list the laptop's repos with it. The listing needs `worktree list -z` (2.36), and a git older than 2.31 answers its questions wrongly without failing, so `winner` checks the version and stops instead: the first login on a newly enrolled laptop chooses no repos and sends nothing, and `winner repos` fails naming the git it found. The background push and the hourly reporter run no git. Stock git is new enough on Ubuntu 24.04, Debian 12 and macOS with Command Line Tools 14 or later, and too old on Ubuntu 20.04 and 22.04 and Debian 11, which stay supported, so the installer warns, never refuses, when git is missing, older or unreadable. `git --version` shows yours. To get a newer one:
- macOS: `xcode-select --install` , or `brew install git` .
- Ubuntu 20.04 or 22.04, from the git-core PPA (its 20.04 builds are no longer updated):

```
sudo add-apt-repository ppa:git-core/ppa
sudo apt update && sudo apt install git
```

- Debian 11: `bullseye-backports` carries git 2.39, now only from Debian's archive and no longer updated, so upgrading to Debian 12 is the lasting fix:

```
echo 'deb http://archive.debian.org/debian bullseye-backports main' | sudo tee
/etc/apt/sources.list.d/bullseye-backports.list
sudo apt update
sudo apt install -t bullseye-backports git
```

Then choose the repos with `winner repos pick`, or, on a laptop enrolled before repo selection that kept sending every repo, run `winner login`. - `FLEETVIEW_CONTROL_URL` names the control plane when the script is run from a checkout instead of fetched from one; the served copy has it filled in. - `FLEETVIEW_AGENT_URL` fetches the binary from somewhere other than the control plane, a mirror say. It needs no token; set `FLEETVIEW_GH_TOKEN` only for an https URL that wants a GitHub token, which is sent as an `Authorization: token` header and never over plain http. The `SHA256SUMS` manifest is looked for *beside* that URL, so the URL has to be a plain `scheme://host/path` naming the file. A presigned URL (anything with a query string), a URL with a `#` fragment, a URL with no path and a trailing-slash path have nothing beside them: set `FLEETVIEW_AGENT_SHA256` to the expected digest, 64 hex characters, and the installer skips the manifest fetch and the signature check. Without one of the two it refuses before downloading anything. A mirror that serves a manifest must also serve the release's `SHA256SUMS.sig` beside it, unchanged, or the installer refuses. A mirror that is not https needs `FLEETVIEW_AGENT_SHA256` even when a manifest does sit beside it: both files would travel in the clear, so whoever can swap the binary can swap the manifest too. That refusal also comes before any download. - `curl` 7.58 or newer, if you set `FLEETVIEW_GH_TOKEN`. From 7.58 curl keeps a custom `Authorization` header on the original host instead of forwarding it across a redirect elsewhere. Every macOS and Linux release this installer supports ships a newer curl (Ubuntu 20.04 has 7.68, Debian 11 7.74), so nothing checks the version at runtime.

3.2.8 Where the control plane runs

Your team's stack runs at `https://<control-plane-host>`. Enrolled laptops push to its Postgres on port 5432 over TLS, verifying the server against the system trust store (`sslmode=verify-full`): Postgres presents the same publicly trusted certificate as the https edge, so a renewal on the host changes nothing here. A laptop whose collector config (`~/.agentsview/config.toml`) carries `sslrootcert=` pins one certificate instead of trusting the system store, and fails once the server presents another; one `winner login` against a control plane that hands out no certificate to pin rewrites the config without it.

4 Self-hosting

4.1 Self-hosting: overview and prerequisites

Read this before installing anything. It says what a Winner AI installation is made of, what the server and your organisation must provide, and every connection that leaves your network. Installing the Winner AI server is the install itself.

Three things decide whether you can install today:

- **Sign-in is through an OpenID Connect provider over https**: Auth0, or the one your organisation already runs, such as JumpCloud.
- **A publicly trusted certificate for the server's names**, in one of two modes. Either the internet reaches the server on 80 and 443, where the certificate authority checks each name, or, for a server only a private network or VPN reaches, the names sit in an Amazon Route 53 public zone and the server has an AWS instance role that may write their challenge record.
- **One linux/amd64 server with Docker.** Nothing else is supported.

4.1.1 What you run

Winner AI has no hosted service. Each organisation runs its own installation: one server that holds the organisation's coding-agent sessions, signs its people in through its identity provider, and serves the Dashboard and the product pages to its Managers. A client on each developer's laptop sends that laptop's sessions straight to the server's database. Nothing of ours connects to the server. We build and sign the releases and answer support; you run everything else.

The database holds every session your developers' coding agents ran: their prompts, the agents' replies and the tools they called. Protect the server and its disks as you would a source repository.

Part	What it does
Edge	Stock Caddy. Terminates https on 443 for the server's names, and on 80 redirects to https and, in the `http` certificate mode, answers the certificate authority.
Sign-in gate	Stock oauth2-proxy. Every page passes it and signs in through your identity provider, apart from the few addresses that are public by design or that a laptop calls with its own token.
Web server	Stock Caddy with one pinned plugin, behind the gate on every request: it routes to the control plane and the dashboard, admits only the installation's Managers to the dashboard, and brands its pages.
Control plane	Our application: sign-in landing, the roster, laptop enrollment, the organisation's schema and database roles, the product pages, Settings, and these guides at `/guide/`.

Part	What it does
Jobs service	The same application's background passes: Prompt Coaching, Skills, Benchmarks, role sync, the sampling of which laptops pushed, and the heartbeat.
Dashboard	The session dashboard Managers open from the control plane: browse, search and read every session of the organisation. Its container runs `agentsview pg serve`.
Database	Postgres with pgvector. The archive of every session; laptops push to it over TLS on 5432.
Certificate helpers	`db-certs` hands Postgres the certificate for its name; in the `dns` mode, `dns-certs` (stock lego) obtains and renews the certificate through Route 53, and `edge-reload` tells the edge when it changes.
AI runner	Stock Ollama, on by default (the `local-llm` profile in `COMPOSE_PROFILES`). Serves the embedding model and, unless you point them elsewhere, the model calls. It runs models on CPU.
Laptop client	`winner` and the session collector (`agentsview`) on each developer's laptop. Laptop client guide covers it.

4.1.2 How the data moves

```
laptop (agentsview pg push) ──5432, TLS──> database (the organisation's schema)
laptop (winner)             ──443──> edge ──> sign-in gate ──> web server ──> control plane
(enrollment, reports)
browser                     ──443──> edge ──> sign-in gate ──> web server ──> control plane,
dashboard ──> database
sign-in gate, control plane ──> your identity provider
control plane, jobs, dashboard ──> AI runner, or your hosted model endpoint
```

A laptop sends its sessions straight to the database, as a database role that belongs to that one laptop and reaches its organisation's schema and nothing else. The control plane is not in that path: it enrolls the laptop and hands it the role, and laptops keep pushing while it is down.

4.1.3 The supported shape

One linux/amd64 server running Docker Compose, installed from the release bundle. Before installing anything, the installer refuses anything but Linux on x86-64, and a Docker daemon that is not linux/amd64. It builds nothing on the server: the control plane's, the Dashboard's and the sign-in gate's images come out of the bundle, and the other public ones are pulled from Docker Hub at fixed, checked versions.

Not supported: an arm64 server, a container service such as Amazon ECS, Kubernetes or another orchestrator, and an installation spread over several servers. Laptops on arm64 are supported; only the server is amd64.

4.1.4 What you operate

- **The server and its operating system**: updates, disk, monitoring, and where the server runs.
- **Hardening.** Winner AI ships no firewall rule, jail or filter, and nothing on a bundle install checks which ports the server publishes.
- Admit only 80, 443 and 5432 from outside, plus your own administration access. Docker's own rules publish container ports past host firewalls such as ufw and firewalld, so set the rule in your network or cloud firewall, or in Docker's `DOCKER-USER` chain, then check from outside that only those three answer.
- 5432 is open to wherever laptops connect from, so anyone there can guess passwords against it. Only per-laptop roles are admitted from outside; banning an address after repeated refused logins slows guessing. A fail2ban jail reading the database container's log that bans after five refused logins in ten minutes, for an hour, is a starting point. Never ban the stack's own subnet (below), and give an address many laptops share (a VPN or an office's NAT) a far higher count: one laptop with a stale password writes about two refused logins a minute, and would lock out everyone behind that address (The Postgres jail).
- **DNS and the certificate**: the server's names and their records.
- **The identity provider**: who may sign in, and who is a Manager.
- **The model endpoint**, if you use a hosted one, and its key.
- **Backups.** The bundle ships the backup and restore commands, but nothing on a bundle install schedules a backup or copies one off the server.
- **Upgrades.** Nothing upgrades itself: Upgrade and roll back says how to install one.
- **Telemetry**: off unless you switch it on (Model endpoint and telemetry).

4.1.5 Prerequisites

Everything below is needed before the install. The model endpoint is optional.

The server

- **linux/amd64 (x86-64), Ubuntu**, with **Docker Engine 28.0 or later and the Docker Compose plugin 2.24.4 or later**. The host check the install runs refuses an older one, and a Docker daemon that is not linux/amd64. Check the requirements says how to install them.
- **A user who can run `docker`**, who runs the install.
- **`ssh-keygen` (OpenSSH), `sha256sum` and `tar`**, to check and unpack the bundle, **`openssl` and `curl`**, for the install's secrets and checks, and **`ss`** (the `iproute2` package), for the host check's ports; Check the requirements has the version `ssh-keygen` must be.
- **Disk.** 5 GB free where Docker keeps its data, 15 GB with the AI runner, plus room for the unpacked bundle. No CPU or memory figure has been measured yet. The database grows with your team's sessions, and the AI runner, which runs models on CPU, is the heaviest service.

- **Ports 80, 443 and 5432 free** on the server, and **a clock within 5 seconds of the real time**, kept by time sync, which the host check compares.
- **No network of its own at `172.16.86.0/24`.** The stack pins that subnet for its own containers, and the database admits every role, its superuser included, from inside it. The server's networks, routes and any VPN range laptops connect from must not use it.
- **Outbound https** to Docker Hub (the install fails if a pull does), your identity provider's issuer (the sign-in gate does not start without it), a public certificate authority, the Route 53 API in the `dns` certificate mode, and the Ollama registry while the AI runner is on. What leaves the server lists every destination by name.

DNS, certificate and ports

- **A public DNS name for the server**, and one for the database (it may be the same name), resolving to it from wherever your people and laptops are. Use names, not IP addresses; `scripts/check-env.sh` refuses an IP address for the database's name.
- **A certificate mode** (`CERTIFICATE_MODE`), which decides how the certificates are taken over ACME. The edge presents one for each name, and the database presents its name's, so laptops verify it against their system trust store.
- **`http`, the default: 80 and 443 reachable from the internet, for both names.** The certificate authority connects to the server on 80 or 443 to check each name.
- **`dns`, for a server nothing on the internet reaches: both names in an Amazon Route 53 public zone, and an AWS instance role on the server** that may write the challenge record in it. The certificate authority reads a TXT record instead of connecting to the server, so no inbound port is needed for the certificate. Network says which mode to choose, and Private VPC: the certificate through DNS the role's policy.
- **5432 reachable from every laptop**, which pushes there over TLS: from the internet, or only from your office and VPN ranges if every laptop connects from them.
- **An e-mail address for the certificate account** (`ACME_EMAIL`).

Sign-in

A production installation signs in through **any OpenID Connect provider** whose issuer is https: Auth0, or the provider your organisation already runs. `scripts/check-env.sh`, which the installer runs, refuses an issuer that is not https, and a provider other than Auth0 while Auth0 Management credentials are set.

On a provider other than Auth0, the provider needs an application for the sign-in gate, and the Managers are a group at the provider or a list of addresses in `.env`; the Auth0 settings hold placeholders and the Management credentials stay blank. Connect your identity provider is the checklist, and JumpCloud: before the install a worked example.

On Auth0, the tenant needs:

- **one organisation** for the installation, with sign-in requiring it and without adding people to it at sign-in;
- **the post-login Action**, which puts the roles and the organisation on each sign-in and refuses a sign-in to the gate through any other organisation, with its two secrets set to the installation's values;

- **a regular web application** for the sign-in gate, with the server's callback and logout addresses;
- **a native application** with the device-code grant and offline access, which only a `winner` older than the control plane's own laptop sign-in uses: a current `winner` signs in through the control plane;
- with Auth0 Management credentials, which let Admin add people and read roles from Auth0, **a machine-to-machine application** for the Management API and **the roles `manager` and `member`**: with the credentials set, the control plane does not start without the roles.

Who may sign in, Fill in `.env` and The post-login Action say how each is set, and Configuration lists every setting.

A model endpoint

None is required. The AI runner in the bundle serves the embedding model and, by default, every model call. To use a hosted model for the rest, you need an OpenAI-compatible endpoint over https and its key, or Amazon Bedrock; Model endpoint and telemetry says what the endpoint must support, how to check it before switching Prompt Coaching on, and why the AI runner keeps serving embeddings (Embeddings).

From Winner AI

- **The release bundle**, `winner-ai-server-<X.Y.Z>.tar.gz`, and the release key's fingerprint, which you confirm with your Winner AI contact over a different channel from the bundle; Verify the bundle's signature prints the one to confirm.
- **A licence file.** A Manager uploads it on Settings, which checks it from the file alone, never with us. Until one is uploaded, the deployment runs 30 days of grace from its first start. Licence covers installing and renewing it.

4.1.6 What leaves the server

These are the connections Winner AI's software on the server makes, as read from its code; no capture of a running stack has been run yet. Your operating system's own traffic (package updates, time, your cloud provider's services) and any copy of a backup you set up are yours, and not listed. Every connection is outbound https, on 443 unless an address you set (the model endpoint, the heartbeat or a telemetry host) names another port. Three exceptions. Two are plain http on 80. When the edge's certificate names an OCSP responder, the edge fetches the certificate's revocation status from that address. And on AWS, whatever signs with the server's instance role reads the role's credentials from the instance metadata service at the link-local address `169.254.169.254`: Amazon Bedrock signed with the instance role, the certificate helper in the `dns` mode, and `scripts/check-dns-zone.sh` and `scripts/env-from-secrets.sh` when you run them. The third is DNS on UDP port 53: `scripts/check-dns-zone.sh`, run before the first `dns`-mode start, asks the public zone's own name servers for its test record directly. The certificate helper itself needs no outbound port 53.

Destination	When	Why
Your identity provider's issuer, `<OAUTH2_PROXY_OIDC_ISSUER`	Always	The sign-in gate reads its configuration, tokens and keys. On Auth0, the control

Destination	When	Why
`_URL>` (on Auth0, `<AUTH0_DOMAIN>`)		plane also checks an older `winner` 's Auth0 token against its keys and, with Management credentials, calls its Management API for the roles, the roster, enrollment and role sync.
A public ACME certificate authority: Let's Encrypt (`acme-v02.api.letsencrypt.org`), and ZeroSSL (`acme.zerossl.com` , and `api.zerossl.com` for its account credentials) when Let's Encrypt fails	At first start and around each renewal	The certificates. In the `http` mode the authority also connects in to 80 or 443 to check each name; the `dns` mode uses Let's Encrypt only.
The Route 53 API	Around each issue and renewal, in the `dns` mode	The certificate helper writes the challenge record, signing with the server's instance role.
Docker Hub: `registry-1.docker.io` , `auth.docker.io` , and the storage host it redirects image data to	At install and upgrade, and when you run `scripts/check-dns-zone.sh`	The installer pulls the public images: the database, the edge and web server base, the certificate helper in the `dns` mode, and the AI runner while `local-llm` is on. The zone check looks up the certificate helper's image. Docker Hub limits pulls without a login per outbound address.
`ghcr.io` , GitHub's registry	At install and upgrade, when `trial` is in `COMPOSE_PROFILES`	The trial's sign-in provider (Dex), pulled by digest.
The Ollama registry, `registry.ollama.ai` , and the storage host it redirects model data to	At every AI runner start, while `local-llm` is on	The runner pulls the embedding model, and the chat model when an endpoint points at the runner.
Your hosted model endpoint, and any endpoint a Manager sets for the organisation on Settings	When one is set	Prompt Coaching, Skills, Benchmarks and the dashboard's insights. What each feature sends to the model lists what each sends.

Destination	When	Why
`raw.githubusercontent.com`, `openrouter.ai`	When the control plane sets up the organisation's schema: at first start (twice when the first attempt fails), at every later start until one succeeds, after an upgrade that changes the collector version, and when a Manager runs the migration from Admin	Setting up the schema refreshes the model prices that cost figures use. There is no switch for it, and laptops make the same refresh.
`FLEETVIEW_HEARTBEAT_URL`	After every jobs sweep, when you set it	Your own dead-man's switch.
The error-report and analytics hosts you set	Only when you switch telemetry on	Telemetry lists each kind and every field it sends.

Allow the two redirect storage hosts by the names your proxy or firewall logs for them: they are content-delivery hosts whose names and addresses change.

There is no licence check-in and no update check. The Dashboard runs with its usage ping and update check switched off, and the control plane sets up schemas with the usage ping off.

Two other parties see traffic because of the installation, not from the server:

- **People's browsers** load fonts from Google Fonts (`fonts.googleapis.com` , `fonts.gstatic.com`): Archivo and JetBrains Mono on the control plane's pages, and Archivo on the dashboard. These guides serve their own.
- **Laptops** reach the control plane, the database and a few hosts of their own; What the laptop sends, and where lists them.

4.2 Network: public, private VPC, AWS Client VPN

Laptops reach a deployment on two ports: 443 for the control plane and 5432 for Postgres, where every push goes. Both present a publicly trusted certificate that laptops verify against their system trust store (`sslmode=verify-full`), so there is no private certificate authority to install and a renewal touches no laptop. There are two modes, and what changes between them is who can reach those ports and how the certificate is obtained:

- **Public:** the internet reaches the host, and the edge takes its certificate over port 80.
- **Private VPC:** nothing from the internet reaches the host. The certificate comes through a DNS record in a public zone, and laptops reach the host over a private network.

Where that private network is an AWS Client VPN, AWS Client VPN says what the deployment needs from it. Laptops that already reach the VPC some other way skip that section; the product needs no VPN of its own.

A deployment behind a CDN is a public one with four more steps (Behind a CDN). This page covers the network around the server; Install the server is the server itself.

4.2.1 Before you install

The install is self-serve, so settle these first. Each one decides a value in `.env` , a record in DNS or a rule in the network.

Settle this	Why	Where it goes
The public name or names: the control plane, and Postgres if it differs	DNS names, no port, never an IP address: the edge asks for certificates for names only, so a laptop verifying with `verify-full` could not check an address. In the `dns` mode they must also be lower-case	`PUBLIC_HOST` , `PUBLIC_PG_HOST` (Fill in `.env`)
Whether the internet can reach the host on 80, 443 and 5432	Decides the mode	`CERTIFICATE_MODE` : `http` if it can, `dns` if not
The DNS service that hosts the public zone, its zone id, and the region the helper's AWS client uses (a Route 53 zone has none)	The `dns` mode writes its challenge record through Route 53 only today	`AWS_HOSTED_ZONE_ID` , `AWS_REGION`

Settle this	Why	Where it goes
Any CAA record on the zone or a parent	It must allow the authority: `letsencrypt.org`, and in the `http` mode ZeroSSL too, the authority the edge falls back to (check ZeroSSL's own CAA value)	Your DNS service
The VPN product and how it attaches (subnet association, Transit Gateway, other)	Decides whether laptops share addresses, and so the jails	`VPN_RANGES`, and the same ranges typed by hand into the jails (The Postgres jail)
Whether the VPN signs each person in, or admits any laptop holding a shared certificate	A per-person sign-in names who is behind 5432; a shared certificate proves only that a laptop holds it	Your VPN
Split or full tunnel, and whether the VPN pushes the VPC's resolver	A laptop must resolve the private name over the tunnel and reach the identity provider and the hosts the laptop reaches: over its own internet with split tunnel, or through the VPC's internet egress (a NAT gateway, say) with full tunnel	Your VPN, AWS Client VPN
The address ranges laptops appear from, and the VPC and client ranges	None may overlap `172.16.86.0/24`, the compose subnet, where `pg_hba.conf` admits every role, not only device roles, with a password. Ranges in `VPN_RANGES` must be IPv4, /30 or wider, and private: `scripts/check-env.sh` checks that, and nothing else on the server reads them	`VPN_RANGES`; your VPC and VPN
The instance metadata settings (`dns` mode on AWS, or Bedrock with the instance role)	IMDSv2 required, hop limit 1, or 2 when Bedrock signs with the instance role (Amazon Bedrock); at hop limit 2 every container can use the role, so it should hold only what each use needs	The instance
Outbound DNS (`dns` mode)	The helper needs none: no outbound port 53 and no public resolver; the VPC resolver suffices. The check you run before the first start, `scripts/check-dns-zone.sh`, does: outbound UDP port 53 to the zone's own name servers	The security group and network ACLs
Outbound https	Each mode has an allow-list	What the server reaches

4.2.2 Choose a mode

Mode	`CERTIFICATE_MODE`	Reaches 443 and 5432	Port 80	VPN ranges (`VPN_RANGES` and the jails)
Public	`http` (or unset)	the internet	the internet: the certificate challenge and the https redirect	none
Private, laptops share the VPN's addresses (a Client VPN associated with subnets)	`dns`	the VPN only	optional, the redirect only	the associated subnets
Private, laptops keep their own addresses (a Transit Gateway attachment, or a network with no translation)	`dns`	the private network only	optional, the redirect only	none

Both private modes install from the bundle, as the public one does (Install the server).

On a server installed from the bundle, `VPN_RANGES` in `.env` is read only by `scripts/check-env.sh`, which checks its shape. fail2ban never reads `.env`: you type the same ranges by hand into the jail's `ignoreip` and the VPN jail (The Postgres jail).

4.2.3 Public deployment

1. **Point the names at the server** with an A record each. The server publishes its ports on IPv4 only, so give no AAAA record that leads to it.
2. **Open 80, 443 and 5432 to the internet**, and nothing else. Port 80 carries the certificate challenge and the redirect to https. Nothing on the server checks which ports are public, so the firewall in front of it is yours to keep (Check the requirements).
3. **Set `.env`** with `CERTIFICATE_MODE` unset or `http` (Fill in `.env`).
4. **Check CAA**: a record that names neither Let's Encrypt nor ZeroSSL, the authority the edge falls back to, refuses the certificate. Check ZeroSSL's own CAA value before you add it.
5. **Postgres stays locked down**, whatever the network allows:
6. from outside, only device roles (`+fleetview_devices`), only over TLS, only with a SCRAM password (`deploy/db/pg_hba.conf`); the superuser and every owner role are refused before a password is read;
7. laptops connect with `sslmode=verify-full` and pin nothing;

8. the jail `fleetview-postgres`, which you set up on the server, bans an address after five refused logins in ten minutes, for an hour, acting in `DOCKER-USER` (The Postgres jail);
9. `VPN_RANGES` stays blank, and the jail ignores no VPN range.
10. **Verify** with Check the installation.

4.2.4 Behind a CDN

A deployment may put a CDN or reverse proxy in front of the edge. Every request then arrives from that network, so the peer address is the CDN's and the visitor's survives only in a header.

- **Give Postgres its own name.** A proxied name carries no port 5432, so set `PUBLIC_PG_HOST` to a name that points at the server itself, not through the CDN. The edge certifies whichever names it serves.
- **Set `EDGE_TRUSTED_PROXIES`** in `.env` to the CDN's ranges, space-separated. The edge then believes the visitor's address from those networks and ignores it from everywhere else, so no client can claim an address by sending the header. Leave it unset when nothing sits in front.
- **The proxy must set `Cf-Connecting-Ip` itself**, overwriting any value the client sent, to the address it saw the visitor connect from; Cloudflare does. That is the only header the edge reads an address from: `X-Forwarded-For` is a list a client can prepend to, and a proxy that passes a client's own `Cf-Connecting-Ip` through lets that client choose the address the edge believes. A proxy that sends only `X-Forwarded-For` leaves the edge reading the proxy's own address, so the setting looks applied and changes nothing.
- **Leave the certificate challenge alone.** In the `http` mode the challenge reaches the edge through the CDN, so the CDN must not answer or redirect `/.well-known/acme-challenge/` itself.

The edge's access log, `docker compose logs edge` in the install folder, shows one line per request with `client_ip` (the visitor) beside `remote_ip` (the peer). Two different values mean the trust works; two identical ones mean the request did not come through a trusted proxy.

4.2.5 Private VPC: the certificate through DNS

This mode has run on AWS: the shipped `dns-certs` helper issued a production certificate through a Route 53 public zone with the instance role, on a clean install in a private network. A certificate for two names has run only against a test authority (Known limits).

1. **One zone, public and private.** Both names, `PUBLIC_HOST` and `PUBLIC_PG_HOST` when it differs, live in one public Route 53 zone, because the helper is given one zone id. That zone holds nothing for them but the challenge records, which the helper writes and removes itself. A private hosted zone of the same name, associated with the VPC, holds an A record for each name to the server's private address; the VPC needs DNS support and DNS hostnames on. Inside the VPC each name resolves to the private address; outside it does not resolve, yet it can still be certified.
2. **Give the server its instance role** with this policy and no more, the zone id and every challenge name filled in (one name when `PUBLIC_PG_HOST` is `PUBLIC_HOST`):

```
{
  "Version": "2012-10-17",
  "Statement": [
    { "Sid": "ReadChangeStatus", "Effect": "Allow",
      "Action": "route53:GetChange", "Resource": "arn:aws:route53:::change/*" },
    { "Sid": "ReadTheZone", "Effect": "Allow",
      "Action": "route53:ListResourceRecordSets",
      "Resource": "arn:aws:route53:::hostedzone/<ZONE_ID>" },
    { "Sid": "WriteOnlyTheChallengeRecord", "Effect": "Allow",
      "Action": "route53:ChangeResourceRecordSets",
      "Resource": "arn:aws:route53:::hostedzone/<ZONE_ID>",
      "Condition": { "ForAllValues:StringEquals": {
        "route53:ChangeResourceRecordSetsNormalizedRecordNames": ["_acme-challenge.
<PUBLIC_HOST>", "_acme-challenge.<PUBLIC_PG_HOST>"],
        "route53:ChangeResourceRecordSetsRecordTypes": ["TXT"] } } }
  ]
}
```

Require IMDSv2 at hop limit 1, or 2 when Bedrock signs with the instance role. The helper runs on the host network, so hop limit 1 is enough for it; at hop limit 2 every container can use the role, its Route 53 statement included, so keep its policy to what both uses need, or sign Bedrock with an API key and stay at 1. `ListHostedZonesByName` is not needed, because the zone id is given.

3. **Allow outbound https** to the certificate authority and the Route 53 API (through a NAT gateway or its equal). The helper needs no outbound port 53 and no public resolver; the VPC resolver suffices (why is in `docker-compose.prod.yml`, at `CERT_DNS_WAIT`). The check in step 5 does: allow outbound UDP port 53 to the public zone's own name servers, and outbound https to Docker Hub, where it looks up the helper's image.
4. **Set `.env`**: `CERTIFICATE_MODE=dns`, `AWS_HOSTED_ZONE_ID` (the **public** zone's id, never the private one's), `AWS_REGION`, and `ACME_EMAIL`, which compose still requires; `scripts/check-env.sh` names any it refuses.
5. **Before the first start, run `scripts/check-dns-zone.sh`** in the bundle folder (Fill in `.env`). With the instance role, it writes a test challenge record, asks each of the zone's name servers for it directly over port 53, removes it, and checks that Let's Encrypt answers and that the helper's image resolves at Docker Hub. Where only https may leave, its read-back fails although the issue could work. An install from the bundle then starts the helper and waits for its first certificate (Install the server).
6. **Tell people first, and expect a gap at first install.** Both names become public with the first certificate, in the public certificate transparency logs, and nobody can enroll until it exists (Certificate renewals).
7. **Security group:** admit 443 and 5432 only from where laptops come from, 80 too if you want the redirect, and nothing from the internet.
8. **Postgres:** the same rules as the public mode, step 5. Laptops must dial the name, never the address, because `verify-full` fails by address. Whether the jails need the VPN's ranges depends on shared addresses (AWS Client VPN).
9. **Verify from a machine on the private network**, which resolves and reaches the host as laptops do (Check the installation).

4.2.6 Certificate renewals

No laptop is involved in a renewal: laptops verify against their system trust store, so a new certificate from the same public authority needs nothing from them.

- **`http` mode.** The edge renews its own certificate about thirty days before it expires. Within five minutes `db-certs` installs it for Postgres and reloads it; `docker compose logs db-certs` shows `installed the edge's certificate for <PUBLIC_PG_HOST> (notAfter=<date>)` and `Postgres reloaded and presents the installed certificate`.
- **`dns` mode.** The `dns-certs` helper checks twice a day and renews when renewal is due. The edge takes the new certificate within a minute, with no restart (`docker compose logs edge-reload` shows `the edge reloaded and presents the helper's certificate`), and Postgres within five minutes. Until the first certificate exists the edge serves nothing and Postgres presents a short-lived self-signed one, so nobody can enroll. A helper that cannot issue retries an hour later; `docker compose logs dns-certs` carries lego's error.

Let's Encrypt no longer e-mails before a certificate expires, and nothing on a server installed from the bundle watches expiry, so check it yourself, on a schedule, from a machine outside the server:

```
echo | openssl s_client -connect <PUBLIC_PG_HOST>:443 -servername <PUBLIC_PG_HOST> 2>/dev/null |
openssl x509 -noout -enddate
echo | openssl s_client -connect <PUBLIC_PG_HOST>:5432 -starttls postgres 2>/dev/null | openssl
x509 -noout -enddate
```

Check both: the first is the certificate the edge holds for the name laptops dial for Postgres, the one `db-certs` copies, and Postgres takes a renewed one a few minutes after the edge. A 5432 date that falls behind 443's means `db-certs` did not take it (`docker compose logs db-certs` says why). Ask 443 for `PUBLIC_PG_HOST`, not `PUBLIC_HOST`: where the two differ, the `http` mode keeps one certificate per name, and behind a CDN `PUBLIC_HOST` shows the CDN's certificate.

To force a renewal in the `dns` mode, run `docker compose exec dns-certs touch /data/.renew-force` in the install folder; the helper logs `dns-certs: forced renewal done`. Let's Encrypt allows five duplicate certificates a week, so do it to test, not as a habit.

Recreate the stack with a full `docker compose up -d`, never the edge on its own (`up -d edge`): in the `dns` mode that leaves the reload helper behind, and no renewal reaches the edge until the next full `up -d`.

4.2.7 AWS Client VPN

Written from a real endpoint with three laptops and from AWS's documentation. Skip it if laptops reach the VPC another way. The VPN itself is yours to run; this is what the deployment needs from it.

The shape that was proved: an endpoint associated with one subnet, split tunnel, the VPC's resolver (the VPC's base address plus two) as its DNS server, one authorisation rule for the VPC's range, and mutual certificate authentication.

- **The client range** must not overlap the VPC, must be between /22 and /12, and cannot be changed later. Keep it, and the VPC, clear of `172.16.86.0/24`.

- **Split-tunnel routes** must carry the VPC's range, or at least the server's subnet and the resolver. With full tunnel every route goes through the VPC, which then needs internet egress (a NAT gateway, say) to the identity provider and the hosts the laptop reaches. Routes and authorisation rules are per range, not per port, so the server's security group is what limits the ports. Changing the endpoint's route table resets every client.
- **DNS works only where the client applies the pushed resolver.** Proved: the OpenVPN community client on Ubuntu with its `update-systemd-resolved` hook, which answered every query from the private zone. The same client without the hook never applied it: the name was not found, so nothing could push. Also proved, on 2026-11-03: the AWS-provided client 6.2.1 on macOS, over full tunnel, which applied the pushed resolver and answered the name from the private zone; whether the name then resolved through no other resolver was not checked. Check from a laptop on the VPN with `getent hosts <control-plane-host>` on Linux or `dscacheutil -q host -a name <control-plane-host>` on a Mac; the answer must be the private address.
- **Association subnets of their own**, holding nothing but the endpoint's interfaces: anything else in them gets the VPN's looser jail.
- **The server's security group:** with a subnet association, admit 443 and 5432 from the endpoint's security group (tested for 5432). A Transit Gateway endpoint cannot be referenced that way, so admit the client range instead (documentation only).

Laptops share addresses. With a subnet association Postgres sees each laptop as the address of one of the endpoint's interfaces in the association subnet: three laptops appeared as two addresses, and a laptop moved between them on reconnect. A laptop with a stale device password retries every 30 seconds, two refusals a minute, and never exits, so the strict jail would ban everyone behind its address within three minutes. Therefore:

- declare the whole association subnets, never single addresses: the VPN recreates its interfaces and a laptop moves between them, so only the subnet holds still. `scripts/check-env.sh` refuses a single address in `VPN_RANGES`, and that check is all `VPN_RANGES` does;
- type the same subnets by hand into `fleetview-postgres`'s `ignoreip`, and into the second jail `fleetview-postgres-vpn`, as The Postgres jail shows, with how to size it and find the stale laptop.

With a Transit Gateway each laptop keeps its own address (documentation only): leave `VPN_RANGES` blank, add no second jail, and the strict jail applies per laptop.

Host hardening is yours: fail2ban, the firewall and the operating system. The product ships no jail and no filter. The jail must act in `DOCKER-USER` (a default action on `INPUT` never sees traffic to a published container port), and its filter must be proved against the database container's journal before it is trusted (The Postgres jail).

4.2.8 The Postgres jail

Port 5432 answers everyone who can reach it, so a stranger can try passwords. `pg_hba.conf` lets only device roles reach a password check, and a fail2ban jail you set up on the server bans an address that keeps failing. The product ships neither the jail nor its filter; both are yours to write and keep.

- **Name it `fleetview-postgres`.** Ban in `DOCKER-USER` after five refused logins in ten minutes, for an hour, reading the database container's journal: the bundle's compose files log that container to journald, with the client address

on every line. Its container is `fleetview-db-1`, or `<project>-db-1` when `.env` sets another `COMPOSE_PROJECT_NAME`.

- **Put the stack's own subnet in `ignoreip`,** `172.16.86.0/24`, so the stack's own misconfiguration is never a ban, and no wider private range than the declared VPN ranges: a VPC or VPN client guessing passwords is outside.
- **Anchor the filter after the journal prefix.** fail2ban's systemd backend puts `<host> <identifier>[<pid>]:` before each entry, so a filter anchored at column 0 matches nothing. Match only a line with the Postgres timestamp right after that prefix: the refusal echoes the user name the client sent, and a name with a newline in it becomes indented continuation lines that could spell out a fake address.
- **Prove the filter** against the journal before you trust it, and again after a Postgres upgrade, which can reword a refusal: `sudo fail2ban-regex systemd-journal /etc/fail2ban/filter.d/fleetview-postgres.conf` must count every refusal as matched.

A ban lifts itself after an hour. A laptop with a stale device password bans itself: its pushes then fail on a refused connection instead of a refused password, and `winner push status` says they are failing. Its owner runs `winner login`, which enrolls it again with a fresh password. On the server:

```
sudo fail2ban-client status fleetview-postgres                  # banned addresses, counts
sudo fail2ban-client set fleetview-postgres unbanip <address>   # let it back in now
journalctl CONTAINER_NAME=fleetview-db-1 --since '-10min' -o cat | grep 'FATAL' | cat -v   # what was refused, and why
```

`cat -v` matters: the user name in a refusal is whatever the client sent. Twenty refusals across addresses in ten minutes is a scan; fewer, from one address, is a laptop looping on a stale password.

Laptops that share the VPN's addresses

Where a VPN hands every connected laptop the address of one of its interfaces (AWS Client VPN), one stale laptop would ban everyone behind its address. Declare those ranges in `VPN_RANGES`, where `scripts/check-env.sh` checks their shape and nothing reads them further. Then type the same ranges by hand into `fleetview-postgres`'s `ignoreip`, and add a second jail named exactly `fleetview-postgres-vpn` on the same filter and action that counts only those ranges, with a higher count and a short ban. Each stanza needs `enabled = true`, since fail2ban leaves a jail off by default; its filter, backend and action are yours to write:

```
[fleetview-postgres]
enabled  = true
ignoreip = 127.0.0.1/8 ::1 172.16.86.0/24 <the VPN ranges>
maxretry = 5
findtime = 10m
bantime  = 1h

[fleetview-postgres-vpn]
enabled  = true
# same filter, backend and action as fleetview-postgres; the address goes
# quoted as one argument, then the VPN ranges, and exit 0 (ignore) means outside
ignorecommand = python3 -c 'import ipaddress,sys; a=ipaddress.ip_address(sys.argv[1]);
sys.exit(any(a in ipaddress.ip_network(n) for n in sys.argv[2:]))' '<ip>' 10.45.2.0/24
10.45.3.0/24
maxretry = 60
findtime = 10m
bantime  = 10m
```

Sizing it. A stale laptop writes about two refusals a minute, so sixty in ten minutes tolerates two stale laptops behind one address and bans at three; raise the count for more. A stale laptop trips the jail again within minutes of each lift, so the ban is short on purpose. Anyone on the VPN, and any machine in its subnets, gets the higher count, which is why the association subnets should hold nothing else.

Finding the stale laptop. The address in a refusal is the VPN's; the role name is the device. Every device role is `dev_` and twelve hex digits. Check the name's shape and hand it to psql as a bound variable, never pasted into the SQL, in the install folder:

```
ROLE=dev_0123456789ab   # the role name from the refusal
[[ $ROLE =~ ^dev_[0-9a-f]{12}$ ]] && docker compose exec -T -e ROLE="$ROLE" db \
  sh -c 'psql -U "$POSTGRES_USER" -d "$POSTGRES_DB" -v role="$ROLE"' \
  <<<"SELECT u.email, d.machine_name, d.hostname, d.status FROM control.devices d JOIN
control.users u ON u.id = d.user_id WHERE d.pg_role = :'role'"
```

Its owner runs `winner login`. A role with no row, or a revoked one, is a laptop still pushing after its device was revoked; its owner signs in again or stops the push (`winner push stop`). Then let the address back in: `sudo fail2ban-client set fleetview-postgres-vpn unbanip <address>`.

4.2.9 A laptop off the VPN

In a private deployment a laptop off the VPN cannot reach the host: the name does not resolve (the public zone holds no record for it), or the private address does not answer.

- **A push fails to connect** and writes no Postgres log line, so no refusal is counted and nothing is banned.
- **The push service keeps running** and retries every 30 seconds. `winner push status` warns once two pushes in a row have failed.
- **A network change in the middle of a push**, such as joining or leaving the VPN, leaves the old connection open on the server, holding locks the laptop's next push waits on. In the 2026-11-03 rehearsal that stall lasted more than twelve minutes, until the connection was ended by hand. Since 0.13.1 the production Postgres's keepalive settings

(`docker-compose.prod.yml`) drop a dead connection in about two minutes, and the push carries on. Until then the push neither fails nor finishes, so `winner push status` warns of nothing.

- **Home shows the laptop Stale** after 24 hours without a push.
- **`winner login`, `winner update` and the hourly report** need the control plane, so they fail until the laptop is back; the scheduled update simply tries again at its next run.
- **Sessions wait on the laptop.** They stay in the laptop's local session store, and the next push that lands sends them. That is how it is designed; a long time off the VPN followed by the catch-up has not been run (Known limits).

4.2.10 What the server reaches

Outbound from the server, by mode. No licence server is contacted: a licence file is checked locally, on the host.

Host	Public	Private (`dns`)	For
Let's Encrypt's ACME API	yes, and ZeroSSL's, the authority Caddy falls back to	yes (Let's Encrypt only)	the certificate (Certificate renewals)
Inbound port 80 from the authority	yes	no	the HTTP challenge
The Route 53 API	no	yes	the challenge record
The public zone's own name servers, on UDP port 53	no	while `scripts/check-dns-zone.sh` runs; the helper itself needs none	the check's read-back of its test record
Instance metadata (link-local)	when Bedrock uses the instance role	yes	the instance role
Your identity provider (on Auth0, your tenant)	yes	yes	sign-in and tokens, and on Auth0 the Management API
Docker Hub, and the storage hosts it redirects downloads to	yes	yes, and for `scripts/check-dns-zone.sh` 's lookup of the helper's image	the stock images, pulled by digest; ours come from `images.tar` (Install the server)

Host	Public	Private (`dns`)	For
`ghcr.io` , GitHub's registry	when `trial` is in `COMPOSE_PROFILES`	same	the trial's sign-in provider (Dex), pulled by digest
The Ollama model registry (`registry.ollama.ai`), and the storage host it redirects downloads to (today `*.r2.cloudflarestorage.com`)	when the compose Ollama pulls a model	same	embeddings and a local model
`raw.githubusercontent.com` , `openrouter.ai`	yes	yes	model prices for cost figures, refreshed when the control plane sets up the organisation's schema (What leaves the server)
A hosted model endpoint	when one is set	same	Model and telemetry
Amazon Bedrock runtime, `bedrock-runtime.<region>.amazonaws.com`	when the provider is `bedrock`	same	Amazon Bedrock
Sentry, PostHog	when set	same	Telemetry
The jobs sweep's dead-man's switch (`FLEETVIEW_HEARTBEAT_URL`), and wherever you copy backups	when set	same	Jobs, Back up, restore and uninstall
Your distribution's package mirror	while preparing the host	same	Docker, `jq` , `rclone` , fail2ban

Setting up an organisation's schema refreshes model prices from GitHub and OpenRouter (the row above). Unconfirmed: whether the Dashboard (`pg serve`) does too. Its usage ping is off; no switch for the price refresh is known.

Laptops reach what What the laptop sends, and where lists. In a private deployment the control plane and Postgres go through the tunnel. With split tunnel the other hosts go over the laptop's own internet; with full tunnel they go through the VPC too, so its egress must reach them. Sign-in sends the browser to the identity provider; the laptop itself contacts it only on a control plane without its own sign-in, as that section says.

4.2.11 How the private path was proved

In four parts, none of which alone covers it.

- **A test rig with no cloud account.** In it the production `dns-certs` helper against a local test authority obtained one certificate for both names and Postgres presented it; a forced renewal reached Postgres with no laptop touched; three laptops behind one gateway pushed with `verify-full`, and Postgres logged one address for all three; a laptop with a stale password wrote two refusals a minute while the others kept pushing.
- **A real AWS account, the certificate:** lego issued from inside a private network through a real Route 53 public zone beside a same-name private zone, with the instance role and the one-record policy, against Let's Encrypt's staging authority.
- **A real AWS account, the laptops:** a real Client VPN endpoint and three Ubuntu laptops; the pushed resolver answered from the private zone, `verify-full` passed against the unmodified system trust store with a production Let's Encrypt certificate, and the strict jail's shared-address ban and the second jail's fix were both shown.
- **A real AWS account, a customer-shaped install** (2026-11-03): the bundle on a clean server in a private network, the shipped `dns-certs` helper issuing a production certificate through Route 53 with the instance role, and a Mac on a full-tunnel Client VPN with the AWS-provided client pushing every session over it.

4.2.12 Known limits

- **Two VPN set-ups were proved on AWS,** both a subnet association with mutual certificates: split tunnel with the OpenVPN community client on Ubuntu 24.04, and full tunnel with the AWS-provided client on macOS. Not proved: a Transit Gateway, SAML or federated sign-in, IPv6, and a second association in another zone.
- **Your VPN set-up is yours:** the product, its routes, authorisation, pushed DNS and client configuration.
- **The `dns` mode is Route 53 only.** On AWS, bare lego issued one name per run from staging, and a production certificate for one name. The shipped `dns-certs` helper issued a production certificate through Route 53 in the 2026-11-03 install. A two-name certificate ran only in the rig, against its test authority, and the rig has no Route 53, so the policy condition listing two challenge names has not run; scheduled renewal under Compose was not run on AWS.
- **The rig** trusts its test authority's root, and has no jail, journald, Client VPN, `winner` binary, enrolment or published port.
- **The jail** was proved only with a stand-in filter; the server's own is the operator's to write and prove.
- **Real laptop networks:** a Mac joining the VPN in the middle of a push was seen, and the stall it caused is bounded (above). Sleep and wake, captive portals and other Wi-Fi changes were not tested.
- **A long time off the VPN** followed by the catch-up was not run.

- **A name served both publicly and privately at once**, so a laptop pushes over either path, was not tested.

4.3 Installing the Winner AI server

This page installs Winner AI on a server you control, from the release's self-hosting bundle, `winner-ai-server-<X.Y.Z>.tar.gz`, in eight steps: check the requirements, verify the bundle's signature and check the server, load the images, fill in `.env`, start, check the result, upload the licence and switch the features on, and enrol the laptops and schedule backups. Start here: the install in order says what has to be ready before step 1, and who does it. `<X.Y.Z>` is the release you were given, and `<control-plane-host>` is the hostname your team will reach Winner AI at. Installing a later release over it, and going back, is Upgrade and roll back.

The bundle holds everything of ours the server runs, so the server needs no account with any registry of ours:

- `images.tar`: our compiled control-plane image and the Dashboard's and sign-in gate's images, all for linux/amd64;
- `caddy-linux-amd64`: the web server that brands the dashboard, as one binary;
- the compose files, `.env.example`, the configuration the stack mounts (`deploy/`), the installer `scripts/install-server.sh`, the host check it runs, `scripts/check-host.sh`, the `.env` check it runs, `scripts/check-env.sh`, the DNS zone check, `scripts/check-dns-zone.sh`, the command that fills `.env` from AWS Secrets Manager, `scripts/env-from-secrets.sh`, the backup and restore commands, `scripts/backup-server.sh` and `scripts/restore-server.sh` with the `scripts/backup-lib.sh` they share, the diagnostics command, `scripts/diagnostics-server.sh`, the upgrade command, `scripts/upgrade-server.sh`, and the rollback command, `scripts/rollback-server.sh`;
- the Auth0 post-login Action, `auth0-post-login-claims.action.js` in the `backend` folder, which the operator deploys to the tenant by hand (The post-login Action);
- `NOTICES.md`, the third-party notices, and this documentation as a PDF under `docs/`;
- `SHA256SUMS`, one line per file above, and `SHA256SUMS.sig`, its signature with the Winner AI release key.

Winner AI is delivered as a signed installer. The server's code is compiled, which makes it hard to read and to change; it does not make it secret. Page templates, database scripts and configuration files are readable. The licence agreement is what protects our code.

4.3.1 1. Check the requirements

The server. The first release supports the server on **linux/amd64 (x86-64) only**, on Ubuntu with Docker Engine 28.0 or later and the Docker Compose plugin 2.24.4 or later. arm64 is not supported. Nothing is built on the server: the web server is the bundle's binary, mounted into a stock Caddy image, and where that binary cannot run the install fails once the stack has started (below). SELinux in enforcing mode, rootless Docker, Docker Desktop and Podman have not been tried. The user who runs the install must be able to run `docker`.

Installing Docker. On a server without Docker, follow Install Docker Engine on Ubuntu, which covers the Ubuntu releases it lists. First remove Ubuntu's own packages, which conflict with Docker's and may be older than the minimums above:

```
sudo apt remove $(dpkg --get-selections docker.io docker-compose docker-compose-v2 docker-doc
docker-buildx podman-docker containerd runc | cut -f1)
```

Then add Docker's apt repository as that page describes, and install Docker Engine and the Compose plugin, the `docker compose` command the install uses:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-
plugin
```

On a server that already has Docker, `docker version` and `docker compose version` tell whether it meets the minimums; the host check refuses one that does not. Docker's post-installation steps say how to let a user run `docker` without `sudo`; membership of the `docker` group is root-equivalent on the server, so give it only to the operator who runs the install, never a shared or service account.

Nothing on this list needs checking by hand: `./scripts/check-host.sh`, which step 2 runs once the bundle is verified, checks the server, Docker, Compose, free disk, the ports, Docker Hub, the clock and `ssh-keygen`, and names every fault in one pass, one `check-host: FAIL:` line each; the script says what it checks and why. The installer runs the same check again, against `.env`, and stops before anything is installed on any fault, after the check's own lines:

```
install-server: FAIL: this host is missing what the install needs (above); nothing was loaded,
pulled or started
```

For the server and Docker, the check prints:

The server	**check-host: FAIL:**
is not Linux on x86-64	`Winner AI's server runs on linux/amd64 only; this host is <os>/<arch>. Nothing is built here to work around it.`
runs a Docker daemon for another platform	`Winner AI's server runs on linux/amd64 only; this host runs a Docker daemon on <os>/<arch>. Nothing is built here to work around it.`
has no Docker	`docker is not installed; install Docker Engine 28.0 or later with the Compose plugin 2.24.4 or later.`
has an older Docker Engine or Compose	`Docker Engine <version> is older than 28.0, ...` or `Docker Compose <version> is older than 2.24.4, ...`
reports a version the check cannot read	`cannot read the Docker Engine version ('<what it printed>'); ...` or `cannot read the Docker Compose version ('<what it printed>'); ...`
has Docker but not the Compose plugin	`the Docker Compose plugin is not installed (docker compose fails: <its reason>); install it, 2.24.4 or later.`

The server	`check-host: FAIL:`
cannot reach its Docker daemon	`cannot reach the Docker daemon as <user>: <Docker's reason>; run this and the install as a user who may run docker.`

Where the server passes those checks but still cannot run the mounted web server, the stack has already started when that shows, so the installer says so after the start:

```
install-server: this host cannot run the mounted web server (<the log line>); install on a
linux/amd64 Ubuntu server with Docker Engine 28.0 or later
install-server: FAIL: the stack started but the web server is not the bundle's; the install did
not succeed
install-server: the stack may still be running; to stop it, in <bundle folder> run:
  FLEETVIEW_VERSION=v<X.Y.Z> docker compose -f docker-compose.yml -f docker-compose.prod.yml -f
docker-compose.bundle.yml down
```

In the `dns` certificate mode the printed command starts with `COMPOSE_PROFILES=` and the profiles the installer ran with, `cert-dns` among them.

Tools. `ssh-keygen` from OpenSSH 8.1 or later, the first release with `ssh-keygen -Y verify` (`ssh -V` says which; Ubuntu 20.04 and later ship a new enough one, in the `openssh-client` package), `sha256sum` and `tar`. `openssl` for the secrets in step 4 and the certificate check in step 6, `curl` for the host check and step 6, `ss` (the `iproute2` package) for the host check's ports, and `jq` for the diagnostics command, which the host check does not look for.

Disk and memory. Where Docker keeps its data, the host check asks for 5 GB free, for the images and a first database, and 15 GB with the AI runner, whose image and models take the other 10 GB. Where Docker keeps its images in containerd's image store on another filesystem, it asks for 3 GB there (10 GB with the AI runner) and 2 GB where Docker keeps its data (5 GB with the AI runner). Leave room beside that for the unpacked bundle. No memory figure has been measured for this page.

Network.

- **A way to a publicly trusted certificate for `<control-plane-host>`**, in one of two certificate modes (Network says which to choose):
- **`http`, the default: a public DNS name pointing at the server, with ports 80 and 443 reachable from the internet.** The edge takes its certificate from a public certificate authority, which checks the name on port 80 or 443; the server needs outbound https to that authority.
- **`dns`, for a server on a private network that nothing on the internet can reach: a Route 53 public zone holding the name, and an instance role on the server** that may write the challenge record in it, with the policy in Private VPC: the certificate through DNS. The `dns-certs` helper takes the certificate through a TXT record in that zone, and the edge and Postgres present it. The server needs outbound https to the certificate authority and the Route 53 API, and no inbound port for the certificate.

Ports 80, 443 and 5432 must be free on the server; the host check says which is not. - **Port 5432 reachable from your team's laptops.** They push their sessions to Postgres over TLS and verify its certificate, the same one the edge holds, against their system trust store. - **Only 80, 443 and 5432 public.** The compose files publish nothing else on a public address, and nothing on the server checks that for you: the host check says only whether the ports are free, never who can reach them. A firewall on the server itself (`ufw` , say) does not stop a port Docker publishes, because Docker publishes past it; filter in front of the server or in Docker's `DOCKER-USER` chain. - **None of the server's networks, routes or VPN ranges may use `172.16.86.0/24` .** The stack's own network is pinned to it, and Postgres trusts that subnet as the stack itself, so a client reaching Postgres from an address inside it would pass as the stack. - **Outbound https to Docker Hub.** Our images come from `images.tar` and are never pulled from any registry. The server pulls the public images itself, from Docker Hub by digest, with no login:

When	Image
Always	the database (`pgvector/pgvector`), the edge and the web server base (`caddy`)
`local-llm` in `COMPOSE_PROFILES`	the AI runner (`ollama/ollama`), which then downloads its models from the Ollama registry
`CERTIFICATE_MODE=dns`	the certificate helper (`goacme/lego`)
`trial` in `COMPOSE_PROFILES`	the trial's sign-in provider (`ghcr.io/dexidp/dex` , from GitHub's registry, by digest); a trial runs no edge

Step 4 has the command that lists the exact references, each Docker Hub image pinned by digest, for the `.env` you write. Docker Hub limits pulls without a login per outbound address (step 5 says what a refused pull looks like). - **Outbound https to your identity provider's issuer.** The sign-in gate does not start without it. - **A clock within 5 seconds of the real time**, kept by time sync (NTP); the host check compares the clock with Docker Hub's.

An identity provider. Auth0, or any OpenID Connect provider such as JumpCloud. On another provider you need the gate's application there, a group for the Managers and the answers in Connect your identity provider, and step 4 takes its values from that page instead of the Auth0 ones. On Auth0 you need a tenant with the applications `.env` names, an organisation for your team, the post-login Action with its two secrets, and at least one manager. Setting up the tenant is not on this page; what it must allow is in Who may sign in, the Action is The post-login Action, and adding a manager is A manager. Connect your identity provider lists every answer the install asks of the provider.

4.3.2 2. Verify the bundle's signature and check the server

The installer checks the bundle's signature itself, against the release key it carries, on its own copy of `SHA256SUMS` , which is the only copy it reads after that, and before it loads any image; it refuses a bundle the release key did not sign. That check is only as good as the installer's own bytes, and a check an installer makes of itself proves nothing, so check them here first, by hand, with the stock `ssh-keygen` and the release key written out here, never a key taken from the bundle. The signature covers `SHA256SUMS` alone, so `sha256sum --check` then holds every file, the installer included, to it.

First, the release key. Write it out and print its fingerprint:

```
keys=$(mktemp -d)   # outside what the archive unpacks into
printf '%s\n' 'winner-release namespaces="release@getwinner.com" ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIMMDttCgimgvdz/6/0q4s5TC8RLUg783EsuPF4AGF0nW' > "$keys/allowed_signers"
awk '{print $3, $4}' "$keys/allowed_signers" | ssh-keygen -lf -
```

It must print `SHA256:EKI6sQilBqVmM1HZ/jNEamMFhDX7GcJWl0ykl69k2hs`. This page travels with the docs, and the docs may arrive by the same channel as the bundle, so **confirm that fingerprint with your Winner AI contact over a different channel before you go on**; where it differs, stop. The same public key is the one the laptop installer, which your control plane will serve at `/install.sh`, checks every `winner` download against.

Then the bundle, in the folder that holds the archive:

```
mkdir winner-install && tar --no-same-owner --no-same-permissions -xzf winner-ai-server-
<X.Y.Z>.tar.gz -C winner-install
cd winner-install/winner-ai-server-<X.Y.Z>
ssh-keygen -Y verify -f "$keys/allowed_signers" -I winner-release -n release@getwinner.com \
  -s SHA256SUMS.sig < SHA256SUMS &&
sha256sum --check --strict --quiet SHA256SUMS   # prints nothing when every file matches
```

`ssh-keygen` must print a line that starts `Good "release@getwinner.com" signature for winner-release`, and `sha256sum` must print nothing. Anything else, `Could not verify signature.` or a file `sha256sum` names, means the bundle is not the one we signed: **stop, and load nothing from it.** Ask your Winner AI contact for the bundle again.

Why the commands look like this:

- **Unpack into a new, empty folder**, so an archive that carries files beside the bundle folder cannot write over your own (`~/.ssh`, `~/.bashrc`).
- **The key file is written outside the unpacked folder**, so a link the archive planted cannot redirect that write.
- **`sha256sum --check` reads only the files `SHA256SUMS` names**, so the installer later refuses a bundle folder holding any other file (beyond `SHA256SUMS.sig` and `.env`), or any link.

Then check the server, in the same folder, once both checks pass:

```
./scripts/check-host.sh   # every fault on one pass; fix them and run it again
```

It must end `check-host: this host has what the install needs`. It runs only after the signature and `sha256sum` checks, because a script from a bundle nothing has vouched for could do anything with the rights of a user who can run `docker`; a missing or too-old `ssh-keygen` is named by the signature check itself. It only reads: it loads, pulls and starts nothing. With no `.env` yet it reads `.env.example`, which keeps the AI runner and the disk it needs; to go without it, leave `COMPOSE_PROFILES` blank in the `.env` you make in step 4. The installer runs the check again, against `.env`.

To start again from this step, as step 5 asks for some failures, first set the old folder aside, then repeat the step from the release key on:

```
cd <the folder that holds the archive>
(umask 077 && cp winner-install/winner-ai-server-<X.Y.Z>/.env winner-env.saved)   # only to keep
the .env you filled in
mv winner-install "winner-install.old.$(date +%s)"
```

With the bundle verified again, bring a kept `.env` back with `(umask 077 && cp --remove-destination ../../winner-env.saved .env)` in place of step 4's first command, then delete `winner-env.saved`.

4.3.3 3. Load the images

Nothing to run here: the installer loads `images.tar` itself in step 5, and refuses to go on when it does not hold the control plane for the bundle's release. It loads our control plane, `ghcr.io/getwinner-com/fleetview-backend:v<X.Y.Z>`, and the stock `ghcr.io/kenn-io/agentsview` and `quay.io/oauth2-proxy/oauth2-proxy` at the versions `docker-compose.yml` names. These images are never pulled from any registry; the Docker Hub images in step 1 are pulled in step 5.

`images.tar` is about a gigabyte, so loading it takes a while. To see that it loads before you fill in `.env`, you may run `docker load -i images.tar` in the bundle folder now, which prints one `Loaded image:` line per image; the installer still loads it again in step 5.

4.3.4 4. Fill in `.env`

```
(umask 077 && cp --remove-destination .env.example .env)
```

`--remove-destination` replaces anything the archive put at `.env` instead of writing through it, and the `umask` makes the copy readable by you alone before any secret goes into it. Then open `.env` and set the keys below. A line that starts with `# KEY=` is a comment: remove the `#` to set it. The comments in `.env.example` say what each key means, and Configuration lists every one.

Always required:

Key	Value
`POSTGRES_PASSWORD`	any strong value
`AUTH0_DOMAIN`	your Auth0 tenant's domain
`AUTH0_AGENT_CLIENT_ID`	the native device-code application a `winner` older than the control plane's own laptop sign-in uses (Configuration)

Key	Value
`AUTH0_PROXY_CLIENT_ID`, `AUTH0_PROXY_CLIENT_SECRET`	the sign-in gate's application
`AUTH0_TEST_ORG_ID`	the key of the installation's one organisation, which the control plane provisions and laptops enroll into: on Auth0, the organisation's id (`org_...`); on another provider, a name you choose. It needs a letter or a digit and must make a schema of at most 56 bytes. Choose it once: the schema is named from it at the first start, and a different key later makes another organisation (One organisation)
`FLEETVIEW_SERVE_DB_PASSWORD`	any strong value
`PGSERVE_AUTH_TOKEN`	any strong value
`OAUTH2_PROXY_COOKIE_SECRET`	32 random bytes, URL-safe base64

One of these, for who is a Manager:

Key	Value
`AUTH0_MGMT_CLIENT_ID`, `AUTH0_MGMT_CLIENT_SECRET`	the control plane's Management API application, both set, and roles come from Auth0
`FLEETVIEW_MANAGER_GROUP`	with both Management keys blank, `manager` : the Auth0 role the post-login Action sends (Configuration)
or `FLEETVIEW_MANAGER_EMAILS`	with both Management keys blank, the Managers' e-mail addresses, comma-separated (`ada@example.com,bob@example.com`), with `OAUTH2_PROXY_ALLOW_UNVERIFIED_EMAIL=false` (Configuration)

`scripts/check-env.sh` refuses one Management key without the other, and a blank pair with neither manager setting. The post-login Action refuses a sign-in through the gate's application from any Auth0 Organization but this installation's, or from none, once its two secrets, `AUTH0_PROXY_CLIENT_ID` and `AUTH0_TEST_ORG_ID`, hold the same values as this `.env`. Set them with the commands in The post-login Action when you deploy it: an Action deployed without them refuses every sign-in. Enable only this installation's Organization on the gate's application as well, the first check of the two: with the pair blank another Organization's Managers would otherwise become Managers here (Who may sign in).

Required because the server serves https:

Key	Value
`PUBLIC_BASE_URL`	`https://<control-plane-host>`, lower case, no trailing slash
`PUBLIC_HOST`	`<control-plane-host>`
`ACME_EMAIL`	the contact on the certificate authority account
`OAUTH2_PROXY_COOKIE_SECURE`	`true`
`PUBLIC_PG_HOST`	the hostname laptops dial for Postgres: `<control-plane-host>`, unless a CDN sits in front of it (Behind a CDN)
`PUBLIC_PG_SSLMODE`	`verify-full`
`FLEETVIEW_RELOAD_DB_PASSWORD`	any strong value
`OAUTH2_PROXY_OIDC_ISSUER_URL`	`https://<AUTH0_DOMAIN>/`, or another provider's issuer
`OIDC_END_SESSION_URL`	`https://<AUTH0_DOMAIN>/v2/logout`, or another provider's end-session address

On a provider other than Auth0, such as JumpCloud, the gate's settings, the manager group and the placeholders for the Auth0 keys it does not use are in Connect your identity provider (for JumpCloud, its own section). Leave both Management keys blank: with another issuer `scripts/check-env.sh` refuses them, because with them a person is found by e-mail in Auth0.

Strong values, one per key:

```
openssl rand -hex 32      # a password or token
openssl rand -base64 32 | tr -- '+/' '-_'   # OAUTH2_PROXY_COOKIE_SECRET: the gate reads only URL-safe base64
```

`POSTGRES_PORT` can stay as it is. A server installed from the bundle publishes Postgres on 5432 and tells laptops 5432, whatever it says; the template's `5433` is for a development machine and for a trial from the bundle.

Choose:

- `ORG_DISPLAY_NAME` : your organisation's name, which the page headers show. Set it before the first start: it is read once, when the organisation is first made, and a later change does not rename it. Left unset, it is `Winner AI` .
- `COMPOSE_PROFILES` : `local-llm` runs the AI runner on the server's CPU. Leave it blank (`COMPOSE_PROFILES=`) to run without it, once every model endpoint is a hosted one, and `ollama/ollama` is then never pulled. Choosing a model is Model and telemetry. To evaluate the bundle before you have a host name or a provider, add `trial` (below, "A trial from the bundle").
- `FLEETVIEW_FEATURES` : which of Prompt Coaching, Skills and Benchmarks run, as a JSON list. Commented out, as in `.env.example` , it is `[]` : all three pages show **Soon** and none of their model passes runs. Leave it so for the install; step 7 switches them on, in order: the model the `FLEETVIEW_LLM_*` keys name, its check, then the list, up to `FLEETVIEW_FEATURES=["coaching","skills","benchmarks"]` for all three. Skills also needs each laptop to run Ollama with `ollama pull nomic-embed-text` .
- `CERTIFICATE_MODE` : leave it unset, or `http` , for the edge to take its own certificate; `dns` for the `dns-certs` helper, with `AWS_HOSTED_ZONE_ID` , the **public** zone's id, and `AWS_REGION` . No AWS key goes in `.env` : the helper signs with the server's instance role. Before the first start in the `dns` mode, run `scripts/check-dns-zone.sh` in the bundle folder: it proves, with the same role, that the zone takes and serves the challenge record and that the certificate authority answers, since a first issue that fails costs one of five a week. The check also needs outbound port 53 to the zone's own name servers, which the helper does not; where only https may leave, its read-back fails although the issue could work.

Leave alone: `FLEETVIEW_TELEMETRY_SECRET` , which the installer writes (step 5); `FLEETVIEW_VERSION` and `COMPOSE_FILE` , which the installer writes once the install has proved itself; `FLEETVIEW_BACKEND_DIGEST` and `FLEETVIEW_CADDY_DIGEST` , which an install from the bundle leaves blank; and `DB_CERTS_SOURCE` and `DB_CERTS_SOURCE_DIR` , which the installer sets from `CERTIFICATE_MODE` . In the `dns` mode it also adds `cert-dns` to `COMPOSE_PROFILES` , the profile that starts the helper; it writes that profile and the certificate source into `.env` once the install has proved itself, so a later `docker compose up -d` keeps the mode.

Every other key, what it means and its default, is in Configuration.

Keep the secrets in AWS Secrets Manager

On AWS you can keep `.env` 's secrets in AWS Secrets Manager instead of typing them into the file. The stack still reads only `.env` : `scripts/env-from-secrets.sh` reads one secret with the server's instance role and writes its keys into `.env` , leaving every other line as it is. Use it when your secrets must live in Secrets Manager, or when you rotate them there.

The server needs the AWS CLI v2 and `jq` , neither of which a stock Ubuntu image has, and an instance role allowed to read the secret. A secret's ARN ends in six random characters, so name it in full or with `-??????` . Drop the `kms:Decrypt` statement when the secret uses Secrets Manager's default key; it is needed only for a customer-managed KMS key:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "arn:aws:secretsmanager:<region>:<account>:secret:<name>-??????"
    },
    {
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "arn:aws:kms:<region>:<account>:key/<key-id>",
      "Condition": {"StringEquals": {"kms:ViaService": "secretsmanager.<region>.amazonaws.com"}}
    }
  ]
}
```

Create the secret once, as one JSON object of `KEY: value` pairs, from a file only you can read, and then delete the file. Do this from a machine whose AWS credentials may create secrets: the server's role may only read it.

```
(umask 077 && cat > server-env.json)    # paste {"POSTGRES_PASSWORD": "...",
"PGSERVE_AUTH_TOKEN": "..."}, then Ctrl-D
aws secretsmanager create-secret --region <region> --name <name> --secret-string file://server-
env.json
rm -f server-env.json
```

Any key from the tables above may go in it. Leave out the keys under **Leave alone**, which the installer writes, and `COMPOSE_PROFILES`, to which the installer adds `cert-dns` in the `dns` mode: a later run would put the secret's value back over the installer's. Keys must be upper-case `.env` names, and values one line each, with no single quote and no backslash at the end; the command refuses anything else and says which key, without printing a value.

Then, in the bundle folder, before step 5:

```
./scripts/env-from-secrets.sh <secret-ARN>                          # the region comes from the ARN
./scripts/env-from-secrets.sh <name> --region <region>
```

It prints `env-from-secrets: wrote <N> keys from <secret> into .env`. With no `.env` yet it starts from `.env.example`, so set the keys not in the secret as above, in the same file. `.env` stays readable by its owner alone. Run it as the user who installs, as for the other steps; run as root, it keeps the file's owner.

An upgrade carries `.env` across to the new release, so it needs no new run. After a rotation, run the command again in the folder of the release that runs now, then restart with `docker compose up -d` there. The database keeps the password it was created with, so leave `POSTGRES_PASSWORD` in the secret as it is: a new value there leaves the stack unable to log in to the database.

The images this `.env` runs. With `.env` filled in, this lists the exact references, each Docker Hub image pinned by digest, the way the installer lists them:

```
FLEETVIEW_VERSION=v<X.Y.Z> docker compose -f docker-compose.yml -f docker-compose.prod.yml \
  -f docker-compose.bundle.yml config --images | sort -u
```

In the `dns` mode the installer adds the `cert-dns` profile, so put it in front: `COMPOSE_PROFILES=` set to the profiles in your `.env`, then `cert-dns`. With `.env.example`'s `local-llm`:

```
COMPOSE_PROFILES=local-llm,cert-dns FLEETVIEW_VERSION=v<X.Y.Z> docker compose -f docker-compose.yml \
  -f docker-compose.prod.yml -f docker-compose.bundle.yml config --images | sort -u
```

With `COMPOSE_PROFILES` blank in `.env`, it is `COMPOSE_PROFILES=cert-dns`. Without the profile the list leaves out `goacme/lego`.

The `ghcr.io/getwinner-com`, `ghcr.io/kenn-io` and `quay.io` lines are the images `images.tar` holds, which the installer never pulls; every other line is an image the server pulls in step 5, `ollama/ollama` only while `COMPOSE_PROFILES` names `local-llm`, and `goacme/lego` only in the `dns` mode. For a trial, leave `-f docker-compose.prod.yml` out, as a trial's `.env` does; its `ghcr.io/dexidp/dex` line is the one image pulled from GitHub's registry.

Close the editor before you go on. The installer refuses a bundle folder holding any file `SHA256SUMS` does not name, an editor's swap or backup file of `.env` included:

```
install-server: FAIL: the bundle folder holds files SHA256SUMS does not name: .env.swp Unpack the bundle again into an empty folder. Nothing was started.
```

For a swap or backup file of your own, deleting it is enough.

4.3.5 5. Start

From the bundle folder, as the user who may run `docker`:

```
./scripts/install-server.sh
```

The first start takes a while, and longer than any one wait: after it hashes every file and loads `images.tar`, it waits in turn for the web server, the control plane and the edge, and each of those waits may last up to thirty minutes. Do not interrupt it. It first checks the signature, then runs the host check again, once its bytes match their signed line, and its `check-host:` lines come between the first two lines below:

```
install-server: SHA256SUMS is signed by the release key
(SHA256:EKI6sQilBqVmM1HZ/jNEamMFhDX7GcJWl0ykl69k2hs)
install-server: installing v<X.Y.Z> from <bundle folder>
install-server: checking every file against SHA256SUMS (images.tar takes a while)
install-server: generated FLEETVIEW_TELEMETRY_SECRET in .env (never shown)
install-server: loading our images from images.tar
install-server: pulling <image>@sha256:<digest> from Docker Hub
install-server: starting the stack
install-server: the web server runs the bundle's binary
install-server: the edge serves ports 80 and 443
install-server: the org schemas hold only what agentsview creates
install-server: installed v<X.Y.Z>
```

with one `pulling` line per Docker Hub image. In the `dns` mode it also prints, around those lines,

```
install-server: the dns certificate mode: the dns-certs helper takes the certificate through the
public zone, and the edge and Postgres present it
install-server: the certificate helper, dns-certs, holds a certificate
install-server: db-certs reads the helper's certificate; Postgres presents it from db-certs' next
pass, within five minutes
```

and pulls the helper's `goacme/lego` image, which the `http` mode never does. The first issue takes a minute or more, and the edge serves nothing until it has the certificate, so the installer waits for the helper first. Postgres takes the certificate on `db-certs` ' next pass, up to five minutes later; until then it presents a self-signed one and no laptop can push. It prints `install-server: caddy-linux-amd64 set to mode 0755` before the `checking` line when the unpacked binary lost its mode.

The telemetry secret. Once `scripts/check-env.sh` accepts `.env` , and before anything starts, the installer writes a random `FLEETVIEW_TELEMETRY_SECRET` into `.env` when none is set. It never prints the value and never changes one already there, since a new secret would cut the deployment's past telemetry off from its future; then it prints `kept the FLEETVIEW_TELEMETRY_SECRET already in .env` instead, with a `WARNING:` when the kept value is shorter than 32 characters. `.env` is the secret's one record: keep it there, never beside the backups. What it turns on is Model and telemetry.

Once it prints `installed` , `.env` names the compose files and the release, so a later `docker compose` command in this folder (`docker compose ps` , `docker compose logs` , `docker compose up -d`) runs the same stack and still pulls none of our images.

When it fails, it prints `install-server: FAIL:` and the reason, and exits non-zero. Up to the `starting the stack` line nothing is running; fix what it names and run it again.

What it says	What to do
`this host is missing what the install needs (above); nothing was loaded, pulled or started`	Fix each `check-host: FAIL:` line above it (step 1), and run it again
`no SHA256SUMS in <folder>; ...` or `no SHA256SUMS.sig in <folder>; ...`	Run the bundle's own `./scripts/install-server.sh` , from the bundle folder, not a copy of it elsewhere. If the folder is the bundle's and still has no `SHA256SUMS.sig` , nothing vouches for it: ask your Winner AI contact for the bundle again
`SHA256SUMS does not match its signature in SHA256SUMS.sig: ...` , `SHA256SUMS.sig is signed by another key (...)` , `SHA256SUMS.sig was made in the licence namespace ...` , or `SHA256SUMS ... is a link or not a regular file`	The bundle is not one Winner AI signed, or it changed after it was signed; nothing was loaded. Start again from step 2 with a new, empty folder, and if step 2's `ssh-keygen` line fails too, ask your Winner AI contact for the bundle again

What it says	What to do
`ssh-keygen is not installed, so the bundle's signature cannot be checked; ...` or `sha256sum is not installed, ...`	Install the OpenSSH client or coreutils (step 1's tools), and run it again
`SHA256SUMS does not name one release (no single docs/winner-ai-docs-X.Y.Z.pdf line)`, `SHA256SUMS has no line for caddy-linux-amd64`, `SHA256SUMS has no line for scripts/check-host.sh`, or `images.tar holds no ghcr.io/getwinner-com/fleetview-backend:v<X.Y.Z>; this bundle is not release v<X.Y.Z>`	The signed bundle itself is not a whole release. Ask your Winner AI contact for a new bundle
`the bundle folder holds files SHA256SUMS does not name: ...`	Delete a file of your own it names (step 4); for any other, start again from step 2 with a new, empty folder
`the bundle folder holds links or other non-files, which no bundle ships: ...`, `scripts/check-host.sh does not match its line in SHA256SUMS; ...`, `scripts/check-host.sh is missing or not a regular file`, `these files do not match SHA256SUMS: ...`, `caddy-linux-amd64 does not match its line in SHA256SUMS; its bytes changed`, `caddy-linux-amd64 is missing or not a regular file`, or `the copy of caddy-linux-amd64 changed while it was made`	The bundle folder changed after you verified it. Start again from step 2 with a new, empty folder
`caddy-linux-amd64 is built for <architecture>, not x86-64; this host cannot run it. Nothing was started.`, or another `caddy-linux-amd64 is ...; this host cannot run it` such as `is not an ELF executable`	Run step 2's `sha256sum --check` line again in the bundle folder. If it names `caddy-linux-amd64`, the folder changed: start again from step 2 with a new, empty folder. If it prints nothing, the signed binary itself is wrong: ask your Winner AI contact for a new bundle
`no .env in <folder>. Run: cp .env.example .env  then fill it in`	Do step 4
`scripts/check-env.sh refused .env (above); nothing was started`, after the check's own message, such as `Missing values in .env:` and the keys	Set what the message names (step 4), and run it again

What it says	What to do
`.env sets FLEETVIEW_TELEMETRY_SECRET on <n> lines; keep one, ...`	Delete every line but the one holding the value the deployment already uses, and run it again
`FLEETVIEW_TELEMETRY_SECRET is set in this shell but not in .env; ...`	Put the value in `.env`, its one record, or `unset FLEETVIEW_TELEMETRY_SECRET` in the shell to have one generated, and run it again
`compose cannot read its files with this .env: <compose's last line>`	Compose's own words name the key; usually a required value is missing or a line is malformed. Fix it in `.env` (step 4)
`docker load could not load images.tar: <Docker's output>`	Docker's words give the reason, most often no space left for the images. Free the space, and run it again
`<image> is neither in the bundle nor a pinned Docker Hub image; nothing was started`	Something in `.env` changed which image a service runs, or the bundle is wrong. Check the keys step 4 says to leave alone, list the images with step 4's last command, and if the line still shows, ask your Winner AI contact
`.env's COMPOSE_FILE names docker-compose.prod.yml, so this folder holds a production install; ...`	A trial was asked for in a folder that holds a production install. Unpack the bundle into a folder of its own for the trial
`compose project <name> has containers of ... from a production install, ...`	A production install shares the trial's compose project: its edge would go on serving ports 80 and 443, and compose names the database volume by the project, so the trial would open the production database. Give the trial another `COMPOSE_PROJECT_NAME` in `.env`; stopping the production install does not help, as its volume stays
`compose project <name> has volumes ... which a trial does not define: ...`	A production install's volumes, its database among them, share the trial's compose project; `docker compose down` keeps them. Give the trial another `COMPOSE_PROJECT_NAME` in `.env`

What it says	What to do
`could not pull <image> from Docker Hub: <Docker's last line>. Nothing was started.`	Docker Hub refused the pull or could not be reached; the reason is in Docker's own words. A refusal for the rate limit means the server's outbound address used up Docker Hub's allowance for pulls without a login (100 per IPv4 address, or IPv6 /64, every 6 hours, at the time of writing), which a server sharing its address with other Docker users can reach. Wait, and run the installer again
`this bundle is v<X.Y.Z>, older than v<newer> which this server runs; migrations do not reverse, ...`	A newer release already runs on this server's database, and an older one would start on tables it does not know. `scripts/rollback-server.sh` in the folder of v goes back to the release the last upgrade to v replaced, if one did; for an older release, name its folder with `--bundle` and the backup taken before the upgrade from it with `--backup` (Going back to the previous release); after an upgrade that failed, run the restore command it printed instead. The installer sees only a stack that runs: over a stopped one it does not know which release last ran, and an older bundle is not refused

A failure the installer did not foresee reads `unexpected error at line <n> (<command>)`; quote it to your Winner AI contact.

After the start, from the `starting the stack` line on, a failure may leave some or all of the stack running, so the installer prints the command that stops it (step 1 shows its shape, and in the `dns` mode it starts with `COMPOSE_PROFILES=`), because `.env` does not yet name the compose files or the release. A `docker compose up` failure can come before any service starts; the stop command is then harmless. Fix what it names and run the installer again, or stop the stack with that command. The failures there:

- `docker compose up failed (above)`, after compose's own output, which names the service and the reason.
- `could not remove dns-certs and edge-reload in the http certificate mode`: an earlier `dns`-mode install left the certificate helper behind, and the installer could not remove it. Run the installer again; if the line stays, send it to your Winner AI contact.
- `the web server container was not created; the install did not succeed`.
- `the stack started but the web server is not the bundle's; the install did not succeed`, after a line saying which: the mount over `/usr/bin/caddy` is missing, the binary differs, it lacks the brand plugin, the web server keeps restarting, or the host cannot run it (step 1).
- `/health did not answer ok within 1800s (it said '...'); the install did not succeed`.
- `/internal/version reports '...', not v<X.Y.Z>; the install did not succeed`.

- In a trial, `the trial's sign-in provider, Dex, was not created; the install did not succeed`, or `the trial's sign-in provider, Dex, is not running (...)`, ending with Dex's last log line, which says why.
- `the edge container was not created; the install did not succeed`.
- `the edge, which serves ports 80 and 443, is not running and healthy (...)`, ending with the edge's last log line. In the `http` mode, check that `<control-plane-host>` resolves to the server and that ports 80 and 443 reach it.
- `the certificate helper, dns-certs, has no certificate (...)`, ending with the helper's last log line, or `the certificate helper, dns-certs, was not created`: in the `dns` mode the helper could not get the certificate, and nothing serves until it does. `docker compose logs dns-certs` says why; `scripts/check-dns-zone.sh` checks the zone, the instance role and the certificate authority. The helper retries on its own an hour later; once it holds a certificate, run the installer again.
- `edge-reload ... is not running`, `edge-reload does not run in the edge's PID namespace`, or `db-certs does not read the helper's certificates`: the `dns` mode's wiring did not come up as compose was told. Run the installer again; if the line stays, send it to your Winner AI contact.
- `an org schema holds code stock agentsview does not create`, with the findings: Code in an org schema that agentsview did not create says how to clear it.
- `the org-schema code audit did not run (exit <n>):`, with the audit's last lines, which say why.

4.3.6 6. Check the installation

Health, from any machine:

```
curl -sS https://<control-plane-host>/health      # {"status":"ok","database":"ok"}
curl -sSI http://<control-plane-host>/ | head -1  # 308, to https
```

Every service up, in the bundle folder; each with a health check reads `healthy`:

```
docker compose ps
```

Postgres presents a public certificate, which is what laptops verify:

```
echo | openssl s_client -connect <PUBLIC_PG_HOST>:5432 -starttls postgres \
  -verify_return_error -verify_hostname <PUBLIC_PG_HOST> 2>/dev/null | openssl x509 -noout -
issuer -dates
```

`<PUBLIC_PG_HOST>` is the value of `PUBLIC_PG_HOST` in `.env`, the name laptops dial (step 4), usually `<control-plane-host>`. The certificate it prints must be issued by a public certificate authority and be within its validity dates. The check verifies that the certificate chains to an authority this machine trusts and was issued for that name, as a laptop does; where either fails, it prints `Unable to load certificate` instead and exits non-zero.

Sign in. From a browser, open `https://<control-plane-host>` and sign in as a manager of the organisation `AUTH0_TEST_ORG_ID` names (A manager). Reach Home and the Dashboard, then sign out; sign-out must end both the gate's session and Auth0's.

Enroll the first laptop. On a laptop of someone in the organisation, install the client from your server and sign in. A new laptop sends no session until its developer chooses which repos it sends (Choose which repos send): tick at least one in the list `winner login` shows, or the dashboard stays empty (a trial has lines of its own, in A trial from the bundle):

```
curl -fsSL --proto '=https' --proto-redir '=https' https://<control-plane-host>/install.sh | bash
winner login --control-url https://<control-plane-host>
winner push status
```

`winner push status` must report the background push running (Laptop client guide covers the laptop side). Then, in Admin (`/admin`), the laptop is on the roster; within the push interval it shows a session count, "· last push `<time>`", "· last change `<time>`" or "· push seen, time unknown", which is what proves a push arrived. "· awaiting first push" alone does not.

When the laptop's push fails to connect to Postgres, port 5432 must reach the server from the laptop's network (step 1).

Steps 7 and 8 finish the install. When something goes wrong from here on, Troubleshooting is the symptom index.

4.3.7 7. Upload the licence and switch the features on

Prompt Coaching, Skills and Benchmarks are all off by default, because each sends prompt text to the model unredacted (What each feature sends to the model). Switch them on in this order:

1. **Upload the licence.** A Manager signs in, opens Settings and uploads the licence file Winner AI sent (Upload it). Until then the deployment runs on 30 days of grace from its first start; a listed feature is live only while the licence is valid or in its grace, and once it lapses all three leave the sidebar.
2. **Point the stack at the model.** With `local-llm` the default `FLEETVIEW_LLM_*` values call the server's own AI runner, slowly on CPU, and there is nothing to set. For a hosted model or Amazon Bedrock, set the model keys in `.env` and recreate the services that read them, as Point the stack at a hosted model says (Amazon Bedrock for Bedrock). Set the Benchmarks Routes, `FLEETVIEW_BENCHMARK_ROUTES` , to model ids your provider serves; on Amazon Bedrock, to Bedrock model or inference-profile ids (Benchmarks Routes on Bedrock).
3. **Check the model**, in the install folder:

   ```
   docker compose exec -T jobs uv run --no-dev fleetview-model-check
   ```

   It ends with a verdict; Check the model before Prompt Coaching says how to read it, and how to give a slow local model more time.
4. **List the features** in `.env` : `skills` and `benchmarks` , as `FLEETVIEW_FEATURES=["skills","benchmarks"]` , and `coaching` only once the model check passes, as `FLEETVIEW_FEATURES=["coaching","skills","benchmarks"]` . A team can switch a listed feature off on Settings, never on.

5. **Recreate the services that read it**, in the install folder:

```
docker compose up -d backend jobs
```

6. **Check.** The pages fill once laptops have pushed sessions (step 8) and a jobs sweep has run over them: the recreated jobs service sweeps as it starts, then every 15 minutes by default. Skills and Benchmarks show their first results after the next sweep that follows a push. Switch Prompt Coaching on says how to read a sweep's result.

4.3.8 8. Enrol the laptops and schedule backups

The laptops. Each developer installs the client from your server, signs in, and ticks the repos it may send in the list `winner login` shows:

```
curl -fsSL --proto '=https' --proto-redir '=https' https://<control-plane-host>/install.sh | bash
winner login --control-url https://<control-plane-host>
```

Hand them the Laptop client guide: it covers macOS and Linux, choosing repos and what to expect day to day. Skills also needs vectors from each laptop: each laptop needs Ollama, which Winner AI does not install, with `ollama pull nomic-embed-text` (Embeddings).

Backups. Nothing on a server installed from the bundle schedules a backup or copies one off the server. Schedule `scripts/backup-server.sh` and copy each backup to storage off the server, as Schedule it and copy it off the server shows, then rehearse a restore once (Rehearse a restore). Nothing watches the certificate's expiry either: Certificate renewals has the check to schedule.

4.3.9 A trial from the bundle

To evaluate the bundle before you have a host name or a provider, install it as a trial: put `trial` in `COMPOSE_PROFILES`, uncomment the trial block of `.env.example` (Connect your identity provider, "A trial with the bundled Dex"), and leave `PUBLIC_BASE_URL` blank or `http://localhost:4180`. The installer reads the profiles from `.env`, never from your shell, and then stacks `docker-compose.bundle.yml` on the base file alone, without the production overlay: no edge, nothing published beyond loopback, the gate at `http://localhost:4180` on that machine only, and no edge check at the end. The host check, reading the same `.env`, then needs ports 4180 and `POSTGRES_PORT` (5433 unless set) free instead of 80, 443 and 5432. Step 2's run of it comes before there is a `.env`, so it reads `.env.example` and checks the production ports: once `.env` names the trial, run `./scripts/check-host.sh` again, and take its answer on ports over step 2's. The installer runs it against `.env` either way. `scripts/check-env.sh` refuses a trial with an https base URL or with `docker-compose.prod.yml` in `COMPOSE_FILE`, and the installer refuses a trial in a folder that holds a production install (step 5's table). It removes nothing. Compose names the database volume by the project, not the folder, so a trial runs as its own project: when `.env` names none, the installer writes `COMPOSE_PROJECT_NAME=fleetview-trial` into it, and a production install made later on the same machine, under the default project `fleetview`, never shares the trial's database. The project comes from `.env` alone, never from your shell, for a trial and a production install alike. Unpack the trial into a folder of its own, and leave its `COMPOSE_PROJECT_NAME` as the installer wrote it. With `local-llm` kept beside `trial`, as `.env.example` suggests, a trial runs the AI runner too, so listing the three pages in `FLEETVIEW_FEATURES` (step 7) turns them on; they fill once laptops push sessions and a jobs sweep has run them on that model, and Skills also needs vectors: each laptop needs Ollama and `ollama pull nomic-embed-text` (Embeddings). Left out, they show Soon. A trial is for evaluation only; a production install starts from a fresh `.env`.

Enroll a laptop against a trial. The production line in step 6 refuses a trial: its `--proto '=https'` flags reject the gate's plain `http://localhost:4180`. A trial installs the client and signs in with:

```
curl -fsS --noproxy localhost http://localhost:4180/install.sh | bash
winner login --control-url http://localhost:4180
```

Dropping the flags is safe only because the address is loopback: the installer still checks `winner` against the release signature, and `winner login` accepts plain http to a loopback address and nothing else. Never drop them for a production host. `--noproxy localhost` keeps a laptop's `http_proxy` from carrying the script, which no signature covers; the installer's own downloads follow the shell's proxy settings, so a laptop that sets one also needs `localhost` in `no_proxy`. The laptop must be the trial machine, or reach it through a tunnel that forwards both the gate and the `POSTGRES_PORT` in the trial's `.env` (5433 unless set) at the same numbers, both free on the laptop, since Dex and the laptop's push know only `localhost`:

```
ssh -N -L 4180:localhost:4180 -L <POSTGRES_PORT>:localhost:<POSTGRES_PORT> <trial-machine>
```

Keep the tunnel up while the laptop is in use: its background push and update timer go through it, and with it closed their pushes fail quietly.

Open the approval page `winner login` prints in that laptop's browser and sign in as one of the trial's Dex accounts (Connect your identity provider, "A trial with the bundled Dex"). Leave `PUBLIC_PG_HOST` and `PUBLIC_PG_SSLMODE` blank in a trial's `.env`: laptops are then told `localhost` with no TLS, which loopback is exempt from.

4.3.10 Collecting a diagnostics file

A support case starts with one redacted file from the server. Run the command as root from the folder of the release that runs now: the one you ran `./scripts/install-server.sh` in, or after an upgrade the folder Upgrade and roll back left running. Unlike the server's other commands (below), it runs with `sudo` and refuses to run without it. It needs `jq` on the server (`apt install jq` or `dnf install jq`):

```
sudo ./scripts/diagnostics-server.sh                # 200 lines a service, into your home
directory
sudo ./scripts/diagnostics-server.sh --lines 1000   # more lines
```

It prints the path of the `0600` file it wrote and sends nothing. It writes into your home directory, never the release folder: a file there that `SHA256SUMS` does not name stops the next install or a rollback from that folder. `--out DIR` writes elsewhere. Run through `sudo` from your own account, it takes a directory only you (and your own group) can write, or one with the sticky bit such as `/tmp` ; from a root shell, only a directory only root can write, so not `/tmp` . It opens no connection and reads no organisation's data, so it works with the database down.

The file holds the build version and commit, each service's image and state, every setting on the Settings ledger with the layer it came from (a key only as set or not set, an endpoint without credentials or query), and each service's last log lines. Those lines lose organisation and user ids, device roles, client addresses, UUIDs, e-mail addresses, URL query strings, the row data and statement text a Postgres error can carry, and the value of every secret setting; a team's own model key is never read. Any other secret in a log line is caught only where a pattern names it: the masking errs toward too much, but it cannot prove a line holds nothing private, so read the file before you attach it.

4.3.11 Commands on the other pages

The other pages write the server's commands for a server installed from the bundle. Read them this way:

- Run the server's commands in `<install>` , the running release's folder (Back up, restore and uninstall says which), as the user who owns `<install>/.env` and can run `docker` , without `sudo` . The installer leaves `.env` readable by that user alone, and compose reads the list of compose files from it, so compose run by another user who can run `docker` fails or runs without the bundle's files. Read `.env` there with plain `grep` .
- Once the installer has printed `installed` , `docker compose` needs no list of compose files and no `COMPOSE_FILE=` in front of it: `COMPOSE_FILE` in `.env` already names the bundle's. A command you meet written for another layout, run in `/srv/fleetview` or with `-f docker-compose.yml -f docker-compose.prod.yml` , is plain `docker compose` in `<install>` : that pair would leave out `docker-compose.bundle.yml` , which keeps compose from pulling an image of ours.
- There is no `make` . `make check-env` is `./scripts/check-env.sh` , run in the bundle folder, before the install as well as after it: in an unpacked bundle it does not ask for the values the installer writes. `scripts/install-server.sh` runs the same check itself (step 5). `make up` is `docker compose up -d` . Every other `make` target, `make eval` included, needs a checkout of the release, not the server.
- Nothing schedules a backup or watches the certificate's expiry on a bundle install unless you set it up (Back up, restore and uninstall, Certificate renewals). The bundle ships no Postgres jail: its commands apply once you set one up (The Postgres jail).

4.3.12 Later releases

To install a later release over this one, or go back to the one an upgrade replaced, see Upgrade and roll back.

4.4 Configuration reference

A deployment is configured by `.env` beside the compose files, started from `.env.example`. This page lists every variable it sets, and the few that are not `.env` keys (Backups and host monitoring). After them come the team settings a manager changes on Settings, and the options of the `winner` client on a laptop.

- **Default** is the value the stack runs with when the variable is unset; *none* means unset leaves it blank or off.
- **Required**: *Required* means `scripts/check-env.sh`, which the bundle installer and the upgrade command run before they start anything, refuses an `.env` without it. *Required in production* means it is required once `PUBLIC_BASE_URL` is https, which is what makes a deployment a production one. *Required outside the `trial` profile* means a trial `.env` may leave it blank. *Required in the `dns` certificate mode* means `scripts/check-env.sh` refuses a `CERTIFICATE_MODE=dns` `.env` without it. *Required with the `bedrock` provider* means `scripts/check-env.sh` refuses a `FLEETVIEW_LLM_PROVIDER=bedrock` `.env` without it. *Written by the bundle installer* means it is never set by hand: on a server installed from the bundle the installer writes it once the install has proved itself, and the upgrade and rollback commands after it. *Not read by the stack* means nothing in a deployment reads it.
- **Layers** are the ones that can supply the value: *release* (a default ships with the release; Settings shows it as `release <version>`), *operator* (whoever runs the deployment sets it in these files; Settings shows `operator`) and *team* (a manager overrides it for their organisation on Settings, which shows `your team`). A team value wins over an operator one, which wins over the release default. Settings shows `operator` for any value compose passes to the stack, with a default of its own (`${NAME:-default}` in `docker-compose.yml`, such as `FLEETVIEW_FEATURES` or the `FLEETVIEW_LLM_*` values) or blank (`${NAME:-}`), even when `.env` leaves it unset: the stack receives the variable either way. A blank `FLEETVIEW_LLM_REGION` or `FLEETVIEW_SENTRY_TRACES_SAMPLE_RATE` is the exception, shown as `release`. For the rows with a compose default, *release* in Layers is the default the stack runs with, not a word Settings shows.

On an installation from the bundle, `.env` is `<install>/.env`, in the folder of the release that runs now. Commands on the other pages says how the other pages' server commands read there.

4.4.1 Global

Variable	Meaning	Default	Required	Layers
`COMPOSE_PROFILES`	Compose profiles switched on. `local-llm` runs the Ollama container, which serves the dashboard's query embeddings and, by default, the chat models. `scripts/check-env.sh` refuses a blank value, except on an install from the bundle, where blank turns the AI runner off. `cert-dns` starts the certificate helper of the `dns` certificate mode, and the bundle installer writes it in	none; the example ships `local-llm`	Required, except on an install from the bundle	operator

Variable	Meaning	Default	Required	Layers
	that mode. `trial` runs the bundled Dex a trial signs in through, as does `*`, which compose reads as every profile; `scripts/check-env.sh` and the control plane refuse either on an https base URL. Under the trial `scripts/check-env.sh` requires the trial block of `.env.example` as written and a base URL that is blank or `http://localhost:4180`, the one address Dex knows, and refuses the production overlay in `COMPOSE_FILE`; an install from the bundle then runs the trial on loopback, with no edge.			
`FLEETVIEW_FEATURES`	Product surfaces switched on, as a JSON list of nav slugs (`coaching`, `skills`, `benchmarks`). Anything not listed shows as Soon, and the list also selects which jobs passes run.	`[]`	Optional	release, operator, team (a team can only switch a feature off)
`FLEETVIEW_VERSION`	The release the server runs: after an install from the bundle, the tag of the images it loaded, which the installer and the upgrade write; a rollback reads the older release's own `.env`.	none	Required in production; written by the bundle installer, so an unpacked bundle's check before the install does not ask for it	operator
`FLEETVIEW_BACKEND_DIGEST`	Blank on a server installed from the bundle, which runs the backend image it loaded; the upgrade command blanks it. Only a server that pulls Winner AI's images from its registry, which a bundle install never does, sets it.	none	Required in production, except on an install from the bundle	operator
`FLEETVIEW_CADDY_DIGEST`	The same for the caddy image: blank on a server installed from the bundle.	none	Required in production, except on an	operator

Variable	Meaning	Default	Required	Layers
			install from the bundle	
`COMPOSE_FILE`	The compose files a server installed from the bundle stacks: the base file, the production overlay and `docker-compose.bundle.yml` (a trial leaves out the production overlay), so a later `docker compose up -d` in the bundle folder never pulls one of our images. `scripts/check-env.sh` reads it to tell an install from the bundle, which needs no digests.	none	Written by the bundle installer	operator
`FLEETVIEW_PIN`	Not read on a server installed from the bundle, where you choose each upgrade and rollback yourself (Upgrade and roll back). Leave it blank.	none	Optional	operator

4.4.2 Database

Variable	Meaning	Default	Required	Layers
`POSTGRES_USER`	The database superuser.	`fleetview`	Optional	release, operator
`POSTGRES_PASSWORD`	Its password; any strong value.	none	Required	operator
`POSTGRES_DB`	The database name. Keep `fleetview`: enrollment hands laptops `fleetview` whatever this says, because `FLEETVIEW_PG_DATABASE` is not forwarded from it.	`fleetview`	Optional	release, operator
`POSTGRES_PORT`	Outside production, the loopback port Postgres is published on and the port enrollment hands laptops: a development machine and a trial. A production server, a bundle install	`5433`	Optional	release, operator

Variable	Meaning	Default	Required	Layers
	included, publishes Postgres on `5432` and hands laptops `5432` whatever this says.			
`PUBLIC_PG_HOST`	The name laptops dial for Postgres. Give it its own DNS-only name when a CDN fronts `PUBLIC_HOST`; never an IP address, because laptops verify the certificate against it.	`localhost`; under the production overlay the edge and the certificate sidecar fall back to `PUBLIC_HOST`, but enrollment still hands laptops `localhost`	Required in production	release, operator
`PUBLIC_PG_SSLMODE`	The sslmode written into laptops' connection strings; `verify-full` in production.	none	Required in production	operator
`VPN_RANGES`	Where laptops reach Postgres through a VPN and share its addresses: the VPN's whole private subnets, comma-separated, never single addresses and clear of the compose subnet. On a server installed from the bundle it is a declaration: `scripts/check-env.sh` checks only its shape, and nothing else reads it. Type the same ranges into your own fail2ban jails by hand (The Postgres jail).	none	Optional	operator
`FLEETVIEW_SERVE_DB_PASSWORD`	Password of the Org's read-only serve role, which its dashboard reads with.	none	Required	operator
`FLEETVIEW_RELOAD_DB_PASSWORD`	Password of the role the certificate sidecar uses to reload Postgres after a renewal.	none	Required in production	operator

4.4.3 Authentication

Variable	Meaning	Default	Required	Layers
`AUTH0_DOMAIN`	The Auth0 tenant domain. An installation signing in through another provider names no tenant, but `scripts/check-env.sh` still requires a value: a placeholder that resolves nowhere, such as `unused.invalid`.	none	Required outside the `trial` profile	operator
`AUTH0_AGENT_CLIENT_ID`	The Auth0 native device-code application a `winner` older than the control plane's own laptop login signed in with. It is public: the control plane serves it beside that login, and only an older `winner` uses it. A current `winner` signs in through the control plane's own login on every installation, Auth0 included. On another provider, a placeholder such as `unused`.	none	Required outside the `trial` profile	operator
`AUTH0_PROXY_CLIENT_ID`	The sign-in gate's application at the provider `OAUTH2_PROXY_OIDC_ISSUER_URL` names, which every browser signs in through; on Auth0, a regular web application. It keeps its Auth0 name so an existing `.env` keeps working.	none	Required	operator
`AUTH0_PROXY_CLIENT_SECRET`	Its secret.	none	Required	operator
`AUTH0_MGMT_CLIENT_ID`	The machine-to-machine application for the Management API: the roster, enrollment, and the jobs role sync that revokes removed members. Blank, each role comes from the sign-in gate (`FLEETVIEW_MANAGER_GROUP` or `FLEETVIEW_MANAGER_EMAILS`, one of which `scripts/check-env.sh` then requires outside a trial), and a trial requires it blank.	none	Optional: set both or neither	operator
`AUTH0_MGMT_CLIENT_SECRET`	Its secret.	none	Optional: set both or neither	operator

Variable	Meaning	Default	Required	Layers
`AUTH0_TEST_ORG_ID`	The installation's one organisation, configured once: the control plane provisions it at start and serves its dashboard; enrollment falls back to it when a laptop's token names no organisation. Changing it needs the whole stack brought up again (`docker compose up -d`), which recreates the web server and `pg serve` : each keeps the key it was started with. Restarting only the backend leaves both on the earlier organisation, so until the whole stack is brought up again the Dashboard answers 503, because the web server admits only the organisation it started with, and spend reads as unavailable: the backend reads no spend while another organisation's serve role can still log in or is connected.	none	Required	operator
`ORG_DISPLAY_NAME`	The organisation's name, shown in the page headers. Read once, when the control plane first makes the Org; a later change does not rename an Org that already exists.	`Winner AI`	Optional	release, operator
`AUTH0_AUDIENCE`	Not read. The backend validates a laptop's Auth0 token against `FLEETVIEW_AUTH0_AUDIENCE` , which compose does not pass, so the tenant's API identifier must be that variable's default, `https://api.fleetview.local` .	none	Not read by the stack	none
`AUTH0_CLAIMS_NAMESPACE`	Not read. The backend reads roles under `FLEETVIEW_AUTH0_CLAIMS_NAMESPACE` , default `https://fleetview.local/` , and the gate's groups claim is `OAUTH2_PROXY_OIDC_GROUPS_CLAIM` .	none	Not read by the stack	none
`OAUTH2_PROXY_COOKIE_SECRET`	The session cookie key: 32 random bytes, URL-safe base64; check-env refuses a value the gate would not start with.	none	Required	operator

Variable	Meaning	Default	Required	Layers
`OAUTH2_PROXY_ALLOW_UNVERIFIED_EMAIL`	Admit a sign-in whose e-mail the identity provider has not verified (on by default). Set `false` where the provider's check is real and people cannot choose their address. `scripts/check-env.sh` refuses a manager e-mail list (`FLEETVIEW_MANAGER_EMAILS`) unless this is `false` , because the list trusts any address the gate admits.	`true`	Optional	release, operator
`OAUTH2_PROXY_OIDC_ISSUER_URL`	The issuer of the provider the sign-in gate (oauth2-proxy) signs people in through. `scripts/check-env.sh` refuses a production `.env` without it, so production names its provider instead of inheriting Auth0 from a missing line, and refuses a production issuer other than `https://<AUTH0_DOMAIN>/` while `AUTH0_MGMT_CLIENT_ID` is set, because with Management credentials a person is found by e-mail in Auth0. A person is their issuer and the provider's subject together, so a second provider presenting the same subject is another person; another provider signs people in on production with the Management keys blank. `scripts/check-env.sh` and the control plane refuse an https `.env` whose issuer is the bundled Dex (`http://localhost:4180/dex` , or `/dex` under `PUBLIC_BASE_URL`), and the `trial` profile requires `http://localhost:4180/dex` .	`https://${AUTH0_DOMAIN}/` , the Auth0 tenant	Required in production	release, operator
`OIDC_END_SESSION_URL`	The provider's end-session address, where sign-out sends the browser with `client_id` and the base URL to return to: `returnTo` for Auth0's `/v2/logout` , `post_logout_redirect_uri` for any other. For Auth0 it is `https://<AUTH0_DOMAIN>/v2/logout` . Its host must be in `OAUTH2_PROXY_WHITELIST_DOMAINS` ,	none; sign-out clears only the gate's cookie and the provider signs the same person straight back in	Required in production	operator

Variable	Meaning	Default	Required	Layers
	which `scripts/check-env.sh` checks. A trial leaves it blank, which `scripts/check-env.sh` requires: the bundled Dex has no end-session endpoint.			
`OAUTH2_PROXY_WHITELIST_DOMAINS`	The hosts the gate may send a browser to after sign-out, comma-separated as oauth2-proxy reads them; a leading `.` or `*.` admits subdomains, and `host:port` or `host:*` a port. The gate applies the same list to every redirect it follows, after sign-in too, so `scripts/check-env.sh` refuses an entry for a whole public suffix (`.com`, `*.co.uk`).	`${AUTH0_DOMAIN}`	Optional	release, operator
`OAUTH2_PROXY_SCOPE`	The scopes the gate asks for, always passed, because given none oauth2-proxy adds `groups` on its own. A provider that puts groups in the token only when asked wants `openid profile email groups`.	`openid profile email`	Optional	release, operator
`OAUTH2_PROXY_OIDC_GROUPS_CLAIM`	The token claim the gate reads groups from. Settings shows it as Groups claim, and without Auth0 Management credentials the control plane reads the role from these groups (`FLEETVIEW_MANAGER_GROUP`).	`https://fleetview.local/roles`, the claim the Auth0 post-login Action writes	Optional	release, operator
`OAUTH2_PROXY_SKIP_OIDC_DISCOVERY`	`true` makes the gate use the three addresses below instead of reading its issuer's discovery document at start. Only a trial sets it: the bundled Dex's issuer is the gate's own address, which the gate cannot read before it listens. `scripts/check-env.sh` refuses any value but `false` on an https base URL.	`false`	Optional	release, operator
`OAUTH2_PROXY_LOGIN_URL`	With discovery skipped, the provider's sign-in page as the browser reaches it; a trial's is `http://localhost:4180/dex/auth`. Set, it also gives the gate its public `/dex/`	none; read from the issuer's discovery document	Optional	operator

Variable	Meaning	Default	Required	Layers
	route to the bundled Dex, so only a trial sets it: `scripts/check-env.sh` refuses it on an https base URL and with the `trial` profile off.			
`OAUTH2_PROXY_REDEEM_URL`	With discovery skipped, the token endpoint the gate redeems a sign-in at; a trial's is `http://dex:5556/dex/token`, on the internal network. `scripts/check-env.sh` refuses it on an https base URL, where the gate discovers it from its https issuer.	none; read from the issuer's discovery document	Optional	operator
`OAUTH2_PROXY_OIDC_JWKS_URL`	With discovery skipped, the provider's signing keys; a trial's is `http://dex:5556/dex/keys`, on the internal network. `scripts/check-env.sh` refuses it on an https base URL, where the gate discovers it from its https issuer.	none; read from the issuer's discovery document	Optional	operator
`FLEETVIEW_GATE_SESSION_SECONDS`	How long a sign-in lasts, 10 to 604800 whole seconds (`scripts/check-env.sh` refuses a unit, underscore or fraction), before the gate sends the person back to the provider to be admitted again. It is the longest a person removed at the provider, or from the team, keeps web access: the gate checks nothing inside the window and asks for no refresh token. A form open when the window ends is lost, and a provider that keeps its own browser session signs the person straight back in. Settings shows it as Sign-in session.	`3600`	Optional	release, operator
`FLEETVIEW_MANAGER_GROUP`	Without Auth0 Management credentials, the provider group whose members are Managers, matched by its exact name among the groups the sign-in gate forwards; where groups arrive without it the person is a Member, and a person removed from it at the provider keeps the role until their sign-in ends, at most `FLEETVIEW_GATE_SESSION_SECONDS`.	none	Optional	operator

Variable	Meaning	Default	Required	Layers
	Blank, groups are not read and `FLEETVIEW_MANAGER_EMAILS` decides. Outside a trial `scripts/check-env.sh` requires it or `FLEETVIEW_MANAGER_EMAILS` where the Management credentials are blank.			
`FLEETVIEW_MANAGER_EMAILS`	Without Auth0 Management credentials, comma-separated addresses that are Managers where sign-in brings no usable groups claim, an ASCII address compared without regard to case and any other exactly. It trusts any address the gate admits, so `scripts/check-env.sh` refuses it unless `OAUTH2_PROXY_ALLOW_UNVERIFIED_EMAIL` is `false`.	none	Optional	operator
`PGSERVE_AUTH_TOKEN`	Bearer token gating the Org's dashboard API. Caddy and the control plane send it; browsers never see it.	none	Required	operator

4.4.4 Ingress and certificates

Variable	Meaning	Default	Required	Layers
`PUBLIC_BASE_URL`	Where browsers reach the proxy; it must match the callback and post-logout addresses registered for the gate's application at the provider (on Auth0, its callback and logout URLs). An https value marks a production deployment and must be `https://` followed by `PUBLIC_HOST`.	`http://localhost:4180`	Optional; on an install from the bundle outside a trial, `scripts/check-env.sh` requires it https	release, operator
`PUBLIC_HOST`	The hostname the production edge serves and obtains a certificate for.	none	Required in production	operator
`ACME_EMAIL`	The contact on the certificate authority account. Let's Encrypt sends no expiry notices, and nothing on a server installed	none	Required in production	operator

Variable	Meaning	Default	Required	Layers
	from the bundle watches the expiry: check it as Certificate renewals says.			
`CERTIFICATE_MODE`	How the deployment gets its certificate. `http`: the edge takes it over port 80, and Postgres presents the edge's. `dns`: the `cert-dns` profile also starts the `dns-certs` helper (stock lego), which obtains one certificate for `PUBLIC_HOST` and `PUBLIC_PG_HOST` through a TXT record in their public zone and keeps it renewed. The edge serves that certificate on 443 instead of taking its own, each renewal within a minute, and Postgres presents it on 5432, each renewal within one `db-certs` poll. Nothing has to reach port 80, which only redirects to https, and the edge serves nothing until the first certificate exists. Certificate authorities log every certificate, so the names become public. `scripts/check-env.sh` refuses any other value, and `dns` beside a base URL that is not https or a `PUBLIC_HOST` or `PUBLIC_PG_HOST` that is not a lower-case DNS name.	`http`	Optional	release, operator
`DB_CERTS_SOURCE`	The volume `db-certs` reads Postgres's certificate from: `edge-data` in the `http` mode, `dns-certs` in the `dns` mode.	`edge-data`	Optional	release, operator (`dns-certs` in the `dns` mode)
`DB_CERTS_SOURCE_DIR`	The directory in that volume: `/edge/caddy/certificates` in the `http` mode, `/edge/served` in the `dns` mode.	`/edge/caddy/certificates`	Optional	release, operator (`/edge/served` in the `dns` mode)
`AWS_HOSTED_ZONE_ID`	The public Route 53 zone the `dns` mode writes its challenge record in. The helper uses the server's instance role; no AWS	none	Required in the `dns` certificate mode	operator

Variable	Meaning	Default	Required	Layers
	key is stored. In the `dns` mode the helper refuses to start without it.			
`AWS_REGION`	The region the helper's AWS client signs its Route 53 calls for; a Route 53 zone itself has no region. In the `dns` mode the helper refuses to start without it.	none	Required in the `dns` certificate mode	operator
`EDGE_TRUSTED_PROXIES`	Space-separated CIDR ranges of a CDN or reverse proxy whose forwarded client address the edge may believe.	none; no forwarded address is believed and logs keep the peer address	Optional	operator
`OAUTH2_PROXY_COOKIE_SECURE`	Mark the session cookie Secure. Set `true` whenever the base URL is https.	`false`	Required in production	release, operator

4.4.5 LLM

Variable	Meaning	Default	Required	Layers
`FLEETVIEW_LLM_ENDPOINT`	The host model's endpoint (OpenAI-compatible, or a Bedrock VPC endpoint) for the coaching, skills and benchmarks passes; https, or a compose service. With `bedrock` it must be an https Bedrock runtime host (`bedrock-runtime.<region>.amazonaws.com` or its VPC interface endpoint).	`http://ollama:11434/v1`	Optional	release, operator, team (a team sets endpoint and model together)

Variable	Meaning	Default	Required	Layers
`FLEETVIEW_LLM_MODEL`	The host model.	`qwen3:4b-instruct-2507-q4_K_M`	Required with the `bedrock` provider (a Bedrock model or inference profile id, not the default)	release, operator, team (set with the endpoint)
`FLEETVIEW_LLM_API_KEY`	The host model's key, sent as a bearer token and never stored; blank sends no header. With `bedrock` it is a Bedrock API key, and blank signs each request instead: with `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` (and `AWS_SESSION_TOKEN` if set) from the process's environment, which compose does not pass, so nothing in the deployed stack uses them; else with the EC2 instance role, read through IMDSv2. One of the pair set without the other is refused, never a fall-through to the role.	none	Optional	operator, team (a team's own key, sealed)
`FLEETVIEW_LLM_FALLBACK`	What an organisation with no key of its own gets: `host` lends it the host model, `refuse` skips its model passes.	`host`	Optional	release, operator
`FLEETVIEW_LLM_PROVIDER`	Which API the host model is called through: `openai` for an OpenAI-compatible endpoint, `bedrock` for Amazon Bedrock's Converse API. A team's own model stays OpenAI-compatible.	`openai`	Optional	release, operator
`FLEETVIEW_LLM_REGION`	The AWS region of the host's Bedrock endpoint, such as `us-east-1`; with `bedrock`, a blank or default `FLEETVIEW_LLM_ENDPOINT` becomes that region's endpoint. Used only with `bedrock`:	none	Required with the `bedrock` provider	operator

Variable	Meaning	Default	Required	Layers
	`scripts/check-env.sh` refuses it set under any other provider, a sign the provider was forgotten. Left blank, Settings shows it as `release`.			
`FLEETVIEW_SETTINGS_KEYS`	Comma-separated Fernet keys that seal team model keys; the first seals, all of them open. Blank means no team key can be stored. Keep it out of database backups.	none	Optional	operator
`FLEETVIEW_COACHING_DAILY_BUDGET`	Prompts the coaching pass judges per organisation per rolling day: the cap on model spend.	`300`	Optional	release, operator, team
`FLEETVIEW_BENCHMARK_ROUTES`	Candidate routes per original model family, as a JSON map; these are replay targets only.	`{"gpt-5.6": ["gpt-5.5", "qwen3:4b-instruct-2507-q4_K_M"], "claude-opus": ["claude-sonnet-5"], "claude-fable": ["claude-sonnet-5"]}`	Optional	release, operator, team
`FLEETVIEW_BENCHMARK_LOCAL_ROUTES`	Route name prefixes the local endpoint serves without a key. They must be models the compose Ollama has pulled.	`["qwen3"]`	Optional	release, operator
`FLEETVIEW_BENCHMARK_LOCAL_ENDPOINT`	That local endpoint.	`http://ollama:11434/v1`	Optional	release, operator
`FLEETVIEW_BENCHMARK_DAILY_RUNS`	Replay runs per organisation per rolling day.	`20`	Optional	release, operator, team

Variable	Meaning	Default	Required	Layers
`FLEETVIEW_BENCHMARK_QUALITY_THRESHOLD`	The judged score, 0 to 100, a replay must reach, with every structured check passed, to count as Equivalent.	`95`	Optional	release, operator, team
`INSIGHTS_ENDPOINT`	The dashboard's generated-insights endpoint. It belongs to the stock dashboard, not to Settings; https when hosted.	`http://ollama:11434/v1`	Optional	release, operator
`INSIGHTS_MODEL`	Its model.	`qwen3:4b-instruct-2507-q4_K_M`	Optional	release, operator
`OPENAI_API_KEY`	Its key, read only from the environment.	none	Optional	operator

4.4.6 Jobs

Variable	Meaning	Default	Required	Layers
`FLEETVIEW_JOBS_INTERVAL_SECONDS`	How often the jobs service sweeps every organisation.	`900`	Optional	release, operator
`FLEETVIEW_ATTRIBUTION_INTERVAL_SECONDS`	How often session ownership is refreshed from the organisation schemas; `0` turns it off.	`300`	Optional	release, operator
`FLEETVIEW_HEARTBEAT_URL`	A dead-man's switch: the https URL the jobs service pings after every sweep.	none; no pings are sent	Optional	operator

4.4.7 Telemetry

Error reports and usage counts. A DSN that passes the host rule turns error reports on for the control plane and the jobs service (what a report carries); timings go with error reports at the sample rate (what a timing carries); with the telemetry secret set, cron check-ins go there too (what a check-in carries); usage counts are sent once the key, the host and the secret are all set and valid (Model and telemetry lists every event). Settings shows what each does in its Telemetry group.

Variable	Meaning	Default	Required	Layers
`FLEETVIEW_SENTRY_DSN`	Where error reports, timings and cron check-ins go; https, outside this host's network. Blank is off.	none	Optional	operator
`FLEETVIEW_SENTRY_ENVIRONMENT`	The environment name on every error report and check-in.	none; reports and check-ins then say production	Optional	operator
`FLEETVIEW_SENTRY_TRACES_SAMPLE_RATE`	The share of requests and jobs passes whose timings go where error reports go, from 0 to 1; 0, or a value that is not a share, sends none. Left blank, Settings shows it as `release`.	none; timings then sample 0.02, 2 in 100	Optional	operator
`FLEETVIEW_POSTHOG_KEY`	The analytics project key; usage counts need it, the host and the secret.	none	Optional	operator
`FLEETVIEW_POSTHOG_HOST`	The analytics host; https, outside this host's network.	none	Optional	operator
`FLEETVIEW_POSTHOG_REPLAY`	Page recording; no effect in this release.	`false`	Optional	release, operator
`FLEETVIEW_TELEMETRY_SECRET`	The key ids are hashed with, at least 32 characters; never shown or sent. Cron check-ins need it; give each deployment its own, and never change it.	none; an install from the bundle generates one	Optional	operator

4.4.8 Backups and host monitoring

These are not `.env` keys, and nothing on a server installed from the bundle reads any of them: they belong to a backup and watch schedule the bundle does not ship. They are listed only so that every name you may meet has a meaning here. On a bundle install, schedule and copy backups as Back up, restore and uninstall says, and watch the certificate as Certificate renewals says.

Variable	Meaning	Default	Required	Layers
`RCLONE_REMOTE`	The off-box target for nightly dumps, an rclone remote of an accepted type. Unset, the backup warns on every run.	none	Optional	operator
`BACKUP_PING_URL`	The backup's dead-man's switch, an https URL.	none	Optional	operator
`WATCH_PING_URL`	The watch's dead-man's switch, an https URL.	none	Optional	operator
`KEEP_DAYS`	Days of dumps kept, locally and off-box.	`14`	Optional	release, operator
`DISK_LIMIT`	Percent of the root filesystem used at which the watch reports a problem.	`85`	Optional	release, operator
`MEM_MIN`	The minimum percent of memory available before the watch reports a problem.	`10`	Optional	release, operator
`OLLAMA_LIMIT_MB`	Memory use of the Ollama container, in MiB, at which the watch reports a problem; it does not limit the container.	`6144`	Optional	release, operator
`AUTH_FAIL_LIMIT`	Refused Postgres logins from outside, within the window, that page.	`20`	Optional	release, operator
`AUTH_FAIL_WINDOW`	That window. Keep it at least twice the watch timer's five minutes, so a burst cannot straddle two runs.	`10min`	Optional	release, operator

Variable	Meaning	Default	Required	Layers
`AUTH_FAIL_VPN_LIMIT`	Refused Postgres logins from the VPN ranges declared in `.env`, within the same window, that page: the total over every address in them, while the second jail counts per address, so at the jail's count it pages at or before any one address's ban. Counted only while `VPN_RANGES` is set.	`60`	Optional	release, operator
`CERT_MIN_DAYS`	Days left on the certificate served on 443 or 5432 below which the watch reports a problem. Keep it under a third of the certificate's lifetime, where renewal begins.	`14`	Optional	release, operator

4.4.9 Development only

Read by nothing in a deployed stack, and not read on a server installed from the bundle; these exist for people working on Winner AI itself. Leave them blank.

Variable	Meaning	Default	Required	Layers
`AUTH0_TEST_ORG_NAME`	Slug of the test organisation Auth0 asks for at sign-in.	none	Not read by the stack	none
`AUTH0_TEST_MANAGER_EMAIL`	The test manager account.	none	Not read by the stack	none
`AUTH0_TEST_MANAGER_PASSWORD`	Its password.	none	Not read by the stack	none
`AUTH0_TEST_MEMBER_EMAIL`	The test member account.	none	Not read by the stack	none
`AUTH0_TEST_MEMBER_PASSWORD`	Its password.	none	Not read by the stack	none
`AUTH0_DEMO_PASSWORD`	The shared password of demonstration accounts made while working on Winner AI itself.	none	Not read by the stack	none

4.4.10 Team settings

A manager changes these on Settings, which shows each value, what it means and the layer that set it. A team value wins over the operator's, and the operator's over the release default; Reset removes the team value. A default written "as" a variable is that variable's row above. Settings also shows fixed rows (Window, Prompt length, Judge batch, Summary bullets, Rubric, Generation exemplars, Mining version, Eligibility, Token cap, Licence grace and Versions): release constants that no layer changes.

Model

On Settings	Key	Release default	Who can change it	Meaning
Endpoint	`llm_endpoint`	as `FLEETVIEW_LLM_ENDPOINT`	operator; team, together with Model	Where every Prompt Coaching, Skills and Benchmarks model call goes.
Model	`llm_model`	as `FLEETVIEW_LLM_MODEL`	operator; team, together with Endpoint	The model that judges prompts, names workflows, writes skills and scores benchmark replays.
Provider	`llm_provider`	as `FLEETVIEW_LLM_PROVIDER`	operator only; read-only on Settings	Which API this host's model is called through, an OpenAI-compatible endpoint or Amazon Bedrock, while a team's own key always uses an OpenAI-compatible endpoint.
Region	`llm_region`	as `FLEETVIEW_LLM_REGION`	operator only; read-only on Settings	The AWS region of this host's Amazon Bedrock endpoint, unused unless the provider is Amazon Bedrock.
API key	`llm_api_key`	none	operator; team, its own key, sealed (needs `FLEETVIEW_SETTINGS_KEYS`)	The key sent with each model call. Settings shows only whether one is set and its last four characters.
Without a team key	`llm_fallback`	as `FLEETVIEW_LLM_FALLBACK`	operator only; read-only on Settings	Whether a team with no key of its own uses the host's model (`host`) or runs no model work (`refuse`).

Prompt Coaching

On Settings	Key	Release default	Who can change it	Meaning
Prompts judged per day	`coaching_daily_budget`	as `FLEETVIEW_COACHING_DAILY_BUDGET`	operator; team, 0 to 5000	The most prompts judged in any rolling day, which caps the coaching bill; 0 pauses coaching.
Frontier model families	`frontier_model_families`	`opus`, `fable`, `gpt-5.6`	team only	The model families that count as top tier for the coaching judge and the coaching band.

Skills

On Settings	Key	Release default	Who can change it	Meaning
Clusters named per sweep	`skills_naming_budget`	`20`	team only, 0 to 100	How many new repeated workflows the model names on each sweep, which bounds the skills bill; 0 pauses naming.
Prompt similarity	`skills_similarity`	`80` (percent similar)	team only, 60 to 99	How alike two prompts must be to count as one workflow; a change regroups on the next sweep.
Prompts per workflow	`skills_min_prompts`	`3`	team only, 3 to 50	How many prompts a workflow needs before it is shown.
Sessions per workflow	`skills_min_sessions`	`2`	team only, 1 to 20	How many separate sessions those prompts must span.

Benchmarks

On Settings	Key	Release default	Who can change it	Meaning
Replay runs per day	`benchmark_daily_runs`	as `FLEETVIEW_BENCHMARK_DAILY_RUNS`	operator; team, 0 to 500	How many replays on cheaper models may run in any rolling day, each a model call that is paid for; 0 pauses replays.
Quality threshold	`benchmark_quality_threshold`	as `FLEETVIEW_BENCHMARK_QUALITY_THRESHOLD`	operator; team, 0 to 100	The judged score a replay must reach, with every structured check passed, to count as Equivalent.
Context budget per turn	`context_token_budget`	`150000`	team only, 10000 to 1000000	The cache-read tokens a turn may carry before the context saving lever prices the rest as reclaimable.
Saving figure freshness	`saving_lever_max_age_hours`	`24` hours	nobody: release only, read-only on Settings	How old a stored subagent or context saving may be before it reads Stale and leaves the total.
Routes to try	`benchmark_routes`	as `FLEETVIEW_BENCHMARK_ROUTES`	operator; team	For each model, the cheaper models replays run on and Prompt Coaching may recommend; live traffic is never rerouted.

Features and host

On Settings	Key	Release default	Who can change it	Meaning
Features on	`features`	as `FLEETVIEW_FEATURES`	operator sets the most that runs; a team can only switch one off	Which of Prompt Coaching, Skills and Benchmarks run for the team.

On Settings	Key	Release default	Who can change it	Meaning
Sweep interval	`jobs_interval_seconds`	as `FLEETVIEW_JOBS_INTERVAL_SECONDS`	operator only; read-only on Settings	How often the jobs service visits every organisation and runs its features.
Model call timeout	`llm_timeout_seconds`	`120` seconds	nobody in a deployment: the stack does not pass its variable through, so Settings shows the release default	How long one model call may take before it is abandoned.
Manager group	`manager_group`	as `FLEETVIEW_MANAGER_GROUP`	operator only; read-only on Settings	Without Auth0 Management credentials, the group from sign-in whose members are Managers.
Manager emails	`manager_emails`	as `FLEETVIEW_MANAGER_EMAILS`	operator only; read-only on Settings	Without Auth0 Management credentials, the addresses that are Managers when sign-in brings no usable groups.
Sign-in session	`gate_session_seconds`	as `FLEETVIEW_GATE_SESSION_SECONDS`	operator only; read-only on Settings	How long a sign-in lasts before the gate sends the person back to the provider to be admitted again.
Groups claim	`groups_claim`	`https://fleetview.local/roles`	operator only, as `OAUTH2_PROXY_OIDC_GROUPS_CLAIM`; read-only on Settings	The token claim the sign-in gate reads groups from.

Telemetry

On Settings	Key	Release default	Who can change it	Meaning
Error reporting	`sentry_dsn`	as `FLEETVIEW_SENTRY_DSN`	operator only; read-only on Settings	Where the backend and the jobs send error reports, timings and cron check-ins, without request bodies, local variables or error messages.
Error report environment	`sentry_environment`	as `FLEETVIEW_SENTRY_ENVIRONMENT`	operator only; read-only on Settings	The environment name every error report and check-in carries.
Timing sample rate	`sentry_traces_sample_rate`	`0.02`	operator only; read-only on Settings	The share of requests and jobs passes whose timings are sent, with no statement text, address or prompt.
Product analytics	`posthog_key`	as `FLEETVIEW_POSTHOG_KEY`	operator only; read-only on Settings	Whether usage events and page errors, never an error's message, go to the analytics project, which needs its key, its host and the identity hashing secret.
Analytics host	`posthog_host`	as `FLEETVIEW_POSTHOG_HOST`	operator only; read-only on Settings	The host usage events go to, which must be https and outside this host's network.
Page recording	`posthog_replay`	as `FLEETVIEW_POSTHOG_REPLAY`	operator only; read-only on Settings	Whether page sessions are recorded, with every text, input and attribute masked; in this release it shows off whatever it is set to.

Settings also shows Identity hashing: whether `FLEETVIEW_TELEMETRY_SECRET` is set, never its value.

4.4.11 The client

The options a person sets on a laptop. Everything else the `winner` client needs it reads from the control plane at `winner login`; the overrides below only replace what it discovers. Laptop client guide covers them, and Client reference has the detail.

Command	Option	Environment	Default	Meaning
`winner login`	`--control-url`	`FLEETVIEW_CONTROL_URL`	the URL recorded in `~/.agentsview/fleetview-control-url`, which the installer and every successful login write; none is compiled in	The control plane to enroll against: the flag, then the variable, then the recorded URL. Plain http is accepted only to a loopback address.
`winner login`	`--auth0-domain`	`FLEETVIEW_AUTH0_DOMAIN`	none: read from the control plane	Override the Auth0 tenant.
`winner login`	`--client-id`	`FLEETVIEW_AGENT_CLIENT_ID`	none: read from the control plane	Override the native application's client id.
`winner login`	`--audience`	`FLEETVIEW_AUDIENCE`	none: read from the control plane	Override the API audience. The three Auth0 overrides apply only against a control plane without its own login; one that serves it ignores them. With all three set and the control plane's config unreadable, login signs in with them, and the session collector (`agentsview`) version check only warns.
`winner login`	`--no-service`	none	off	Enroll only: no background push, hostname reporter or automatic updates.
`winner update`	`--control-url`	`FLEETVIEW_CONTROL_URL`	the recorded URL	The control plane to converge against, resolved as for `winner login`.
`winner update`	`--check`	none	off	Report what would change and install nothing.
`winner update`	`--force`	none	off	Replace a `winner` that is not a release build (your own `go build`), or an `agentsview` whose version cannot be read.

Command	Option	Environment	Default	Meaning
`winner push logs`	`-f`, `--follow`; `-n`	none	off; 50 lines	Follow the log; how many lines to show.
`winner push status`	`-v`	none	off	Add the service manager's own report.
`winner diagnostics`	`-o`, `--output`	none	`~/winner-diagnostics-<UTC time>.txt`	The file to write; an existing file is never overwritten.
installer	none	`FLEETVIEW_BIN_DIR`	`~/.local/bin`	Where `winner`, and `agentsview` when the installer installs it, are placed.
installer	none	`FLEETVIEW_CONTROL_URL`	filled in by the control plane that serves the script	Needed only when the script runs from a checkout.
installer	none	`FLEETVIEW_AGENT_URL`	the control plane's download	Fetch the binary from a mirror instead.
installer	none	`FLEETVIEW_AGENT_SHA256`	none: the signed `SHA256SUMS` beside the binary	The expected digest, read only with `FLEETVIEW_AGENT_URL`; required for a mirror that is not https, or whose URL has no `SHA256SUMS` beside it. With it set the release signature is not checked: the digest is yours.
installer	none	`FLEETVIEW_GH_TOKEN`	none	A GitHub token for an https mirror that needs one.

4.5 Model endpoint and telemetry

Which model Winner AI calls, how to point it at a hosted OpenAI-compatible provider or Amazon Bedrock instead of the compose Ollama, how to check that model with `fleetview-model-check` before Prompt Coaching is switched on, what each feature sends to that model, and whether the stack sends telemetry. It assumes a server installed from the bundle (Install the server).

On an installation from the bundle, read the server's commands as Commands on the other pages says: in `<install>`, as the user who owns `<install>/.env`, with plain `docker compose` and no `sudo`.

4.5.1 What calls a model

What calls it	Chosen by	Default
The jobs passes behind Prompt Coaching, Skills and Benchmarks; the control plane uses the same values (Settings, and Skills' Generate)	`FLEETVIEW_LLM_ENDPOINT`, `FLEETVIEW_LLM_MODEL`, `FLEETVIEW_LLM_API_KEY`, and for Amazon Bedrock `FLEETVIEW_LLM_PROVIDER` and `FLEETVIEW_LLM_REGION`	the compose Ollama at `http://ollama:11434/v1`, `qwen3:4b-instruct-2507-q4_K_M`, no key, provider `openai`
Generated insights in the dashboard	`INSIGHTS_ENDPOINT`, `INSIGHTS_MODEL`, `OPENAI_API_KEY`	the same compose Ollama and model
The dashboard's query embeddings and the bootstrap push	nothing; fixed	the compose Ollama, `nomic-embed-text`, 768 dimensions
Each laptop's own embeddings	nothing; handed out at enrollment	the laptop's `http://localhost:11434/v1`, `nomic-embed-text`, 768 dimensions

The first two rows are what this page changes. The last two are under Embeddings.

4.5.2 What the endpoint must support

This section is for the default provider, an OpenAI-compatible endpoint (`FLEETVIEW_LLM_PROVIDER=openai`). Amazon Bedrock has its own section.

- **OpenAI-compatible chat completions.** Each call is `POST <endpoint>/chat/completions`, so the endpoint is the provider's base URL, the one that ends in `/v1`.

- **Strict structured output:** `response_format` of type `json_schema` with `strict: true`. A provider that does not support it answers every call with an error, and the model check below fails.
- **A bearer key.** The key is sent as `Authorization: Bearer <key>`. A blank key sends no header.
- **https.** `FLEETVIEW_LLM_ENDPOINT` must be https or a compose-internal host. Anything else is refused with `FLEETVIEW_LLM_ENDPOINT must be https or a compose-internal host; refusing to use <scheme>://<host>[:port]`. For a jobs pass the refusal is the `error` on its `pipeline_runs` row; Skills' Generate shows that the host's model endpoint is misconfigured. `INSIGHTS_ENDPOINT` must be https too: the Dashboard refuses any http endpoint other than the compose Ollama, `http://ollama:11434`.

The client retries rate limits and transient errors itself, honouring Retry-After, and adapts to providers that reject `temperature` or want `max_completion_tokens`, so do not work around those.

Amazon Bedrock is a separate provider, not an endpoint of this kind: its OpenAI-compatible endpoint is not used, and the host model reaches Bedrock through its own API, described in the next section.

4.5.3 Amazon Bedrock

An operator can run the host model on Amazon Bedrock. The control plane calls Bedrock's Converse API and signs each request itself, so there is no gateway, SDK or extra container. It applies to the host model only: a team's own model stays OpenAI-compatible, and embeddings and the dashboard's insights are unchanged. Prompt Coaching and Skills need nothing else; Benchmarks also needs its Routes set to Bedrock ids (below).

- **Three settings.** `FLEETVIEW_LLM_PROVIDER=bedrock`, `FLEETVIEW_LLM_REGION` (an AWS region such as `us-east-1`) and `FLEETVIEW_LLM_MODEL`, which must be a Bedrock model or inference-profile id: the default model is Ollama's. Leave `FLEETVIEW_LLM_ENDPOINT` blank, and the endpoint is derived as `https://bedrock-runtime.<region>.amazonaws.com`. Set it only for a VPC endpoint: a Bedrock endpoint must be an https Bedrock runtime host (`bedrock-runtime.<region>.amazonaws.com` or its VPC interface endpoint), so the control plane and `scripts/check-env.sh` refuse a hosted OpenAI-compatible endpoint left from before rather than send it the Bedrock API key or a signed request. Settings shows the provider and region read-only.
- **Credentials, in this order.** A Bedrock API key in `FLEETVIEW_LLM_API_KEY`, sent as a bearer token; else the AWS variables `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` and `AWS_SESSION_TOKEN` in the process's environment, which the compose file does not pass, so nothing in the deployed stack uses them; else the instance role of the EC2 host, read through IMDSv2 and refreshed before it expires. Nothing is read until the first call.
- **The instance role needs hop limit 2.** A container reaches the metadata service one network hop further than the host does, so the instance's metadata hop limit must be 2 for the role to reach it (Network). The trade-off is that every container on the host can then use the role, so the role should allow only `bedrock:InvokeModel` on the chosen model and inference-profile ARNs, plus the DNS mode's TXT write where that mode is used. A Bedrock API key avoids the role and the hop limit.
- **The first call on a schema is slow.** Bedrock compiles each structured output schema on its first use, which AWS documents as taking up to about 90 seconds, and the default 120 s call timeout covers it. AWS caches the result per account for 24 hours from that first use, so the slow call recurs after that.

- **Errors.** A failed call records the HTTP status and AWS's error type, for example `HTTP 429 ThrottlingException`, and never the message. A host with no credentials from any of the three sources fails the pass with a fixed reason, without sending a request.
- **Routes must be Bedrock ids.** The release-default Benchmarks Routes' candidates (`claude-sonnet-5` under the `claude-opus` and `claude-fable` keys, `gpt-5.5` under `gpt-5.6`) are not Bedrock model or inference-profile ids, so every replay that is not local answers 400 until the operator or the team sets Routes that name ids Bedrock serves.
- **Check it first.** `fleetview-model-check` (next sections) builds its client the same way. It runs in the jobs container, which gets no AWS variables, so there it uses the Bedrock API key or the instance role. Its `refused:` line can also name the region, or an endpoint that is not an https Bedrock runtime host.
- **Tried so far: no Claude model on Bedrock has passed the model check yet.** On 2026-10-29, `us.anthropic.claude-sonnet-4-6` in us-east-1 answered the calls, with thinking off and `temperature` both accepted. It did not pass the rule below: Model fit F1 was 0.76 against a baseline of 0.77, and `any` F1 0.70 against 0.78, mostly because it over-flagged Context. The run took 11 calls and 29,983 + 7,638 tokens, about USD 0.20. The same day Claude Sonnet 5 (`us.anthropic.claude-sonnet-5`) was refused for that account with `AccessDeniedException`, so it has not been measured. Run the model check on your own model before you turn Prompt Coaching on.

An example in us-east-1

The host `.env` for Bedrock in us-east-1, with the instance role and no key:

```
FLEETVIEW_LLM_PROVIDER=bedrock
FLEETVIEW_LLM_REGION=us-east-1
FLEETVIEW_LLM_MODEL=us.anthropic.claude-sonnet-4-6
FLEETVIEW_LLM_ENDPOINT=
FLEETVIEW_LLM_API_KEY=
```

- Claude models on Bedrock are called through an inference profile, so the model is a profile id such as `us.anthropic.claude-sonnet-4-6` (routed in the United States) or `global.anthropic.claude-sonnet-4-6`. Run `aws bedrock list-inference-profiles --region us-east-1` to see which profiles the account has.
- The Sonnet 4.6 live runs, and the policy below, used a `us.` profile only; the one call made on a `global.` profile, for Sonnet 5, was refused for the account.
- For a VPC interface endpoint, set `FLEETVIEW_LLM_ENDPOINT` to its https runtime host instead of leaving it blank.
- To use a Bedrock API key instead of the role, put it in `FLEETVIEW_LLM_API_KEY`.
- Then recreate the services, as in step 3 of Point the stack at a hosted model.

The instance role's policy

The role needs one statement, `bedrock:InvokeModel`, on the inference profile and on the foundation model in every region that profile routes to. Nothing broader: at hop limit 2 every container on the host can use this role.

1. Read the ARNs from AWS rather than typing them:

```
aws bedrock get-inference-profile --region us-east-1 \
  --inference-profile-identifier us.anthropic.claude-sonnet-4-6 \
  --query '[inferenceProfileArn, models[].modelArn]'
```

2. Put them in the policy. This is the policy the live run used, with `<ACCOUNT_ID>` in place of the account:

```
{
  "Version": "2012-10-17",
  "Statement": [
    { "Effect": "Allow", "Action": "bedrock:InvokeModel",
      "Resource": [
        "arn:aws:bedrock:us-east-1:<ACCOUNT_ID>:inference-profile/us.anthropic.claude-sonnet-
4-6",
        "arn:aws:bedrock:us-east-1::foundation-model/anthropic.claude-sonnet-4-6",
        "arn:aws:bedrock:us-east-2::foundation-model/anthropic.claude-sonnet-4-6",
        "arn:aws:bedrock:us-west-2::foundation-model/anthropic.claude-sonnet-4-6"
      ] }
  ]
}
```

3. Converse is authorised by `bedrock:InvokeModel`; no other action is used.
4. In the `dns` certificate mode, the same role also carries that mode's Route 53 statements, which every container can then use too. To keep them from the containers, use a Bedrock API key and leave the hop limit at 1; the key alone does not stop a container reaching the role.
5. On EC2 the instance needs IMDSv2 required and hop limit 2, set before the install: `aws ec2 modify-instance-metadata-options --instance-id <instance-id> --http-put-response-hop-limit 2 --http-tokens required`.
6. In the live run, a container at hop limit 2 read the role and Bedrock accepted the signed call. At hop limit 1 the pass failed with the fixed reason: `set FLEETVIEW_LLM_API_KEY to a Bedrock API key, set AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY, or give the instance a role with metadata hop limit 2`.

What the AWS account needs

Each of these is set in AWS, not in Winner AI, and a missing one shows as an error on the first call:

- **Access to the model.** Anthropic models need the account's first-use details for Anthropic and the model's Marketplace agreement. `aws bedrock get-foundation-model-availability --model-id <model>` shows the agreement and the authorisation.
- **A quota in the region.** In Service Quotas, under Amazon Bedrock, the applied tokens-per-minute quota for the model's cross-region (or global) inference must be above 0.
- On 2026-10-29 a model whose availability read available and authorised, with a quota of 0, was still refused with `AccessDeniedException`.
- Only AWS can raise a quota of 0: a quota increase or AWS Sales.

- **A new account's daily token allowance.** AWS limits a new account's tokens per day, and a call over it is throttled until the next day. A full model check took about 38,000 tokens on Sonnet 4.6.

What each error means

A failed call records the HTTP status and AWS's error type, never AWS's message. A pass stores it on its `pipeline_runs` row, which Skills, Coaching and Settings show a Manager with when a call last worked, and the model check prints `the model call failed:` followed by:

Recorded	Means	Check
`HTTP 403 AccessDeniedException`	The role's policy does not cover the profile or a foundation-model ARN, or the account may not use the model	The policy above. Then the model's availability and its applied quota (What the AWS account needs)
`HTTP 403` with another type, such as `UnrecognizedClientException` or `InvalidSignatureException`	The credentials themselves were refused	The key, or the role and its session
`HTTP 429 ThrottlingException`	The model's quota in the region, or the account's daily allowance	Service Quotas; a daily allowance resets the next day. The client already retries with backoff, so do not add retries
`HTTP 400 ValidationException`	Bedrock refused the request or the schema, or the model id does not exist in the region	The model id and the region. The client already handles the three refusals it knows, once each and without failing: `thinking`, `temperature` and too high a `maxTokens`

`AccessDeniedException` looks the same whether the policy or the account is at fault, and nothing the stack records or prints tells the two apart. Check the policy first, then the account (What the AWS account needs).

Benchmarks Routes on Bedrock

Every candidate in `FLEETVIEW_BENCHMARK_ROUTES` is sent to Bedrock as the model, except one that matches `FLEETVIEW_BENCHMARK_LOCAL_ROUTES` (`qwen3` by default), which goes keyless to the compose Ollama. Each of the others must be an inference-profile or model id that Bedrock serves and that the role's policy covers. For example, on one line:

```
FLEETVIEW_BENCHMARK_ROUTES={"claude-opus": ["us.anthropic.claude-sonnet-4-6"], "claude-fable": ["us.anthropic.claude-sonnet-4-6"]}
```

- The keys are the release defaults' Claude families. A Route under a family with no entry, such as the default `gpt-5.6`, has no candidate on Bedrock.
- Add each further profile's ARNs to the role's policy.
- Benchmarks was not run on Bedrock in the live runs, so this example has not been tried.

4.5.4 Point the stack at a hosted model

The recommended path is one model for the whole deployment, set in the host `.env`.

1. In `<install>/.env`, set the three `FLEETVIEW_LLM_*` values, and the three insights values to the same provider:

   ```
   FLEETVIEW_LLM_ENDPOINT=https://<provider-host>/v1
   FLEETVIEW_LLM_MODEL=<model-id>
   FLEETVIEW_LLM_API_KEY=<key>
   INSIGHTS_ENDPOINT=https://<provider-host>/v1
   INSIGHTS_MODEL=<model-id>
   OPENAI_API_KEY=<key>
   ```

   Keep `local-llm` in `COMPOSE_PROFILES`: the compose Ollama still serves embeddings (Embeddings, below). Leave `coaching` out of `FLEETVIEW_FEATURES` until the model passes the model check (next section). The default is `[]`, so a stock deployment has no feature on.

2. Set the Benchmarks Routes. The default candidate list includes `qwen3:4b-instruct-2507-q4_K_M`, sent keyless to the compose Ollama, and the Ollama container pulls a chat model only while one of the two chat endpoints is `http://ollama:11434`. Once both are hosted and the container is recreated (step 3), it stops pulling that model; set the Routes so that Benchmarks does not depend on it. Prompt Coaching's Model fit suggests its cheaper models from the same map, so set it even if only Coaching is on. Set `FLEETVIEW_BENCHMARK_ROUTES` to a JSON map whose candidates are model ids your provider serves, on one line:

   ```
   FLEETVIEW_BENCHMARK_ROUTES={"claude-opus": ["<cheaper-model-id>"], "gpt-5.6": ["<cheaper-model-id>"]}
   ```

   - A key matches the original Route's name, or a dash- or dot-separated prefix of it. Each candidate is sent verbatim as `model`. The keys above are the release defaults' families; use your own originals.
   - Any candidate whose id starts with `qwen3` still goes keyless to the compose Ollama, because `FLEETVIEW_BENCHMARK_LOCAL_ROUTES` matches by prefix. When a hosted candidate starts with `qwen3`, also set `FLEETVIEW_BENCHMARK_LOCAL_ROUTES=[]`.
   - A team that saved its own Routes to try on Settings keeps them; this value applies to the teams that have not.

3. Recreate the services that read these values, `ollama` among them:

   ```
   # in <install>
   docker compose up -d
   ```

   Expected: compose recreates each service whose values changed, `ollama` included, and reports it started. The edits stay in place across releases: an upgrade from the bundle carries `.env` across (Upgrade and roll back).

4. Check: on a manager's Settings, the group "Model · bring your own key" shows the Endpoint and Model rows with the new values.

Per team, instead. A manager can give their organisation its own endpoint, model and key on Settings; that needs `FLEETVIEW_SETTINGS_KEYS` (The settings key). A team endpoint must be https and a public address: a local name or a private address is refused, the compose `ollama` aside. The model check below measures only the host model in `FLEETVIEW_LLM_*`, never a team's. The host's own `FLEETVIEW_LLM_API_KEY` is never sent to a team's endpoint: a team uses its endpoint, model and key together, or the host's three together.

`FLEETVIEW_LLM_FALLBACK` decides what a team with no key of its own gets: `host`, the default, lends it the host model; `refuse` makes no model call for it at all. Its passes still do the work that needs no model, and the product pages say No model configured. A deployment whose teams bring their own key sets `refuse`, so the host's key and budget are never spent on them.

The settings key

A team's model key is stored sealed with `FLEETVIEW_SETTINGS_KEYS`, so a database dump or backup alone does not reveal it. A server without that value cannot read a stored key, and every model pass for that team records the reason and does only the work that needs no model. Keep the value in `.env` only, never beside the backups. Generate one with the running backend, in `<install>`:

```
docker compose exec -T backend \
  uv run --no-dev python -c 'from cryptography.fernet import Fernet;
print(Fernet.generate_key().decode())'
```

The value is a comma-separated list: the first key seals, every key listed opens. Rotate without a gap:

1. Put the new key first: `FLEETVIEW_SETTINGS_KEYS=<new>,<old>`.
2. Recreate the two services that read it: `docker compose up -d backend jobs`.
3. Re-seal every stored key under the new one: `docker compose exec -T jobs uv run --no-dev fleetview-jobs rekey`. It prints how many keys it re-sealed and how many a Manager replaced or removed while it ran, which it leaves as they are; if any stored key opens with neither value it changes nothing and says so.
4. Drop the old key (`FLEETVIEW_SETTINGS_KEYS=<new>`) and recreate `backend` and `jobs` again.

A lost settings key cannot be recovered: each team's Manager enters its key again on Settings.

4.5.5 Check the model before Prompt Coaching

`fleetview-model-check` runs the production coaching judge over the labelled set the release ships, against the host's model: the endpoint, model and key in `FLEETVIEW_LLM_*`. On Bedrock it also reads `FLEETVIEW_LLM_PROVIDER=bedrock` and `FLEETVIEW_LLM_REGION`, and the endpoint may be left blank, since the region's own is derived. It scores the answers against the labels and compares them with the baseline recorded for that release. It runs inside the jobs container, from the image the server already runs, so it needs no checkout of the release; recreate `jobs` first if you have just changed `FLEETVIEW_LLM_*`.

```
# in <install>
docker compose exec -T jobs uv run --no-dev fleetview-model-check
```

An optional smoke run first: add `--limit 10` to judge only the first ten rows. It proves the endpoint accepts the calls and shows what a call costs, and it is never a verdict, whatever the number: it ends `not a verdict` and exits 3.

The last line is the verdict, and the exit code says the same:

Last line	Exit	Means
`verdict: pass`	0	A full run, compared with a baseline, neither score more than 0.05 below it
`verdict: fail - ...`	1	A score more than 0.05 below the baseline, or no baseline at this rubric version for this labelled set
`verdict: fail`	1	The run did not finish; the line directly above says why, and the `cost` line above that what it spent. A `refused:` line has no `cost` line above it: no call was made
`not a verdict: limited to ...`	3	A `--limit` run
`not a verdict: the limited run did not finish`	1	A `--limit` run that did not finish; the lines above it read as for `verdict: fail`

A usage error exits 2.

The model passes when neither its Model fit F1 nor its `any` F1 is more than 0.05 below the newest full-run baseline recorded at the same rubric version for the exact labelled set this release ships. A score above the baseline, however far, is a pass. Only the host model is measured, never a team's own, and only the coaching judge: Skills and Benchmarks call the same model but nothing here measures them.

A full run prints this shape, with the run's and the baseline's own values in place of the placeholders:

```
model check: <model-id> (hosted endpoint), rubric v<N>, <rows> of <rows> rows

coaching eval: rubric v<N>, <model-id> (hosted), <date>, <judged> rows (<excluded> excluded), <n>
calls, <p>+<c> tokens
...
cost: <n> calls, <p> prompt + <c> completion tokens
vs baseline <baseline-model> <baseline-date> (rubric v<N>, tolerance 0.05)
  Model fit f1  0.xxx vs 0.xxx (+0.xxx)
  any f1        0.xxx vs 0.xxx (+0.xxx)
verdict: pass
```

Between the header and the cost line it prints the per-category precision, recall and F1 table, the starts_task agreement, how many of the judge's rewrites the faithfulness check dropped, and the rewrite judge's agreement. The `cost` line is what the run cost in calls and tokens; a full run is about a dozen calls of up to ten prompts each.

What it prints is scores, counts, calls and tokens. It prints no prompt and nothing the model or the endpoint wrote, and writes no file. The one log line it can add, on standard error, is the judge's warning that an item was answered twice even when asked again, naming a labelled row's session id and ordinal and no text. When the run does not finish:

- `refused: FLEETVIEW_LLM_ENDPOINT must be https or a compose-internal host` means the endpoint would carry the key in the clear; it is refused before anything is sent, as the stack refuses it.
- `the model call failed: HTTP <status> ...` means the endpoint refused the calls: for example a 401, or a 400 from one with no strict `json_schema` support. What the endpoint said beyond the status is not printed (see below); send it a minimal request with the same model and `response_format` using `curl` to read it.
- `the model left items unanswered after a second ask` means the model did not answer every item, even when asked again.
- `the run stopped on an unexpected <Type>` means the run failed for a reason it does not describe; only the error's type is printed.
- A call that runs past `FLEETVIEW_LLM_TIMEOUT_SECONDS` (120 by default) fails the run. A CPU-bound local Ollama answering ten prompts at a time can take longer than that, and a full run keeps it busy for every call; give one run more time by adding `-e FLEETVIEW_LLM_TIMEOUT_SECONDS=900` after `exec -T`.

If the model does not pass, leave Prompt Coaching off and try another model.

4.5.6 Switch Prompt Coaching on

1. Add `coaching` to `FLEETVIEW_FEATURES`, for example `FLEETVIEW_FEATURES=["coaching"]`. Skills and Benchmarks have their own slugs, `skills` and `benchmarks`. A team can switch a feature off on Settings but never on: the operator's list is the ceiling.
2. Recreate `backend` and `jobs`:

```
# in <install>
docker compose up -d backend jobs
```

3. The recreated jobs service sweeps as it starts, so there is no need to wait for the timer (900 seconds by default). Wait until its log shows the sweep finished, then read the newest coaching rows:

```
# in <install>
docker compose logs jobs | grep -E 'rows written this sweep|jobs sweep failed'
docker compose exec -T db sh -c 'psql -U "$POSTGRES_USER" -d "$POSTGRES_DB" -c "$1"' sh \
  "SELECT feature, model, error, finished_at FROM control.pipeline_runs WHERE feature =
'coaching' ORDER BY started_at DESC LIMIT 5"
```

The database user and name are read inside the `db` container, as compose parsed them from `.env`, so a quoted value in `.env` reads the same as it does to Postgres.

Expected: the `logs` command prints `jobs: <n> rows written this sweep, <m> passes had no model to call`; if it prints nothing, the sweep is still running, so run it again. `jobs sweep failed for <organisation>` means that organisation's passes raised and the sweep carried on; `jobs sweep failed` with nothing after it means the whole sweep crashed. Either way, read the traceback below the line with `logs jobs`. Then the newest `coaching` row has an empty `error`, and its `model` is the configured model once the pass judged something; `model` is empty when there was nothing new to judge.

4.5.7 Embeddings

The compose Ollama stays on while `local-llm` is in `COMPOSE_PROFILES`. It pulls `nomic-embed-text` at every start and serves the dashboard's query embeddings and the bootstrap push. That model is fixed in `deploy/pgserve/config.toml`, not a compose setting, and it must be the laptops' model too, because vectors from different models do not compare. A hosted embedding model is therefore not an option.

Enrollment points each laptop at its own `http://localhost:11434/v1` with `nomic-embed-text`, 768 dimensions. Winner AI installs neither Ollama nor the model on a laptop. A laptop without them still pushes its sessions, but no vectors, and Skills reads only the vectors laptops push, so that laptop's prompts are absent from Skills. To give a laptop vectors, install Ollama on it and pull the model: `ollama pull nomic-embed-text`.

4.5.8 What each feature sends to the model

This is what a model endpoint receives from Prompt Coaching, Skills and Benchmarks, per call. An automated test per row captures the request, from the feature's own read but for Skills' Generate, whose test starts from what the mining read returns, and holds the kinds of data listed here and the shape of the request: it fails when one of them starts or stops being sent, or when the request gains a field or a message. It cannot rule out a kind of data nobody thought to plant in its fixtures.

A prompt is sent with nothing redacted, so it carries whatever the person typed or pasted into it: a path, a name, a file's content. Prompt Coaching and Skills first cut the blocks an agent adds around a prompt (Claude Code's hook, command, reminder and message wrappers, Codex's context items and image wrapper, and the Codex app's browser context, heartbeat block and writing-block edits, with the request heading it puts before the person's words) and send what is left; a prompt that is nothing but such a block, or opens with a hook, command, reminder or task block, is not sent, and neither is a Codex app scheduled automation run that opens with its `Automation:` header, a prompt one agent wrote for another (Codex's agent history, a plan handed to a fresh context, Claude Code's background task notice), one with a `$ARGUMENTS` line, one with a wrapper's closing tag that opens no block, or one with a wrapper block that is never closed. The cut finds a block where its opening tag starts a line: one that begins in the middle of a line, follows another block's closing tag on the same line, or sits inside a code fence stays in what is sent. Benchmarks replays a prompt whole, and a prompt a hook or template wrote can inline an earlier reply or a diff; naming a pattern sends what a person typed of its prompt, what its shown text would show of a known opening, and otherwise the prompt as stored.

Call	Sent
Prompt Coaching, judging prompts	Each prompt a person typed, cut to what they typed: at least 40 characters of it, from a prompt of at most 8,192, nothing redacted. Beside each one: the agent's name, the model its session mostly ran on, whether it is the session's first prompt, the team's Routes to try for that model, the session's outcome, and the counts stored with the session (user messages, tool failures, edit churn, and whichever prompt-quality counters are not zero). The agent, model, outcome, Routes and frontier families are sent as plain names: words of letters, digits and `. _ - : / @`, at most 64 characters, so up to that much of what a laptop reported as its agent, model or outcome sits beside the prompt as ordinary words. Two Routes that come to the same plain name are numbered. The instructions name the team's frontier model families. One call carries the prompts attributed to one member only, up to ten; a session nobody is attributed to yet is sent on its own.
Prompt Coaching, the main points to improve	Per person, and once for the sessions nobody is attributed to yet, from what the judge stored and nothing read again from a transcript: how many prompts were judged, were eligible and had findings, how many sessions were judged and had a Model fit verdict, and how many findings each category has; the judge's reasons, up to 30 per category; the cheaper Route it named; and up to 8 of its suggested rewrites, each cut to 600 characters. A rewrite restates the person's prompt, so it carries what that prompt said.
Skills, naming a workflow	Up to 5 prompts of the workflow, each cut to what a person typed: at least 50 characters of it, from a prompt of at most 8,192.
Skills, Generate and Regenerate	Up to 10 distinct typed prompts of the workflow, each cut to what a person typed and then to 1,800 characters; the workflow's stored name and summary, which the model wrote from its prompts when it was named; how many prompts and sessions it has; and its scope, which is the project name at least 60% of its prompts came from, or `Global`. The name, summary and scope are marked as data, like the prompts.
Benchmarks, naming a pattern	One prompt of the pattern, cut to 1,800 characters. When the cut Prompt Coaching makes leaves any of it, only what it leaves is sent. A prompt that opens with a hook, command or other opening Winner AI knows is sent only as far as a pattern's shown text would show it, at most its first 100 characters: the blocks it opens with, up to but never including a slash command's arguments, text after a closing tag, or a block that is never closed or that holds a shell-mode command, a slash command's arguments or its own opening tag; a command run in Claude Code's shell mode as its bare `<bash-input>` tag, the command left out, and a prompt that opens with a slash command's arguments as its bare `<command-args>` tag; an opening that is a line of text rather than a block, such as Claude Code's interruption notice, as those 100 characters, whatever follows it on them up to a shell-mode command or a slash command's arguments; and an opening that cannot be placed, nothing. Any other prompt is sent as stored, a block an agent added included: a prompt an agent composed, such as a review request Claude Code writes for Codex, a Codex app scheduled automation run with the name and saved prompt it carries, or a Codex app writing-block edit with the block text it carries; or a prompt a person typed that the cut's stricter rules drop whole, such as one opening "adversarial review". One that holds a shell-mode command or slash command arguments at the start of a line, in a code fence or out of one, is not sent at all. A pattern whose

Call	Sent
	prompts open with a known hook or plugin template is never named, so never sent. Any other pattern is, whichever kind of session its prompt was issued in, an SDK or automated run included.
Benchmarks, replaying a task	The task's prompt, whole, and every `system` message stored for its session, sidechain ones included, to the endpoint that serves the Route being tried: the model endpoint, or the local one for a local Route. A task's prompt is one a hook or template wrote, so it carries whatever that template put in it, which can be an earlier reply or a diff.
Benchmarks, judging a replay	The same prompt, every reply of the original session outside its sidechains, as stored, which includes the model's thinking where the agent records it, and the reply the Route gave.
The model check, `fleetview-model-check`	The labelled set the release ships in its image, judged as Prompt Coaching judges a prompt: each labelled prompt, cut as above, beside its row's agent, session model, position, outcome and counters, ten to a call; then each labelled rewrite beside its prompt, for the rewrite judge. The instructions name a fixed list of frontier families and the cheaper Routes for each, not the team's. Nothing is read from the database, so no team's prompt is sent.

Text a member typed, and text a model wrote from it (a workflow's name, a judge's reason), goes to the next call between markers that the instructions declare to be data. Inside the markers, a run of three or more `<` or `>` is written out as `{lt}` or `{gt}`, bracket by bracket, so the text cannot close a marker or open one. The same goes for the characters drawn as an angle bracket that Winner AI lists (fullwidth signs, angle quotation marks, mathematical and CJK brackets) and for a run broken only by spaces or invisible characters; the list is not every glyph a model might read as a bracket. Nothing else in the text is changed. This keeps such text in its place in the request. It does not prove a model will ignore an instruction written there.

Every call also carries its fixed instructions and the model's name. The replay adds a limit on the answer's length, worked out from the original's token counts; every other call adds the shape its answer must take, and when an answer does not fit that shape the call is made once more with the model's own answer and the validation error added to it. No call adds the session's id, the laptop's name or who typed the prompt, and none reads the sessions' tool-call table, where a tool's result is kept; Benchmarks replays and judges only sessions that used no tool. Only Skills' Generate adds a project name, and only the Benchmarks judge reads a session's replies.

When a call fails, the log line and the `error` stored on the `pipeline_runs` or `benchmark_runs` row say the HTTP status or the kind of failure, the model, the name of the shape asked for and how many tries were made, and why no connection could be made when none was (a refused certificate, an unknown name). They never hold the request, the endpoint's response or the model's answer, because an endpoint that refuses a request can quote it and a rejected answer is a model's writing about a member's prompts. To see why a provider refused a call, send it a minimal request with the same model and `response_format` using `curl`. A failed write to the control database does not print the values it was writing.

Sessions flagged as holding a secret

The session collector (`agentsview`) looks for secrets in every session it stores and records each one it finds; Home counts them as Credential exposures. A session with any such finding is flagged, whatever the finding's confidence and wherever in the session it is. The control plane adds no detection of its own and does not read what was matched.

While the findings table lists a finding for a session, no call in the table above sends anything from it: not its prompts, its replies, its system messages or the judge's stored results about it. The check is made each time a feature reads, so a session flagged after it was first read is left out for as long as it stays flagged. If the lookup itself fails, nothing is sent: the pass fails, or the page shows no stored result. What it means for results made before the flag:

- **Prompt Coaching.** Findings already shown stay on the page until a release next changes the rubric version. They are not sent with the main points to improve.
- **Skills.** A workflow with an example from a flagged session is not listed while it is a candidate or a draft, and cannot be generated or regenerated. An approved skill stays listed and installable. Examples from a flagged session are not shown as evidence.
- **Benchmarks.** A run whose sample session is flagged is not shown and does not count toward the proven saving. A pattern's name stays until a release next changes the pattern version; a name is never sent to a model.
- **The disputes export.** A Winner AI developer's tool, run from a checkout of the release and never by the server, writes the prompts a manager disputed on Prompt Coaching into a labelled set in a file it is given, outside the repository, which Winner AI's model evaluation then sends to the model under test; the set a release ships never holds them. It reads as Prompt Coaching does, so a prompt from a flagged session is not written and is listed with the prompts no longer in the organisation's data; a row an earlier export wrote from it is taken out of the file. Until that export is run again, the row stays in the file and the next eval sends it.

What the rule does not do:

- **It does not outlast the finding.** The findings table is the collector's, and every laptop of the organisation can change it; the collector may also replace a session's findings when the session is pushed again. A session whose findings are removed is not flagged and is sent like any other. That is not caught today.
- **It does not catch a secret the collector did not flag.** The rule is as good as its detector.
- **It recalls nothing.** A request made before the flag was already sent, and nothing is deleted from the database.
- **It does not cover Insights**, which is the Dashboard's own call; Telemetry, below, says what that sends.

4.5.9 Telemetry

- **Winner AI sends error reports, timings and usage counts only when the operator sets them.** The control plane and the jobs service read seven operator settings for them (configuration reference, Telemetry). Error reports leave when `FLEETVIEW_SENTRY_DSN` is set to a DSN that passes the host rule; "What an error report carries", below, lists every field. Timings go to the same project while error reports are on, for the share of requests and jobs passes that `FLEETVIEW_SENTRY_TRACES_SAMPLE_RATE` sets; "What a timing carries", below, lists every field. While error reports are on and the telemetry secret is valid (set and at least 32 characters), the jobs sweep sends cron check-ins there, unless `FLEETVIEW_JOBS_INTERVAL_SECONDS` is under 60, and `fleetview-checkin` lets a host job (the backup, the certificate renewal) do the same once its script calls it; the backup and renewal scripts this release ships do not call it yet. "What a check-in carries", below, lists every field. Usage counts leave only when

`FLEETVIEW_POSTHOG_KEY`, `FLEETVIEW_POSTHOG_HOST` and `FLEETVIEW_TELEMETRY_SECRET` are all set and valid; they go from the control plane and the jobs service, best effort, to that host's `/batch/`. An unreachable analytics host never fails a request or a pass. Settings shows the state of each in its Telemetry group, and "Usage counts, event by event", below, lists every event. What a laptop sends is under "What is never instrumented, and what a laptop does", below.

- **The browser's usage counts have a path on the deployment's own origin, `/lumen/`, and each control-plane page posts its page view, and any uncaught error or unhandled rejection in its own scripts, there.** The page loads the analytics library, a pinned build of posthog-js, from the control plane and never from the vendor, and only while usage counts are on: with them off a page carries neither the library nor its settings. While usage counts are on, the web server forwards `POST /lumen/e/` and nothing else, over verified https, to `FLEETVIEW_POSTHOG_HOST`; a host that names a path keeps it shut, because the edge posts only at the host's root. The vendor never sees the cookie, authorization, identity, client-address, `Referer` or `Origin` headers, and its answer can set no cookie or other header that acts on the whole origin, and is sandboxed. Every other address under `/lumen/`, page recording's included, answers 404, and so does a request whose `Origin` is missing or is not this deployment's `PUBLIC_BASE_URL`. The path is behind sign-in like every page, and a removed person is refused.
- **A hosted model receives what it is sent.** "What each feature sends to the model", above, lists it call by call. Insights sends aggregates and a few sample prompts, not transcripts.

What blank settings guarantee

`FLEETVIEW_SENTRY_DSN`, `FLEETVIEW_POSTHOG_KEY` and `FLEETVIEW_POSTHOG_HOST` are blank in a new installation, and blank is off. While they are blank, nothing is sent, no vendor library is loaded, no telemetry script is rendered into a page and no connection is opened to an error-report or analytics host. The telemetry secret alone sends nothing: an installation from the bundle writes one before any vendor value is set. A blank `FLEETVIEW_SENTRY_TRACES_SAMPLE_RATE` is not off, it is 2 in 100, but no timing is sent while the DSN is blank.

An installation may be handed an `.env` with our values already filled in. Blanking them and restarting the backend and the jobs service, as below, returns it to sending nothing: Settings then reads `off` for Error reporting and Product analytics.

Switching each kind on and off

Each kind is switched by the `.env` settings below; the configuration reference gives each its row. Settings shows the state of each in its Telemetry group, which no one can change from the page.

Kind	Setting	Sent while	What Settings shows
Error reports	`FLEETVIEW_SENTRY_DSN`	the DSN is valid	**Error reporting**: `on` with the host and the project number; `off` while it is blank; `invalid`. **Error report environment**: the name every report and check-in carries. While `FLEETVIEW_SENTRY_ENVIRONMENT` is blank, Settings shows `none` and

Kind	Setting	Sent while	What Settings shows
			reports and check-ins carry `production`.
Timings	`FLEETVIEW_SENTRY_TRACES_SAMPLE_RATE`	error reports are on and the share is above 0	**Timing sample rate**: the share as a decimal; `off` while error reports are not on, or when the value is `0` or not a share between 0 and 1. Blank is 2 in 100.
Cron check-ins	none of their own	error reports are on, the telemetry secret is valid, and the sweep interval is 60 seconds or more	No row of their own: **Error reporting** reads `on` and **Identity hashing** reads `set`.
Usage counts	`FLEETVIEW_POSTHOG_KEY`, `FLEETVIEW_POSTHOG_HOST` and `FLEETVIEW_TELEMETRY_SECRET`	all three are set and valid	**Product analytics**: `on`; `off` while the key is blank, whatever else is set; `incomplete`; `invalid`. **Analytics host**: the host name, `not set` or `invalid`.
Page views and browser errors	none of their own: they ride usage counts	usage counts are on	**Product analytics**, as above.
Page recording	`FLEETVIEW_POSTHOG_REPLAY`	never in this release, whatever it says	**Page recording**: always `off`.
Identity hashing, for every id Winner AI sets	`FLEETVIEW_TELEMETRY_SECRET`	not a kind: the key every id Winner AI sets is hashed with, which never leaves the deployment; a page with no one to hash sends the library's own random id instead, as "Usage counts, event by event" says	**Identity hashing**: `set`, `not set` or `invalid`.

- **Incomplete** means the analytics key is set but the host or the secret is not. Nothing is sent.
- **Invalid** means a value is set but fails its rule: a DSN or analytics host that is not https or that points at this host's own network, a DSN that names no key or no numeric project, or a secret shorter than 32 characters. Nothing of that kind is sent. Where one value is invalid and another is missing, Settings shows `invalid`, because that names

what to fix; while the analytics key is blank, Product analytics reads `off` whatever else is set. An invalid secret also stops check-ins and leaves error reports without the hashed deployment id.

- `0` as the sample rate keeps error reports and drops timings.

The backend and the jobs service read these settings when they start. After editing `.env`, restart both, then reload Settings:

```
# in <install>
docker compose up -d backend jobs
```

What an error report carries

Every report is rebuilt from a fixed list of fields before it leaves, so a field this list does not name never reaches the vendor. The vendor also sees the address the deployment connects from, as the receiver of any https request does.

- **Every report:** an event id, the time, the level, the platform, the release version, the environment name (`FLEETVIEW_SENTRY_ENVIRONMENT`, or `production` when that is blank), and the SDK's name, version, package and integration list.
- **The failure:** each exception in its chain by type and module, never its text, with how it was caught (the mechanism's type and whether it was handled), and each stack frame by file, function, module, line and whether it is our code, never a local variable or a line of source.
- **A route failure**, an uncaught error or a 5xx a route returns on purpose: the method and the route template (never the path as requested), and no header. A failure before a route is matched carries neither, and a 4xx refusal sends no report.
- **Tags:** `component` (`backend` or `jobs`); `deployment`, the hashed deployment id, when the telemetry secret is valid; `failure_class` (`model_call` or `provisioning`); `error_class`, the kind of a model failure (`transport`, `http_status`, `body_not_json`, `body_shape` or `invalid_output`); and `pass` and `feature` when a jobs pass failed.
- **A laptop's failure report**, at most ten an hour per laptop: the message `winner login failed` or `winner update failed`, the logger name `winner`, with `command`, `stage`, `http_status` when there was one, `error_type`, `winner_version`, `agentsview_version` when the laptop could read it, `os`, `arch` and `device`, the hashed Device id, left out when the telemetry secret is unset or too short to hash with, and nothing else.
- **Never:** a request body, a query string, a cookie or a header, an identity or a user, a server name, a local variable, an error or log message, a breadcrumb, prompt text, a repository name, a file path other than a stack frame's, a key, or a setting value other than the environment name above. No sessions or client reports are sent.

What a timing carries

A timing says how long a request to one route, or one jobs pass for one team, took, and how long each database statement, outbound call and model call inside it took. Each is rebuilt from a fixed list of fields before it leaves, as an error report is, and goes to the same project. `FLEETVIEW_SENTRY_TRACES_SAMPLE_RATE` is the share sampled: blank is 2 in 100, and `0`, or a value that is not a share between 0 and 1, sends none. None is sent while error reports are off.

- **Every timing:** an event id, its type (`transaction`), its start and end time, the platform, the release version, the environment name, the SDK's name, version, package and integration list, a trace id and span id this deployment made (never one a caller sent), its kind (`http.server` for a request, `pass` for a jobs pass), its source (`route` or `task`), and the tags `component` (`backend` or `jobs`) and `deployment` , the hashed deployment id, when the telemetry secret is valid. No trace header travels with it.
- **A request:** the route template (never the path as requested), the method when it is one of `GET` , `HEAD` , `POST` , `PUT` , `PATCH` , `DELETE` or `OPTIONS` , and the status class (`1xx` to `5xx`). A request that matched no route, and the health check, send none.
- **A jobs pass:** the pass name, never the team.
- **Inside either,** each with its span id, the span id it sits under, the trace id, its kind (`db` , `http.client` or `model`) and its start and end time: a database statement as its kind alone (`SELECT` , `INSERT` , `UPDATE` , `DELETE` , `WITH` , `BEGIN` , `COMMIT` , `ROLLBACK` , `SET` , `CREATE` , `ALTER` , `DROP` , `GRANT` , `REVOKE` , `DO` , `COPY` , or `other`); an outbound call as `http` ; a model call as the model's name for this host's model or `team` for a team's own, with its prompt and completion token counts when it succeeded.
- **Never:** statement text or its values, an outbound call's address, path, query or host, a header, a query string, a request body, a prompt, a reply, code, a repository name, a file path, an error message, a local variable, an identity or a raw id. No outbound call carries a trace header, and none sent in is followed.

What a check-in carries

A check-in is rebuilt from a fixed list of keys before it leaves, as a report is, so a key this list does not name never reaches the vendor. It is sent only while error reports are on and the telemetry secret is valid: without the hashed deployment id every deployment would share one monitor.

- **Type:** `check_in` , which marks the item as a check-in.
- **Event id:** a random id, which the SDK needs to send it.
- **Check-in id:** a random id that pairs a run's end with its start.
- **Monitor slug:** the monitor, its component and the deployment, one of `jobs-sweep-<id>` , `backend-backup-<id>` and `backend-certificate-renewal-<id>` , where `<id>` is the first 16 hex characters of the hashed deployment id.
- **Status:** `in_progress` , `ok` or `error` .
- **Duration:** seconds, on the end of a run whose start was seen, otherwise none.
- **The monitor's schedule:** its interval, check-in margin and longest run time, from which the vendor makes and updates the monitor, and the failure and recovery thresholds and the time zone, `UTC` .
- **Environment:** the environment name, as an error report carries it.
- **Never:** an error's text, a path, a host or file name, an org, a tag or any free text. The monitor and status come from closed lists and the duration is a number the run measures.

What a browser error carries

A control-plane page reports an error its own scripts raise, an uncaught error or an unhandled rejection, only while usage counts are on and only when what was thrown is an `Error` with a stack. The page rebuilds the event from a fixed list before it leaves, so nothing else the library gathered goes with it. A page sends at most five a load, and none for an error with no frame in the page's own scripts or whose message holds the page's address.

- The error's type name where it is one of JavaScript's own (`Error` , `TypeError` and the other standard constructors), else `Error` .
- For each frame of the page's own scripts, its line and column, its function where that is a plain name, and the route template as its file.
- The route template as the page (`$current_url` , `$pathname`).
- The hashed user as the distinct id, as for a page view.
- `component` , which is `web` , and `deployment` , the hashed deployment id.
- The page view's browser properties and the library's `token` .

Never: the message, the address or its query, page text, an attribute value, typed input, a frame of another script (the library's, an extension's, another origin's), or a trail of clicks, inputs or earlier steps.

Usage counts, event by event

Every event the control plane or the jobs service sends carries `deployment` , this deployment's id hashed under its secret, and `$geoip_disable: true` ; a page view or a page error, which the browser sends, carries no `$geoip_disable` , and only a page error carries `deployment` . The lifecycle events below and `laptop enrolled` also carry `component` , the part of Winner AI that sent them: `backend` or `jobs` . An event whose id is not a person's also carries `$process_person_profile: false` ; a person's key actions are sent under a hashed user, and so is a page view or a page error where the page has one to hash. Every id Winner AI sets is an HMAC-SHA256 under `FLEETVIEW_TELEMETRY_SECRET` , so the analytics host sees no raw id and cannot reverse one. The one exception is a page view or a page error on a page with no hashed user: the library then sends its own random id, kept for the tab, which is no HMAC and names no one; a tab that signs out or changes person starts a fresh id and never carries the last person's hash.

Never sent: a prompt, a reply, code, a repository name, a file path, a form field, a name, an email, a setting's value, a model key, an endpoint host, an error message or a raw id. The one model name that leaves is `$ai_model` for the operator's own host model; a team's model is sent as `team` , because its name may be the customer's. A Bedrock model id given as an ARN is sent as its last segment, because an ARN holds the AWS account id.

Event	When	Id	Properties
`member added` , `device revoked` , `person removed` , `person restored` , `org schema migrated` , `benchmark run queued` , `team model`	A person's change, after its route answers below 400	The person, hashed	`route` (the route template, never the address), `status_class` (`2xx` or `3xx`), `org` (hashed), `deployment`

Event	When	Id	Properties
`saved`, `team model reset`, `team model key saved`, `team model key removed`, `licence uploaded`, `skill generated`, `skill regenerated`, `skill candidate dismissed`, `skill edited`, `skill approved`, `coaching finding disputed`, `laptop login decided`, `repo choice queued`			
`setting saved`, `setting reset`	A person's change, after its route answers below 400	The person, hashed	`route` (the route template, never the address), `status_class` (`2xx` or `3xx`), `org` (hashed), `setting_key` (the setting's key on Settings, never its value), `deployment`
`pass completed`	Once per pass run	The team, hashed	`feature`, `pass_version`, `outcome` (succeeded, failed or skipped), `rows_in`, `rows_out`, `duration_ms`, `model_source`, `deployment`
`$ai_generation`	Each model call, from a pass or from Skills' Generate and Regenerate, a failed call included	The team, hashed, inside a pass or a Skills generation; this deployment, hashed, otherwise	`$ai_trace_id`, `$ai_model`, `$ai_provider` (`openai-compatible`, or `amazon-bedrock` for a call through Amazon Bedrock), `$ai_input_tokens`, `$ai_output_tokens`, `$ai_latency`, `attempts`, `finish_reason` (`stop`, `length`, `tool_calls`, `content_filter` or `function_call`; any other value the endpoint gives is sent as `other`), `$ai_is_error`, `error_class` (the class

Event	When	Id	Properties
			name, never the message), `feature`, `pass_version`, `model_source`, `deployment`
`winner login failed`, `winner update failed`	When an enrolled laptop reports that the command failed, at most ten an hour per laptop	The laptop, hashed	`stage`, `http_status`, `error_type`, `winner_version`, `agentsview_version`, `os`, `arch`, `deployment`
`laptop enrolled`	Each successful enrolment, a re-login included: a re-login enrols a new laptop row under a new hashed id, so these count enrolments, not distinct laptops	The laptop, hashed	`org` (hashed), `os`, `arch`, `winner_version` (each a release target or a bare version; any other value is sent as `other`), `component`, `deployment`
`deployment installed`, `telemetry switched on`	The first start with usage counts on; a start with them on after one with them off	This deployment, hashed	`component`, `deployment`
`deployment heartbeat`	Once per UTC day, from the jobs service	This deployment, hashed	`version` (this release, or `other`), `seats` (people with at least one enrolled laptop), `laptops_pushing`, `laptops_stale`, `laptops_quiet`, `laptops_never`, `winner_version_counts`, `os_counts`, `arch_counts` (counts of enrolled laptops per value), `component`, `deployment`
`deployment upgraded`	When `scripts/upgrade-server.sh` completes an upgrade, from the new release's control plane	This deployment, hashed	`version` (the new release, or `other`), `component`, `deployment`

Event	When	Id	Properties
`deployment rolled back`	Not sent yet	This deployment, hashed	`version`, `component`, `deployment`
`licence state changed`	From the jobs service, when its sweep finds the licence state moved; the first sweep after the upgrade records the state without sending	This deployment, hashed	`licence_state` (`licensed`, `grace` or `lapsed`), `component`, `deployment`
`$pageview`	Each load of a control-plane page while usage counts are on, sent by the browser through `/lumen/`; a form whose answer is a whole page (a refused form, Add row and Remove row, the new member's one-time page) counts as a page view of the form's own route; never the documentation, the laptop-login approval page or the dashboard	The person, hashed; where the page has no one to hash (a person with no roster row where Auth0 holds the roles, or the host-run bypass before its roster row exists), the library's own random id for the tab, which names no one	`$current_url` and `$pathname` (both the route template the server matched, never the address or its query), `$lib`, `$lib_version`, `$browser`, `$browser_version`, `$os`, `$device_type`, `$session_id`, `$window_id`, `$process_person_profile`, and the library's own `token` (the analytics project key, public by the vendor's design) and `distinct_id` (the hashed id, or that random id), without which the event is not sent or is filed under no one
`$exception`	An uncaught error or unhandled rejection in a control-plane page's own script while usage counts are on, at most five per page load	The person, hashed, as for a page view	`$current_url` and `$pathname` (both the route template, never the address), `$exception_list` (the error's type name and, for each frame of the page's own scripts, its line, column, function where that is a plain name, and the route template as its file; never the message), `component` (`web`), `deployment`, `$lib`, `$lib_version`, `$browser`, `$browser_version`, `$os`,

Event	When	Id	Properties
			`$device_type`, `$session_id`, `$window_id`, `$process_person_profile`, and the library's own `token` and `distinct_id`, as for a page view

The deployment heartbeat is not the operator's dead-man's switch (`FLEETVIEW_HEARTBEAT_URL`), which is unchanged and sends nothing to Winner AI.

A Benchmarks replay call to a Route is counted as `team`, in `$ai_model` and `model_source` alike, because its model is named by the Route.

A refused form sends no key action, and neither do the Benchmarks routes editor's Add row and Remove row, which only redraw the page. Each redrawn page still sends its page view, under the form's route: `/settings/{key}` for the two buttons.

What is never instrumented, and what a laptop does

- **The Dashboard is never instrumented.** Its pages carry no analytics library, no error reporter and no page view. The only script Winner AI adds to them is the brand script, which sends nothing.
- **The Dashboard's telemetry is switched off.** The pg serve container runs with `AGENTSVIEW_TELEMETRY_ENABLED` set to `0` and `--no-update-check`, and the control plane's schema setup runs with `AGENTSVIEW_TELEMETRY_ENABLED` set to `0`.
- **The collector's telemetry on laptops is switched off too.** Every service `winner` installs sets `AGENTSVIEW_TELEMETRY_ENABLED` to `0` and `AGENTSVIEW_DISABLE_UPDATE_CHECK` to `1`, and the collector's config, which enrollment writes, sets `disable_update_check = true`. A daemon you start yourself afterwards, from the agentsview desktop app or a shell, carries your own environment instead, so those two variables do not reach it. What the laptop sends, and where says what a laptop still reaches, which includes the collector's model-price refresh, which has no switch.
- **A laptop contacts no telemetry vendor.** `winner` holds no vendor address, key or telemetry state. It reports a failed `winner login` or `winner update` to the control plane it enrolled against and nowhere else. The one exception to this bullet is the daemon you start yourself, above. What else a laptop reaches, none of it telemetry, is in What the laptop sends, and where.
- **The report is not stored.** While error reports or usage counts are on, the control plane builds the events listed above from it (the laptop's error report, and `winner login failed` or `winner update failed`) and keeps no copy. It holds only the time of each report, in memory, to keep a laptop to ten an hour, whether or not telemetry is on. With both off it builds nothing, and the laptop is answered the same either way.
- **Page recording cannot be switched on in this release.** It is built and held off whatever `FLEETVIEW_POSTHOG_REPLAY` says, and its Settings row reads `off`.

There is no developer switch. A developer cannot turn the report off on the laptop; the operator's settings above decide whether the control plane passes anything on. `winner login` prints this notice once the laptop is enrolled, and When `winner` fails shows it too:

> If `winner` fails after this laptop is enrolled, it sends a short report to your organisation's Winner AI server: the command, the step, the HTTP status, the kind of error, the versions, and your operating system and processor type. The report never contains prompts, code, repository names, file paths, emails or the text of an error. If your organisation's operator has switched telemetry on, the server passes those fields on to Winner AI's error reports and, under a scrambled laptop id, its usage counts. `winner` sends the report nowhere else.

Where it goes

Each kind goes where its setting points. The values Winner AI hands over point at Winner AI's own projects: error reports, timings and check-ins at Sentry, and usage counts, page views and browser errors at PostHog. Winner AI holds no copy of a deployment's telemetry secret, so it cannot reverse a hashed id.

Some of what happens to an event happens in those projects, where a deployment cannot see it. We keep them set as follows:

- **Sentry:** its data scrubber is required, with its default rules; IP addresses are not stored; events are kept for 30 days. Sentry still derives a country and city from the address an event arrives from before it drops the address, so an error report can be kept with a location near the server that sent it.
- **PostHog:** the client address is discarded and its GeoIP transformation is off, so no location is kept; events are kept for 30 days; session recording, autocapture, heatmaps, exception autocapture, web vitals and console-log recording are off.
- **Keys:** every deployment we hand values to sends usage counts under one shared analytics key, and deployments are told apart there only by the hashed `deployment`. Each gets an error-report key of its own, which we can revoke alone.

If you later blank `FLEETVIEW_SENTRY_DSN` or the telemetry secret, or rotate the secret, which changes the hashed deployment id, the check-in monitors we hold for your deployment stop hearing from it and report it missed. That alert stands with us until we remove the monitor, so we may ask whether you meant it. Nothing on your side needs doing.

4.6 Back up, restore and uninstall

For an installation from the self-hosting bundle. The database is the archive: everything your team's laptops pushed lives there and nowhere else, so a backup you have restored is the only copy that counts. Installing a later release, and going back, is in Upgrade and roll back.

Every server command here runs as the user who owns `<install>/.env` (the scripts read it, and only its owner can) and who can run `docker`. `<install>` is the folder the running release was unpacked into, the one holding its `.env` and `scripts/`: `winner-install/winner-ai-server-<X.Y.Z>` for the first release, `winner-install-<X.Y.Z>/winner-ai-server-<X.Y.Z>` for a later one. Write it as an absolute path.

Older release folders stay on disk after an upgrade, and every one of them drives the same running stack: the stack's name is fixed, not taken from the folder. Run these commands only from the running release's folder; `docker compose ls` shows the compose files the running stack was started from, and `grep ^FLEETVIEW_VERSION= .env` which release a folder holds.

4.6.1 What a backup holds, and what it does not

A backup is one gzipped `pg_dumpall` of the whole database: the organisation's sessions, the control plane's own tables (people, laptops, settings, the licence) and every database role, the laptops' own included, so a restore brings back the whole installation rather than its tables. Each backup is restored into a throwaway database and its row counts compared with the live one before it is given its name, so a file with that name has already restored once.

It does not hold `.env`, and a restore needs the `.env` the database ran with. The backup carries the database's own passwords, so a stack started with any other `POSTGRES_PASSWORD` cannot log in to the database it restored; and without the same settings key it cannot read the team model keys sealed with it. Keep a copy of `.env` somewhere safe, and not beside the backups, so that no single stolen copy holds both. Note which release each backup was taken under, and restore it into that release.

Certificates are taken again and the AI runner pulls its model again on a new server, so neither needs a backup.

4.6.2 Take a backup

```
# the user who owns <install>/.env
mkdir -p ~/winner-backups
<install>/scripts/backup-server.sh ~/winner-backups/<date>.sql.gz
```

It runs against the live stack, which keeps serving. It prints `backup <file>: <bytes> bytes, restore test PASS (<n> tables match)` when the backup is good; otherwise one line, `backup: refused: ...` or `backup: FAIL: ...`, says why, and the exit status is non-zero. A failed run keeps its dump as `<file>.failed`, for diagnosis, or as `<file>.part` when it could not rename it. A laptop push between the dump and the count can fail a run with differing counts: run it again under another name.

Each run needs free disk for the dump where you write it and, while it is proved, a scratch copy of the database in Docker's own storage. A nightly backup needs that room every night, beside the backups you keep.

The file is readable by its owner only and holds every transcript and every database login: keep every copy that way. Delete `.failed` and `.part` files once you have read why they failed, and old `pre-restore-*` files on the same schedule as old backups; never schedule the deletion of a `pre-restore-*.failed` file, which can be the only copy of the database as it was before a restore. The command refuses a path inside the installation folder, a name that exists already (or whose `.part` or `.failed` does), a name ending in `.part` or `.failed`, a folder it cannot write, and a database fenced for a restore: a dump taken mid-restore would pass its own test against a half-replayed database.

It also refuses a database that holds a CHECK constraint, generated column or domain check stock agentsview does not create (`backup: refused: the database holds ... CHECK constraint(s) or generated column(s) ...`), since a restore would run it as the database superuser. Clear it as Code in an org schema that agentsview did not create says, listing it with `docker compose exec -T backend uv run --no-dev fleetview-provision audit` in `<install>`, then back up again.

4.6.3 Schedule it and copy it off the server

Nothing on a bundle install schedules a backup or copies one anywhere. Both are yours. Run the command above by hand first and see the `PASS` line, then add a line with `crontab -e` as the same user, nightly:

```
# crontab -e, as the user who runs the install; <install> as an absolute path
15 3 * * * <install>/scripts/backup-server.sh "$HOME/winner-backups/winner-$(date -u
+\%Y\%m\%d).sql.gz" >> "$HOME/winner-backups/backup.log" 2>&1
```

Cron finds `docker` only in `/usr/bin` or `/bin`: for any other install of Docker, put a `PATH=` line above it. `<install>` changes with every upgrade, since each release unpacks into a new folder: point the line at the new folder once the upgrade is done. A rollback (`scripts/rollback-server.sh`) goes the other way: point it back at the previous release's folder, which runs again. The newer release's `winner-install-<X.Y.Z>` folder is kept as `winner-install-<X.Y.Z>.rolled-back-<UTC stamp>` when the upgrade made it, and otherwise stays where it is, never to be run.

Nothing tells you when a night fails. Read the last line of `backup.log`, which is the `restore test PASS` line only when the latest night succeeded, or have your own monitoring alert when no good backup is newer than two days:

```
# the user who takes the backups; prints nothing when there is none
find ~/winner-backups -name 'winner-*.sql.gz' -mtime -2
```

Copy each good backup off the server, encrypted, with your own tool: a backup on the server it protects does not survive losing that server.

4.6.4 Restore

```
# the user who owns <install>/.env
<install>/scripts/restore-server.sh ~/winner-backups/<date>.sql.gz
```

A restore replaces the whole database with the backup. The command checks the file and proves it replays before it changes anything, then stops the stack, shuts every laptop out of the database, backs up the database as it was (`pre-restore-<stamp>.sql.gz`, beside the backup), replays the backup, proves the row counts match, and starts the stack again. The dashboard is down and laptops wait until it ends. It prints `restore: restored <file>: <n> tables match the backup`, then where the database as it was is kept, and last `restore: a laptop enrolled after this backup was taken must run `winner login` again`. With `--keep-fence`, as the rollback command runs it, a line after the first says `the fence stays up and the stack stopped, as --keep-fence asked`.

If it prints `restore: refused: ...`, nothing was stopped or changed: the reason names what to fix, such as the file, a path, or a stack it cannot find.

If it prints `restore: FAIL: ...`, the line after it says what state it left; no line after it means nothing was stopped:

- `nothing was changed; the stack was started again`, or `nothing was restored; the fence is lifted and the stack started again`: the database is as it was and the stack keeps running. Read the reason, fix it, and restore again. `nothing was changed, but starting the stack failed`: the database is as it was; start the stack with `docker compose up -d`, then fix the reason and restore again.
- `nothing was restored, but lifting the fence or starting the stack failed`: run the same command again, as the line says.
- `the database stays fenced and the stack stopped`, or `nothing more was restored; the database stays fenced from the earlier restore`: a replay began in this run, or may have in an earlier one, so the database may not be the backup and must not take a push. Leave the stack stopped and do not start it by hand. Restore again, from the same backup or from the `pre-restore-` file the run that first raised the fence named, and keep that file: it can be the only copy of what was there.

A run that prints `restore: restored ...` can still have printed `WARNING: could not start every stopped service` above it: the database is restored, so start the stack with `docker compose up -d` in `<install>`.

While it runs, Postgres listens only inside its own container, so every laptop is refused a connection. A refused connection writes no Postgres log line, so no laptop is banned; its push retries as it would while the server is down, and catches up once the restore ends. The restore needs free disk for one more dump beside the backup and, while each is proved, a scratch copy of the database in Docker's own storage. Run one restore at a time: two are not locked against each other.

After a restore:

- **A laptop enrolled after the backup was taken cannot push.** Its owner runs `winner login` again, which enrolls it afresh. Its refused pushes count toward the Postgres jail where you run one, so unban its address if the jail already has (The Postgres jail).
- **A laptop revoked after the backup was taken** shows as enrolled again on Admin but cannot log in to the database; the command names each one. Revoke it again on Admin.

After losing the server, the restore is the rehearsal below with production's names: a new server, the release the backup was taken under, installed with the `.env` you kept, DNS pointed at the new server, then `restore-server.sh`. The new server's empty database had no laptop logins, so the command reads every laptop in the backup as revoked after it, takes each one's login away and names it: there that list is every laptop, not revocations to repeat. Each person whose laptop should push again runs `winner login` on it. Until they do, every laptop's pushes are refused, and where you run the Postgres jail on the new server those refusals count toward it: unban an address it has already banned, as The Postgres jail shows.

4.6.5 Rehearse a restore

Every backup is already restored once, into a throwaway database. The rehearsal proves the rest: that you can build a server from the bundle, the `.env` you kept and a backup, and that people can sign in to the result. Do it once the installation is running, and again after each upgrade.

Rehearse on a second server, never in a second folder on the production server: every folder drives the same stack, so a restore there replaces production's database.

1. **Prepare a test server** that meets the same requirements as production (Check the requirements), with its own DNS name (two, if production's `PUBLIC_PG_HOST` differs from `PUBLIC_HOST`). In the `http` certificate mode, keep ports 80 and 443 open to the internet, since the edge takes its certificate over port 80; in the `dns` mode, put the test names in the same Route 53 zone and give the test server an instance role that may write their challenge record. Either way, admit only your own address to 5432: once restored, it holds every transcript and every laptop's database login.
2. **Install the same release on it** from its bundle, as Installing the Winner AI server says, signature and checksum checks included, with a copy of the production `.env` in which you change only:
3. `PUBLIC_HOST`, `PUBLIC_PG_HOST` and `PUBLIC_BASE_URL`, to the test server's names;
4. `FLEETVIEW_HEARTBEAT_URL`, `FLEETVIEW_SENTRY_DSN` and `FLEETVIEW_POSTHOG_KEY`, to blank, so the test server neither answers for production's dead-man's switch nor reports as a second installation;
5. the `FLEETVIEW_FEATURES` line, removed, so no model pass runs there.

To sign in to it, add `https://<test-host>/oauth2/callback` to the sign-in application's allowed callback addresses at your identity provider, and its base URL to the allowed logout addresses where the provider has them.

1. **Copy the latest backup to it** over `scp` or `rsync` (never a shared drive or a chat), and restore it: `<install>/scripts/restore-server.sh <backup>`. It must print `restored <file>: <n> tables match the backup`. It also names every laptop as revoked after the backup, as on a new server after losing one: leave them; no laptop pushes to the test server.
2. **Sign in as a Manager, and look without changing anything.** Admin lists your people and laptops as they were at the backup, and the Dashboard shows the newest sessions you expect from before it. Do not add anyone there: the test server shares production's identity provider, and on Auth0 a person added there is added to production's tenant.
3. **Take the test server down.** On the test server only, in its `<install>`: `docker compose down --volumes --rmi all`. Leave the laptops, any crontab and the identity provider's applications alone: they are production's. Then

delete the test server's copies of the backup and `.env`, the `pre-restore-*` file the restore wrote beside the backup, its DNS records, the callback and logout addresses you added, and in the `dns` mode the test server's instance role, or the test names you added to a policy.

A rehearsal that fails at step 3 means your backups would fail you too: send support the line the command printed. One that fails at step 4 after step 3 printed its `restored` line is first the test server's DNS, certificate or Auth0 addresses; once those are right, a step 4 that still fails means the same as a step 3 failure.

4.6.6 Uninstall

Removing an installation deletes its database, the only copy of every session. Take a last backup first, unless you are certain nobody will want the sessions again, and copy it and `.env` off the server, kept apart as above: a restore of that backup needs that `.env`.

Laptops first, so none keeps calling an address that stops answering. Each person removes the client from their own laptop as Uninstall in the laptop client guide says.

Then the server. Check you are on the right server and in the running release's folder (`grep -E '^(PUBLIC_HOST|FLEETVIEW_VERSION)=' .env`), then:

```
# the user who owns <install>/.env; in <install>
docker compose down --volumes --rmi all
```

That stops and removes every container, deletes the stack's volumes (the database, the dashboard's data, the certificates and the AI runner's models) and removes the images this release ran. It cannot be undone. It removes the stock images too (Postgres, Caddy, agentsview, oauth2-proxy, Ollama, and the certificate helper in the `dns` mode), so another stack on the same server that uses one pulls or loads it again. Images of earlier releases stay: `docker image ls` lists them. Remove the crontab line that ran backups (`crontab -e`), then delete the `winner-install*` folders, with the `.env` in each, and the backups you no longer need, wherever they were copied. Keep the off-server copy of `.env` for as long as you keep a backup it restores.

Last, what lives elsewhere: the DNS records, the firewall rules for 80, 443 and 5432, the Postgres jail and its filter where you set them up on the server, in the `dns` mode the instance role and its Route 53 policy, your dead-man's switch and any telemetry project, and the sign-in application at your identity provider (on Auth0, the applications and the organisation).

4.7 Upgrade and roll back

For an installation from the self-hosting bundle. An upgrade takes the server to the release in a newer bundle and keeps every person, Device and session; a rollback takes it back to the release the last upgrade replaced, or to an older one you name.

Every command here runs from the folder of the release that runs now, as the user who owns its `.env` (mode 0600, so only its owner can read it) and may run `docker`. Another user who may run `docker` is refused, because it cannot read the file: the upgrade says `cannot read <folder>/.env, ...`, and the rollback says `<folder>/.env records no FLEETVIEW_VERSION`. An upgrade backs the database up first, and a rollback restores that backup: Back up, restore and uninstall says how backups are taken, kept and restored.

4.7.1 Installing a later release

One command takes the server to the release in a newer bundle and keeps every person, Device and session. The uploaded licence, Settings and roles are kept too, so nothing is uploaded or set again. Before you upgrade, read the new release's **Requires of you** section in the Changelog. Download the new `winner-ai-server-<X.Y.Z>.tar.gz` anywhere outside the installed folder, then, from the installed release's folder (`winner-install/winner-ai-server-<old X.Y.Z>` for the first upgrade), as the user who owns its `.env`, giving the path you downloaded it to:

```
./scripts/upgrade-server.sh /path/to/winner-ai-server-<X.Y.Z>.tar.gz
```

There is nothing to verify by hand: the installed release checks the new bundle's signature against the release key it carries, then every file, and refuses anything else. `./scripts/upgrade-server.sh --help` lists its steps in order. It unpacks the new release beside the installed one, in `winner-install-<X.Y.Z>/winner-ai-server-<X.Y.Z>`, carries `.env` across (the telemetry secret with it), runs the new release's host and `.env` checks and pulls its Docker Hub images, then backs up the database with `scripts/backup-server.sh`. Until that backup has passed its restore test, nothing that runs is touched. Only then does it load and start the new release, which migrates the database, and wait for it to answer with every migration applied:

```
upgrade-server: step 1 of 10: check the new bundle: ...
upgrade-server: the new bundle is v<X.Y.Z>, signed by the release key, every file as signed
upgrade-server: upgrading v<old X.Y.Z> in <old folder> to v<X.Y.Z> in <new folder>
...
upgrade-server: the database is backed up at <folder the archive is in>/pre-upgrade-v<old X.Y.Z>-
<UTC stamp>.sql.gz
...
upgrade-server: every migration v<X.Y.Z> holds is applied
upgrade-server: the web server runs the new bundle's binary
upgrade-server: the edge serves ports 80 and 443
upgrade-server: the org schemas hold only what agentsview creates
upgrade-server: upgraded v<old X.Y.Z> -> v<X.Y.Z>; the previous release stays in <old folder> for
a rollback, its images loaded
```

When usage counts are off, or the telemetry secret is not set, a line before the last says `usage counts are off or the telemetry secret is not set; the upgrade was not reported`.

By default the backup goes in the folder the new bundle's archive is in. `--backup <file>` writes the backup somewhere else; like any backup it must be outside both release folders. The new release's folder is where you run `docker compose` from now on: its `.env` names the compose files and the release. Keep the previous folder and its images, and never prune images: they are what a failed upgrade, and a rollback, start again. `winner-install-<X.Y.Z>/winner-upgrade.record` records the release replaced, its folder and the backup's path, for the rollback.

Laptops after an upgrade. Nothing runs on a laptop. Each one keeps pushing with the `agentsview` and `winner` it has until `winner update` brings it to the versions the new release runs. A Mac or Linux laptop whose `winner login` ran without `--no-service` runs it on its own each day (Running on its own), except a Linux laptop without systemd, which gets only the crontab lines `winner login` printed, to add by hand. On any other laptop its owner runs `winner update`.

On the server, point your nightly backup line at the new release's folder (Schedule it and copy it off the server), and rehearse a restore again (Rehearse a restore).

Auth0 after an upgrade or a rollback. The tenant's post-login Action is deployed by hand, so neither command changes it. Compare it with the copy in the release that now runs, as The post-login Action says, and deploy that copy if they differ.

A slow upgrade is not a failed one. A release that moves the collector version rebuilds every organisation's schema on its first start, which can take half an hour, and the control plane does not answer until it is done; the command says so about once a minute. If it is still migrating after thirty minutes (`UPGRADE_WAIT_SECONDS`), it puts nothing back, says `still migrating after 1800s; this is slow, not failed`, and exits 3. Watch it with `docker compose logs -f backend` in the new release's folder, and when it is done, run `./scripts/upgrade-server.sh --finish` there.

When it refuses, before it starts the new release, it prints one `upgrade-server: FAIL:` line saying `nothing was changed`, removes the new folder, and the installed release runs as before. A refusal once the backup has passed, `docker load` failing included, also says `the backup at <file> stays`:

What it says	What to do
`SHA256SUMS.sig is signed by another key (...)`, `... was made in the licence namespace`, or `SHA256SUMS does not match its signature in SHA256SUMS.sig: ...`	The archive is not a release Winner AI signed for you, or it was changed. Download it again; if it still says so, ask your Winner AI contact
`these files of the new bundle do not match SHA256SUMS: ...`, `the new bundle holds files SHA256SUMS does not name:`	The archive changed after it was signed. Download it again

What it says	What to do
`...`, or `... holds a link or a special file`	
`no .env in <folder>; ...`, `cannot read <folder>/.env, ...` or `<folder>/.env records no FLEETVIEW_VERSION; ...`	Run it from the installed release's folder, as the user who owns its `.env`
`<tool> is not installed, and the new bundle cannot be checked without it`	Install the tool it names (`ssh-keygen`, `sha256sum`, `tar` or `od`), and run it again
`the new bundle carries no scripts/<name>.sh to finish the upgrade with, or to roll it back`	The archive is not a whole release. Download it again
`this server already runs v<X.Y.Z>`	Nothing to do
`the new bundle is v<X.Y.Z>, older than v<old X.Y.Z> which this server runs; migrations do not reverse, ...`	The upgrade command never installs an older bundle over a newer release. To go back to the release the last upgrade replaced, roll back (Going back to the previous release); to an older release, roll back with `--bundle` naming its folder and `--backup` the backup taken before the upgrade from it
`winner-install-<X.Y.Z> already exists; ...`	An earlier attempt left that folder, and it may hold a release that ran. Read `winner-install-<X.Y.Z>/winner-upgrade.record` first. `state=migrating`: the upgrade was slow, not failed; run `./scripts/upgrade-server.sh --finish` in its release folder. `state=started`: it was interrupted while the new release started; do what that run printed, as **When it fails after the start** below says. `state=done`: that release is installed, so run from its folder. No record: nothing started from it; move it aside, and run it again
`the backup: ...`	Choose another `--backup` path; the reason says why this one was refused
`<folder>/.env names COMPOSE_FILE=<files>, not the bundle's docker-compose.yml:docker-compose.prod.yml:docker-compose.bundle.yml`, `COMPOSE_FILE in <folder>/.env names <path>, which is not an absolute path to a file`, or `... inside a release folder`	The upgrade command moves a production install only, never a trial (A trial from the bundle), whose `.env` names no production overlay. On a production install, put `COMPOSE_FILE` in that `.env` back to the bundle's three files, with any file of your own after them as an absolute path outside both release folders, and run it again

What it says	What to do
`<old folder> has no scripts/backup-server.sh to back the database up with`	The installed release's folder is incomplete: run it from the folder the installer or the last upgrade left running
`this host is missing what v<X.Y.Z> needs (above)`	Fix each `check-host: FAIL:` line above it, and run it again
`v<X.Y.Z>'s scripts/check-env.sh refused .env (above)`	The new release needs a change to `.env`: make it in the installed release's `.env`, and run it again
`<image> is neither in the new bundle nor a pinned Docker Hub image`	A compose file of your own, named in `COMPOSE_FILE` after the bundle's three, adds an image that is not pinned by `@sha256:` on Docker Hub. Pin it by digest, and run it again
`could not pull <image> from Docker Hub: ...`	As at install (step 5): wait, and run it again
`the running release does not answer as v<old X.Y.Z> ...` or `the database of the release in <folder> is not running`	Start the installed release (`docker compose up -d` in its folder) and run it again
`<image>, which v<old X.Y.Z> runs, is no longer loaded, ...`	An image of the installed release was removed. One of ours: load its `images.tar` again (`docker load -i images.tar` in its folder). A Docker Hub image, named by `@sha256:` and not in `images.tar`: pull it again by that name (`docker pull <image>`)
`the backup failed: ...`	Fix what the backup's own line names (Take a backup)
`the backup did not report a passed restore test: ...`	The pre-upgrade backup was written but its restore test did not pass; the line quotes the backup's last line. Fix what it names, and run it again
`docker load could not load the new images.tar: ...` or `docker load left no ...`	The line gives Docker's own reason. Fix it, and run it again; the backup it took stays

When it fails after the start, it puts the previous release's containers back from its own folder and prints one line saying why, the command that restores the database as it was before the upgrade, and where it kept the attempt:

```
upgrade-server: FAIL: the upgrade to v<X.Y.Z> failed: the migration did not complete: ...; v<old
X.Y.Z>'s containers are running again
upgrade-server: to restore the database as it was before the upgrade, in <old folder> run:
scripts/restore-server.sh <backup>
upgrade-server: the new release and its record are kept in <parent>/winner-install-
<X.Y.Z>.failed-<UTC stamp>
```

If the containers cannot be put back, the first line ends `v<old X.Y.Z>'s containers could not be put back: <reason>` instead: start the previous release yourself with `docker compose up -d` in its folder, then read on.

Ctrl-C, or a TERM signal, after the start puts nothing back. It prints `interrupted while v<X.Y.Z> was starting; nothing was put back`, the `docker compose up -d` that puts the previous release back, with its folder, and the same restore command. From step 2 until the start, an interrupt is a refusal like any other.

Migrations do not reverse, so a failed one may leave the database in a state the previous release does not expect: if it misbehaves, run the restore command it printed (Restore), which takes the database back to the backup, and send the failure line to your Winner AI contact. When the new release carried another collector version, a line before the restore command says so: the previous release has just run its older collector over organisations' schemas the new release may have rebuilt, so run the restore before your team uses it.

4.7.2 Why an older release is refused

Migrations only go forward. The control plane migrates the database to its own release when it starts, so an older release would start on tables it does not know. The upgrade command therefore refuses a bundle older than the release this server runs, before its long check of `images.tar`, and the installer (`scripts/install-server.sh`) refuses one over a newer release that runs. The only way back is the rollback, below: it restores the pre-upgrade backup first, and starts the previous release only once that restore has passed. The installer sees only a stack that runs: over a stopped one it does not know which release last ran, and an older bundle is not refused.

The upgrade command's refusal begins:

```
upgrade-server: FAIL: the new bundle is v<X.Y.Z>, older than v<old X.Y.Z> which this server runs;
migrations do not reverse, ...
```

4.7.3 Going back to the previous release

Migrations do not reverse, so the release an upgrade replaced never starts on the database the newer one migrated. The rollback command puts both back: the database as it was before the upgrade, from the backup the upgrade took, then the previous release. From the folder of the release that runs now, as the user who owns its `.env`:

```
./scripts/rollback-server.sh
```

It reads `winner-upgrade.record` beside that folder for the previous release's folder and the pre-upgrade backup (`--bundle <folder>` and `--backup <file>` name others, and both are needed without a record). It checks the previous folder's signature and every file as the upgrade checks a new bundle, refuses a backup that is partial or marked failed, and changes nothing until you type the previous release back:

```
rollback-server: rolling back v<X.Y.Z> in <folder> to v<old X.Y.Z> in <old folder>
rollback-server: everything written since the upgrade at <time> will be replaced by <backup>:
people, Devices, settings and the sessions laptops pushed
rollback-server: the database as it is now is backed up first, to <folder of the backup>/pre-
rollback-v<X.Y.Z>-<UTC stamp>.sql.gz
rollback-server: a laptop enrolled since must run `winner login` again; a Device revoked since
comes back unable to log in and must be revoked again
Type v<old X.Y.Z> to roll back to it:
```

`--pre-rollback-backup <path>` writes that backup somewhere else; like any backup it must be outside both release folders. `--yes` skips the question. It then restores the backup with `scripts/restore-server.sh` (Restore), which keeps laptops out while it replays, and starts the previous release only once the restore has passed. It is done when that release answers with every migration applied:

```
rollback-server: the newer release and its record are kept in <parent>/winner-install-
<X.Y.Z>.rolled-back-<UTC stamp>; never run docker compose there
rollback-server: rolled back v<X.Y.Z> -> v<old X.Y.Z>; the database is as it was at <time>
rollback-server: the database as it was before the rollback is in <pre-rollback backup>
rollback-server: run docker compose from <old folder> from now on; point your nightly backup line
at it
rollback-server: a laptop enrolled since the upgrade must run `winner login` again; a Device
revoked since cannot log in and must be revoked again from the roster
```

With `--backup` naming another backup than the record's, or without a record, the two lines that give the upgrade's time name the backup instead: the first says `everything written since <backup> was taken will be replaced by it`, the last `the database is as it was when <backup> was taken`.

The newer folder is kept as `winner-install-<X.Y.Z>.rolled-back-<UTC stamp>` when the upgrade made it, and the first line says so; without a record it stays where it is, and the first line reads `the newer release stays in <folder>; never run docker compose there`. Never run `docker compose` there. A laptop enrolled since the upgrade lost its access in the restore and must run `winner login` again, and a Device revoked since comes back unable to log in: revoke it again from the roster.

Sessions laptops pushed between the upgrade and the rollback are gone from the archive, and laptops do not push them again: they are only in the pre-rollback backup. When the two releases pin different collector versions, the last line says so: a laptop that ran `winner update` against the newer release runs the newer collector, which fails every push until its next `winner update` takes it back down.

A refusal prints one `rollback-server: FAIL:` line saying `nothing was changed`. A restore that fails before it replays changes nothing either. One that fails once it has started replaying leaves laptops fenced out and the stack stopped, and prints the two commands to choose from: `scripts/restore-server.sh <pre-rollback backup>` in the newer release's folder, which puts the newer release back as it was, or `scripts/rollback-server.sh` there again, which restores the pre-upgrade backup once more. A failure after the restore has passed leaves the database fenced, so no laptop writes to a database half way between two releases, and says which step failed: fix it, then run `./scripts/rollback-server.sh --finish` in the same folder. A rollback cut off during its restore (a dropped SSH session or a reboot) is taken up by running `./scripts/rollback-server.sh` again, which restores once more under the fence it finds. An upgrade that failed, or was cut off before it finished, has nothing to roll back: from the folder of the release it started from, the command refuses even where an earlier upgrade's record says done, and names the restore that undoes the attempt from that attempt's own pre-upgrade backup. To roll back the upgrade before it anyway, name its previous folder with `--bundle` and its backup with `--backup`.

4.8 Troubleshooting

A symptom index for whoever runs a deployment. Each row names the likely cause, the command that confirms it and the section that fixes it; the fix itself lives on that page, or in the row when it is short. Each command here says where it runs: the server, the person's laptop, a shell whose Auth0 CLI is signed in to the tenant (`auth0 login`), or a machine outside the server.

On an installation from the bundle, read the server's commands as Commands on the other pages says: in `<install>` , the folder of the release that runs now, as the user who owns `<install>/.env` , with plain `docker compose` and no `sudo` . Rows that use the Auth0 CLI need a shell signed in to your tenant.

You see	Likely cause	Confirm with	Fix
Prompt Coaching, Skills or Benchmarks shows **Soon**	It is not in `FLEETVIEW_FEATURES` , which is empty by default, or the team switched it off on Settings	`grep ^FLEETVIEW_FEATURES= .env` (server); the Features on row on Settings	Switch Prompt Coaching on
Signing in ends on Auth0's error page about the callback or redirect address	The sign-in application does not list `<PUBLIC_BASE_URL>/oauth2/callback` , or `PUBLIC_BASE_URL` is not the address it lists	`curl -sS -o /dev/null -w '%{redirect_url}\n' https://<control-plane-host>/` (a machine outside the host) prints the address the gate sends browsers to, with its `redirect_uri`	On Auth0, Who may sign in, the callback and logout addresses; on another provider, Connect your identity provider
A laptop's pushes time out or are refused before any login, and `winner push status` names the failure	Port 5432 does not reach the server from where the laptop is: a firewall, or the database name resolving elsewhere; or a restore that failed left the database fenced; or, where you set up the Postgres jail, it banned the address (the jail row below)	`nc -vz <database-host> 5432` (laptop): a timeout is the network, a refusal the server; `docker compose exec db sh -c 'psql -U "$POSTGRES_USER" "$POSTGRES_DB" -c "SHOW listen_addresses"'` (server) prints `localhost` when fenced. A fence writes no log line, and a failed restore leaves the backend	Network; for a fence, Restore

You see	Likely cause	Confirm with	Fix
		stopped: the restore that left it printed `restore: FAIL`	
After a restore, one laptop's pushes are refused while others push	It was enrolled after the backup was taken, so the restored database does not admit its role	`winner push status` (laptop)	Restore: its owner runs `winner login` again
On a bundle install, browsers warn about the certificate or laptops fail on it, after weeks of working	The edge could not renew: port 80 or 443 closed to the certificate authority, or a name's DNS record changed. Nothing watches expiry on a bundle install	`curl -sSvo /dev/null https://<control-plane-host>/` (a machine outside the host) prints the certificate's `expire date`, or `certificate has expired`; `docker compose logs --tail 200 edge` (server)	Certificate renewals
On a bundle install, no new backup, or `backup.log` ends in `backup: FAIL` or `backup: refused`	The line says why: often counts that differed because a laptop pushed mid-run, a name that exists, a full disk, or a restore in progress	`tail -n 20 ~/winner-backups/backup.log` (server)	Take a backup
The AI runner's log shows its model pull failing	The server cannot reach the Ollama registry, or the storage host it redirects to	`docker compose logs --tail 100 ollama` (server)	Allow outbound https to the registry and its storage host, as What the server reaches lists them
A member sees "Manager role needed"	By design: the manager pages and the Dashboard are for managers; a member's Home shows only their own sessions, and they also have `/devices`, their own laptops' repos at `/repos`, their own findings at `/coaching`, their own candidates at `/skills` and	On Auth0, `auth0 api get organizations/<org id>/members/<user id>/roles` (Auth0 CLI); on another provider, the person's groups claim against the manager group and the e-mail list that Settings shows	Who is a Manager

You see	Likely cause	Confirm with	Fix
	their own patterns at `/benchmarks`		
Sign-in refused with "Your email address is not verified. Verify it, then sign in again."	The post-login Action: Auth0 has not verified the address, and the control plane did not create the account	`auth0 api get "users-by-email?email=<address, @ as %40>"` shows `email_verified` (Auth0 CLI)	The post-login Action
Sign-in refused with "This installation does not serve the organization you signed in with. Sign in through your own organization."	The post-login Action: the sign-in came through another Auth0 Organization or none, or the Action's `AUTH0_TEST_ORG_ID` secret holds another installation's value	`auth0 api get actions/actions/<action id>` (Auth0 CLI) lists the secrets' names, never their values, under `deployed_version.secrets`; Auth0 shows no value back, so run the update with `AUTH0_PROXY_CLIENT_ID` and `AUTH0_TEST_ORG_ID` from `<install>/.env`	The post-login Action
Sign-in refused with "Sign-in is not configured for this installation. Ask its operator to set the post-login Action's secrets."	The post-login Action was deployed without `AUTH0_PROXY_CLIENT_ID` or `AUTH0_TEST_ORG_ID`, or with one empty, and refuses every sign-in on the tenant	`auth0 api get actions/actions/<action id>` (Auth0 CLI) lists the secrets' names under `deployed_version.secrets`	The post-login Action: run the update with both values
`winner login` fails with 403 "user is not a member of the organization"	A `winner` from before the control plane's own login enrolled a person who is not a member of the bootstrap organisation, the one its Auth0 device flow resolves	`auth0 orgs members list <org id>` (Auth0 CLI)	Run the installer again on that laptop for a current `winner`, whose login goes through the control plane: Install on macOS, Install on Ubuntu or Debian. Why: Members and the first team
The hostname reporter's log says `the control plane no`	The laptop's device was revoked, its person was	`winner report` (laptop) prints the same line and exits 1 after one request	Run `winner login` on that laptop: Sign in. If its sign-in page says

You see	Likely cause	Confirm with	Fix
`longer accepts this laptop's pass`	removed, or a later login replaced its pass		the account was removed, ask a Manager: Leaving
`winner login` says "this client is older than the server's sign-in", or, from a client up to 0.12.0, that the control plane "returned an incomplete config"	The laptop's `winner` predates the sign-in the control plane serves	`winner update --check` (laptop) names the control plane's `winner` version; up to 0.12.0 it fails the same way	Run the update step the message names; up to 0.12.0, run the installer again: Install on macOS, Install on Ubuntu or Debian
`winner login` says "Could not list this laptop's repos (... listing repos needs git 2.36 or newer), so none are chosen and nothing is sent", `winner repos` fails with the same reason, or `winner login` says "Kept sending every repo: could not read every name agentsview holds (... needs git 2.36 or newer)"	The laptop's git is missing or older than 2.36, as stock git is on Ubuntu 20.04 and 22.04 and Debian 11; "could not ask git its version (...)" means git itself failed, which on a Mac is usually `/usr/bin/git` without the Command Line Tools	`git --version` (laptop)	Requirements says how to get git 2.36 or newer; then `winner repos pick` (laptop), or `winner login` where it kept sending every repo
`winner login` refuses because this laptop's agentsview is newer than the control plane's	The control plane's collector version is the ceiling a laptop may run	`winner update --check` (laptop) reports both versions on any platform; the install it previews runs only on macOS and Linux, amd64 or arm64	The collector version ceiling
The roster shows a laptop Stale, or its sessions stop arriving	The background push stopped, or its pushes fail	`winner push status` (laptop)	Day to day
Support asks what a laptop's push is doing	Support needs the laptop's versions, push state and recent push log	`winner diagnostics` (laptop) writes one file to attach	Collecting diagnostics

You see	Likely cause	Confirm with	Fix
A laptop's pushes fail on a refused connection, on a server where you set up the Postgres jail (the bundle ships none)	The jail banned its address after repeated refused logins, usually a stale device password	`sudo fail2ban-client status fleetview-postgres` (server; fail2ban is the host's, not the stack's, so this one needs `sudo`), and `fleetview-postgres-vpn` where you added it	The Postgres jail
The jobs sweep's dead-man's switch (`FLEETVIEW_HEARTBEAT_URL`) pages	No ping arrived: the jobs service stopped or hangs. Or a ping reported a failure: a sweep that raised, a lost database included, or one in which an organisation's sweep raised. A single pass that fails is logged (`pass <name> failed for <org>`) and does not fail the ping	`docker compose ps --all` and `docker compose logs --tail 500 jobs` (server); a failed sweep logs `jobs sweep failed`	`docker compose up -d` (server) restarts a stopped service; a failed sweep names its reason in the log
The edge serves no valid certificate, so browsers warn and laptops cannot push with `verify-full`	The edge could not obtain a certificate over ACME: the hostname has no public DNS record, port 80 is not reachable from the internet, or a CAA record refuses the authority	`docker compose logs --tail 200 edge` (server) shows its certificate attempts; from outside the server, `curl -sSI http://<control-plane-host>/` answers 308 when port 80 is reachable	Public deployment
No sign-in page in the `dns` certificate mode, and the edge log says `edge: no certificate from the dns-certs helper yet`	The `dns-certs` helper holds no certificate yet: a wrong `AWS_HOSTED_ZONE_ID`, an instance role without the zone, no outbound https to Let's Encrypt or the Route 53 API, or a CAA record that refuses Let's Encrypt	`docker compose logs --tail 200 dns-certs` (server) carries lego's error	Private VPC: the certificate through DNS, Certificate renewals

You see	Likely cause	Confirm with	Fix
A laptop on the VPN cannot resolve the host, or its pushes fail with the name not found	Its VPN client did not apply the pushed resolver, or the split-tunnel routes do not carry the resolver	`getent hosts <control-plane-host>` on Linux, `dscacheutil -q host -a name <control-plane-host>` on a Mac (laptop, on the VPN), must print the private address	AWS Client VPN
The backend stops at start with `Auth0 tenant <domain> is missing the role(s) ...`	The tenant has no role named `manager` or `member`, which member creation resolves by name; with Management credentials the control plane checks both at start and refuses to start without them	`docker compose logs --tail 200 backend` (server)	Create the missing role in the Auth0 Dashboard under User Management, Roles, as the message says, then `docker compose up -d backend` (server)
The installer or an upgrade fails with `an org schema holds code stock agentsview does not create`, or the backend log names `org schema … holds … owned by …`	A laptop's device role planted a function, view, trigger or other code object in the organisation's schema, or one was left behind	The findings the line lists	Code in an org schema that agentsview did not create
Home's Team spent reads "unavailable"	The backend's spend read from `pg serve` is failing or its stored figure is stale, or the organisation has no `pg serve`; the tile's note says which	`docker compose logs --tail 200 backend` (server) shows the `pg serve … summary:` line with the reason	The read waits 120 seconds, so a timeout there is a `pg serve` or database problem, not a slow page; start with the `pg-serve` and `db` services' logs
Every Dashboard change (star, pin, rename, trash) fails with `cross-site request refused by the edge`	People browse an address that is not `PUBLIC_BASE_URL`, which the web server compares with each write's `Origin` exactly	`grep ^PUBLIC_BASE_URL= .env` (server), against the browser's address bar	Open the server at `https://<PUBLIC_HOST>`, the only `PUBLIC_BASE_URL` the `.env` check accepts on an https install; changing

You see	Likely cause	Confirm with	Fix
			`PUBLIC_BASE_URL` would also move the sign-in callback
The jobs log's `rows written this sweep` line counts more than 0 `passes had no model to call`	No usable model: the host does not lend its model and the team set no key, the team's model is incomplete, the team key cannot be read with this host's `FLEETVIEW_SETTINGS_KEYS`, or, with no model lent by the host, the team endpoint is one of the installation's own services	The Model rows on Settings say why; `docker compose logs --tail 500 jobs` (server) names each pass that was `skipped for` or `runs without a model for` an organisation, with the reason	Switch Prompt Coaching on; Team settings
`fleetview-model-check` ends `verdict: fail` with `refused:` or `the model call failed:` on the line directly above	In that order: the endpoint is neither https nor a compose-internal host, so the key was never sent, or under `FLEETVIEW_LLM_PROVIDER=bedrock` the region is not an AWS region or the endpoint is not an https Bedrock runtime host; the endpoint refused the calls, with the status it answered, a call ran past `FLEETVIEW_LLM_TIMEOUT_SECONDS`, or under Bedrock `no AWS credentials` found neither a key nor an instance role	The command itself, in the jobs container (server)	Check the model before Prompt Coaching; Amazon Bedrock

You see	Likely cause	Confirm with	Fix
Home says "No organization yet."	No organisation provisioned on this deployment holds the person: the bootstrap organisation is not set or names another organisation, startup provisioning failed, or the person has no row here and none of the Auth0 organisations they belong to is provisioned here	`docker compose logs --tail 200 backend` (server) shows `startup provisioning failed` or `the installation's Org was not provisioned: ...` when it did; `auth0 api get "users/<user_id>/organizations"` (Auth0 CLI) lists the person's organisations	Authentication, the `AUTH0_TEST_ORG_ID` row, which is the `.env` key behind the `FLEETVIEW_BOOTSTRAP_ORG_ID` the page names. The control plane cannot create an organisation: create it in your provider, add its people, and set its id there
The Dashboard answers 403 "this installation serves another organization"	The person manages an organisation other than the one `AUTH0_TEST_ORG_ID` names; an installation serves that one alone	`docker compose logs --tail 200 backend` (server)	By design: an organisation this installation served before keeps its rows, and its people still sign in, but it has no dashboard and no spend
The Dashboard answers 502 and `pg-serve` restarts with "names no schema for this installation's Org"	The backend has not provisioned the Org the key names: the key is unset, or startup provisioning failed	`docker compose logs --tail 50 pg-serve` and `docker compose logs --tail 200 backend` (server), whose log shows `startup provisioning failed` or `the installation's Org was not provisioned: ...` when it did	Authentication
The Dashboard answers 503 "No dashboard is provisioned for your organization."	The web server and `pg serve` were started for a different `AUTH0_TEST_ORG_ID` than the backend now holds: the key changed and only the backend was restarted. A `pg serve` still running for another organisation loses	Bring the whole stack up again: `docker compose up -d` (server)	Authentication, the `AUTH0_TEST_ORG_ID` row

You see	Likely cause	Confirm with	Fix
	its database login at the backend's start, so until the whole stack is brought up again spend reads as unavailable too: the backend reads no spend while another organisation's serve role can still log in or is connected		

4.8.1 Code in an org schema that agentsview did not create

An organisation's stock schema holds two functions, three triggers and its own CHECK constraints, and nothing else that runs code. Any other function, view, trigger, rule, operator, policy, CHECK constraint, generated column or CHECK on a domain an org column uses there was planted by a laptop's device role, and runs with the rights of whoever reads or writes past it, a restore included. The owner it reports is not proof either way. The installer and each upgrade refuse to finish on one, and the backend and jobs log each at ERROR on every start. A view, materialized view, foreign table or policy also stops that team's reads: Home and the roster's session counts show nothing, and the Coaching, Skills and Benchmarks pages fail, until it is cleared. Row-level security switched on with no policy is not a finding.

Nothing is dropped for you: whether the object is an attack or a leftover is a judgement. List and read them in `<install>`:

```
docker compose exec -T backend uv run --no-dev fleetview-provision audit   # one line per finding; changes nothing
docker compose exec db sh -c 'psql -U "$POSTGRES_USER" "$POSTGRES_DB"'      # then \df+, \do+ or \d+ <schema>.<name> to read it
```

1. **Revoke the laptop.** Where the owner is a device role (`dev_…`), revoke that laptop on Admin first, so it cannot plant the object again.
2. **Drop each object you judge planted**, with the `DROP` for the kind the audit printed. This step is only for a code object: a line that starts with a team's schema and names the object's kind second (a function, view, trigger and the like). The grant lines and the `-` lines below are never cleared with a `DROP`. Never drop or recreate the schema itself: it is your team's archive, and nothing restores it short of a backup. Where Postgres refuses and suggests dropping an extension instead, do not: that removes everything the extension holds. Write to support with the message.
3. **Run again what refused:** the installer, the upgrade, or the backup or restore command.

A rollback that refuses on this leaves the database fenced and the stack stopped, so the audit above cannot run. Write to support with the rollback's lines before you change anything; once the object is cleared, in the folder of the newer release, run `scripts/rollback-server.sh --finish`.

The same audit also prints `grant`, `grant under another grantor`, `excess grant`, `default grant` and `foreign owner` lines: one laptop gave another, or a role that is not the team's, rights over the team's tables. A `default grant` line is the same right on everything one of the team's roles creates later; the line says whose and what. The backend's start revokes a `grant` or `default grant` it can, so one still listed was made since or was refused at that start. Clearing these takes superuser statements specific to each line, so write to support with the audit's lines, and revoke the laptop the line names on Admin meanwhile.

A line whose first field is `-` is about the whole database server, not a team's schema:

- `database postgres public=CONNECT` or `database template1 public=CONNECT`: every role may connect to one of Postgres's own databases, which no laptop's role should.
- `function pg_catalog.lo_...(...) public=EXECUTE`: every role may run one of Postgres's own large-object functions, through which laptops of two organisations could pass data.

These are Postgres's own databases and functions, not something a laptop planted: **never drop or alter them**. The backend takes both rights away at every start, so a line here means a right came back since (a restore into a new server brings both back) or that start's revoke failed, which the backend log records at ERROR (`could not revoke ...`, `... stays executable by every role`, or `database ... stays open to every role`). Restart the backend with `docker compose restart backend`, then run the audit again. If a `-` line stays, write to support with it and the backend's ERROR line.

When in doubt about any line, write to support before you drop anything.

4.8.2 Getting help

When a row here does not settle it, write to support as Where to write says. Add what the Confirm column above showed, except log output, and the exact line a failed `backup-server.sh` or `restore-server.sh` printed. Attach the server's redacted diagnostics file, as Collecting a diagnostics file says, after reading it yourself. Do not attach raw server logs, which can name people, laptops and your server, and never send `.env`, a backup, a key or a password.

4.8.3 What changed in a release

Each release's changes are on the changelog; its **Requires of you** part says what you must change and whether your people must reinstall the client or sign in again. The guides an installation serves at `/guide/` are its own release's, and each bundle carries the same pages as `docs/winner-ai-docs-<X.Y.Z>.pdf` inside the unpacked folder, so a new bundle's PDF says what changes before you install it.

4.9 Connect your identity provider

How people sign in to Winner AI through the identity provider your organisation already runs, who becomes a Manager, and how people join and leave. It is written for the operator who installs the server and the administrator of the provider, who are often the same person, and it needs no help from us.

The sign-in gate admits anyone your provider admits and proves who they are, and decides no role; the control plane decides the role on every request. A **Manager** sees the whole organisation, its roster and the dashboard. A **Member** sees only their own, and every manager page refuses them.

> **Another provider means no Auth0 Management credentials**
>
> A production installation signs in through any OpenID Connect provider, JumpCloud among them, with `AUTH0_MGMT_CLIENT_ID` and `AUTH0_MGMT_CLIENT_SECRET` blank. With them set, the control plane finds each person by e-mail in Auth0, so `scripts/check-env.sh` and the control plane refuse an https installation whose issuer is not Auth0's. An installation with Auth0 Management credentials reads roles from Auth0; with them blank the group rules below apply, on Auth0 too.

On an installation from the bundle there is no `make` : `scripts/install-server.sh` runs `scripts/check-env.sh` against `.env` before it loads or starts anything, and stops on a refusal with nothing started, so `make check-env` and `make up` below are what a checkout of the source runs instead. Commands on the other pages says how to read them, and who runs the server's commands, once the server is installed.

4.9.1 Before you install

This is the checklist the install asks you for. Have every answer to hand, and the install needs nothing else from us. Where the provider offers a choice, the answer is the one that gives the control plane a stable identity and a group claim it can read.

You need	Why
Whether your provider's plan includes single sign-on with a custom OpenID Connect application	The gate is an OpenID Connect client. A plan with no way to register one cannot sign people in
The provider's **issuer** address, and its region where it has one	`OAUTH2_PROXY_OIDC_ISSUER_URL` . A provider with a region per organisation may have one issuer per region
Whether the server can reach the issuer over https from where it runs	The gate fetches the provider's discovery document, tokens and keys from the server, not from a browser

You need	Why
An application at the provider for the gate, with its **client id** and **client secret**	`AUTH0_PROXY_CLIENT_ID` and `AUTH0_PROXY_CLIENT_SECRET` hold them, whichever provider they belong to
The name of the **group** whose members are Managers	`FLEETVIEW_MANAGER_GROUP`
Whether that group is **bound to the application**	Some providers put only the groups that connect a person to the application in the token, so a manager group that is not bound never arrives
The name of the **attribute or claim** that carries group membership, and whether it holds a list or one string	`OAUTH2_PROXY_OIDC_GROUPS_CLAIM`. Some providers let the administrator name it; there is no fixed name
Whether the provider sends `email_verified`, and whether people can choose their own address	Decides `OAUTH2_PROXY_ALLOW_UNVERIFIED_EMAIL` and whether the manager e-mail fallback is safe
The provider's **end-session** address	`OIDC_END_SESSION_URL`, where sign-out ends the provider's own session
Whether an access policy applies to the application, such as a managed-device or network condition	A laptop is approved in a browser. A developer whose browser the policy refuses cannot finish approving
The public address people browse to	`PUBLIC_BASE_URL`, which every redirect below is built from, and which a production installation sets to https
A key for the installation's one organisation	`AUTH0_TEST_ORG_ID`: on Auth0 the Organization's id (`org_...`), on any other provider any name you choose for the installation, set once; see One organisation

4.9.2 How sign-in works

1. A person opens the address. The gate, oauth2-proxy, sends them to your provider, and on their return checks the token and the issuer.
2. The gate admits them if the provider did. Who may sign in at all is the provider's decision: assign the application to the people or the groups you want.

3. The gate forwards the person's permanent id, e-mail and groups to the control plane, which trusts them because the gate is the only ingress.
4. On the first page they open, or their first laptop approval, the control plane finds or makes the person's row. **Without Auth0 Management credentials**, the row is keyed on the issuer and the provider's permanent id together, never on the e-mail; another provider presenting the same id is another person, and the control plane recomputes the role from the groups on every request after. **With them**, which only an Auth0 installation sets, the control plane finds the person by e-mail in Auth0, and the role comes from Auth0, kept current by its role sync (Onboard on Auth0).

4.9.3 Set up any OpenID Connect provider

At the provider

Create an application for the gate:

- A web application using the authorisation code flow, authenticating with a client secret.
- The **redirect address**: `<PUBLIC_BASE_URL>/oauth2/callback`, for example `https://winner.example.org/oauth2/callback`.
- The scopes `openid`, `profile` and `email`, and `groups` where the provider puts groups in the token only when asked.
- The groups attribute included in the ID token, under a claim name you choose and write down.
- The **post-logout redirect** address, where the provider has one: `PUBLIC_BASE_URL`.
- The permanent identifier left at its default. A provider that lets `sub` be remapped to an attribute people can change breaks the row each person is keyed on.
- The application assigned to the people who may sign in, and the manager group bound to it.

Keep the issuer once people have signed in. Without Auth0 Management credentials, the case on any provider but Auth0, a person is the issuer and the permanent id together, so a new issuer, another provider or another region of the same one, meets everyone as a new person with no laptops, in the role their first request's groups give them (or the manager e-mail list, where that decides), while their old rows keep their history. Nobody is matched to an old row by e-mail.

In `.env`

These settings sit in the installation's `.env`. The gate's issuer, groups claim and whitelist, left blank, mean Auth0's values; the scopes mean `openid profile email`; the end-session address and the two manager settings mean none.

Setting	Holds
`AUTH0_PROXY_CLIENT_ID`, `AUTH0_PROXY_CLIENT_SECRET`	The gate's application at your provider. They keep their Auth0 names
`OAUTH2_PROXY_OIDC_ISSUER_URL`	Your provider's issuer
`OAUTH2_PROXY_SCOPE`	The scopes the gate asks for. `openid profile email`, plus `groups` where the provider needs it
`OAUTH2_PROXY_OIDC_GROUPS_CLAIM`	The claim the gate reads groups from
`OIDC_END_SESSION_URL`	The provider's end-session address
`OAUTH2_PROXY_WHITELIST_DOMAINS`	The hosts the gate may send a browser to after sign-out: the end-session address's host, comma-separated. A leading `.` or `*.` admits subdomains
`OAUTH2_PROXY_ALLOW_UNVERIFIED_EMAIL`	`false` where the provider's `email_verified` reflects a real check and people cannot choose their address. JumpCloud's is a constant, below
`FLEETVIEW_MANAGER_GROUP`	The group whose members are Managers
`FLEETVIEW_MANAGER_EMAILS`	Comma-separated fallback addresses, below
`AUTH0_MGMT_CLIENT_ID`, `AUTH0_MGMT_CLIENT_SECRET`	Blank on any provider but Auth0: an https installation refuses them with another issuer
`AUTH0_DOMAIN`, `AUTH0_AGENT_CLIENT_ID`	On a provider other than Auth0, placeholders that resolve nowhere (`unused.invalid`, `unused`): `scripts/check-env.sh` still requires both

```
# the shape of the block, with placeholders: nothing here is a value to copy
AUTH0_PROXY_CLIENT_ID=<client id>
AUTH0_PROXY_CLIENT_SECRET=<client secret>
OAUTH2_PROXY_OIDC_ISSUER_URL=https://<issuer>
OAUTH2_PROXY_SCOPE=openid profile email groups
OAUTH2_PROXY_OIDC_GROUPS_CLAIM=<the claim name>
OIDC_END_SESSION_URL=https://<end-session address>
OAUTH2_PROXY_WHITELIST_DOMAINS=<the end-session address's host>
OAUTH2_PROXY_ALLOW_UNVERIFIED_EMAIL=false
FLEETVIEW_MANAGER_GROUP=<the manager group's name>
```

Check `.env` before anything starts: on a server installed from the bundle, `scripts/install-server.sh` runs `scripts/check-env.sh` itself (Install the server, step 5); in a checkout of the source, run `make check-env`. It refuses an end-session address whose host the gate would not send a browser to, a whitelist entry that covers a whole public suffix (`.com`, `*.co.uk`), and on an https installation an issuer or end-session address that is not https. Each refusal names the values it compared.

The scopes are always passed to the gate: `openid profile email` unless you write others. A provider that puts groups in the token only when asked needs `groups` written out; a provider that lists no such scope may refuse the sign-in if you add it.

Who is a Manager

Without Auth0 Management credentials, the control plane reads the role from the groups the gate forwards:

- A person whose groups include `FLEETVIEW_MANAGER_GROUP` is a **Manager**. The match is exact: a group called `all managers` does not make its members Managers where the group is `managers`.
- The gate forwards a list claim's groups, on one header line or several, and a string claim as it is; the control plane joins any lines with commas and splits on commas. A group name with a comma in it cannot be the manager group, and spaces inside a name are kept.
- Where `FLEETVIEW_MANAGER_GROUP` is set, a person whose groups arrive and do not include it is a **Member**, even if their address is on the e-mail list: the group alone decides.
- The e-mail fallback decides only where sign-in brings **no usable groups claim**: none at all, an empty one, or one in a form the control plane cannot read; or where `FLEETVIEW_MANAGER_GROUP` is blank. An address in `FLEETVIEW_MANAGER_EMAILS` is then a Manager, and any other a Member.
- A groups value that reads as something other than text, such as `123` or `true`, counts as no claim. So a manager group named only with digits never matches for a person in no other group: give it a name with a letter in it.
- A person who has been removed stays removed. Neither the group nor the list brings them back.

The e-mail list trusts any address the gate admits. `scripts/check-env.sh` therefore refuses a `FLEETVIEW_MANAGER_EMAILS` unless `OAUTH2_PROXY_ALLOW_UNVERIFIED_EMAIL` is `false`. `false` refuses a sign-in whose `email_verified` is false; a provider that sends no `email_verified` is admitted either way, so the list is safe only on a provider where nobody can choose another person's address.

A change to `FLEETVIEW_MANAGER_EMAILS` or `FLEETVIEW_MANAGER_GROUP` takes effect at the next request once the backend has been recreated with the new `.env`: in `<install>`, run `./scripts/check-env.sh`, then `docker compose up -d`. A change at the provider does not reach the control plane at all until the person signs in again: the groups are those of their last sign-in, and the gate's session lasts `FLEETVIEW_GATE_SESSION_SECONDS`, an hour by default, before the gate sends the person back to the provider. A person taken out of the manager group stays a Manager until then.

Check it

1. Sign in as someone in the manager group. Home, Admin and the Dashboard open, and Settings shows the manager group, the e-mail list and the groups claim under Features and host.

2. Sign in as someone outside it, in a private window. Their Home opens on their own sessions ("Your sessions."), and every manager page says the Manager role is needed.
3. Open Admin as the Manager. Both people are on the roster.

You see	Likely cause
The sign-in page of the provider says the redirect address is not registered	The address at the provider is not `<PUBLIC_BASE_URL>/oauth2/callback` exactly
A Manager is told the Manager role is needed	The group never arrived: it is not bound to the application, the claim name differs from `OAUTH2_PROXY_OIDC_GROUPS_CLAIM`, the scope that brings it is missing, or the group's name differs from `FLEETVIEW_MANAGER_GROUP` by a character
Everyone is a Manager	The manager group is a broad one every person is in
Everyone is a Member	The e-mail list is deciding, because `FLEETVIEW_MANAGER_GROUP` is blank or the sessions carry no usable groups claim, and the person is not on it
Sign-out leaves the person signed in	The provider keeps a session of its own that the end-session address does not end (on JumpCloud, the User Portal's: JumpCloud), or the provider wants an `id_token_hint`, which the control plane never sends (Sign-out). Less often, `.env` was changed without `./scripts/check-env.sh` and the end-session address's host is not in `OAUTH2_PROXY_WHITELIST_DOMAINS`, so the gate dropped the redirect

Sign-out

The control plane's sign-out link clears the gate's cookie and, where `OIDC_END_SESSION_URL` is set, sends the browser on to the provider's end-session address with the gate's client id and the public address to return to. The control plane never holds the ID token, so it sends no `id_token_hint`; a provider that insists on one signs the person out less completely.

4.9.4 One organisation

An installation serves one organisation, configured once. Every person the provider admits belongs to it, and nothing at sign-in or enrolment names an organisation.

- Its key is `AUTH0_TEST_ORG_ID` in `.env`, which the control plane reads as `FLEETVIEW_BOOTSTRAP_ORG_ID`. It keeps the Auth0 name in this release, whichever provider you use.
- On Auth0, the key is the Organization's id (`org_...`). On any other provider it is any name you choose for the installation, set once; the rule below applies to it.

- The key names the organisation's database schema, once, at the first start. It needs a letter or a digit, and must make a schema of at most 56 bytes. After that the schema is stored and never derived again, so no schema is renamed, dropped or recreated.
- Choose the key once, for the installation's life. Changing it does not move anyone: each person who signed in before stays in the organisation they were first recorded in, so a Manager among them gets a 403 at the Dashboard.
- A change of key needs the whole stack brought up again with `docker compose up -d`, which recreates the web server and the dashboard's `pg serve`. Restarting only the backend leaves the Dashboard answering 503 until you do, because the web server admits only the organisation it was started with.
- A manager of any other organisation the control plane once provisioned gets a 403 at the Dashboard.

4.9.5 A trial with the bundled Dex

For an evaluation with no provider of your own, the install can run Dex, a small stock OpenID Connect provider, beside the gate. It is for loopback only and **production refuses it**.

- It holds two accounts in one file: `manager@example.com`, in the manager group, and `member@example.com`, in none. Both have the password `winner-trial`.
- It is switched on by the `trial` compose profile, and reached at `http://localhost:4180` and nothing else, since Dex knows only that address.
- `scripts/check-env.sh` and the control plane both refuse an https base URL with the trial on, or the bundled Dex as issuer, so a trial cannot become a production sign-in by a mistake in either.
- The trial needs no Auth0 values beyond placeholders for the gate's client and the one organisation, and the Auth0 Management credentials stay blank: that is what makes the control plane read roles from Dex's groups.
- Sign-out clears the gate's cookie and nothing else, since Dex has no end-session endpoint. Whether Dex keeps its own browser session after that has not been checked.

Add these lines to `.env` and put `trial` in `COMPOSE_PROFILES`. On a server installed from the bundle, then run `./scripts/install-server.sh`, which checks `.env` and installs the trial on loopback, with no edge and no public port (Install the server, "A trial from the bundle"). In a checkout of the source, run `make check-env`, then `make up`:

```
OAUTH2_PROXY_OIDC_ISSUER_URL=http://localhost:4180/dex
OAUTH2_PROXY_SCOPE=openid profile email groups
OAUTH2_PROXY_OIDC_GROUPS_CLAIM=groups
FLEETVIEW_MANAGER_GROUP=manager
OAUTH2_PROXY_SKIP_OIDC_DISCOVERY=true
OAUTH2_PROXY_LOGIN_URL=http://localhost:4180/dex/auth
OAUTH2_PROXY_REDEEM_URL=http://dex:5556/dex/token
OAUTH2_PROXY_OIDC_JWKS_URL=http://dex:5556/dex/keys
```

The last four exist because the gate cannot read Dex's discovery document at start: Dex's issuer is the gate's own address, which does not answer until the gate is up. The trial names the endpoints instead, and `scripts/check-env.sh` refuses all four on an https base URL.

Why production refuses it. The users and their password hashes ship in a file anyone holding it can read. The page for Dex's local accounts documents no second factor, lock-out or password reset, and nobody at your organisation administers it, so a leaver stays until the file is edited. Its ID tokens last 24 hours by default, as measured on a stock Dex. And a bundled issuer is a second identity system beside the one you already run.

4.9.6 How people join and leave

Joining

1. **At the provider**, assign the application to the person or to a group they are in. If they should be a Manager, put them in the manager group, and make sure that group is bound to the application.
2. **They sign in.** The first page they open, or their first laptop approval, makes their row, keyed on the issuer and the provider's permanent id, with the role the rule above gives them. Until then they are not on the roster, so a person who has never signed in is not shown.
3. **They install the client** (Laptop client guide) and run `winner login`, which prints a code and an address. They open the address in any browser, confirm the code, and the terminal carries on and the laptop is enrolled. That login is the control plane's own: the person approves the laptop at `/login/approve` behind the gate, and the laptop is enrolled as them, in the installation's one organisation. A Member's laptop pushes sessions into the organisation's database like a Manager's, and the roster shows it.

A Member who signs in sees their own Home: the figures of their own sessions, with a link to a page of their own laptops that shows whether each is still enrolled, and one of their laptops' repos. Where `FLEETVIEW_FEATURES` switches them on, and while the licence or its grace allows them (Licence), they also see a Prompt Coaching page of the findings on their own prompts; a Skills page that lists the repeated workflows found in their own prompts and offers the approved Skills made from them; and a Benchmarks page of their own prompt patterns, with what a cheaper model did when it re-ran one. They read their own sessions on their laptop, in the session collector (`agentsview`), and the web Dashboard is managers only.

To start a new installation, have one person in the manager group sign in first, then assign the application to everyone else; the roster fills as they sign in.

Adding someone at the provider is the whole of it. Admin's Add member creates the person in Auth0, so it is for an Auth0 installation only; on any other provider the form is still shown and cannot work, and the control plane can create no one there.

Leaving

Removal at the provider stops **new** sign-ins. It does not stop what the person already has:

- Their web session at the gate lasts until its sign-in ends, at most `FLEETVIEW_GATE_SESSION_SECONDS`, an hour by default.
- A laptop that is already enrolled keeps sending. A laptop pass has no fixed expiry; it ends when a Manager removes the person or revokes the laptop.

So a Manager removes a leaver on **Admin**:

- **Remove**, on the person's row, ends their laptop passes, drops each of their laptops' database logins and marks the row `revoked`. A laptop's next report is refused, and its next push fails. Nothing is deleted: the person and their laptops stay in the archive, so history and attribution are unchanged.
- A removed person who signs in is told they were removed from the team. Signing in again does not bring them back, and neither does being in the manager group.
- **Restore**, on a removed person's row, makes them a Member again with no laptop. They run `winner login` on their laptop again to enrol it, and the group or the e-mail list may make them a Manager again on their next request. Restore refuses while any of their laptops is still enrolled; Remove finishes that, and is also how a removal whose database login could not be dropped is retried.
- **Revoke**, on one laptop, ends that laptop's pass and its database login only. The person's other laptops keep reporting.
- A Manager cannot remove themselves, and two Managers removing each other at once leave one.

Remove and Restore appear only where no Auth0 Management credentials are set, because an installation that has them runs a sweep that would write a removal back to whatever Auth0 says. There, Admin says people are removed in Auth0, and the next sync revokes their laptops (Onboard on Auth0).

4.9.7 JumpCloud

A worked example on JumpCloud. The JumpCloud administrator makes the application with JumpCloud: before the install, which reads on its own, and hands back the values in its last table; this section is the operator's side, from those values to the first sign-in. Both were followed on a live organisation on 2026-11-03, and what that first run showed is at the end.

Decide the public address people will browse to, `<base URL>`, before the administrator starts: it is `PUBLIC_BASE_URL`, and the application is registered against it.

In `.env`

Put the values the administrator handed back in `.env`; the checklist's last table says which value fills which setting. The block below is for the United States region, the only one tried. In the European Union and India regions the issuer is `https://oauth.id.eu.jumpcloud.com/` or `https://oauth.id.in.jumpcloud.com/`, but the User Portal's logout address there has never been tried, and the install cannot start without an end-session address: contact Winner AI support before the install.

```
AUTH0_PROXY_CLIENT_ID=<Client ID>
AUTH0_PROXY_CLIENT_SECRET=<Client Secret>
OAUTH2_PROXY_OIDC_ISSUER_URL=https://oauth.id.jumpcloud.com/
OAUTH2_PROXY_SCOPE=openid profile email
OAUTH2_PROXY_OIDC_GROUPS_CLAIM=<group attribute name>
FLEETVIEW_MANAGER_GROUP=<Managers group name>
OIDC_END_SESSION_URL=https://console.jumpcloud.com/userconsole/logout
OAUTH2_PROXY_WHITELIST_DOMAINS=console.jumpcloud.com
```

The end-session address is JumpCloud's User Portal logout, not the OIDC one in the issuer's discovery document. The session that signs a person straight back in is the User Portal's, and only its logout ends it. The OIDC address refuses a sign-out with a return address and no `id_token_hint`, and without them it ends the OIDC session but leaves the User Portal's. Sign-out then lands on JumpCloud's sign-in page.

The scopes carry no `groups`: JumpCloud's discovery document lists `openid`, `profile`, `email`, `offline` and `offline_access` and no `groups` scope, and the group claim arrives through the group attribute the administrator named on the application. Leave `OAUTH2_PROXY_ALLOW_UNVERIFIED_EMAIL` unset and do not use the e-mail fallback with JumpCloud: What the first run showed says why.

`AUTH0_DOMAIN` and `AUTH0_AGENT_CLIENT_ID` name no Auth0 tenant here, but `scripts/check-env.sh` still requires a value for each: write placeholders that resolve nowhere.

```
AUTH0_DOMAIN=unused.invalid
AUTH0_AGENT_CLIENT_ID=unused
AUTH0_MGMT_CLIENT_ID=
AUTH0_MGMT_CLIENT_SECRET=
```

The control plane reads the manager group only where `AUTH0_MGMT_CLIENT_ID` is blank, which `scripts/check-env.sh` accepts with `FLEETVIEW_MANAGER_GROUP` or `FLEETVIEW_MANAGER_EMAILS` set. An installation on JumpCloud leaves the Management keys blank: `scripts/check-env.sh` and the control plane refuse them there.

Check the JumpCloud sign-in

Sign in as a person in the Managers group and as one in the Members group. As in Check it above, the Manager reaches Home, Admin and the Dashboard, and the Member's Home opens on their own sessions ("Your sessions."), and every manager page says the Manager role is needed.

A laptop enrols through the control plane's login: each person runs `winner login` and approves their own laptop's code (How people join and leave).

What the first run showed

A first install on a live JumpCloud organisation, on 2026-11-03, answered what this page had left open:

1. **Scopes.** `openid profile email`, with no `groups` scope, signs people in, and the group attribute named on the application arrives.
2. **`email_verified`.** With the Email scope ticked, the console adds a constant attribute, `email_verified` with the value `true`, which cannot be removed. Every user arrives verified, whatever happened to their address. So the claim proves nothing on JumpCloud: the manager e-mail fallback is not safe there, and the allowance for an unverified e-mail stays on (the release default), which refuses nobody.
3. **Sign-out.** It ends the JumpCloud session only through the User Portal's logout, the end-session address in the block above. After signing out, opening the address asks for the JumpCloud password again.

The checklist marks, on the step each belongs to, the console details its sources did not state, among them which client authentication type the gate needs.

4.10 JumpCloud: before the install

This page is for the administrator of a JumpCloud organisation that will run Winner AI. It makes one application in the JumpCloud Admin Portal, the one the installation's sign-in gate uses, and ends with a table of the values to give back to whoever runs the installation. Nothing on it needs the installation to exist yet.

`<base URL>` below is the address people will browse to, for example `https://winner.example.com`. Whoever runs the installation decides it; ask them for it before you start. It is always https.

Each step ends with a line in italics saying what it was checked against: the first run on a live organisation, on 2026-11-03, JumpCloud's own documentation on the date given, or both. A **Not checked** line names what that run did not try.

> **Followed once on a live organisation**
>
> An installation that signs in through JumpCloud leaves the Auth0 Management credentials blank, and the Managers group decides who is a Manager. This page was followed on a live JumpCloud organisation, in the United States region, on 2026-11-03; Connect your identity provider says what that run showed.

4.10.1 Confirm first

- **Your plan includes single sign-on with a custom OpenID Connect application.** The sign-in gate is an OpenID Connect client; without one there is nothing to register. JumpCloud's pricing page does not say which plan includes it, so ask JumpCloud if Access, SSO Applications offers no custom OIDC application.

  Checked: the first run, 2026-11-03. JumpCloud's pricing page, read 2026-10-07, ties custom OIDC applications to no plan.

- **No access policy refuses the browser that approves a laptop.** A developer approves their laptop in a browser. A policy on the application that requires a managed device or a network refuses a browser it does not recognise; JumpCloud's Device Trust certificates on Linux work only in Google Chrome. If a policy applies, say which, so developers know which browser to use.

  Not checked: the first run, 2026-11-03, did not try an access policy. JumpCloud's conditional access documentation, read 2026-10-07.

- **The installation's server can reach JumpCloud over https.** The gate fetches JumpCloud's discovery document, tokens and keys from the server, not from a browser, so a server that cannot reach the issuer (step 10) over https cannot sign anyone in. Ask whoever runs the installation.

  Checked: the first run, 2026-11-03. Not from JumpCloud's documentation: it depends on the installation's network.

4.10.2 Before you start

Have `<base URL>`. Then, in Identity Management, User Groups, create two groups: one for the people who are Managers, for example `winner-managers`, and one for everyone else who may sign in, for example `winner-members`. The names are yours; write the Managers group's name down exactly as you typed it, and give it no comma: Winner AI splits the group attribute on commas, so a Managers group with a comma in its name never matches. In Identity Management, Users, put each person in the group they belong in; a Manager need not also be in the Members group.

Never use a broad group such as All Users as the Managers group: it is in everyone's claim, so everyone would be a Manager. Binding a broad group as the Members group admits everyone in it, which is a choice, not an error.

Checked: the first run, 2026-11-03, with a Managers and a Members group bound to the application. JumpCloud's single sign-on documentation, read 2026-10-07.

4.10.3 In the JumpCloud Admin Portal

1. **Create the application.** Access, SSO Applications, Add New Application, then Custom Application, Manage Single Sign-On, Configure SSO with OIDC (typing `OIDC` in the search box reaches the same place). Give it a Display Label such as `Winner AI`, then Configure Application and open the SSO tab. Make it in the console: JumpCloud's admin API stored no OpenID Connect settings when tried, and documents no way to. Only this one application is needed; the laptops need none of their own.

   Checked: the first run, 2026-11-03. Console path from JumpCloud's single sign-on with OIDC page, read 2026-10-07; the admin API tried by 2026-10-22.

2. **Set the redirect address.** Redirect URIs: `+Add URI`, then `<base URL>/oauth2/callback`. Login URL: `<base URL>`. The gate builds its callback from the base URL, so the redirect URI must match it exactly, scheme and port included. Winner AI compares only the redirect URI; nothing checks the Login URL.

   Checked: the first run, 2026-11-03, with an https address.

3. **Choose the client authentication type.** Client Authentication Type: Client Secret Basic. The console selects Client Secret POST by default, so change it; Basic is the one the gate was run with. Not Public (None PKCE): it issues no secret, and the gate needs one. The installation has no setting for the type.

   Checked: the first run, 2026-11-03, with Client Secret Basic. JumpCloud's discovery document, fetched 2026-10-22, lists both.

4. **Leave the subject claim at its default.** Subject Claim: JumpCloud User ID. Winner AI keeps each person on this id, so remapping it to an attribute that can change, such as the e-mail or the username, breaks that person's record when the attribute changes.

   Checked: the first run, 2026-11-03. JumpCloud's OIDC page, read 2026-10-07.

5. **Grant types and scopes.** Grant Types: Authorization Code is ticked and cannot be unticked; leave Refresh Token unticked, because the gate, as this release configures it, never refreshes a sign-in. Attribute Mapping: tick the **Email** and **Profile** standard scopes. Do not add a `groups` scope: JumpCloud lists none, and whether it ignores one or fails the sign-in is not yet known. Hand over: the scopes are `openid profile email`.

   Checked: the first run, 2026-11-03. Grant types from JumpCloud's OIDC page, read 2026-10-07; scopes from its discovery document, fetched 2026-10-22.

6. **Send the e-mail.** The Email scope sends the person's `email` and `email_verified`. The console adds `email_verified` as a constant attribute with the value `true`, greyed out, so every user arrives verified whatever happened to their address. Leave it: it cannot be removed. A Manager is therefore named by the Managers group, never by a list of e-mail addresses.

   Checked: the first run, 2026-11-03. The two claims from JumpCloud's OIDC claims page, read 2026-10-22.

7. **Include the group attribute.** Still in Attribute Mapping, tick **include group attribute** and name it, for example `groups`. JumpCloud refuses to activate the application while the name is blank. Whether JumpCloud sends the attribute as a list in the ID token, and how many groups it carries, is not yet known. Hand over: the name, exactly as typed.

   Checked: the first run, 2026-11-03: the named attribute arrived, and a bound Managers group made its member a Manager. How many groups it carries was not measured. JumpCloud's OIDC page and claims page, read 2026-10-07.

8. **Bind the Managers and Members groups.** On the application's **User Groups** tab, tick both groups from Before you start. JumpCloud admits only people in a bound group, and puts only bound groups in the group attribute, so a Managers group that is not bound never reaches Winner AI and its people sign in as Members, or not at all. Hand over: the Managers group's name, exactly.

   Checked: the first run, 2026-11-03: the tab is labelled User Groups. JumpCloud's claims page, read 2026-10-07, and its page on authorising users, read 2026-11-04, which says to select the User Groups tab.

9. **Activate, and copy the client id and secret.** Click Activate. The console then shows the **Client ID** and the **Client Secret** once: copy both before you click Got It. A lost secret is regenerated, not shown again. Hand over: both, the secret as What to hand back says.

   Checked: the first run, 2026-11-03. JumpCloud's OIDC page, read 2026-10-07.

10. **Find the issuer for your region.** JumpCloud has one issuer per region, and the wrong one fails the sign-in with "The requested OAuth 2.0 Client does not exist". The issuer keeps its trailing slash.

Region	Issuer	End-session address
United States	`https://oauth.id.jumpcloud.com/`	`https://console.jumpcloud.com/userconsole/logout`
European Union	`https://oauth.id.eu.jumpcloud.com/`	not known; see below
India	`https://oauth.id.in.jumpcloud.com/`	not known; see below

    The end-session address is the User Portal's logout, not the `end_session_endpoint` in the issuer's discovery document. The session that signs a person straight back in is the User Portal's, and only its logout ends it; in the United States run, the discovery document's address refused Winner AI's sign-out. Hand over: the issuer and the end-session address.

In the European Union and India regions, contact Winner AI support before the install. Their User Portal logout addresses have never been tried, and nobody has checked that one signs a person out. The installation cannot start without an end-session address: its settings check refuses an https installation with none, and one whose host the gate may not send a browser to. Do not use the discovery document's address in its place.

Checked: the first run, 2026-11-03, in the United States region. The United States discovery document fetched 2026-10-22, the European Union one 2026-10-07 and the India one 2026-11-04. No sign-in or sign-out was tried in the European Union or India region.

4.10.4 What to hand back

Give these to whoever runs your installation. They go in its settings; the identity provider guide shows where.

What you hand back	From	Setting it fills
The issuer, with its trailing slash	step 10	`OAUTH2_PROXY_OIDC_ISSUER_URL`
The end-session address; in the European Union and India regions, ask Winner AI support first	step 10	`OIDC_END_SESSION_URL`
The end-session address's host, for example `console.jumpcloud.com`	step 10	`OAUTH2_PROXY_WHITELIST_DOMAINS`
The Client ID	step 9	`AUTH0_PROXY_CLIENT_ID`
The Client Secret	step 9	`AUTH0_PROXY_CLIENT_SECRET`
The scopes, `openid profile email`	step 5	`OAUTH2_PROXY_SCOPE`
The group attribute's name	step 7	`OAUTH2_PROXY_OIDC_GROUPS_CLAIM`
The Managers group's name, exactly	step 8	`FLEETVIEW_MANAGER_GROUP`

The two client settings keep their Auth0 names but hold the client of whichever provider the issuer names. Send the answers to Confirm first with the table. Nothing else fills a setting: the Members group admits its people by being bound, the client authentication type and the subject claim are settled in JumpCloud alone, and step 6's `email_verified` is a constant, so there is nothing to hand back for it.

Send the Client Secret only to whoever runs your installation, through your password manager or the channel your organisation uses for secrets, never in a plain e-mail. Never send it to the makers of Winner AI: only your installation needs it.

4.11 Onboarding on Auth0

The steps a Manager or the operator follows to put a team on Winner AI when the installation signs in through Auth0. The team is one Auth0 Organization on your tenant, the one `AUTH0_DOMAIN` names, and the control plane's bootstrap organisation is that Organization.

On an installation from the bundle, read the server's commands as Commands on the other pages says: in `<install>`, as the user who owns `<install>/.env`, with plain `docker compose` and no `sudo`. Most blocks below are not server commands: they run in a shell signed in to the tenant (`auth0 login`; the block under A manager needs two more scopes), and need no server. They are written for bash: on a Mac, whose terminal runs zsh, type `bash` first, since zsh reads the `# ...` after a command as more of the command.

4.11.1 The organisation

One Auth0 Organization per installation, provisioned in the control plane before anyone signs in. It is the one `AUTH0_TEST_ORG_ID` (`FLEETVIEW_BOOTSTRAP_ORG_ID`) names, and the backend provisions it at every start. The installation serves that organisation alone; a manager of any other organisation the control plane once provisioned is refused at the Dashboard.

4.11.2 Who may sign in

The sign-in gate admits every session the identity provider admits and checks no group or role, so the tenant alone decides who reaches the control plane. Winner AI sets none of these settings for you.

- **The gate's application** (`AUTH0_PROXY_CLIENT_ID`) requires an organisation: `organization_usage: require` with `organization_require_behavior: pre_login_prompt`, so the login page asks for the organisation first. Enable only this installation's Organization on it.
- **The Organization's enabled connection** keeps `assign_membership_on_login: false`. Where database signup is open (the default), anyone who registers and verifies an address can name the Organization at the prompt and join it.
- **The callback and logout addresses.** The gate's application lists `https://<control-plane-host>/oauth2/callback` among its Allowed Callback URLs and `https://<control-plane-host>` among its Allowed Logout URLs.

The post-login Action (below) also refuses a sign-in through any Organization but this installation's, or through none. It checks which Organization a sign-in came through, not how the person joined it, so both settings stay required.

Check it by hand after creating the application or changing its settings: sign in as a verified user who belongs to no organisation. Auth0 should stop them before any page of the control plane loads. The Action's "This installation does not serve the organization you signed in with" means the application does not require an organisation and the Action caught it; fix the setting all the same. Reaching Home, even its empty page, means a setting is wrong: read the application's organisation settings and each enabled connection's `assign_membership_on_login`.

4.11.3 People

This page is the Auth0 path. How any provider's people sign in, who is a Manager and how people join and leave is the identity provider guide.

Every person is an Auth0 user in the tenant's database connection, which must be named `Username-Password-Authentication`, the name Admin and the terminal block below create users in. Each is a member of exactly one organisation provisioned on this deployment, with one org-scoped role there. The control plane keeps a row per person in `control.users` and keeps its role true to Auth0 on its own, with no operator step. To remove someone, remove them from the organisation in Auth0 (Organizations, the org, Members) or delete the user; nothing on the host needs editing. Without Auth0 Management credentials no sync asks Auth0, so a Manager removes the person on Admin instead, and Restore brings them back as a Member. If the manager group names them, or, where no manager group is set or no usable groups claim arrives, `FLEETVIEW_MANAGER_EMAILS` does, their next request lifts them to Manager, so remove that entitlement before Restore for Member-only access.

One organisation per person. The installation serves one organisation, but the control plane may still hold rows of another it once provisioned. Admin resolves a person from their existing row by email and, failing that, from the first provisioned organisation Auth0 lists them in; enrollment refuses a person whose row belongs to another organisation. Adding an address that already has an Auth0 user adds that user to this organisation with the chosen role and no password; they sign in with their existing login or reset it. The add is refused, before anything changes in Auth0, when the address or its Auth0 user has a roster row in another organisation the control plane once provisioned, or when that user is an Auth0 member of one. Membership of an Auth0 organisation this deployment has not provisioned does not refuse. To see which organisation holds a refused address:

```
auth0 api get "users-by-email?email=<address>" | jq '.[].user_id'
auth0 api get "users/<user_id>/organizations" | jq '.[].id'
```

Then on the server, in `<install>`, open psql in the database container and look the row up:

```
docker compose exec db sh -c 'psql -U "$POSTGRES_USER" "$POSTGRES_DB"'
```

```
select u.id, u.email, o.auth0_org_id from control.users u
  join control.orgs o on o.id = u.org_id
  where lower(u.email) = '<address>' or u.auth0_user_id = '<user_id>';
```

Nothing in Admin moves a person between organisations. Add them under another address, or, when the other organisation's row is stale, delete their Auth0 user and the row, in the same psql, with the `id` the lookup printed (`delete from control.users where id = '<id>';`; a row that still has devices refuses, since its devices reference it), then add them again. Delete the user rather than only removing them from that organisation: the other organisation's manager may hold its temporary password, and a user this control plane created is linked without a verified address.

An existing login must be vouched for. Where database signup is open (the default), an Auth0 user for an address proves only that someone typed it. Admin links one only when Auth0 has verified the address, this control plane created the user, the user is already in this organisation, or their id is on this organisation's roster, revoked rows included. Otherwise the add is refused with nothing changed in Auth0. Have the person verify their address, which the verification email does:

```
jq -n --arg u "<user_id>" '{user_id:$u}' | auth0 api post jobs/verification-email
```

Delete that Auth0 user only once the person who holds the address confirms the login is not theirs, or it is confirmed stale and nobody uses it; otherwise verification is the path. An add creates a fresh user, with a temporary password, only when no Auth0 user holds the address; the `users-by-email` lookup above lists every one.

A manager

Two ways to add one: Admin hands a new user a temporary password, the terminal emails a set-password link.

Through Admin (`/admin` , "Add member", role *manager*), which needs a Manager already and the control plane's Management API credentials (`AUTH0_MGMT_CLIENT_ID` and `AUTH0_MGMT_CLIENT_SECRET` ; without them the add fails): the control plane creates the user, if the address has none, with a temporary password shown once on the next page, adds them to the organisation and assigns the role. Hand the password over out of band; nothing makes the person change it. An existing user is added without a password and signs in with their existing login or resets it.

From a terminal, with the Auth0 CLI logged in to the tenant, when there is no manager yet or you would rather not see a password at all. `ROLE_NAME` is `manager` or `member` ; the same block adds either.

The CLI's default login cannot add a member to an Organization or give them a role there. Log it in with the two scopes those steps need, which it requests beside its defaults; a CLI already logged in without them logs in again:

```
auth0 login --scopes "create:organization_members,create:organization_member_roles"
```

Then, in the same shell:

```
set +x                                  # no shell tracing while a password is in a variable
ORG=<organization id>                   # org_...
ROLE_NAME=manager                       # or member
EMAIL=<address>; NAME=<display name>
TENANT=<tenant domain>; PROXY_CLIENT_ID=<AUTH0_PROXY_CLIENT_ID>
ROLE=$(auth0 roles list --json | jq -er --arg n "$ROLE_NAME" '.[] | select(.name==$n) | .id')
PW="Fv-$(openssl rand -base64 18 | tr -d '/+=')!x9"   # never printed; replaced by the email
below
USER_ID=

if [ -z "$ROLE" ]; then
  echo "no role named $ROLE_NAME: stop" >&2
elif [ "${#PW}" -lt 20 ]; then
  echo 'no password was generated: stop' >&2
# the password reaches jq on stdin, never as a process argument
elif ! USER_ID=$(printf '%s' "$PW" | jq -n --rawfile p /dev/stdin --arg e "$EMAIL" --arg n
"$NAME" \
    '{connection:"Username-Password-Authentication", email:$e, name:$n, password:$p,
email_verified:true}' \
  | auth0 api post users | jq -er .user_id); then
  echo 'the user was not created: stop' >&2
elif ! jq -n --arg u "$USER_ID" '{members:[$u]}' | auth0 api post "organizations/$ORG/members";
then
  echo "$USER_ID was created but not added to the organisation: stop" >&2
elif ! jq -n --arg r "$ROLE" '{roles:[$r]}' | auth0 api post
"organizations/$ORG/members/$USER_ID/roles"; then
  echo "$USER_ID was added but given no role: stop" >&2
else
  # Auth0 emails a set-password link; the generated password is never used
  jq -n --arg c "$PROXY_CLIENT_ID" --arg e "$EMAIL" \
      '{client_id:$c, email:$e, connection:"Username-Password-Authentication"}' \
    | curl -sS --fail-with-body -X POST "https://$TENANT/dbconnections/change_password" \
        -H 'Content-Type: application/json' --data-binary @-
fi
unset PW
```

`auth0 api post` reads the body from stdin on purpose: `--data` hangs. Each step runs only once the one before it succeeded: `jq -er` fails on a missing id, and `--fail-with-body` on a non-2xx from Auth0. A failure prints which step stopped, and leaves your shell open with the variables set. A stop after the user was created names its id: a second run of the block would be refused because the user exists, so run the remaining lines by hand with that id. The control plane writes the person's row the first time they sign in, so nothing else is needed.

Why not Auth0 organisation invitations: the sign-in gate does not forward the invitation parameters to `/authorize`, so an invitee would reach the sign-in page without their invitation; the set-password email above gives the same experience without that hop.

Members and the first team

A member, a leaver and the first team are the same on every provider: the identity provider guide describes them. What stays here is what only an Auth0 installation does, which is creating the person in Auth0, in the two sections above.

Enrollment lands a laptop in the organisation of the person who approved its code at `/login/approve`, as that person: the laptop pass carries both, and the installation serves that one organisation. The proxy admits members as well as managers, so each person approves their own laptop's code. Anyone in another organisation is refused at sign-in by the post-login Action (below), and the Dashboard and the approval would refuse them too: `/login/approve` resolves only an organisation provisioned here. A `winner` from before the control plane's own login still uses the Auth0 device flow, whose token carries no organisation: it lands in the control plane's **bootstrap organisation** (the one `AUTH0_TEST_ORG_ID` names) and is refused for a person who is not a member of it. That check reads only the first page of members Auth0 returns, so in a larger organisation it may refuse a member too; a current `winner` is not affected.

4.11.4 The post-login Action

The tenant's post-login Action puts the roles and organisation claims on the tokens. Where database signup is open (the default), anyone can register an address a team's roster trusts without owning it; the Action therefore refuses, in the browser and in `winner login` alike, a sign-in unless Auth0 verified the address or the control plane created the account (its `app_metadata` tag, which the user cannot write). Admin's rule is wider: it also links an existing user already in this organisation or on its roster (An existing login must be vouched for, under People), and the Action still refuses that user at sign-in until the address is verified. For the sign-in gate's application only, it also refuses a sign-in through any Organization but this installation's, or through none: without Management credentials the gate's group decides who is a Manager, so another Organization enabled on that application would make its Managers Managers here. A `winner` that still uses the Auth0 device flow carries no Organization, so that `winner login` is not checked for one. A tenant runs the copy each release ships in its bundle, `backend/auth0-post-login-claims.action.js`, deployed by hand. Run these on the operator's own machine, not the server, which needs neither `jq` nor a tenant login: in the folder of the release's bundle unpacked there and checked with the release key, the signature and `sha256sum`, as Verify the bundle's signature shows (its server check, `./scripts/check-host.sh`, is for the server, not this machine), from a shell signed in to the tenant (`auth0 login`). Auth0 takes the code as a JSON string, and `auth0 api` reads a body from stdin because `--data` hangs.

First: mark verified the accounts you created yourself and know, before the Action is deployed (the last list below says which, and how). Auth0's documentation says post-login Actions run on a refresh-token exchange, so a laptop still enrolled through the Auth0 device flow, for an account the Action would refuse, fails at its next hourly report.

Second: the Action's two secrets. The Action reads `AUTH0_PROXY_CLIENT_ID`, the gate's application, and `AUTH0_TEST_ORG_ID`, this installation's Organization, as Action secrets that hold the same values as the installation's `.env` lines of those names; keep the two in step. Every command below sends both with the code, so have the `.env` values to hand. Without either, or with one empty, the Action refuses every sign-in on the tenant, `winner login` included, and the next hourly report of every laptop still enrolled through the Auth0 device flow, whose report exchanges a refresh token (Auth0's documentation says post-login Actions run on that exchange; a laptop on the control plane's own pass is not affected); recover by running the update below with the values, or in the Auth0 dashboard. A wrong `AUTH0_PROXY_CLIENT_ID` skips only the Organization check: the Action does not recognise the gate and checks no Organization, so only the tenant settings in Who may sign in stand, while the verified-or-created check still refuses as before. The pair belongs to the Action, and the last update sent wins: a second installation's operator who updates the same Action on the same tenant replaces the first's pair, and it is then the first installation's gate that goes unchecked.

Deploy the release's copy. The first time on a tenant, create it; the trigger version `v3` is the one Auth0 offers for post-login:

```
NAME=fleetview-roles                    # the name the tenant already uses, when it has one
PROXY_CLIENT_ID=<AUTH0_PROXY_CLIENT_ID from the installation's .env>
ORG_ID=<AUTH0_TEST_ORG_ID from the installation's .env>
if [ -z "$PROXY_CLIENT_ID" ] || [ -z "$ORG_ID" ]; then
  echo 'both secret values are needed: stop' >&2
else
  jq -n --arg n "$NAME" --rawfile c backend/auth0-post-login-claims.action.js \
        --arg p "$PROXY_CLIENT_ID" --arg o "$ORG_ID" \
      '{name:$n, supported_triggers:[{id:"post-login", version:"v3"}], code:$c,
        secrets:[{name:"AUTH0_PROXY_CLIENT_ID",value:$p},{name:"AUTH0_TEST_ORG_ID",value:$o}]}' \
    | auth0 api post actions/actions
fi
```

An action does nothing until it is deployed and bound to the Login flow. The bindings endpoint replaces the whole list, so a body holding only this Action would unbind every other post-login Action the tenant has (an MFA or policy Action). Build the body from the current bindings plus this Action, and read it before the PATCH; the list's order is the order the Actions run in, and this appends the Action last:

```
# the lookup filters by name; check the parameter against your tenant
ACTION_ID=$(auth0 api get "actions/actions?actionName=$(jq -rn --arg n "$NAME" '$n|@uri')" | jq -
er --arg n "$NAME" '.actions[] | select(.name==$n) | .id')
if [ -z "$ACTION_ID" ]; then
  echo 'no action named that: stop' >&2
else
  echo '{}' | auth0 api post "actions/actions/$ACTION_ID/deploy"
  auth0 api get actions/triggers/post-login/bindings | jq .      # read what is bound now
  auth0 api get actions/triggers/post-login/bindings \
    | jq --arg id "$ACTION_ID" --arg n "$NAME" \
        '{bindings: ([.bindings[] | {ref:{type:"action_id", value:.action.id},
display_name:.display_name}]
          | if any(.ref.value == $id) then . else . + [{ref:{type:"action_id", value:$id},
display_name:$n}] end)}' \
    | tee /dev/stderr | auth0 api patch actions/triggers/post-login/bindings
fi
```

The body is printed to the terminal before it is sent. If any existing entry is missing from it, the field names differ on your tenant: stop and build the list by hand from the first command's output.

To update an action the tenant already has, replace its code and both secrets, and deploy. It sends both secrets every time, so no update depends on what Auth0 keeps from an earlier one. It stops with a message, rather than exiting the shell, when a value or the action is missing, so `NAME` and `ACTION_ID` stay set for the checks after it:

```
NAME=<the action's name>
PROXY_CLIENT_ID=<AUTH0_PROXY_CLIENT_ID from the installation's .env>
ORG_ID=<AUTH0_TEST_ORG_ID from the installation's .env>
ACTION_ID=$(auth0 api get "actions/actions?actionName=$(jq -rn --arg n "$NAME" '$n|@uri')" | jq -
er --arg n "$NAME" '.actions[] | select(.name==$n) | .id')
if [ -z "$PROXY_CLIENT_ID" ] || [ -z "$ORG_ID" ]; then
  echo 'both secret values are needed: stop' >&2
elif [ -z "$ACTION_ID" ]; then
  echo 'no action named that: stop' >&2
else
  jq -n --rawfile c backend/auth0-post-login-claims.action.js \
        --arg p "$PROXY_CLIENT_ID" --arg o "$ORG_ID" \
    '{code:$c, secrets:[{name:"AUTH0_PROXY_CLIENT_ID",value:$p},
{name:"AUTH0_TEST_ORG_ID",value:$o}]}' \
    | auth0 api patch "actions/actions/$ACTION_ID" \
    && echo '{}' | auth0 api post "actions/actions/$ACTION_ID/deploy"
fi
```

Compare what the tenant runs with the release's copy. Print the action and read its output: the code that runs is on its deployed version, the draft is the top-level `code`. Check the field names against your tenant's output before trusting a clean diff:

```
ACTION_ID=$(auth0 api get "actions/actions?actionName=$(jq -rn --arg n "$NAME" '$n|@uri')" | jq -
er --arg n "$NAME" '.actions[] | select(.name==$n) | .id')
auth0 api get "actions/actions/$ACTION_ID" | jq .       # find the deployed version's code in
this output
auth0 api get "actions/actions/$ACTION_ID" | jq -j '.deployed_version.code' > /tmp/deployed-
action.js
diff /tmp/deployed-action.js backend/auth0-post-login-claims.action.js
```

No output from `diff` means they match. An empty or `null` file means the field path is wrong or the action was never deployed; take the code from the printed output instead. Do the compare after every release that changes the file and on every tenant a deployment uses.

Check the deployed Action holds both secrets. This reads the deployed version's secret names, the ones the running code sees, and never their values (the field path is from Auth0's documentation, not yet seen on a tenant):

```
auth0 api get "actions/actions/$ACTION_ID" | jq -r '.deployed_version.secrets[]?.name' | sort
```

Expect exactly `AUTH0_PROXY_CLIENT_ID` and `AUTH0_TEST_ORG_ID`; if it prints nothing, the deployed version holds no secrets or the field path differs on your tenant: find the secrets' names in the printed action above. Auth0 shows no secret's value back, so sign in to the control plane yourself afterwards: a wrong organisation id refuses you with "This installation does not serve the organization you signed in with". Nothing shows a wrong client id; when in doubt, run the update again with the `.env` values.

Which accounts need marking verified. Mark verified only accounts you created yourself and know. Marking one is vouching for its address; marking a stranger's self-signup does exactly what the Action exists to stop.

- Created by Admin ("Add member"): nothing to do. The control plane tags what it creates, and the person signs in with the temporary password as before.

- Created by the terminal block under "A manager" above: nothing, it sets `email_verified: true`.
- Created another way, in the Auth0 dashboard or by an import, by you, for an address you know is the person's: mark it verified (the loop below), but only with your own record of the Auth0 user id that create returned. A lookup by address finds whoever holds it, a stranger's earlier self-signup included, even after your create answered 409. After a 409, or without that record, do not mark it: the account may be the person's own, used by other applications. Ask them to prove the address with the verification email under "An existing login must be vouched for" above, and delete an account only once you know it is not theirs. `created_at` does not tell such an account from a self-signup for the same address, so never use it as the evidence.
- Anyone else who is unverified, including every self-signup: do not mark them. They prove the address with the verification email under "An existing login must be vouched for" above, or you delete them.

The listing below shows who the Action will refuse, not who to mark. It returns the first 100 users; add `&page=1`, `&page=2` and so on for more. A search query stops at 1,000 matching records however far you page, so a larger tenant is not listed in full this way: narrow the query (by `connection` or by a `created_at` range in `q`) or use Auth0's bulk user export from the dashboard (the menu names vary; check yours).

```
auth0 api get 'users?
q=email_verified%3Afalse&search_engine=v3&fields=email%2Capp_metadata%2Ccreated_at&include_fields
=true&per_page=100&page=0' \
  | jq -r '.[] | select(.app_metadata.fleetview_created != true) | [.email, .created_at] | @tsv'
for USER_ID in '<user id>' '<user id>'; do   # the ids your own creates returned, never a lookup
by address
  echo '{"email_verified":true}' | auth0 api patch "users/$USER_ID"
done
```

Check the listing's query syntax against your tenant if it returns an error or nothing. Where you keep a test tenant, do all of this there before deploying the Action on the tenant your team signs in with.

4.11.5 Checks

- `auth0 orgs members list <org id>` lists everyone expected, and `auth0 api get organizations/<org id>/members/<user id>/roles` shows the role each should have.
- Each manager signs in at `https://<control-plane-host>` and sees Home and the Dashboard; each member's `winner login` ends with the push service installed, and the roster (`/admin`) shows the laptop from enrollment. "Awaiting first push" there also means the roster could not read the org schema, so a push is proven by a session count or a last-push time.

4.12 Licence

How to install and renew the licence file, what the dates on its banner mean, and what a lapse switches off and what it never does. It is written for the operator who runs the server. The upload itself is a Manager's action on Settings, so an operator who is not a Manager hands the file to one; Connect your identity provider says who is a Manager. One licence covers an installation and the one organisation on it.

4.12.1 What the licence is

A signed file Winner AI sends you, carrying a licence id, the licensee's name, an issue date, an expiry date and a number of seats. The server checks it from the file alone when it is uploaded: there is no licence server to reach and no tool to run by hand.

The expiry date is the last licensed day, in UTC. A first licence carries 40 seats; the seats and expiry in force are the ones Settings shows. There is no suspension: we cannot switch an installation off from afar, and the expiry date is the only limit. Where this page says to ask us, use the contact Getting started and help names.

Nothing of the licence's contents is sent to us. With usage counts on, an upload, the licence's state changes and the daily seat count are among the events Telemetry lists, and with error reports or timings on, the upload's request is reported like any other, by its route, never with the file.

4.12.2 Upload it

Upload the file exactly as it was sent: line endings, blank lines and spaces around a line do not matter, but a change to any of its content breaks the signature.

1. Save the file exactly as it arrived.
2. A Manager signs in, opens Settings and goes to the **Licence** row, picks the file in **Licence file** and presses **Upload** (**Replace** when one is stored).
3. Settings answers "Licence stored." and the row shows `<licensee> · <licence id>` and `Issued <date>, expires <date> (UTC) · <n> seats · <m> in use`. That line is the check: it is what the server read from the file.

With none stored, the row reads "No licence is stored." and the seats in use. An expired file is still accepted; what it means follows its dates, below.

4.12.3 If the upload is refused

The message appears on the Licence row, and the licence already in force stays in force.

Message	What to do
This is not a Winner AI licence file, or it was damaged on the way: upload the file exactly as it was sent.	Upload the original attachment, unopened and unedited. If it is refused again, ask us to send it again.
The licence file's signature is not of a kind Winner AI accepts.	Ask us for a new file.
The licence file was not signed by Winner AI's licence key.	This file did not come from us. Upload the one we sent, or ask us for it.
The licence file's signature was not made for a licence.	The file is signed for something else. Upload the licence file.
The licence file was changed after it was signed.	Upload the original, unedited; if that is refused too, ask us to send it again.
The licence file is of a newer format than this release reads: upgrade Winner AI to use it.	Install a later release (Installing a later release), then upload it again.
The licence file is signed, but its fields are not valid.	Ask us for a new file.
This licence was issued before the one in force, so it cannot replace it.	The licence in force is newer; keep it. If you expected this file to replace it, ask us.
Another licence was stored while this one was checked: upload the file again.	Upload it again.

4.12.4 Seats

A seat is a person with at least one enrolled Device. A second laptop is free; revoking a person's last Device frees the seat; a removed person whose laptop is still enrolled still holds one.

Over the licence's number, Settings and the roster say "`<n>` more people than the licence's `<seats>` seats. Nobody is refused; this is settled at renewal.", with "person" for one person and "seat" for one seat. The warning shows only while a licence is stored, and enrolment never reads the count.

4.12.5 The dates

Days are UTC dates on the server's clock. The banner names the switch-off date, so it is the one to trust.

State	Days	What the banner shows
Licensed	Up to and including the expiry date	No banner.
Grace	The 30 days after the expiry date	A banner: "The licence expired on `<expiry>`. Coaching, Skills and Benchmarks switch off on `<date>`: `<n>` days left.", then "1 day left", then "last day".
Switched off	From the 31st day after the expiry date	The banner says "The licence expired on `<expiry>`. The 30 days of grace are over.", and the three pages are off (below).

With no licence stored, grace starts on the deployment's first start instead (that day the banner reads 30 days left), and the banner opens "No licence is installed."

For a licence that expires on 2027-10-31:

Date (UTC)	State	Banner
2027-10-31	Licensed	None.
2027-11-01	Grace	"The licence expired on 2027-10-31. Coaching, Skills and Benchmarks switch off on 2027-12-01: 29 days left."
2027-11-29	Grace	The same, ending "1 day left."
2027-11-30	Grace	The same, ending "last day."
2027-12-01	Switched off	"The licence expired on 2027-10-31. The 30 days of grace are over."

Managers see the banner at the top of Winner AI's pages; the Dashboard carries none. A Member's own pages carry none either, but the page where anyone approves a laptop's sign-in carries it, so a Member sees it there. Every banner links "Upload a licence on Settings.", which only a Manager can open. The 30 days are fixed by the release: Settings shows them as the fixed row **Licence grace**, and no setting changes them.

The dates follow the server's clock: keep it right.

4.12.6 What switches off, and what never does

When grace ends, Prompt Coaching, Skills and Benchmarks switch off: their pages, their actions, Skill downloads, and the background runs behind them. The sidebar drops all three, and opening one at its address shows "The licence has lapsed." and:

> Coaching, Skills and Benchmarks are off. Nothing they recorded is deleted, and they come back as they were once a valid licence is uploaded.

A Manager gets the button "Upload a licence on Settings"; anyone else reads:

> Ask a Manager to upload a licence on Settings.

Settings' **Features on** row reads "none", with its form hidden, and shows:

> Unavailable: the licence has lapsed: upload a valid licence to switch these back on.

Settings will not switch them back on.

Nothing is deleted. The findings, Skills and results the three pages hold are not shown while they are off, and come back unchanged.

Never switched off, in any state: signing in, the Dashboard and the sessions in it, Home (without the three features' figures while they are off), the roster (adding people, revoking laptops), laptop sign-in and enrolment, pushes, client downloads and updates, roles following your identity provider, and Settings with the upload.

4.12.7 Renew

Renewing is uploading the new file the same way, with **Replace**; nothing else changes. A new file replaces the stored one unless it was issued earlier; one issued the same day is accepted.

After a lapse the same upload is the whole recovery: whichever of the three pages were on before it open again at once, and their background runs resume on their next scheduled run. A file that has itself expired more than 30 days ago is stored, and its own dates decide: after a lapse the three stay off, and uploaded during the first-start grace, with no licence stored before it, it ends that grace and switches the three off at once.

Seats over the number are settled at this renewal: tell us how many you need.

5 Changelog

What each release of Winner AI changes, from 0.4.0 on, and whether it asks anything of the people running it. The format follows Keep a Changelog, and versions follow semantic versioning.

A change customers would notice adds a line to **Unreleased** in the same pull request that makes it. Cutting a release renames that section to the version being tagged.

Releases 0.5.1 to 0.9.0, published before the release workflow began taking its notes from this page, still carry notes generated from pull request titles. From the next release on, each released section is the body of that version's GitHub release: the release workflow copies it into the draft and refuses a tag with no section here before the build. A section's **Requires of you** always carries a **Members:** line saying whether members must reinstall the client or sign in again, "nothing" when they need not. Released sections use plain Markdown, because GitHub shows MkDocs-only syntax such as admonitions as raw text.

5.1 Unreleased

5.2 0.13.5 - 2026-11-05

5.2.1 Changed

- Laptops and the Dashboard run the session collector (`agentsview`) 0.44.0. An empty Cursor conversation row no longer stops every push. A cut-off one still stops that laptop's Cursor sessions, and a new message to another agent's session it already sent can wait for the next periodic push, every 15 minutes. The first push after a laptop updates sends every session again, and takes longer than usual: one test laptop with 237 sessions took about 8 seconds, and a real laptop's history is larger.
- The guide and the control plane's pages describe Winner AI in its own terms: the laptop program is the session collector, and the server's session browser is the Dashboard. The Dashboard's remaining "agentsview" chrome is renamed too: the About dialog's label, the chart palette's name in Settings > Appearance, and the Remote access description.

5.2.2 Requires of you

- **Operators:** upgrade the server first: its first start migrates every organisation's data, within the deploy's thirty-minute wait. Then every laptop takes the new collector through `winner update` . From the first laptop push on 0.44.0, a laptop still on collector 0.42.0 is refused until it updates, and going back to the previous release means restoring the backup the upgrade took.
- **Members:** nothing on a laptop whose `winner login` installed its services: the daily update brings collector 0.44.0. A laptop enrolled with `--no-service` gets it with `winner update` .

5.3 0.13.4 - 2026-11-04

5.3.1 Changed

- The guide now reads right for a server installed from the bundle, after an audit of every page against the release: JumpCloud lists the India region and says to ask support first in the European Union and India regions, the private-network steps say the zone check needs outbound port 53, the Postgres jail example is switched on, pointing the stack at a hosted model recreates Ollama too, and the macOS client's folder-access limit on an unsigned build is stated.
- The server install guide now says how to install Docker Engine and the Compose plugin on Ubuntu from Docker's own repository, removing Ubuntu's older `docker.io` packages first.

5.3.2 Fixed

- On a server installed from the bundle, laptops were told to push to `POSTGRES_PORT`, which the template sets to 5433, while Postgres is published on 5432; laptops are now always told 5432.
- The files the bundle ships name the guide's pages instead of source files the bundle lacks, and name the settings an operator sets. A restore whose stack failed to start says so instead of "started again", and an upgrade refuses an unreadable `.env` with the real reason.
- Admin showed "1 sessions" on a laptop that had pushed one session; it now reads "1 session".
- A repo choice made on the web page is no longer shown applied while a background push that did not stop, or agentsview's own push service that could not be removed, is still sending the old list. The choice stays pending until a later report sees every push on the new list.
- A Member approving a laptop login saw the Manager sidebar, whose Admin, Settings and Dashboard links each answered "Manager role needed"; the page now shows a Member only Home and Sign out.
- A laptop whose pass the control plane refuses told a removed person only to run `winner login`, which cannot let them back in; `winner report` and the hostname reporter's log now add that if the sign-in page says the account was removed, a Manager restores it.
- After an upgrade a browser could keep showing the previous release's guide pages; the guide now tells the browser to check for a newer copy on every visit.
- After a Restore nothing told the Manager what the person does next; Admin and the identity guide now say that the restored person runs `winner login` on their laptop again.
- A model whose calls all failed made Skills and Coaching look like a team with no data. Both pages and Settings now say the model call failed, with its HTTP status or Bedrock error type, whose model it was and when a call last worked; a Skills pass whose naming calls all failed is recorded as failed.

5.3.3 Requires of you

- **Operators:** nothing to change. On a server installed from the bundle whose `.env` kept the template's `POSTGRES_PORT=5433`, laptops enrolled before this release were told the wrong port: each runs `winner login` again after the upgrade. The post-login Action changed only in a comment, so it need not be deployed again.

- **Members:** nothing.

5.4 0.13.2 - 2026-11-04

5.4.1 Added

- On AWS, the server's `.env` secrets can live in AWS Secrets Manager: `./scripts/env-from-secrets.sh` reads one secret with the server's instance role and writes its keys into `.env`, before the install and again after a rotation. The bundle ships it.
- Each guide page offers its Markdown with a Copy page as Markdown button, and the guide serves an llms.txt index for AI agents.

5.4.2 Changed

- Getting started and help now gives the support address, support@getwinner.com.
- The identity guide and the configuration reference now give one answer on admitting a sign-in whose e-mail the provider has not verified: it stays on by default, and an operator who wants verified addresses only sets `OAUTH2_PROXY_ALLOW_UNVERIFIED_EMAIL=false`.
- The guides now match 0.13.1 on the points a first install meets: sign-in through any OpenID Connect provider, `check-env.sh` before the install, full-tunnel VPN and macOS laptops, what a network change mid-push does, the licence's first-start grace, and the commands each page gives.
- The guide now lays out the install in order, from choosing the server's name to backups, and adds the install steps for the licence, the features, laptops and backups. Pages written for Winner AI's own team no longer ship in it.

5.4.3 Fixed

- The server bundle now ships the Auth0 post-login Action that the onboarding guide deploys to the tenant; earlier bundles lacked it, so an operator who installed from a bundle had no copy to deploy or compare.
- The client installer now warns when git is missing or older than 2.36, which `winner login` and `winner repos` need to list a laptop's repos, and the client guides say how to get it on Ubuntu 20.04 and 22.04, Debian 11 and macOS. Such a laptop used to install without a word and then choose no repos at sign-in.

5.4.4 Requires of you

- **Operators:** only on Auth0, and only where the server was installed from a bundle: compare the tenant's post-login Action with this release's copy (`docs/onboarding.md`, "The post-login Action", has the commands), and deploy it if it differs or is missing.
- **Members:** nothing.

5.5 0.13.1 - 2026-11-03

5.5.1 Fixed

- The command beside `OAUTH2_PROXY_COOKIE_SECRET` in `.env.example` made a secret the sign-in gate refuses about three times in four, so the gate stopped at start and nobody could sign in. The comment now gives a URL-safe command, and `./scripts/check-env.sh` refuses a secret the gate would not start with, before anything starts.
- The server diagnostics file could keep the domain of a person's e-mail address, and drop the log fields beside it, when the edge logged a request's headers. Every address is now masked in full, on the server and in `winner diagnostics`.
- A laptop that changed network during a push, by joining or leaving a VPN for example, could leave its database session holding locks for about two hours, so every push from that laptop waited behind it. The server's Postgres now drops a client that stops answering within about two minutes, and the laptop's next push goes through.
- A new installation no longer names its organisation "Winner AI Test" in its page headers. The name is now `ORG_DISPLAY_NAME` in `.env`, read when the organisation is first made, and `Winner AI` when it is unset. An installation that already exists keeps its name.
- `./scripts/check-env.sh`, run by hand in the unpacked bundle before the install, now accepts a `.env` filled in as install step 4 says. It used to require the release and image keys the installer writes, and told the operator to ask the repo owner.
- On a JumpCloud organisation in the United States region, signing out now ends the person's JumpCloud session. The identity guide and the JumpCloud checklist now give the User Portal's logout as the end-session address, `https://console.jumpcloud.com/userconsole/logout`. The address they gave before refused the sign-out and left the person signed in.
- The upgrade command and the installer, refusing a bundle older than the release the server runs, no longer tell the operator to roll back to that bundle's release: the rollback goes back to the release the last upgrade replaced, and the refusal now says so and names `--bundle` and `--backup` for an older release. The upgrade command's `--help` now says the default backup goes in the folder the new bundle's archive is in.
- After an upgrade that failed or was cut off, `scripts/rollback-server.sh` in the release it started from now refuses and names the restore that undoes the attempt; it used to offer to undo the upgrade before it, from that upgrade's older backup.
- On a laptop with a long history of coding sessions, the background push now starts agentsview and waits up to 30 minutes for its first sync to finish before pushing, rather than running beside it, which could leave that sync stalled part-way with nothing pushed. Meanwhile `winner repos` says agentsview is still starting and how far its sync has got, instead of that it is not running. On macOS `winner push stop` now also stops the running agentsview daemon, including one started from the agentsview desktop app or a shell, which it used to leave running.

5.5.2 Requires of you

- **Operators on a JumpCloud organisation in the United States region:** set `OIDC_END_SESSION_URL=https://console.jumpcloud.com/userconsole/logout` in `.env`, and in `OAUTH2_PROXY_WHITELIST_DOMAINS` replace `oauth.id.jumpcloud.com` with `console.jumpcloud.com`, keeping

any other host already listed. Then run `docker compose up -d`. In the European Union region, step 10 of the JumpCloud checklist says what is still unconfirmed.

- **Members:** nothing on a laptop whose `winner login` installed its services: the daily update brings the client's first-sync and diagnostics fixes. A laptop enrolled with `--no-service` gets them with `winner update`.

5.6 0.13.0 - 2026-11-03

5.6.1 Added

- With usage counts on, a script error on a Winner AI page is reported with its type and our own line numbers, never its message.
- A guide page, Upgrade and roll back, gathers installing a later release and going back to the previous one, says why an older release is refused, and what laptops do after each.
- A Licence page in the guide: installing and renewing the licence file, the dates its banner counts, and what switches off when the grace ends.
- Model and telemetry now says in one place what blank telemetry settings guarantee, which setting switches each kind and what Settings shows for it, what is never instrumented, and where the values we hand over send events.
- A repo choice a Member saves on the web page now takes effect on their laptop at its next hourly report, with no terminal: the laptop sends the name and state of each of its repos and nothing else about them, applies each choice itself as `winner repos allow` or `block` would, and reports a choice it refuses with the reason. A laptop enrolled with `--no-service` keeps using `winner repos`.

5.6.2 Changed

- Enrolling a laptop whose hostname contains a comma is now refused, because the Dashboard's machine filter would read that one name as several laptops.
- `fleetview-deploy`, the host deploy timer, now installs each release from its signed server bundle placed in `/srv/fleetview`, through the bundle's own `scripts/upgrade-server.sh`, and goes back to a pinned previous release through `scripts/rollback-server.sh`, which restores the database as it was before that upgrade. Only `FLEETVIEW_PIN` in the running release's `.env` rolls back; a release named on the command line must be newer. It needs no registry token and pulls none of our images, it does not retry a failed upgrade every tick (an upgrade still migrating is slow, not failed, and each tick runs its `--finish` until it is done), and it proves that the running backend is the image loaded from the bundle with every migration applied, the web server is the pinned stock Caddy base running the bundle's binary, and Postgres reads the certificate mode's certificates; in the DNS certificate mode also that the certificate helper holds a certificate and the edge is told of each renewal, as the installer checks. Each tick still checks the web server, the edge, the proxy and the certificate wiring, and starts the release's services again from its folder when one has drifted, unless the certificate settings in its `.env` are not the ones it was last proved with, a rollback is under way, or the database is fenced for a restore. It refuses to start a compose file `COMPOSE_FILE` names that anyone but root can write.

5.6.3 Fixed

- A server installed from the bundle now has the server diagnostics command: `sudo ./scripts/diagnostics-server.sh`, run in the folder of the release that runs now, writes into your home directory the same redacted file for a support case that `sudo fleetview-diagnostics` writes on a server set up with the host scripts, and names its path. It needs `jq` on the server.
- The guide's Model and telemetry, Connect your identity provider, Onboard on Auth0, Deploy and Operations pages now give the commands for a server installed from the bundle, Install the server says once how to read a host's commands there, and Configuration and Architecture name the `.env` check every install runs. Before, they gave only a host's `/srv/fleetview`, `make check-env` and an explicit compose file pair, which fail in an install folder or leave out the bundle's own compose file.
- A person's "You spent" on Prompt Coaching counts each session once when their laptop is known by more than one name, such as its old hostname and the ID a newer client records.
- `winner report` now sends a laptop's earlier hostnames, and from agentsview 0.43.0 its installation ID, when agentsview has closed the local session database cleanly before the hourly report; it used to send none then.
- `winner login` now says when the client is older than the control plane's sign-in and gives the step that updates it, naming the control plane's `winner` version, instead of saying the control plane returned an incomplete config; `winner update` now reads that version from such a control plane instead of failing the same way.
- The server install guide now says, at the step that fills in `.env`, that Prompt Coaching, Skills and Benchmarks show Soon until `FLEETVIEW_FEATURES` lists them, and that they fill only once a model answers; it used to leave all three showing Soon without saying why.
- Prompt Coaching, Skills and Benchmarks no longer treat the prompts a zcode agent wrote to its own subagents as prompts a member typed, and Benchmarks prices those subagents' replies in its subagent saving like any other agent's. Coaching results already judged for such prompts are not recomputed and stay on the Coaching page until they leave its window.
- The server install guide now gives a trial's laptops their own install and sign-in lines; the documented line only accepted an https control plane, so a trial's laptop could not install the client.

5.6.4 Requires of you

- **Operators:** on a hosted model, the upgrade spends model calls once, on every organisation whether or not it uses zcode: every Benchmarks task is replayed on each candidate Route and judged again, within `benchmark_daily_runs` a day, and typed patterns are renamed, twenty per sweep. Until its replays return, a task that showed Tested and a proven saving shows its estimate again. The subagent and context saving lines are missing until the first sweep after the upgrade. A Skills workflow found again keeps its name.
- **Operators:** only where `fleetview-deploy` runs: it now moves only a release installed from the bundle (`docs/install-server.md`) and takes each release's `winner-ai-server-<X.Y.Z>.tar.gz` from `/srv/fleetview`. On a host still on the registry deploy, do this before this release is published: stop the old deploy with `sudo systemctl disable --now fleetview-deploy.timer`, install the running release from its bundle, first carrying `/srv/fleetview/.env` into the release folder (root's, mode 0600), then run the host scripts' installer from a checkout that holds it, which starts the timer again. The host scripts are not in the bundle. Left running, the old deploy copies this release's files over `/srv/fleetview` and hands over to the new script, which fails every tick

while the containers stay on the previous release. `/etc/fleetview/ghcr-token` is no longer read, and can be removed. The deploy keeps the certificate mode the installer set up and never switches it.

- **Operators:** nothing, unless a host overlay overrides `OAUTH2_PROXY_SKIP_AUTH_ROUTES`: it must add `^/api/v1/devices/repos$` and `^/api/v1/devices/repo-choices$`, which laptops now call.
- **Members:** nothing; the daily update brings the new client.

5.7 0.12.0 - 2026-10-30

This is the first release you can install on your own server from the signed self-hosting bundle. Releases 0.10.0 and 0.11.0 were prepared but never published, so an installation upgrading from 0.9.0 takes their changes with this one: their own sections below list them in full, and **Requires of you** here gathers what each asks of an operator.

5.7.1 Added

Signing in through your own identity provider

- A production installation can sign people in through JumpCloud or any other OpenID Connect provider, not only Auth0. The sign-in gate's issuer, scopes, groups claim and the hosts sign-out may send a browser to are `.env` settings (`OAUTH2_PROXY_OIDC_ISSUER_URL`, `OAUTH2_PROXY_SCOPE`, `OAUTH2_PROXY_OIDC_GROUPS_CLAIM`, `OAUTH2_PROXY_WHITELIST_DOMAINS`), each defaulting to Auth0's, and sign-out ends the provider's session at the address `OIDC_END_SESSION_URL` names. The `.env` check refuses an end-session address the gate would not send a browser to, an issuer or end-session address that is not https on an https installation, and a whitelist entry for a whole public suffix (`.com`, `*.co.uk`).
- Each person is keyed on the provider's issuer and their id there together, so a second provider presenting the same id is another person, and an Auth0 installation keeps every person and laptop through the upgrade with nothing to do. With a provider other than Auth0, leave both `AUTH0_MGMT_` keys blank and name the Managers with `FLEETVIEW_MANAGER_GROUP`; the `.env` check and the server refuse the Management keys there. Keep the issuer once people have signed in: a new one meets everyone as a new person.
- On an installation with no Auth0 Management credentials, the control plane reads a person's role from the sign-in provider's groups: a person in the group `FLEETVIEW_MANAGER_GROUP` names is a Manager, and where sign-in brings no usable groups the addresses in `FLEETVIEW_MANAGER_EMAILS` are. Settings shows both and the groups claim under Features and host. The `.env` check refuses `FLEETVIEW_MANAGER_EMAILS` unless `OAUTH2_PROXY_ALLOW_UNVERIFIED_EMAIL` is `false`. An installation with Auth0 Management credentials resolves roles as before.
- On an installation with no Auth0 Management credentials, a Manager removes a person on Admin: their laptop passes end, their laptops' database logins are dropped and the row reads `revoked`. Restore makes them a Member again with no laptop until they run `winner login`. With Management credentials, Admin says people are removed in Auth0. A removed person who signs in is told they were removed from the team. Revoking one laptop also ends its laptop pass, and the person's other laptops keep reporting.
- The control plane serves its own laptop login on any sign-in provider: a laptop starts a login, a signed-in person approves its code at `/login/approve`, and the laptop collects a laptop pass once. `winner login` signs in through it whatever the provider: it prints a code and the complete approval address, which work in any browser on any

machine, so it also works over SSH. The hourly reporter keeps the laptop pass and asks no sign-in provider for a token. With `--no-service`, a reporter an earlier login installed is handed the new pass. A laptop that last signed in with Auth0 keeps reporting as before until its next `winner login`.

- A trial installation signs in through a bundled, stock Dex under the compose profile `trial`: two users in `deploy/dex/config.yaml`, `manager@example.com` (a Manager) and `member@example.com`, password `winner-trial`, behind the same sign-in gate and on loopback only. The `.env` check and the control plane refuse an https base URL with the trial on or with the bundled Dex as the issuer. Four gate settings, `OAUTH2_PROXY_SKIP_OIDC_DISCOVERY`, `OAUTH2_PROXY_LOGIN_URL`, `OAUTH2_PROXY_REDEEM_URL` and `OAUTH2_PROXY_OIDC_JWKS_URL`, default to discovering from the issuer, and the `.env` check refuses them on an https base URL.

Installing, upgrading and backing up the server from the bundle

- Each release attaches the self-hosting bundle, `winner-ai-server-<X.Y.Z>.tar.gz`: the compiled control-plane image, the stock agentsview and oauth2-proxy images at their pinned versions, the web-server binary for linux/amd64, the compose files, the environment template, the configuration the stack mounts, the notices file and the documentation PDF. Its `SHA256SUMS` lists every file and is signed with the release key, in the same namespace as the `winner` download's. The database, web-server base and local-model images are not in it; the server pulls those itself, by digest, from Docker Hub with no login.
- The bundle installs itself: `./scripts/install-server.sh` in the unpacked bundle checks the bundle's signature against the release key it carries, never one from the bundle, then every file against `SHA256SUMS`, all before it loads any image, and refuses a bundle with no signature, a changed `SHA256SUMS` or a signature by another key, naming which. A server without `ssh-keygen` is told to install OpenSSH 8.1 or later. Checking the bundle by hand first, as the install guide shows, is still how you know the installer itself is ours. It then runs `./scripts/check-host.sh`, which names every fault of the server in one pass, one line each (linux/amd64, Docker Engine 28.0 and Compose 2.24.4 or later, free disk, ports 80, 443 and 5432 free, Docker Hub in reach, the clock within 5 seconds and a working `ssh-keygen`), and stops on any. It starts the stack and proves the web server runs the bundle's binary and the control plane reports the bundle's release. It refuses an environment file the check refuses and an image Docker Hub will not serve, each before anything starts, and fails rather than reports success when an org schema holds code stock agentsview does not create.
- An operator upgrades a server installed from the bundle with one command, `scripts/upgrade-server.sh <new bundle>`. It checks the new bundle's signature and every file, the new release's environment check and Docker Hub images, and backs up the database, all before it touches the running release. It then loads and starts the new release and waits for it to report the new version with every migration applied. On a failure it puts the previous release's containers back and prints the restore command with the backup's path. The installer and the upgrade command refuse a bundle older than the release the server runs, saying that migrations do not reverse.
- An operator rolls a server back to the release an upgrade replaced with one command, `scripts/rollback-server.sh`, from the newer release's folder. It checks the previous release's signature and every file, refuses a backup that is partial or marked failed, says that everything written since the upgrade will be replaced and waits for the operator to confirm. It then restores the pre-upgrade backup, keeping laptops out and the database as it was before the rollback in a backup of its own, and starts the previous release only once the restore has passed. A laptop enrolled since the upgrade must run `winner login` again, and a Device revoked since must be revoked

again. Known limit: sessions laptops pushed between the upgrade and the rollback are lost from the archive, since stock agentsview does not push them again; they are in the pre-rollback backup.

- The bundle carries `scripts/backup-server.sh <file>`, which writes one tested, owner-only backup of the whole database, and `scripts/restore-server.sh <file>`, which refuses a partial or failed backup, backs up what it replaces, keeps laptops and the stack out until the restore matches the backup, and names any Device revoked since the backup was taken. Both refuse a database holding a CHECK constraint or generated column stock agentsview does not create, which a restore would run as the database superuser, and the deploy audit reports one.
- The bundle installs as a trial: with `trial` in `COMPOSE_PROFILES` and the trial block of `.env.example`, `scripts/install-server.sh` runs the stack on `http://localhost:4180` with the bundled Dex, no edge and nothing published beyond loopback. The `.env` check refuses a trial that stacks the production overlay, and the installer refuses a trial in a folder that still holds a production install. A trial runs as its own compose project (`fleetview-trial` unless `.env` names one), so a production install made later on the same machine never shares its database.
- An installation serves one organisation, the one `AUTH0_TEST_ORG_ID` names, configured once. The dashboard takes that organisation from the control plane at start, and nothing in the compose files or the web-server configuration names an organisation, schema or role. A manager of any other organisation gets a 403 at the Dashboard.

Private networks and the DNS certificate mode

- `CERTIFICATE_MODE=dns` in `.env` starts a stock certificate helper (lego) beside the edge. It obtains one certificate for `PUBLIC_HOST` and `PUBLIC_PG_HOST` through a TXT record in their public Route 53 zone (`AWS_HOSTED_ZONE_ID`, `AWS_REGION`), using the server's instance role, and keeps it renewed. The edge serves that certificate for both names and asks no certificate authority for one, so the server needs no inbound port 80; port 80 still redirects to https. The edge presents each renewal within a minute without a restart, and Postgres within five minutes, so enrolled laptops keep verifying it with nothing pinned. Until the first certificate exists the edge serves nothing. The default, `http`, changes nothing.
- The bundle installs in the DNS certificate mode: the installer starts the certificate helper, waits for its first certificate and fails, saying why, when the helper cannot get it. It records the mode's profile in `.env`, so a later `docker compose up -d` in the bundle folder keeps it.
- Before such an install, `scripts/check-dns-zone.sh` on the server checks the certificate's prerequisites: with the server's instance role it writes one test record in the Route 53 zone, reads it back from the zone's name servers and removes it, then checks the certificate authority answers and the helper's image can be fetched. It prints a pass or fail line per check and no credential, and changes nothing else.
- The `.env` check refuses a `CERTIFICATE_MODE` other than `http` or `dns`, and in the `dns` mode an `.env` without `AWS_HOSTED_ZONE_ID` and `AWS_REGION`, with a base URL that is not https, or with a `PUBLIC_HOST` or `PUBLIC_PG_HOST` that is not a lower-case DNS name. No AWS key is asked for.
- Where laptops reach Postgres through a VPN and share its addresses, the operator can declare the VPN's ranges in `VPN_RANGES`. The host watch then counts refused logins from them apart, against `AUTH_FAIL_VPN_LIMIT` (60 in ten minutes), and a second fail2ban jail, `fleetview-postgres-vpn`, bans them on that count, so one laptop with a

stale password no longer locks out every laptop behind its address for an hour. A deploy refuses ranges it cannot trust. With nothing declared, nothing changes.

Amazon Bedrock

- An operator can run Prompt Coaching, Skills and Benchmarks on Amazon Bedrock (`FLEETVIEW_LLM_PROVIDER=bedrock`, `FLEETVIEW_LLM_REGION`), signed with the server's instance role or a Bedrock API key. A team's own model stays OpenAI-compatible. The model guide gives the settings, the instance role's policy, what the AWS account needs and what each Bedrock error means. No Claude model on Bedrock has passed the model check yet.
- `fleetview-model-check` checks the server's model before Prompt Coaching is switched on, from the installed stack alone: it runs the coaching judge over the labelled set the release ships, compares the scores with the release's baseline and ends `verdict: pass` or `verdict: fail`, with the run's cost in calls and tokens. It prints no prompt and nothing the model wrote. `--limit N` is a quick try that is never a verdict.

Members' own pages

- A Member's Home shows their own sessions and nothing of the team's: sessions over time, today and in the range, output tokens, sessions by coding agent, active sessions, when their data was last seen, their latest sessions, and how many of their own prompts were flagged while Prompt Coaching is on. Only sessions the control plane has matched to them are counted. A Member with none yet is told why and pointed to their laptops.
- A Member can open Prompt Coaching at `/coaching` for the findings on their own prompts and what their own sessions spent, with no person picker and nothing of the rest of the team.
- A Member can open Skills to see the repeated workflows found in their own prompts, with their own evidence only, and download an approved Skill made from one of them.
- A Member can open Benchmarks to see their own prompt patterns: each pattern whose latest run on its usual model was one of their own sessions, and what a cheaper model did when it re-ran their own example. No figure counted across the team is shown, and Run benchmark stays a Manager's.
- `/devices` shows a person their own laptops: each one's name, when it was enrolled, and whether it is still enrolled or was revoked. Someone with no laptop is shown how to run `winner login`.
- `/repos` shows the repos each of a person's laptops last reported, and lets them allow or block each one. The page only queues the choice for the laptop, which alone applies it; until the laptop reports back, the page shows the choice as pending, or the laptop's reason if it refuses. A name that stands for more than one checkout shows as refused and cannot be allowed.

Choosing which repos a laptop sends

- `winner login` at a terminal on a laptop with no repo choice lists the laptop's repos with none ticked and sends only the ones the developer ticks; quitting the list saves nothing and sends nothing. Without a terminal it chooses nothing, sends nothing and prints the command to choose. A laptop that has chosen keeps its choice and is not asked again.
- `winner repos` and `winner repos list` show each repo agentsview knows on the laptop, grouped into sent to your team, chosen but waiting for `winner repos apply`, waiting for you, and not chosen, with its session count and

the checkouts behind it, and take a filter (`winner repos list web`). They need git 2.36 or newer and say so on an older one.

- `winner repos pick` opens the same list, filtered by typing `/text` , ticked with numbers and ranges, `a` and `n` ; `winner repos allow` and `winner repos block` take repo names (`web-app` is read as `web_app`) or `--all` . `winner repos mode ask|block` says what happens to a repo found later: `ask` , the default, keeps it unsent and lists it as waiting; `block` keeps it unsent without asking. Allowing a repo says its earlier sessions are sent too; blocking one says its new sessions stop and those already sent stay.
- `winner repos apply` applies a changed choice without signing in again: the background push stops, takes the new list and resumes. A push stopped with `winner push stop` stays stopped and takes the list when started.
- A repo name that several checkouts share, Cursor's folder names included, is never sent: allowing it would send all of them. `winner repos separate <repo>` , or allowing the name at a terminal, asks a new name for each checkout and offers to allow each once agentsview lists it. A name already sent cannot be separated: what was sent stays merged under it. `winner repos rename <repo>` changes nothing and says why: renaming a repo already sent would make the next push delete what was sent from your team's server.

Support and the guides

- `winner diagnostics` writes one file for a support e-mail: the `winner` and agentsview versions, the push service's state, the control plane the laptop enrolled against, the newest push failure and the last lines of the push log, with passwords, tokens, e-mail addresses, paths and machine names removed. It sends nothing; read the file before you attach it.
- On a server set up with the host scripts, `sudo fleetview-diagnostics` writes one file for a support case: the build, every setting on the Settings ledger with the layer it came from, each service's image and state, and each service's last log lines, redacted. Keys show only as set or not set, and the stack's secrets are masked wherever their exact values appear; any other secret a log line carries is caught only by pattern, so read the file before you send it. It sends nothing.
- When `winner login` or `winner update` fails on an enrolled laptop, `winner` sends a short report of the command, the step, the HTTP status, the kind of error, the versions, the operating system and the processor type to the control plane it enrolled against, and nowhere else; `winner login` prints a notice saying so. The report never holds the error's text, a path, a URL or a repository name, and never changes how the command ends.
- New guides: a self-hosting overview and prerequisites page; Install the server, from the signed bundle start to finish; Network, for choosing between a public deployment and a private one reached over a VPN; Connect your identity provider, with a JumpCloud walk-through and a checklist for a JumpCloud administrator, both written from JumpCloud's documentation and not yet followed on a live organisation; backup, restore and uninstall; a laptop client guide for macOS and for Ubuntu or Debian; troubleshooting for an installation from the bundle; and getting started and getting help, whose support address and response targets are still to be confirmed. The Auth0 steps stay on Onboard on Auth0.

Licence

- Settings has a Licence section. A Manager uploads the licence file Winner AI sent, and the page shows the customer, licence id, issue and expiry dates and seats. A file that was changed, signed by another key, or issued before the licence in force is refused with the reason.

- Settings shows how many seats are in use: the people with at least one enrolled laptop, each counted once however many laptops they have. When the licence's seat number is exceeded, Settings and Admin both say by how many; nobody is refused, and the difference is settled at renewal. Nothing reports the count to Winner AI.
- Until a licence is installed and in date, the Manager pages carry a banner naming the expiry date (or saying that none is installed), and for the 30 days of grace after that, the day Coaching, Skills and Benchmarks switch off. Without a licence the 30 days count from the installation's first start, which a restart or an upgrade does not move.
- After the grace, Coaching, Skills and Benchmarks show a page that says the licence has lapsed: their jobs stop, their actions and Skill downloads are refused, the sidebar no longer lists them, Home hides their figures, and Settings shows them off with the reason. `fleetview-jobs replay` and `fleetview-jobs gate` refuse a task while Benchmarks is off for its team. Nothing is deleted: uploading a valid licence brings them back with everything they had. Sign-in, the roster, laptops, pushes, downloads and the Dashboard keep working.

5.7.2 Changed

- Anyone your identity provider admits can now sign in. A Member sees their own pages, listed above; every manager page and the Dashboard answer them with a page saying the Manager role is needed, and show no team data.
- A sign-in lasts one hour, not a week. When it ends the gate sends the person back to your identity provider, which admits or refuses them again, so a person you remove loses web access within the hour. A provider that keeps its own browser session signs people straight back in. `FLEETVIEW_GATE_SESSION_SECONDS` sets the lifetime, and Settings shows it.
- An installation that signs in through Auth0 may leave `AUTH0_MGMT_CLIENT_ID` and `AUTH0_MGMT_CLIENT_SECRET` blank and take each person's role from the gate's groups, as one on any other provider does: `FLEETVIEW_MANAGER_GROUP=manager`, the Auth0 role the post-login Action already sends. The `.env` check then requires `FLEETVIEW_MANAGER_GROUP` or `FLEETVIEW_MANAGER_EMAILS`, and refuses one Management value set without the other.
- On Auth0, the post-login Action refuses a sign-in to the control plane through any Organization but the installation's own, or through none, once it is redeployed from this release with its two secrets. Laptops are not affected.
- A laptop sends sessions only from the repos its developer chose. `winner login` writes the chosen list into the laptop's agentsview config and restarts the background push; a laptop with no choice, or one `winner` cannot read, sends no session at all. The choice is a file on the laptop, `~/.agentsview/fleetview-repos.json`. The hourly start-commit report covers only the chosen repos too.
- A laptop enrolled before repo selection keeps sending the repos it has, and is asked only about repos found later: its next `winner login` to the same organisation, or its first `winner update` after the one that installs this release, chooses every repo agentsview lists on it then, once. If agentsview holds sessions under a name outside that list, or the check cannot complete, the laptop keeps sending every repo, says why, and checks again at the next update or login.
- The laptop's repo picker and `winner repos help` say what is sent, that managers see it, and that sessions already sent stay with the team and cannot be withdrawn. Of the repos not chosen they say "Sessions from repos you don't choose stay on this laptop, unless one of them carries the name of a repo you chose.", and, when the laptop's search index embeds through a server other than the laptop itself, that their text goes there.

- The images your server pulls from Docker Hub are pinned to exact digests: PostgreSQL 16.15 with pgvector 0.8.7, Caddy 2.11.7 at the edge, and Ollama 0.6.8. The dashboard web-server image is built from pinned Caddy 2.9.1. An upstream fix to any of them arrives with a Winner AI release rather than on its own.

5.7.3 Removed

- The second organisation's dashboard slot: the `acme` compose profile, the `pg-serve-b` service and its two `_B` keys. On a deployment that ran it, that organisation's dashboard stops at the upgrade and its data in Postgres is untouched; its Docker volume and `.env` lines are no longer read and may be removed by hand.

5.7.4 Fixed

- A `winner repos separate` that stops partway no longer removes a folder rule under the name you chose for a checkout once that name was sent: removing it would have made the next push delete the sessions stored under it since.
- Prompt Coaching and Skills no longer judge or show the block the Codex app sends when a person edits a writing block by hand; a prompt that is only that block is skipped, and Benchmarks counts it as an agent's. Coaching re-judges the last 30 days under the daily budget, so its findings thin out until the sweeps catch up.
- A person removed in Auth0 also loses their laptop passes at the next sync. A pass they had not yet used to enrol used to work again once Auth0 added them back.
- On Linux, when `winner login` cannot reach a systemd user manager, the hint it prints carries your user name (`loginctl enable-linger alice`), so it works on a machine with no login session.
- A laptop enrolling with an older `winner` on an installation without Auth0 Management credentials is told to update `winner` and run `winner login`, instead of getting a server error.
- On an installation from the bundle, the installer, `backup-server.sh` and `restore-server.sh` reach only the installation in their own folder. A shell that exported `COMPOSE_FILE`, `COMPOSE_PROJECT_NAME`, `COMPOSE_ENV_FILES` or `COMPOSE_PROFILES` could point them at another installation's database.

5.7.5 Telemetry

Telemetry is off, and stays off unless your operator enters values Winner AI supplies. Error reports, timings and cron check-ins leave only once `FLEETVIEW_SENTRY_DSN` is set; usage counts only once `FLEETVIEW_POSTHOG_KEY`, `FLEETVIEW_POSTHOG_HOST` and the telemetry secret are all set. The telemetry secret the installer writes into `.env` (`FLEETVIEW_TELEMETRY_SECRET`) only hashes ids and sends nothing by itself. Settings shows the state of each in its Telemetry group. What is sent, once switched on:

- Error reports from the control plane and the jobs service: the failure's type, its code location, the route template and a few closed tags, never request bodies, local variables, error text or raw ids. Timings for 2 in 100 requests and jobs passes go with them, never statement text, an address or a prompt; `FLEETVIEW_SENTRY_TRACES_SAMPLE_RATE` changes the share and `0` turns timings off.
- With the telemetry secret set, the jobs sweep checks in with Winner AI's cron monitors at its start and end, and `fleetview-checkin` lets a host job such as the backup do the same: its monitor, status, schedule, duration and

environment, never an error's text.

- Usage counts under hashed ids (a page with no person to hash uses a random id per browser tab) and never with text: a Manager's saved changes, each pass run's outcome, each model call's tokens and timing, when the installation is installed or upgraded, when usage counts are switched on, when a laptop enrols, when the licence state changes, one page view per control-plane page naming its route (never its address or query), and one heartbeat a day with the version, the seats in use, the laptops per push state and counts of laptop versions, systems and processors. Never a person's name, a host name or text. The page analytics library is a pinned build the control plane serves itself; with usage counts off no page loads it, and page recording stays off whatever `FLEETVIEW_POSTHOG_REPLAY` says.
- A laptop's failure report goes only to the control plane it enrolled against, which passes it on, closed fields only, at most ten an hour per laptop and never stored or logged, only while error reports or usage counts are on.
- agentsview's own usage ping and update check are off on the server and on laptops.

`docs/model-and-telemetry.md`, "Telemetry", lists every field and every event.

5.7.6 After you install

- Run the checks in `docs/install-server.md`, "Check the installation": `/health` answers `ok`, every service in `docker compose ps` is healthy, Postgres presents a certificate a laptop can verify, a Manager signs in and out, and a first laptop's push shows on Admin with a session count or a last-push time.
- In the DNS certificate mode, give Postgres up to five minutes after the first certificate before a laptop's first push; until then laptops cannot verify it.
- A new laptop sends nothing until its developer ticks at least one repo in the list `winner login` shows, or the Dashboard stays empty.
- Upload the licence file Winner AI sent on Settings, and check that its seat number covers the people with an enrolled laptop.
- Before you switch Prompt Coaching on, run `fleetview-model-check` against your model and switch it on only on `verdict: pass`.
- Check that Settings' Telemetry group shows what you expect: everything off unless you entered the values Winner AI supplied.

5.7.7 Requires of you

A new installation: follow `docs/install-server.md` from step 1. The upgrade steps below do not apply; those under **Every installation** do, each where it says it applies.

An installation upgrading from 0.9.0: this release migrates the control database at its first start, with nothing to run by hand. Do these first, in this order.

- **Operators:** take a backup (`sudo fleetview-backup`) before you deploy. A rollback to 0.9.0 after the migration is not code-only: expect to restore that backup.

- **Operators:** add two lines to `.env`, with your tenant's domain written out, or the deploy's `.env` check refuses it and names both: `OAUTH2_PROXY_OIDC_ISSUER_URL=https://<AUTH0_DOMAIN>/` and `OIDC_END_SESSION_URL=https://<AUTH0_DOMAIN>/v2/logout`.
- **Operators:** the sign-in proxy no longer admits only managers, so who may sign in at all is your identity provider's decision. Check that it admits only your team before you upgrade: on Auth0 the sign-in proxy's application must require an organisation, and the organisation's enabled connection must keep `assign_membership_on_login: false`, or anyone who registers and verifies an address can join the organisation at the login prompt (`docs/onboarding.md`, "Who may sign in", has the check).
- **Operators:** on Auth0, redeploy the post-login Action from this release with its two secrets, the `.env` values of `AUTH0_PROXY_CLIENT_ID` and `AUTH0_TEST_ORG_ID` (`docs/onboarding.md`, "The post-login Action", has the commands and a check). Before you do, mark verified only the accounts you created yourself and know; the Action refuses an unverified one. Until you redeploy, nothing changes; an Action deployed without the secrets refuses every sign-in until they are set. Then sign in yourself to check it. With the Management credentials blank, also Remove from Admin's roster anyone who is not a member of this installation's Organization (`auth0 orgs members list <org id> --number 1000`; the default lists only 100).
- **Operators:** upgrade first, then switch provider. The migration stamps every existing person with the issuer the installation signs in through at that moment, so moving from Auth0 to another provider in the same step stamps them with the new one. Once people have signed in through the new provider, they are new people: their old rows keep their laptops and history, and nothing carries them across.
- **Operators:** `PUBLIC_BASE_URL` must be the bare address a browser shows, such as `https://devtool.example.com`: lower case, no trailing slash, no path and no default port, or the deploy refuses it.
- **Operators:** the production overlay pins the compose subnet, and the first deploy takes the whole stack down once to move its network there. The server's own networks, VPC and routes must not use that subnet. Afterwards, narrow the `fleetview-postgres` jail's `ignoreip` to the pinned subnet (`docs/network.md`, "The Postgres jail").
- **Operators:** a deploy fails when a team schema holds code stock agentsview does not create; `docs/troubleshooting.md`, "Code in an org schema that agentsview did not create", says how to find and clear it. The `postgres` and `template1` databases admit only a superuser or a role granted `CONNECT` there by name.
- **Operators:** use `fleetview-jobs` and `fleetview-provision` in any script of your own; `python -m fleetview_backend.jobs` and `python -m fleetview_backend.provision` no longer run in the backend image. A mirror you point `FLEETVIEW_AGENT_URL` at must publish the release's `SHA256SUMS.sig` beside `SHA256SUMS`, unchanged, or the laptop installer refuses it.
- **Operators:** on a hosted model, the upgrade spends model calls once: the coaching window is judged again, and every benchmark task is replayed and judged again under the daily limit.
- **Operators:** once Linux laptops are enrolled, do not roll back below 0.11.0. `winner` 0.10.0 cannot update itself on Linux, so laptops told to converge to it stay on it after you roll forward again; if you did, re-run the one-line installer on each Linux laptop, which keeps its enrolment and its running push.
- The 0.10.0 and 0.11.0 sections below list the rest, each marked for whom it applies, such as the demo seed and a `fleetview-postgres` jail set up before 0.10.0.

Every installation:

- **Operators:** upload the licence file Winner AI sent on Settings within 30 days of the installation's first start (for an upgrade from 0.9.0, its first start on this release), or the Prompt Coaching, Skills and Benchmarks jobs stop, and the sidebar and Home drop them, until you do. Nothing is deleted meanwhile.
- **Operators:** only on Amazon Bedrock signed with the instance role: set the instance's metadata hop limit to 2 (`aws ec2 modify-instance-metadata-options --instance-id <instance-id> --http-put-response-hop-limit 2 --http-tokens required`). A container is one hop further from the metadata service than the server, and at hop limit 1 every model pass fails. `docs/model-and-telemetry.md` , "Amazon Bedrock", says what that opens to every container.
- **Operators:** only in the DNS certificate mode: give the server the instance-role policy `docs/network.md` gives, run the deploy checks from a machine on the VPN, and expect laptops to push only once Postgres has taken the first certificate, up to five minutes after the edge has it.
- **Operators:** only where laptops reach Postgres through a VPN and share its addresses: set `VPN_RANGES` in `.env` to the VPN's whole associated subnets, add them to the `fleetview-postgres` jail's `ignoreip` , and add a second jail named `fleetview-postgres-vpn` with a higher count and a short ban (`docs/network.md`). Otherwise nothing.
- **Operators:** nothing to set for the sign-in lifetime: `FLEETVIEW_GATE_SESSION_SECONDS` is one hour by default. Raise it only knowing it is also the longest a removed person keeps web access.
- **Operators:** nothing, unless `.env` sets only one of `AUTH0_MGMT_CLIENT_ID` and `AUTH0_MGMT_CLIENT_SECRET` : set both or leave both blank, or the `.env` check refuses it.
- **Operators:** only if your deploy account is to run `fleetview-diagnostics` : add it to that account's sudo allowlist beside `fleetview-ps` and `fleetview-logs` .
- **Members:** on macOS, nothing to reinstall. On Linux, install with the one-line command and run `winner login` ; this is the first published release whose installer runs there. A laptop already enrolled keeps sending the repos it has: its next `winner login` or `winner update` keeps every repo agentsview lists for it. A repo found after that, and every repo on a new laptop, is sent only once chosen, at the list a login shows at a terminal or with `winner repos pick` (`docs/laptop-client.md`). You are also sent back to sign in about once an hour, silently where your provider keeps its own session; a form left open across that moment is lost.

5.8 0.11.0 - 2026-10-15

5.8.1 Added

- The one-line installer now installs `winner` and the control plane's `agentsview` on Linux: Ubuntu 20.04 or later and Debian 11 or later, amd64 or arm64, into `~/.local/bin` with no `sudo` . Before downloading anything it refuses a C library agentsview cannot run on (musl, or glibc older than 2.28) and an `ssh-keygen` too old to check the release signature (older than OpenSSH 8.1, as on Debian 10), and says which.
- `winner update` now installs on Linux (amd64 and arm64): it brings `winner` and agentsview to the control plane's versions, verified against the same signed and upstream checksums as on macOS, and restarts the background push. A Linux `winner login` whose agentsview is ahead of or behind the control plane now points at `winner update` instead of a manual download.
- On Linux, `winner login` now installs two systemd user timers: `winner-update.timer` , which runs `winner update` at sign-in and daily at noon (a missed noon runs at the next start), and `winner-reporter.timer` , which

reports this laptop's names hourly. `--no-service` installs neither. Without systemd, login prints cron lines with the absolute path to `winner` instead, and says nothing was installed.

- The Dashboard's web server and the production edge now carry health checks. When either cannot run its program, or refuses its configuration, `docker compose up --wait` fails and `docker compose ps` no longer shows it healthy, where Compose used to report a running service; the host watch reports one that runs but stops answering as unhealthy.
- Settings shows a Telemetry group: whether error reports and usage counts are on, where they would go and whether the identity hashing secret is set, never a key. Six operator variables configure them (`FLEETVIEW_SENTRY_DSN`, `FLEETVIEW_SENTRY_ENVIRONMENT`, `FLEETVIEW_POSTHOG_KEY`, `FLEETVIEW_POSTHOG_HOST`, `FLEETVIEW_POSTHOG_REPLAY`, `FLEETVIEW_TELEMETRY_SECRET`); nothing is sent yet.

5.8.2 Changed

- `fleetview-jobs run --once` now exits 1 when an organisation's sweep raised, where it always exited 0, so a script that reads its exit code sees the failure. `--org <Auth0 organisation id>` sweeps that organisation alone and exits 2 when no organisation has that id. `fleetview-seed-demo` passes it: a deployment's demo seed now sweeps only the demo org, and its sweep step fails when that sweep raised.
- `make seed-demo` is now the whole local demo in one command. It seeds the team, attributes every session, reads Team spent, runs one jobs sweep on the stack's local Ollama model, never the model `.env` configures unless `SEED_DEMO_CONFIGURED_MODEL=1` asks for it, and prints what Home, Prompt Coaching, Skills and Benchmarks now show. It refuses before seeding, with the fix, when the stack is not up and healthy, when the Ollama profile is off or its models are not pulled, when `FLEETVIEW_FEATURES` leaves out Coaching, Skills or Benchmarks, or when it cannot stop the stack's jobs service for the run.
- The production TLS edge runs Caddy 2.11.7, up from 2.9, which brings upstream fixes, security fixes among them. Request headers are now capped near 16 KiB over HTTP/2 and 20 KiB over HTTP/1.1, where 2.9 allowed 1 MB: past the cap the edge answers `431 Request Header Fields Too Large`, or over HTTP/2 may close the connection instead. It aborts a connection whose upload or download makes no progress for a minute; a stream that is quiet between writes is not affected. It drops request header names that contain a dot or an underscore. Nothing to do on upgrade: the edge keeps its certificate and ACME account.
- The backend image now runs the control plane as one compiled module instead of Python source. Its templates and migrations stay plain files. The API schema no longer carries operation and model descriptions, `fleetview-jobs --help` and `fleetview-provision --help` no longer print their description paragraph, and the response schema sent to your model for Skills names and Benchmarks labels no longer carries a description.
- The backend image no longer carries the demo seed scripts; `fleetview-seed-demo` copies them in from a directory you name.
- A team's model endpoint must now be plain ASCII with no spaces or control characters, and its host may not have an empty label (`a..b`). A host written with a non-ASCII dot or a trailing space could otherwise name an address on this host's own network. An internationalised name still works in its `xn--` form.

5.8.3 Requires of you

- **Operators:** `python -m fleetview_backend.jobs` and `python -m fleetview_backend.provision` no longer run in the backend image. Use `fleetview-jobs` and `fleetview-provision`, as the 0.10.0 notice asked.
- **Operators:** `fleetview-seed-demo` now needs `--seed-files <dir>`, a directory holding the six demo seed scripts of the deployed release, owned by root and writable by no one else.
- **Members:** on macOS, nothing to reinstall or sign in again. On Linux, this is the first release whose installer runs there: install with the one-line command and run `winner login`. `winner update` replaces a release binary downloaded by hand as it does an installed one, but leaves a `winner` built from source alone, updating only agentsview, unless `--force` is given.

5.9 0.10.0 - 2026-10-08

5.9.1 Added

- Benchmarks now measures and prices tasks from agents that record only session-level token totals, such as Windsurf, VS Code Copilot, Roo Code, Zed and Goose, when a session holds a single prompt. Claude Code and Codex are measured as before.
- The Benchmarks page now says which of a team's agents it cannot measure (Cursor, Kiro CLI and Aider record no model) and no longer lists their tasks as untested opportunities.
- Enrolled macOS laptops report each session's start commit and branch to their control plane, the commit read from the reflog without touching the repository. A session with no commit reports the reason instead, such as a folder that is gone or is not a repository. The folder's path never leaves the laptop.
- On macOS, `winner login` asks once whether winner may read Desktop and Documents when recent sessions ran there, after saying what it reads (the git reflog, nothing else). Allow lets the hourly reporter record start commits for those sessions; Don't Allow leaves them without one. Laptops with no session in either folder are asked nothing.
- `winner login` on macOS installs a daily `winner update`, so an enrolled laptop converges to its control plane's versions after a release without anyone running it; `--no-service` skips it.
- A configuration reference lists every variable in `.env.example` and the host's monitoring file with its meaning, default, whether it is required and which layers can set it.
- A manager sets on Settings how similar two prompts must be for Skills to group them, and how many prompts and sessions a workflow needs before it is shown. A change regroups on the next sweep, and the new groups are named within the naming budget over the sweeps after; skills already generated stay as they are, and a team that sets nothing sees the same workflows as before.
- A generated skill draft now lists, above Approve, each tool, command, service or domain it names that the prompts it came from never mention, so a manager can check it before approving. The lint result now says it checks the file's structure, not what the skill says.
- Each release now publishes static `winner` binaries for Linux on amd64 and arm64 beside the macOS one, listed in `SHA256SUMS`, and every control plane serves them for download without sign-in. The installer and `winner update` do not use them yet.
- The documentation has a page on pointing Winner AI at a hosted OpenAI-compatible model, checking it with the coaching eval before Prompt Coaching is switched on, and whether the stack sends telemetry (`docs/model-and-`

`telemetry.md`).

- The same page lists, call by call, everything Prompt Coaching, Skills and Benchmarks send to the model, and says what happens to a session agentsview flagged as holding a secret. A test per call holds the list to what the code sends.
- Prompt Coaching's Omit / Mark as wrong preview is now a real Disagree control: a manager marks a finding's category wrong or the prompt fine, the finding stops counting in that person's findings and figures at once and in their points to improve after the next sweep, stays listed under its category as a closed card marked Disputed, and is kept to measure the next rubric change. "Main points to improve" names the rubric version the findings were judged under.
- The documentation has a troubleshooting page that maps what an operator sees to its likely cause, the command that confirms it and the section that fixes it. The configuration reference now also lists every team setting on Settings, with its release default and who can change it, and the `winner` client's options.
- `fleetview-watch` reports a problem when the certificate served on 443 or on 5432 has fewer than `CERT_MIN_DAYS` (14 by default, set in the host's monitoring file) days left. Let's Encrypt no longer e-mails before a certificate expires, so this is the warning that renewal has stopped.
- Every deployment serves the documentation at `/guide/` to whoever signs in to the web app, with its fonts and assets included, so reading it sends nothing to another site; the sidebar links to it. Each release also carries the same pages as one PDF, `winner-ai-docs-<version>.pdf`.

5.9.2 Changed

- The jobs and provisioning commands now start as `fleetview-jobs` and `fleetview-provision`, for example `docker compose exec -T jobs uv run --no-dev fleetview-jobs rekey` or `docker compose exec -T backend uv run --no-dev fleetview-provision audit`. The jobs service, the deploy's audit step, the demo seed's sweep, the docs and every printed hint use them. The `python -m` forms still work on this release (see Requires of you).
- Re-running `fleetview-seed-demo` on a deployment already seeded, or `make seed-demo` locally, is expected to take minutes rather than hours. Each demo session's words and token counts now come from its own id, so only its timestamps move between runs; the deployment seed keeps each persona's `vectors.db` and prompt digests beside its `sessions.db` under `/var/lib/fleetview-seed-demo`, so an unchanged message is neither embedded nor judged again; kept Coaching verdicts move to their sessions' new start times; and the live personas push once more at the end, so Home's Active sessions reads above zero straight after the run. The first run after upgrading is the exception (see Requires of you).
- Prompt Coaching reaches the prompt feedback sooner. The band keeps its four figures and moves each one's sentence into "How to read these numbers", above that figure's explanation; "Main points to improve" is now a closed toggle under it that shows how many points there are; and each finding shows its first five cards, with the rest behind "Show N more".
- Each Benchmarks row now says whether a cheaper model can be tested on it now and, if not, why. "Can be tested now" means the row has a benchmark task to run; a prompt that carries everything it needs but is not a task reads "Cannot be tested" and says so. A prompt a person types, with no task to run, reads "Cannot be tested": typed work waits for a team to opt its repository in, which is not available yet, and the row adds what its latest example's laptop

reported about its start commit. The "could be tested" count and the Testable filter now count only rows with something to run, so both can be smaller than before. A pattern whose benchmark task no longer qualifies no longer offers a Run benchmark button that the click would refuse.

- Prompt Coaching no longer shows, for a newly judged prompt, a suggested rewrite that adds a number, file path or code name the original prompt never mentioned; the card says there is no rewrite to show instead, and its findings stay. The upgrade judges the coaching window again (see the last line of Changed), so every rewrite shown has been through the check.
- Home's heading is now the single line "Attention and live state."; the muted second line "Risk first, then current activity." is gone.
- The sidebar now opens with a larger Winner AI wordmark alone. The team name and organisation id that sat under it on every page are gone from the sidebar; Settings shows both.
- A nightly backup whose restore test fails now keeps its dump as `fleetview-<stamp>.sql.gz.failed` instead of under the good name, so choosing a dump from `/var/backups/fleetview`, or anything that globs `fleetview-*.sql.gz` there, can no longer take it for a backup (the off-box copy only ever took a passing run's dump). A dump is never deleted on a failure, and failed dumps are pruned on the same `KEEP_DAYS` window as good ones.
- The restore procedure now stops the deploy, watch and backup timers for the length of a restore, keeps laptops out by taking Postgres off TCP until the restored data is verified, checks the row counts before the stack comes back up, and takes login away from any device revoked after the dump was taken.
- The "Subagents on a cheaper model" and "Context re-read above the budget" lines under Could be saved now say when their figure was last computed. If the job behind them has not refreshed one for over a day, that line is marked Stale and left out of the Could be saved total on Benchmarks and Home, and it counts again as soon as the next sweep refreshes it.
- The Skills page opens on the candidates and the selected skill: the demo simulation panel moved to the bottom of the page. The candidates and library lists now scroll inside their own box, sized to the window, and a list reopens on the row just chosen.
- Leaving flagged sessions out (see Security) measures Prompt Coaching and Benchmarks again. After upgrading, the next sweeps judge the coaching window again once; replay every benchmark task again on every candidate Route, under the daily limit, and judge each replay; and rename typed patterns, twenty per sweep. Until its replays return, a task that showed Tested and a proven saving on Benchmarks and Home shows its estimate again. A Skills workflow found again keeps its name.

5.9.3 Fixed

- The backend and jobs containers no longer contact the Python package index when they start, so they start on a host with no outbound internet access instead of exiting.
- Coaching's "Your prompt" and the prompts it judges, and the Skills evidence and candidates mined from them, now hold only what a person typed. Text an agent wraps around a prompt (Codex approval reviews and their agent history, stop-gate task blocks, image attachments, subagent hand-backs, messages from other sessions, command templates, the Codex app's in-app browser context and heartbeat block) is cut, and a prompt that is nothing but that text, or opens with a hook, command or task block, is skipped, as are the Codex app's scheduled automation runs that open with its `Automation:` header. Coaching re-judges the last 30 days under the daily

budget, so its findings thin out until the sweeps catch up. The Benchmarks pattern inventory shows text from a prompt only when it opens with a marker an agent or tool writes, and then only that opening (see the Benchmarks lines below): the stricter rules can mistake a person's words for an agent's, and there that would show those words to managers.

- The installer now installs the agentsview version your control plane runs, verified against upstream's checksums and placed next to `winner` without `sudo`, instead of upstream's latest release, which `winner login` then refused for running ahead of the control plane. Where it cannot learn that version (the control plane reports none, is reached over plain `http` to a host that is not loopback, or a mirror install names no `FLEETVIEW_CONTROL_URL`) it now fails after installing `winner` and names the release page, instead of installing the latest.
- A laptop the roster has only just started tracking no longer reads Stale from an old embeddings heartbeat when it pushed a session change in the last day; it reads Pushing, on the roster and on Home, for the first day.
- A roster row no longer shows a session time that belongs to another row. When two rows claim one hostname, only the row the name counts for reads it, on the roster and in Home's status, so a revoked laptop no longer shows its replacement's fresh "last change" beside its Revoked pill.
- The Skills demo's Install button now reports itself disabled to assistive tech once it starts, instead of announcing "Delivered" as a button that can still be pressed; keyboard focus stays on it, and a reload still resets it to Install.
- Adding a member whose address already has an Auth0 user no longer fails with a server error. Admin adds that user to the organisation with the chosen role and no temporary password, and they sign in with their existing login or reset it. The add is refused, with nothing changed in Auth0, when the address or its Auth0 user is already on another organisation's roster on the same deployment, or the user is an Auth0 member of another organisation provisioned there. It is also refused when Auth0 has not verified the address and the user was not created by this control plane, is not in the organisation and is not on its roster, so a login someone registered for an address they do not hold never receives the organisation's role. Any other Auth0 failure is shown on the page.
- Benchmarks no longer marks a pattern whose runs read files as testable: it reads "Cannot be tested" and is left out of the band's "could be" count and the Testable filter. A pattern with a runnable task under its signature reads "Can be tested now" whatever its other runs needed.
- Skills candidates no longer count prompts from background runs, subagent sessions, sidechains or automation, nor agent replies stored beside a Codex prompt, so a workflow's prompt and session counts and its evidence are what people typed, as on Coaching. On the next sweep a candidate whose prompts no longer form a workflow is dropped, while one found again keeps its name and is refreshed with the new counts and evidence, as is the candidate behind a generated skill; dismissed candidates are left as they are.
- Benchmarks no longer describes the review prompts Claude Code composes for Codex as typed by a person: it reads who wrote a prompt from its text as Coaching and Skills do, and shows no text for those prompts, only a name. A prompt with no words, such as a lone screenshot, stays a person's. Slash commands, shell output and the agent's own interruption and continuation text are no longer called hooks. Pattern rows are rebuilt and the new ones named over the next sweeps.
- Benchmarks no longer shows words a person typed after a command, shell or reminder opening: a slash command's arguments, a command run in Claude Code's shell mode, and text typed after a reminder or task block are cut from the prompt shown, and a shell command's pattern shows a generated name instead while still being described as a shell command. The model that names a pattern is sent no more of such a prompt than the page shows, so a shell command or a slash command's arguments never reach it. A prompt whose opening block is

never closed, or holds a shell command or a slash command's arguments, shows no text and is not sent. Pattern rows are rebuilt and named again over the next sweeps.

- A jobs heartbeat URL (`FLEETVIEW_HEARTBEAT_URL`) that is plain http to anything but the stack's own hosts no longer stops the jobs sweep: the jobs service logs a warning and never pings it.
- The Benchmarks passes no longer stall the jobs sweep on a team with long sessions of many prompts. Finding which prompt each reply answers took time that grew with the square of a session's length, tens of minutes on a large team, and held up Benchmarks and every pass after it; it now takes seconds. What the passes store is unchanged, and nothing needs doing on upgrade.
- The Benchmarks passes no longer stall on one very long prompt. Grouping a prompt read all of it to keep its first 100 characters, and a long run of dotted or hyphenated words with no spaces took time that grew with the square of its length, 45 seconds for 160 KB; only the start of a prompt is read now. Which prompts group together is unchanged.
- A Benchmarks replay that timed out, or came back from the provider in a form that could not be read, now counts against the daily run limit. It had been treated as costing nothing, although the provider may have billed it. A replay the provider refused or never received still takes no slot.
- `winner push status` used to quote the oldest end of a progress line, so a long push read "1% … 10%" and dropped the current figure; it now quotes the latest redraw. A log that ended in a bare erase-line used to be quoted as nothing, and now quotes the readable line above it. `winner push uninstall` , and `winner push stop` on macOS, now pass what launchctl and systemctl printed through the same sanitiser before echoing it; on Linux a stop echoes nothing, because systemctl writes to the terminal itself.
- `winner push status` used to say "running" while every push was failing, because the reasons land in agentsview's own `pg-watch.log` , which it did not read; it now reads that file too. While the push is running and the last two or more pushes failed with none succeeding since, its first line reads "running (pid N), but the last N pushes failed:" with the newest reason, the "(pid N)" left out only when the service manager reports no pid, and its next line says when that failure was. A keyword-form `password=` is now masked too, in the quoted last line of every reading and in what `winner push stop` and `uninstall` echo through the sanitiser above; nothing else in the other readings changed, and `winner push logs` still prints the service's log as it is.
- Revoking a laptop no longer drops a column, a default or a whole shared table that leaned on a sequence, type or table that laptop created: those are now handed over first, tables and sequences to the team's device group and types to the schema's owner. What still has to go is named in the backend's log when it can be found.
- Off macOS (Linux included), `winner login` no longer tells a laptop whose agentsview is behind or ahead of the control plane to run `winner update` , which installs macOS builds only and refuses there. It links upstream's release page for the control plane's agentsview version instead, and `winner update` no longer tells such a laptop missing agentsview to rerun it.
- The reload database password logs in through psql and libpq again when it holds some non-ASCII text (a zero-width space, a character SASLprep prohibits, some right-to-left text). Since 0.9.0 the control plane prepared such a password differently from libpq, so the role refused it. The next backend start sets it again.
- The subagent and context lines of Benchmarks' "Could be saved", Prompt Coaching's Model fit estimates and its "Could be saved" band now date each figure by every price row it was priced from, as a range when they differ, instead of by the newest of those rows. A Benchmarks line stored before the upgrade keeps its one date until the

next sweep reprices it; a Model fit estimate is priced once, when its session is judged, so one judged before the upgrade keeps its one date.

- Benchmarks no longer offers Run benchmark where the click would be refused or the run silently skipped. When the team's model key cannot be read with this host's settings keys, the page says so and offers no Run; when the host does not lend its local model, the local Routes say so on their card, and a run asked for on one is refused instead of queued. A Route the page will not run reads Cannot be benchmarked, with the reason, instead of Ready to benchmark. When the team has used its daily run limit, each Route card says so and offers no Run, and the row reads "Cannot be tested" with that reason, until a slot frees.
- A Skills workflow's summary no longer opens by calling the engineer "He" or "She": it says "The engineer". The upgrade rewrites the summaries already stored the same way.
- A feature the host enables after a team saved Settings' Features row now runs for that team. The team's saved value used to list what it left on, so a feature added later stayed off for it; the row now records what the team switched off, and the upgrade converts every saved row. One choice cannot be recovered: a feature a team switched off that the host no longer enables at the upgrade comes back on if the host enables it again.
- Settings' Frontier model families list offers `gpt-5.6` as the release list spells it, with its session count, instead of a bare `gpt` that would mark every GPT model frontier. Each family counts a session once, where a session on two models one entry covers used to count twice.
- Settings' Routes editor answers a JSON client's Add row or Remove row with a 422 JSON error telling it to post `op=save`, instead of the HTML page; nothing is saved. Browsers are unchanged.
- `winner update` no longer tells a Mac still on agentsview's own push service that it was enrolled with `--no-service`, and no longer leaves that service running the agentsview it just replaced. On a Mac, every update that is not `--check` now replaces it with Winner AI's, as `winner login` does, though a Winner AI push stopped with `winner push stop` stays stopped; the update that installs this release still runs the old code, so the one after it does the replacing. The `--no-service` line appears only when neither service is installed. `winner push status` names agentsview's own service while it is there, and the command that replaces it: `winner update` on a Mac, `winner login` on Linux.

5.9.4 Security

- The hourly reporter no longer runs git. It works out where a session's repository really is and reads the repository's reflog file itself, so a link, a `.git` file naming another repository, a repository's settings, or anything inherited from whatever started the job can no longer make it report another repository's commit, or, on macOS, read a repository in a folder you did not allow. A Mac without the developer tools now gets start commits too, and no git version is needed on any system. Sessions inside a git submodule, and in a repository that uses git's reftable format or has a link in place of its `.git` or its reflog, are recorded without a start commit.
- A failed organisation bootstrap can no longer write the organisation's database owner password to the log or into the error it reports.
- The control plane's cross-team reads (Home, Coaching, Skills, Benchmarks, the roster's session counts and attribution) now log in as their own read-only Postgres role instead of borrowing the superuser's connection, so nothing a read runs can climb back to the superuser. A read refuses a team schema holding a view, a materialized view, a foreign table or a row-level-security policy (switching row-level security on without one is not refused), and

every start and deploy reports any function, view, operator, trigger, rule or policy there that stock agentsview does not create.

- Enrolled laptops no longer send agentsview's anonymous usage ping or ask GitHub for a newer agentsview, matching the server: every service `winner login` installs switches both off, and the agentsview config enrollment writes switches off the update check, the one of the two it has a setting for. Any laptop gets this at its next `winner login`. A Mac whose push is running also gets it from `winner update`, but only from the update after the one that installs this release, and that update rewrites only the push service; the reporter and update timer, which never start the agentsview daemon, wait for the next login. `docs/install.md`, "What a laptop contacts", lists everything a laptop still reaches.
- A laptop's database role could create a function, view, trigger or policy in its own team's schema during the moment the backend re-installed its guard at start, and could alter a routine it could not otherwise alter. Both are closed: the guard is now replaced in one step, and it refuses the two ways of altering a routine it used to miss.
- An enrolled laptop's database role could take the locks the control plane takes turns on, and so stop it upgrading every team's database at start, or hang enrollment and hostname reports for a name, for as long as it stayed connected. Those locks now sit on tables a laptop's role cannot touch. A hostname claim still waiting on another enrollment of the same name after five seconds now fails with a message to try again, instead of waiting indefinitely.
- A laptop's database role could hand its team's search tables to another team's role or to everyone, which let that role read sampled text from them, could connect to the server's `postgres` and `template1` databases, and could pass data to a cooperating laptop of another team through a large object. A laptop's role can no longer grant its tables or set default privileges, and grants that already exist are revoked at every start and logged (the first twenty in a team's schema by name, the rest as a count), or reported where a start cannot remove them; every start also closes those two databases to laptops and takes large objects away from them, and a deploy fails if any of this is still open. A laptop can still move its group's tables to the group every team's laptops share by reassigning their ownership: a deploy now fails on that too, and it is not yet prevented.
- On a deployment running the production overlay, a client with a private source address (a VPC peer, a VPN user, a host on the same LAN) could reach Postgres's password prompt as any role, the superuser included, with or without TLS. Roles other than a laptop's are now admitted only from the stack's own network, on a subnet the overlay pins. Every other address, private or public, gets a laptop's role over TLS and nothing else, refused before a password is read. A deploy loads this release's access rules into the database and fails if they do not take (see Requires of you).
- A page on a neighbouring address of the deployment's own site (another subdomain, say), open in a signed-in manager's browser, could star, rename or trash that team's sessions in the dashboard. The dashboard now refuses any change that does not come from its own pages.
- A session agentsview flagged as holding a secret is not sent to the model by Prompt Coaching, Skills or Benchmarks while it is flagged, whatever the finding's confidence, and neither are results already made from it. A skill candidate or draft with an example from such a session leaves the Skills page and cannot be generated; an approved skill stays. A benchmark run from such a session is not shown or counted toward the proven saving. A session whose findings are removed from agentsview's table is sent again; `docs/model-and-telemetry.md` says what the rule does not do.
- An address nobody verified and the control plane did not create can no longer sign in to a team, in the browser or with `winner login`. The browser sign-in is refused; `winner login` prints the reason, to verify the address. The

tokens never carry a role or an organisation. People a manager adds through Admin sign in as before.

- A failed model call could leave part of a member's prompt, or of the model's answer about it, in the jobs log and in the error stored for the run. A failure is now recorded by its HTTP status or kind, the model, the shape asked for and the number of tries, and a failed write to the control database no longer prints the values it was writing. The provider's own explanation is no longer in the log: to read it, send a minimal request with the same model and `response_format` using `curl`.
- Text a member typed, or that the model wrote from it, could pose as instructions in a later model call inside the same team. A prompt could close the marker around it or open a new one; a workflow's name and summary went to Skills' Generate outside any marker; an agent or model name a laptop reported could forge a line of the coaching judge's request; and one judge call mixed several members' prompts. Now such text cannot hold a marker as the requests write it, nor one spelled with the lookalike characters Winner AI lists (the list is not every glyph a model might read as a bracket); the name, summary and scope are sent as data; those names and the team's own Routes and model families are reduced to plain names; a workflow or pattern name must be one short line or it is not stored; and one judge call holds the prompts attributed to one member. Whether a model obeys text inside a marker was not tested.
- On a deployment running the production overlay, a request that carried its own `X-Forwarded-Uri` header naming a public path, such as `/health`, was let past sign-in to the path it really asked for. That reached only pages that check no identity of their own: the build version (`/internal/version`) and the API schema and its viewers (`/openapi.json`, `/docs`, `/redoc`). Manager pages, the admin API and the dashboard still answered 401, and no session or customer data was readable. The edge now removes that header and writes `X-Forwarded-Proto` and `X-Forwarded-Host` itself on every request, so the sign-in proxy believes no forwarded header a client or a CDN's client sent. The localhost stack was never affected, and `OAUTH2_PROXY_REVERSE_PROXY` is no longer a setting: only the production overlay switches that mode on.
- A skill draft nobody approved could be downloaded. Download and the install text now exist only for an approved version; a draft above an approved version no longer replaces the file the page hands out.
- Benchmarks could call a run the same answer on the judge's score alone when the original reply had no decision (ALLOW or BLOCK and the like) to check the run against. Such a run is still scored and is now always a different answer, and its card says there was no structured check to confirm the answers match. Runs already stored that way are corrected by the upgrade, with no model call.
- The installer and `winner update` install `winner` only when the release's `SHA256SUMS` carries a valid signature by the Winner AI release key, so a control plane or mirror that swaps in a binary we did not release, with a checksum to match, is refused. The signature does not say which release: a control plane can still serve an older signed one. The installer checks it with the `ssh-keygen` macOS already ships; `agentsview` keeps upstream's own checksum, and a digest you supply with `FLEETVIEW_AGENT_SHA256` skips the signature. A laptop's first update to this release is still checked by checksum alone.

5.9.5 Requires of you

- **Operators:** `OAUTH2_PROXY_REVERSE_PROXY` in `.env` is no longer read. If you had set it to `true` on a stack without the production overlay, run the production overlay instead: it switches the mode on behind an edge that cleans the headers the mode believes. A deployment already on the overlay needs nothing but this release.

- **Operators:** redeploy the post-login Action from this release to each tenant (`docs/onboarding.md` , "The post-login Action" has the commands and one to compare the deployed copy). Before you do, mark verified only the accounts you created yourself and know, such as the localhost placeholders. A demo user an earlier seed run left unverified is deleted and seeded again, and marked only if you kept your own record of the Auth0 user id that run created. Do not mark any other unverified account, and never one that someone signed up with: it either proves its address with the verification email or is deleted. The query on that page lists who the Action will refuse, not who to mark.
- **Operators:** on a hosted model endpoint this update spends model calls once: the coaching window is judged again, and every benchmark task is replayed and judged again under the daily limit (see Changed).
- **Operators:** a Prompt Coaching judge call now holds one member's prompts, ten at most, or one session's when the session is not yet attributed to anyone, where it used to hold any ten prompts, so a team with several members can take more calls. A sweep stops at twice the calls its daily budget would take in full batches (60 at the default budget of 300) and the next sweep continues; over a day the budget still bounds the prompts judged. This change adds no benchmark re-judging.
- **Operators:** a deploy now fails when a team schema holds code stock agentsview does not create, most likely planted by a laptop before 0.6.0. `docs/troubleshooting.md` , "Code in an org schema that agentsview did not create", says how to find and clear it.
- **Operators:** the `postgres` and `template1` databases now admit only a superuser, or a role you have granted `CONNECT` there by name (`GRANT CONNECT ON DATABASE postgres TO <role>`). A grant to everyone does not last: the next start revokes it. The application database is unchanged.
- **Operators:** `PUBLIC_BASE_URL` must be the bare address a browser shows, such as `https://devtool.example.com` : lower case, no trailing slash, no path and no default port. `make up` or a deploy now refuses any other spelling, because the dashboard would refuse every star and rename under it.
- **Operators:** if you seed a demo org with `fleetview-seed-demo` , the first run after upgrading is as slow as a first seed (about 3.5 hours on a CPU-only host), and Coaching re-judges the whole demo org over the following sweeps, up to its daily budget (see Changed).
- **Operators:** the production overlay now pins the compose subnet, and the first deploy of this release takes the whole stack down once to move its network there. The host's own networks, VPC and routes must not use that subnet. The deploy fails with the stack left up while a `fleetview-seed-demo` run holds the network; the timer's next run is the retry. A `fleetview-postgres` jail set up before this release still ignores every private range: narrow its `ignoreip` to the pinned subnet once that deploy has run (`docs/network.md` , "The Postgres jail").
- **Operators:** a dump taken before this release kept the name `fleetview-<stamp>.sql.gz` even when its restore test failed. Before restoring one, confirm its stamp's `restore test PASS` line in `journalctl -t fleetview-backup` ; from this release a failed dump is named `.failed` (see Changed).
- **Operators:** move any script or note of your own that starts `python -m fleetview_backend.jobs` or `python -m fleetview_backend.provision` to `fleetview-jobs` or `fleetview-provision` (see Changed). Both old forms still work on this release and stop once the backend ships compiled.
- **Operators:** this release migrates the control database at its first backend start, with nothing to run by hand. A rollback to 0.9.0 after that is not code-only: take a `sudo fleetview-backup` before you deploy and expect to restore it if you roll back.

- **Operators:** a mirror you point `FLEETVIEW_AGENT_URL` at must publish the release's `SHA256SUMS.sig` beside `SHA256SUMS`, unchanged, or the installer refuses it. A release built by hand needs the release key holder to sign `SHA256SUMS` before it is published. Rolling the control plane back past this release, or serving a bundle with no `SHA256SUMS.sig`, makes every `winner update` on a laptop already on this release fail on `winner` until you roll forward; `agentsview` still updates.
- **Members:** nothing to reinstall or sign in again. On macOS, `winner update` brings the `winner push` fixes above; on Linux, this release's `winner-linux-amd64` or `winner-linux-arm64`, placed by hand, brings them and the `winner login` fix. agentsview's usage ping and update check go off at the next `winner login`; a Mac whose push is running gets that from `winner update` too, from the update after the one that installs this release, and one whose push is stopped only from the login.

5.10 0.9.0 - 2026-09-25

5.10.1 Added

- Every enrolled laptop now reports a push heartbeat, whether or not it has run `agentsview embeddings build`: the jobs sweep watches the marker row each push rewrites. A laptop seen pushing since the upgrade whose push service then dies turns Stale on the roster within a day instead of reading like a quiet week.

5.10.2 Changed

- The roster and Home show "Liveness unknown" where they showed "Quiet", and on Home a member none of whose devices has ever pushed reads "Never pushed" instead of "Enrolled".
- On a wide monitor Settings now widens with every other page instead of stopping at 1180px, and the "Why" on a Prompt Coaching finding keeps a comfortable line length. Nothing changes below 1600px.
- The wide-screen step shipped in 0.5.1 without a note here: from 1600px the page column and the type grow, and from 2400px the column keeps widening on its own, so tables and grids take the room that used to sit in the margins. Prose keeps its line length, so a wider screen buys more columns and wider tables and never a longer line. The step follows the reader's own browser font size and zoom.

5.10.3 Fixed

- A failed role statement no longer writes a database password into the backend's log or Postgres's own: the control plane now sends Postgres the password's SCRAM verifier, never the password. Existing roles keep working and pick up a verifier the next time their password is set.
- The jobs log no longer calls a pass skipped when it did its model-free work without a model (the benchmark queue drain, the coaching owner back-fill, the Skills stale-candidate cleanup): it now says the pass ran without a model. The per-sweep line now reads `N passes had no model to call` instead of `N passes skipped for no model`; update any log search that matched the old text. A run's recorded settings now name only the settings that run read.
- A benchmark run requested while the organisation's stored team key can no longer be unsealed is refused on the click with the reason, instead of being queued and then dropped by the next sweep without a word.

- `winner login` now points at `winner update` against a control plane that publishes no `winner` version too (every deployment before 0.5.0, and any untagged build). That run converges agentsview and leaves the `winner` binary alone; the message used to send the developer to a manual agentsview install instead.
- Benchmarks dates each money figure by the price rows it actually read, as a range when they differ, instead of by the newest row in the price table. Its notes now also say an "up to" estimate appears on patterns marked "Cannot be tested", so the estimate is never read as a sign the pattern is testable.
- A benchmark replay that fails no longer uses up the day's replay limit: only replays that answered or were cut at the token cap count against it, so one provider error no longer stops benchmarking until the next day.
- Benchmarks' pattern costs and "Could be saved", and Prompt Coaching's cheaper-Route estimate, no longer count one Claude reply several times. A long reply is stored in several parts that each repeat its input, so these figures read several times too high for Claude sessions; each reply is now priced once, so Home's "Could be saved" tile and its yearly figure go down with them. The next sweep measures every recurring pattern again, so after upgrading each pattern is renamed once and its benchmark runs start over. Prompt Coaching keeps the estimate it already showed for a session it judged before the upgrade; only sessions judged after it get the corrected figure.
- Every laptop on a team replicates vectors into a new embedding generation, not only the one that started it, and the dashboard's semantic search can read them. Revoking that laptop no longer deletes the team's vectors for the generation. A backend restart repairs laptops enrolled earlier; nothing needs re-enrolling.
- The benchmark sweep's log no longer calls a pair "already replayed" when it is only waiting out the 24 hours after a failed attempt; it now says it is waiting.
- A skill Winner AI writes from a candidate on Skills is written from the candidate's different prompts. When one teammate repeated a prompt word for word, the model could be shown that one prompt ten times and none of the variations; each prompt now reaches it once, nearest the cluster's centre first. A candidate whose cluster forms again takes the new order on the next sweep, and its Evidence list then opens on different prompts too.
- `winner update` no longer stops both binaries when one sits in a directory this user cannot write: it updates the other, then exits non-zero naming the one it skipped and why. A failed `agentsview` download or install likewise no longer stops `winner` from updating. `--force` now also reinstalls an `agentsview` whose version cannot be read, and an `agentsview` release archive that opens with a pax global header no longer fails the update. `winner login` against a different control plane than the one recorded on the laptop says it replaced the record and prints the command that goes back.
- Home, Benchmarks and Prompt Coaching no longer take 10 to 20 seconds to open, and their spend figures no longer read "unavailable", for a team with a large month of usage. Winner AI now reads spend from the dashboard every ten minutes in the background and the pages show the stored figure; Home says when it was read ("last 30 days, as of 14:05"). After an upgrade the figures appear within a few minutes of the control plane starting.
- Sessions a laptop pushes late (its history on first enrollment, or a backlog after days offline) are now attributed to their person. Upgrading re-scans each team's sessions once to recover the ones missed before.

5.10.4 Requires of you

- **Operators upgrading a running deployment:** the upgrade clears every team's attribution cursor once so the next ticks re-scan. If the old backend was still attributing while the migration ran, it can write its old cursor back and that

team's re-scan never happens. If a team's older sessions still show no owner after the upgrade, run this once against the control database; sessions that already have an owner are skipped:

```
UPDATE control.orgs SET attribution_watermark = NULL,
  attribution_watermark_id = '', attribution_scan = NULL,
  attribution_scan_id = NULL;
```

- **Members:** nothing to reinstall or sign in again. On macOS, `winner update` brings the `winner update` and `winner login` fixes above; it installs macOS builds only.

5.11 0.8.0 - 2026-09-25

5.11.1 Added

- Home's "Could be saved" tile now says what its monthly figure comes to over a year ("about $847K a year"), for teams that budget annually.

5.11.2 Changed

- `fleetview-seed-demo` seeds a 30-person team spending about $300k a month in place of the five-person one. The five keep their names, so a re-seed refreshes their sessions rather than adding to them. A fresh seed takes most of an hour, most of it spent building embeddings.

5.11.3 Fixed

- The nightly backup's restore test no longer fails at random with "the scratch Postgres never became ready". It waited on the scratch server's first-start init phase; it now waits for the real server.

5.11.4 Requires of you

- **Operators who seed a demo org:** the 30 demo accounts must exist in the Auth0 tenant as members of that org before `fleetview-seed-demo` runs; it creates no account. Run `make seed-demo` once against the same tenant to create them.
- **Members:** nothing.

5.12 0.7.1 - 2026-09-24

5.12.1 Fixed

- `fleetview-seed-demo` now embeds every persona on a host whose Ollama runs on CPU. It had timed out and pushed without vectors, leaving Skills on stale prompts.

5.12.2 Requires of you

- **Members:** nothing.

5.13 0.7.0 - 2026-09-24

5.13.1 Added

- Benchmarks and Home's "Could be saved" now include context re-read above a per-turn budget: what long sessions pay to re-read their conversation past 150,000 tokens a turn, labelled as an estimate with its pricing date and linked to Prompt Coaching's Context findings. The budget is a team setting on the Settings page.
- Operators can seed or refresh a deployment's demo organisation from its host with `fleetview-seed-demo --org <id>`: five demo members with a month of sessions, pushed the way a laptop pushes, after a verified backup. It needs the demo accounts to exist in the deployment's Auth0 tenant already.

5.13.2 Requires of you

- **Members:** nothing.

5.14 0.6.0 - 2026-09-24

5.14.1 Added

- Benchmarks' "Could be saved" block now lists its figure by saving lever, starting with recurring patterns.
- Benchmarks and Home's "Could be saved" now include subagents that could run on a cheaper model, labelled as an estimate with its pricing date.

5.14.2 Fixed

- Vectors replicate again after a migration pass. Before, the pass handed the embedding generation's chunk table to the org's owner, and every laptop's push then skipped vectors, so Skills and semantic search stopped seeing new prompts.

5.14.3 Security

- A laptop's Postgres role can no longer switch to its team's shared device role to get around the rule that stops it creating functions, triggers and other code objects. The same rule now also refuses aggregates, views and row-level-security policies, so a laptop cannot put code in the path of the control plane's cross-team reads.

5.14.4 Requires of you

- **Managers whose Skills stopped growing:** click **Migrate** on Admin once after upgrading. That hands the chunk table back to the laptops, and their next push replicates the vectors they missed.
- **Members:** nothing. Once a manager has clicked **Migrate**, each laptop's next push replicates the vectors it missed.

5.15 0.5.1 - 2026-09-22

5.15.1 Added

- The edge serves Postgres under **its own hostname** when one is set (`PUBLIC_PG_HOST`), and obtains and installs a certificate for it, so a deployment whose web hostname sits behind a CDN can still let laptops reach Postgres by a name the CDN does not front. With it unset the edge serves one name, as before.
- Behind a CDN the edge can record the visitor's address rather than the CDN's: when the peer is in `EDGE_TRUSTED_PROXIES` , the edge takes the client address from that peer's `Cf-Connecting-Ip` header; from any other peer the header is ignored and the peer address stands. Access logs carry both. What the edge forwards onward is the one address it resolved, never the inbound `X-Forwarded-For` list.
- `winner login` on Linux installs Winner AI's own background push, a systemd user unit, the way it already does on macOS, and `winner push status|logs|stop|start|uninstall` drive it. A Linux laptop enrolled before this pushed nothing until someone ran agentsview's installer by hand; the next `winner login` replaces that service with ours. A machine reached only over SSH needs `loginctl enable-linger` for the push to start at boot.
- Every Benchmarks row without a tested saving shows an "up to" estimate: its own tokens priced again at the cheapest candidate model the team's settings map it to. The "Could be saved" tile shows it too, never added to a tested saving.

5.15.2 Fixed

- On a deployment that does not lend its model, a queued benchmark run is now answered with the reason it could not be replayed instead of waiting forever, Skills stops listing workflows the team no longer repeats and keeps each kept candidate's counts current, and coaching rows are still credited to the teammate who ran the session.
- A `winner push status` whose log ended in a progress redraw no longer overwrites the line printed above it: the one line it echoes has control characters stripped, while `winner push logs` still prints the file as it is.
- When `winner login` finds a laptop's agentsview out of step with the control plane's, it now points at `winner update` , which moves the laptop to exactly the version the control plane runs. It used to suggest the upstream installer, which takes the laptop to the latest release instead and leaves every push failing.
- The hourly hostname report no longer asks Auth0 to resolve the organisation: it reads it from the reporting member's own record, so a fleet of laptops no longer spends a Management API call each, every hour, against a rate-limited quota.
- A benchmark that failed on a cheaper model, for example during a provider outage, is no longer blocked for good: the next day's sweep tries it again, a manager can retry it at once from Benchmarks, and the failed attempt stays on record.

- A skill candidate on Skills reports the members it actually has. Its repeated prompts, distinct sessions, contributor count, last-seen date and Evidence used to stay at the values of the sweep that first stored it when a teammate repeated a prompt word for word; every sweep now rewrites them from what it found. A dismissed candidate keeps the numbers it was dismissed on.
- `winner push stop` used to report success for every failed bootout; it now exits non-zero and shows launchctl's own message when the stop genuinely failed, rather than only when the push was already stopped. A stop systemd refused on Linux now says the push may still be running too.
- A failed background-push install now says when it already removed agentsview's own push service, and how to restore it, instead of leaving the laptop with no pusher and no explanation.
- A `PATH` that cleaned away to nothing no longer overrides launchd's own default with emptiness, which could leave the background push unable to resolve any binary at all.
- Enrolling a laptop under a name another of your own laptops already answers to is refused, as it already was for another member's laptop: two laptops on one name leave every session under it unattributed. Rename the machine and run `winner login` again, or revoke the older device first.
- Home's per-person session count no longer counts a name two of a member's laptops claim twice, so it agrees with that member's roster row.
- `scripts/check-env.sh` refuses a `PUBLIC_PG_HOST` that is an IP address — in either notation, including an IPv4 address written as IPv6 — with the reason: no public CA will certify one, so `verify-full` could never accept it.

5.15.3 Requires of you

- **Operators behind a CDN:** set `PUBLIC_PG_HOST` to a DNS-only name that resolves straight to the host and `EDGE_TRUSTED_PROXIES` to the CDN's ranges; leave both unset when nothing sits in front. Port 80 must stay reachable from the internet, not only through the CDN: the second certificate is challenged at the host directly (`docs/network.md` , "Behind a CDN").
- **Members:** on macOS, nothing beyond the hostname case below; `winner update` brings the push fixes above. No Linux build is published, so a Linux laptop builds `winner` from this release and runs `winner login` once to get Winner AI's background push.
- **Members whose deployment changed its Postgres hostname:** the DSN in `config.toml` still names the old host. Run `winner login` once and the laptop pushes to the new name; until then pushes fail with a TLS name mismatch.

5.16 0.5.0 - 2026-09-18

5.16.1 Added

- `winner update` converges a laptop's `winner` and `agentsview` to the versions its control plane runs, in either direction, after checking each download against its published checksums. `--check` reports what would change without installing anything.

- `winner push status`, `logs`, `stop`, `start` and `uninstall` manage the background push. On macOS, `winner login` now installs Winner AI's own launch agent, which runs the stock `agentsview pg push --watch`, in place of agentsview's service.
- A laptop keeps its sessions when its hostname changes. On macOS, `winner login` installs an hourly reporter that tells the control plane every name the laptop has pushed under, and the roster, Home, spend, Coaching and Skills count every one of them. Re-enrolling a renamed laptop replaces its old device row instead of adding a second. `--no-service` skips the reporter as well as the push.
- Settings lists every value the organisation runs with, what it does and which layer set it (the release, the operator or the team). A manager changes the team's values in place, Reset returns one to the operator or release value, and a manager can switch Coaching, Skills or Benchmarks off for the team within what the deployment enables.
- A manager can give their organisation its own model endpoint, model and API key on Settings. The key is stored encrypted with the new `FLEETVIEW_SETTINGS_KEYS` and never shown again; `FLEETVIEW_LLM_FALLBACK` decides whether an organisation without a key uses the host's model or none, and with none, Coaching, Skills and Benchmarks say "No model configured". Both are optional: without them nothing changes.
- The Prompt Coaching page lets a manager omit a finding or mark it as wrong, labelled a demo simulation; nothing is recorded and a reload resets it.
- The roster links each person to the dashboard's Usage page filtered to their laptops, labelled with their name; a person with no device shows a named empty state instead.

5.16.2 Changed

- The Benchmarks controls (daily runs and quality threshold) move from the Benchmarks page to Settings, and `POST /benchmarks/controls` is gone. Saved values carry over; daily runs are capped at 500.
- `winner login` now records the control plane it used, so a later bare `winner login` or `winner update` on that laptop needs no flag.
- The Prompt Coaching judge reads the team's frontier model families and Routes to try from Settings, and only recommends a route on the team's Routes to try. Each new verdict records the lists it was judged with; stored verdicts are not judged again.
- Every headline figure reads the same on every page: Home's tiles, Admin's metrics and the Skills, Prompt Coaching and Benchmarks bands now share one type style, so the Benchmarks band's numbers are no longer set in the mono face. No figure's value changes.
- Prompt Coaching and Skills refuse a `FLEETVIEW_LLM_ENDPOINT` that is plain http to anything but a compose-internal or local host, as Benchmarks already did. A host pointing it at plain http on another machine must move that endpoint to https before upgrading, or those passes fail every sweep.
- The public `/api/v1/agent/config` also returns `winner_version`, the release the control plane runs, which `winner update` converges to.
- "How to read these numbers" starts closed on Benchmarks and Coaching.

5.16.3 Requires of you

- **Members:** a laptop installed before this release has no `winner update`, keeps agentsview's own push service and has no hostname reporter. Re-run the installer, then `winner login`, once; after that, run `winner update` to keep it current.
- **Operators:** the hostname reporter authenticates with a refresh token, so the Auth0 API must allow offline access (and the native app the Refresh Token grant). Without it, `winner login` says the reporter was not installed, and a renamed laptop's sessions stay unattributed until its member runs `winner login` again. A host overlay that overrides `OAUTH2_PROXY_SKIP_AUTH_ROUTES` must add `^/api/v1/devices/report-names$`. Check that `FLEETVIEW_LLM_ENDPOINT` is https or compose-internal before upgrading. The control plane migrates the control schema at startup; read the backend log after upgrading.

5.17 0.4.0 - 2026-09-16

5.17.1 Added

- The control plane serves the installer and the `winner` binary itself, so a developer installs with one `curl` from their own deployment, with no GitHub access and a checksum check.
- The installer records the control plane URL on the laptop, so `winner login` needs no flag there.
- The Dashboard carries a "Back to Winner AI" control that returns to Home.
- Benchmarks records every recurring prompt, with the reason one cannot be replayed kept beside it, and names the patterns people type.
- The Skills page carries a labelled demo panel that simulates delivering a suggested skill to the team. Nothing is installed and nothing is recorded.
- A documentation site built from this repository, with a local preview (`make docs`, `make docs-serve`) and a build that runs on every pull request.

5.17.2 Changed

- `winner login` no longer falls back to a built-in control plane address: it takes `--control-url`, `FLEETVIEW_CONTROL_URL` or the installer's record, and exits non-zero naming all three when none is set. It refuses a control plane URL that is not https, unless it is localhost.
- `winner login` refuses a laptop whose agentsview is newer than the server's, and the server re-migrates each organisation's schema after an agentsview bump, so a laptop that ran ahead no longer fails every push silently.
- The Benchmarks page is a ranked table of prompt patterns, by cost, runs or time, and Run benchmark queues the run for the next sweep instead of holding the page.
- The Prompt Coaching band explains each figure in plain English and marks which are estimates.
- Values a team may tune are stored per organisation; the Benchmarks controls move there and behave as before.
- With Management API credentials set, the control plane refuses to start when the Auth0 tenant lacks the `manager` or `member` role, naming the missing one.

- Session attribution reads a bounded batch per tick, and `FLEETVIEW_ATTRIBUTION_INTERVAL_SECONDS` tunes it (0 turns it off).
- The control schema is migrated automatically at startup.
- The jobs service receives only the settings it reads, and every read of an organisation's data runs as the read-only `control_miner` role.

5.17.3 Fixed

- Re-running `winner login` on the same laptop replaces its old device instead of counting the machine twice.
- `winner login` no longer fails at random with "token is not yet valid (iat)".
- On Windows and Linux, `winner login` no longer points at a push command that cannot run there.
- Benchmark savings price each token type at its own rate, count one prompt at a time, show one row per prompt, and no longer clip the table.
- A coaching Finding with no rewrite says so instead of showing "None".
- The API reports the real version and the name Winner AI.
- The installer's `FLEETVIEW_AGENT_URL` download works without a GitHub token.
- The restore runbook stops every service holding the database and has the operator check the replay's errors and row counts before starting the stack.

5.17.4 Requires of you

- **Members:** a laptop installed before this release has no recorded control plane, and `winner login` never records one itself. Each login needs `--control-url` or `FLEETVIEW_CONTROL_URL` until the installer is run again, which records it. A laptop whose agentsview is newer than the server's must install the server's version first.
- **Operators:** a control plane served over plain http must move to https, because laptops refuse it anywhere but localhost. The Auth0 tenant must hold roles named `manager` and `member` before the control plane will start with Management API credentials set. The control plane migrates the organisation schemas and the control schema at startup. A failed migration is logged as `startup provisioning failed; serving anyway` and tried again at the next start, so read the backend log after upgrading. Benchmark figures are recomputed on the next sweep, so they will move.