Looking after a runner

From the SourceLace manual: How your IT department keeps a SourceLace Runner healthy once it is installed: watching it, how long each version is supported, upgrading to a new version and going back, license end dates, backing up and restoring, moving it to another machine, changing its token, master

SourceLace version 0.1.0 · Made November 22, 2026
The latest version is at https://sourcelace.ai/docs

Looking after a runner

How your IT department keeps a SourceLace Runner healthy once it is installed: watching it, how long each version is supported, upgrading to a new version and going back, license end dates, backing up and restoring, moving it to another machine, changing its token, master key or AI key, and removing it.

The installation is in Installing a runner, and problems in Runner troubleshooting. The examples use the image `123456789012.dkr.ecr.us-west-2.amazonaws.com/sourcelace-runners/runner-corvanta`, the Docker Compose folder `/opt/sourcelace-runner` and the Helm release `corvanta-runner` in the namespace `sourcelace-runner`, as in the installation.

A routine to follow

How often	What	Who
Continuously	Your monitoring calls `https://<runner page>/health` and alerts on anything but HTTP 200	Container administrator
Continuously	SourceLace shows admins a notice when a runner with agents stays quiet (10 minutes unless changed on the **Limits** page)	SourceLace admin
Weekly	A look at the **Runners** page: status, version, license, free disk and memory, last error	SourceLace admin
Daily (automatic)	Database backups	Database administrator
Every few months	A test restore of the database with the master key	Database administrator
With each new version	An upgrade, following the plan agreed with your SourceLace contact	Container administrator
Monthly	Operating system and Docker or Kubernetes updates on the host	Container administrator

Watching the runner

- **Health address**: `https://<runner page>/health` (or `http://127.0.0.1:8000/health` on the machine) answers HTTP 200 with `{"status":"ok", "version":"...", "database":"ok", ...}`, or HTTP 503 with `"status":"unavailable"` and `"database":"unreachable"` when the runner cannot reach its database. It needs no sign-in and shows no data.
- **The Runners page** in SourceLace: status, last contact, version, license and health for each runner (Is the runner healthy?).
- **The runner's Settings page**: the connection to SourceLace, the license, runs in progress, reports waiting to be sent and errors of the last 24 hours by type.
- **The log**: `docker compose logs -f runner`, or `kubectl -n sourcelace-runner logs deployment/corvanta-runner-sourcelace -f`. Alert on lines starting with `ERROR:sourcelace:` or `WARNING:sourcelace:Runner:`. The log holds no tokens, keys, passwords or data from your systems.

Check it worked: your monitoring shows the health address green, and the **Runners** page shows **Healthy**.

How long each version is supported

What it means: each runner version is supported until **6 months after the next version is released**. The newest version has no end date. So you always have at least 6 months to move from one version to the next, and SourceLace does not have to keep old versions working for ever.

What the Runners page shows	What it means	What to do
Current	It runs the newest version.	Nothing.
Update available · until a date	A newer version exists; this one is supported until that date.	Plan the upgrade with your SourceLace contact.
Upgrade due · until a date	Less than a month of support is left. Admins also see a notice at the top of the app.	Upgrade before that date (Upgrading).
Unsupported	Its support ended. The runner takes no new work. Runs in progress finish, changes waiting for approval can still be decided, and nothing is deleted.	Upgrade it: it takes work again at its next contact with SourceLace.

Critical security fixes: when a version has a serious security problem, SourceLace may end its support earlier than 6 months after the next version, once a fixed version is out. The Runners page and the notice then say why (such as the security advisory) and the new date. Upgrade before it.

Check it: the **Runners** page shows each runner's support under its version, and **Get the runner** shows it for every version. The runner's own **Settings** page shows it too, under **Version support**.

Upgrading

What this does and why: a new version brings fixes, including security fixes, and new features to the runs on your runner. Nothing updates by itself: your SourceLace contact tells you about new versions and plans upgrades with you (a scheduled upgrade plan is part of Premium support). An out-of-date runner keeps working, but may refuse runs that use newer rules and misses fixes. The upgrade takes a runner with one instance offline for about a minute; nothing is lost. A runner with two or more instances (Failover) is upgraded one instance at a time and keeps taking work throughout.

1. Get the new version, such as `0.5.0`, from **Runners**, **Get the runner** (Get the runner): sign in with a fresh registry login (Registry logins) and pull it, or download the image file and load it with `docker load -i` (then skip `docker compose pull runner` below). For Kubernetes, copy it into your own registry. Your SourceLace contact also tells you about new versions.
2. **Back up** the database (Backups).
3. Note the version you run now, for going back: the **Runners** page shows it.
4. Upgrade.

   Docker Compose:

```
cd /opt/sourcelace-runner
export SOURCELACE_IMAGE=123456789012.dkr.ecr.us-west-2.amazonaws.com/sourcelace-runners/runner-c
  orvanta:0.5.0
docker compose pull runner
docker compose up -d
```

   `pull` downloads the new version while the old one still runs; `up -d` replaces the runner's container with the new version. Update `.env` too, if you keep the image there.

Kubernetes: copy the new version into your own registry (Step 5, item 2), change `image.tag` to `"0.5.0"` in `corvanta-runner.yaml`, then:

```
helm upgrade corvanta-runner <path to the chart> -n sourcelace-runner -f corvanta-runner.yaml
```

With `replicaCount` 2 or more, Kubernetes replaces the pods one at a time by itself. With two Docker Compose machines, upgrade one, check it (below), then the other. For a while the instances run different versions; that is expected.

5. The new version starts and updates its database tables by itself. Runs that were going are handled as after any restart: nothing is done twice (If the runner restarts or loses its connection).

Check it worked:

- `curl -sS http://127.0.0.1:8000/health` shows `"version":"0.5.0"` and `"database":"ok"`.
- The log shows `Database schema is at version ...` and no `ERROR`.
- The **Runners** page shows the new version and **Healthy**, not **Out of date**.

Going back a version

What this does and why: if a new version causes a problem, you can start the version you ran before. The runner's database changes only by adding, so an older version still works with it; the older version writes a warning and leaves the database as it is.

1. Start the previous version, as in Upgrading step 4, with the old version number.
2. The log shows, as a warning: `The database schema is at version ..., newer than this release's ... (a rollback?). Leaving it as it is.` This is expected.
3. Tell your SourceLace contact what went wrong with the new version.

Check it worked: `/health` shows the previous version; the **Runners** page shows **Healthy** or **Out of date** (expected while you run an older version). Runs that use rules the older version cannot read fail with a message saying so, until you upgrade again.

You do not need to restore the database to go back a version. Restore it only if the database itself was damaged (Restoring).

Backups

What this does and why: the runner's database holds run results, files, approvals waiting, people's sign-ins and secrets, all encrypted with the master key. A backup is only useful with that key, and the key alone holds no data, so keep them apart.

1. **The database**: use your PostgreSQL service's automatic backups (for a managed service, its snapshots and point-in-time recovery), or a nightly dump such as:

```
pg_dump --format=custom --file=runner-$(date +%F).dump "postgresql://runner:<password>@db.corvan
  ta.internal:5432/runner"
```

`--format=custom` makes a compressed file that `pg_restore` can read. Keep the files as your backup policy says.

2. **The master key**: in your secret store, with a note of which runner it belongs to. Keep earlier keys too while backups made with them exist (Changing the master key).

3. **What needs no backup**: the folder for large results, the temporary folder, and the container itself. `runner.env` or the Kubernetes secrets can be rebuilt from your secret store.

Check it worked: a recent backup exists, and a test restore into a spare database worked.

Restoring

What this does and why: brings the runner back after its database was lost or damaged. Results made after the backup are not on the restored runner; SourceLace still has those runs' status and steps.

1. Stop the runner: `docker compose stop runner`, or `kubectl -n sourcelace-runner scale deployment/corvanta-runner-sourcelace --replicas=0`.
2. Restore the database, for example into an empty database owned by the runner's user:

```
pg_restore --clean --if-exists --no-owner --role=runner \
  --dbname="postgresql://runner:<password>@db.corvanta.internal:5432/runner" runner-2026-10-19.d
  ump
```

   For a managed service, restore its snapshot or point in time instead, and point `DATABASE_URL` at the restored database.
3. Make sure the runner has the **same master key** that was in use when the backup was made, and its token.
4. Start the runner: `docker compose start runner`, or scale the deployment back to 1.

Check it worked: `/health` shows `"database":"ok"`; the runner's **Runs** page shows the runs from before the backup; the **Runners** page shows **Healthy**. A message `A tenant key could not be unwrapped with SOURCELACE_ENCRYPTION_KEY.` means the master key is not the one the backup was made with.

Moving a runner to another machine

What this does and why: the runner is defined by its token, its master key and its database. Move those and the runner continues where it was, with its results and people's sign-ins.

1. On the new machine, install Docker and the Compose plugin, sign in with a fresh registry login and pull the same version (Step 3) or load the image file, and copy `docker-compose.yml` and `runner.env` (through your secret store, not email). Prepare the folder for large results (4.4).
2. Make sure the new machine has the same outbound network access (Network) and can reach the database.
3. **Stop the old runner** (`docker compose down` on the old machine). Never run two copies with the same token.
4. If the database moves too, back it up and restore it on the new server (Restoring), and update `DATABASE_URL`.
5. Start the runner on the new machine (`docker compose up -d`) with the same version.
6. Point your TLS proxy, or the DNS name of the runner's page, at the new machine.

If the runner page's **address changes**, also change `SOURCELACE_PUBLIC_URL`, your DNS and certificate, and the callback address of each OAuth app registered for the runner at its vendor. The "Open result" links of earlier runs keep pointing at the old address.

Check it worked: the **Runners** page shows **Healthy** with the new **Results page** address if it changed, and the runner's **Runs** page shows the earlier runs.

Changing the runner token

What this does and why: a runner's token cannot be shown or changed again. If it may have leaked, or was lost, you replace the runner registration in SourceLace while keeping the runner's database and everything in it.

1. **Check your free places**: the **Runners** page shows how many runners of each label your contract includes. If none is free, step 3 must come before step 2, and agents on the runner cannot run between the two steps.
2. A SourceLace admin **adds a new runner** with the same label (it can have the same name) and stores its token.
3. The admin **revokes the old runner**. Its token stops working at once; runs queued for it end as failed.
4. Put the new token in `SOURCELACE_RUNNER_TOKEN` (or the Kubernetes secret) and restart: `docker compose up -d --force-recreate runner`, or `kubectl -n sourcelace-runner rollout restart deployment/corvanta-runner-sourcelace`. Keep the same database and master key.
5. In SourceLace, choose the new runner under **Where it runs** for each agent that used the old one.

Check it worked: the new runner is **Healthy** on the **Runners** page with the same results page address, the old one shows **Revoked**, and a **Run now** on one of the agents works.

Changing the master key

What this does and why: you can move to a new master key without losing data. The runner encrypts with the first key in the list and can still read what any listed key encrypted.

1. Make a new key: `docker run --rm --entrypoint sourcelace-keygen 123456789012.dkr.ecr.us-west-2.amazonaws.com/sourcelace-runners/runner-corvanta:<version>`, and store it in your secret store.
2. Set `SOURCELACE_ENCRYPTION_KEY` to the **new key first, a comma, then the old key**, such as `<new key>,<old key>`.
3. Restart the runner (as in step 4 above). Everyone signed in to the runner's page signs in again.
4. Keep the old key in the list, and in your secret store. Do not remove it without asking your SourceLace contact, since parts of the database may still be encrypted with it.

Check it worked: the runner starts, `/health` shows `"database":"ok"`, and the runner's **Runs** page opens earlier results.

Changing the AI key or a source secret

An admin signs in to the runner's page, opens **Settings**, and enters the new value under **AI key and results** or **Secret source options**, then presses **Save**. It takes effect for the next AI call. If the AI key was set in `SOURCELACE_RUNNER_AI_KEY`, change it there instead and restart, or remove it there and use the **Settings** page from then on.

Check it worked: the **Settings** page shows the AI key "ready" (or the option "saved"), and a **Run now** works.

License end dates

What it means: each runner renews its license by itself, about once a day, for up to 30 days at a time. A license never lasts past your organization's **runner license end date**: the end date of your contract, unless your contract renews by itself (then there is no end date), or another date agreed with your SourceLace contact (for example while a renewal is being signed).

- From 30 days before that date, admins see "Your runner licenses end on ..." on the **Runners** page and in a notice at the top of the app.
- After that day, the runners take no new work. Runs in progress finish, changes waiting for approval can still be decided, and nothing is deleted.
- When the contract is extended (or a later date is agreed), each runner gets a new license at its next contact with SourceLace, within about 15 seconds, and takes work again. Nobody installs anything.

Check it: the **Runners** page shows, under each runner's status, until when it is licensed; the runner's **Settings** page shows the same.

Registry logins

A registry login from **Get the runner** works for one hour, so get a fresh one right before you download a new version (and another if it expires before you are done), and sign in with it as in Step 3. Running containers and images already on your machines or in your own registry never need it again. For Kubernetes, copy each new version into your own registry as in Step 5, item 2, before you change `image.tag`.

SourceLace keeps your organization's images while each version is supported and for about a month after. If runners are switched off for your organization or your runner licenses end, its images are deleted from SourceLace's registry 30 days later; what you already downloaded keeps working.

Revoking and removing a runner

What this does and why: revoking cuts the runner off from SourceLace at once; removing it then clears it from your network.

1. A SourceLace admin presses **Revoke** on the **Runners** page and confirms. The token stops working at once: the runner gets no more work, nobody can sign in to its page, and runs queued for it end as failed. A run that was already going on the runner stops at the runner's next contact with SourceLace (usually within about 15 seconds, at most about a minute), is marked as failed and writes nothing more; changes waiting for approval on it can then only be rejected. Stop the containers (next step) to end everything.
2. Stop and remove the runner: `docker compose down` in its folder, or `helm uninstall corvanta-runner -n sourcelace-runner`.
3. Its database still holds its results, files and people's sign-ins, encrypted. Export anything your policy requires you to keep (people can open results only while the runner runs and is not revoked), then delete the database and its backups, the folder for large results and, last, the

master key.

4. Remove the firewall rules, DNS name, certificate and TLS proxy configuration, and revoke the OAuth apps registered for the runner at their vendors.

Check it worked: the **Runners** page shows the runner **Revoked**, no agent names it under **Where it runs**, and the runner's address no longer answers.