Runner requirements and planning

From the SourceLace manual: Everything your IT department needs to plan a SourceLace Runner and get it approved before installing: where it can run, how big it must be, the database, running more than one instance so a failure needs nobody, every network connection, the accounts and secrets, the peo

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

Runner requirements and planning

Everything your IT department needs to plan a SourceLace Runner and get it approved before installing: where it can run, how big it must be, the database, running more than one instance so a failure needs nobody, every network connection, the accounts and secrets, the people and time it takes, and a checklist to tick off.

This is the first of four chapters for the IT department that installs and looks after a runner:

1. **Runner requirements and planning** (this chapter).
2. Installing a runner: step by step, with complete examples and a check after each step.
3. Runner troubleshooting: every message the runner can show, what it means and how to fix it.
4. Looking after a runner: health, upgrades, going back a version, backups and restores, moving a runner and changing its secrets.

What a runner is, what it costs and what stays where is in Runners: agents inside your own network; how it is secured, for your security reviewers, in Runner security.

Words used in these chapters

- **Runner**: a program SourceLace gives you that runs your organization's AI agents inside your network. People still build and manage agents in the SourceLace app.
- **Image**: the packaged runner program. Docker or Kubernetes downloads ("pulls") it from a **registry** (a private download server for images) and starts it as a **container** (a running copy of the image).
- **Registry login**: a user name (`AWS`) and password a SourceLace admin gets from the **Runners** page (**Get the runner**, **Pull from registry**) to download your organization's runner image. It works for one hour and can download nothing else: pull the image right after you get it, and get a new login from the same page if it has expired. Kubernetes clusters use a copy of the image in your own registry instead (Installing a runner, step 5).
- **Runner token**: a long secret starting with `slr_` that proves to SourceLace which runner is asking. A SourceLace admin creates it in the app; it is shown once.
- **Master key**: a secret you make yourself when you install. The runner encrypts everything it keeps (people's sign-ins, results, secrets, your AI key) with it. SourceLace never has it.
- **The runner's page**: the runner's own web pages, at an https address inside your network such as `https://runner.corvanta.internal`, where people open results, approve changes and connect their sources, and admins change its settings.
- **TLS proxy**: the reverse proxy, load balancer or Kubernetes ingress in your network that holds the https certificate for the runner's page and passes requests on to the runner.
- **Instance**: one running copy of the runner. A runner can run as several instances that share its database, master key and page address, so one can fail without anyone doing anything (Failover).

At a glance

Area	What you need
Where it runs	One 64-bit Linux machine (x86-64, also called amd64) with Docker Engine and the Docker Compose plugin, **or** a Kubernetes cluster with Helm 3. In your data center or in your own cloud account. For failover, two or more machines or Kubernetes nodes (Failover).
Processor and memory	For the runner alone: at least 1 GB of memory, 2 GB recommended; about one processor core for the default two runs at a time. See Sizing.
Disk	About 50 GB of free local disk for large results while agents work, plus up to 20 GB of temporary space. Nothing on it needs a backup.
Database	Its own PostgreSQL database, version 16 or 17, with backups. No extensions needed.
Network	Outbound HTTPS (443) to SourceLace, your AI provider and the image registry; your own systems on their ports; **no inbound connection from the internet**. An internal DNS name, an internal TLS certificate and a TLS proxy for the runner's page.
Secrets	A secret store or password manager for the runner token and the master key.
People	A container administrator, a database administrator, a network or security engineer and a SourceLace admin; one experienced person can cover several roles. See Skills and time.
SourceLace	The Enterprise plan with runners on your contract (Who can have a runner), and your organization's own Anthropic or OpenAI API key.

Where it can run

The runner is a container image for **64-bit Linux on x86-64 (amd64) processors**. It runs anywhere that runs such containers and meets the network rules below. These are examples of hosts that meet the requirements, not a list of separately tested products:

- **A Linux virtual machine or server** with Docker Engine and the Docker Compose plugin (the `docker compose` command; the older separate `docker-compose` program is not enough): on premises (such as VMware or Hyper-V), Amazon EC2, an Azure virtual machine or Google Compute Engine. Any current Linux distribution that Docker supports, such as Ubuntu LTS, Debian, Red Hat Enterprise Linux or Amazon Linux. Use a Docker Engine version that Docker still supports.
- **A Kubernetes cluster** with Helm 3: your own cluster, Amazon EKS, Azure Kubernetes Service (AKS) or Google Kubernetes Engine (GKE). The chart uses the standard Kubernetes ingress (`networking.k8s.io/v1`), so Kubernetes 1.19 or newer; use a version your provider still supports.

ARM processors (such as AWS Graviton or Ampere): runner images are built for x86-64. Use an x86-64 machine or node.

What the container needs from the host: nothing special. It runs as an unprivileged user (number 10001), its own files are read-only, it writes only to a temporary folder and the folder for large results, and it needs no privileged mode, no extra Linux capabilities and no access to the Docker socket. It has no shell inside, so nobody can open a command line in it.

Sizing

The runner runs **two agent runs at a time** by default (the setting `SOURCELACE_AGENT_PARALLEL_RUNS`, up to 100). Your organization's own limit on runs at once in SourceLace still applies on top. Most of a run's time is spent waiting for the AI model and your systems. Memory matters most when runs hold large results: the runner keeps them in a local engine that it sizes automatically, at each start, to the memory, processors and disk it finds (the setting `SOURCELACE_ENGINE_MEMORY_MB` can only make it smaller). With less than about 1 GB of memory, large results are turned off.

These sizes are **starting points**, not limits. Watch free memory and disk on the **Runners** page and adjust:

Size	Runs at a time	Processor	Memory for the runner	Disk for large results	Temporary space
Small (the default)	2	1 core	2 GB	50 GB	up to 20 GB
Medium	4	2 cores	4 GB	50 GB	up to 20 GB
Large	8	4 cores	8 GB	100 GB	up to 20 GB

- **On one machine with Docker Compose**, add what the operating system and Docker need, and the database's own processor and memory if it runs on the same machine (for example 1 core and 2 GB more). In the Docker Compose example the temporary space is kept in memory (see Storage).
- **In Kubernetes**, the chart's runner values ask for a quarter of a core and 1 GB to be guaranteed, with a 2 GB memory limit. Raise `resources` for a medium or large runner.
- Each **instance** of a runner (one container or one pod) runs that many runs at a time. For more capacity, give it more runs at a time and more memory, or run more instances of it (Failover); your organization's limit on runs at once in SourceLace applies to all of them together.

Storage

What	Size	Where	Backup
Large results while agents work	About 50 GB free	Local disk on the machine or Kubernetes node. No shared or network storage is needed.	None: results are encrypted, kept only while a run uses them, deleted when they expire, and may be gone after a restart.
Temporary space	Up to 20 GB	Docker Compose example: in memory (a `tmpfs`). Kubernetes: the node's temporary storage.	None.
The runner's database	Start with 20 GB, with automatic growth	Your PostgreSQL (Database).	Yes: this is what you keep.

Database

- **PostgreSQL 16 or 17.** The examples use 16, and SourceLace's own tests use 17. Ask your SourceLace contact before using another version.
- **Its own database**, not shared with other software, and a database user that **owns** it: the runner creates and updates its own tables each time it starts. The user needs no rights outside that database, and no extensions are needed.

- **Connections**: each instance of the runner opens up to 20 connections to its database (15 for its work, and up to 5 to agree with its other instances who does what). Allow at least 25 per instance for its user: 75 for three instances.
- **Managed or your own**: any PostgreSQL service works, for example Amazon RDS for PostgreSQL, Azure Database for PostgreSQL or Google Cloud SQL for PostgreSQL, or a PostgreSQL server your team runs.
- **What it holds**: run results, the files agents make, changes waiting for approval, people's sign-ins and secrets (encrypted with the master key), and the runner's own records, kept for your retention period (30 days unless you change it).
- **Backups**: your usual PostgreSQL backups, plus the master key kept separately (Backups).
- **TLS**: to require an encrypted connection to the database, add `?sslmode=require` to its address.
- The Docker Compose example includes a small PostgreSQL next to the runner, with a fixed password, for trying it out only.

Failover: more than one instance

What it is. A runner can run as several **instances**: identical copies of it, on different machines or Kubernetes nodes, that share **one database, one master key and one page address**. Each asks SourceLace for work on its own, and whichever asks first takes the next run. If a machine fails, its runs move to another instance within about three minutes, and new runs go to the instances still working. Nobody needs to do anything.

Why share one database. People's sign-ins to your systems, the runner's secret settings and the changes waiting for approval are kept only in the runner's database, inside your network. Instances that share it all have them: a person signs in once, and an approval can be decided through any instance.

	One instance	Two or more instances
A machine or pod stops	Runs wait until it is back; admins are told when it has been quiet too long	Another instance takes its runs over within about three minutes; nobody is told unless all stop
Upgrading	A short pause while it restarts	One instance at a time, with no pause (Upgrading)
What you install	One machine or one pod	The same settings on each: the same database address, master key, runner token and page address
Your contract	One runner	Still one runner: up to 10 instances take work at once, at no extra cost

What you need for two or more:

- **Kubernetes:** set `replicaCount` to 2 or more in your values (Installing, Kubernetes). The chart then keeps at least one instance up while nodes are drained, and you can spread them over nodes with the example `affinity` in the chart's runner values.
- **Docker Compose:** a second Linux machine with the same Docker Compose file and the same settings file, but pointing at your shared PostgreSQL instead of the example database, and your TLS proxy sending the runner page's address to both machines.
- **The database** is then the one thing they all depend on: use a PostgreSQL service that keeps a standby copy and switches to it by itself (such as Amazon RDS Multi-AZ, Azure Database for

PostgreSQL with high availability, or Google Cloud SQL with high availability), or your own PostgreSQL with a standby.

- **Each instance names itself** with its host name (on Kubernetes, the pod's name). The **Runners** page in SourceLace lists each one, its version and whether it is asking for work. To choose the names yourself, set `SOURCELACE_RUNNER_INSTANCE` on each, such as `frankfurt-1` and `frankfurt-2`.

Limits:

- At most **10 instances** of one runner take work at once. An 11th waits, and its page says why, until one of the others stops.
- Instances of one runner must share its database. A copy started with **its own** database is not an instance: it would not have people's sign-ins or the approvals. To run in two places with separate databases, add a second runner.

Network

The runner only ever connects **out**. It asks SourceLace for work, calls your AI provider and reaches your own systems. Nothing on the internet connects in, and SourceLace never connects to the runner.

Every connection

From	To	Port and protocol	What for
The runner	SourceLace: `sourcelace.ai` (leave this unless your SourceLace contact gives you another address)	443, HTTPS	Asks for work about every 15 seconds (SourceLace can ask it to wait longer); while a run is going, confirms about every minute that it still holds it; reports each run's status; reads your sources' settings every 5 minutes; receives its license, renewed about once a day within the same requests; completes people's sign-ins to its page. The connectivity check also opens SourceLace's health address.
The machine that downloads the runner image from SourceLace (the runner machine with Docker Compose, or the machine that copies it into your own registry for Kubernetes)	SourceLace's image registry on Amazon ECR: the registry host **Get the runner** shows (it ends in `.dkr.ecr.us-west-2.amazonaws.com`), and Amazon's storage for image layers, `prod-us-west-2-starport-layer-bucket.s3.us-west-2.amazonaws.com`	443, HTTPS	Downloading the runner image, only when you install or upgrade. Not needed if you download the image as a file instead (Get the runner).
The runner	Your AI provider: `api.anthropic.com` (Anthropic) or `api.openai.com` (OpenAI)	443, HTTPS	Every call to the AI model, with your organization's own key.

From	To	Port and protocol	What for
The runner	Each of your own systems the runner's agents use, and the sign-in addresses of their vendors (for example `login.microsoftonline.com` for Microsoft 365)	Their own ports: 443 for cloud apps; the database's port for databases, such as 5432 (PostgreSQL), 1433 (SQL Server), 1521 (Oracle) or 3306 (MySQL)	Reading and, where your admins allow it, changing data, as each agent's owner; renewing people's sign-ins. Each source's chapter in this manual lists its addresses.
The runner	Its own PostgreSQL database	5432, or the port you chose	Everything the runner keeps.
The runner	Your DNS servers	53	Finding the addresses above.
The runner	Your time server	123 (NTP), from the machine	Keeping the clock right (Clock).
People's browsers, inside your network	Your TLS proxy, at the runner page's address	443, HTTPS	Opening results, approvals, sources and settings on the runner's page.
Your TLS proxy	The runner	8000, HTTP. In the Docker Compose example, only on the machine's own loopback address (127.0.0.1)	Passing people's requests on; also `/health` for your proxy's or monitoring's health check.
People's browsers	`sourcelace.ai`	443, HTTPS	Signing in to the runner's page goes through SourceLace, which people already reach to use the app.

Inbound from the internet: none. Do not publish the runner's page on the internet or a public load balancer.

SourceLace's own IP addresses do not apply to a runner. Some source chapters ask you to allow SourceLace's IP addresses in a vendor's allow-list. A runner reaches your systems from **your** network, so allow your own outbound address for the runner there instead.

Outbound proxy

If your network reaches the internet only through a proxy, give the runner the standard proxy settings:

- `HTTPS_PROXY`: the proxy, such as `http://proxy.corvanta.internal:3128`. A user name and password may be part of the address; the runner never shows them on its pages.
- `NO_PROXY`: what the runner reaches directly, separated by commas, such as `.corvanta.internal,10.0.0.0/8`. List your internal systems here.

Connections to SourceLace, to your AI provider and to most cloud apps use the proxy. Some connections **always go directly**, never through a proxy: databases, and sources whose address your admin types in (such as REST and SOAP sources and outside MCP servers). Allow those in your firewall. The runner's connectivity check shows whether it used the proxy (Check connectivity).

TLS inspection and your own certificate authority

The runner checks every certificate it is shown, and connects with TLS 1.2 or newer. It trusts the usual public certificate authorities. If your network inspects TLS (a proxy that re-signs traffic with your company's own certificate authority), or one of your systems uses a certificate from your own or a self-signed authority, the runner refuses those connections until it trusts that authority. Either:

- let the runner's traffic to `sourcelace.ai` and your AI provider through without inspection, or

- give the runner a file of trusted certificate authorities in PEM format and point `SSL_CERT_FILE` at it: a mounted file with Docker Compose (Installing a runner, 4.5), or a ConfigMap with the Helm chart (on Kubernetes). This file **replaces** the runner's built-in list, so it must hold your company's authority **and** the usual public authorities. Your operating system's bundle with your authority added is the usual choice.

Firewall rules in plain form

Your network team can turn these into rules for your firewall or proxy. Replace the parts in angle brackets.

```
ALLOW  <runner host or subnet>  ->  sourcelace.ai                       TCP 443   HTTPS
ALLOW  <runner host or subnet>  ->  api.anthropic.com or api.openai.com  TCP 443   HTTPS
ALLOW  <image-pulling host>     ->  <registry host from Get the runner>  TCP 443   HTTPS  (insta
  ll and upgrade only)
ALLOW  <image-pulling host>     ->  prod-us-west-2-starport-layer-bucket.s3.us-west-2.amazonaws.
  com  TCP 443  HTTPS  (install and upgrade only)
ALLOW  <runner host or subnet>  ->  <each system the agents use>         TCP <its port>
ALLOW  <runner host or subnet>  ->  <database host>                      TCP 5432
ALLOW  <runner host or subnet>  ->  <DNS servers>                        UDP/TCP 53
ALLOW  <internal user networks> ->  <TLS proxy>                          TCP 443   (runner page)
ALLOW  <TLS proxy>              ->  <runner host>                        TCP 8000  (only if the
  proxy is on another machine)
DENY   internet                 ->  <runner host>                        any
```

And as a request you can send as it is:

> Please allow outbound HTTPS (TCP 443) from the SourceLace Runner at `<runner host or subnet>` to `sourcelace.ai` and to `<api.anthropic.com or api.openai.com>`, and, from the host that pulls the runner image, to `<registry host from Get the runner>` and `prod-us-west-2-starport-layer-bucket.s3.us-west-2.amazonaws.com`. Please allow the runner to reach `<each internal or cloud system, with its host and port>` and its database `<database host>:<port>`. No inbound connection from the internet is needed. Internally, please allow users to reach `<runner page address>` on 443 through `<TLS proxy>`, which forwards to the runner on TCP 8000.

The runner page's address

The runner's page needs its own **https** address inside your network, such as `https://runner.corvanta.internal`:

- an internal **DNS name** pointing at your TLS proxy;
- a **TLS certificate** for that name from your internal certificate authority, so people's browsers trust it;
- a **TLS proxy** (such as nginx, HAProxy, a load balancer or a Kubernetes ingress) that only your network reaches, passing requests to the runner on port 8000. Examples are in Installing a runner.

Two more addresses come from it: people come back to `<runner page address>/login/callback` after signing in, and the OAuth apps your organization registers for sources on the runner use `<runner page address>/connect/callback`. Choose the name carefully: changing it later means changing those OAuth apps too.

Clock

Keep the machine's clock synchronized (NTP). The runner's license is valid until a time, and a clock far ahead of real time makes it look expired.

Accounts and access

Who	What they need
A SourceLace admin in your organization	The admin role in SourceLace, to create the runner and copy its token, and later to sign in to the runner's **Settings** page.
Your container administrator	A registry login from a SourceLace admin (or the image file), and rights to run containers on the machine (membership of the `docker` group, or `sudo`), or to install a Helm release in a namespace and create secrets there.
Your database administrator	Rights to create a database and a user that owns it. The runner's user needs no rights outside its own database.
Your network team	Rights to change outbound firewall or proxy rules, internal DNS, and the TLS proxy or ingress, and to issue an internal certificate.
Your secret store	A place for the runner token and the master key, such as your password manager or a secrets manager (AWS Secrets Manager, Azure Key Vault, Google Secret Manager, HashiCorp Vault or Kubernetes secrets).

Looking after it

What running a runner involves after installation; the steps are in Looking after a runner:

- **Health**: watch `https://<runner page address>/health` with your monitoring (HTTP 200 when well, 503 when the runner cannot reach its database), and the runner's status on the **Runners** page.
- **Logs**: the runner writes its log to the container's standard output. Docker keeps it, and your usual log collection can pick it up from Docker or Kubernetes. It holds no tokens, keys, passwords or data from your systems.
- **Backups**: the database, and the master key kept apart from it.
- **Upgrades**: a new runner version when SourceLace releases one, following the upgrade plan you agree with your SourceLace contact.
- **Clock**: synchronized.
- **Disk and memory**: shown on the **Runners** page for each runner.

Skills and time

No certifications are needed. In a small setup, one experienced administrator often covers the first three roles.

Role	Tasks	Level needed
Linux and container administrator	Install Docker and Compose, or use Kubernetes and Helm; sign in to a private registry and pull an image (for Kubernetes, also push it to your own registry); fill in a settings file or Kubernetes secrets; start, restart and upgrade the runner; read its log	Comfortable running Docker Compose on Linux, or installing a Helm chart with your own values

Role	Tasks	Level needed
Database administrator	Create the PostgreSQL database and its user; set up backups and test a restore; upgrade PostgreSQL over time	Comfortable creating databases and users and running backups in your PostgreSQL service
Network or security engineer	Internal DNS name, internal TLS certificate, TLS proxy or ingress; outbound firewall rules; the outbound proxy and TLS inspection if you have them	Comfortable adding a DNS record, issuing an internal certificate and changing firewall or proxy rules
SourceLace admin (in your organization)	Create the runner and copy its token; enter the AI key, secret source options and OAuth apps on the runner's page; choose which agents run there	Comfortable with the SourceLace app's admin pages, and with registering an OAuth app at a vendor as each source's chapter describes

Time, as an estimate to plan with:

- **First installation:** about half a day to one day of work in total across the roles above, plus however long your organization takes to approve and open firewall rules, issue a certificate and create a database.
- **Ongoing care:** about one to two hours a month to check health, backups and disk, plus about an hour for each upgrade (backup, new version, check).

Where SourceLace helps:

- **Onboarding** (the one-time onboarding fee): help with installing the runner, and a security review.
- **Premium support** (part of every runner contract): faster response times, named contacts and a scheduled upgrade plan.
- The runner's own **Check connectivity** and **Download support bundle**, and Before you contact support, so that one message to support is enough.

Pre-installation checklist

Print this page or copy the list, and tick each line before you start Installing a runner.

Done	Item	Owner
☐	Runners are on your contract, and the **Runners** page shows in the SourceLace app	SourceLace admin
☐	The install bundle downloaded, the version chosen, and the image pulled or downloaded as a file from **Get the runner** (Get the runner)	SourceLace admin and container administrator
☐	For Kubernetes: a registry of your own the cluster pulls from, to hold a copy of the runner image	Container administrator
☐	A machine (x86-64 Linux, Docker Engine and Compose plugin) or a Kubernetes namespace (Helm 3), sized from Sizing	Container administrator
☐	About 50 GB of free local disk for large results	Container administrator
☐	The machine's clock synchronized (NTP)	Container administrator
☐	An empty PostgreSQL 16 or 17 database and its owner user, with backups on, at least 20 connections	Database administrator
☐	The database's address written down, and the runner allowed to reach its port	Database administrator

Done	Item	Owner
☐	Internal DNS name for the runner's page	Network engineer
☐	Internal TLS certificate for that name, and a TLS proxy or ingress only your network reaches	Network engineer
☐	Outbound rules open: SourceLace, AI provider, registry, each system the agents use, DNS (Firewall rules)	Network engineer
☐	Outbound proxy settings (`HTTPS_PROXY`, `NO_PROXY`), if your network has a proxy	Network engineer
☐	Your certificate authority file, if your network inspects TLS or your systems use internal certificates	Network engineer
☐	A place in your secret store for the runner token and master key	Container administrator
☐	Your organization's Anthropic or OpenAI API key, for the runner's **Settings** page	SourceLace admin
☐	For each source on the runner that signs in through your own OAuth app: who will register it at the vendor with the runner's callback address	SourceLace admin
☐	Which agents will run on the runner, and who owns them (each owner connects their own sources on the runner)	SourceLace admin