Search index: OpenSearch, Elasticsearch, Azure AI Search

From the SourceLace manual: Your own search index, searched from SourceLace. Many organizations already index their files (or other content) in a search service they run in their own cloud. Connected as a Search index source, that index answers searches from the SourceLace app, AI apps and agents, w

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

Search index: OpenSearch, Elasticsearch, Azure AI Search

Your own search index, searched from SourceLace. Many organizations already index their files (or other content) in a search service they run in their own cloud. Connected as a **Search index** source, that index answers searches from the SourceLace app, AI apps and agents, with a link back to each file. SourceLace copies nothing: each search goes to your index when someone asks, and the hits are held only like any other result (up to 30 minutes).

Status: Pilot. New or partial, offered to pilot customers. It is built on each service's published search API and has not been used with a customer's live index yet.

At a glance

	Connects through	Sign-in	Network	Status	First test question
Search index (OpenSearch, Elasticsearch, Azure AI Search)	The service's search API over HTTPS	Personal sign-in with `oauth`; shared account with an API key or login	Allow SourceLace's IP addresses, if the service limits them	Pilot	**Test connection** on **Connections**, then "Find the Corvanta renewal in search_index:docs."

Add each source with the usual steps. The network answers are explained in What your network needs.

When to use it

- **Your files are in Amazon S3, Azure Storage or Google Cloud Storage and you already index them.** Your index searches everything it holds at once, so it suits large stores and files people search all the time. For a few folders you do not index, an S3, Azure Storage or Google Cloud Storage source can search inside files itself: see How S3, Azure Storage and Google Cloud Storage are searched.
- **One index, one source.** Each source searches the one index you name. Add a source for each index people should search.
- **Searching only.** A Search index source finds things; it does not open or change files. Each hit links back to the file where it is kept, and if that store is also a SourceLace source, people can read the file there.

How people sign in

Kind: **Search index** (`search_index`). The admin picks how people sign in (`auth`):

Choice	What it means	Works with
Each person's own account (`oauth`, recommended)	Each person signs in as themselves, so your search service applies its own permissions to each search, including document-level security where you set it up.	Azure AI Search with Microsoft Entra ID. Elasticsearch and OpenSearch when your cluster accepts sign-in through your identity provider.

Choice	What it means	Works with
A shared API key (`api_key`)	Everyone searches as one key. Labelled **Shared account** everywhere; use access by group to choose who may use the source.	Elasticsearch (an API key) and Azure AI Search (a query key).
A shared login (`basic`)	Everyone searches as one user name and password. Labelled **Shared account**.	Elasticsearch and OpenSearch.

With a shared account, create a key or user that can only **read** the one index, nothing else.

1. Prepare the index (your search service's admin)

Follow the part for your search service. About 20 minutes.

Azure AI Search

Who: an Azure **Owner** or **User Access Administrator** on the search service, and, for each person's own sign-in, a Microsoft Entra **Application Administrator**.

Each person's own account (`oauth`, recommended):

1. In the Azure portal, open the search service. Under **Settings → Keys**, set **API access control** to **Role-based access control** or **Both**, and save. With the default (API keys only), Azure refuses everyone's sign-in.
2. Under **Access control (IAM) → Add → Add role assignment**, choose **Search Index Data Reader** and add the people or groups who should search through SourceLace.
3. In the **Microsoft Entra admin center → App registrations → New registration**, name it "SourceLace search", keep **Accounts in this organizational directory only**, and add the redirect URI type **Web** with exactly `https://mcp.sourcelace.ai/connect/callback`.
4. In the new app, **API permissions → Add a permission → APIs my organization uses**, choose **Azure AI Search** (also listed as Azure Cognitive Search), then **Delegated permissions → user_impersonation**. Optionally **Grant admin consent**, so people are not asked one by one.
5. **Certificates & secrets → New client secret.** Copy its **Value** at once (it is shown only once). The **Secret ID** next to it is not the secret.
6. From the app's **Overview**, copy the **Application (client) ID** and the **Directory (tenant) ID**.
7. Give the SourceLace admin: the service's address (`https://<service>.search.windows.net`, from the service's **Overview**), the index name, the client ID, the secret and the tenant ID.

A shared query key (`api_key`): under **Settings → Keys → Manage query keys → Add**, name it "SourceLace" and copy the key. A query key can only search.

Links: the Azure AI Search index made by the **Import data** wizard keeps each file's address in `metadata_storage_path`, encoded; SourceLace decodes it into a link. Microsoft's own guide: Azure AI Search documentation.

Elasticsearch

Who: someone with the `manage_security` privilege on the cluster (on Elastic Cloud, a deployment admin).

1. **Copy the Elasticsearch endpoint.** In the Elastic Cloud console, open the deployment and click **Copy endpoint** next to **Elasticsearch**. It has `.es.` in it. The Kibana address (with `.kb.`) does not work.
2. **Create an API key that may only read the index.** In Kibana, **Stack Management → API keys → Create API key**, name it "SourceLace", turn on **Control security privileges**, and give it read on the one index, for example:

```
{"sourcelace": {"indices": [{"names": ["docs"], "privileges": ["read", "view_index_metadata"]}]}
  }
```

   `view_index_metadata` lets **Test connection** check the field names. Copy the key's **Encoded** value: it is shown once, when the key is created. The key's id and its `api_key` part on their own do not work.
3. **Note the field names.** In Kibana, **Stack Management → Index Management**, open the index and look at **Mappings**. SourceLace's defaults are `title`, `url`, `content` and `modified`. Indexes filled by FSCrawler use `file.filename`, `path.virtual` (or `file.url`), `content` and `file.last_modified`; Elastic's web crawler uses `title`, `url`, `body_content` and `last_crawled_at`.
4. **For a shared login instead:** create a user whose role has `read` (and `view_index_metadata`) on the index.
5. **For each person's own account:** your cluster must accept tokens from your identity provider (a JWT or OpenID Connect realm). Register SourceLace there as an OAuth client with redirect URL `https://mcp.sourcelace.ai/connect/callback`, and give the SourceLace admin its authorization and token addresses, client ID and scopes.

Elastic's own guide: Elastic documentation.

OpenSearch

Who: a security admin of the cluster (the `all_access` or `security_manager` role).

1. In OpenSearch Dashboards, **Security → Roles → Create role**: index permissions `read` and `indices:admin/mappings/get` on the one index.
2. **Security → Internal users** (or your identity provider's backend roles): create a user for SourceLace and map it to that role, or map the people's own roles for each person's sign-in through OpenID Connect or JWT.
3. Note the field names from the index's mapping, as for Elasticsearch.

OpenSearch's own guide: OpenSearch documentation.

Network, every service: if the service accepts only some IP addresses, allow SourceLace's (ask support for the list).

Check that it worked: the SourceLace admin saves the source and presses **Test connection**. With a shared account, it searches once and checks that the index has the fields the source names. With each person's own account, one person clicks **Connect** and asks for a file they know is in the index.

2. Add the source (SourceLace admin)

Option	Type	Default	Example	What it is
`service`	Choice	(required)	`azure_ai_search`	`elasticsearch`, `opensearch` or `azure_ai_search`.
`endpoint`	Text	(required)	`https://corvanta.search.windows.net`	The service's address, starting with `https://`.
`index`	Text	(required)	`contracts`	The one index this source searches.
`auth`	Choice	(required)	`oauth`	How people sign in: `oauth`, `api_key` or `basic` (see above).
`secret`	Secret	(none)		The API key, query key or password, for a shared account. Stored encrypted and never shown again.
`username`	Text	(none)	`svc-sourcelace`	The user name, with `basic`.
`client_id`	Text	(none)	`1a2b3c4d-...`	With `oauth`: the OAuth client registered for SourceLace.
`client_secret`	Secret	(none)		Its client secret, for a confidential client. Stored encrypted and never shown again.
`tenant`	Text	(required with `oauth`)	`corvanta.onmicrosoft.com`	Azure AI Search with `oauth`: your Microsoft Entra tenant's ID or domain (Entra admin center, **Overview**, Tenant ID).
`authorize_url`, `token_url`, `scopes`	Text	Filled in for Azure AI Search		Elasticsearch and OpenSearch with `oauth`: your identity provider's addresses and scopes.
`title_field`	Text	`title` (Azure AI Search: `metadata_storage_name`)	`file.filename`	The field that holds each file's title or name.
`link_field`	Text	`url` (Azure AI Search: `metadata_storage_path`)	`path.real`	The field that holds where the file is kept. Shown as a link when it is a web address.
`content_field`	Text	`content`	`content`	The field that holds the text to search and show passages from.
`modified_field`	Text	`modified` (Azure AI Search: `metadata_storage_last_modified`)	`file.last_modified`	The field that holds when the file last changed, for searches by date.

Then press **Test connection**: with a shared account, SourceLace searches the index once, says whether it answered, and checks that the index has the title, link, text and date fields the source names.

How people use it

People ask in plain words ("Find the Corvanta renewal contract in the contracts index"). Search index sources take the `files` query language with one action, `search`:

```
{"action": "search", "text": "corvanta renewal", "modified_after": "2026-09-01", "limit": 20}
```

Each hit has the file's `name`, a `link` back to it, its `location` as the index gives it, when it last `modified`, a `snippet` showing where the words are, and the index's own `score`. Add `"match": "name"` to search titles only. Search index sources are also searched by Searching everything, where it is on. Each search is in your audit trail, with how many hits it found; never their text.

When something goes wrong

What you see	What to do
"Set the option service: elasticsearch, opensearch or azure_ai_search ..." (or endpoint, index, auth)	Fill in that option on **Connections**.
"OpenSearch has no API keys: choose basic (a shared login) or oauth."	Choose another way to sign in.
"Azure AI Search takes no login: choose api_key (a query key) or oauth (Microsoft Entra ID)."	Choose another way to sign in.
"... refused the search: your account may not search the index ..."	Ask your search service's admin for read access to the index (Azure AI Search: Search Index Data Reader).
"... refused the search: the shared account the admin set may not search the index ..."	The key or login is wrong or cannot read the index. Enter a new one on **Connections**.
"... has no index called ...."	Check the `index` option.
"... could not search (...)"	The service's own reason follows, such as a field that does not exist. Check the field options.
"That is Kibana's address. Use the Elasticsearch endpoint instead ..." or "... answered with a redirect, which SourceLace never follows. ..."	Enter the Elasticsearch endpoint: Elastic Cloud console, the deployment, **Copy endpoint** next to Elasticsearch.
"Set Microsoft Entra tenant: your tenant's id or domain ..."	Enter the Directory (tenant) ID from the app's **Overview** in the Entra admin center.
"Azure AI Search refused your sign-in. In the Azure portal, ..."	Do steps 1 and 2 of Azure AI Search, then connect again.
"... did not accept the API key. Paste the key's Encoded value ..."	Create a new key and paste its **Encoded** value (step 2 of Elasticsearch).
"... lacks the read privilege on the index .... Give its role read on ...."	Give the key's, user's or person's role `read` on the index.
"The index has no field ... (the text to search), so searches find nothing. Text fields it has: ..."	Set **Text field** to one of the fields listed.
"The index has no field ..., so Title field shows nothing. ..." (or Link field, Date field)	Set that option to one of the fields listed.
"The index's list of fields could not be read with this key, so the field options were not checked."	Nothing is wrong. To have them checked, give the key `view_index_metadata` (Elasticsearch); an Azure query key can never read them.
"... is busy. Try again in a moment."	Your search service asked SourceLace to slow down. Try again shortly.

What you see	What to do
"A search index source only searches ..."	Ask for a search. To read a file, open its link, or read it from the file source that holds it.
"... sent back something other than search results."	The address answered with something that is not a search service (such as a web page).