



Connect your website to LazySEO

Step-by-step setup for every publishing integration: WordPress, Webflow, Ghost, Shopify, Sanity, Dev.to, Webhook and Custom API, plus how to publish on a subdomain.

Last reviewed 1 October 2026 · Latest version: lazyseo.io/help/integrations · Help: support@lazyseo.io

Connect your website

LazySEO writes your articles and publishes them to your website through an integration: a saved connection to your blog platform. Pick the integration that matches where your blog lives, follow its guide, and LazySEO can publish for you.

Which integration should I choose?

Your situation	Choose	Why
My site runs on WordPress (including WooCommerce)	WordPress	Built into WordPress, no plugin needed.
My site is built with Webflow	Webflow	Adds items to your blog CMS collection.
My blog runs on Ghost	Ghost	Uses a Ghost custom integration.
I want posts on my Shopify store blog	Shopify	Adds posts to a blog in your store.
My content lives in Sanity	Sanity	Writes Portable Text documents with images.
I want to post on the DEV Community	Dev.to	Posts to your dev.to account.
I have a developer and a custom-built site	Webhook	Your developer adds an endpoint that receives each article as signed JSON.
I have a developer and an existing API or headless CMS	Custom API	Shape the request to an API you already have.
I only have a domain or subdomain and no website or CMS yet	Set up a blog first	Install WordPress on your domain or a subdomain, then connect it.

Webhook or Custom API?

Webhook: LazySEO decides the format. It sends the same JSON payload every time (title, slug, content as Markdown or HTML, meta description, image URL, topics), optionally signed with a secret. Your developer writes a small endpoint that saves it. Best when you are building the receiving side anyway.

Custom API: your API decides the format. You enter its URL, method and authentication headers, and a body template with placeholders such as `{{title}}` and `{{content}}`. Best when the API already exists and you can't change it.

Using Wix, Squarespace, Framer, HubSpot or another builder? There is no direct integration for these. If your platform has an API for creating blog posts that accepts a key in a header, **Custom API** may work. Otherwise, a common option is a WordPress blog on a subdomain: see **Publish on a subdomain**.

When does LazySEO publish?

The **Publishing Settings** on the Integrations page control when articles go out:

- **Require manual approval** on: new articles wait in Review until you approve them.
- **Auto-publish articles** on: approved articles are sent to every enabled integration automatically, as soon as they are ready or when their scheduled date arrives (LazySEO checks every 15 minutes).

- **Auto-publish articles** off: you publish each article yourself with **Publish** on the article page.

Contents

1. WordPress

Self-hosted WordPress and WooCommerce sites, using the built-in REST API and an application password.

2. Webflow

Adds articles as live items in a Webflow CMS collection, such as your Blog Posts collection.

3. Ghost

Ghost(Pro) or self-hosted Ghost, using a custom integration and the Ghost Admin API.

4. Shopify

Adds articles as blog posts in your Shopify store, using an Admin API access token.

5. Sanity

Writes articles as documents in your Sanity dataset, with Portable Text and images uploaded to Sanity.

6. Dev.to

Posts articles to your account on the DEV Community (dev.to), using a DEV API key.

7. Webhook

For developers: LazySEO sends each article as signed JSON to an endpoint on your site, and your code creates the post.

8. Custom API

For developers: send articles to an API you already have (headless CMS, backend), with your own headers and JSON body.

9. Subdomain or no website

Only have a domain, or want a separate blog at blog.yourdomain.com? Set up a blog platform first.

1. SETUP GUIDE

WordPress

LazySEO publishes to WordPress through the WordPress REST API, signing in with your username and an application password. Application passwords are built into WordPress (since version 5.6), so you do not need a plugin.

Online version: lazyseo.io/help/integrations/wordpress

What you need

- A WordPress site that loads over **https://**. WordPress only offers application passwords on sites served over HTTPS.
- A WordPress user that can publish posts and upload images: the **Author**, **Editor** or **Administrator** role. **Contributor** and **Subscriber** accounts cannot publish, and the connection test tells you so. Pick **Editor** (or Administrator) if you want LazySEO to create new tags; an Author can only use tags that already exist. A dedicated user (for example "LazySEO", role Editor) makes it easy to see and revoke what LazySEO posts.
- The WordPress REST API, which is on by default. Any permalink setting works, including **Plain**: LazySEO then uses the `?rest_route=` form of the API address.

Step by step

1. Create an application password in WordPress

- Log in to your WordPress admin as the user LazySEO should publish as.
- Go to **Users** → **Profile** (or **Users** → **All Users** and edit that user).
- Scroll down to **Application Passwords**.
- In **New Application Password Name**, type `LazySEO` and click **Add New Application Password**.
- Copy the password WordPress shows (24 characters in groups of four, like `abcd efgh ijkl mnop qrst uvwx`). WordPress shows it only once. The spaces are fine to keep.

2. Open the integration form in LazySEO

- Open your project in LazySEO.
- In the left sidebar, under **Settings**, click **Integrations**.
- Click **Add Integration**.
- In the window that says "Choose the type of integration you want to add.", click **WordPress**.

3. Save the integration

Fill in the fields as described in the table below, leave **Enabled** switched on (disabled integrations are skipped during publish) and click **Add integration**.

Filling in the form in LazySEO

Field	What to enter	Example
Name REQUIRED	Any name that helps you recognise this connection. Only shown inside LazySEO.	Company blog (WordPress)

Field	What to enter	Example
Site URL REQUIRED	The address of your WordPress site, including https:// (or http://). Paste it the way you see it in the browser: LazySEO follows redirects (http to https, with or without www) and removes a /wp-admin or /wp-json ending if you include one. The form reminds you to remove those endings, but you can still save. A trailing slash is fine. If WordPress is installed in a subfolder, include the subfolder.	https://www.example.com (or https://www.example.com/blog)
Username REQUIRED	The WordPress username (the login name) of the user who owns the application password. Not the display name.	lazyseo
Application Password REQUIRED	The application password you just created. Not the password you log in with.	abcd efgh ijkl mnop qrst uvwx
Enabled	Leave on. When it is off, LazySEO skips this integration when publishing.	

Test the connection

- Right after you save, the window shows **Test the connection**: click **Run quick test**. You can test again at any time from the integration card with **Test**.
- **Quick test** ("Check endpoint is reachable") is read-only: LazySEO finds your site's REST API, signs in with the username and application password, and checks that the user can write and publish posts (the `edit_posts` and `publish_posts` capabilities). A Contributor or Subscriber account fails with a reason that says so. Nothing is created.
- On the card's **Test** menu, **Full test** ("Send a test blog post (draft)") sends a real test article titled "[TEST] LazySEO Test Article" with the slug `lazyseo-test-article`. In WordPress it is created as a **Draft** post, so it is not visible to visitors. Delete it from **Posts** once you have checked it.
- If a test fails, LazySEO shows the reason it got from your platform (for example a rejected key or a missing permission). Check it against the troubleshooting section below, click **Edit** on the card to fix the field, and test again.

What gets published

Title	The article title becomes the post title.
Content	The full article body as HTML. LazySEO converts the article's Markdown to HTML (headings, lists, links, tables, code), sends articles you edited in the LazySEO editor as the HTML you saved, and drops a first heading that only repeats the title, because WordPress shows the title itself.
Slug	The article slug is used as the post slug (the end of the URL).
Status	Published immediately (WordPress status "publish"). The Full test creates a draft instead.
Excerpt	The meta description is saved as the WordPress post excerpt. LazySEO does not write to SEO plugin fields such as Yoast or Rank Math.

Tags	The article topics become WordPress tags. LazySEO uses the existing tag when there is one, and creates missing tags if the user can manage tags (Editor or Administrator). With an Author account, missing tags are skipped and the post is still published; Sync History lists the skipped tags in the publish's response.
Featured image	The article banner is uploaded to your Media Library and set as the featured image. If that upload fails, the post is still published without a featured image, and Sync History shows why. The image is uploaded again each time the article is republished.
Category	LazySEO does not choose a category, so WordPress uses your default category.
Post URL	LazySEO saves the link WordPress returns as the article's published URL.
Updates	If the same article is published again to the same integration, LazySEO updates the existing post instead of creating a second one. If the post was deleted in WordPress in the meantime, LazySEO publishes it again as a new post.
Unpublishing	Unpublishing an article inside LazySEO does not remove it from your site. Delete or unpublish the post in your platform if you need it gone.

Troubleshooting

There is no "Application Passwords" section on the profile page.

Your site is probably not served over HTTPS, or a security plugin has turned application passwords off. Enable HTTPS (most hosts offer free SSL) or check your security plugin settings (for example Wordfence: "Disable WordPress application passwords").

Test fails, or publishing fails with HTTP 401 (for example `rest_not_logged_in`, `incorrect_password` or `application_passwords_disabled`).

Check the **Username** and that **Application Password** is the application password, not your login password. `application_passwords_disabled` means a security plugin has turned application passwords off. If the details are right and you see `rest_not_logged_in`, your host or a security plugin may be stripping the `Authorization` header. Ask your host to pass the Authorization header through to PHP, and allow REST API access in your security plugin.

The test says the user "can't write posts" or "can only submit posts for review".

The WordPress user is a Subscriber or a Contributor. Give it the Author, Editor or Administrator role (Users → All Users → edit the user → Role), or use an application password from a user that has one of those roles.

Publishing fails with HTTP 403 (for example `rest_cannot_create` or `rest_cannot_publish`).

The WordPress user is not allowed to publish posts or upload files. Give it the Author, Editor or Administrator role.

The test says "No WordPress REST API found" (it checked `/wp-json` and `?rest_route=/`).

Check that **Site URL** is the address of your WordPress site (with the subfolder, if WordPress is installed in one). If it is, a security plugin or firewall is blocking the REST API: allow access to it in the plugin settings.

Publishing fails with HTTP 404 `rest_no_route`.

The posts endpoint is missing, usually because a plugin has disabled the REST API or the posts route. Re-enable it in that plugin.

The error says "WordPress redirected the request".

Your site sent LazySEO to another address. Set **Site URL** to the address the redirect points to (the one shown in the error).

The error shows an HTML page from Cloudflare, a firewall or "Access denied".

A firewall is blocking server-to-server requests. Allow requests to the REST API on your site (`/wp-json/` , or `?rest_route=` with Plain permalinks), for example with a WAF rule or an allowlist in your security plugin.

The post was published but has no featured image.

The image upload to your Media Library failed (upload permission, file size limit or a security plugin). The reason is shown in the publish's response under **Sync History**. The rest of the post was published normally; you can add the image by hand.

Some topics are missing from the post's tags.

The WordPress user is an Author, so it can only use tags that already exist. Create those tags in **Posts** → **Tags**, or give the user the Editor role so LazySEO can create them.

The Media Library has several copies of the same banner image.

Each publish uploads the banner again, including when an article is republished to update it. You can delete the older copies that no post uses.

Questions

Do I need to install a WordPress plugin for LazySEO?

No. LazySEO uses the REST API and application passwords that are built into WordPress 5.6 and later.

Does it work with Plain permalinks?

Yes. When the `/wp-json/` address is not available, LazySEO uses the `?rest_route=` form of the REST API, which works with every permalink setting.

What happens if I delete a LazySEO post in WordPress and publish the article again?

LazySEO notices that the post no longer exists and creates it again as a new post.

Does it work with WooCommerce stores?

Yes. WooCommerce runs on WordPress, so articles are published as normal WordPress blog posts on the same site.

Can I revoke LazySEO's access later?

Yes. In WordPress go to Users → Profile → Application Passwords and click Revoke next to the LazySEO password. LazySEO can no longer publish until you add a new one.

2. SETUP GUIDE

Webflow

LazySEO adds each article as an item in one Webflow CMS collection, using the Webflow Data API (v2) and a site API token. Before each publish it reads the collection's fields and fills the ones it recognises, so the main part of this setup is checking that your collection has those fields and no other required fields.

Online version: lazyseo.io/help/integrations/webflow

What you need

- A Webflow site on a plan that includes the CMS, with a collection for blog posts (the Webflow "Blog Posts" collection template works well).
- Site administrator access in Webflow (only site admins can create API tokens).

Step by step

1. Check the fields in your collection

LazySEO matches fields by their field slug and type. In the Designer, open the **CMS** panel and the collection settings to see your fields. Webflow builds a field's slug from the name you give it when you create it, so if you add a field, use one of these names:

- **Name** (`name`) and **Slug** (`slug`): built into every collection. Filled with the article title and slug.
- **Article body**, type Rich text: the first field with the slug `post-body`, `body`, `content`, `rich-text`, `post-content` or `article-body`; if there is none, the first Rich text field in the collection. The collection needs at least one Rich text field. The Blog Posts template has **Post Body**.
- **Summary**, type Plain text or Rich text, slug `post-summary`, `summary`, `excerpt`, `description` or `short-description`: the meta description. The Blog Posts template has **Post Summary**.
- **Meta description**, type Plain text or Rich text, slug `meta-description`, `seo-description`, `meta-desc` or `seo-meta-description`: the meta description again, so you can bind it to the page's SEO settings.
- **Images**, type Image, slug `main-image`, `thumbnail-image`, `image`, `featured-image`, `cover-image` or `hero-image`: every one of these fields gets the article banner. The Blog Posts template has **Main Image** and **Thumbnail Image**.
- **Tags** (optional), type Plain text, slug `tags`, `keywords` or `topics`: the article topics as comma-separated text. Option and Reference fields are not filled, because Webflow needs its own IDs for them.
- Any other **required** field (a required author or category reference, for example) stops publishing with an error that names it. Make those fields optional in Webflow.

2. Find the Collection ID

- In the Designer, open the **CMS** panel and open the settings of your blog collection; the Collection ID is shown there.
- You don't need the Site ID: LazySEO publishes with the Collection ID and the token only.

3. Create an API token

- In **Site settings**, open **Apps & integrations** in the left sidebar.
- Scroll to **API access** and click **Generate API token**.
- Name it **LazySEO** and give it **CMS** read and write access. LazySEO needs read access to see the collection's fields and write access to add items.
- Click **Generate token** and copy it. Webflow shows it only once.

4. Open the integration form in LazySEO

- Open your project in LazySEO.
- In the left sidebar, under **Settings**, click **Integrations**.
- Click **Add Integration**.
- In the window that says "Choose the type of integration you want to add.", click **Webflow**.

5. Save the integration

Fill in the fields as described in the table below, leave **Enabled** switched on (disabled integrations are skipped during publish) and click **Add integration**.

Filling in the form in LazySEO

Field	What to enter	Example
Name REQUIRED	Any name that helps you recognise this connection. Only shown inside LazySEO.	Webflow blog
Site ID (optional)	Not used for publishing. Leave it empty.	
Collection ID REQUIRED	The ID of the collection articles should be added to.	5f4d2c9a7b6 c5d4e3f2a1b0 c
API Token REQUIRED	The site API token with CMS read and write access, created on the same site as the collection.	a long string of letters and numbers
Site URL (optional)	The public address of your site, without a folder. LazySEO builds each article's link as Site URL + "/" + the collection's slug + "/" + the article slug, which is where Webflow serves collection pages: with the collection slug blog , a link looks like https://www.example.com/blog/my-article . Leave it empty if you don't need the link in LazySEO; articles still publish, but LazySEO has no live link to show or to send to Google Search Console.	https://www .example.co m
Enabled	Leave on. When it is off, LazySEO skips this integration when publishing.	

Test the connection

- Right after you save, the window shows **Test the connection**: click **Run quick test**. You can test again at any time from the integration card with **Test**.
- **Quick test** ("Check endpoint is reachable") reads your collection with the API token and checks that it has a Rich text field for the article body and no required fields LazySEO can't fill. Webflow can't confirm

write access without writing, so a token without CMS write access only shows up on the Full test or the first publish. Nothing is created.

- On the card's **Test** menu, **Full test** ("Send a test blog post (draft)") sends a real test article titled "[TEST] LazySEO Test Article" with the slug `lazyseo-test-article`. In Webflow it is added as a **Draft** item in your collection, so it is not live. Delete it from the CMS once you have checked it.
- If a test fails, LazySEO shows the reason it got from your platform (for example a rejected key or a missing permission). Check it against the troubleshooting section below, click **Edit** on the card to fix the field, and test again.

What gets published

Name	The article title.
Slug	The article slug. Slugs must be unique in a collection.
Body field	The full article body as HTML. LazySEO converts the article's Markdown to HTML (headings, lists, links, tables, code), sends articles you edited in the LazySEO editor as the HTML you saved, and drops a first heading that only repeats the title, because your collection template shows the title itself.
Summary and meta description fields	The meta description, written to each of these fields your collection has (shortened to the field's maximum length if needed).
Image fields	The article banner image, sent as an image URL for Webflow to import.
Tags field	The article topics as comma-separated text, only if the collection has a Plain text <code>tags</code> , <code>keywords</code> or <code>topics</code> field.
Status	Published to your live site straight away (Webflow's live item endpoints), so you don't need to publish the site. The Full test creates a draft item instead.
Post URL	Only recorded when Site URL (optional) is filled in: Site URL + "/" + collection slug + "/" + item slug.
Updates	If the same article is published again to the same integration, LazySEO updates the existing post instead of creating a second one. If the post was deleted in Webflow in the meantime, LazySEO publishes it again as a new post.
Unpublishing	Unpublishing an article inside LazySEO does not remove it from your site. Delete or unpublish the post in your platform if you need it gone.

Troubleshooting

The error says the collection "has required fields LazySEO can't fill".

The error lists the fields. Make them optional in the collection settings, then publish again. LazySEO only fills the fields listed in step 1.

The test says the collection "has no Rich Text field for the article body".

Add a Rich text field to the collection, ideally named **Post Body** (slug `post-body`).

HTTP 400 "Validation Error" that names a field.

Webflow rejected a value. The most common cause is a slug that another item in the collection already uses: change the article slug in LazySEO, or delete the old item. If the error names one of your own fields, check its type and validation settings in Webflow.

HTTP 401 ("Webflow rejected the API token").

The **API Token** is wrong or was deleted. Generate a new one in Site settings → Apps & integrations and paste it in again.

HTTP 403 ("The Webflow token lacks permission").

The token does not have CMS read and write access. Create a token on the same site with CMS read and write access.

HTTP 404 ("Webflow collection not found").

The **Collection ID** is wrong (it is not the Site ID), or the token belongs to a different site. Copy the ID again from the collection settings in the Designer.

HTTP 429 ("rate limit reached").

Webflow limits how many API requests can be made per minute. Wait a minute and use **Retry** on the article.

The article's tags don't appear in Webflow.

LazySEO only writes tags into a Plain text field with the slug `tags`, `keywords` or `topics`. Option, Reference and Multi-reference fields are skipped.

The link LazySEO shows for the article goes to a 404 page.

Set **Site URL (optional)** to your site's address without a folder, for example `https://www.example.com`. LazySEO adds the collection slug itself. If you changed the collection's URL in Webflow, the recorded link uses the collection slug Webflow reports.

Questions

Which Webflow collection should I use?

Your blog posts collection. The Webflow Blog Posts template already has Post Body, Post Summary, Main Image and Thumbnail Image; add a Plain text field named Meta Description (and one named Tags, if you want the topics) and it has everything LazySEO fills.

Do I need to publish my Webflow site after LazySEO publishes an article?

No. LazySEO publishes each item to the live site directly. Only the Full test creates a draft item.

Do I need the Site ID?

No. The field is optional and not used. LazySEO needs the Collection ID and an API token with CMS read and write access.

3. SETUP GUIDE

Ghost

LazySEO publishes to Ghost through the Ghost Admin API. You create a custom integration in Ghost, which gives you an Admin API key and an API URL, and paste both into LazySEO.

Online version: lazyseo.io/help/integrations/ghost

What you need

- A Ghost site (Ghost(Pro) or self-hosted).
- A Ghost staff account that can manage integrations (Owner or Administrator).

Step by step

1. Create a custom integration in Ghost

- In Ghost Admin, open **Settings** (the gear icon).
- Go to **Integrations** and click **Add custom integration**.
- Name it `LazySEO` and click **Create**.
- Copy the **Admin API key**. It has two parts separated by a colon, like `6489a1b2c3d4e5f6a7b8c9d0:1a2b3c...`. Do not use the Content API key: it cannot publish.
- Copy the **API URL**, for example `https://your-site.ghost.io`.

2. Open the integration form in LazySEO

- Open your project in LazySEO.
- In the left sidebar, under **Settings**, click **Integrations**.
- Click **Add Integration**.
- In the window that says "Choose the type of integration you want to add.", click **Ghost**.

3. Save the integration

Fill in the fields as described in the table below, leave **Enabled** switched on (disabled integrations are skipped during publish) and click **Add integration**.

Filling in the form in LazySEO

Field	What to enter	Example
Name REQUIRED	Any name that helps you recognise this connection. Only shown inside LazySEO.	<code>Ghost blog</code>
Site URL REQUIRED	The API URL shown on the custom integration page in Ghost. Leave off <code>/ghost</code> and <code>/ghost/api</code> , because LazySEO adds the API path itself (if you paste one of them, LazySEO removes it).	<code>https://your-site.ghost.io</code>
Admin API Key REQUIRED	The full Admin API key, including the part before and after the colon.	<code>6489a1b2c3d4e5f6a7b8c9d0:1a2b3c4d...</code>
Enabled	Leave on. When it is off, LazySEO skips this integration when publishing.	

Test the connection

- Right after you save, the window shows **Test the connection**: click **Run quick test**. You can test again at any time from the integration card with **Test**.
- **Quick test** ("Check endpoint is reachable") checks the Admin API key. A key that is not in the `id:secret` form (for example the Content API key) fails straight away; otherwise LazySEO signs in to the Ghost Admin API and reads one post, which proves both the API URL and the key. Nothing is created.
- On the card's **Test** menu, **Full test** ("Send a test blog post (draft)") sends a real test article titled "[TEST] LazySEO Test Article" with the slug `lazyseo-test-article`. In Ghost it is created as a **Draft** post. Delete it from **Posts** once you have checked it.
- If a test fails, LazySEO shows the reason it got from your platform (for example a rejected key or a missing permission). Check it against the troubleshooting section below, click **Edit** on the card to fix the field, and test again.

What gets published

Title	The article title.
Content	The full article body as HTML. LazySEO converts the article's Markdown to HTML (headings, lists, links, tables, code), sends articles you edited in the LazySEO editor as the HTML you saved, and drops a first heading that only repeats the title, because Ghost shows the title itself. It is sent with Ghost's <code>source=html</code> option, so Ghost converts it into its editor format.
Slug	The article slug.
Status	Published. The Full test creates a draft.
Excerpt	The meta description is saved as the post's custom excerpt.
Feature image	The article banner image URL is set as the feature image.
Tags	The article topics are added as Ghost tags (Ghost creates tags that don't exist yet).
Post URL	LazySEO saves the URL Ghost returns as the article's published URL.
Updates	If the same article is published again to the same integration, LazySEO updates the existing post instead of creating a second one. If the post was deleted in Ghost in the meantime, LazySEO publishes it again as a new post. Before updating, LazySEO reads the post's latest version from Ghost, so edits made in Ghost in between don't cause a conflict error. The update does replace the title, body, excerpt, feature image and tags with LazySEO's version.
Unpublishing	Unpublishing an article inside LazySEO does not remove it from your site. Delete or unpublish the post in your platform if you need it gone.

Troubleshooting

The test says the Admin API key should look like "id:secret", or publishing fails with "invalid adminApiKey — expected format: {id}:{secret}".

You pasted the Content API key or only part of the Admin API key. Copy the whole Admin API key into **Admin API Key**: two strings of letters and numbers joined by a colon.

HTTP 401 or 403 ("Ghost rejected the Admin API key").

The key is wrong or the custom integration was deleted. Copy the Admin API key again from Settings → Integrations.

HTTP 404 ("Ghost Admin API not found at this URL").

Check **Site URL**: use the API URL shown on the integration page.

HTTP 409 ("edit conflict").

The post was changed in Ghost at the same moment LazySEO updated it. Use **Retry** on the article.

Posts arrive in Ghost with an empty body.

This happened with older LazySEO versions. Publish the article again: LazySEO now sends the content in a form Ghost always imports.

Questions

Does it work with Ghost(Pro) and a custom domain?

Yes. Use the API URL that Ghost shows on the custom integration page.

4. SETUP GUIDE

Shopify

LazySEO publishes to a blog in your Shopify store (Online Store blog posts) through the Shopify Admin REST API (version 2026-07). It needs your store address, the ID of the blog, and a permanent Admin API access token with content permissions.

Online version: lazyseo.io/help/integrations/shopify

What you need

- A Shopify store with the Online Store sales channel and at least one blog (Shopify creates one called "News" by default).
- A permanent **Admin API access token** (it starts with `shpat_`) from a custom app in your store, with the `write_content` and `read_content` scopes.
- Check which kind of app you have before you start. Custom apps created in the Shopify admin (**Settings** → **Apps and sales channels** → **Develop apps**) before January 1, 2026 keep working and have a permanent token: that is what this integration uses. Since that date Shopify only lets you create new apps in its Dev Dashboard, and those get access tokens that expire after 24 hours, which LazySEO cannot renew yet. If you only have (or can only create) a Dev Dashboard app, email support@lazyseo.io and we will help you find another way to publish to your store.

Step by step

1. Get an Admin API access token (existing custom app)

- In Shopify admin, go to **Settings** → **Apps and sales channels** → **Develop apps** and open your custom app.
- Under **Configuration**, edit the **Admin API integration** scopes and tick `write_content` and `read_content` (these cover blogs and articles). Save. Shopify may ask you to install or update the app to apply the new scopes.
- Open **API credentials** and copy the **Admin API access token** (starts with `shpat_`). Shopify reveals it only once; if it was already revealed and you no longer have it, you need to uninstall and reinstall the app to get a new one.

2. Find your Blog ID

- In Shopify admin, go to **Content** → **Blog posts** (older admin: **Online Store** → **Blog posts**) and click **Manage blogs**.
- Open the blog you want LazySEO to publish to.
- Look at the browser address bar: the number after `/blogs/` is the Blog ID, for example `/blogs/84726351921`.

3. Open the integration form in LazySEO

- Open your project in LazySEO.
- In the left sidebar, under **Settings**, click **Integrations**.
- Click **Add Integration**.
- In the window that says "Choose the type of integration you want to add.", click **Shopify**.

4. Save the integration

Fill in the fields as described in the table below, leave **Enabled** switched on (disabled integrations are skipped during publish) and click **Add integration**.

Filling in the form in LazySEO

Field	What to enter	Example
Name REQUIRED	Any name that helps you recognise this connection. Only shown inside LazySEO.	Shopify blog
Store URL REQUIRED	Your store's myshopify.com address, starting with https://. Not your custom domain: the Admin API is served on the myshopify.com address. LazySEO looks up your primary domain itself for the article links.	https://your-store.myshopify.com
Admin API Access Token REQUIRED	The token from your custom app, starting with shpat_.	shpat_0123456789abcdef...
Blog ID REQUIRED	The numeric ID of the blog from the address bar.	84726351921
Enabled	Leave on. When it is off, LazySEO skips this integration when publishing.	

Test the connection

- Right after you save, the window shows **Test the connection**: click **Run quick test**. You can test again at any time from the integration card with **Test**.
- Quick test** ("Check endpoint is reachable") reads the blog with your access token, so it checks the Store URL, the token and the Blog ID together. A wrong Blog ID fails with "Blog ... was not found in this store". Nothing is created.
- On the card's **Test** menu, **Full test** ("Send a test blog post (draft)") sends a real test article titled "[TEST] LazySEO Test Article" with the slug `lazyseo-test-article`. In Shopify it is created as a **hidden** (unpublished) blog post. Delete it from Blog posts once you have checked it.
- If a test fails, LazySEO shows the reason it got from your platform (for example a rejected key or a missing permission). Check it against the troubleshooting section below, click **Edit** on the card to fix the field, and test again.

What gets published

Title	The article title.
Content	The full article body as HTML. LazySEO converts the article's Markdown to HTML (headings, lists, links, tables, code), sends articles you edited in the LazySEO editor as the HTML you saved, and drops a first heading that only repeats the title, because Shopify shows the title itself.
Handle	The article slug is used as the post handle (the end of the URL).
Visibility	Visible (published). The Full test creates a hidden post.
Excerpt	The meta description is saved as the post excerpt (summary).
Image	The article banner, sent as an image URL for Shopify to import.

Tags	The article topics, as comma-separated tags.
Post URL	LazySEO records the public link: your store's primary domain + <code>/blogs/</code> + the blog handle + <code>/</code> + the article handle, for example <code>https://www.example.com/blogs/news/how-to-brew-coffee</code> . If that lookup fails, the article is still published, just without a recorded link. No link is recorded for the hidden Full test post.
Updates	If the same article is published again to the same integration, LazySEO updates the existing post instead of creating a second one. If you deleted the post in Shopify in the meantime, the update fails with HTTP 404 instead of creating a new post (see troubleshooting).
Unpublishing	Unpublishing an article inside LazySEO does not remove it from your site. Delete or unpublish the post in your platform if you need it gone.

Troubleshooting

HTTP 401 ("Shopify rejected the access token").

The **Admin API Access Token** is wrong, or the app was uninstalled. Copy the token again (or reinstall the app to get a new one).

HTTP 403 ("The Shopify app lacks permission").

The app is missing the `write_content` scope. Add `write_content` and `read_content` to the app's Admin API scopes and update the app.

HTTP 404 ("Blog ... was not found in this store" or "could not find the blog or article").

Check **Blog ID** (only the number) and **Store URL** (your `.myshopify.com` address, with `https://`).

Publishing an article again fails with HTTP 404, although the test passes.

The post was deleted in Shopify after LazySEO first published it. LazySEO tries to update that post and Shopify can't find it; for Shopify it does not yet create a new post automatically. Email support@lazyseo.io with the article's name and we will help.

The connection worked yesterday and fails today with HTTP 401.

You probably used a token from a Dev Dashboard app; those expire after 24 hours. Use a custom app created in the store admin, or email support@lazyseo.io.

Questions

Can I create a new custom app to get a token?

Not from the Shopify admin: since January 1, 2026, new apps are created in Shopify's Dev Dashboard, and their access tokens expire after 24 hours. Custom apps created in the admin before then keep working with a permanent token, and you can still change their scopes. If you don't have one, email support@lazyseo.io.

Which Shopify API version does LazySEO use?

The Admin REST API version 2026-07, which Shopify supports until July 2027.

5. SETUP GUIDE

Sanity

LazySEO writes each article as a document in your Sanity dataset through the Sanity HTTP API. The article body is converted to Portable Text, and the banner and inline images are uploaded to your Sanity asset library. Your own website front end then displays the documents, as it does for posts you write in Sanity Studio.

Online version: lazyseo.io/help/integrations/sanity

What you need

- A Sanity project with a dataset and a document type for blog posts (often called `post`).
- Permission to create API tokens in the project (in sanity.io/manage).
- Ideally, a developer who can check your post schema against the fields listed below.

Step by step

1. Find your project ID and dataset

- Go to sanity.io/manage and open your project. The **Project ID** is shown on the project page (8 characters, like `abc12345`).
- The dataset name is listed under **Datasets** (usually `production`).

2. Create an API token with write access

- In the project, go to **API** → **Tokens** and click **Add API token**.
- Name it `LazySEO` and choose the **Editor** permission (Viewer tokens cannot write).
- Save and copy the token. Sanity shows it only once.

3. Check your post schema

LazySEO creates documents with these fields. Fields that are not in your schema are still saved, but Studio shows them as "Unknown field" and your front end will not use them:

- `title` (string), `slug` (slug), `excerpt` (text: the meta description), `publishedAt` (datetime)
- `body`: an array of blocks (Portable Text) that also accepts `image`; with **Tables** set to Table block, it must also accept `table` (sanity-plugin-table)
- `mainImage` (image): the article banner
- `tags` (array of strings): the article topics

4. Open the integration form in LazySEO

- Open your project in LazySEO.
- In the left sidebar, under **Settings**, click **Integrations**.
- Click **Add Integration**.
- In the window that says "Choose the type of integration you want to add.", click **Sanity**.

5. Save the integration

Fill in the fields as described in the table below, leave **Enabled** switched on (disabled integrations are skipped during publish) and click **Add integration**.

Filling in the form in LazySEO

Field	What to enter	Example
Name REQUIRED	Any name that helps you recognise this connection. Only shown inside LazySEO.	Sanity production
Project ID REQUIRED	Your Sanity project ID from sanity.io/manage.	abc12345
Dataset REQUIRED	The dataset to write to. Pre-filled with production.	production
API Version REQUIRED	The Sanity API version, as a date. Keep the pre-filled value unless your developer asks for another one.	2024-01-01
API Token REQUIRED	The token you created. It must have write permissions (Editor or above).	sk...
Document Type REQUIRED	The name of your blog post schema type, exactly as in your Studio schema. Pre-filled with post.	post
Tables	Bulleted text — works with any schema turns each table row into a bullet point. Table block — needs sanity-plugin-table creates real table blocks; pick it only if your body field accepts a <code>table</code> type.	Bulleted text — works with any schema
Use blockquote style	Tick it to send quotes as <code>blockquote</code> blocks. Leave it off if your schema restricts block styles; quotes then stay normal paragraphs.	
Enabled	Leave on. When it is off, LazySEO skips this integration when publishing.	

Test the connection

- Right after you save, click **Run quick test** in the window, or test later from the integration card with **Test**.
- **Quick test** ("Check endpoint is reachable") checks read and write access without writing anything: it runs a read query for your **Document Type**, then a dry-run write that Sanity checks but does not save. A read-only (Viewer) token fails with "The Sanity token can read but not write".
- On the card's **Test** menu, **Full test** ("Send a test blog post (draft)") writes a real test document titled "[TEST] LazySEO Test Article" as a draft (ID `drafts.lazyseo-lazyseo-test-article`) and deletes it again straight away, so nothing is left in your dataset.
- If a test fails, LazySEO shows the reason it got from your platform (for example a rejected key or a missing permission). Check it against the troubleshooting section below, click **Edit** on the card to fix the field, and test again.

What gets published

Document ID	<code>lazyseo-</code> followed by the article slug, for example <code>lazyseo-how-to-brew-coffee</code> .
_type	The Document Type you entered.
title / slug	The article title and slug.

excerpt	The meta description.
body	The article converted to Portable Text: headings, paragraphs, lists, links, bold, italic and code. Inline images are uploaded to your asset library. YouTube embeds become a linked paragraph. Tables follow the Tables setting.
mainImage	The article banner, uploaded to your asset library. If the image can't be downloaded or uploaded, the document is saved without it.
publishedAt	The time the article was first published. Republishing keeps this first value.
tags	The article topics.
Status	Articles are written as published documents (ID <code>lazyseo-<slug></code>), not Studio drafts, so they are live as soon as your front end reads them. A draft status, used by the Full test, writes to <code>drafts.lazyseo-<slug></code> instead.
Post URL	Your front end decides the URL, so LazySEO does not record a published URL for Sanity.
Updates	Publishing the same article again updates only the fields LazySEO fills (title, slug, body, excerpt, mainImage, tags). Fields you add or edit in Studio, such as authors or categories, are kept; edits you make in Studio to LazySEO's own fields are replaced by LazySEO's version. If the document was deleted in Sanity in the meantime, LazySEO creates it again.
Unpublishing	Unpublishing an article inside LazySEO does not remove it from your site. Delete or unpublish the post in your platform if you need it gone.

Troubleshooting

HTTP 401.

The **API Token** is wrong or was deleted. Create a new token and paste it again.

HTTP 403 (for example "Insufficient permissions"), or the test says "The Sanity token can read but not write".

The token cannot write. Create a token with **Editor** permission.

HTTP 404 or "Dataset not found".

Check **Project ID** and **Dataset**.

Studio shows "Unknown field" on LazySEO documents.

Your schema does not define that field (often `excerpt` or `tags`). Add it to your post schema, or ignore it if you don't need it.

Studio shows unknown blocks of type `table` in the body.

Set **Tables** to **Bulleted text — works with any schema**, or add `sanity-plugin-table` to your schema.

Articles are in Sanity but not on my site.

Your front end must query the same **Document Type** and dataset. Check with your developer that it uses the fields listed above (for example `slug.current`).

An article I edited in the LazySEO editor lost its headings, lists or bold text in Sanity.

Articles saved from the LazySEO editor are stored as HTML, and the Sanity conversion currently keeps only their text for those. Articles as LazySEO generates them keep their formatting. Email support@lazyseo.io if

you need an edited article fixed.

Questions

Does LazySEO create drafts in Sanity?

No. Articles are written as normal (published) documents with an ID that starts with lazyseo-. Only the Full test writes a draft, and it deletes it again right away. Turn on Require manual approval in LazySEO if you want to review articles before they reach Sanity.

Will republishing overwrite changes I made in Sanity Studio?

Only in the fields LazySEO fills: title, slug, body, excerpt, mainImage and tags. Other fields, such as authors or categories you added in Studio, are kept, and publishedAt keeps the date of the first publish.

6. SETUP GUIDE

Dev.to

LazySEO can post articles to your account on DEV (dev.to) through the DEV API. DEV is a community site, not your own website: posts there build reach on DEV, but they don't add pages to your domain.

Online version: lazyseo.io/help/integrations/devto

What you need

- A DEV (dev.to) account.
- Keep in mind that LazySEO does not set a canonical URL on DEV posts. If the same article is also published on your own site, add the canonical URL to the DEV post yourself, or use DEV as the only destination for those articles.

Step by step

1. Generate a DEV API key

- On dev.to, open **Settings** and go to **Extensions**.
- Under **DEV Community API Keys**, enter a description such as `LazySEO`.
- Click **Generate API Key** and copy the key.

2. Open the integration form in LazySEO

- Open your project in LazySEO.
- In the left sidebar, under **Settings**, click **Integrations**.
- Click **Add Integration**.
- In the window that says "Choose the type of integration you want to add.", click **DEV.to**.

3. Save the integration

Fill in the fields as described in the table below, leave **Enabled** switched on (disabled integrations are skipped during publish) and click **Add integration**.

Filling in the form in LazySEO

Field	What to enter	Example
Name REQUIRED	Any name that helps you recognise this connection. Only shown inside LazySEO.	DEV account
API Key REQUIRED	The API key you generated on dev.to.	a1B2c3D4e5F6g7H8i9J0k1L2
Enabled	Leave on. When it is off, LazySEO skips this integration when publishing.	

Test the connection

- Right after you save, the window shows **Test the connection**: click **Run quick test**. You can test again at any time from the integration card with **Test**.

- **Quick test** ("Check endpoint is reachable") signs in to the DEV API with your API key and reads your own DEV account. Nothing is created.
- On the card's **Test** menu, **Full test** ("Send a test blog post (draft)") sends a real test article titled "[TEST] LazySEO Test Article" with the slug `lazyseo-test-article`. On DEV it is created as an **unpublished** draft. Delete it from your DEV dashboard once you have checked it.
- If a test fails, LazySEO shows the reason it got from your platform (for example a rejected key or a missing permission). Check it against the troubleshooting section below, click **Edit** on the card to fix the field, and test again.

What gets published

Title	The article title.
Body	The article's Markdown, sent as the post's Markdown body (<code>body_markdown</code>), which is the format DEV is written in. A first heading that only repeats the title is dropped, because DEV shows the title itself. Articles you edited in the LazySEO editor are stored as HTML and are passed through as HTML, which DEV renders inside the Markdown.
Status	Published. The Full test creates an unpublished draft.
Description	The meta description.
Tags	Up to four article topics, converted to DEV's tag format: lowercase, letters and numbers only, no duplicates. For example "SEO Strategy" becomes <code>seostrategy</code> and "Next.js" becomes <code>nextjs</code> .
Cover image	The article banner image URL.
Post URL	LazySEO saves the URL DEV returns as the article's published URL.
Updates	If the same article is published again to the same integration, LazySEO updates the existing post instead of creating a second one. If you deleted the post in DEV in the meantime, the update fails with HTTP 404 instead of creating a new post (see troubleshooting).
Unpublishing	Unpublishing an article inside LazySEO does not remove it from your site. Delete or unpublish the post in your platform if you need it gone.

Troubleshooting

HTTP 401 ("DEV.to rejected the API key").

The **API Key** is wrong or was revoked. Generate a new one in Settings → Extensions.

HTTP 422 with a message about a field.

DEV rejected part of the article; the message says which field. Tags are already converted to DEV's format, so this is usually about the title, the cover image URL or the body. Contact support@lazyseo.io if it keeps happening.

HTTP 429 ("rate limit reached").

DEV limits how many articles can be posted in a short time. Wait a few minutes and use **Retry** on the article.

Publishing an article again fails with HTTP 404.

The post was deleted on DEV after LazySEO first published it. LazySEO tries to update that post and DEV can't find it; for DEV it does not yet create a new post automatically. Email support@lazyseo.io with the

article's name and we will help.

Questions

Will posting to DEV help my own website rank?

Not directly. DEV posts live on dev.to. They can send readers to your site through links in the article, but they do not add pages to your domain.

7. FOR DEVELOPERS

Webhook

With a webhook, LazySEO sends every article as a JSON request to a URL you choose, and code on your side saves it as a post. This works with any custom-built site (Next.js, Laravel, Rails, Django, a static site generator with a build hook, and so on), but only if your site has an endpoint that receives the request. LazySEO cannot create that code for you; a webhook with nothing listening does nothing.

Online version: lazyseo.io/help/integrations/webhook

What you need

- A developer who can add an HTTPS endpoint to your site or backend.
- A place to store posts (database, headless CMS, Markdown files in a repository, and so on).
- Optional but recommended: a random secret to sign requests, for example generated with `openssl rand -hex 32`.

Step by step

1. Build the receiving endpoint

Accept the request described below, verify the signature if you set a secret, turn `content` into HTML (it is usually Markdown, see "The article content"), create or update the post, and reply with a 2xx status and a JSON body containing the post id and URL. You can start from the Node.js or PHP example further down.

2. Open the integration form in LazySEO

- Open your project in LazySEO.
- In the left sidebar, under **Settings**, click **Integrations**.
- Click **Add Integration**.
- In the window that says "Choose the type of integration you want to add.", click **Webhook**.

3. Save the integration

Fill in the fields as described in the table below, leave **Enabled** switched on (disabled integrations are skipped during publish) and click **Add integration**.

Filling in the form in LazySEO

Field	What to enter	Example
Name REQUIRED	Any name that helps you recognise this connection. Only shown inside LazySEO.	Production Webhook
Endpoint URL REQUIRED	The full URL of your endpoint. Use https:// in production.	<code>https://www.example.com/api/lazyseo-webhook</code>
Method REQUIRED	POST or PUT. Used for every article sent to the endpoint.	POST

Field	What to enter	Example
Headers (optional)	Extra headers sent with every request, added with Add header . Useful for an API key your endpoint expects. Header values other than common non-secret ones (such as Accept) are stored encrypted and hidden after saving.	Authorization → Bearer my-endpoint-key
Signing Secret (optional)	If set, LazySEO signs every request with this secret (see "Verify the signature"). Store the same value on your server.	3f9c2a... (64 random hex characters)
Enabled	Leave on. When it is off, LazySEO skips this integration when publishing.	

Test the connection

- Right after you save, click **Run quick test** in the window, or test later from the integration card with **Test**.
- **Quick test** sends a small POST request (always POST) with `"_test": true` and waits up to 10 seconds for a 2xx response. Your endpoint should answer it without creating a post.
- On the card's **Test** menu, **Full test** sends a test article titled "[TEST] LazySEO Test Article" with `"status": "draft"`, using your **Method**, so you can check that your endpoint creates the post correctly. Delete the test post afterwards.
- If a test fails, LazySEO shows the reason, for example "Your endpoint answered HTTP 404; it must reply with a 2xx status", a timeout, or a certificate or DNS problem.
- Under **Sync History** on the Integrations page, each successful publish shows the request LazySEO sent and your endpoint's response. A publish that still fails after all retries is listed with the error.

What gets published

Request	One JSON request per article to your Endpoint URL , with your Method .
Content	The article body in <code>content</code> : Markdown for articles as LazySEO generates them (it can contain embedded HTML blocks such as <code><figure></code> or a YouTube <code><iframe></code>), and HTML for articles edited and saved in the LazySEO editor. Render Markdown before you display it (see "The article content").
Images	The banner image is sent as a URL (<code>featuredImageUrl</code> and <code>bannerUrl</code>); download it if you want to host it yourself.
Status	<code>"published"</code> for real articles, <code>"draft"</code> for the Full test.
Updates	When an article is sent again after a successful publish, the payload also contains <code>externalId</code> : the id your endpoint returned the first time. If you no longer have that post, create it again (create-or-update), so republishing a deleted post brings it back.

The request LazySEO sends

Header	Value
<code>Content-Type</code>	<code>application/json</code> (unless you override it with a custom header)
Your custom headers	Every header from Headers (optional)

Header	Value
X-Signature	sha256= followed by the HMAC-SHA256 of the body, in hex. Only sent when Signing Secret (optional) is set.

Payload fields

Fields without a value are left out of the JSON (for example `metaDescription` when an article has none). Ignore fields you don't recognise, so new fields don't break your endpoint.

Field	Type	Description
<code>title</code>	string	Article title.
<code>slug</code>	string	URL-friendly slug, e.g. <code>how-to-brew-better-coffee-at-home</code> .
<code>content</code>	string	Full article body. Markdown (GitHub style, may contain embedded HTML blocks) for generated articles; HTML for articles edited in the LazySEO editor. The title is sent separately in <code>title</code> .
<code>metaDescription</code>	string, optional	SEO meta description.
<code>topics</code>	string[]	Topic labels for the article (may be empty).
<code>tags</code>	string[]	Same values as <code>topics</code> .
<code>status</code>	string	"published" for real articles, "draft" for the Full test.
<code>bannerUrl</code>	string, optional	URL of the article banner image.
<code>featuredImageUrl</code>	string, optional	Same image as <code>bannerUrl</code> .
<code>externalId</code>	string, optional	Only on updates: the id your endpoint returned when the article was first published, as a string (a numeric id such as <code>123</code> comes back as <code>"123"</code>).

Example payload

```
{
  "title": "How to Brew Better Coffee at Home",
  "slug": "how-to-brew-better-coffee-at-home",
  "content": "Great coffee starts with fresh beans...\n\n## Choose the right grind\n\n**French press:** coarse\n- **Espresso:** fine\n\n...",
  "metaDescription": "Learn how to brew better coffee at home with the right beans, grind size and water temperature.",
  "topics": ["Coffee Brewing", "Home Barista"],
  "tags": ["Coffee Brewing", "Home Barista"],
  "status": "published",
  "bannerUrl": "https://cdn.example.com/banners/how-to-brew-better-coffee.jpg",
  "featuredImageUrl": "https://cdn.example.com/banners/how-to-brew-better-coffee.jpg"
}
```

The article content: Markdown or HTML

Articles as LazySEO generates them are Markdown, so `content` usually arrives as Markdown. If you display it as it is, readers see `##` and `**` instead of headings and bold text. Render it with a Markdown library that allows embedded HTML, for example `marked` or `markdown-it` (with `html: true`) in Node.js, `Parsedown` or `league/commonmark` in PHP, or `Python-Markdown`.

Articles that were edited and saved in the LazySEO editor arrive as HTML. To tell them apart, check whether `content` starts with an HTML block tag such as `<p>`, `<h2>`, `<div>` or `<figure>`; LazySEO uses the same check. Both examples below do this. If your system stores Markdown, you can also keep generated articles as Markdown and only convert the HTML ones.

Test payloads

Quick test (always POST)

```
{
  "_test": true,
  "title": "LazySEO Webhook Test",
  "slug": "lazyseo-webhook-test",
  "content": "<p>This is a test payload from LazySEO to verify your webhook endpoint is working.</p>",
  "status": "draft"
}
```

Full test (your Method)

```
{
  "title": "[TEST] LazySEO Test Article",
  "slug": "lazyseo-test-article",
  "content": "<h1>LazySEO Test Article</h1><p>This is a <strong>test article</strong> sent by LazySEO ...</p>",
  "metaDescription": "Test article from LazySEO integration test.",
  "status": "draft",
  "tags": ["lazyseo-test"]
}
```

What your endpoint should return

- Any **2xx** status means success. Any other status (or no response) is a failure, and LazySEO stores your response body as the error message, so return a helpful one.
- To link the article to your post, reply with `Content-Type: application/json` and a body containing `id` (or `externalId`) and `url` (or `publishedUrl`). The `id` can be a string or a number; LazySEO stores it as a string.
- LazySEO saves `url` as the article's published URL. It is used to link to the live article and, if you have connected Google Search Console, to ask Google to index it. Without a `url`, the article is still marked as published.
- LazySEO sends `id` back as `externalId` the next time the same article is published, so you can update the post instead of creating a duplicate.

Example response

```
{
  "id": "123",
  "url": "https://www.example.com/blog/how-to-brew-better-coffee-at-home"
}
```

Verify the signature

When **Signing Secret (optional)** is set, LazySEO computes an HMAC-SHA256 of the exact request body with your secret and sends it, hex-encoded, as `X-Signature: sha256=<hex>`. Compute the same value over the **raw** body bytes (before any JSON parsing or re-serialising) and compare it with a constant-time comparison. Reject the request with 401 if it doesn't match.

Retries and timeouts

- If your endpoint fails, LazySEO tries the same article up to 3 times, a few seconds apart.
- For scheduled auto-publishing, LazySEO tries again later (at least an hour apart, up to 3 rounds). After that it stops and sends you a notification; you can publish again with **Retry** on the article.
- Reply quickly. LazySEO waits up to 30 seconds for a publish request and 10 seconds for the Quick test, then counts it as failed (and retries the publish). If your endpoint does slow work (image processing, a site rebuild), save the post, reply, and do the rest in the background.

- Because of retries, your endpoint can receive the same article more than once. Create or update by `slug` (or by `externalId`) instead of always inserting a new row.

Deletion requests (optional)

The webhook is designed to send a `DELETE` request to the same URL with the body `{"externalId": "..."}` (signed the same way) to remove a post. Handling it is optional: unpublishing an article inside LazySEO does not currently send it, so it does not remove the post from your site.

Example receiver: Node.js (Express)

server.js

```

// npm install express marked
const express = require('express');
const crypto = require('crypto');
const { marked } = require('marked');

const app = express();
const SECRET = process.env.LAZYSEO_SECRET; // same value as "Signing Secret" in LazySEO
const posts = new Map(); // replace with your database or CMS

// Keep the raw body: the signature is computed over the exact bytes LazySEO sent.
app.use('/lazyseo-webhook', express.raw({ type: 'application/json', limit: '10mb' }));

function hasValidSignature(rawBody, header) {
  if (!SECRET) return true; // no Signing Secret configured
  if (!header) return false;
  const expected = 'sha256=' + crypto.createHmac('sha256',
SECRET).update(rawBody).digest('hex');
  const a = Buffer.from(header);
  const b = Buffer.from(expected);
  return a.length === b.length && crypto.timingSafeEqual(a, b);
}

// "content" is Markdown for generated articles and HTML for articles edited
// in the LazySEO editor. Same check LazySEO uses: HTML starts with a block tag.
function contentToHtml(content) {
  const isHtml = /^s*(<!doctype|html|head|body|div|p|h[1-
6]|ul|ol|table|section|article|header|main|figure|blockquote)[s>]/i.test(content);
  return isHtml ? content : marked.parse(content); // keeps embedded HTML such as
<figure> or <iframe>
}

app.all('/lazyseo-webhook', (req, res) => {
  if (!hasValidSignature(req.body, req.get('X-Signature'))) {
    return res.status(401).json({ error: 'invalid signature' });
  }
  const data = JSON.parse(req.body.toString('utf8'));

  if (req.method === 'DELETE') {
    posts.delete(String(data.externalId));
    return res.json({ ok: true });
  }
  if (data._test) {
    return res.json({ ok: true }); // "Quick test" ping: don't create a post
  }
}

```

```
// Create, or update when LazySEO sends an article again (same slug / externalId)
const id = String(data.externalId || data.slug);
posts.set(id, {
  title: data.title,
  slug: data.slug,
  html: contentToHtml(data.content || ''),
  excerpt: data.metaDescription || '',
  image: data.featuredImageUrl || null,
  tags: data.tags || [],
  draft: data.status !== 'published',
});

// Return the post id and its public URL
res.json({ id, url: 'https://www.example.com/blog/' + data.slug });
});

app.listen(3000);
```

Example receiver: PHP

lazyseo-webhook.php

```

<?php
// lazyseo-webhook.php
// Needs a "posts" table with a UNIQUE index on "slug",
// and Parsedown for Markdown: composer require erusev/parsedown
require __DIR__ . '/vendor/autoload.php';
$secret = getenv('LAZYSEO_SECRET'); // same value as "Signing Secret" in LazySEO
$raw = file_get_contents('php://input');

header('Content-Type: application/json');

if ($secret) {
    $expected = 'sha256=' . hash_hmac('sha256', $raw, $secret);
    $given = $_SERVER['HTTP_X_SIGNATURE'] ?? '';
    if (!hash_equals($expected, $given)) {
        http_response_code(401);
        echo json_encode(['error' => 'invalid signature']);
        exit;
    }
}

$data = json_decode($raw, true);
if (!is_array($data)) {
    http_response_code(400);
    echo json_encode(['error' => 'invalid JSON']);
    exit;
}

$pdo = new PDO('mysql:host=localhost;dbname=blog;charset=utf8mb4', 'db_user',
'db_password');

if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {
    $pdo->prepare('DELETE FROM posts WHERE id = ?')->execute([$data['externalId'] ??
'']);
    echo json_encode(['ok' => true]);
    exit;
}

if (!empty($data['_test'])) {
    echo json_encode(['ok' => true]); // "Quick test" ping: don't create a post
    exit;
}

// "content" is Markdown for generated articles and HTML for articles edited in
// the LazySEO editor. HTML starts with a block tag; render everything else.
$content = (string) ($data['content'] ?? '');

```

```

$isHtml = preg_match('/^\s*<(!doctype|html|head|body|div|p|h[1-6]|ul|ol|table|section|article|header|main|figure|blockquote)[\s>]/i', $content);
$html = $isHtml ? $content : (new Parsedown())->text($content);

// Create the post, or update it if the slug already exists
$sql = 'INSERT INTO posts (slug, title, content, excerpt, image_url, status)
      VALUES (:slug, :title, :content, :excerpt, :image, :status)
      ON DUPLICATE KEY UPDATE title = VALUES(title), content = VALUES(content),
      excerpt = VALUES(excerpt), image_url = VALUES(image_url), status =
VALUES(status)';
$pdo->prepare($sql)->execute([
    ':slug'    => $data['slug'],
    ':title'   => $data['title'],
    ':content' => $html,
    ':excerpt' => $data['metaDescription'] ?? '',
    ':image'   => $data['featuredImageUrl'] ?? null,
    ':status'  => ($data['status'] ?? '') === 'published' ? 'published' : 'draft',
]);

$stmt = $pdo->prepare('SELECT id FROM posts WHERE slug = ?');
$stmt->execute([$data['slug']]);
$id = $stmt->fetchColumn();

echo json_encode([
    'id'    => $id,
    'url'   => 'https://www.example.com/blog/' . $data['slug'],
]);

```

Troubleshooting

Quick test fails.

Your endpoint must answer a POST with a 2xx status within 10 seconds, even when **Method** is PUT. Check that the URL is public (not localhost or behind a VPN), uses a valid HTTPS certificate, and that your endpoint accepts the `_test` payload.

Publishing fails with HTTP 401 or 403 from your endpoint.

Your endpoint rejected the request. Check the signature code (use the raw body) and any header your endpoint expects in **Headers (optional)**.

Publishing fails with "Timed out waiting for ...".

Your endpoint took longer than 30 seconds to answer. Reply as soon as the post is saved and do slow work in the background.

Publishing fails with HTTP 413.

Articles are long documents. Raise the request body size limit of your server or framework (for example `limit` in Express).

Posts on my site show ##, ** or – instead of headings, bold text and lists.

`content` is Markdown for generated articles. Render it to HTML with a Markdown library before you save or display it (see "The article content").

The same article was created twice.

Your endpoint received a retry, or the earlier reply had no `id`. Make it create-or-update by `slug` or `externalId`, and return the post `id` in the JSON reply.

LazySEO shows no link to the published article.

Return JSON with a `url` field and the `Content-Type: application/json` header.

Saving the integration says "Please re-enter the credentials (password, token or secret headers) when changing the destination URL."

For security, saved secrets are not reused for a new URL. Enter the **Signing Secret (optional)** and secret header values again together with the new **Endpoint URL**.

Questions

What is the difference between Webhook and Custom API?

With Webhook, LazySEO sends a fixed JSON payload (optionally signed) and you write the code that receives it. With Custom API, you point LazySEO at an API that already exists and shape the request body with a template so the API accepts it.

Can I use a webhook with Zapier, Make or n8n?

Yes, if the automation tool gives you a webhook URL that accepts JSON and answers with a 2xx status. The tool then has to create the post in your CMS.

8. FOR DEVELOPERS

Custom API

Custom API sends each article to an HTTP endpoint you already have, such as a headless CMS or your own backend, with the method, headers and JSON body you define. Use it when you can't or don't want to write a new receiver: you adapt the request to the API instead.

Online version: lazyseo.io/help/integrations/custom-api

What you need

- An API endpoint that creates a blog post from a JSON request.
- The way that API authenticates (for example a bearer token or an API key header).
- The JSON body the API expects for a new post.

Step by step

1. Write the body template

Start from the JSON your API expects and put placeholders where the article values go. Each placeholder is replaced with the value, escaped for use inside a JSON string, so always put placeholders inside double quotes.

2. Open the integration form in LazySEO

- Open your project in LazySEO.
- In the left sidebar, under **Settings**, click **Integrations**.
- Click **Add Integration**.
- In the window that says "Choose the type of integration you want to add.", click **Custom API**.

3. Save the integration

Fill in the fields as described in the table below, leave **Enabled** switched on (disabled integrations are skipped during publish) and click **Add integration**.

Filling in the form in LazySEO

Field	What to enter	Example
Name REQUIRED	Any name that helps you recognise this connection. Only shown inside LazySEO.	Headless CMS API
Endpoint URL REQUIRED	The URL that creates a post.	<code>https://api.example.com/articles</code>
Method REQUIRED	POST, PUT or PATCH. The same method is used for new articles and for updates (see "Updates" below).	POST
Headers (optional)	Authentication and any other headers your API needs, added with Add header . <code>Content-Type: application/json</code> is sent by default. Header values other than common non-secret ones are stored encrypted and hidden after saving.	Authorization → Bearer <code>sk_live_abc123</code>

Field	What to enter	Example
Body Template (optional)	The JSON body, with placeholders. If you leave it empty, LazySEO sends the same JSON payload as the Webhook .	<pre>{ "title": "{{title}}", "content": "{{content}}" }</pre>
Enabled	Leave on. When it is off, LazySEO skips this integration when publishing.	

Test the connection

- Right after you save, click **Run quick test** in the window, or test later from the integration card with **Test**.
- **Quick test** sends a **HEAD** request (or **GET** if your server answers 405) to the **Endpoint URL** with your headers, waits up to 10 seconds, and passes on any status below 400. Many APIs answer 401, 404 or 405 to a GET on a create endpoint, so a failed Quick test does not always mean publishing will fail; the reason LazySEO shows says which status your API returned.
- On the card's **Test** menu, **Full test** sends a real request with a test article titled "[TEST] LazySEO Test Article". Your template decides what is sent; there is no placeholder for the draft status, so the test post may be created as published in your system. Delete it afterwards.
- Under **Sync History** on the Integrations page, each successful publish shows the request LazySEO sent and your API's response. A publish that still fails after all retries is listed with the error.

What gets published

Request	One request per article to Endpoint URL with Method , Content-Type: application/json and your headers.
Body	Your Body Template (optional) with placeholders filled in, or the full webhook payload if the template is empty.
Content	The article body ({{content}}): Markdown for articles as LazySEO generates them, HTML for articles edited and saved in the LazySEO editor. If your API expects HTML, it has to render the Markdown (see "Markdown or HTML" below).
Timeout	LazySEO waits up to 30 seconds for your API to answer a publish request, then counts it as failed and retries.
Updates	If your API returned an id the first time, publishing the same article again sends the request to Endpoint URL + / + that id, with the same method and body. Choose a method your API accepts for both, or expect a new post each time. If the post was deleted in your system, that update request fails (usually HTTP 404); LazySEO does not fall back to creating a new post.

Placeholders

Placeholders are written as **{{name}}**. Anything else in double curly braces is left as it is.

Placeholder	Value
{{title}}	Article title

Placeholder	Value
<code>{{content}}</code>	Full article body: Markdown for generated articles, HTML for articles edited in the LazySEO editor
<code>{{slug}}</code>	URL slug
<code>{{metaDescription}}</code> <code>}</code>	Meta description (empty if none)
<code>{{featuredImageUrl}}</code> <code>}</code>	Banner image URL (empty if none)

Example: a typical REST blog API

An API that creates posts at `POST https://api.example.com/v1/posts`, authenticates with a bearer token and expects the post inside a `data` object:

Field	Value
Endpoint URL	<code>https://api.example.com/v1/posts</code>
Method	<code>POST</code>
Headers (optional)	<code>Authorization</code> → <code>Bearer YOUR_API_TOKEN</code>

Body Template (for an API that accepts Markdown)

```
{
  "data": {
    "title": "{{title}}",
    "slug": "{{slug}}",
    "body_markdown": "{{content}}",
    "excerpt": "{{metaDescription}}",
    "cover_image": "{{featuredImageUrl}}",
    "status": "published"
  }
}
```

Markdown or HTML

`{{content}}` is sent exactly as the article is stored in LazySEO. Generated articles are Markdown (GitHub style, possibly with embedded HTML blocks such as `<figure>` or a YouTube `<iframe>`); articles edited and saved in the LazySEO editor are HTML. Custom API does not convert between the two.

If your API stores Markdown, map `{{content}}` to its Markdown field. If it only accepts HTML, have it render Markdown with a library that allows embedded HTML (for example `marked` or `markdown-it` in Node.js, `Parsedown` in PHP), and pass content that already starts with an HTML tag such as `<p>` or `<h2>` through unchanged. If you can't change the API, use a [Webhook](#) receiver that converts the content and then calls your API.

Responses

- Any **2xx** status means success. Any other status is a failure, and the response body is shown as the error.
- If the response has `Content-Type: application/json`, LazySEO reads the post id from a top-level `id`, `externalId` or `postId` field, and the public link from `url`, `publishedUrl`, `link` or `permalink`. Fields nested deeper (for example inside `data`) are not read.
- The id can be a string or a number; LazySEO stores it as a string and uses it in the update URL.

Retries

Failed requests, including requests your API doesn't answer within 30 seconds, are retried like webhooks: up to 3 attempts a few seconds apart, and for scheduled auto-publishing up to 3 more rounds at least an hour apart. Make sure a repeated request doesn't create a duplicate post, for example by making `slug` unique in your API.

Troubleshooting

HTTP 400 or "invalid JSON" from your API.

Check that the template is valid JSON and every placeholder is inside double quotes, for example `"{{content}}"`. Compare your template with what your API expects; the error shows your API's response.

HTTP 401 or 403.

Check the authentication header name and value in **Headers (optional)**.

HTTP 404 or 405 when an article is published again.

Updates go to **Endpoint URL** + `/` + the returned id, with the same method. If your API uses a different method or path for updates, it will reject them. A 404 can also mean the post was deleted in your system since LazySEO first published it.

Quick test fails but Full test works.

That's normal for APIs that don't answer HEAD or GET on the create endpoint. Rely on the Full test.

Posts show `##`, `**` or `-` instead of headings, bold text and lists.

`{{content}}` is Markdown for generated articles. Send it to a Markdown field of your API, or render it to HTML on the API side (see "Markdown or HTML").

Publishing fails with "Timed out waiting for ...".

Your API took longer than 30 seconds to answer. Make it reply as soon as the post is saved.

Saving the integration says "Please re-enter the credentials (password, token or secret headers) when changing the destination URL."

For security, saved secret headers are not reused for a new URL. Enter the header values again together with the new **Endpoint URL**.

Questions

Can I send tags or topics through the body template?

Not with a placeholder: the template supports title, content, slug, metaDescription and featuredImageUrl. Leave the template empty to receive the full payload, which includes topics and tags.

Is {{content}} HTML or Markdown?

Markdown for articles as LazySEO generates them, and HTML for articles edited and saved in the LazySEO editor. LazySEO sends it as stored, without converting it.

9. SETUP GUIDE

Subdomain or no website

LazySEO writes articles and publishes them into a blog platform (a CMS) such as WordPress, Ghost or Webflow. It does not host a website for you. So if you only have a domain, or want a blog on a subdomain like `blog.yourdomain.com`, you first need a blog platform running there. The simplest route for most people is WordPress, which almost every web host can install in one click.

Online version: lazyseo.io/help/integrations/subdomain

What you need

- A domain name you control (where you bought it, or where its DNS is managed).
- Web hosting that offers one-click WordPress (LWS, Hostinger, OVHcloud, SiteGround and most cPanel hosts with Softaculous do), or an account on a hosted platform such as Ghost(Pro).

Step by step

1. Create the subdomain

In your hosting control panel, open the Domains or Subdomains section and create the subdomain, for example `blog.yourdomain.com`. Point it to its own folder (often suggested automatically, like `public_html/blog`). If your domain is registered with your host, the DNS record is usually created for you; otherwise see "DNS basics" below.

2. Install WordPress on it

Use your host's one-click installer (called Softaculous, an "Auto installer", "WordPress toolkit" or similar). Choose the subdomain as the install location, leave the directory field empty so WordPress sits at the root of the subdomain, and pick an admin username and a strong password.

3. Turn on HTTPS

Enable the free SSL certificate (often Let's Encrypt) for the subdomain in your hosting panel, then check that `https://blog.yourdomain.com` loads with a padlock. WordPress only offers application passwords over HTTPS, and LazySEO needs one.

4. Prepare WordPress

- In **Settings** → **General**, make sure both addresses start with `https://`.
- In **Settings** → **Permalinks**, choose **Post name** for readable post addresses. LazySEO works with any setting, including Plain.
- Pick a theme and add a link to the blog from your main site, so visitors and search engines can find it.

5. Connect it to LazySEO

Follow the [WordPress guide](#) and use `https://blog.yourdomain.com` as the **Site URL**.

Other ways to get a blog

- **Ghost(Pro)**: hosted Ghost, no server to manage. Connect your own domain or subdomain in Ghost, then follow the [Ghost guide](#).
- **Webflow**: a site plan with the CMS, connected to your domain in Webflow's publishing settings. Then follow the [Webflow guide](#).

- **Shopify:** if you sell online with Shopify, your store already includes a blog. See the [Shopify guide](#).

Subdomain or subfolder: which is better for SEO?

A subfolder (yourdomain.com/blog) keeps the blog under your main site, so links and authority the blog earns are generally consolidated with the rest of your domain. It is usually the safer choice when your main site's platform can host the blog itself, or when your developer can serve WordPress under /blog.

A subdomain (blog.yourdomain.com) also works fine. Google crawls and ranks it as part of the web like any other site; it may just take a little longer to build authority of its own. It is often the practical choice when the main site runs on a platform that can't host a blog. Either way, link to the blog from your main navigation and link back from articles to your main pages.

DNS basics

If your domain's DNS is managed somewhere other than your host (for example at your registrar or Cloudflare), you add one record there yourself:

- Your host or platform tells you which one to use and the exact value. Use one of them for the subdomain, not both.
- DNS changes usually work within minutes but can take up to 48 hours. Turn on HTTPS after the subdomain loads.

Record	Name / Host	Points to	Use when
A	blog	Your host's IP address (e.g. 203.0.113.10)	Your host gives you an IP address.
CNAME	blog	A hostname from your host or platform (e.g. yourname.ghost.io)	Your host or platform gives you a hostname.

Troubleshooting

The subdomain shows "site can't be reached".

The DNS record is missing or hasn't updated yet. Check the A or CNAME record and wait a little.

The browser warns that the connection is not private.

The SSL certificate isn't active for the subdomain yet. Enable it in your hosting panel (it can take a few minutes after DNS works).

There is no "Application Passwords" section in WordPress.

WordPress hides it when the site isn't on HTTPS. Turn on SSL and make sure the WordPress and site addresses in Settings → General start with https://.

Questions

Can LazySEO publish if I don't have a website at all?

LazySEO needs a blog platform to publish into. Install WordPress on your domain or a subdomain with your host, or create a Ghost(Pro) site, then connect it.

Is a subdomain bad for SEO?

No. A subfolder such as `yourdomain.com/blog` generally keeps authority consolidated with your main site, but a subdomain such as `blog.yourdomain.com` works fine too. Link it from your main site so search engines and visitors find it.

Which record do I need, A or CNAME?

Use an A record when your host gives you an IP address, and a CNAME record when it gives you a hostname. Your host or platform tells you which one.