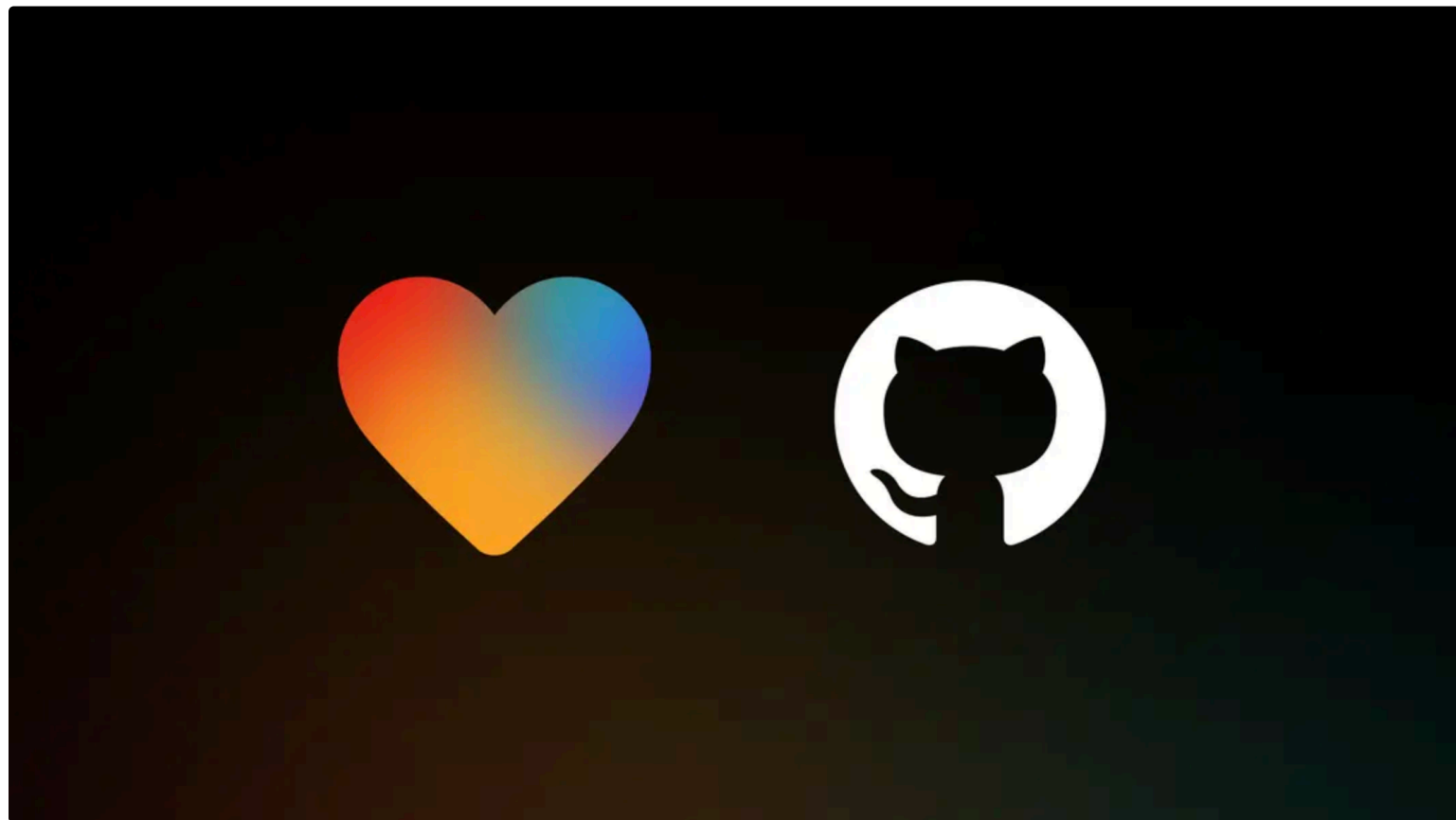


[Integrations](#)

## GitHub Integration

Learn how to connect and sync your projects with GitHub repositories



### What is GitHub

Integrating GitHub into your Lovable project ensures you have full version control, collaboration tools, and code portability throughout your app's lifecycle.

#### TL;DR:

- **Git** is a version control system that tracks changes in your code.
- **GitHub** is the industry-standard platform for hosting Git repositories and tracking code changes, which makes it essential for both solo builders and teams using Lovable.
- By linking your project to GitHub, every change in your Lovable app is backed up in a Git repository in real time. This means you get a complete history of your code, the ability to collaborate with developers in familiar workflows, and the freedom to host or deploy your app outside of Lovable if needed.
- **GitHub integration** brings transparency, safety, and flexibility to your Lovable development process.

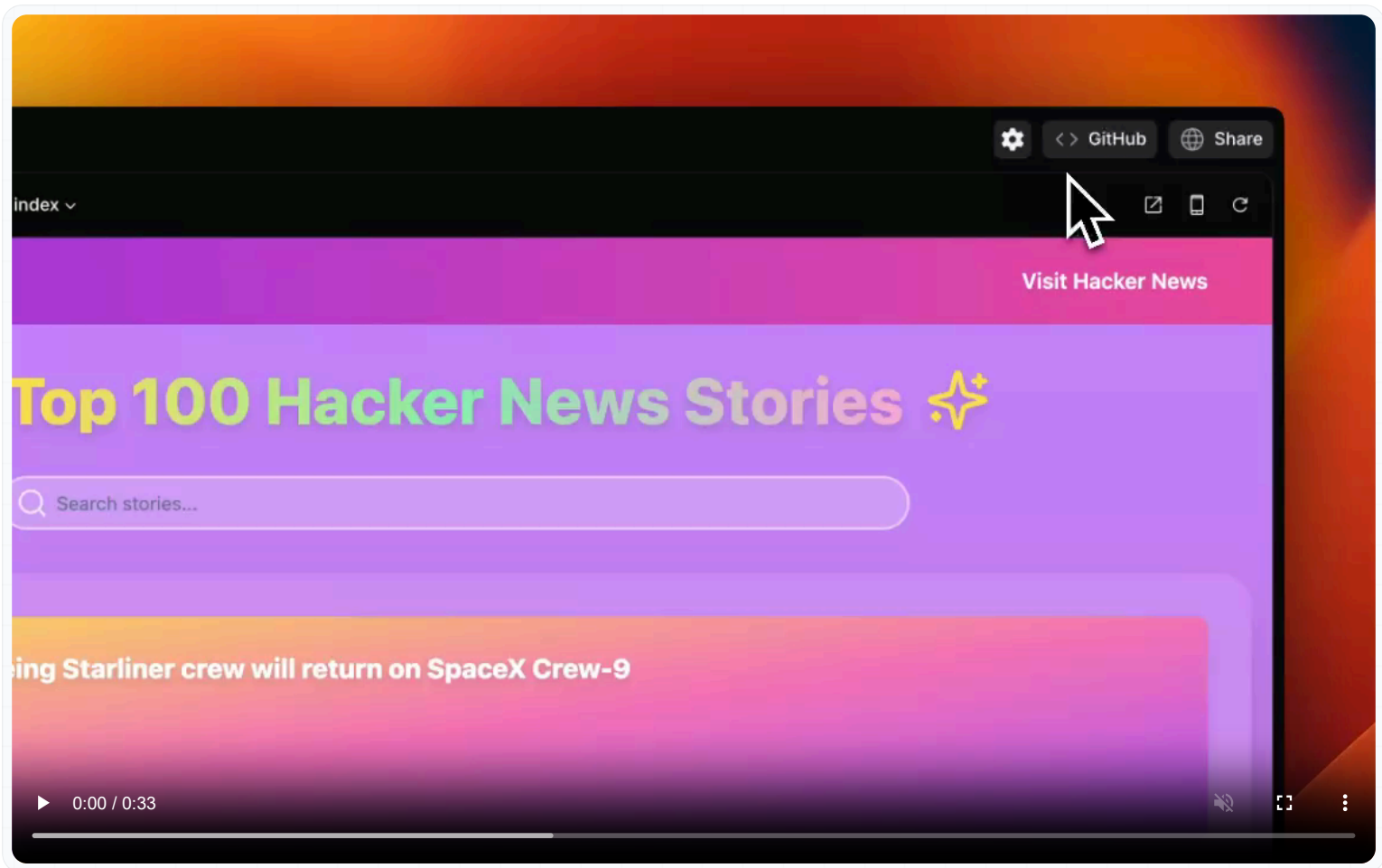
### Key Benefits of GitHub Integration:

Lovable backs every project with a Git repository, making it easy for developers to push commits directly to GitHub. This means:

- **Version History & Backup:** All your code is tracked with Git, so you can review changes or roll back to any previous state. Every commit is saved to GitHub, providing an external backup of your project

- **Workflow Integration:** Use GitHub's tools (branches, pull requests, issues, CI/CD, etc.) alongside Lovable. For example, you can run tests or deploy your app via GitHub Actions when code is merged, all while continuing to use Lovable's editor and AI to develop features.
- **Deployment Flexibility:** Connecting to GitHub gives you the option to host your app on your own infrastructure or platform of choice. You're not locked into Lovable's hosting – you can export the code and deploy it anywhere. (Lovable will still sync changes with GitHub, so you can continue editing in Lovable even if you host the app elsewhere.)

## Connecting Lovable to Your GitHub Account (Setup Guide)



To start using GitHub with Lovable, you need to connect your GitHub account and create a repository for your project. Here's how to set it up:

- 1 **Initiate GitHub Connection:**  
In the Lovable editor, click on **GitHub → Connect to GitHub** (usually found in the top-right corner of the project editor). This will begin the authorization process.
- 2 **Authorize Lovable on GitHub:**  
You'll be redirected to GitHub to authorize the Lovable GitHub App. Sign in to GitHub if prompted. When asked to grant access, choose either **All repositories** or **Only select repositories**. Granting access to all repos is easiest, but selecting specific repos is fine too – either way, Lovable will be able to create a new repo for your project in your account or organization
- 3 **Select GitHub Account/Org**  
If your GitHub user belongs to organizations, you may need to confirm which GitHub account or organization to install the Lovable app to. You can create the project's repo under your personal GitHub or any organization where you have permission.

Once GitHub is connected, return to your Lovable project. Click the **Create Repository** button (located in the top-right of the Lovable editor interface). Lovable will then create a new GitHub repository for this project and push the current project code to that repo. This initial push might take a few seconds. After this step, your Lovable project's code is now on GitHub.

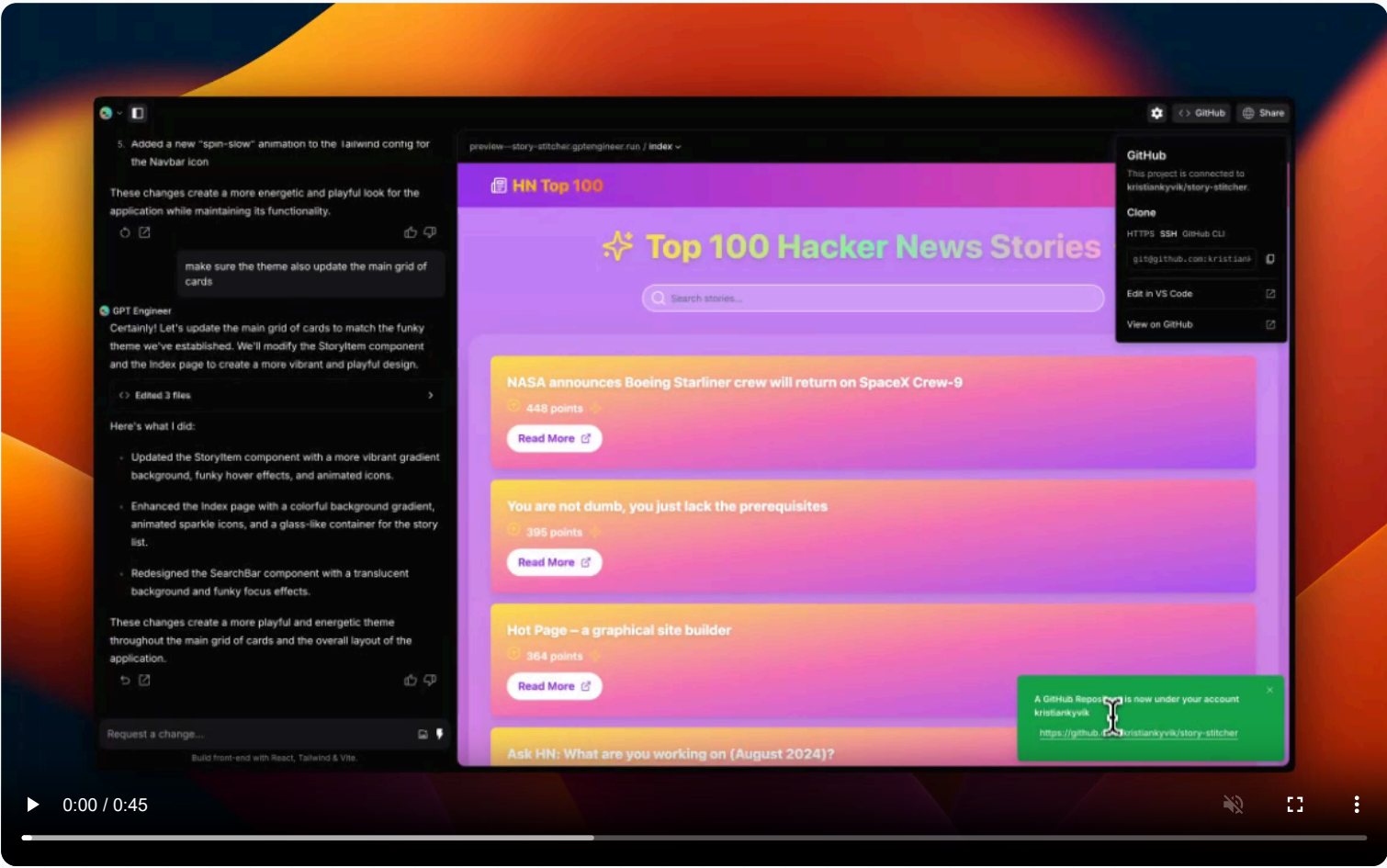
## 5 Verify the Link

GitHub and navigate to your account (or org) – you should see the new repository with your project's name. It will contain all the code of your Lovable app. In Lovable, you might also see a confirmation or the GitHub repository name linked. From now on, any changes in Lovable will sync to this repo and vice versa.

💡 If you plan to connect multiple Lovable projects to GitHub, you can repeat the above for each project (each will create its own repo). Each Lovable project can have one linked GitHub repo. Remember that your Lovable account remains linked to whichever GitHub login you authorized until you disconnect it. If you ever need to switch the GitHub account, you'd have to disconnect the current one in your Lovable settings and then connect a new account.

## How Syncing Works (Lovable ↔ GitHub)

Once your project is connected to GitHub, Lovable ensures that code changes stay in sync between the Lovable editor and your GitHub repository. The integration is **bidirectional**, but it follows some specific rules and behaviors:



- Default Branch Sync Only:** Lovable currently tracks **only the default branch** of your GitHub repository. This is typically the `main` branch (or `master`, depending on your repo). If you push commits to other branches on GitHub, those changes will **not** appear in your Lovable project until they are merged into the default branch. For example, if you're working on a `feature/Login` branch, Lovable won't show those updates in the editor until you merge `feature/Login` into `main` on GitHub. Similarly, Lovable will only push its changes to the default

>

project. You don't need to manually refresh or import anything; the editor will reflect the new code typically within a few seconds of the push. This allows developers to work in their own environment and see the results in Lovable instantly.


- **Pushing Changes from Lovable:** If you make changes in Lovable (for example, editing code via Lovable's Dev Mode or having the AI generate new code), those changes are also saved to the Git repository backing your project. Lovable will push these commits to GitHub automatically, updating the repository. In practice, you can treat Lovable as another git client – when the AI or you modify code in the Lovable editor, it's as if you committed and pushed to GitHub. For instance, adding a new page or editing a file in Lovable will result in a commit (with the changes) appearing in the GitHub repo's commit history. This ensures both Lovable and GitHub always have the same latest code.
- **No Manual Pull/Push Needed:** The sync is continuous. You do not have to manually "pull" updates into Lovable or "push" updates to GitHub – Lovable handles it. Pushing code to GitHub triggers Lovable to fetch the changes, and editing in Lovable triggers a push to GitHub behind the scenes. Your focus can be on building, not on syncing. If you ever suspect a change didn't sync, a quick refresh of the Lovable editor or checking the GitHub repo can confirm the latest state (but in general it's automatic).
- **Conflict Handling:** Because both Lovable and GitHub can change the code, there's a chance of a git conflict if changes happen on the same file at the same time. In practice this is rare, but if a conflict does occur, you would resolve it just like any Git conflict. For example, you might need to go to the GitHub repo and manually merge or fix conflicting changes, then commit. Lovable will then pull the resolved code. To avoid conflicts, it's best to coordinate who is editing or use branches for larger changes (merging to main when ready).

In summary, the GitHub integration means your Lovable project's code and the GitHub repository are effectively one and the same source, mirrored in real time. You can confidently switch between the Lovable editor and other development tools knowing both will stay updated

## Importing an Existing GitHub Repository into Lovable

What if you already have a codebase on GitHub and want to use Lovable with it? Currently, **Lovable does not yet support directly importing or linking to an existing external GitHub repo as a new project**. In other words, you can't point Lovable at an existing repository and have it pull in all that code automatically (as of now). However, there are ways to bring your existing code into Lovable:

- **Manual Import via a New Repo:** One workaround is to create a new Lovable project and connect it to GitHub (following the setup steps above), which gives you an empty repository (or a repository with the starter code from Lovable). Then, you can **manually add your existing code to that repository**. For example, you could clone the new GitHub repo to your local machine, copy in the files from your old project, commit and push. Once you push your existing code into the repo's default branch, Lovable will sync those files and you'll see them in the Lovable editor. This effectively migrates your code into a Lovable project.
- **Copy-Paste Smaller Snippets:** If your project is small, you might also start a Lovable project and copy code pieces (like individual files or functions) into Lovable via the editor or the AI (for instance, by pasting code into Lovable's code editor or prompting the AI with your code). This is less ideal for large projects but can work for minor modules.
- **Planned Import Feature:** The Lovable team is aware that direct repo import is a desired feature, and it's something under consideration for future updates. Keep an eye on Lovable's roadmap or announcements for any feature that allows importing an existing repo more directly.

 Importing an existing codebase may involve setting up any necessary configuration in Lovable (e.g., environment variables, dependencies) to mirror the original project's setup. After you've brought in the code, test the project within Lovable to ensure everything runs. From then on, you can use Lovable's AI features on your code and still sync back to GitHub.

## Developing with GitHub and Lovable in Parallel

Once connected, you can build and edit your application using GitHub (or your local development environment) and Lovable simultaneously. This is great for developers who want to use their own tools or for teams where some members prefer coding by hand. Here are some ways to work effectively with the integration:

- **Viewing the Repository:** In the Lovable editor, you'll have an option to view the GitHub repository (often via a **"View on GitHub"** link or button). Clicking this will take you straight to the repository page on GitHub. From there, you can browse the code, open files, and even edit directly on GitHub if needed.



ⓘ A developer on your team could clone the repo, make extensive changes with full IDE support, run tests locally, and then push commits to GitHub. Because of the integration, those commits will sync back to Lovable automatically, so the Lovable project will update to include the changes.

- **Normal GitHub Workflows:** You can leverage all of GitHub's collaboration features on your project's repo: create **branches** for new features, open **pull requests** to review code before merging, file **issues** and use project boards to track tasks, etc.. Lovable doesn't interfere with these; it simply watches the default branch for actual code changes. For instance, you might do all your work for a new feature on a `feature` branch, push it to GitHub, get it reviewed via a pull request, and then merge it to `main`. Once merged, Lovable will pull in those changes so the running app updates. This way, teams can enforce code reviews and use GitHub's project management, while still benefiting from Lovable's rapid development capabilities.
- **Combining AI and Manual Coding:** You can continue to use Lovable's AI assistant to generate code or new components, and then inspect or fine-tune that code via GitHub. The integration allows a hybrid workflow: for example, a non-technical founder might use Lovable to scaffold a feature with AI, and a developer can then clean up or optimize that code in the GitHub repo. Both sets of changes merge seamlessly.

ⓘ For example, use **GitHub Actions** to run tests automatically on each pull request, or to deploy the app to a hosting service whenever `main` gets updated

- **Remember:** even while doing heavy development through GitHub, you can always go back to the Lovable editor to use the visual tools or AI chat for assistance. The codebase is the same, so any changes you made by hand will be present in Lovable. This integration makes Lovable a part of your toolchain rather than an isolated environment.

## Documenting Changes and Rolling Back

Working with GitHub encourages good software development practices like documenting changes and managing versions. Here are some tips on managing your code changes and reverting when necessary:

- **Use Meaningful Commit Messages:** Whenever you commit changes (especially when committing via a local IDE or GitHub's interface), write a clear commit message describing *what* you changed and *why*. For example, `"Fix signup form validation bug"` is much more useful than `"update code"`. Good commit messages help team members (and your future self) understand the history of the project. They serve as documentation for the evolution of your app. If Lovable's AI made a lot of changes in one go, consider breaking them into logical commits with messages for each major change, if you're doing it manually. If commits are being generated automatically, you can always squash or edit them later on GitHub for clarity.
- **Track Versions in Lovable:** Lovable provides a version history in the editor (similar to Google Docs revision history) that keeps track of changes made through the platform. You can access the version history panel to see previous states of your project and even restore an earlier version with one click. This is effectively a "rollback" feature built on top of Git – if something goes wrong after an AI-generated change or an edit, you can revert to a known good state. It's a quick way for non-developers to undo mistakes without using GitHub directly.
- **Rolling Back with Git or Lovable (Reverts):** Since your project is under Git, you can perform rollbacks by reverting commits on GitHub as well. For example, if a certain commit introduced an issue, you (or a developer) can click "Revert" on GitHub for that commit, or use `git revert` locally and push. This will create a new commit that undoes the changes of the bad commit. Lovable will sync this revert commit and your project will go back to the previous behavior. Always test your app after a revert to ensure the issue is resolved and no other functionality is affected.
- **Use Branches for Big Changes:** A good strategy to reduce the need for rollbacks is to use feature branches for major changes. You might develop a big feature on a separate branch and only merge it to the main branch (which Lovable uses) after it's fully tested. This way, your main branch (and the live app in Lovable) stays stable, and you only introduce changes when they are ready. If something on the feature branch isn't working, it doesn't affect your production code in Lovable until merged.
- **Documentation of Changes:** If you are collaborating with others or handing off the project, consider maintaining a brief **changelog** or using GitHub's release notes. While commit messages are the primary log, summarizing changes in human-readable form (e.g., "Implemented user authentication, updated UI for profile page, fixed payment bug") can be helpful for non-technical stakeholders to follow progress. You can do this in a Markdown file in the repo or in GitHub Releases.
- **Lovable's Rollback Feature:** Lovable automatically backs up your project, so if you made a mistake, you can use the built-in rollback in the Lovable interface to undo accidental changes. This is essentially using Git version control under the hood. It ensures that even non-technical

>

If a published update causes issues, you have clear records of what changed and multiple ways to revert or fix forward. This significantly de-risks the development process.

## Troubleshooting & FAQ (GitHub Sync)

Below are some common questions and issues that users encounter with the GitHub integration, along with guidance on how to resolve them. Click on a question to view the answer.

### ▼ Why aren't my commits showing up in Lovable?

**Possible causes:** The most common reason is that the commits were made on a branch that Lovable isn't tracking. Remember, Lovable only syncs with the repository's **default branch** (usually `main` or `master`). If you committed to a different branch and haven't merged those changes into the default branch, Lovable will not see them. To fix this, merge the branch into the default branch (or change the default branch – see the next question). Once the commit is on the default branch, Lovable should pull it in automatically.

Another possibility is an issue with the integration link. For instance, if you **renamed the GitHub repository or changed your GitHub username** after connecting, the webhook/connection might be broken. In such cases, reconnecting your GitHub account to Lovable can re-establish the link. Make sure the Lovable GitHub App still has access to the repo (check your GitHub settings → Applications). If it was removed or lost access, re-authorize it and then try pushing the commit again.

Also, double-check that the commit was successfully pushed to GitHub (not just committed locally). If you haven't run `git push` (in case of local development), the commit won't be on GitHub for Lovable to detect.

If your commits **are on the default branch** and you still don't see updates in Lovable, try manually refreshing the project in the Lovable editor. In rare cases, there might be a slight delay or a hiccup in the webhook. Usually, though, commits appear almost instantly when everything is set up correctly.

### ▼ How do I switch the default branch that Lovable syncs with

Lovable will follow whatever branch is set as the **default** in your GitHub repository settings. By default this is `main` (for new GitHub repos) unless you changed it. If you want Lovable to track a different branch (say you want to use a `development` branch as the main source), you need to change the default branch on GitHub:

1. **On GitHub:** Go to your repository's **Settings** > **Branches** > **Default branch**. Choose the branch you want to set as the new default, and confirm the change. (If the branch doesn't exist yet, you'll need to create it and perhaps populate it with the current code.)
2. **After Changing:** Once you set a new default, Lovable should start treating that branch as the one to sync. Since Lovable only tracks the default branch, changing it effectively points Lovable to the new branch. Give it a moment to fetch the code from the new default branch. All future commits to that branch will sync to Lovable, and edits in Lovable will be pushed to that branch.
3. **Verify in Lovable:** Check the Lovable editor to ensure it's now showing the code from the new default branch. You might see an update notification or you might simply notice that any differences (if the branch had different code) are now reflected. If it doesn't seem to update, try disconnecting and reconnecting the GitHub integration in Lovable (as a last resort). But typically, this isn't required — it should follow the default automatically.
4. **Merge Old Changes:** A crucial step is to make sure any important commits from the old default branch get merged into the new default. Changing the default branch on GitHub **does not** automatically merge the content. If you switched default branches because you want to shift development to a new line of work, ensure that the new branch has all the latest code. You may need to merge the previous main into the new branch so nothing is lost. After merging, Lovable will show the combined changes.

Keep in mind that all collaborators will now be pushing to the new default branch by default. Update your team or documentation to avoid confusion. Lovable will now ignore the formerly default branch (it will only pay attention to the new one). If needed, you can always switch back by re-designating the old branch as default in GitHub settings.

>

install the Lovable GitHub App on your repo which sets up the communication. So, **when you push a new commit to the default branch**, GitHub notifies the Lovable service (via a webhook event). Lovable then automatically pulls the latest changes into your project. This all happens in the background within seconds. That's why, as soon as you push code to GitHub, you'll usually see your Lovable project update almost immediately with no action needed on your part.


On the flip side, when you make changes in the Lovable editor, Lovable uses the GitHub connection to push those commits to your repository. This might happen through the Lovable backend using GitHub's API – for you it's seamless: you'll just notice that new commits (with appropriate messages) show up in your GitHub repo whenever you change something in Lovable.

In summary, Lovable is constantly "listening" for GitHub commits and "talking" to GitHub when changes occur in the editor. It's a two-way sync facilitated by the integration you set up during the connection. You don't have to manage any of this process; just know that behind the scenes, webhooks/API calls are keeping everything aligned. If the sync ever seems to stop, it could be due to a broken connection (in which case reconnecting the integration or re-authorizing the GitHub App can re-enable the webhook).

#### ▼ Can I host my Lovable app myself after connecting to GitHub?

Yes. Integrating with GitHub gives you full access to your code, which means you can deploy or host the application outside of Lovable if you want. By default, Lovable provides hosting for your project (when you click "Publish" in Lovable, it deploys your app to Lovable's cloud). This doesn't change just because you linked GitHub – you can continue to use Lovable's hosting **and** have the code on GitHub at the same time.

However, if you prefer to host the app on your own server or another platform, you can do so by using the code from the GitHub repo. For example, you could take the repository and deploy it to Vercel, Heroku, AWS, or any environment of your choice. Since the code is a standard web app codebase (Lovable projects are typically full-stack JavaScript/TypeScript apps), you can set it up like any project.

 **Even if you host elsewhere, you can still use Lovable to develop.** You might treat Lovable as a powerful visual editor/AI assistant, and when you're ready to update your self-hosted app, you push commits to GitHub (via Lovable). Your external deployment pipeline (if set up) would then deploy those new commits. In this scenario, Lovable is for building and GitHub is the single source of truth for both Lovable and your own hosting.

Do note that if you do host the app yourself outside of Lovable, you'll need to manage environment variables, secrets, database connections, and other config in your new environment just as you would for any app. The Lovable integration with services like Supabase or Stripe will still be in your code – those continue to work as long as the appropriate config is in place on the new host.

By following this guide and utilizing GitHub with Lovable, you'll combine the speed of AI-assisted development with the reliability and control of traditional software practices. This integration is designed to be approachable for non-technical users (you get a safe backup and easy way to hand off code) while providing all the power that developers expect in a modern toolchain.

Happy coding!

Was this page helpful?

 Yes

 No

[← Integrations](#)

[Supabase Integration →](#)