# **Contributing**

We would ❤️ for you to contribute to Appwrite and help make it better! We want your experience contributing to Appwrite to be fun, enjoyable, and educational for anyone and everyone. All contributions are welcome, including issues and new docs, as well as updates and tweaks, blog posts, workshops, and more.

## **How to Start?**

If you are worried about or don’t know where to start, check out the next section that explains what kind of help is needed and how you can get involved. You can reach out with any questions on our [Discord](https://appwrite.io/discord) server. You can also submit an issue, and a maintainer can guide you!

## **Repo Structure**

├── src

│ ├── lib // All non-route components, accessible over "import ... from '$lib'"

│ │ ├── components // Re-usable components

│ │ ├── elements // Re-usable elements

│ │ ├── layout // Global components for the layout (Nav/Content/Container)

│ │ └── stores // Global stores (state management)

│ └─── routes

│ ├── console // Routes that need authentication

│ │ ├──[project]

│ │ │ ├── database // Database Service

│ │ │ │ ├── [collection] // Nested Route for the collection "/console/[PROJECT\_ID]/database/[COLLECTION\_ID]"

│ │ │ │ ├── \_create.svelte // Component to Create collections

│ │ │ │ └── index.svelte // Entrypoint for "/console/[PROJECT\_ID]/database"

│ │ │ ├── storage // Storage Service "/console/[PROJECT]/storage"

│ │ │ └── auth // Users Service "/console/[PROJECT]/auth"

│ │ └──...

│ ├── login.svelte // Component for Login "/console/login"

│ └── register.svelte // Component for Register "/console/register"

├── build // Compiled application

└── static // Static assets

## **Development**

Once you've created a project and installed dependencies with npm install, start a development server:

npm run dev

### **Build**

npm run build

You can preview the built app with npm run preview, regardless of whether you installed an adapter. This should *not* be used to serve your app in production.

### **Tests**

npm test

This will run tests in the tests/ directory.

### **Format**

Code should be the same everywhere; for this reason, a code-formatter is mandatory.

npm run format

### **Linter**

npm run lint

### **Diagnostics**

Diagnostic tool that checks for following:

* Unused CSS
* Svelte A11y hints
* TypeScript compiler errors

npm run check

## **Submit a Pull Request 🚀**

The branch naming convention is as follows

TYPE-ISSUE\_ID-DESCRIPTION

example:

doc-548-submit-a-pull-request-section-to-contribution-guide

When TYPE can be:

* feat - is a new feature
* doc - documentation only changes
* cicd - changes related to CI/CD system
* fix - a bug fix
* refactor - code change that neither fixes a bug nor adds a feature

All PRs must include a commit message with a description of the changes made!

Start by forking the project and use the git clone command to download the repository to your computer. A standard procedure for working on an issue would be to:

1. Before creating a new branch, pull the changes from upstream to make sure your default branch is up to date.

$ git pull

1. Create a new branch from the default branch. For example doc-548-submit-a-pull-request-section-to-contribution-guide

$ git checkout -b [name\_of\_your\_new\_branch]

1. Work - commit - repeat ( be sure to be in your branch )
2. Push changes to GitHub

$ git push origin [name\_of\_your\_new\_branch]

1. Submit your changes for review. If you go to your repository on GitHub, you'll see a Compare & pull request button. Click on that button.
2. Start a Pull Request (PR) by clicking on Create pull request. Make sure to update the PR description following the template provided.
3. Wait for a code review.
4. If you need to make changes based on feedback, make sure to re-request a review from your reviewer after you've made the necessary changes.



1. After approval, your PR will be merged.
2. You can delete your branch after it has been merged.

## **Guidelines**

### **Consistency**

Before committing, always make sure to run all available tools to improve the codebase:

* Formatter
	+ npm run format
* Tests
	+ npm test
* Diagnostics
	+ npm run check

### **Performance**

Page load times are a key consideration for users of all browsers and device types.

There are some general things we can do in front-end development:

* Minimize HTTP requests
* Minimize blocking – content should be readable before client-side processing
* Lazy load “supplementary” content (especially images)

### **Don't Repeat Yourself (DRY)**

If you repeat anything that has already been defined in code, refactor it so that it only ever has one representation in the codebase.

If you stick to this principle, you will ensure that you will only ever need to change one implementation of a feature without worrying about the need to change any other part of the code.

### **Separation of concerns**

Separate *structure* from *presentation* from *behavior* to aid maintainability and understanding.

* Keep CSS (presentation), JS (behavior), and HTML (structure) in the same respective Svelte component
* Avoid writing inline CSS or Javascript in HTML
* Avoid writing CSS or HTML in Javascript
* Don't choose HTML elements to imply style
* Where appropriate, use CSS or Svelte rather than Javascript for animations and transitions
* Try to use templates when defining markup in Javascript

### **Write code to be read**

Follow the principles of ['Keep It Simple, Stupid'](http://en.wikipedia.org/wiki/KISS_principle) (KISS); hard-to-read or obfuscated code is difficult to maintain and debug. Don't be too clever; write code to be read.

### **Identify technical debt**

Use code comment annotations (@todo) to mark parts of your code that require further work. This will allow the measurement and management of technical debt.

Don't use @fixme (which defines things that are broken) — you shouldn't be committing broken code to the repo.

### **Dependencies**

Please avoid introducing new dependencies to Appwrite without consulting the team. New dependencies can be very helpful, but they also introduce new security and privacy issues, complexity, and impact total docker image size.

Adding a new dependency should contribute vital value to the product with minimum possible risk.

## **Introducing New Features**

We would 💖 you to contribute to Appwrite, but we would also like to make sure that Appwrite is as great as possible and remains loyal to its vision and mission statement 🙏.

For us to find the right balance, please open an issue explaining your ideas before introducing a new pull request.

This will allow the Appwrite community to have a sufficient discussion about the new feature value and how it aligns with the product roadmap and vision.

This is also important for Appwrite maintainers to discuss the feature's design and architecture to help the contribution fit organically in the Appwrite codebase. Some bigger features might need to go through our [RFC process](https://github.com/appwrite/rfc).

## **Other Ways to Help**

Pull requests are great, but there are many other areas where you can help Appwrite.

### **Blogging & Speaking**

Blogging, speaking about, or creating tutorials about one of Appwrite’s many features are very effective at helping Appwrite. Mention [@appwrite](https://twitter.com/appwrite) on Twitter and/or email team@appwrite.io so we can give you pointers and tips on the best way to spread the word by promoting your content on the different Appwrite communication channels. Please add your blog posts and videos of talks to our [Awesome Appwrite](https://github.com/appwrite/awesome-appwrite) repo on GitHub.

### **Presenting at Meetups**

Presenting at meetups and conferences about your Appwrite projects is another excellent means to get the word out about Appwrite. Your unique challenges and successes in building things with Appwrite can provide great speaking material. We’d love to review your talk abstract/CFP, so get in touch with us if you’d like some help!

### **Sending Feedback & Reporting Bugs**

Sending feedback is a great way for us to better understand your different use cases of Appwrite. If you had any issues, bugs, or want to share about your experience, feel free to do so on our GitHub issues page or on our [Discord channel](https://discord.gg/GSeTUeA).

### **Submitting New Ideas**

If you think Appwrite could use a new feature, please open an issue on our GitHub repository, stating as much information as you can think about your new idea and its implications. We would also use this issue to gather more information, get more feedback from the community, and have a proper discussion about the new feature.

### **Improving Documentation**

Submitting documentation updates, enhancements, designs, or bug fixes help us improve our documentation. Spelling or grammar fixes are also very much appreciated.

### **Helping Someone**

 Please consider searching for Appwrite on Discord, GitHub, or StackOverflow and helping someone else who needs help.

## **Code of Conduct**

Help us keep Appwrite open and inclusive. Please read and follow our [Code of Conduct](https://github.com/appwrite/.github/blob/main/CODE_OF_CONDUCT.md).