Resources:

**Git commands for beginners:**

<https://www.atlassian.com/git/tutorials/syncing>

**Git cheat sheet:**

<https://education.github.com/git-cheat-sheet-education.pdf>

<https://neuroplausible.com/github>

**10 ways to contribute to an open-source community:**

*(without contributing code)*

1. **Provide reports** of what you liked and what you did not like. This includes bug reports as well as simple communication with the appropriate person(s). It is good to hear from users as well about how the project has helped them and to find out the details of their set up.
2. **Create feature requests** that explain your use case. Describe why you find it is useful and how others can benefit. Without code contributions it is of course harder to get that feature into the code. But, if you can explain why the feature is useful and how others can benefit from it as well, you will often find out that others have similar pain points and eventually someone might implement that new feature.
3. **Test the code** while it is being developed. No matter how many automated tests are in place the reality is that the project runs on a combination of hardware and software as well as other environmental items that have not been tested by the project team (which in fact can't all be tested). So, taking a daily or weekly snapshot, installing it, and giving feedback is very helpful and welcome. For the project I work on, we changed some graphs and had one community member giving almost daily feedback about his experience from the latest code, which resulted in a lot of fixes and improvements.
4. **Write documentation**. Many project contributors are good coders but not (documentation) writers. Some documentation is barely readable and needs proofreading for grammar, spelling, and sentence construction corrections. This helps the overall implementation and evolution of a project. In other cases, documentation describes technical details but lacks any information for beginners. Plus, outlier cases, workarounds, and best practices should write down and included. If you find the same question answered repeatedly, you may also be able to write or update a Frequently asked Questions (FAQ) document, so that the answers are readily available for future reference.
5. **Translate the user interface and documentation**. While many users understand English quite well, it is also true that many enjoy documentation that is written in their native language. After writing the first German book on JBoss AS, I was contacted by people telling me that they had read all available documentation in English already but that they still benefited from the book in their native language because they could concentrate more on the technical content without the distraction of reading in a foreign language.
6. **Answer questions users have on the forums and mailing lists.** You may be surprised that you know more than you thought you did. And the user on the other end will be very grateful for your help. Also, when you attempt to answer a question, you yourself will then better understand the project. This will help you write better bug reports, feature requests, and documentation. Bonuses to helping answer questions is that users who get faster answers are then more attracted to the project and much more likely to stay and contribute *and* core project members can spend more time coding. These both work to strengthen the project.
7. **Help design the user interface, logo, and website.** Many programmers tend to create very technical user interfaces that are not aesthetically pleasing and may not attract new users. Good and self-describing interfaces do not by themselves provide new or additional functionality but can very much improve the user experience. The same also applies to the website and any logos that are needed. Improving the visual appearance of the project can thus very much contribute to reducing the support efforts and can at the same time invite new users to try it.
8. **Promote the project** by talking about it at your local user group, writing a blog post, and/or spreading updates via social media channels if you use them. Even if you think others must have heard about the project, don't assume. Hearing someone talk about their personal experience with a project is much stronger and involve others in a different way (versus browsing the project website and/or source code).
9. **Provide hardware** if there is a need for dedicated build or test servers. You can provide access to hardware in a datacentre directly to the developers or indirectly by running continuous integration or testing yourself then providing results back to the project.
10. **Thank the community** for their work and contributions to the cause you are working for and goals you are working towards.