How to Have a Successful Open Source Project

This wikiHow teaches you how to start and maintain a successful open source project. Aside from working hard and focusing on the end goal, the key to creating a successful open source project often lies in defining your goals early on in...

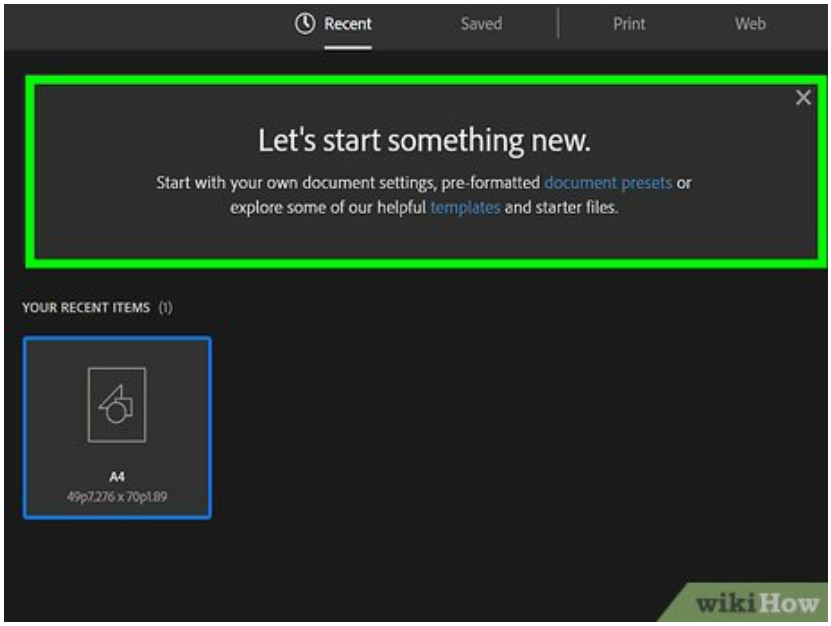
Part 1 of 3:

Preparing to Start

1. 

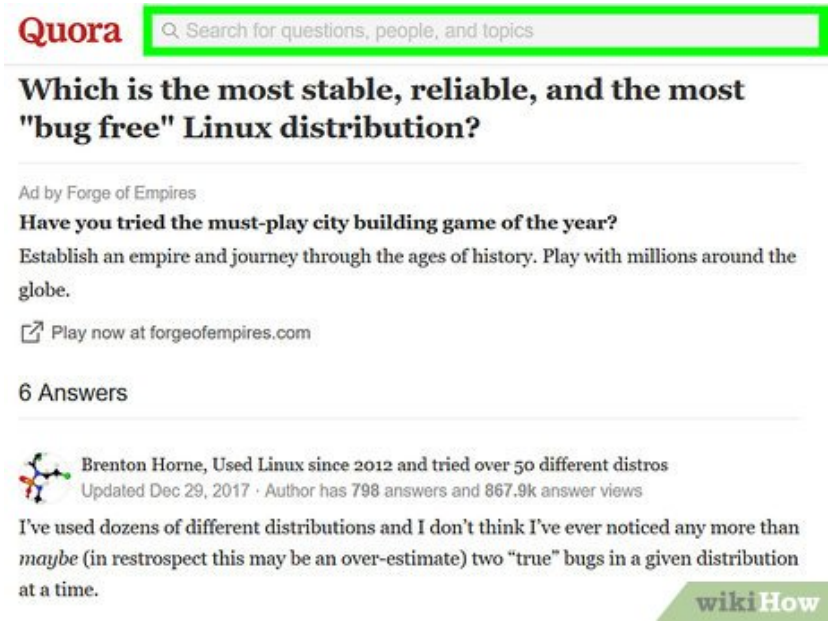
Know where to start. At its core, your open source project should provide a solution to a problem, especially if the problem is likely to evolve in the future. The first step in cultivating a successful open source project lies in finding a problem to solve, determining whether or not the problem is important enough to address, and defining your objectives from there.

1. If you already have a project in motion, be sure to write down the problem that it solves before proceeding.

2. 

Make sure that your project is necessary. Demand is one of the primary components of successful open source projects. If there isn't any demand or need for your initial project idea—or if the current demand is being fulfilled by another project—you might consider joining a different ongoing project or selecting a different problem to focus on.

1. Many ongoing open source projects accept rigorous community input, so don't be afraid to search for and join an existing version of your project instead.

3. 

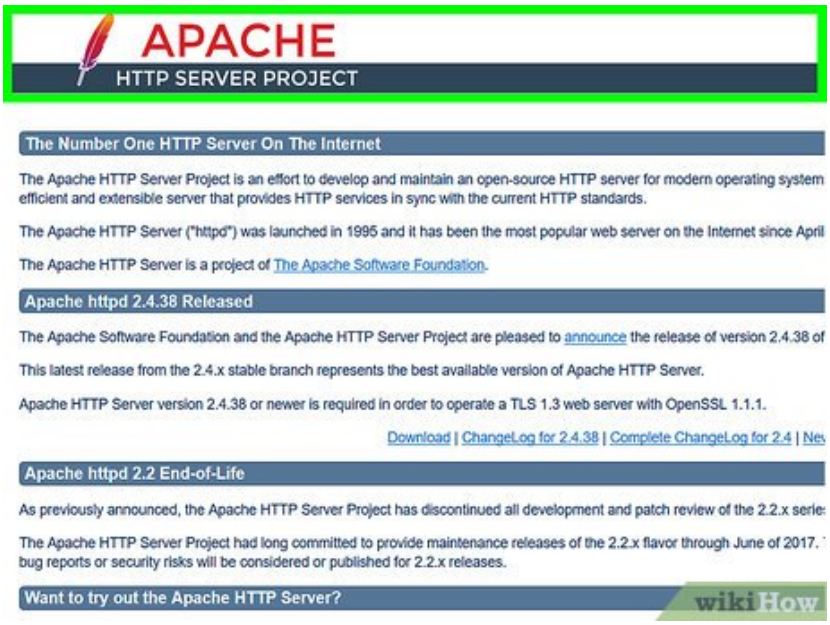
Avoid taking on large or vague problems. Not only will these problems usually achieve more official solutions in time, trying to focus on a large problem both dilutes your focus and makes it difficult to appeal to all of your audience's needs without investing an unreasonable amount of time in the project.

1. Instead, focus on a small problem which affects a large number of people (for example, a bug in a Linux distribution).

4. 

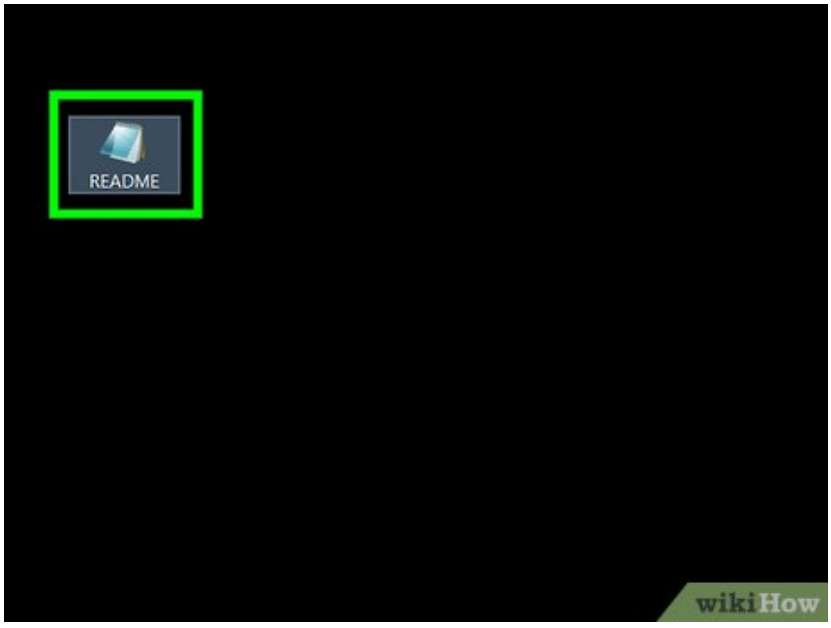
Define your project's success. Since open source projects address different categories of issues, "success" for your project will vary. Making a note of what you're attempting to achieve and how you'll know that you achieved it will help you focus on one main goal for the duration of the project.[1]

1. For example, you might consider your open source project successful if it launches, while others might consider a project successful only when it reaches a certain number of downloads.

5. 

Pick an existing and approved Open Source License for your project. Most developers know what "GPL", "LGPL" "BSD" (Berkeley Software Distribution) and "Apache" mean, which means they also know what they can do with such code and what they are not allowed to do. This will help you avoid any legal or intellectual property issues along the way.

1. Writing your own license can be time-consuming, and you'll most likely need to hire an attorney to confirm that the document checks all of the boxes.

6. 

Write the README file for your project. This may sound like an action best saved for last rather than first, but writing the README as best you can without the actual project in front of you will force you to define three crucial things: who your project is for (audience), what your project is used for (use), and where you can find additional resources (help).[2]

1. Naturally, you won't be able to list the technical instruction for your project in the README file.

Part 2 of 3:

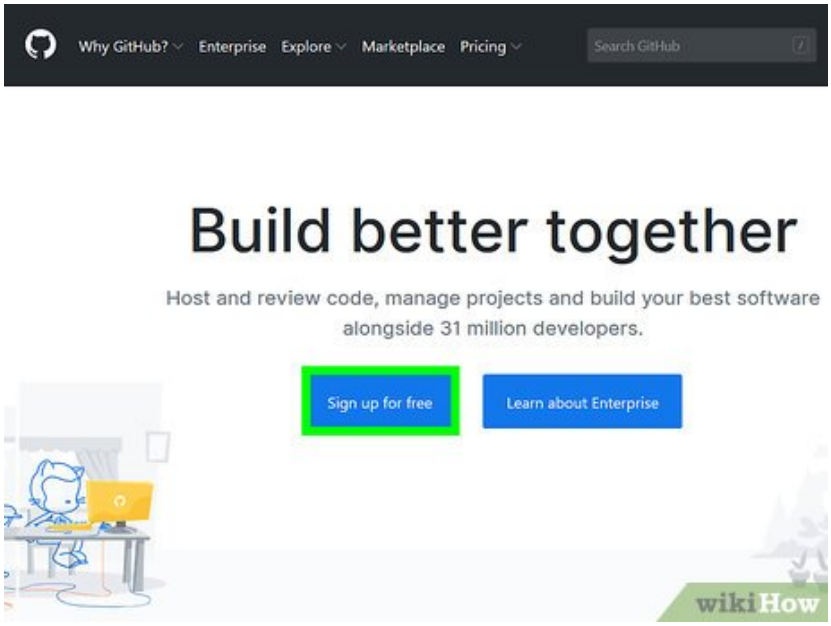
Starting the Project

1. 

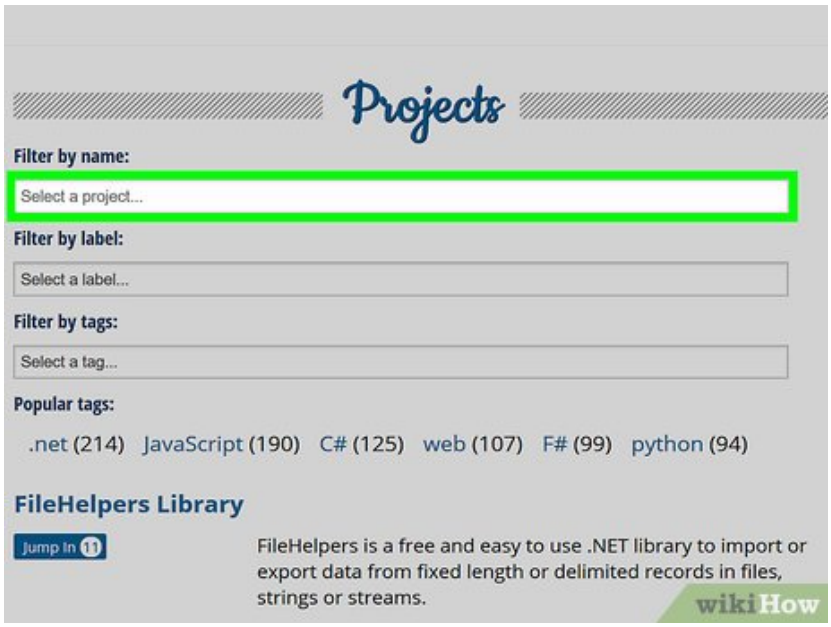
Find contributors beforehand. While you may have anything from the initial skeleton of your project to a working beta version, recruiting a few close contributors to help with the project before posting the

project anywhere will help establish a team; similarly, you'll have direct access to feedback from a few close people when you start rather than having to sort through scattered community feedback.

1. Failing to find contributors before you launch your project may result in collaborators not feeling as though they're actively a part of the process.
2. Many open source project leaders offer coding lessons or other non-material compensation to their first few contributors.

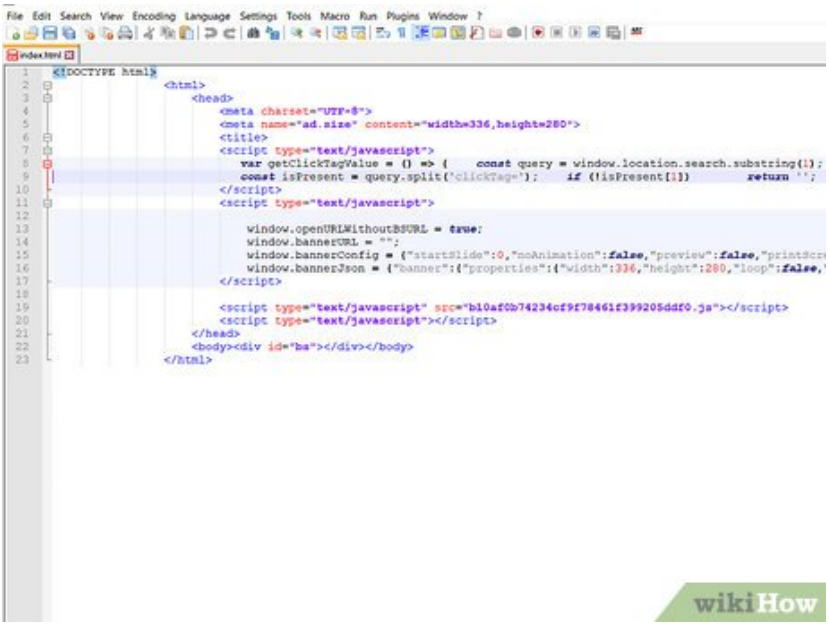
2. 

Get hosting. It is relatively easy to sign up for free hosting for an open source project; common options include SourceForge and GitHub. Not only does doing this save money, it also puts your project in a place where people are likely to search for up-and-coming open source projects.[3]

3. 

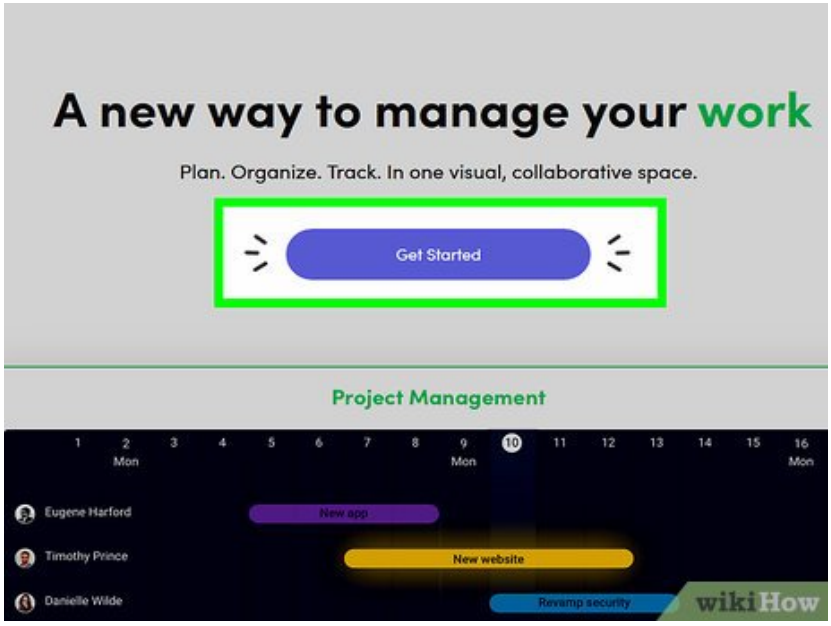
State that your project is open source. Although this seems like a very simple thing, it is one of the most overlooked aspects of an open source project. Remember, people will only view your project for a few seconds before deciding whether or not to download it; knowing that your project is open source (and,

thus, a work-in-progress) may help them form a different opinion.

4. 

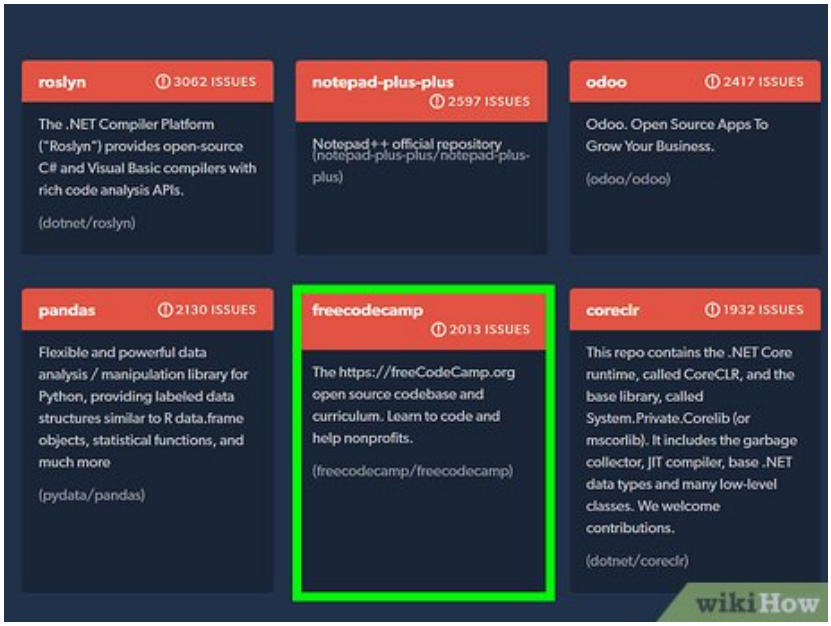
Establish transparency. The "open" part of open source means that people need to be able to see what you're doing with the code. Some easy ways to ensure that everyone has equal access to your resources include the following:

1. Store your code online so that anyone can access it.
2. Post your license, your README, and your release schedule in an easy-to-access location.
3. State your goals for the project.
4. Record and release any "private" meeting information (e.g., audio recordings or transcripts).

5. 

Release iterations of your project. Especially when you have consistent contributors or sponsors, you'll want to stick to your release schedule as accurately as possible. This will allow the community to get an idea of how your project feels before its full release goes public, and you'll be able to receive a large amount of feedback that you can use to tweak future releases.

1. It's important to remember that, while you don't need to use every piece of feedback from the community, they will want to see that you're implementing some common suggestions.

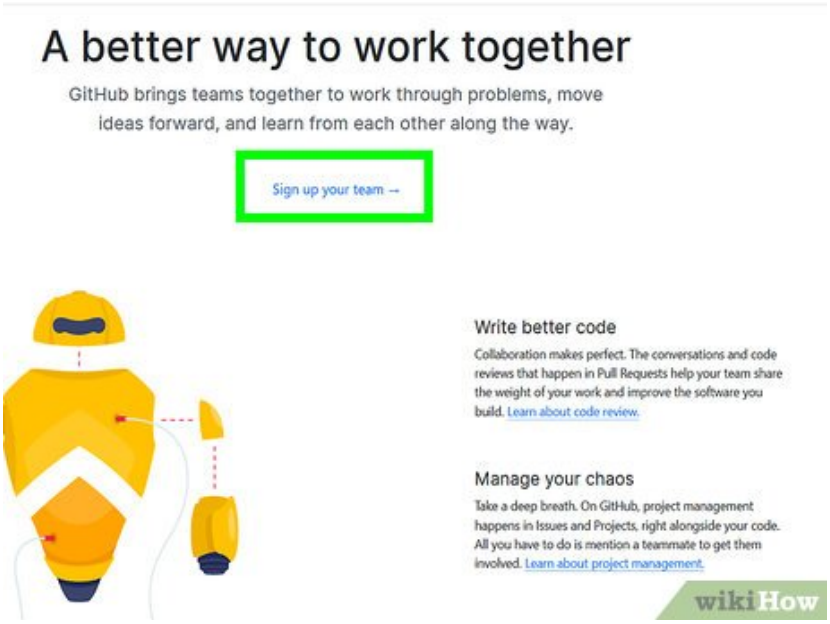
6. 

Allow community edits to your code. Although you will have to roll back vandalism and edits that don't make sense in terms of the code itself, making your code public will help you find new contributors. It will also fit the transparency culture one finds with many open source projects, which may influence future sponsors.

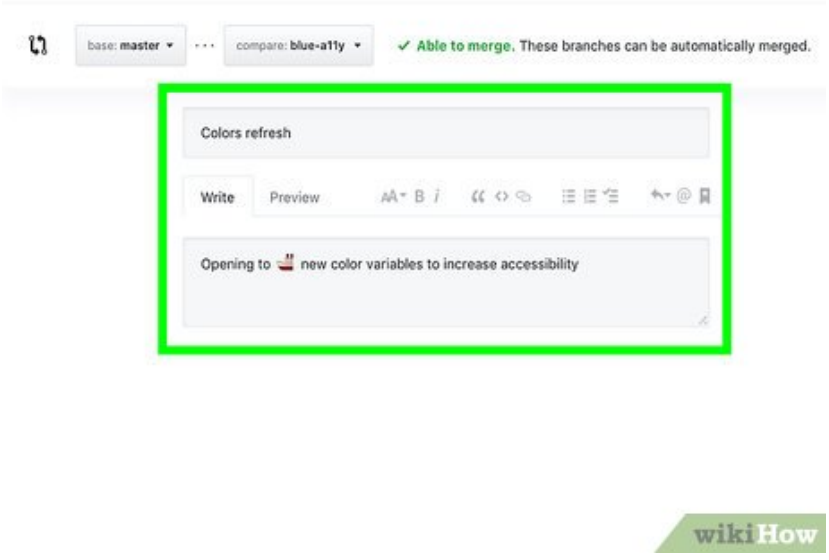
1. You can always protect the structural code and ban contributors who spam or vandalize your project if need be.

Part 3 of 3:

Maintaining the Project

1. 

Interact with the community. No matter how low- or high-profile the project, your open source work will eventually attract some form of interest and/or criticism from the community. Rather than turning them away or ignoring them, it's best to talk with interested community members in order to increase the chances of them becoming contributors.[4]

2. 

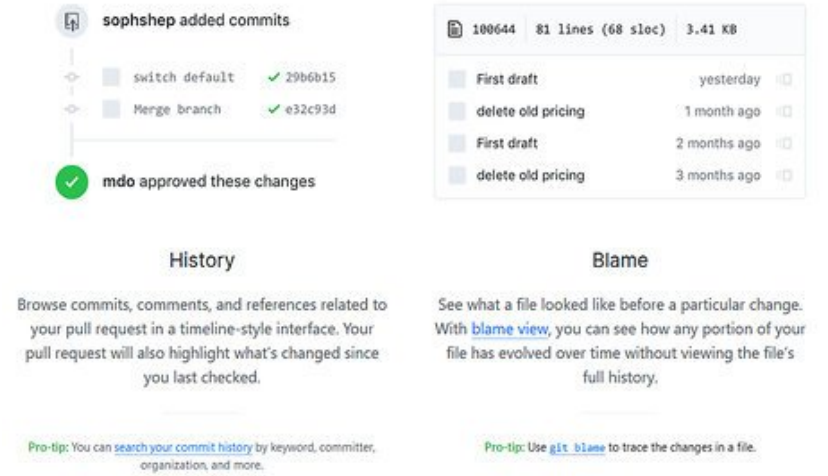
Don't do all the work yourself. As mentioned above, many community members may come to you with suggestions or ideas about how to improve your project. It's easy to take this as an invitation to make the changes yourself; instead, consider asking an interested community member to make the changes.

1. Doing this both establishes a sense of teamwork with the involved community member(s) and frees up some time for you to focus on other issues.

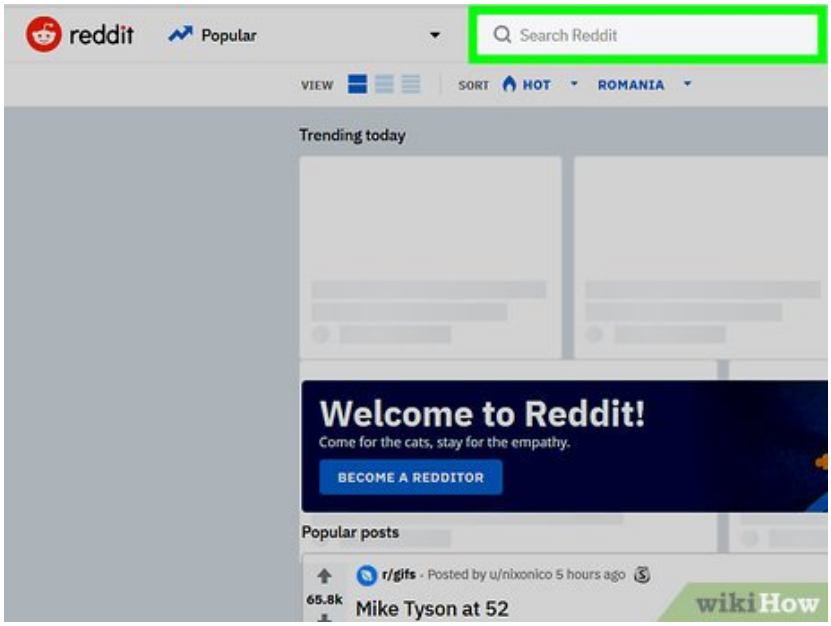
3. 

Avoid private communications. The "open" part of open source projects isn't conducive to private meetings or implementation of information without total transparency.

1. If you do end up having a private meeting about a feature or an idea, make sure to record the meeting and upload it to your project's page.

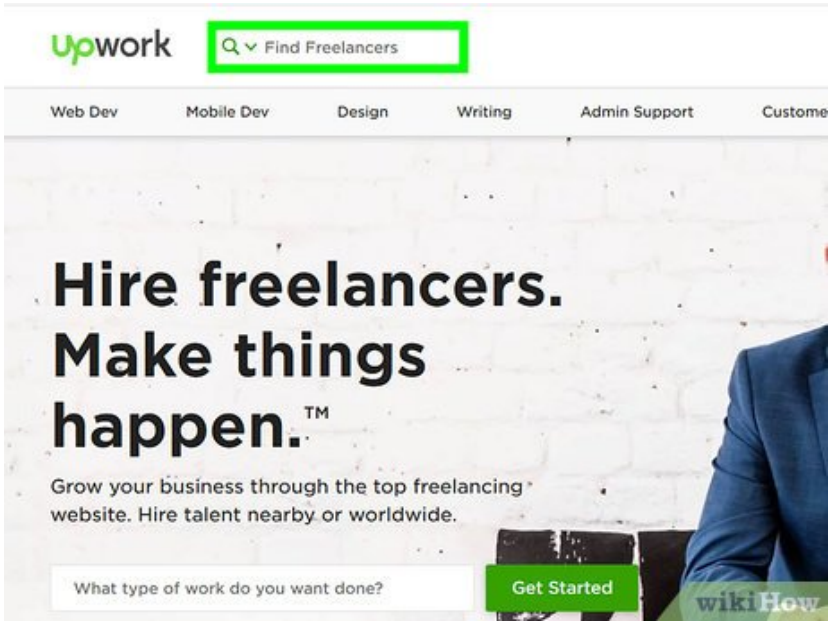
4. 

Implement pull requests. Pull requests are ways in which community members can contribute to your project. While you'll want to review these in the later stages of your project, allowing community members to tweak your code as the project proceeds will ensure that it's as well-rounded as possible.[5]

5. 

Market your project. Just like you would market a paid product, you'll need to promote your open source project through social media pages and general engagement.

1. There are virtually countless ways to promote your project, but using Reddit's programming subreddit will allow you to ask questions, respond to comments, and otherwise engage with your target audience.

6. 

Have someone to carry on the project. Invariably, your project's success will result in it needing significantly less attention than you've given it thus far. If possible, appoint a project manager to take over the project's well-being until it either becomes irrelevant or needs an update; this will allow you to focus on other projects (or take a much-needed break).