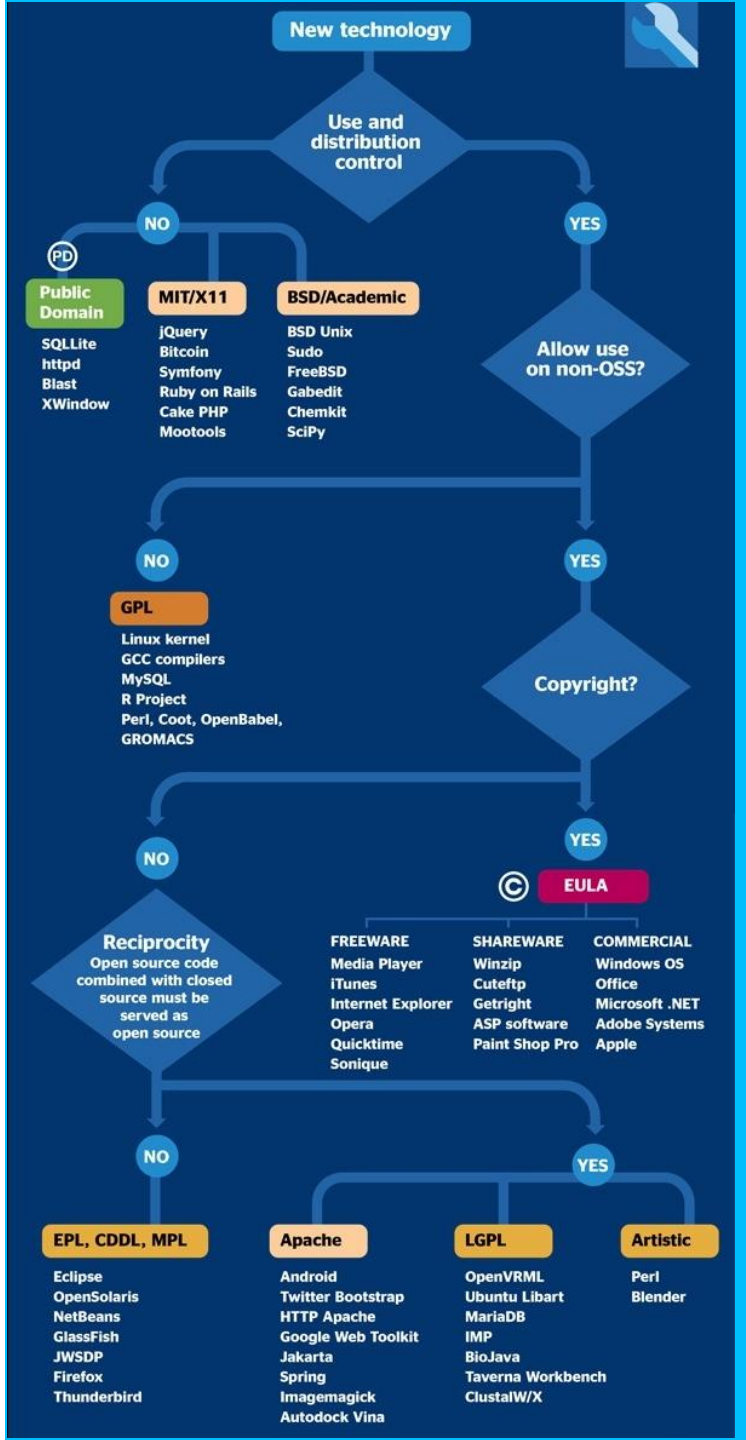
Choosing a License

Choosing an Open-Source Software License



https://fossa.com/blog/apply-license-open-source-software-project/

Choosing a License

Which open source license is appropriate for my project?

- When choosing a license to apply to your project, use an existing license instead of making up a new one.
- Such licenses are familiar to many people already. If you use one of them, people won't feel they have to read the legalese in order to use your code, because they'll have already done so for that license a long time ago.
- Thus, you reduce or remove one possible barrier to entry for your project.

List of OSS Licenses

1. **GNU General Public License version 3** (GPL-3.0, https://www.gnu.org/licenses/gpl.html)
2. **GNU Affero General Public License version 3** (AGPL-3.0, https://www.gnu.org/licenses/agpl.html)
3. **Mozilla Public License 2.0 (**MPL-2.0, https://www.mozilla.org/MPL/)
4. **GNU Library or "Lesser" General Public License version 3** (LGPL-3.0, https://www.gnu.org/licenses/lgpl.html)
5. **Eclipse Public License 1.0 (**EPL-1.0, https://www.eclipse.org/legal/epl-v10.html) (Note that version 2 of the EPL was almost ready as of mid-2014, and may be out by the time you read this.)
6. **MIT license** (MIT, https://opensource.org/licenses/MIT)
7. **Apache License 2.0** (Apache-2.0, https://apache.org/licenses/LICENSE-2.0)
8. **BSD 2-Clause** ("Simplified" or "FreeBSD") license (BSD-2-Clause, https://opensource.org/licenses/BSD-2-Clause)

Choosing a License

- If you have nothing else to guide you and you want a copyleft license, then choose either the GPL-3.0 or the AGPL-3.0
- Note that there are some arguments for choosing the Apache License 2.0 as a default non-copyleft license, and they are nearly as compelling as those for choosing MIT.

If you're starting from a blank slate, it's hard to go wrong with the MIT License. It's short, very easy to understand, and allows anyone to do anything so long as they keep a copy of the license, including your copyright notice. You'll be able to release the project under a different license if you ever need to.

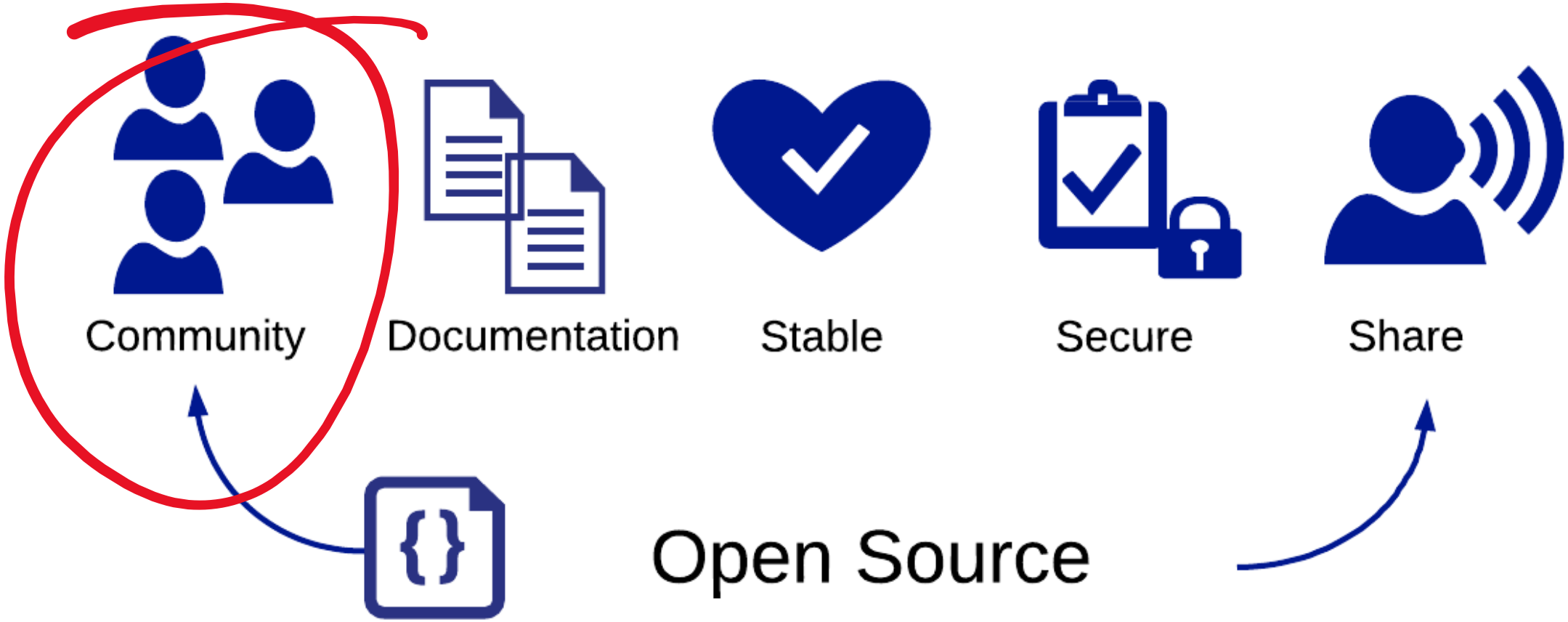
Choosing a License

- While the Apache License 2.0 has the advantage of containing some explicit defenses against misuse of software patents, which might be important to your organization depending on the kind of project you're launching, the MIT license is fully compatible with all versions of the GNU General Public License, meaning that you can distributed, under any version of the GPL, mixed-provenance works that contain MIT-licensed code.
- The GPL-compatibility situation for the Apache License, on the other hand, is more complicated — by some interpretations, it is compatible with GPL version 3 only

Choosing a License

Project Objectives

- You may also want to consider the communities you hope will use and contribute to your project:
  - **Do you want your project to be used as a dependency by other projects?** Probably best to use the most popular license in your relevant community. For example, MIT is the most popular license for npm libraries.
  - **Do you want your project to appeal to large businesses?** A large business will likely want an express patent license from all contributors. In this case, Apache 2.0 has you (and them) covered.
  - **Do you want your project to appeal to contributors who do not want their contributions to be used in closed source software?** GPLv3 or (if they also do not wish to contribute to closed source services) AGPLv3 will go over well.



Source: https://hackernoon.com/what-i-learned-from-building-my-first-successful-open-source-project-72f7429145a0

Open Source Project Community

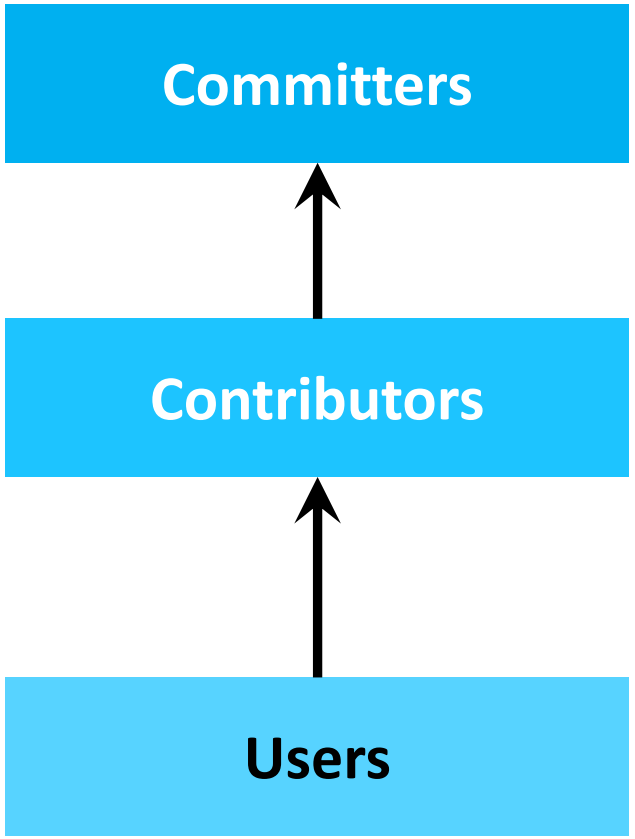
- A software that is collectively developed by a community of **technologists** interested in a **particular application** or **tool** and then **distributed** at no cost to the **broader community** of individuals who can find a use for it.

- An Open-Source project community is
  - The **group** of **peoples and companies** engaged in an open-source project
- The **developer community** is
  - The subset of the project community that is **developing** the software
- The user community is
  - The subset of the project community that is **using the software**

Open-Source Project Community- The Onion Model of FOSS

- To understand the structure and dynamics of FOSS communities, the “Onion model” provides a useful framework.
  - **Leaders**: Project founders or long-term maintainers. Set the vision and make key decisions.
  - **Core Members**: Trusted contributors managing key sections of the codebase.
  - **Active Developers**: Regular contributors involved in coding, bug fixing, and feature development.
  - **Peripheral Developers**: Occasional contributors focusing on specific tasks.
  - **Bug Fixers**: Identify and fix issues, enhancing stability.
  - **Readers**: Engage with documentation and code without contributing.
  - **Passive Users**: Use the software without participating actively.

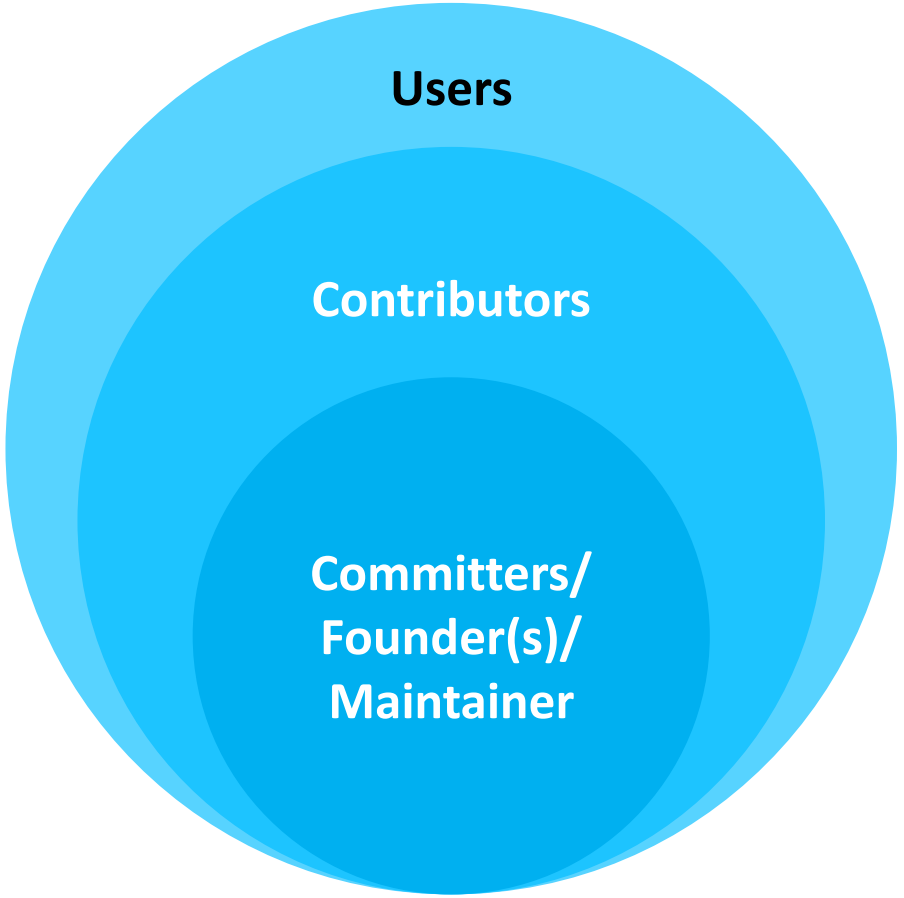
Open-Source Project Community- The Onion Model of FOSS



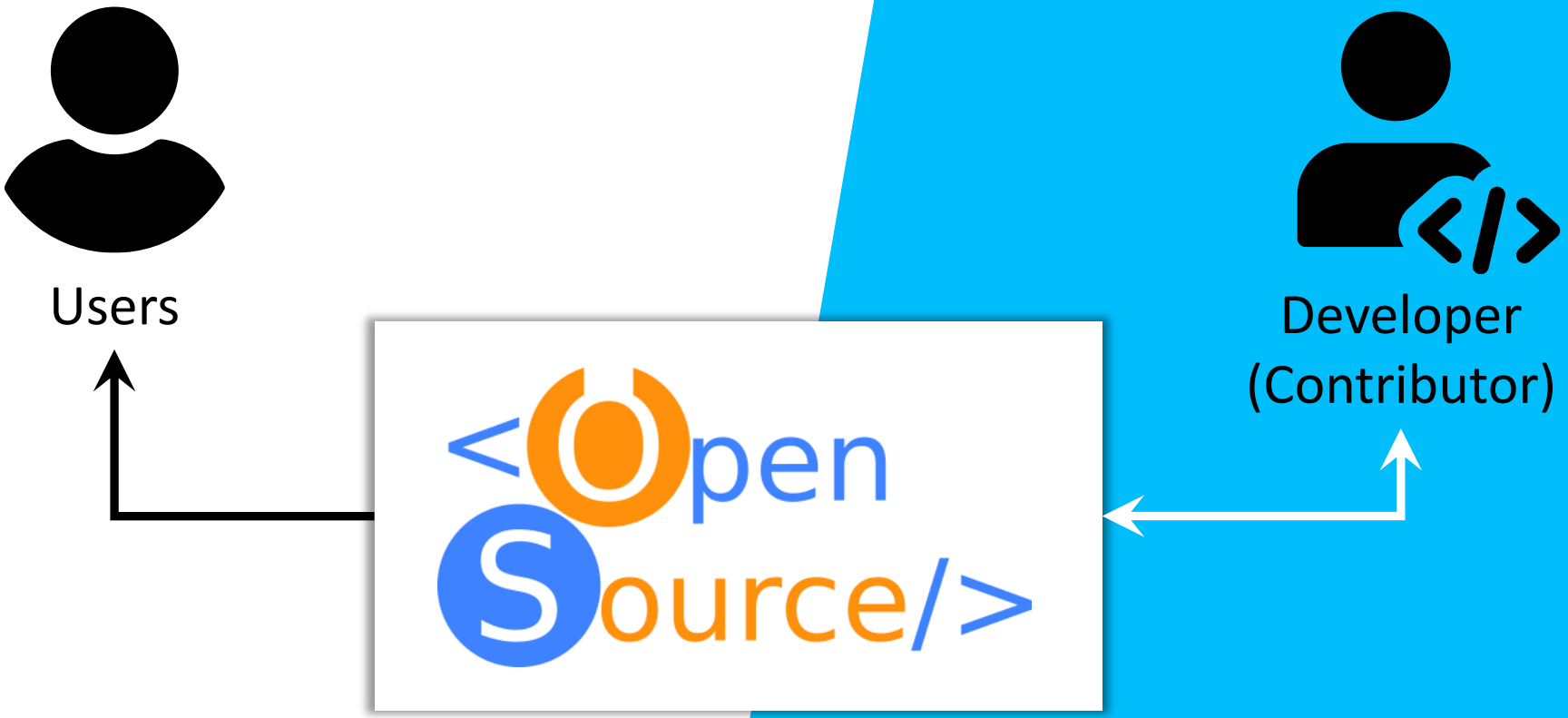
- Project Leader
- Core member
- Has commit(write) rights
- Perform bulk of the work, patch review

- Developer
- Provide small features, bug fixes
- Submit patches (no commits rights)

- People who know and use the project
- If so, help with comments, feedback..



Open-Source Project Community- The Onion Model of FOSS



Open-source projects should be run in such a way as to make that transition welcoming to anyone interested

Open-Source Project Community- The Onion Model of FOSS

Contributors Types

Core Contributors

- Most senior and experienced people in the project
- Provide guidance and mentorship to the newer community members
- Hold the *commit bit* - they are able to approve changes made to the project

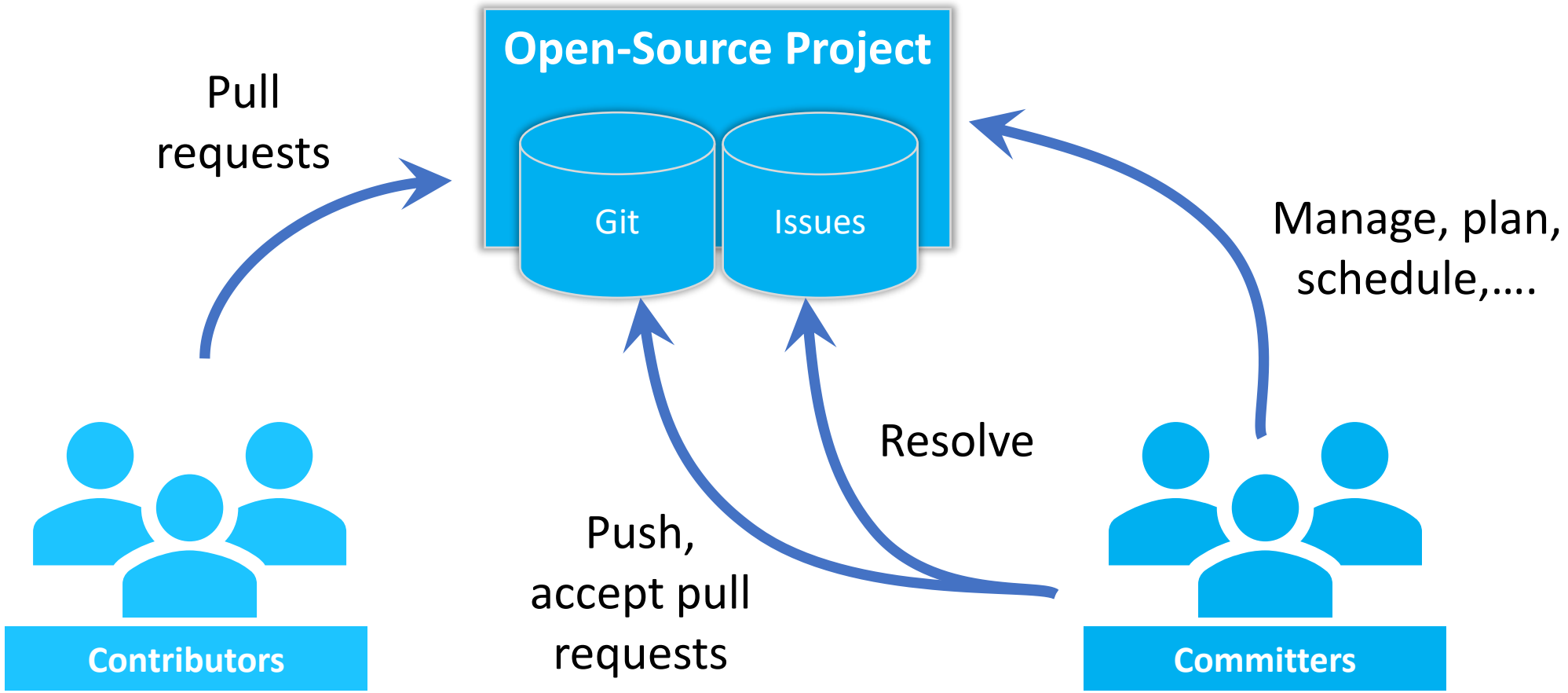
Non-core contributors

- Make regular contributions
- Active in issue discussions
- Provide feedback to the newer members of the community

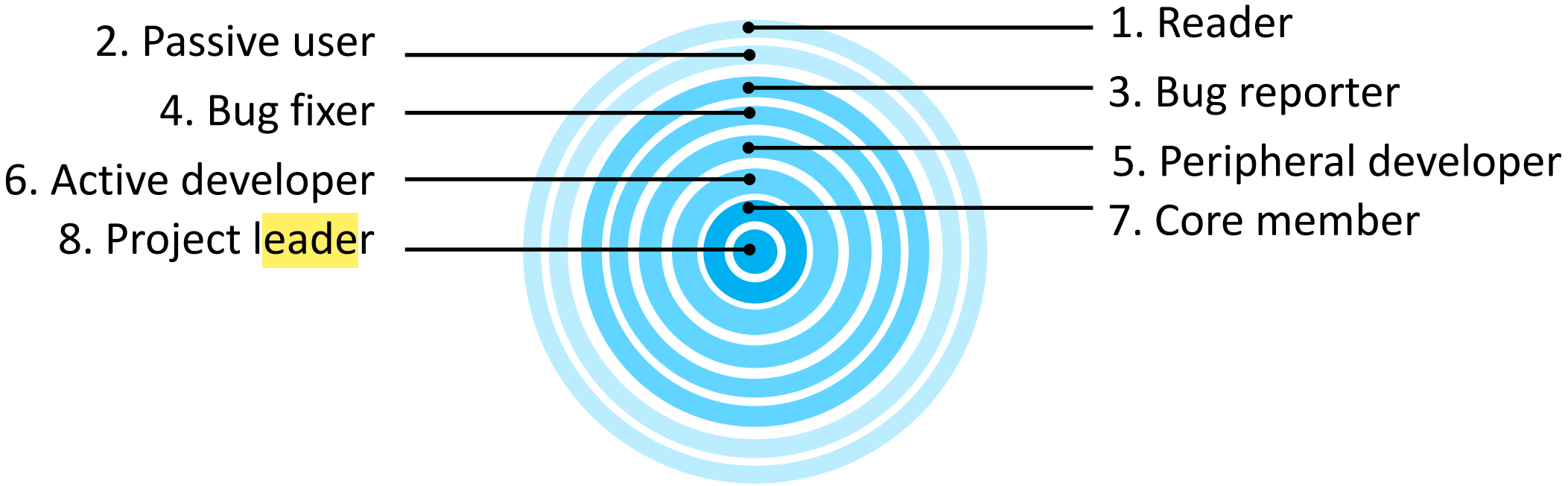
New contributors

- People interested in contributing to the project
- Still learning the structure of the project and how to interact with the community

Open-Source Project Community- The Onion Model of FOSS

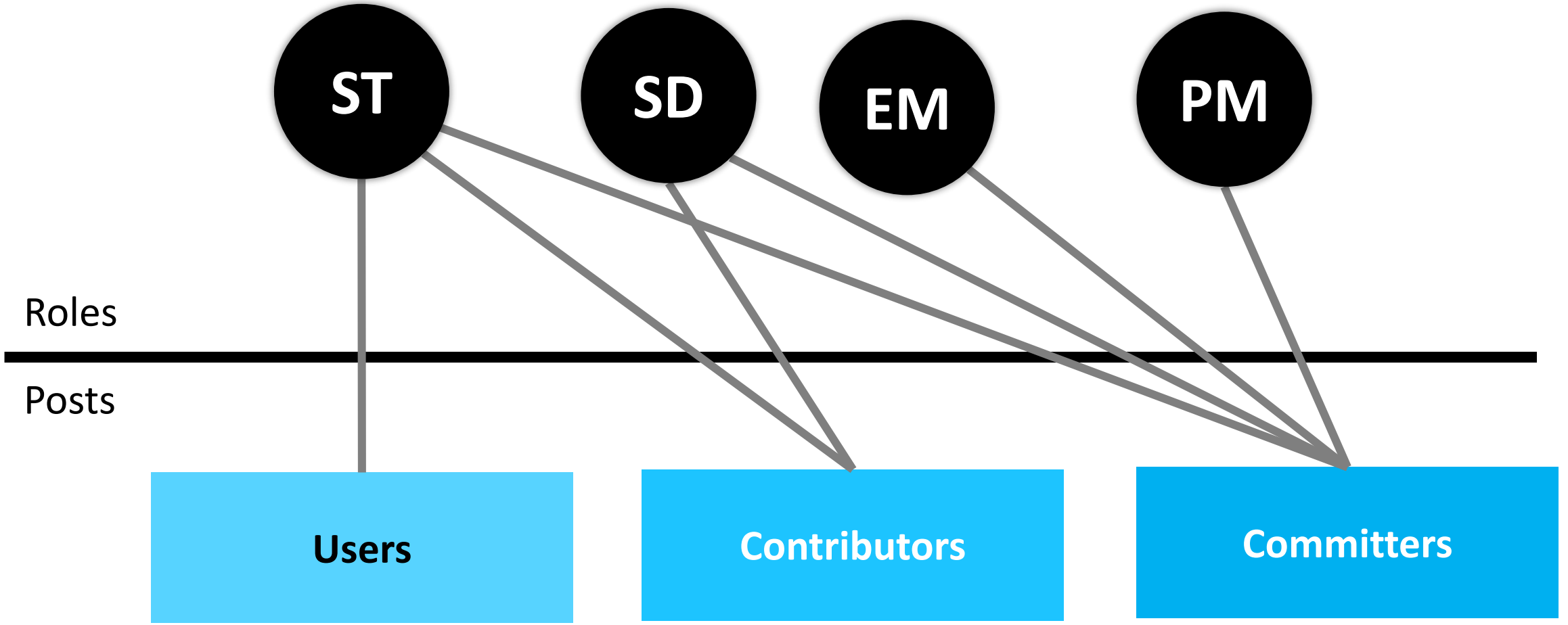


Open-Source Project Community- The Onion Model of FOSS



Users | **Contributors** | **Committers**

Roles and Post in Open-Source Projects



Why contribute to open source?

- Contributing to open source can be a rewarding way to learn, teach, and build experience in just about any skill you can imagine.

- Why do people contribute to open source? Plenty of reasons!
  - Improve software you rely on
  - Improve existing skills
  - Meet people who are interested in similar things
  - Find mentors and teach others
  - Build public artifacts that help you grow a reputation (and a career)
  - Learn people skills
  - It’s empowering to be able to make changes, even small ones

What it means to contribute

If you’re a new open-source contributor, the process can be frightening. How do you find the right project? What if you don’t know how to code? What if something goes wrong?

Not to worry! There are all sorts of ways to get involved with an open-source project, and a few tips will help you get the most out of your experience.

What it means to contribute

- **Do you like planning events?**
  - Organize workshops or meetups about the project, like @fzamperin did for NodeSchool
  - Organize the project's conference (if they have one)
  - Help community members find the right conferences and submit proposals for speaking
- **Do you like to design?**
  - Restructure layouts to improve the project's usability
  - Conduct user research to reorganize and refine the project's navigation or menus, like Drupal suggests
  - Put together a style guide to help the project have a consistent visual design
  - Create art for t-shirts or a new logo, like hapi.js's contributors did

What it means to contribute

- **Do you like to write?**
  - Write and improve the project's documentation
  - Curate a folder of examples showing how the project is used
  - Start a newsletter for the project, or curate highlights from the mailing list
  - Write tutorials for the project, like PyPA's contributors did
  - Write a translation for the project's documentation

- **Do you like organizing?**
  - Link to duplicate issues, and suggest new issue labels, to keep things organized
  - Go through open issues and suggest closing old ones, like @nzakas did for ESLint
  - Ask clarifying questions on recently opened issues to move the discussion forward

What it means to contribute

- **Do you like to code?**
  - Find an open issue to tackle, like @dianjin did for Leaflet
  - Ask if you can help write a new feature
  - Automate project setup
  - Improve tooling and testing
- **Do you like helping people?**
  - Answer questions about the project on e.g., Stack Overflow (like this Postgres example) or Reddit
  - Answer questions for people on open issues
  - Help moderate the discussion boards or conversation channels

What it means to contribute

- **Do you like helping others code?**
  - Review code on other people's submissions
  - Write tutorials for how a project can be used
  - Offer to mentor another contributor, like @ereichert did for @bronzdoc on Rust

Finding a project to contribute to

- Now that you've figured out how open source projects work, it's time to find a project to contribute to!
- You can also use one of the following resources to help you discover and contribute to new projects:
- GitHub Explore
- Open Source Friday
- First Timers Only
- CodeTriage
- 24 Pull Requests
- Up For Grabs
- Contributor-ninja
- First Contributions
- SourceSort

Reading Materials

- Book
  - **Producing Open-Source Software How to Run a Successful Free Software Project** → Chapter 9
  - **Getting started with open-source development** → Chapter 3 – Licensing
  - **Understanding Open Source and Free Software Licensing** → Chapter 1: Open Source Licensing, Contract, and Copyright Law

- https://opensource.guide/legal/

Reading Materials

- **Books:**
  1. VM Brasseur, *Forge Your Future with Open Source*, The Pragmatic Programmers, LLC. 2018.
  2. Karl Fogel, *Producing Open Source Software: How to Run a Successful Free Software Project*, O'Reilly Media, 2009.
- https://www.gnu.org/philosophy/free-sw.html
- https://blog.lizardwrangler.com/2008/01/22/january-22-1998-the-beginning-of-mozilla
- https://opensource.org/licenses
- https://www.coredna.com/blogs/comparing-open-closed-source-software
- https://linkedretail.com/open-vs-closed-source/
- https://www.assignmentprime.com/blog/open-vs-closed-source-software
- https://www.gnu.org/philosophy/free-sw.en.html
- https://opensource.org/osd
- https://pavanganeshbutha1998.blogspot.com/2019/10/free-software-philosphy.html
- https://www.youtube.com/watch?v=Ag1AKIl_2GM&t=89s

Reading Materials

- https://www.youtube.com/watch?v=02m1T9c7CFk
- https://www.coursera.org/lecture/open-source-software-development-methods/open-source-governance-models-TEhwM
- https://hackernoon.com/what-i-learned-from-building-my-first-successful-open-source-project-72f7429145a0
- https://opensource.guide/how-to-contribute/
- https://fossa.com/blog/apply-license-open-source-software-project/
- https://opensource.guide/legal/

Attribution

- Some Slides are copied from **Prof. Stewart Weiss Lectures :** http://www.compsci.hunter.cuny.edu/~sweiss/course_materials/csci395.86/slides/introduction.html#1

Thanks

Office Time: Monday-Friday (1000 - 1800)

You can send me an email for meeting, or any sort of discussion related to class matters.

jamil@sejong.ac.kr