

CEN 5016: Software Engineering

Spring 2024



University of
Central Florida

Dr. Kevin Moran

Week 6 - Class II: Open Source Software + Security





- *Assignment 4*
 - Due Monday
 - Exploring Static Analysis Tools and CI with a simple Python app
 - Accept the Assignment on GitHub Classroom
- *SDE Project Part 2*
 - Due Friday, March 1st
 - Will get you Feedback on your plan by tomorrow EoD!
 - Two parts:
 - Process & Implementation Snapshot
 - Checkpoint Presentation
- *Grades*
 - Assignment 2+ SDE project released
 - We are working on Assignment 3
 - Assignment 1 and all Quizzes/In-class activities will be posted today



- *SDE Project Backlog*
 - *Some Tips:*
 - Relate your Backlog to your User Stories
 - Break down each user story into multiple tasks (probably 3-4 is reasonable, but use your best judgment)
 - *Describe* the technical details of how you will complete each issue and estimate how long it will take
 - Schedule issues across milestones
 - Assign teammates to issues!

Midterm Exam Format



- 2 Parts, In-class exam, closed book, 200 points total
 - *Part 1:* Multiple Choice
 - 12-15 questions
 - Will test basic knowledge of concepts, select the best answer for each question
 - *Part 2:* Short Answer Questions
 - 4-5 questions
 - Concepts from class, SE scenarios, answer in a paragraph
- Covers material from Weeks 1-6
- You will have the **entire** class period to complete the exam
- Please bring your UCF ID to the exam

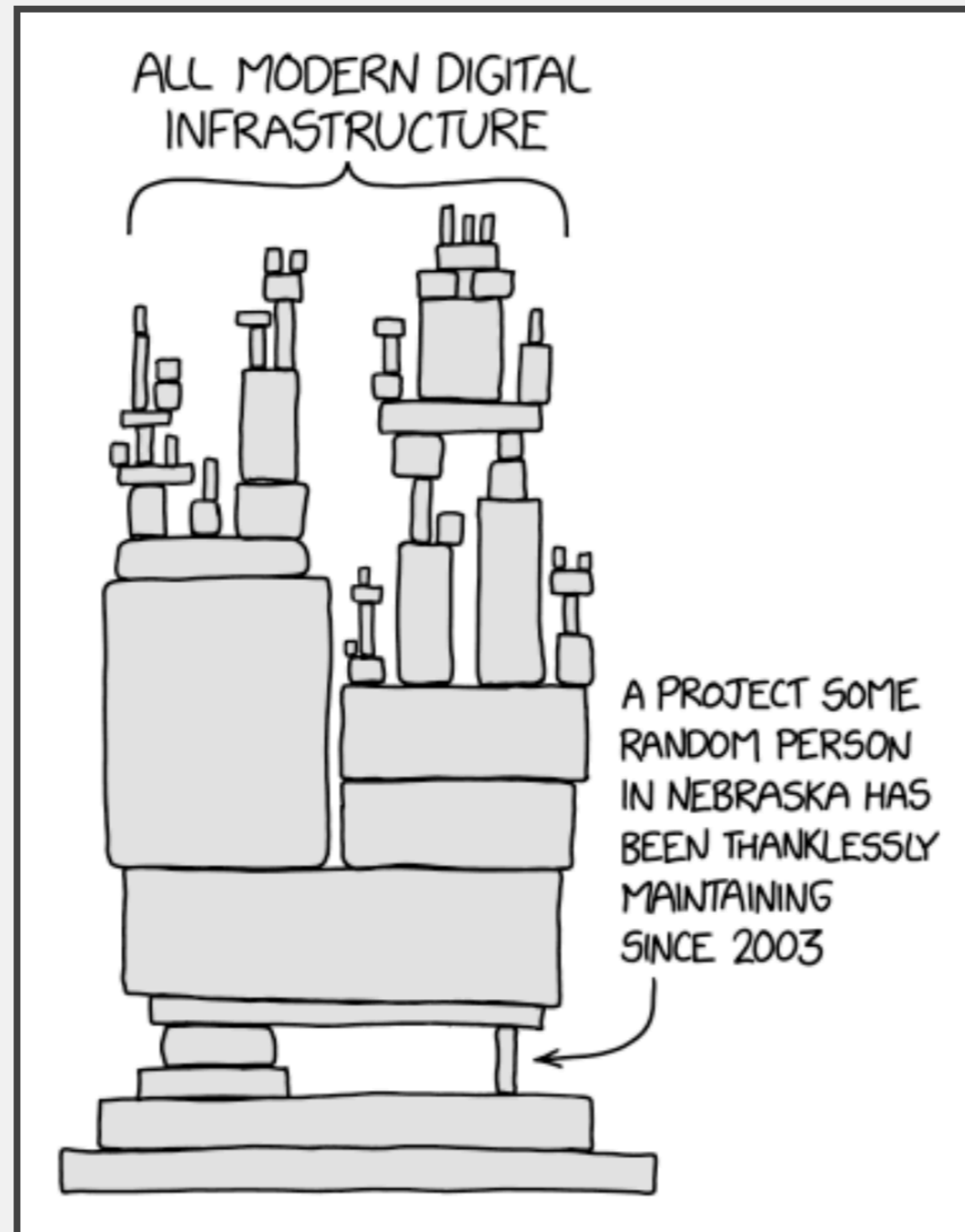
Open-Source Software





- Distinguish between open-source software, free software, and commercial software.
- Identify the common types of software licenses and their implications.
- Distinguish between copyright and intellectual property.
- Express an educated opinion on the philosophical/political debate between open source and proprietary principles.
- Describe how open-source ecosystems work and evolve, in terms of maintainers, community contribution, and commercial backing
- Identify various concerns of commercial entities in leveraging open-source, as well as strategies to mitigate these.

The Importance of Open-Source



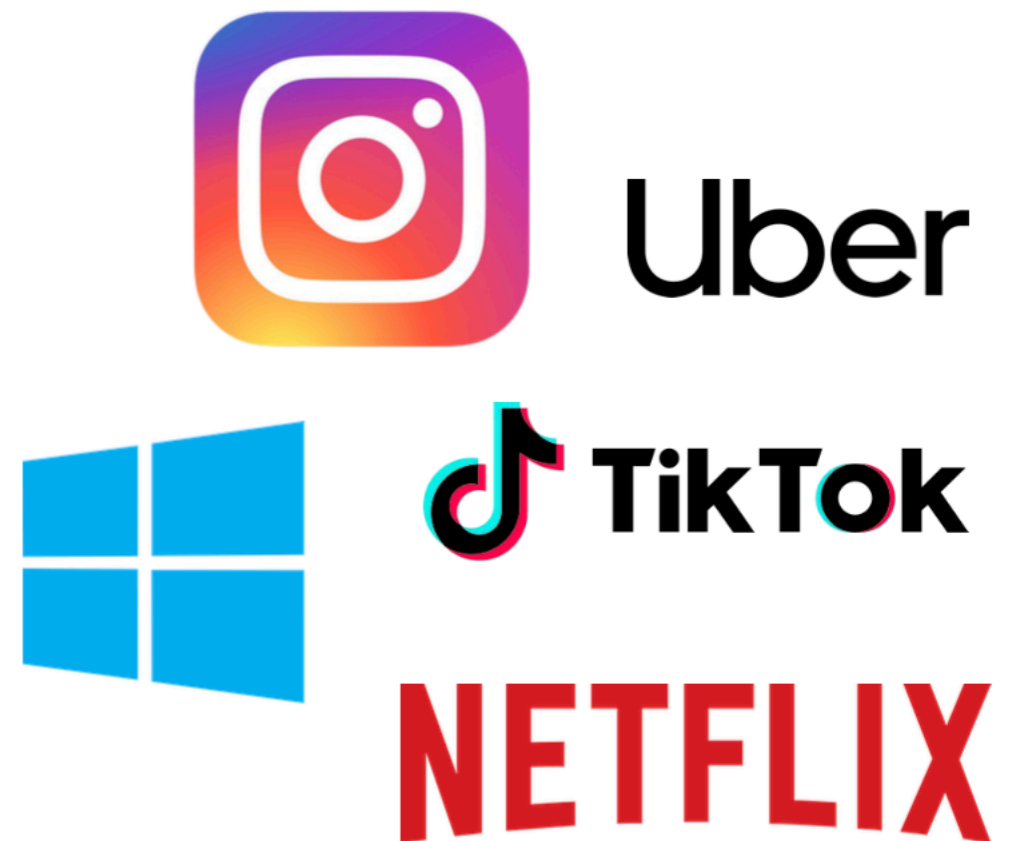
What is Open Source Software?



Open-source



Proprietary



What is Open Source Software?



- Source code availability
- Right to modify and create derivative works
- (Often) Right to redistribute derivative works



- Intention is to be used, not examined, inspected, or modified.
- No source code – only download a binary (e.g., an app) or use via the internet (e.g., a web service).
- Often contains an End User License Agreement (EULA) governing rights and liabilities.
- EULAs may specifically prohibit attempts to understand application internals.

Contrast with Proprietary Software: A Black Box



7:28 65%

End User License Agreement

1. Grant of License

The Licensor hereby grants you limited, personal, non-exclusive, non-transferable, revocable license to install the Application on your mobile device for your personal use. You may not (and shall not permit or assist any third party to): (i) copy (except as expressly permitted by this License), decompile, reverse engineer, disassemble, attempt to derive the source code, modify, or create derivative works of the Application, any updates, or any part thereof; (ii) rent, lease, lend, sell, redistribute or sublicense the Application; (iii) use the application in any manner that could damage, disable, overburden, or impair the Application (or any server or networks connected to the Application) or interfere with any third party's use and/or enjoyment of the Application (or any server or networks connected to the Application); (iv) intentionally interfere with or circumvent the Application's security features; (v) use, test or otherwise utilize the Application in any manner for purposes of developing or implementing any method or software that is intended to monitor or interfere (including intercept or capture data) with the functioning of the Application (or any server or networks connected to the Application); or (vi) otherwise use the Application in any unlawful manner, for any unlawful purpose or in any other manner not expressly granted in this License. The terms of this License will govern any updates provided by the Licensor that replace and/or supplement the original Application.

Any open source software that may be accompanying the Application is provided to you under the terms of such open source license agreement. This License does not apply to any such open source software accompanying the Application, except as expressly stated herein.

2. Ownership

The software, content, visual interfaces, interactive features, information, graphics, design, compilation, computer code and all other elements of the Applications (the "Materials") are protected by intellectual property rights—including copyright, trade dress, patent, trade secret and trademark laws of the United States, other jurisdictions, and international conventions, and all other applicable laws (collectively, "Applicable Intellectual Property Laws"). All Materials are the property of the Licensor or its subsidiaries or affiliated companies and/or third-party licensors. The Licensor reserves all rights not expressly granted in this License. You shall not acquire any right, title or interest to the Materials, whether by implication, estoppel, or otherwise, except for the limited rights set forth in this License. You hereby agree to abide by all Applicable Intellectual Property Laws.

3. Privacy and Consent to Use of Data

You agree that the Licensor, its affiliates, and their corresponding service providers may collect, maintain, and use technical data and related information about you and your device

Decline Accept

7:28 65%

End User License Agreement

1. Grant of License

The Licensor hereby grants you limited, personal, non-exclusive, non-transferable, revocable license to install the Application on your mobile device for your personal use. You may not (and shall not permit or assist any third party to): (i) copy (except as expressly permitted by this License), decompile, reverse engineer, disassemble, attempt to derive the source code, modify, or create derivative works of the Application, any updates, or any part thereof; (ii) rent, lease, lend, sell, redistribute or sublicense the Application; (iii) use the application in any manner that could damage, disable, overburden, or impair the Application (or any server or networks connected to the Application) or interfere with any third party's use and/or enjoyment of the Application (or any server or networks connected to the Application); (iv) intentionally interfere with or circumvent the Application's security features; (v) use, test or otherwise utilize the Application in any manner for purposes of developing or implementing any method or software that is intended to monitor or interfere (including intercept or capture data) with the functioning of the Application (or any server or networks connected to the Application); or (vi) otherwise use the Application in any unlawful manner, for any unlawful purpose or in any other manner not expressly granted in this License. The terms of this License will govern any updates provided by the Licensor that replace and/or supplement the original Application.

Any open source software that may be accompanying the Application is provided to you under the terms of such open source license agreement. This License does not apply to any such open source software accompanying the Application, except as expressly stated herein.

2. Ownership

The software, content, visual interfaces, interactive features, information, graphics, design, compilation, computer code and all other elements of the Applications (the "Materials") are protected by intellectual property rights—including copyright, trade dress, patent, trade secret and trademark laws of the United States, other jurisdictions, and international conventions, and all other applicable laws (collectively, "Applicable Intellectual Property Laws"). All Materials are the property of the Licensor or its subsidiaries or affiliated companies and/or third-party licensors. The Licensor reserves all rights not expressly granted in this License. You shall not acquire any right, title or interest to the Materials, whether by implication, estoppel, or otherwise, except for the limited rights set forth in this License. You hereby agree to abide by all Applicable Intellectual Property Laws.

3. Privacy and Consent to Use of Data

You agree that the Licensor, its affiliates, and their corresponding service providers may collect, maintain, and use technical data and related information about you and your device

Decline Accept



- *Free software origins (70-80s ~Stallman)*
 - ~~Cultish~~ Political goal
 - Software part of free speech
 - free exchange, free modification
 - proprietary software is unethical
 - security, trust
 - GNU project, Linux, GPL license
- *Open source (1998 ~O'Reilly)*
 - Rebranding without political legacy
 - Emphasis on internet and large dev/user involvement
 - Openness toward proprietary software/coexist
 - (Think: Netscape becoming Mozilla)



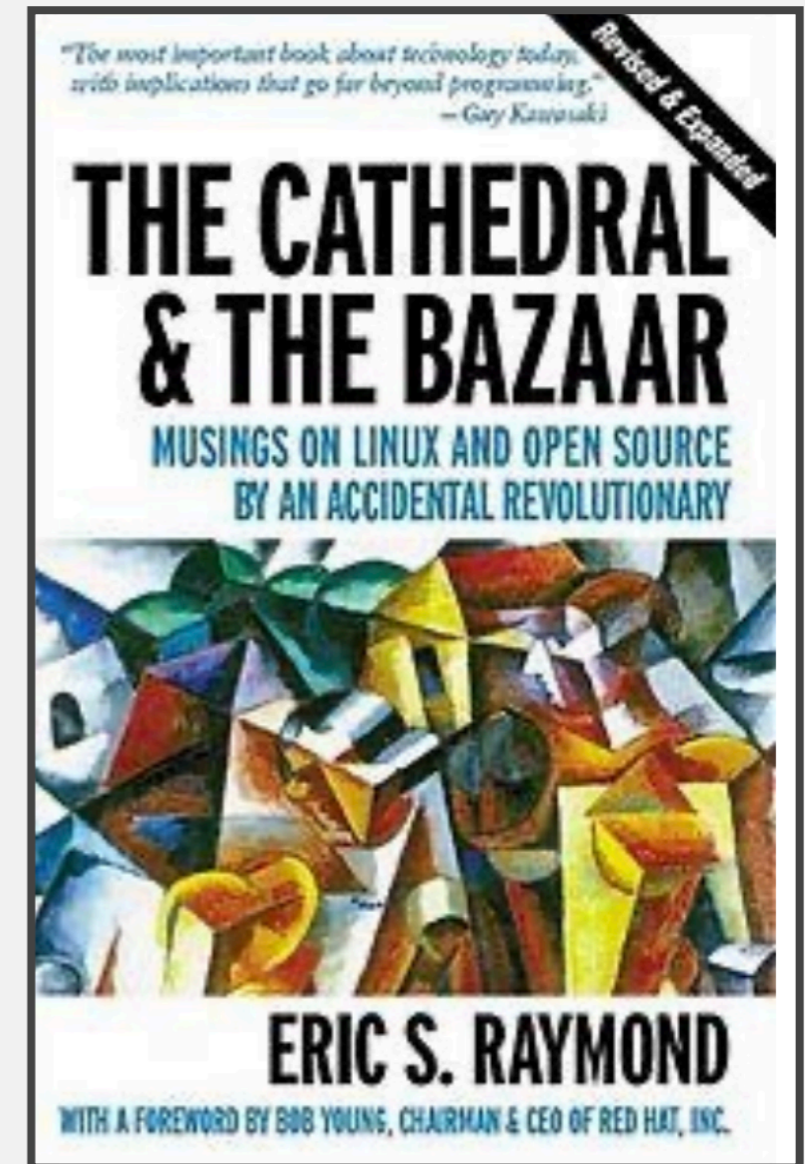
Perception (from some):

- Anarchy
- Demagoguery
- Ideology
- Altruism

Open-Source Ecosystems



The Cathedral and the Bazaar





Cathedral

- Developed centrally by a core group of members
- Available for all once complete (or at releases)
- Examples: GNU Emacs, GCC (back in the 1990s)
- “Sort of” examples today: Chrome
Intellij

Bazaar

- Developed openly and organically
- Wide participation (in theory, anyone can contribute)
Examples: Linux

OSS has many stakeholders / contributors



- **Core members**
 - Often (but not always) includes the original creators
 - Direct push access to main repository
 - May be further split into admin roles and developers
- **External contributors**
 - File bug reports and report other issues
 - Contribute code and documentation via pull requests
- **Other supporters**
 - Beta testers (users)
 - Sponsors (financial or platform)
 - Steering committees or public commenters (for standards and RFCs)
- **Spin-offs**
 - Maintainers of forks of the original repository



- Mature OSS projects often have strict contribution guidelines
 - Look for CONTRIBUTING.md or similar
- **Common requirements:**
 - Coding style (recall: linters) and passing static checks
 - Inclusion of test cases with new code
 - Minimum number of code reviews from core devs
 - Standards for documentation
 - Contributing licensing agreements (more on that later)



- Some OSS projects are managed by for-profit firms
 - Examples: Chromium (Google), Moby (Docker), Ubuntu (Canonical), TensorFlow (Google), PyTorch (Meta), Java (Oracle)
 - Contributors may be a mix of employees and community volunteers
 - Corporations often fund platforms (websites, test servers, deployments, repository hosting, etc.)
 - Corporations usually control long-term vision and feature roadmap
- Many OSS projects are managed by non-profit foundations or ad-hoc communities
 - Examples: Apache Hadoop/Spark/Hbase/Kafka/Tomcat (ASF), Firefox (Mozilla), Python (PSF), NumPy (community)
 - Foundations fund project infrastructure via charitable donations
 - Long-term vision often developed via a collaborative process (e.g., Apache) or by benevolent dictators (e.g., Python, Linux)
- Corporations still heavily rely on community-owned OSS projects • Many OSS non-profits are funded by Big Tech (e.g., Mozilla by Google)

Example: Apache



WHAT MAKES THE APACHE WAY SO HARD TO DEFINE?

The Apache Way is a living, breathing interpretation of one's experience with our community-led development process. Apache is unique, diverse, and focused on the activities needed at a particular stage of the project's lifetime, including nurturing community and building awareness. What is important is that they embrace:

- **Earned Authority:** all individuals are given the opportunity to participate, but their influence is based on publicly earned community. Merit lies with the individual, does not expire, is not influenced by employment status or employer, and is not project-specific (cannot be applied to another). [More on merit.](#)
- **Community of Peers:** individuals participate at the ASF, not organizations. The ASF's flat structure dictates that roles are of equal weight, and contributions are made on a volunteer basis (even if paid to work on Apache code). The Apache community operates with respect in adherence to our [Code of Conduct](#). Domain expertise is appreciated; Benevolent Dictators For Life are encouraged. [More on participation.](#)
- **Open Communications:** as a virtual organization, the ASF requires all communications related to code and decision-making to be asynchronous collaboration, as necessitated by a globally-distributed community. Project mailing lists are archived, public, and searchable.
 - dev@ (primary project development)
 - user@ (user community discussion and peer support)
 - commits@ (automated source change notifications)
 - occasionally supporting roles such as marketing@ (project visibility)

...as well as restricted, day-to-day operational lists for Project Management Committees. Private decisions on code, policies, or other discourse and transactions must be brought on-list. [More on communications](#) and the [use of mailing lists](#).

- **Consensus Decision Making:** Apache Projects are overseen by a self-selected team of active volunteers who are contributors. Projects are auto-governing with a heavy slant towards driving consensus to maintain momentum and productivity. While consensus is established at all times, holding a vote or other coordination may be required to help remove any blocks with binding decisions. [More on decision making and voting.](#)
- **Responsible Oversight:** The ASF governance model is based on trust and delegated oversight. Rather than detailed rules, governance is principles-based, with self-governing projects providing reports directly to the Board. Apache Committers are responsible for reviewing commits, employing mandatory security measures, ensuring license compliance, and protecting the Apache community from abuse. [More on responsibility.](#)



OUR SPONSORS

The Apache Software Foundation could not exist without the continued generous support from the community. We would like to take this opportunity to thank our sponsors. If you are interested in sponsoring the ASF, please read our [sponsorship page](#).

FOUNDATION SPONSORS

Platinum Sponsors:

FACEBOOK yahoo!

Facebook

Yahoo!



Apple



Google

Corporate Outlook Towards Open-source



-2-

February 3, 1976

An Open Letter to Hobbyists

To me, the most critical thing in the hobby market right now is the lack of good software courses, books and software itself. Without good software and an owner who understands programming, a hobby computer is wasted. Will quality software be written for the hobby market?

Almost a year ago, Paul Allen and myself, expecting the hobby market to expand, hired Monte Davidoff and developed Altair BASIC. Though the initial work took only two months, the three of us have spent most of the last year documenting, improving and adding features to BASIC. Now we have 4K, 8K, EXTENDED, ROM and DISK BASIC. The value of the computer time we have used exceeds \$40,000.

The feedback we have gotten from the hundreds of people who say they are using BASIC has all been positive. Two surprising things are apparent, however. 1) Most of these "users" never bought BASIC (less than 10% of all Altair owners have bought BASIC), and 2) The amount of royalties we have received from sales to hobbyists

“...most of you steal your software...”

Is this fair? One thing you don't do by stealing software is get back at MITS for some problem you may have had. MITS doesn't make money selling software. The royalty paid to us, the manual, the tape and the overhead make it a break-even operation. One thing you do do is prevent good software from being written. Who can afford to do professional work for nothing? What hobbyist can put 3-man years into programming, finding all bugs, documenting his product and distribute for free? The fact is, no one besides us has invested a lot of money in hobby software. We have written 6800 BASIC, and are writing 8080 APL and 6800 APL, but there is very little incentive to make this software available to hobbyists. Most directly, the thing you do is theft.

What about the guys who re-sell Altair BASIC, aren't they making money on hobby software? Yes, but those who have been reported to us may lose in the end. They are the ones who give hobbyists a bad name, and should be kicked out of any club meeting they show up at.

I would appreciate letters from any one who wants to pay up, or has a suggestion or comment. Just write me at 1180 Alvarado SE, #114, Albuquerque, New Mexico, 87108. Nothing would please me more than being able to hire ten programmers and deluge the hobby market with good software.

Bill Gates
Bill Gates
General Partner, Micro-Soft

Redmond top man Satya Nadella: 'Microsoft LOVES Linux'

Open-source 'love' fairly runneth over at cloud event



20 Oct 2014 at 23:45, Neil McAllister





MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling ma-

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. Our abstraction is inspired by the *map* and *reduce* primitives present in Lisp

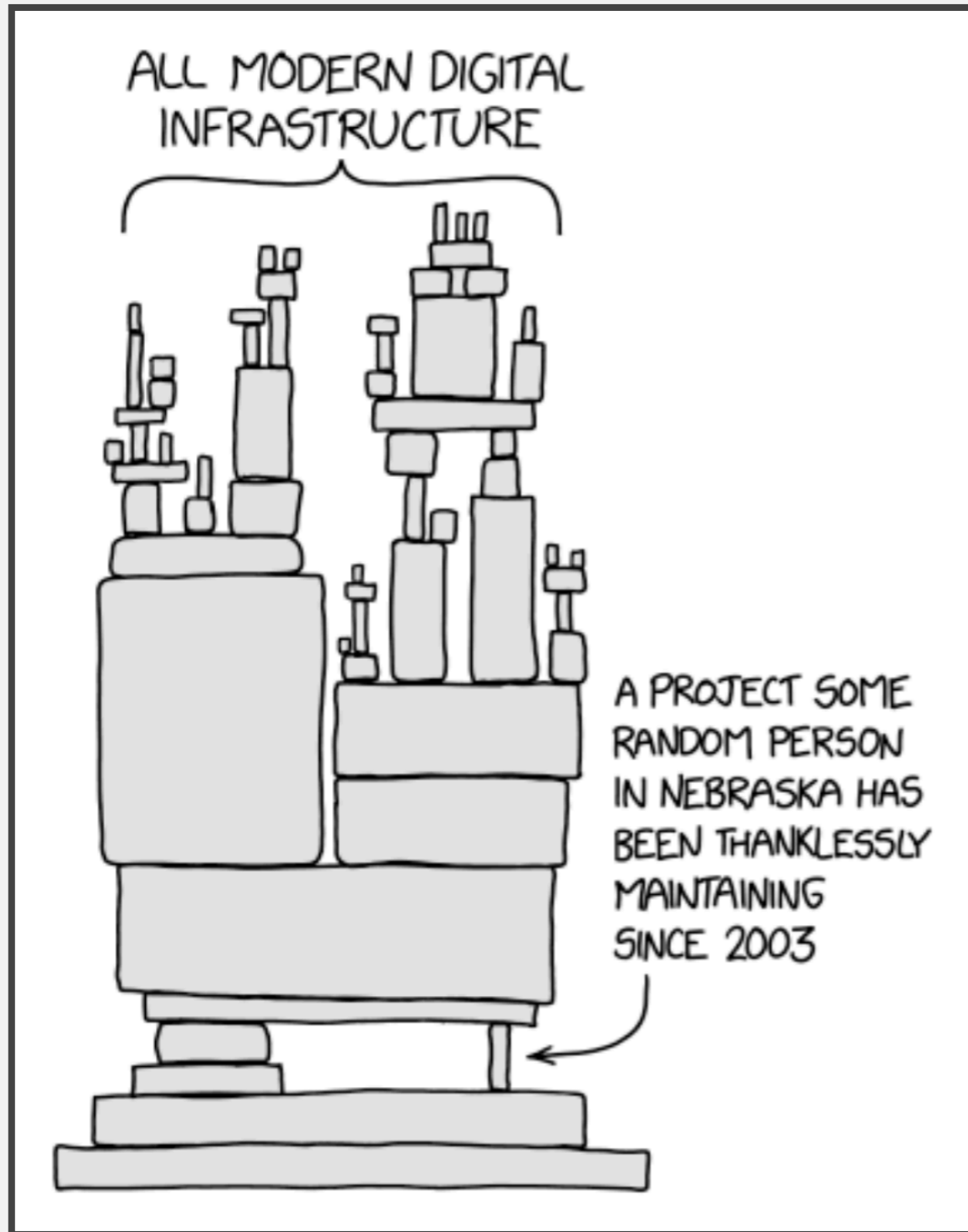


Use of Open-Source Software in Companies



- Is the license compatible with our intended use?
 - More on this later
- How will we handle versioning and updates?
 - Does every internal project declare its own versioned dependency or do we all agree on using one fixed (e.g., latest) version?
 - Sometimes resolved by assigning internal “owners” of a third-party dependency, who are responsible for testing updates and declaring allowable versions.
- How to handle customization of the OSS software?
 - Internal forks are useful but hard to sync with upstream changes.
 - One option: Assign an internal owner who keeps internal fork up-to-date with upstream.
 - Another option: Contribute all customizations back to upstream to maintain clean dependencies.
- Security risks? Supply chain attacks on the rise.

Use of Open-Source Software in Companies



QUARTZ | Make business better.™

HOME LATEST BUSINESS NEWS MONEY & MARKETS TECH & INNOVATION LIFESTYLE LEADERSHIP EMAILS PODCASTS EN ESPAÑOL

TECH & INNOVATION

How one programmer broke the internet by deleting a tiny piece of code

```
1 module.exports = leftpad;
2 function leftpad (str, len, ch) {
3   str = String(str);
4   var i = -1;
5   if (!ch && ch !== 0) ch = ' ';
6   len = len - str.length;
7   while (++i < len) {
8     str = ch + str;
9   }
10  return str;
11 }
12
13
```


Software Licenses

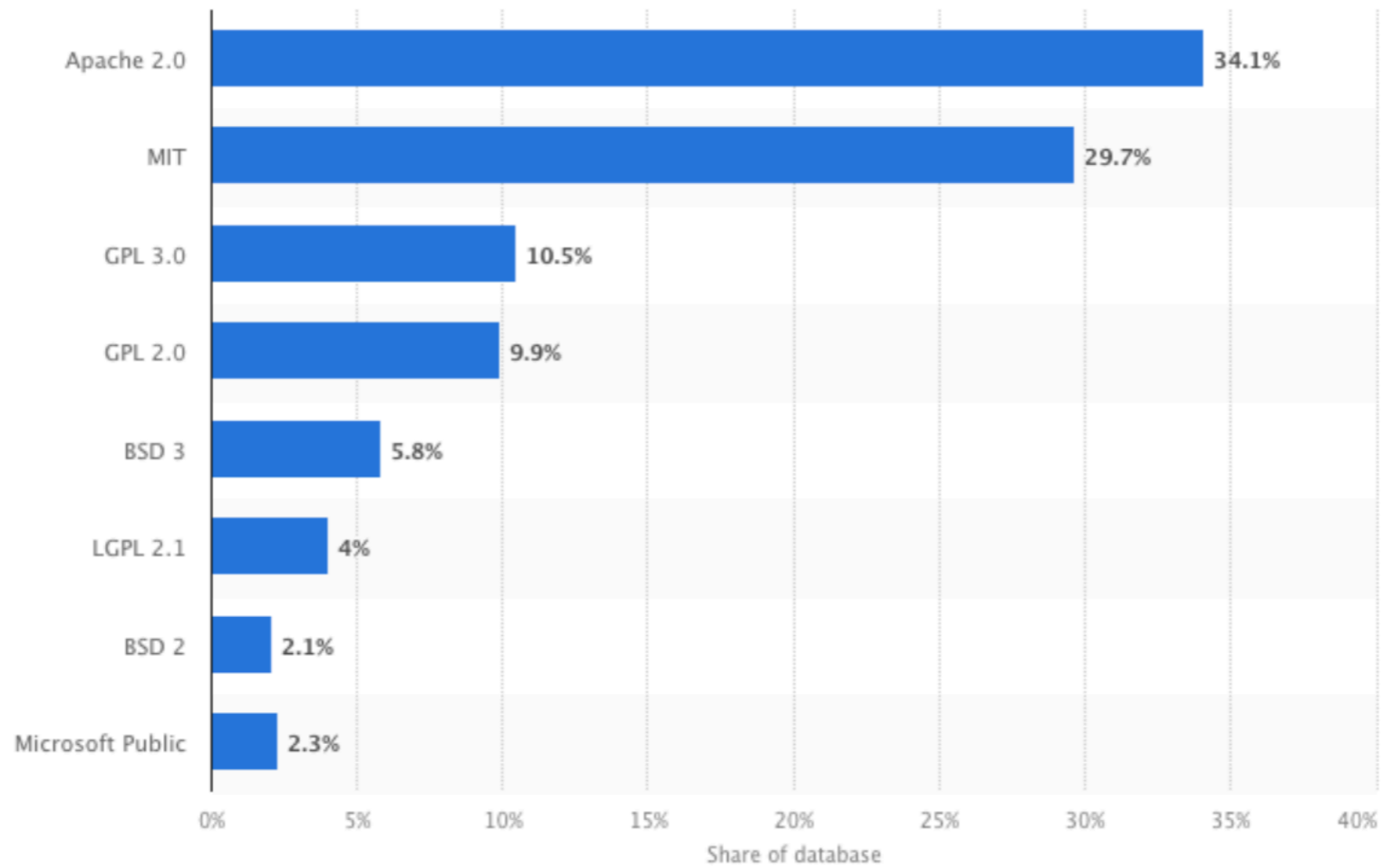


Note: I am not a lawyer (this is not legal advice)

Most popular Software Licenses



Most popular open source licenses worldwide in 2021



© Statista 2023

[Show source](#)

[Additional Information](#)

Which License to Choose?




← → ↻ 🏠 choosealicense.com 📧 546 🔒 🌐 🗄️

Choose an open source license


An open source license protects contributors and users. Businesses and savvy developers won't touch a project without this protection.

{ Which of the following best describes your situation? }


I need to work in a community.


Use the [license preferred by the community](#) you're contributing to or depending on. Your project will fit right in.

If you have a dependency that doesn't have a license, ask its maintainers to [add a license](#).


I want it simple and permissive.

The [MIT License](#) is short and to the point. It lets people do almost anything they want with your project, like making and distributing closed source versions.

[Babel](#), [.NET](#), and [Rails](#) use the MIT License.


I care about sharing improvements.

The [GNU GPLv3](#) also lets people do almost anything they want with your project, *except* distributing closed source versions.

[Ansible](#), [Bash](#), and [GIMP](#) use the GNU GPLv3.

{ What if none of these work for me? }

My project isn't software.

[There are licenses for that.](#)

I want more choices.

[More licenses are available.](#)

I don't want to choose a license.

[Here's what happens if you don't.](#)



- Nobody should be restricted by the software they use. There are four freedoms that every user should have:
 - the freedom to use the software for any purpose,
 - the freedom to change the software to suit your needs,
 - the freedom to share the software with your friends and neighbors, and
 - the freedom to share the changes you make.
- Code must be made available
- Any modifications must be relicensed under the same license (copyleft)



- Nobody should be restricted by the software they use. There are four freedoms that every user should have:
 - the freedom to use the software for any purpose,
 - the freedom to change the software to suit your needs,
 - the freedom to share the software with your friends and neighbors, and
 - the freedom to share the changes you make.
- Code must be made available
- Any modifications must be relicensed under the same license (copyleft)



- Software must be a library
- Similar to GPL but does not consider dynamic binding as “derivative work”
- So, proprietary code can depend on LGPL libraries as long as they are not being modified
- See also: GPL with classpath exception (e.g., Oracle JDK)



- Simple, commercial-friendly license
- Must retain copyright credit
- Software is provided as is
- Authors are not liable for software
- No other restrictions



- Sun open-sourced OpenOffice, but when Sun was acquired by Oracle, Oracle temporarily stopped the project.
- Many of the community contributors banded together and created LibreOffice
- Oracle eventually released OpenOffice to Apache
- LibreOffice changed the project license so LibreOffice can copy changes from OpenOffice but OpenOffice cannot do the same due to license conflicts



- IP and Patents cover an idea for solving a problem
 - Examples: Machine designs, pharma processes to manufacture certain drugs, (controversially) algorithms
 - Have expiry dates. IP can be licensed or sold/transferred for \$\$\$.
- Copyrights cover particular expressions of some work
 - Examples: Books, music, art, source code
 - Automatic copyright assignment to all new work unless a license authorizes alternative uses.
- Exceptions for trivial works and ideas.

Contributor License Agreements (CLA)



- Often a requirement to sign these before you can contribute to OSS projects
- Scoped only to that project
- Assigns the maintainers specific rights over code that you contribute
- Without this, you own the copyright and IP for even small bug fixes and that can cause them legal headaches in the future



- Open-source software harnesses the collective power of stakeholders not directly associated with main developers
- Open-source ecosystems thrive in many application domains where reuse is common (e.g., platforms, frameworks, libraries)
- Corporations rely on open-source even if they develop proprietary software or services.
- Open-source licenses must be chosen carefully to align with intended use case.
- You will all contribute to OSS in this class!