Tragedy of the Commons

Idea In Short

Do not confuse free access to code with free maintenance of code. Open-source software often functions like a digital commons: anyone can use it, modify it and depend on it, while the work of reviewing patches, triaging issues, responding to security vulnerabilities, updating dependencies, documenting behavior and governing releases remains concentrated in a much smaller pool of maintainers. The tragedy of the commons appears when the benefits of open use are widely distributed but the burdens of upkeep remain underprovided, causing the common resource to become fragile. In software, the resource that gets depleted is not the code itself, which can be copied indefinitely, but maintainer attention, institutional support and the human effort required to keep projects safe and reliable. The leadership challenge is therefore not to restrict openness reflexively, but to build contribution and funding systems strong enough to sustain the commons

Open-source software looks inexhaustible. Anyone can copy the code, deploy it, fork it and build on it at almost no distribution cost. That abundance can create a dangerous illusion. While code can be replicated endlessly, maintenance cannot. The tragedy of the commons in open source appears when broad use of shared software is not matched by enough contribution to the human work required to sustain it. The commons at risk is not source code as an object. It is maintainer attention, review capacity, funding and long-term stewardship.

What the tragedy means in open source

The classic tragedy of the commons describes a situation in which individuals acting in their own interest overuse or under-maintain a shared resource until it degrades. In open source, the analogy works only if the scarce resource is identified correctly. The code itself is nonrival in distribution, but maintenance effort is not. Scholars and practitioners therefore describe the open-source problem as a tragedy of the digital commons or a tragedy of non-maintenance, where the shared resource being depleted is human support capacity1.

That distinction matters because it explains why successful projects can become fragile. More usage is normally a positive sign, but when usage growth is not matched by contributor growth, the maintenance burden becomes concentrated. A project can become more important and less sustainable at the same time.

This is why the open-source tragedy is not overconsumption of code copies. It is exhaustion of the people maintaining the system.

Why the analogy fits open source maintenance

Open source invites widespread use by design. Companies and developers adopt libraries and frameworks because they reduce costs, speed development and provide reusable infrastructure. But the direct benefits of adoption go mostly to users and organizations, while the costs of triage, releases, patch review, documentation, dependency upkeep and security response remain concentrated on maintainers.

Multiple open-source analyses describe this as a commons problem in which the benefits of use are broadly distributed but the support burden remains narrow2. The economic asymmetry is simple:

> each user gains privately from adoption, while the cost of under-maintenance is shared diffusely until it becomes severe

That is why self-interest does not reliably produce sustainability. Rational adoption can coexist with collective neglect.

Why maintenance is the true scarce resource

The most important correction in this discussion is that software is not pasture. It does not get depleted by being copied. The scarce commons is maintainer labor. Reviewing pull requests, investigating bugs, answering questions, deciding governance disputes, publishing releases and responding to vulnerabilities all require time and expertise that do not scale automatically with downloads.

This point has been made clearly in both legal and open-source community discussions: in FOSS, the tragedy is underproduction and maintenance failure rather than literal over-harvesting of the software artifact3. Another practical formulation is even sharper:

> the common pool resource in open online projects is effort

That is the real object leaders need to monitor. Not downloads, but maintainable workload.

Why popularity can increase fragility

Popularity sounds like success, but in open source it can intensify the maintenance problem. More users mean more bug reports, more feature requests, more support expectations, more security exposure and more downstream dependencies. If the maintainer base remains small, each new layer of adoption adds obligations faster than the community's support capacity grows.

This is why many foundational packages become hidden infrastructure risks. They may sit deep inside supply chains, relied on by thousands of companies, while being maintained by a handful of volunteers or underfunded contributors. The public value grows, but the support model does not keep pace.

Commentary on open-source sustainability repeatedly points to exactly this pattern: the more successful the commons becomes, the easier it is for maintenance burdens to overwhelm the people carrying them4.

How the problem shows up in practice

The tragedy rarely appears as instant collapse. It appears as slow degradation. Issues accumulate. Pull requests wait. Releases slow down. Security fixes are delayed. Documentation drifts. Maintainers stop responding or quietly leave. Users still experience the project as available, but the resilience underneath is weakening.

This makes the problem easy to ignore until something breaks publicly. A package may

remain usable long after the maintenance model has become unhealthy. By the time organizations notice, the cost of repair is much higher.

Open-source community discussions increasingly frame this as a sustainability crisis tied directly to asymmetry between widespread dependence and concentrated maintenance responsibility5.

Why the free rider problem and tragedy overlap

The tragedy of the commons and the free rider problem are closely related in open source, but they are not identical. Free riding describes the incentive for users, including large firms, to benefit from open-source code without contributing money, labor, or governance support back. The tragedy of the commons describes what happens at the system level when that behavior is widespread enough to deplete maintainer capacity.

This distinction is useful because it separates behavior from outcome. Free riding is one mechanism. Maintainer burnout, under-maintenance and abandonment are the system consequences. In open source, both concepts often point to the same structural imbalance.

Practitioner discussions of the free rider problem in OSS make this link directly, arguing that free riding leaves overworked and underpaid maintainers carrying infrastructure-level responsibility for everyone else6.

What leaders should do

First, leaders should identify critical dependencies and ask who maintains them, under what funding model and with what redundancy. Dependency maps without maintainer health analysis are incomplete.

Second, organizations that derive real value from open source should contribute in proportion to that value. Money matters, but so do engineering time, issue triage, documentation, testing, governance help and security support. Reciprocity must become operational rather than rhetorical.

Third, communities should build institutions around high-value projects. Sponsorship, foundations, staffing, formal stewardship and governance structures can reduce the burden

on individual volunteers. Tragedy is less likely when maintenance becomes a shared responsibility with visible support.

The deeper lesson

The tragedy of the commons matters in open source because it reveals the hidden cost structure of digital abundance. What looks free and scalable at the level of code distribution may be fragile and scarce at the level of human maintenance. The commons survives only if the people doing the maintenance can keep doing it.

That is the executive lesson. In open source, the real shared resource is not the repository. It is the ongoing human capacity to keep the repository trustworthy, secure and alive.

- 1Tragedy of the Digital Commons
- 2The tragedy of the commons and open source software
- 3Free/Open Source Software Development and the Tragedy of the Commons
- 4Open Source Software and the Tragedy of the Commons
- 5Open source and how we sustain ourselves
- 6Addressing open source's free rider problem

Summary

The tragedy of the commons remains a powerful lens for open-source maintenance because it exposes the hidden scarcity behind apparent abundance. Code can scale infinitely at near-zero distribution cost, but triage capacity, security response, release discipline, documentation work and long-term stewardship do not. When firms and developers rely on open-source infrastructure without contributing proportionately, the result is not immediate collapse but growing maintenance debt, slow response, burnout and abandoned projects. Yet tragedy is not inevitable. Communities can organize around governance, reciprocity, funding, sponsorship, institutional stewardship and norms that protect critical maintainers. The strategic lesson is that open source works best when users understand that the real commons is human maintenance capacity and act accordingly