Commons Clause: Understanding an Evolution in Software Licensing

By: Heather J. Meeker

This article was prepared by open source lawyer Heather Meeker on behalf of Redis Labs.

The recent adoption by Redis Labs of the Commons Clause – a restriction on selling software – garnered a lot of attention. This article is intended to help you understand our choice of the Commons Clause, including its rationale, its workings, and its meaning.

The Redis Software

Like many companies, Redis Labs makes available a variety of software elements, some of which are under open source licenses, some of which are under Commons Clause and some of which are under a commercial license. First and perhaps most importantly, Redis, the database, is available under the BSD license and remains under those terms. Redis Modules, which are add-ons to Redis, are able to be created by anyone, and are licensed by their authors as they choose. Certain modules created by Redis Labs (e.g. RediSearch, Redis Graph, ReJSON, Redis-ML, Rebloom) are licensed *Apache 2.0 modified with Commons Clause*. Redis Enterprise is closed source and requires a commercial license from Redis Labs.

Why Commons Clause is Necessary

The reason some businesses have switched from open source licensing to Commons Clause licensing is easy to understand. If a company releases software under an open source license, anyone can use it freely. In many cases businesses are happy to see their software adopted, and they involve the community in development. But some companies – let's call them cloud infrastructure providers or Infrastructure as a Service (IaaS) vendors – have made huge businesses running open source software as a service, without adding anything of value to the software, contributing back to the community, or helping support the development or maintenance of the software. Ultimately, we think that this strategy of free-riding without contributing back will backfire on IaaS vendors. But in the meantime, this free riding is a problem today – the developers of that software can be starved for resources, and the IaaS vendors are not carrying their weight or participating in the community. In economic theory, that kind of situation is referred to as the "Tragedy of the Commons" – when resources are free, they get used, but the users don't have any incentive to contribute to the upkeep of those resources to keep them available and viable for all.

We are not the first to observe this problem. There has been a lot written about this issue from a business point of view. Open source business models are complex and evolving. If you want to know more, see, for example:

- Salil Deshpande of Bain Capital, one of the advocates of the Commons Clause, on the IaaS vendor problem.
- Heather Meeker, on the challenges of the open source business model.
- Dirk Riehle, SAP Research, SAP Labs LLC, on Single-vendor Commercial Open Source models. This article is a more scholarly treatment of commercial open source models.

Commons Clause Prohibits Licensees from Selling Software Without Adding Value
The restriction in the Commons Clause says a licensee cannot Sell the software. Under the law, a clause like this needs to speak for itself, and the drafting team for the Commons Clause tried to make the language of the clause as clear as possible. We can only speak for ourselves, but at Redis Labs, we view the highlighted portions of this definition this way:

> "Sell" means practicing any or all of the rights granted to you under the License to provide to third parties, **for a fee** or other consideration (including without limitation fees for hosting or consulting/ support services related to the Software), a **product or service** whose value derives, entirely or **substantially**, from the **functionality** of the Software.

- **For a fee**: If you charge for the software, or charge for use of the software via IaaS, that is Selling. If you charge for an application using Redis software, that is not Selling. For example, Commons Clause-licensed Redis Graph is a graph database module for BSD-licensed Redis. The Commons Clause does not restrict you from creating applications with Redis Graph (such as an application that visualizes the rate of messages exchanged between a million users on a social media channel). You can also distribute Redis Graph along with your application. You can even make your application available via IaaS. But Commons Clause does not allow you to take Redis Graph, call it MyGraph and offer it via IaaS.
- **Product or service**: This means that Selling is the same whether you market or deploy the software as an on-premises or a cloud-based or IaaS offering. Today, these two technical paradigms are close substitutes for each other. It does not mean that you cannot offer professional services around the software.
- **Functionality**. This word clarifies that ancillary services, such as development or other professional services, are not restricted, because they don't derive value from the functionality of the software. In contrast, IaaS offerings are Selling, because they derive their value from the functionality of the software.
- **Substantially**. If you run an e-commerce business and use RediSearch to help users find products to order, that is not a service that derives its value substantially from RediSearch -- in that case, RediSearch is only a tool to which you have added significant value. On

the other hand, if you run an on-line search company and you use RediSearch as your search engine, even with a thin UI, that is a substantial use of the software, because RediSearch is the engine that executes on most of the value of your search functionality.

Is Commons Clause Open Source?
Commons Clause is not open source. It is a new hybrid licensing model.

Open Source has a specific definition, which is managed by the Open Source Initiative. That definition has 10 requirements, and to be "open source," a license has to meet all of them. One of them (Number 6) prohibits "discrimination" against "fields of endeavor." In software licensing, that "discrimination" translates into what is called a "field of use" restriction. Open source licenses have no license *restrictions*, only *conditions*. Commons Clause is a license *restriction* that covers a particular field of endeavor, which is selling the software without adding value. (For more on the nature of the restriction, see below.)

Commons Clause meets most of the requirements of the Open Source Definition (free redistribution, source code availability, ability to make derived works, and others), but not all of them. So, it is not open source. It is a source-available license. In contrast, "proprietary" has no specific definition, but a proprietary license is usually a limited license to end use of binary code only, and does not include the availability of source code. So, Commons Clause is a hybrid model – using some of the characteristics of open source and one important characteristic (a field of use restriction) of proprietary software.

Why AGPL is not Sufficient
Some companies have tried to address the free-riding concern by using AGPL as one option in a dual licensing model. The AGPL was written to compel IaaS vendors to share changes. But it does not solve our business problem, because in fact, most IaaS vendor do not change the code they use – so they have no obligation to share anything and are free-riding. The failure of AGPL to address this problem does not mean open source has failed, it just means that AGPL does not solve our problem. Many users of open source voluntarily contribute back to community development even when the license does not compel them to do so. But the IaaS vendors have not done this.

AGPL does not work for us because it applies too many other conditions. AGPL imposes significant conditions on all users of the code, even those who are not IaaS vendors. Most private companies still "black list" AGPL code because they do not have the infrastructure to comply with AGPL. Their open source compliance processes are usually triggered when the software is ingested into their organization for use in a distributed product. However, AGPL's source code offer requirements are triggered when a company changes the software. In most organizations, trapping, and triggering a legal review on, each modification is too much overhead.

So AGPL is, at the same time, too much and too little: it doesn't curb the actions of the IaaS vendors, but it would limit adoption by others.

Creative Commons Non-Commercial
Some people have correctly observed that the Creative Commons "Non-Commercial" or "NC" license also addresses commercial free riding. However, Creative Commons licenses are not drafted as software licenses. Moreover, the "non-commercial" designation is too broad a restriction. It says you cannot use the material licensed for activities "primarily intended for or directed toward commercial advantage or private monetary compensation." That is much broader than the Commons Clause restriction. Also, many people wonder about what it means. See for example the wiki discussion here. Finally, CC-NC, disallows applying "technical measures" restricting use of the software, and that is not an issue with which we want to burden those using our software.

Changing AGPL
If there is an effort to change AGPL to address concerns like ours, we will follow it with great interest. But it is not clear to us that AGPL ever could be changed enough to address the IaaS vendor problem and still fit the Open Source Definition. Even if it could, it is unlikely that a revision could be done soon enough to meet our business needs. The drafting of GPL3/APL3 took over two years and involved a great deal of controversy. Getting any revision of an open source license approved by FSF and OSI would be a long and possibly contentious process.

Full Proprietary Licensing
Full proprietary licensing is another option that could certainly address the free riding SaaS vendor problem. Companies like Cockroach DB and Elastic have their own licensing models that include limited licenses. We don't intend to speak for or against them or their license models. We chose Commons Clause because we wanted to give our users some freedoms that fully proprietary licenses don't offer. We also want our developers to have free access to our code, and only limit use by IaaS vendors.

Corporate Policy and Commons Clause Software
Developers may wonder if their companies will approve use of Commons Clause software. We hope so, and are happy to work with potential users and developers to understand the Commons Clause. As a baseline, your company should be no less likely to approve use of Commons Clause software than use of "freeware" or other limited licenses. The Commons Clause restriction is a narrow one, restricting only one kind of use. It is unlikely to affect most users and developers. For example, if you want to use Redis modules to run your own business, or to build an application on top of it, you won't be affected. But you can't sell the Redis modules, either as

distributed software or as an IaaS offering. If you want to use the Redis software in a way that would be restricted by Commons Clause, please contact Redis Labs to discuss alternative licensing terms.

Commons Clause and Parallel Licensing Documents
Commons Clause is a restriction that is added to a base open source license. A company releasing code under the Commons Clause can pick any base open source license; Redis Labs picked Apache 2.0 -- a permissive license. Although Commons Clause's "portmanteau" approach is somewhat new, putting license terms in multiple documents is a long-standing practice.

For example, almost all proprietary software products today impose at least two simultaneously applicable sets of license terms. Suppose a vendor sells a product FOOBAR with some files covered by Apache 2.0 and the rest proprietary. In that case, the licensee must abide by both sets of terms: the Apache 2.0 terms only cover some portions of the product, and those rights are granted by the authors of the Apache 2 portions. The proprietary terms cover the FOOBAR product as a whole, and those terms are presented by the vendor. This model covers almost every proprietary product in the market today.

Conceptually, we think this very common paradigm is more difficult to understand than Commons Clause. Commons Clause applies both sets of terms at once to an entire software package. The licensee needs to comply with both at once: the Apache 2 license grants a broad set of rights with some conditions, and the Commons Clause imposes a restriction.

Writing New Licenses
Some have commented that they disapprove of combining Commons Clause and existing licenses. The alternative is to write an entirely new proprietary license. Deciding between these two options is not easy. But writing entirely new licenses would cause more proliferation and make Commons Clause licensing harder to understand. Software licensing works best when it can be described in a few words. No one wants to spend time reviewing software licenses, they just want to use great software. We think the "portmanteau" approach is easy to understand: Apache 2.0 + Commons Clause. It is harder to understand an entirely new license, or -- considering the range of open source licenses that could be used as a baseline for Commons Clause -- a whole new set of licenses. Commons Clause also allows vendors to pick the underlying license to suit their needs -- copyleft, network copyleft, permissive, as they choose.

Why Redis Labs Chose Apache 2.0 + Commons Clause instead of BSD (or MIT) + Commons Clause
Apache 2.0 is a popular permissive license because it grants an express patent license. That gives both licensors and licensees a baseline level of comfort about patent issues, by increasing certainty as to the scope of patent rights being granted. BSD and MIT licenses are not

standardized, not clearly drafted, and don't say anything about patents. At Redis Labs, we were not interested in using a copyleft license, mainly because permissive licensing has worked well for us in the past, other than the free-riding problem, of course. We did not think it was necessary to impose additional conditions like those of GPL to accomplish our goals.

A New Clause, but an Old Idea
Commons Clause is only new in that it standardizes the language of a license restriction and relies on an open source license for the license grant. License restrictions against commercial use are extremely common -- for example, Creative Commons Non-Commercial. Value add requirements are also very common in proprietary software licenses. This is not a new idea, just a new implementation of it.

Commitment to Commons Clause
All license paradigms are, in a certain sense, works in process. Over time, the needs of a business change, and when they do, it is best for the business if it can change its license terms to suit them. If better solutions, even open source solutions, come along, we will consider them. But we needed to solve our business challenges now. If the community addresses our concerns by releasing new licenses -- open source or not, we will be ready to embrace those choices. We know a lot of other companies share our concerns. Commons Clause is an experiment, but it works for us now.

Criticism of Commons Clause
There has been a flurry of commentary about Commons Clause -- some critical, some positive. Criticism has sometimes been informed and thoughtful, and unfortunately, sometimes not so. We don't think we need to answer every criticism, but the attention from our release of Commons Clause software has been so significant, that we thought we should make a few observations. The criticism seems to be in a few categories:

- **It isn't open source**. That is correct. Some people believe, as a moral principle, that all software should be open source. That's their opinion, and they have a right to express it, but it is a moral judgment, not a business judgment. As owners of the software, we have the right to choose our licensing terms.

- **It is confusing**. Anything new can be confusing. We have written this article to explain some things about Commons Clause. But even so, we did not adopt Commons Clause in an effort to trick anyone into using code thinking it is open source or under Apache 2.0. We have been transparent and clear about that. Also, candidly, most of the complaints we have received, claiming that people are confused by Commons Clause, have been hearsay (claiming that others have been confused), or from people who clearly did not bother to read our licensing statement at all. (For example, some people complained that we applied a Creative Commons license. The only similarity between that and Commons Clause is their initials, meaning that those complaining did not even bother to read the title of the clause we applied -- much less its brief terms.).

- **It is license proliferation**. It is a new model, but the alternative is proprietary licenses, which are customized to every product. Commons Clause thus actually limits license proliferation by allowing companies to easily adopt a lightweight license restriction without writing an entirely new license, and to use a license restriction that is short, simple, and easy to understand.

- **It violates the rights of Apache**. Apache 2.0 and Commons Clause are independent documents, but we have been clear that both apply simultaneously to our code. We have not changed the Apache license, we have used it letter for letter. Some people think that means the Apache license has been changed, but that argument is more sophistry than common sense. Combining two things does not mean that each has been modified. We have not claimed that Apache Foundation endorses our license or our product. And we think that changing the name of the Apache license without changing a word of it is more confusing, rather than less so. In trademark law, what we did is called a nominative use -- which is sometimes called a "fair use" -- and it does not require permission.

- **It is unlawful/confusing to combine existing licenses with new ones**. As we mentioned above, we have not seen any actual confusion among those who actually read our terms. It is not unlawful to combine licensing terms. Some people do not like this aspect of Commons Clause, because they think it threatens the nature of open source licensing. But we know that open source licensing is healthy, happy, and will be fine on its own - Redis itself remains BSD-licensed but we just are not offering it for the optional Redis Labs modules.

Note: This article was prepared by Heather Meeker on behalf of Redis Labs. It expresses the views of Redis Labs. Any references to other companies, whether or not they have adopted Commons Clause, are informational only and do not represent the views of those companies. Questions about this article may be directed to legal@redislabs.com.