Bild: KI Midjourney | Bearbeitung c't

Eine Revolution frisst ihre Kinder

Abkehr von klassischen Open-Source-Lizenzen

Open Source war einmal angetreten, um Nutzern die größtmögliche Freiheit im Betrieb von Software zu garantieren und Software-Titanen den Kampf anzusagen. Inzwischen haben die aber gelernt, die verbrieften Freiheiten zu ihrem Vorteil zu nutzen.

Von Kristian Köhntopp

Es brodelt in der Open-Source-Gemeinde: In jüngerer Vergangenheit lässt sich beobachten, dass etablierte Softwareunternehmen mit großer Nutzerbasis die Freiheiten der Nutzer ihrer Open-Source-Projekte durch einen Lizenzwechsel einschränken. Prominentes Beispiel ist das Unternehmen HashiCorp, das auf Software für Cloud-Infrastruktur spezialisiert ist und seine Kernprodukte unter die BSL v1.1 (Business Source License) gestellt hat, eine formal nicht offene Lizenz. Die Quittung kam prompt: Inzwischen wurde die Software als OpenTofu geforkt und steht jetzt unter der Schirmherrschaft der Linux Foundation.

HashiCorp ist nicht die erste Firma, die so etwas tut: Red Hat hat vor längerer Zeit die Positionierung von CentOS als freie Alternative zu Red Hat Enterprise Linux (RHEL) geändert und kürzlich die Herausgabe der RHEL-Quelltexte durch eine EULA (End User License Agreement) eingeschränkt, Elastic hat ElasticSearch und Kibana unter die formal nicht offene SSPL (Server Side Public License) gestellt,

und auch MongoDB ist nur noch unter der SSPL zu haben.

Noch viel früher gab es Umzüge von der GPLv2 zur GPLv3, die Einführung der AGPL, und einige Firmen wie Apple haben ein striktes GPLv3-Verbot für ihre eigenen Softwareprodukte ausgesprochen und migrieren interne Software komplett von der GPL weg. Sind diese Entwicklungen eine Bedrohung für Open Source? Und was sollen diese Lizenzänderungen eigentlich bezwecken? Um die Konfliktlinien zu verstehen, lohnt sich ein Blick in die Vergangenheit.

Vier Freiheiten und eine Zertifizierung

Open Source und Free Software gab es schon, bevor diese Worte dafür existierten. Doch die eigentliche „Open-Source-Revolution" begann in den 1990er Jahren mit dem Linux Kernel und den sich daraus ergebenden Distributionen. Der Begriff „Open Source" wurde von Bruce Perens und Eric Raymond geprägt und in der Open-Source-Definition und den Debian Free Software Guidelines formalisiert.

Beides sind Verfeinerungen und Klarstellungen der ursprünglichen „Vier Freiheiten" der Mutter aller Open-Source-Lizensen, der GNU General Public License (GPL), die da lauten:

- Freiheit 0: Die Freiheit, das Programm für jeden Zweck auszuführen.
- Freiheit 1: Die Freiheit, die Funktionsweise des Programms zu untersuchen und eigenen Bedürfnissen der Datenverarbeitung anzupassen. Das impliziert Zugang zum Quelltext.
- Freiheit 2: Die Freiheit, das Programm weiterzugeben. Die GPL fordert aber, dass auch der Quelltext des Programms, der dem weitergegebenen Programm entspricht, angeboten wird.
- Freiheit 3: Die Freiheit, das Programm zu verbessern und diese Verbesserungen unter der originalen Lizenz weiterzugeben. Auch dies impliziert Zugang zum Quelltext und das Recht, diesen zu ändern.

Während die GPLv2 die Lizenz ist, die diese Freiheiten das erste Mal kodifiziert hat, haben sich schnell eine ganze Reihe von Lizenzen entwickelt, die alle „Open Source" nach den OSI-Richtlinien (Open Source Initiative) oder den DFSG (Debian Free Software Guidelines) sind und von denen einige zwar frei, aber inkompatibel mit der GPL sind. Das hat insbesondere in den späten 90er Jahren viele Usenet-Newsgroups dauerhaft mit Diskussionen gefüllt.

Der LAMP-Stack: Eine wilde Mischung von Lizenzen

Der bekannte LAMP-Stack für die Webentwicklung ist auch ein Beispiel für den Lizenzwirrwarr, mit dem man es bei Open Source zu tun haben kann. LAMP beinhaltet den Linux Kernel, den Apache Webserver, die MySQL-Datenbank und die Programmiersprache PHP. All diese Bausteine haben unterschiedliche Lizenzen.

Während der Linux-Kernel GPLv2-lizenziert ist, steht der Apache Webserver unter der Apache License 2. Diese ist freizügig („permissive"), das heißt, es fehlen Bestimmungen, die erzwingen, dass Veränderungen gegenüber dem Originalprogramm unter derselben Lizenz wie das Originalprogramm weitergegeben werden müssen.

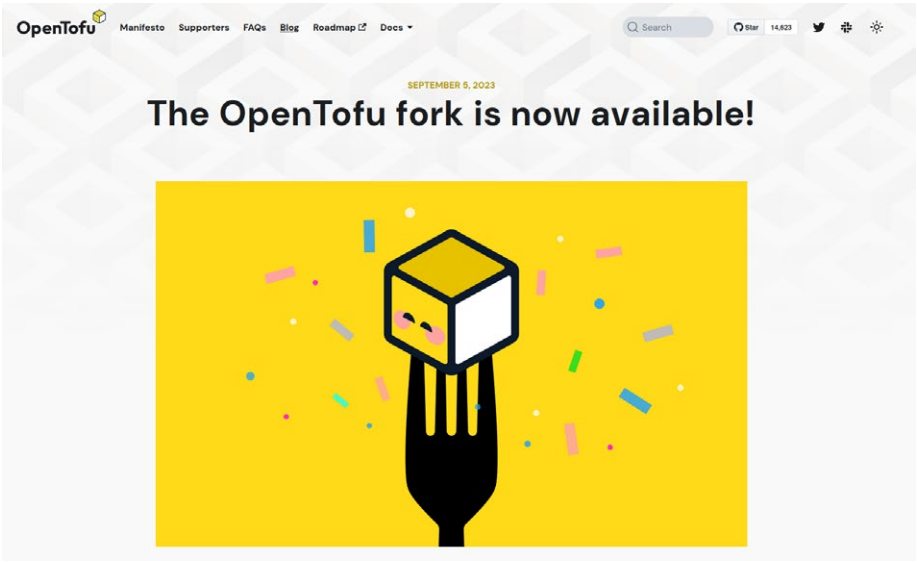
MySQL hat gleich zwei Lizenzen: Die Datenbank kann unter der GPL genutzt werden und unterliegt dann den Bestimmungen und Freiheiten, die oben genannt wurden. Der Urheberrechtsinhaber ist jedoch eine Firma (ursprünglich MySQL, dann SUN und jetzt Oracle), die alle Rechte innehat. Entwickler müssen die Rechte an ihren Patches per Contributor License Agreement (CLA) an Oracle abtreten. Dadurch kann Oracle den Quelltext auch unter einer kommerziellen Lizenz anbieten.

c't kompakt

- In letzter Zeit wenden sich vermehrt Projekte von klassischen Open-Source-Lizenzen ab.
- SaaS-Geschäftsmodelle erschweren die nachhaltige Entwicklung von Open-Source-Software.
- Cloudanbieter richten die Freiheiten, die Open-Source-Lizenzen gewähren, gegen deren Entwickler.

Und PHP steht seit Version 4 unter der PHP-License. Auch das ist eine freizügige Lizenz, die Nutzung und Weitergabe von PHP mit oder ohne Modifikationen, als Quelltext oder ausführbares Programm erlaubt, jedoch Einschränkungen hinsichtlich des Gebrauches der Wortmarke „PHP" enthält.

Das führt zu einer wichtigen Frage der späten 90er Jahre: Was passiert, wenn man Software mit unterschiedlichen Lizenzen mischt und zusammen in einer Distribution verbreitet oder sie in einem Übersetzungsprozess zusammen linkt? Das GNU Projekt hat relativ schnell klargestellt, dass die Grenzen der Lizenz die Prozessgrenzen sind. Wenn man also ein Programm unter einer anderen Lizenz entwickelt und mit GNU-Software zusammen linkt, sodass das Resultat hinterher in ein-



Nachdem HashiCorp angekündigt hatte, sein Infrastruktur-Werkzeug Terraform unter eine nicht freie Lizenz namens BSL zu stellen, formierte sich schnell Widerstand in der Open-Source-Community. Inzwischen gibt es einen Fork namens OpenTofu (vormals OpenTF).

und demselben Prozess ausgeführt wird, dann geht das legal nur dann in Ordnung, wenn die andere Lizenz mit der GPL kompatibel ist.

Obwohl die GPL eine Open-Source-Lizenz ist, sind nicht viele Lizenzen mit der GPL kompatibel. Das liegt daran, dass die GPL die Freiheit wahrende Einschränkungen enthält: Man kann die Ausführung der Software nicht einschränken („Nicht in Atomraketen!“) und ein Programm, das GPL-Komponenten enthält, muss auch im Quelltext mit allen Änderungen angeboten werden. Das bedeutet: nicht nur die GPL-Komponenten, sondern alle Komponenten in diesem Prozessraum.

Die Debatte um die „infektiöse“ GPL

Aus der Pflicht zur Veröffentlichung der Quellen ergab sich dann eine weitere Diskussion in den frühen Nullerjahren. Damals wurden Linux und Open Source so groß, dass man sie nicht ignorieren konnte, und eine Menge Start-ups kokettierten mit Open Source. Die GPL war damals zwar die am weitesten verbreitete Lizenz, aber wenn man ein Start-up betreibt, das sich durch Risikokapital finanziert, dann ist es keine attraktive Aussicht für Investoren, alle Quellen der eigenen Software unter der GPL herausgeben zu müssen.

Aus dieser Zeit stammt der Anti-Open-Source-Slogan der „viralen“ oder „ansteckenden“ GPL, der im Kern jedoch Unsinn ist: Zum einen ist die GPL leicht einzuhegen, denn sie endet ja klar definiert an Prozessgrenzen, andererseits sind die Bestimmungen der GPL sinnvoll, weil sie eine direkte Ausbeutung der Leistung anderer erschwert („Quid pro quo: Wenn Du von Open Source profitierst, dann sollst Du auch auf die gleiche Weise dazu beitragen.“).

GPLv3 gegen Softwarepatente

Venture Capitalists brachten als Reaktion eine weitere Bedrohung für Open Source in Stellung, auf die reagiert werden musste: Softwarepatente. Wenn Start-ups ihre Software unter der GPL herausgeben müssen, können sie ihr geistiges Eigentum eventuell anders einschränken, um später dafür Lizenzgebühren einzutreiben. Damals waren das Softwarepatente, die als reale Bedrohung positioniert wurden.

In der Rechtfertigung für die Version 3 der GPL wird dies auch direkt thematisiert, und die GPLv3 enthält Regelungen, die vereinfacht besagen: „Wenn Du Patente verwendest, um Benutzer von beliebiger GPLv3-lizenzierter Software zu verklagen, dann erlischt Dein Recht, jegliche GPLv3 Software zu verwenden.“ Diese Klausel ist wirksam und sie hat unter anderem dazu geführt, dass Apple jede Form von GPLv3-lizenzierter Software in seinen Produkten vermeidet und sukzessive jegliche GPLv3-Software in Apple-Produkten durch anders lizenzierte Software ersetzt.

Die Bestimmung, dass die GPL an Prozessgrenzen endet, ist auch für das GNU-Projekt manchmal problematisch, und so gibt es Varianten der GPL und Ausnahmen. Zum Beispiel ist generierter Code von unter der GPL stehenden Programmen nicht GPL-geschützt: Dies ermöglicht Codegeneratoren wie gcc, bison und flex. Und für einige Bibliotheken, darunter die glibc, gibt es die LGPL, auch „Lesser GPL“ genannt, die es erlaubt, die Bibliotheken gegen Nicht-GPL-Software zu linken.

„GPL statt LGPL“ als Waffe

Andersherum haben einige Firmen die GPL als Waffe eingesetzt. Zum Beispiel waren Bibliotheken wie die „libmysqlclient.so“ bis einschließlich MySQL 3.23 unter der LGPL verfügbar, aber ab Version 4 stehen sie unter der GPL. Linkt man diese Bibliothek also in sein kommerzielles Programm und will es ausliefern, dann muss man eine kommerzielle MySQL-Lizenz unter dem Dual-Licensing Programm kaufen. MySQL hat das im Slogan „If you are free, we are free. If you are commercial, we are commercial“ zusammengefasst.

Das war eine Zeit lang ein Problem für die Programmiersprache PHP, die ja gegen diese Bibliothek gelinkt hat, um auf die Datenbank im Rahmen des LAMP-Stacks zugreifen zu können, aber eine formal nicht GPL-kompatible Lizenz hat. PHP ist dem Problem mit einer Re-Implementierung des Protokolls als „mysqlnd“ begegnet, und parallel dazu hat MySQL das Problem gelöst, indem sie eine Lizenz-Ausnahme für PHP (und andere) definiert haben.

Weniger GPL, mehr MIT, BSD und Apache

All diese Wirrungen sind ein Grund, warum „moderne“ Open-Source-Software meist nicht mehr die GPL verwendet, sondern andere Lizenzen wie MIT, BSD oder Apache bevorzugt. Sie machen eine Reihe von kapitalfinanzierten Geschäftsmodellen um Open Source einfacher, wenn man eine Firma um ein (oft nur dem Namen nach) Open-Source-Produkt stricken will. Ab 2005 beginnt die Blütezeit von auf Open Source basierenden Geschäftsmodellen. Das hat 15 Jahre lang gut genug funktioniert, um eine Reihe von Projekten groß zu machen.

In den 2000er-Jahren sah sich Softwaregigant Microsoft durch GPL und Open Source bedroht. An Werbeanzeigen wie dieser wird deutlich, dass der Kampf proprietäre versus freie Software früher mit härteren Bandagen ausgetragen wurde.

Einige dieser Projekte sind nicht auf einzelne Firmen als Träger angewiesen: PHP, KDE und Postgres sind zum Beispiel stabile und groß angelegte OSS-Projekte, hinter denen nicht eine einzelne große Firma und ihre Venture-Kapitalgeber stehen – sie alle zeichnen sich durch eine breite und vielfach verankerte Basis von Beitragenden aus.

Seit einigen Jahren zeigen sich Risse: Red Hat, Elastic, MongoDb, MariaDB (ein MySQL-Fork), HashiCorp und viele andere haben ihre Lizenzen geändert oder sich umstrukturiert, um einen bestimmten Use-Case auszuschließen. Da dies eine Verletzung der Freiheit 0 darstellt, sind es damit auch keine Open-Source-Projekte im Sinne der OSI und der DFSG mehr.

Der Elefant im Raum ist die Cloud

Amazon Web Services (AWS) ist ein System und eine Firma, die stellvertretend für ein Geschäftsmodell steht: Es monetarisiert den in Unternehmen oft vernachlässigten und unterfinanzierten Aspekt „Operations“ und sieht vor, Software für Dritte nicht zu entwickeln, sondern vorrangig zu betreiben.

Damit ist Amazon sehr erfolgreich: Schon länger laufen in dessen Rechenzentren mehr Neuinstallationen von MySQL (und MongoDB, Postgres oder Elastic und vielen anderen Datenbanken) als lokal beim Kunden („on premise“). Dies ist eine Klasse von Systemen, die notorisch schwierig zu betreiben sind, weil Fehler beim Betrieb Daten unwiederbringlich ruinieren können. Natürlich kann man eine fehlerhafte Version einer Datenbank auf die vorhergehende Version zurückrollen, aber die Daten, die durch den Fehler zerstört worden sind, sind immer noch dauerhaft weg.

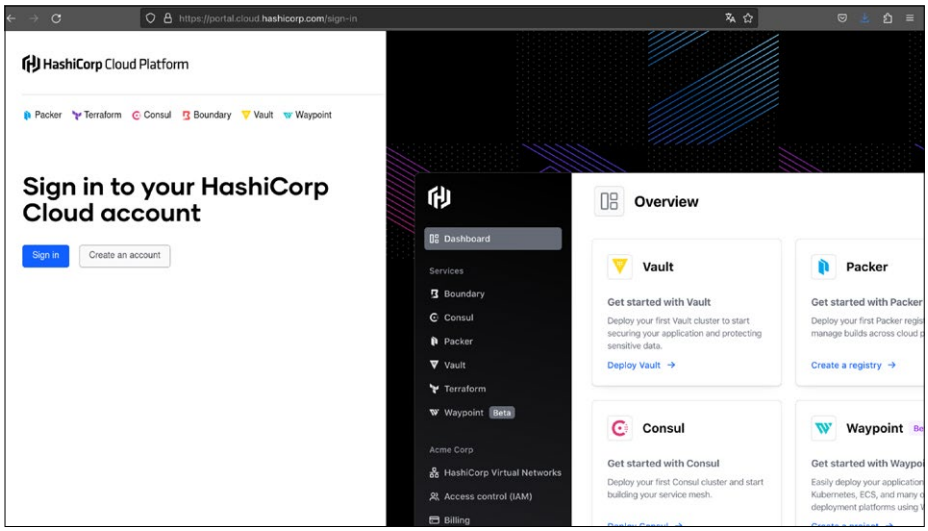
Die systematische Unterschätzung der Kosten und des Aufwandes von Operations macht das Prinzip „Software as a Service“ (SaaS) erfolgreich. SaaS-Anbieter verkaufen nicht nur einen Dienst, sondern im Grunde genommen vornehmlich ein Betriebskonzept für diesen Dienst, der einen reibungslosen und unterbrechungsfreien Betrieb direkt nach der Inbetriebnahme erlaubt. Weil das Betriebskonzept erprobt und vorgeschrieben ist, kann man die Qualität der Dienstleistung standardisieren, messen und dann stabile Verträge mit Dritten darüber eingehen.

Kunden begrüßen das, denn Software zu betreiben ist ein Problem, das mindestens so schwierig ist wie Software zu entwickeln. Viele Kunden wollen sich damit nicht auseinandersetzen. Oft werden dabei Verfügbarkeiten, Antwortzeiten und Kosten in Kauf genommen, die genau so vorher bei den eigenen Inhouse-Teams vollkommen inakzeptabel waren, nur um beispielsweise den Aufgabenkomplex „Betrieb der Datenbanken“ aus dem Haus zu bekommen – das zeigt, wie unangenehm Firmen die Auseinandersetzung mit „Operations“ ist.

Dabei ist der Betrieb dieser Software durch AWS nicht durch eine kommerzielle Vereinbarung zwischen dem Hersteller der Software und Amazon gedeckt, sondern erfolgt auf der Grundlage der Freiheit 0 der GPL: Amazon darf die Software zu jedem beliebigen Zweck ausführen, auch als Service für Dritte. In Folge streicht Amazon das Geld für den Betrieb dieser Installationen ein, aber der eigentliche Softwareentwickler, die Herstellerfirma oder -gruppe bekommt davon nichts. Das ist kein nachhaltiges Finanzierungsmodell, aber durch die GPL gedeckt und sehr vorteilhaft für AWS, sodass man dort auch keine Veranlassung sieht, etwas daran zu ändern.

Business Source License als frühe Reaktion

Das ist schon recht früh als Problem erkannt worden: Bei MariaDB findet man die erste Erwähnung der „Business Source License“ (BSL) im Blog von Monty Widenius schon 2013, und MariaDB setzt sie für bestimmte Komponenten seit 2016 ein, mit leichten Veränderungen nach Anregungen von Bruce Perens dann seit 2017 in der Version 1.1. Andere Produkte ziehen nach und gehen ebenfalls unter die BSL: Couchbase, Uptrace, Kurtosis, Sentry, CockroachDB und neuerdings alle HashiCorp-Kernprodukte.



HashiCorp betreibt die HashiCorp Cloud Platform (HCP), wo Kunden Terraform und Co. als Managed-Service nutzen können. Der Wechsel zur BSL verbietet es Mitbewerbern, Konkurrenzprodukte auf Basis von Terraform zu entwickeln.

Die Idee hinter der BSL ist, dass die Software weiterhin frei nutzbar und der Quelltext verfügbar bleibt, aber mit Einschränkungen. Die wichtigste Einschränkung meint üblicherweise „AWS darf das nicht nutzen“, formuliert als „Die Software darf nicht als SaaS für Dritte angeboten werden“. Man kann die Software also weiter so verwenden wie zuvor. Man kann sie auch in AWS selbst betreiben oder man kann sie innerhalb der eigenen Organisation für andere Abteilungen bereitstellen.

Aber man wird den Betrieb der Software nicht als Dienst von einem Cloudbetreiber kaufen können, sondern nur vom Ersteller der Software selbst (etwa via AWS Marketplace). Diese Einschränkung dient dazu, die Finanzierung der Entwicklung der Software sicherzustellen, indem AWS die Möglichkeit genommen wird, diesen Geldstrom vollständig für sich selbst abzuzweigen. Die BSL kommt mit einer weiteren Regel, die besagt, dass Codeänderungen spätestens nach vier Jahren unter eine anerkannte Open-Source-Lizenz fallen (meist die Apache-2.0-Lizenz oder die GPL), sodass ältere Versionen der Software auf jeden Fall immer Open Source sind.

Formal ist BSL-Software keine Open Source im Sinne der OSI oder DFSG, weil anders als bei der GPL die Freiheit 0 („Ausführen für welchen Zweck auch immer“) eingeschränkt wird. Man bezeichnet sie deshalb auch als „Source Available“. Auch die von MongoDB, Elastic und Kibana verwendete Server Side Public License verfolgt dieselbe Idee, nur anders: Die SSPL ist eine AGPLv3 Variante, die ebenfalls die Freiheit 0 einschränkt und verbietet, die Software für Dritte („as a Service“) zu betreiben, es sei denn, man erwirbt eine kommerzielle Lizenz.

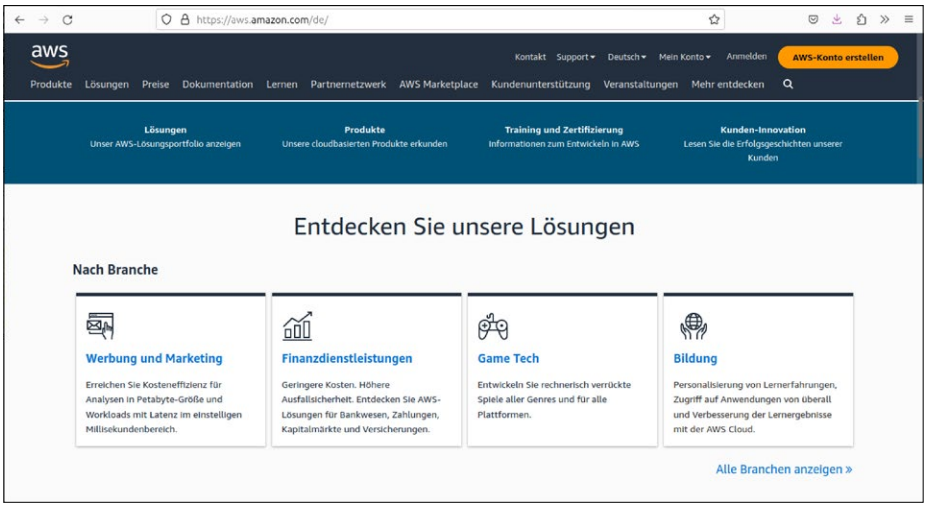
Unfreie freie Software

Die Grundidee hinter den Freiheiten der GPL ist Geben und Nehmen: „Du kannst unsere Software nutzen, musst aber selbst gutartig mitspielen.“ Die immer weiter wachsende Stellung von AWS macht es den Erstellern von Open Source zunehmend schwerer, ihren Unterhalt mit der Entwicklungsarbeit zu verdienen – AWS bietet deren Software als Service an und verwebt ihn mit weiteren Dienstleistungen. Die Freiheit 0 der freien Software wendet sich hier gegen die Ersteller der Software.

Die BSL und die SSPL sind eine Reaktion darauf. Sie sind formal gesehen keine freie Software nach OSI oder DFSG. Aber vielleicht ist es auch für Open Source wieder einmal an der Zeit, den Veränderungen und den neuen Bedrohungen ins Auge zu sehen und die Lizenzen und die Definitionen der neuen Zeit anzupassen, so wie dies mit der GPLv3 und der AGPL schon einmal geschehen ist.

Vielleicht bewegen sich FSF, OSI und Debian aber auch nicht und beharren auf ihren Definitionen von „freier Software“. Die Entwickler von Software unter der BSL und der SSPL ficht das nicht an: Ihre Software ist jetzt schon „fast frei“ und wegen ihrer speziellen Regel nach einigen Jahren dann auch formal freie Software nach alter Definition. Einen Grund, die BSL oder SSPL abzulehnen, gibt es nicht, ausgenommen Fundamentalismus für freie Software. Aber wenn diejenigen, denen etwas am Open-Source-Gedanken liegt, nicht wollen, dass AWS die einzige Firma mit einem erfolgreichen Open-Source-Geschäftsmodell ist, muss sich etwas ändern. Sonst landen Nutzer schon bald wieder bei langweiliger, traditioneller kommerzieller Software, wie man es früher von Microsoft und Oracle gewohnt war. *(ndi@ct.de)* ct

Liste von Open-Source-Lizenzen nach OSI-Definition, OpenTofu-Manifest:
ct.de/yfyh



Immer mehr Unternehmen mieten Ressourcen wie Datenbanken von Cloudanbietern wie AWS an, anstatt eigene Server zu betreiben. Die Entwickler von Open-Source-Software gehen dabei oft leer aus.