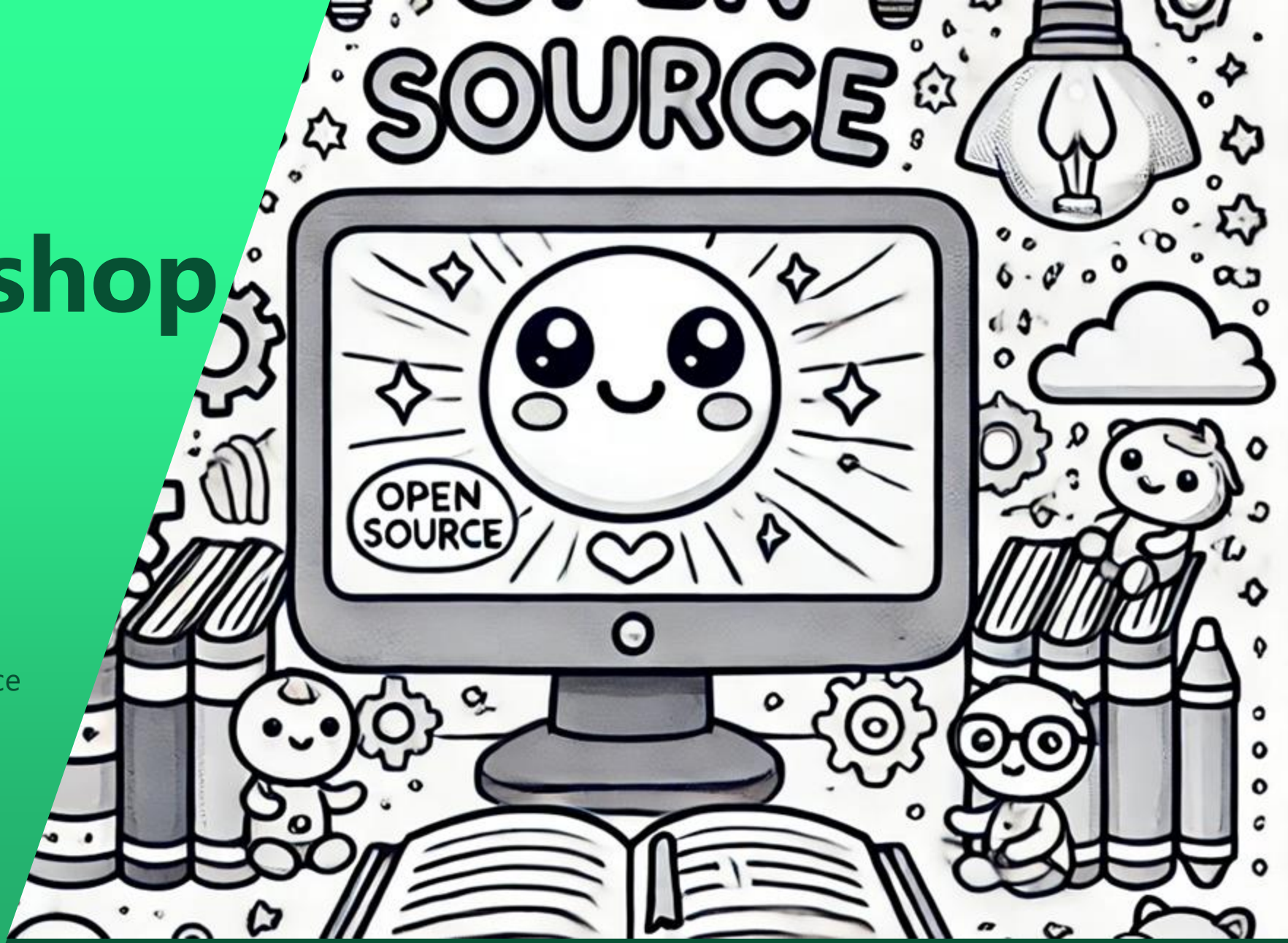
bechtle

OSS-Workshop for Sales

Was ich schon immer über Open Source wissen wollte



Hybrider Workshop

11.02.2025

Kurze Vorstellung

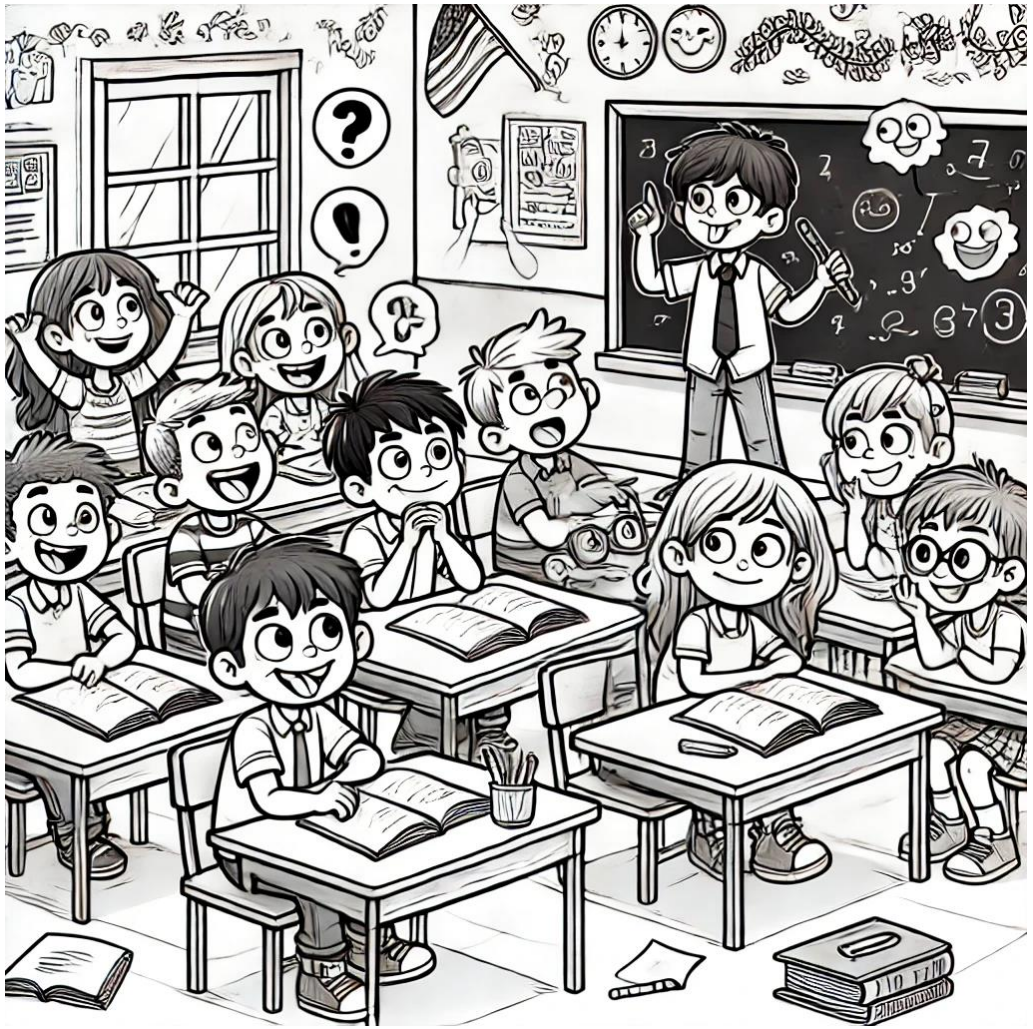
- Manfred Stendel
- Ltr. Technical Sales Consulting / Solution Sales
- Bei Bechtle seit 2020

Gedachter Ablauf

1. **Eröffnung**
2. **Was ist Open Source**
3. **Wie entwickelt sich Open Source**
4. **Was sind Anwendungsfälle beim Kunden**
5. **Wie kann ich Themen beim Kunden ansprechen**

Organisatorisches

- Inhalte wurden in Teilen via Perplexity erstellet
- Zeichnungen sind via Coloring Book Hero erstellt
- Weitere Quellen sind im Transkript hinterlegt
- Zu jedem Thema machen wir eine Fragerunde !



Die Frage geht an euch ☺

Was ist Open Source ?

Eine mögliche Definition I

Open Source Software (OSS) bezeichnet Software, deren Quellcode öffentlich zugänglich ist. Dies ermöglicht es Nutzern, den Code einzusehen, zu verändern und weiterzugeben. Die grundlegenden Prinzipien von Open Source basieren auf -...

Merkmale von Open Source Software

1. Offenlegung des Quellcodes: Der Quellcode ist für jeden einsehbar und verständlich, meist in einer höheren Programmiersprache verfügbar.

2. Freiheit zur Nutzung und Weitergabe: Es gibt keine Einschränkungen hinsichtlich der Nutzung oder Verbreitung der Software, solange die Lizenzbedingungen eingehalten werden.

3. *Anpassbarkeit*: Nutzer können den Quellcode verändern und die modifizierte Version weitergeben, **oft ohne Lizenzgebühren**.

4. *Dezentrale Entwicklung*: Open Source Projekte werden häufig gemeinschaftlich entwickelt, basierend auf Peer-Review und Community-Beiträgen. Transparenz, Kollaboration und Freiheit in der Nutzung und Weiterentwicklung der Software.

Und...

Eine mögliche Definition II

im Gegensatz zu proprietärer Software (Closed Source) erlaubt Open Source Software vollständigen Zugriff auf den Quellcode. Proprietäre Software hingegen schränkt die Nutzung, Veränderung und Weitergabe ein und wird durch Urheberrechte geschützt.

Quellen

[https://github.com/mstendel/OSS_workshop/blob/main/transcript.md

Ohne Lizenzkosten …. leider nicht immer !

Lizenzierung nach Anzahl der Kerne (Core-Based Licensing):
•Bei Container-basierten Bereitstellungen wird pro vCPU (virtuelle CPU) des Containers lizenziert.
•SQL Server benötigt mindestens 4 Core-Lizenzen pro Container, auch wenn der Container weniger vCPUs nutzt.
•Jede virtuelle Core-Lizenz berechtigt, einen Container auszuführen.

Hier der Nachweis aus den PTs:

Nutzung von SQL Server mit Containertechnologie
Für Zwecke der Lizenzierung der Nutzung von SQL Server-Software, die innerhalb eines Containers in einer Container-Laufzeitumgebung wie Docker, cri-o oder containerd zum Laufen kommt, wird (i) ein Container als Virtuelle Betriebssystemumgebung und (ii) die für diesen Container verfügbaren Physischen oder Virtuellen Cores als Hardwarethread betrachtet. Die Nutzung durch den Kunden unterliegt dem Pro-Kern-Lizenzmodell oder dem Server/CAL-Lizenzmodell und allen anderen Lizenzbestimmungen, die für die SQL Server-Lizenzen relevant sind, die der Kunde dem Lizenzierten Server in Verbindung mit dieser Nutzung ordnungsgemäß zugewiesen hat. Zur Verdeutlichung: Wenn Hyperthreading aktiviert ist und der Kunde die Nutzung nach dem Lizenzmodell „Virtuelle OSE pro Kern“ lizenziert, muss der Kunde eine Kernlizenz für jeden Hardwarethread zuweisen, der einem Container zugeordnet ist, wobei eine Mindestanzahl von vier Lizenzen erforderlich ist.

Zusammenfassung

Open Source Software (OSS) ist frei zugängliche Software, deren Quellcode jeder einsehen, verändern und weitergeben darf. Sie fördert Transparenz, Zusammenarbeit und Innovation, bietet Unternehmen Flexibilität und reduziert Abhängigkeiten.

weitergeben darf… aka Lizenzen

Welche Lizenztypen gibt es ?

Permissive Lizenzen (z. B. MIT, Apache): Einfach, flexibel, keine starken Einschränkungen – gut für kommerzielle Nutzung.

Copyleft Lizenzen (z. B. GPL, LGPL): Streng, erfordert Offenlegung des Quellcodes bei Derivaten.

Hybrid-Lizenzen (z. B. MPL): Kombination von Offenheit und geschäftlicher Flexibilität.

Lizenz	Lizenztyp	Kommerzielle Nutzung erlaubt	Quellcode-Offenlegung	Patentschutz	Modifikation erlaubt	Kompatibel mit proprietärer Software	Typische Einsatzbereiche
GNU GPL	Copyleft	Ja, aber Quellcode offenlegen	Erforderlich	Nein	Ja, aber unter GPL	Nein	Betriebssysteme, Anwendungssoftware (Linux, WordPress)
MIT	Permissiv	Ja	Nicht erforderlich	Nein	Ja	Ja	Webentwicklung, Bibliotheken (React, Node.js)
Apache 2.0	Permissiv	Ja	Nur für modifizierte Versionen	Ja	Ja, mit Hinweis auf Änderungen	Ja	Cloud, Big Data, Infrastruktur (Hadoop, Kubernetes)
BSD (2-Clause, 3-Clause)	Permissiv	Ja	Nicht erforderlich	Nein	Ja	Ja	Netzwerksoftware, Unix-Tools (FreeBSD, OpenSSH)
Creative Commons (CC)	Kreativ (Variabel)	Ja, je nach Variante	Variiert	Nein	Variiert	Variiert	Texte, Bilder, Multimedia (Wikipedia, Flickr)
Mozilla Public License (MPL)	Hybrid	Ja	Nur für modifizierte Versionen	Nein	Ja, aber unter MPL	Ja, mit Einschränkungen	Enterprise-Software, Entwicklungstools (Firefox, Thunderbird)
LGPL (Lesser GPL)	Schwaches Copyleft	Ja	Nur für Änderungen der Bibliothek	Nein	Ja, Bibliothek bleibt LGPL	Ja, mit Einschränkungen	Bibliotheken für gemischte Nutzung (FFmpeg, GTK)
Public Domain (Unlicense)	Keine Einschränkungen	Ja	Nicht erforderlich	Nein	Ja	Ja	Freigegebene Softwareprojekte, Open Data

Der Unterschied zwischen Lizenz und Subskription

Lizenz:

- Einmalige Zahlung für dauerhaftes Nutzungsrecht
- Keine regelmäßigen Kosten
- Updates & Support meist separat
- Weiterverkauf möglich (unter bestimmten Bedingungen)

Subskription:

- Zeitlich begrenztes Nutzungsrecht (monatlich/jährlich)
- Regelmäßige Zahlungen erforderlich
- Laufende Updates & Support enthalten
- Kein Eigentum, kein Weiterverkauf

✅ **Fazit:**
Lizenzen bieten ein einmaliges, dauerhaftes Nutzungsrecht, während Subskriptionen flexible, laufend aktualisierte Lösungen mit regelmäßigen Kosten darstellen.

???

Fragen

bechtle

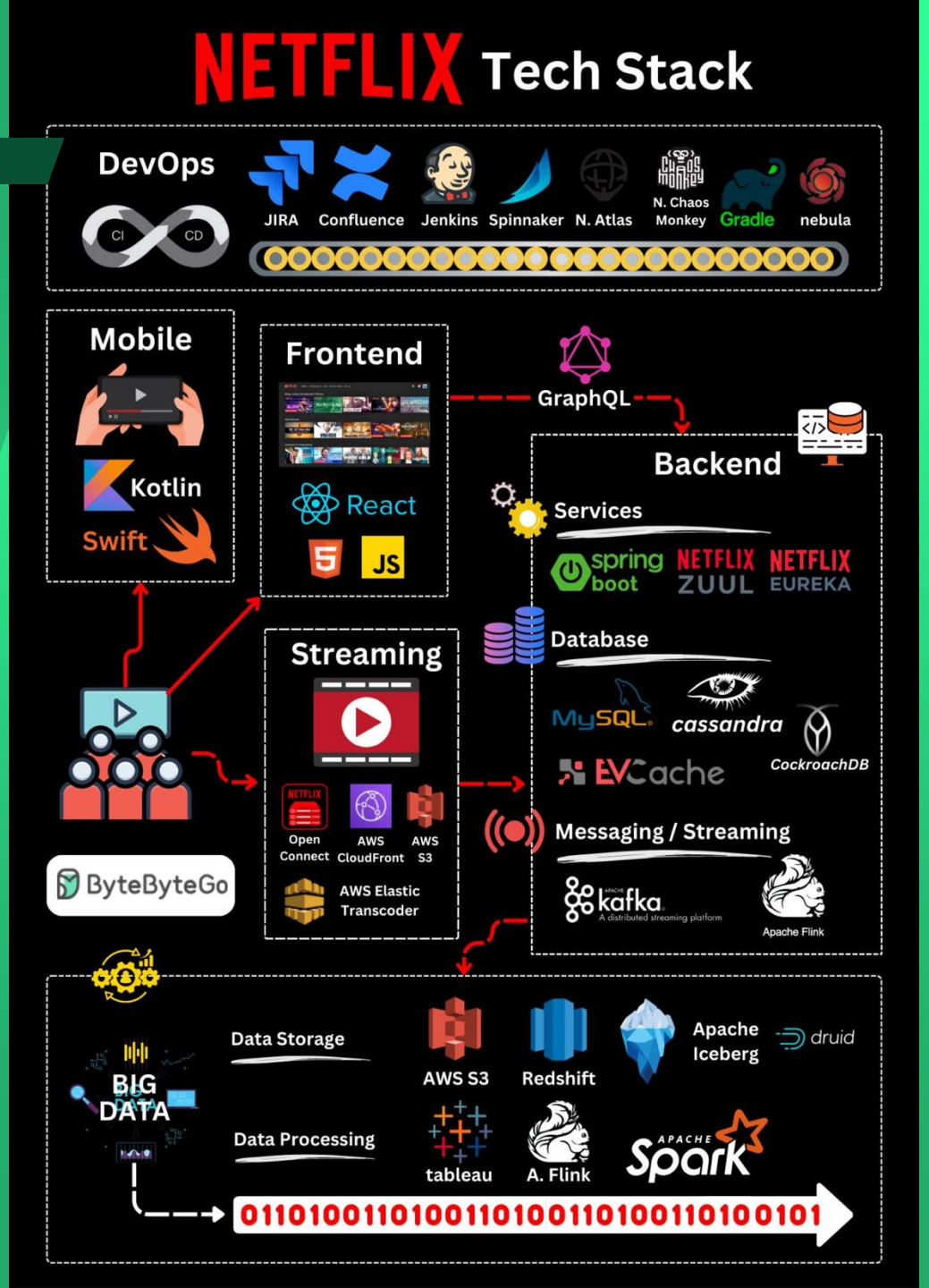
Wie entwickelt sich Open Source ?

Evolution ist ein Vorteil

11.02.2025

Keine Angst...

so wird es nicht



Etwas Geschichte

Die Geschichte von Open Source entspricht der des Internets

In den 1950ern und 1960ern verwendeten die Forscherinnen und Forscher, die die ersten Internettechnologien und Telekommunikationsnetzwerkprotokolle entwickelten, eine offene und kollaborative Umgebung. Das Advanced Research Projects Agency Network (ARPANET), das später zur Basis des modernen Internets werden sollte, hat das Prinzip von Peer-Review und offenen Feedback-Prozessen begünstigt. Nutzergruppen tauschten ihren Quellcode untereinander aus und entwickelten den Quellcode der anderen weiter. Foren erleichterten den gegenseitigen Austausch und die Entwicklung von Standards für eine offene Kommunikation und Kollaboration. Zur Geburtsstunde des Internets Anfang der 1990er waren Werte wie Zusammenarbeit, Peer-Review und Offenheit bereits fester Bestandteil seines Fundaments.

Quelle:https://github.com/mstendel/OSS_workshop/blob/main/transcript.md

Merkmale der OSS-Entwicklung

- **Kollaborative Entwicklung:**

OSS wird in der Regel von einer Vielzahl von Entwicklern und Unternehmen gemeinsam entwickelt. Dies geschieht oft über Plattformen wie GitHub, wo Projekte öffentlich zugänglich sind und Peer-Reviews sowie Community-Beteiligung zentral sind.

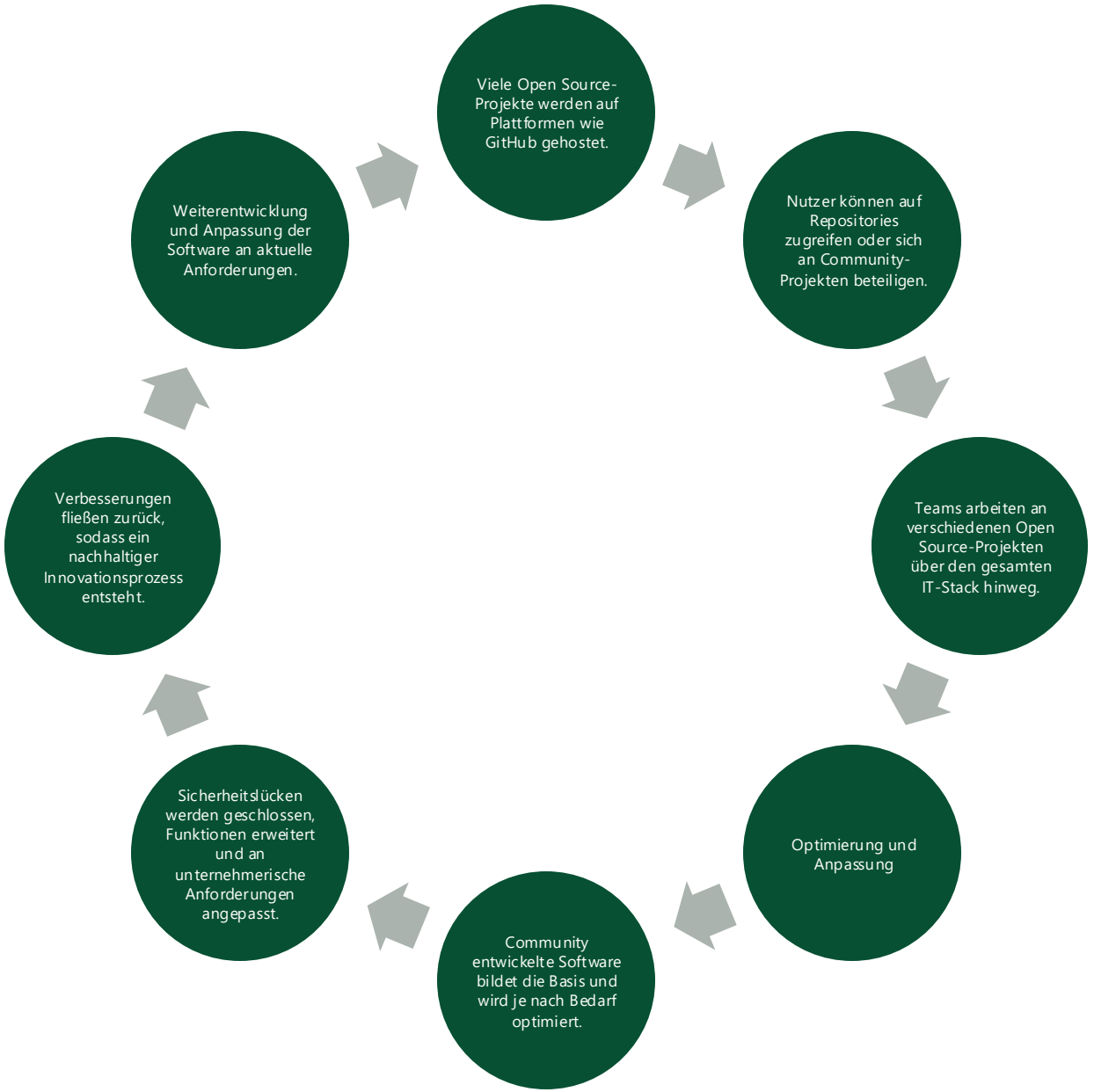
- **Dezentralisierung**:

Die Entwicklung ist nicht auf ein einzelnes Unternehmen oder eine Person beschränkt. Dadurch wird der Entwicklungsaufwand geteilt, und die Software bleibt unabhängig von bestimmten Anbietern.

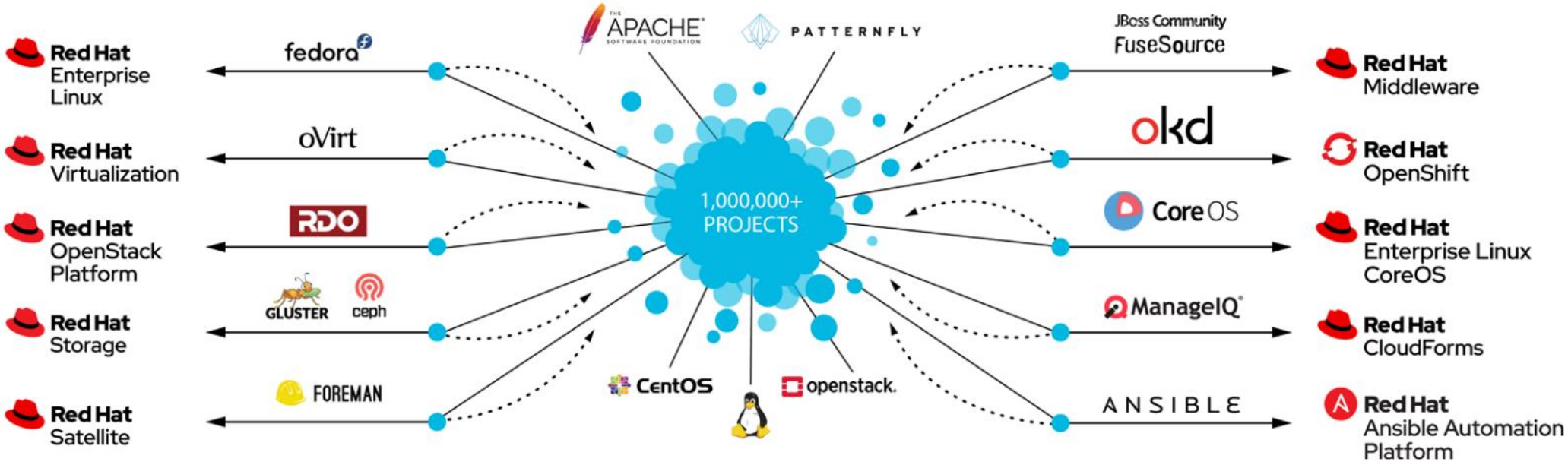
- **Community-Engagement**:

Unternehmen und Entwickler tragen durch Codebeiträge, Fehlerbehebungen oder neue Funktionen zur Weiterentwicklung bei. Verbesserungen werden häufig der gesamten Community zur Verfügung gestellt, was einen Kreislauf von Geben und Nehmen schafft.

Kreislauf der OSS-Entwicklung



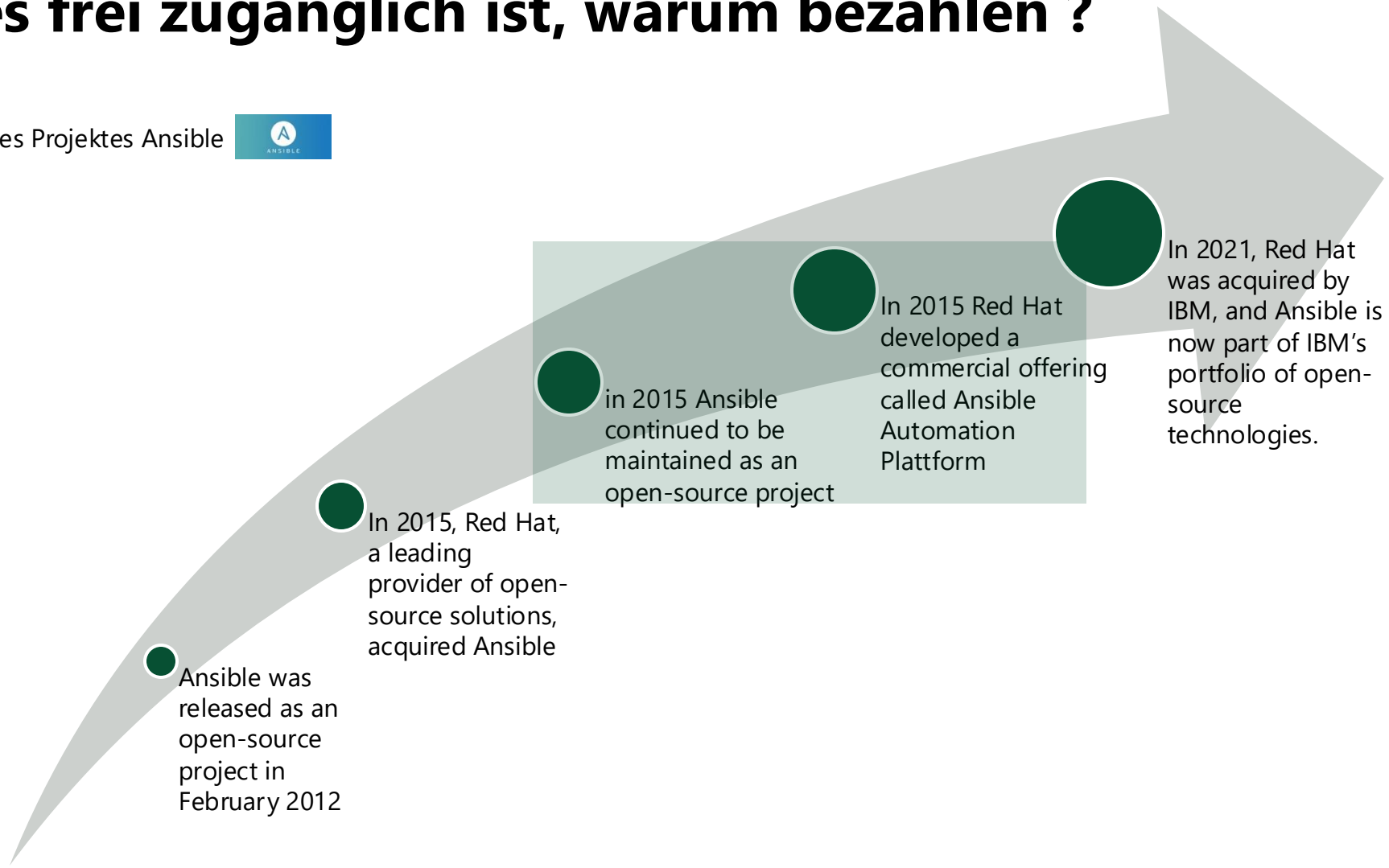
Vom Projekt zum Produkt



Quelle: https://www.redhat.com/en/resources/frequently-asked-questions-about-open-source-software-communities

Wenn alles frei zugänglich ist, warum bezahlen ?

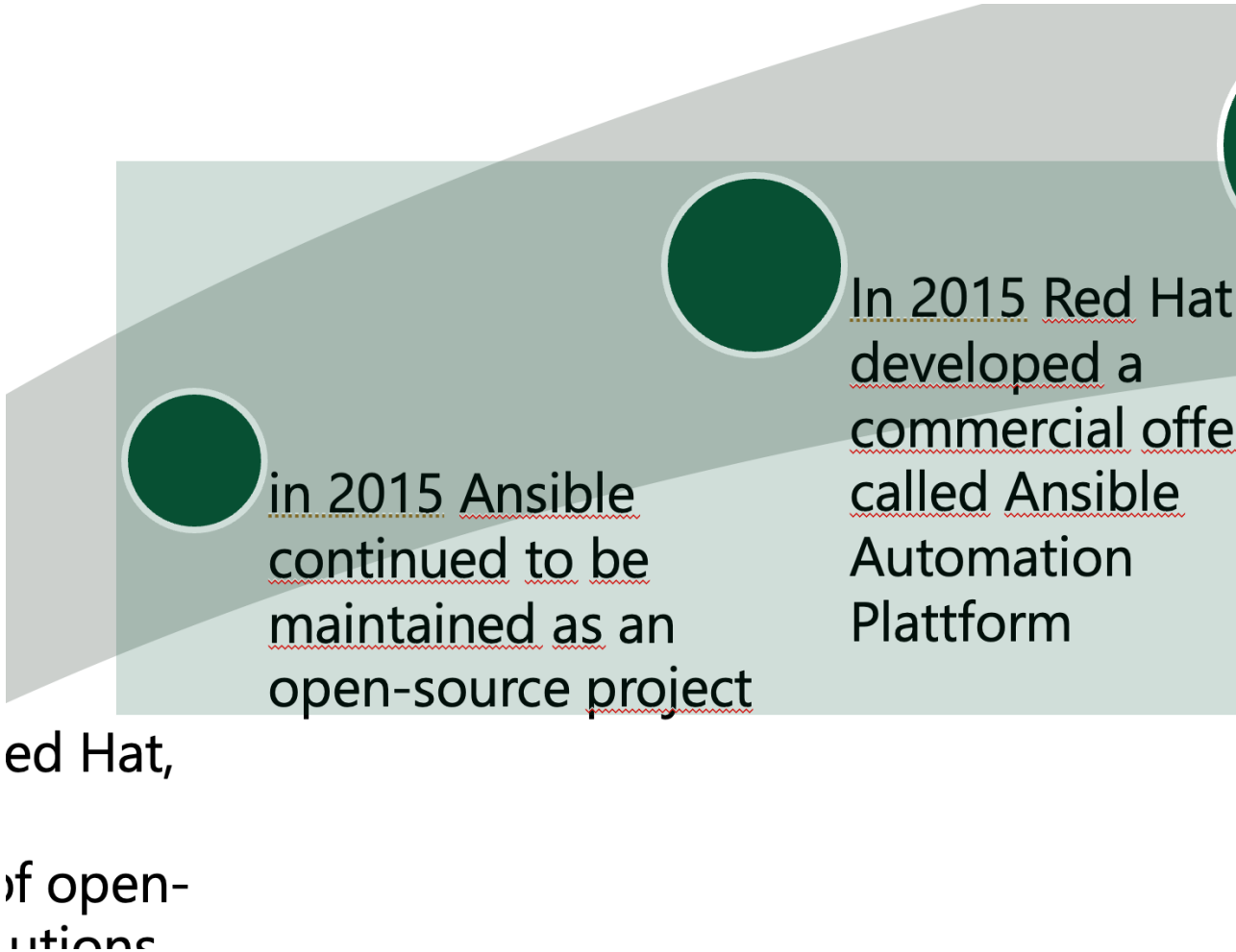
Eine Erklärung anhand des Projektes Ansible



Wo Unternehmen unterstützen

Was macht ein Unternehmen im Open Source Projekt:

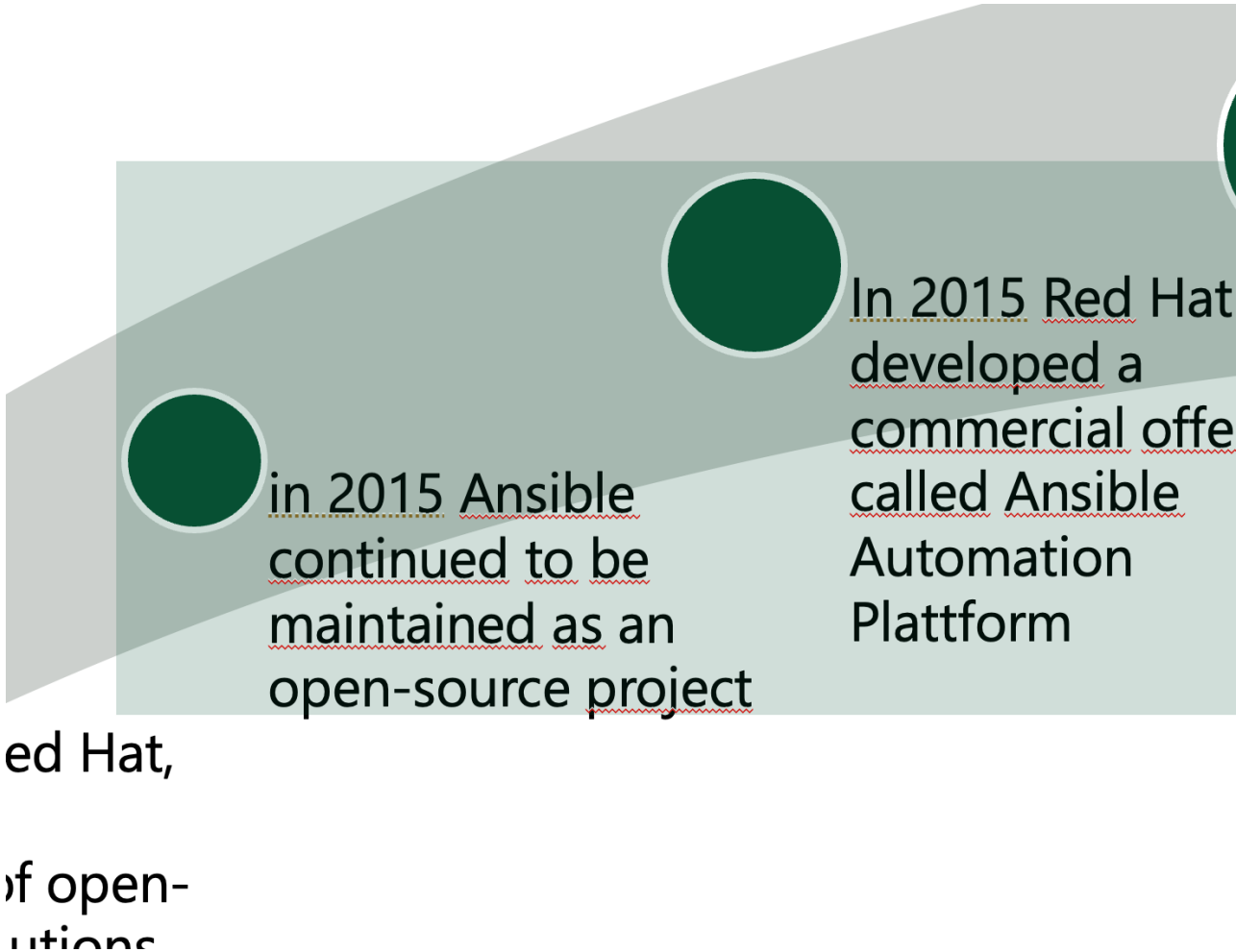
- Es stellt eigene Entwickler, die an dem Open Source Projekt arbeiten
- Pflegt die Dokumentation
- Stellt Repositorys für Module zur Verfügung
- Pflegt die Softwarebibliothek mit aktuellen Versionen und Features



Was bezahle ich dann ?

Im Beispiel Ansible beinhaltet eine Subskription:

- Support durch den Hersteller
- Zertifizierte Pakete und Module
- Geschützte Lieferketten
- Security Updates
- Geprüfte Dokumentation
- *Bei Ansible Automation auch die Betriebssysteme*



bechtle

Schattenseiten von OSS

Ein Beispiel aus 2024

11.02.2025

XZ Utils Backdoor – Einer der gefährlichsten Open-Source-Angriffe 2024

XZ Utils ist eine Open-Source-Bibliothek für **Datenkomprimierung**, die das **.xz-Format** verwendet. Sie ist eine **wichtige Abhängigkeit** in vielen Linux-Distributionen und wird unter anderem von **OpenSSH** genutzt.

Typische Anwendungen

- **Software-Pakete & Systemdateien:** Viele Linux-Distributionen nutzen .xz, um Speicherplatz zu sparen.
- **Datenarchivierung:** Langfristige Speicherung von großen Dateien mit hoher Effizienz.
- **Datenübertragung:** Weniger Datenvolumen bedeutet schnellere Downloads & geringere Netzwerklast.

Modus Operandi

Die Hintertür wurde von einem **bösartigen Open-Source-Maintainer** über einen **längeren Zeitraum schrittweise eingeführt**.
Der Angriff erfolgte in mehreren Stufen:

1.Infiltration des Projekts

1. Ein Entwickler mit dem Pseudonym **"JiaT75"** wurde als Maintainer des Projekts aktiv.
2. Über Monate hinweg gewann er das Vertrauen der Community, indem er regelmäßige Updates veröffentlichte.

2.Manipulation des Codes

1. In den Versionen **5.6.0 und 5.6.1** von XZ Utils wurde **bösartiger Code versteckt**.
2. Die Änderungen waren **hochgradig verschleiert**, sodass sie nicht sofort auffielen.

3.Exploit über OpenSSH

1. OpenSSH nutzt XZ Utils für die Datenkompression.
2. Der eingeschleuste Schadcode ermöglichte es Angreifern, **authentifizierte Verbindungen zu hijacken und Root-Rechte zu erlangen**.
3. Dadurch hätten sich Angreifer **remote in betroffene Systeme einloggen können, ohne gültige Zugangsdaten zu besitzen**.

Wie wurde der Angriff entdeckt

•Ein Entwickler namens **Andres Freund** (Principal Software Engineer bei Microsoft) bemerkte **ungewöhnlich hohe CPU-Last** auf Systemen mit der neuesten Version von XZ Utils.

•Nach genauerer Untersuchung stellte er fest, dass der OpenSSH-Daemon (sshd) **modifiziert worden war**, um ein verborgenes Backdoor-Verhalten zu ermöglichen.

•Sofort wurden Linux-Distributionen gewarnt, die Versionen **5.6.0 und 5.6.1 zu sperren**.

Zeitlicher Ablauf

Datum	Ereignis
Okt 22	Ein neuer Contributor mit dem Pseudonym **"JiaT75"** beginnt, in Open-Source-Communities aktiv zu werden.
2023 - Anfang 2024	Der Contributor gewinnt Vertrauen und wird einer der **Haupt-Maintainer von XZ Utils**.
Feb 24	Erste **manipulierte Code-Änderungen** werden in den Entwicklungszweig von XZ Utils eingeführt.
06. Mär 24	Die Version **XZ Utils 5.6.0** wird veröffentlicht, die bereits die **Backdoor enthält**, jedoch noch keine sichtbaren Auswirkungen zeigt.
13. Mär 24	Die Version **XZ Utils 5.6.1** wird veröffentlicht, mit weiteren Verfeinerungen des schädlichen Codes.
Ende März 2024	Debian Testing und Fedora Rawhide **beginnen, die infizierten Versionen in ihren Repositories auszuliefern**.
28. Mär 24	**Andres Freund bemerkt hohe CPU-Last in OpenSSH** und beginnt mit seiner Analyse.
29. Mär 24	Nach intensiver Untersuchung **entdeckt Freund die versteckte Backdoor in liblzma**.
30. Mär 24	Debian, Fedora und andere Linux-Distributionen **stoppen sofort die Verteilung von XZ Utils 5.6.0 und 5.6.1**.
01. Apr 24	Öffentliche Sicherheitswarnungen werden veröffentlicht, und die Schwachstelle erhält die **CVE-2024-3094**.
Anfang April 2024	**XZ Utils 5.6.2** wird ohne schädlichen Code veröffentlicht, und Distributionen liefern Notfall-Updates aus.

Fazit

- **Langfristige Bedrohungen für Open Source**:
  - Angreifer können sich **über Jahre hinweg** in ein Projekt einschleichen und irgendwann schädlichen Code einführen.

- **Sicherheit braucht mehr als Vertrauen**:
  - Open-Source-Projekte dürfen sich **nicht nur auf Maintainer verlassen**,
    sondern müssen Mechanismen wie **automatische Code-Analysen und mehrstufige Reviews** einsetzen.

- **Wichtigkeit von unabhängigen Sicherheitsforschern**:
  - Ohne die Aufmerksamkeit von Entwicklern wie **Andres Freund** hätte diese Backdoor
    möglicherweise Millionen von Systemen kompromittieren können.

- **Abhängigkeiten und SBOMs (Software Bill of Materials) sind kritisch**:
  - Unternehmen und Entwickler müssen ihre Open-Source-Abhängigkeiten **besser überwachen**,
    um sicherzustellen, dass keine kompromittierten Pakete in ihre Software gelangen.

???

Fragen

bechtle

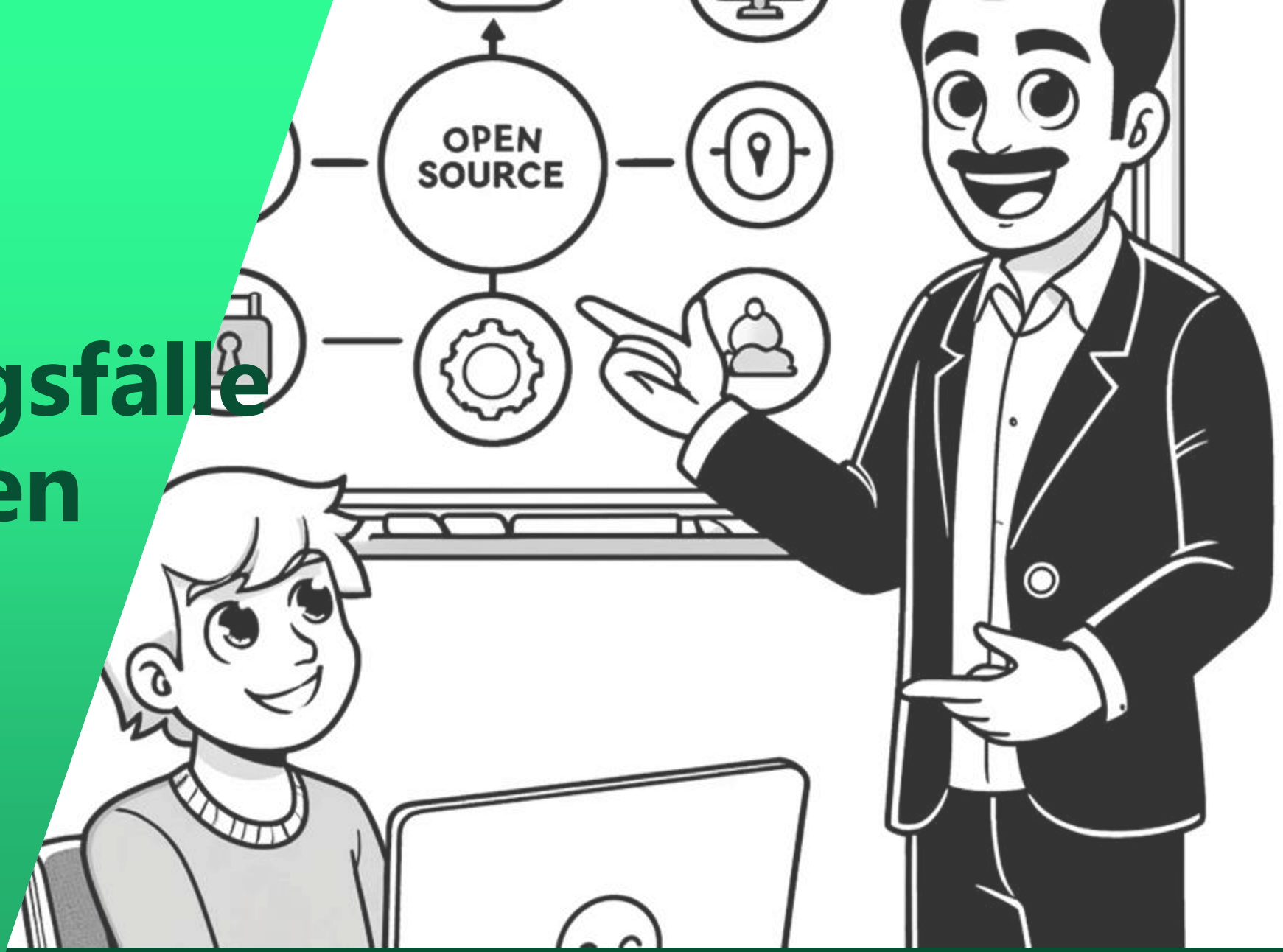
Pause

11.02.2025

bechtle

Was sind Anwendungsfälle beim Kunden

Gibt es den Anwendungsfall ?



11.02.2025

Es gibt nicht den Anwendungsfall, sondern immer „nur“ Alternativen

Sprung zurück zur Definition, hier verstecken sich die Anwendungsfälle

Open Source Software (OSS) ist frei zugängliche Software, deren Quellcode jeder einsehen, verändern und weitergeben darf. Sie fördert Transparenz, Zusammenarbeit und Innovation, bietet Unternehmen Flexibilität und reduziert Abhängigkeiten.

Mögliche Entscheidungspunkte I

1. Wenn Kosten gesenkt werden sollen (Kostenreduktion & Lizenzfreiheit)

✅ **Lizenzkosten sparen** – Keine teuren Software-Abonnements oder Lizenzgebühren

✅ **Bessere Budgetkontrolle** – Keine plötzlichen Preiserhöhungen durch Hersteller

2. Wenn Unabhängigkeit von Herstellern gewünscht ist (Vendor Lock-in vermeiden)

✅ **Keine Abhängigkeit von einzelnen Anbietern** – Software kann angepasst und weiterverwendet werden

✅ **Langfristige Kontrolle über Software** – Unternehmen entscheiden über Updates & Weiterentwicklung

3. Wenn Flexibilität & Anpassbarkeit wichtig sind

✅ **Software kann individuell modifiziert werden** – keine erzwungenen Funktionen oder Einschränkungen

✅ **Schnittstellen & Kompatibilität** – Open Source kann leichter mit bestehenden Systemen verbunden werden

Mögliche Entscheidungspunkte II

4. Wenn Sicherheit & Transparenz entscheidend sind

✅ **Quellcode ist offen einsehbar** – Keine versteckten Hintertüren oder Tracking-Funktionen

✅ **Höchste Sicherheit durch Community-Review** – Sicherheitslücken werden schnell gefunden & behoben

5. Wenn Skalierbarkeit & Zukunftssicherheit gefragt sind

✅ **Kein Lizenzlimit** – Open Source kann beliebig skaliert werden

✅ **Beliebig viele Nutzer, Server & Geräte** – Keine teuren Upgrades oder neue Lizenzen erforderlich

6. Wenn Datenschutz & DSGVO-Konformität wichtig sind

✅ **Datenhoheit** – Unternehmen können Software selbst hosten und Daten in der eigenen Infrastruktur behalten

✅ **Vermeidung von US-Cloud-Diensten** – DSGVO-konforme Alternativen ohne Risiko

Mögliche Entscheidungspunkte III

📌 Nicht geeignet, wenn…

❌ **Es keine internen IT-Ressourcen gibt, um Open Source zu verwalten**

🔻 **Beispiel:**
Ein kleines Unternehmen ohne eigene IT-Abteilung will auf Open Source umsteigen, hat aber niemanden, der sich um Installation, Updates und Sicherheit kümmert.

❌ **Das Unternehmen vollständig auf proprietäre Lösungen angewiesen ist**

🔻 **Beispiel:**
Ein Unternehmen nutzt Microsoft 365 tief integriert mit Teams, SharePoint und Dynamics CRM.

❌ **Eine garantierte Hersteller-Supportstruktur erforderlich ist**

🔻 **Beispiel:**
Ein Krankenhaus nutzt medizinische Software, die gesetzlich zertifiziert sein muss. Es braucht 24/7-Support und garantierte SLA (Service Level Agreements).

Mögliche Entscheidungspunkte

Fazit für Kundenentscheidungen

✅ Open Source ist ideal, wenn **Kosten gesenkt, Flexibilität gewonnen und Unabhängigkeit erhöht werden soll**.

✅ Es eignet sich besonders für **Unternehmen mit IT-Know-how**, die Software selbst verwalten können.

❌ Wenn ein Unternehmen **gar keine internen IT-Ressourcen hat**, sind **kommerzieller Hersteller-Support oder SaaS-Lösungen** oft besser geeignet.

???

Fragen

bechtle

Was sind Anwendungsfälle beim Kunden

Beispiele

11.02.2025

Beispiele I

✓ Kosten senken – Kein Lizenzmodell, keine teuren Hersteller-Abos

✅ **Beispiel:**
Ein mittelständisches Unternehmen mit 500 Mitarbeitern nutzt bisher Microsoft Office mit jährlichen Lizenzkosten von ca. 200 € pro Benutzer.
👉 **Umstieg auf LibreOffice** spart dem Unternehmen **100.000 € jährlich** an Lizenzkosten.

✅ **Beispiel:**
Ein Startup nutzt Google Drive für Dateiablage und zahlt pro Nutzer eine monatliche Gebühr.
👉 **Wechsel zu Nextcloud**: Keine laufenden Lizenzkosten, volle Kontrolle über die Daten.

✓ Herstellerunabhängigkeit – Open Source bleibt langfristig nutzbar

✅ **Beispiel:**
Ein Unternehmen setzt auf eine proprietäre CRM-Software (z. B. Salesforce). Nach 3 Jahren steigen die Lizenzkosten erheblich, und die Vertragsbedingungen ändern sich.
👉 **Wechsel zu Odoo (Open Source-ERP)**: **Volle Kontrolle** über das System, keine Lizenzabhängigkeit.

✅ **Beispiel:**
Ein Unternehmen nutzt VMware für Virtualisierung, doch nach einer Übernahme durch Broadcom steigen die Preise drastisch.
👉 **Umstieg auf XEN oder KVM**: Keine teuren Lizenzen, volle Kontrolle über die Virtualisierung.

Beispiele II

✓ **Zukunftssicherheit & Skalierbarkeit – Beliebig viele Nutzer & Geräte ohne Mehrkosten**

✅ **Beispiel:**
Ein wachsendes Unternehmen nutzt eine proprietäre Firewall, die pro zusätzlichen Standort teuer lizenziert werden muss.
👉 **Wechsel zu pfSense (Open Source-Firewall)**: **Unbegrenzte Nutzung** ohne Mehrkosten pro Standort.

✅ **Beispiel:**
Ein Unternehmen setzt auf Microsoft SQL Server, muss aber für größere Datenmengen **teure Skalierungsgebühren**zahlen.
👉 **Wechsel zu PostgreSQL oder MariaDB**: **Keine Skalierungskosten**, auch bei Millionen von Datensätzen.

✓ **Flexibilität & Anpassbarkeit – Software kann an Unternehmensprozesse angepasst werden**

✅ **Beispiel:**
Ein Online-Shop nutzt eine SaaS-E-Commerce-Plattform wie Shopify, ist aber an deren Zahlungsanbieter gebunden.
👉 **Wechsel zu Open Source-Lösung Magento oder WooCommerce**: **Freiheit bei Zahlungsmethoden & Anpassungen**.

✅ **Beispiel:**
Ein Logistikunternehmen benötigt eine spezielle Funktion in seiner ERP-Software, die der Hersteller nicht anbietet.
👉 **Open Source-ERP (z. B. Odoo, ERPNext) wird angepasst**, um den individuellen Bedarf zu decken.

Ein kleiner Leitfaden !

Den Leitfaden findest du hier : Leitfaden

Eine Liste der Produkte, die Bechtle anbieten kann: Liste

???

Fragen

https://github.com/mstendel/OSS_workshop

bechtle

Ende

Sollte ihr noch Fragen haben:
manfred.stendel@bechtle.com

11.02.2025