Engineer

Linux

Kernel, auf dem die meisten Server laufen

Was es ist

Linux ist ein freier, Unix-ähnlicher Betriebssystemkernel: das Programm, das zuerst startet, die Hardware ansteuert, Prozessorzeit und Speicher unter allen anderen Programmen aufteilt und entscheidet, welches auf welche Datei oder welchen Netzwerkport zugreifen darf. Im Alltag wird der Begriff auf das gesamte um diesen Kernel aufgebaute Betriebssystem ausgedehnt. Manche schreiben GNU/Linux für die Kombination, um die GNU-Werkzeuge zu würdigen, die gewöhnlich dazugehören.

Linus Torvalds begann 1991 mit einem von Grund auf neu geschriebenen Unix-Klon. Ein loses Netz von Entwicklern, die über das Internet zusammenarbeiten, hat ihn seitdem weiterentwickelt. Er richtet sich nach den Standards POSIX und Single UNIX Specification. Für ein Unix-ähnliches System geschriebene Software lässt sich deshalb gewöhnlich mit wenigen Änderungen darauf bauen. Er begann auf dem 32-Bit-Intel-386 und läuft heute auf einer langen Liste von 32-Bit- und 64-Bit-Architekturen. Der Code ist ausschließlich unter Version 2 der GPL lizenziert, mit einem ausdrücklichen Hinweis, dass Programme, die den Kernel lediglich aufrufen, nicht davon erfasst sind. Wegen dieses Hinweises läuft proprietäre Software auf Linux, ohne dass daraus eine Lizenzfrage entsteht.

Am Kernel allein kann man sich nicht anmelden. Ein funktionsfähiges System braucht außerdem eine C-Bibliothek, eine Shell, einen Dienstemanager, eine Paketverwaltung und Tausende Programme. Die Projekte, die das alles zu einem installierbaren Ganzen zusammenstellen, heißen Distributionen. Debian, Ubuntu, Red Hat Enterprise Linux und dessen Nachbauten, SUSE, Fedora und Alpine sind Distributionen. Jede liefert ihren eigenen Kernel-Build, ihre eigenen Voreinstellungen und eine eigene Zusage dazu, wie lange sie Sicherheitsprobleme beheben wird.

Die Alternativen hängen von der Aufgabe ab. Auf Servern wird Linux gegen Windows Server und die BSD-Familie abgewogen: FreeBSD und OpenBSD sind vollständige Unix-ähnliche Systeme, die als Ganzes entwickelt statt aus Einzelteilen zusammengesetzt werden. Auf Laptops trifft es auf Windows und macOS.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Linux?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Server, E-Mail, Konten und Updates werden im Alltag betreut, damit die Dinge, auf die alle angewiesen sind, unauffällig funktionieren.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 10 PostgreSQL-/MS-SQL-Server auf Ubuntu — 10 bereitgestellte und funktionsfähig gehaltene Datenbankserver

10 PostgreSQL-/MS-SQL-Server auf Ubuntu Kontakt aufnehmen →

Warum es wichtig ist

Für ein Unternehmen, das Software auf eigenen Servern oder in einer Cloud betreibt, steht das Betriebssystem meist schon fest, bevor jemand danach fragt. Webserver, Datenbanken, Container-Engines und Monitoring-Werkzeuge werden zuerst für Linux gebaut und dort am gründlichsten getestet. Die verbleibende Entscheidung ist enger umrissen und sollte richtig getroffen werden.

Die Wahl zwischen Linux und Windows Server hängt von der Software ab, nicht vom System. Ein Unternehmen, das auf Active Directory, IIS oder älteren .NET-Anwendungen aufbaut, hat handfeste Gründe, bei Windows zu bleiben. Fast alles andere in einem modernen Webstack, von PostgreSQL und Nginx bis Kubernetes, setzt Linux voraus. Selbst Microsoft SQL Server hat eine Linux-Ausgabe.

Die Kostenfrage ist einfach. Den Kernel und Community-Distributionen wie Debian und Ubuntu auf beliebig vielen Rechnern zu installieren kostet nichts, und es gibt keine zu prüfende Lizenzanzahl. Windows Server wird unter einer kommerziellen Lizenz verkauft. Bei Linux fließt das Geld in Menschen und, falls gewünscht, in ein Supportabonnement bei einem Anbieter wie Canonical, Red Hat oder SUSE.

Linux-Container verstärken diese Bindung. Jedes Image verpackt einen Linux-Benutzerraum in eine Datei. Es auszuführen bedeutet, gewöhnliche Prozesse auf einem Linux-Kernel auszuführen. Auf einem Mac oder Windows-Laptop starten Werkzeuge für diese Container deshalb zuerst eine kleine virtuelle Linux-Maschine. Apples eigenes Werkzeug `container` führt jeden davon in einer eigenen leichtgewichtigen virtuellen Maschine aus. Wer Linux-Container wählt, nimmt Linux in seinen Stack auf, unabhängig vom Betriebssystem auf den Laptops der Entwickler.

Was zwei Linux-Server tatsächlich unterscheidet, ist die Distribution und die Dauer ihrer Pflege. Das Kernelprojekt veröffentlicht alle neun bis zehn Wochen eine neue Mainline-Version und pflegt einige Longterm-Zweige über Jahre. Für 6.12 sind beispielsweise Korrekturen bis Dezember 2028 vorgesehen. Distributionen bauen ihre eigene Pflege darauf auf. Der Kernel einer Distribution wird von der Distribution unterstützt, nicht von den Kernelentwicklern. Wählen Sie die Distribution nach ihrem Supportzeitraum und den Fähigkeiten, die das Team bereits besitzt.

Eine Regel des Kernelprojekts macht Aktualisierungen weniger riskant, als dieses Tempo vermuten lässt: Eine Änderung, die zuvor funktionierende Software beschädigt, gilt als Regression. Regressionen werden behoben, oft durch das Zurücknehmen der Änderung. Deshalb laufen vor langer Zeit gebaute Programme weiterhin auf neuen Kerneln.

Der Preis dieser Auswahl ist, dass „wir betreiben Linux“ sehr wenig aussagt. Zwei Distributionen unterscheiden sich in Paketverwaltung, Firewall-Werkzeug, Netzwerkkonfiguration und Veröffentlichungsrhythmus. Ein Team, das eine davon kennt, muss Teile der anderen neu lernen.

Wie es in der Praxis aussieht

Den Blick aus nächster Nähe bietet ein einzelner Host mit 464 MB, der eine Forge und einen Webserver betreibt. Sein Alarm für knappen Speicher oder Plattenplatz ist ein zeitgesteuerter Totmannschalter. Alle fünfzehn Minuten, mit zufälliger Zeitverschiebung, misst ein Skript verfügbaren Speicher und freien Plattenplatz. Es sendet nur dann ein Lebenszeichen an einen externen Dienst, wenn beide über ihren Grenzwerten liegen. Schweigen ist deshalb der Alarm. Das Skript hatte immer den richtigen Wert gelesen, den verfügbaren Speicher, während sechs Variablennamen darum herum „free“ sagten. Weil der Kernel ungenutzten Speicher als Cache verwendet, liegt freier Speicher auf einem

gesunden Linux-Rechner nahe null. Bei einer Grenze von zehn Prozent hätte ein Leser auf dem Rechner mit 464 MB zwölf Megabyte freien Speicher gesehen. Eine „Korrektur“ um ein Zeichen hätte aus einem funktionierenden Alarm einen gemacht, der dauerhaft anschlägt. Die Namen wurden korrigiert. Ein empfangenes Signal bestätigt, dass das Skript lief und seine Prüfungen der Grenzwerte für verfügbaren Arbeitsspeicher und freien Plattenplatz bestanden wurden. Es bescheinigt weder den allgemeinen einwandfreien Zustand des Hosts noch, dass die Website ausgeliefert wird. Die Überwachung kann Warnzeichen sichtbar machen; wann diese auftreten, hängt jedoch von der Arbeitslast und den beobachteten Bedingungen ab.

In einer früheren Rolle baute die Gründerin für eine Organisation deren erstes Observability-Dashboard. Es sammelte CPU-, RAM-, HDD-, Temperatur- und weitere Messwerte von entfernten Linux-Servern über SSH und überwachte Docker-Container auf dieselbe Weise. Eine zweite Version wechselte zu Grafana und Prometheus. Es lief auf einem großen Fernseher im Büro, mit Alarmen bei Grenzwertüberschreitungen. Die Teams erkannten und behoben Vorfälle 40 % schneller.

Für den gesamten IT-Bestand wurden Zugriffsregeln schriftlich festgehalten, statt sich auf Erinnerungen zu verlassen. Die Linux-Server gehörten zu den Systemen, für die 1.000 RBAC-Regeln galten, neben Entwicklerumgebungen, Anwendungsinstanzen sowie den Datenbanken PostgreSQL und MS SQL. Ansible wandte die Regeln über Umgebungen hinweg konsistent an. Entwickler konnten nur ihre zugewiesenen Server erreichen. Geliefert wurde ein dokumentierter rollenbasierter Regelsatz mit einem wiederholbaren Verfahren, ihn anzuwenden und zu überprüfen.

Wo es schiefgeht

Das Dashboard zeigt fast vollen Speicher auf einem völlig gesunden Server. Linux behandelt ungenutzten Speicher als Verschwendung und füllt ihn mit zwischengespeicherten Dateidaten. Diesen Speicher gibt es sofort frei, wenn ein Programm ihn anfordert. Der Wert „free“ zählt nur Speicher, der gar nichts tut, und nähert sich deshalb auf jedem beschäftigten Rechner null. Die Antwort auf „kann dieser Server mehr Arbeit übernehmen?“ gibt MemAvailable: die eigene Schätzung des Kernels, was neue Programme ohne Auslagern verwenden könnten. Lösen Sie auf Grundlage dieses Werts Alarme aus:

```
grep -E '^(MemFree|MemAvailable):' /proc/meminfo
```

Ein Dienst verschwindet, und die Erklärung steht im Kernelprotokoll statt in seinem eigenen. Standardmäßig lässt Linux Programme mehr Speicher reservieren, als der Rechner hat, in der Erwartung, dass vieles davon nie genutzt wird. Geht diese Rechnung nicht auf und der Speicher tatsächlich aus, wählt der Out-of-Memory-Killer einen Prozess, gewöhnlich einen großen, und beendet ihn. Die Anwendung erhält keine Gelegenheit mehr, etwas zu schreiben. Der Eintrag steht im Kernelprotokoll, das `journalctl -k` anzeigt. Geben Sie jedem Dienst eine eigene Speicherobergrenze, über systemds `MemoryMax=` oder ein Containerlimit. So wird ein ausuferndes Programm gestoppt, bevor der Kernel auswählen muss. Beobachten Sie den verfügbaren Speicher, statt auf das Beenden zu warten.

Ein an die Kernelentwickler gesendeter Fehlerbericht bleibt unbeantwortet. Der Rechner läuft mit dem Kernel seiner Distribution und deren eigenen Patches; die Kernelentwickler unterstützen nur ihre eigenen Veröffentlichungen. `uname -r` zeigt die Version. Alles nach dem Bindestrich, etwa in `6.8.0-45-generic`, kennzeichnet einen Build der Distribution. Melden Sie das Problem zuerst der Distribution oder reproduzieren Sie es auf einem aktuellen Kernel von kernel.org, bevor Sie es an die Kernelentwickler weitergeben.

Sicherheitskorrekturen bleiben aus, während der Server wie bisher weiterläuft. Jede Distributionsversion und jeder Longterm-Kernel hat ein Enddatum, das der Rechner selbst nicht ankündigt. Für die Longterm-Zweige 5.10 und 5.15 ist dieses beispielsweise für Dezember 2026 vorgesehen. Halten Sie das Ende des Supports im selben Inventar wie den Servernamen fest. Planen Sie das Upgrade mit einer Version Vorlauf statt erst zum Stichtag.

Weiterführende Lektüre

- Über den Linux-Kernel — die kurze Einführung des Kernelprojekts, was Linux ist und warum Einsteiger eine Distribution brauchen.
- Veröffentlichungen — Mainline-, Stable- und Longterm-Kernel mit dem vorgesehenen Ende jedes Longterm-Zweigs.
- Das /proc-Dateisystem — wo MemFree, MemAvailable und die anderen Speicherzähler definiert sind.
- Speicherbilanzierung bei Overcommit — die drei Overcommit-Modi und was jeder einem Programm zusichert.

- Regressionen melden — die Regel „keine Regressionen“ und wie ein Defekt gemeldet wird, damit er als solcher zählt.
- Lizenzregeln des Linux-Kernels — ausschließlich GPL-2.0 und der Systemaufrufhinweis, der darüber laufende Programme frei lässt.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Linux?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/linux/