Einführung in den Nix-Paketmanager

Der Paketmanager

Ein Paketmanager behandelt Pakete, Softwareverteilungen und Daten in Archivdateien. Pakete enthalten Metadaten wie den Namen der Software, die Beschreibung ihres Zwecks, die Versionsnummer, den Hersteller, die Prüfsumme (vorzugsweise eine kryptografische Hash-Funktion) und eine Liste der Abhängigkeiten, die für die ordnungsgemäße Ausführung der Software erforderlich sind. Bei der Installation werden Metadaten in einer lokalen Paketdatenbank gespeichert. Paketmanager pflegen in der Regel eine Datenbank mit Softwareabhängigkeiten und Versionsinformationen, um Softwarekonflikte und fehlende Voraussetzungen zu vermeiden. Sie arbeiten eng mit Software-Repositorys, Binär-Repository-Managern und App-Stores zusammen.

Allerdings haben traditionelle Paketmanager wie apt und pacman Schwachstellen, die möglicherweise unzuverlässig für den Einsatz in Umgebungen sind, die sehr stabile Systeme benötigen - wie Server.

Nix wurde als zuverlässiges und reproduzierbares Paketverwaltungssystem konzipiert, das die üblichen Probleme der traditionellen Paketmanagern vermeidet. Aber um die Lösung zu verstehen, müssen wir zuerst das Problem verstehen.

Nachteile der traditionellen Paketmanager

Traditionelle Paketmanager wie apt oder pacman installieren alle Pakete global. Wenn du ein Paket aktualisierst oder neu konfigurierst, wird einfach die entsprechende Datei eingetragen und ändert damit den Zustand des Paketes.

Was macht NIX?

Aktualisierungen sind nicht sofort. Bei jeder Aktualisierung werden die bestehenden Pakete während der gesamten Dauer der Aktualisierung angewendet. Das bedeutet, dass bei einer Unterbrechung der Aktualisierung auf eine neue Konfiguration - z.B. Stromausfall auf halber Strecke - das System sich immer noch in einem stabilen Zustand befindet.

Die Hölle der Abhängigkeiten.

Diesem Problem ist sogar eine ganze Wikipedia-Seite gewidmet. Angenommen, du müssen ein Paket A installieren, das von Paket B und Paket C abhängig ist. B und C sind ihrerseits von zwei verschiedenen Versionen von D abhängig. Traditionelle Paketmanager können nicht zwei Versionen desselben Pakets zur gleichen Zeit verwenden. In diesem Fall kannst wir also entweder B oder D installieren, aber nicht A nicht installieren. Diese besondere Form des Problems wird als Diamant-Abhängigkeitskonflikt genannt.

Diamant-Abhängigkeitskonflikt

Was meint das???

Es gibt viele andere Abhängigkeitsprobleme wie zirkuläre Abhängigkeit und lineare Abhängigkeits-Konflikte, die alle aus der Tatsache resultieren, dass traditionelle Paketmanager nicht zwei Versionen desselben Pakets

nebeneinander installieren können.

Wie Nix diese Probleme löst

Stelle dir Nix als eine Maschine vor, die alle Eingaben in Bezug auf ein Paket entgegennimmt d.h. seine Abhängigkeiten, die Versionen aller Abhängigkeiten, C/CXX-Flags, Quellen, Konfigurationsdateien, Umgebungsvariablen, externe Patches, usw. und gibt dann einen Ausdruck (Nix Expression) aus, der dann ein Paket mit einem eindeutigen Hash erzeugt, der nicht geändert werden kann, und es unter /nix/store/some-unique-package-name. Eine Neukompilierung des Pakets durch Änderung der Abhängigkeitsversion oder durch einer anderen Abhängigkeit führt zu einem völlig anderen Paket mit einem anderen Hash.

Mehrere Versionen

Aufgrund des Hashing-Schemas landen verschiedene Versionen eines Pakets in verschiedenen Pfaden im Nix-Speicher, damit sie sich nicht gegenseitig stören. Noch einmal zum Problem der Abhängigkeitshölle, da die beiden Versionen von Paket D nun völlig unterschiedliche Pakete sind, kannst du Paket B mit einer Version und Paket C mit einer anderen Version bauen ohne jeglichen Konflikt.

Atomare Änderungen und Rollbacks

Da sich die aktualisierten Paketen völlig von den bestehenden unterscheiden, werden die alten Pakete nicht gelöscht, sondern die neuen Pakete werden einfach über die älteren Pakete installiert, d.h. das aktualisierte Paket steht im Zugriff vor der älteren Version. Da keine Änderung an den derzeit verwendeten Paketen vorgenommen wird, wird das System nicht beschädigt, wenn der Aktualisierungsprozess zwischendurch unterbrochen wird. Die eigentliche Änderung, d.h. die Verknüpfung neuer Versionen von Paketen über die älteren erfolgt in einem Augenblick der vollständigen Installation.

Da die alten Versionen der Pakete noch vorhanden sind, bedeutet dies, du kannst den Rechner jederzeit auf die alten Pakete zurücksetzen. Wenn du der Meinung bist, dass diese ganze Mehrfachversions-Kram zu viel Platz verbraucht, kannst du jederzeit einen Garbage-Collection-Befehl ausführen, der deine unbenutzten Pakete löscht.

Wie man Nix installiert

Wenn du Nix ausprobieren willst, aber nicht deine komfortable Distribution verlassen willst, die du aufgebaut hast, kannst du den Nix-Paketmanager auf jeder Linux Distribution mit nur einem einzigen Befehl installieren.

> **STOP!!!**
>
> **Warnung:** Das Ausführen von curl some-url | sh, wie in der Nix-Dokumentation vorgeschlagen, wird als Sicherheitsrisiko betrachtet, da es unbekannten Code ausführt. Es wird daher empfohlen, das Skript manuell herunterzuladen und zu überprüfen, bevor es ausgeführt wird.

Wenn du dir sicher bist, führe einfach den folgenden Befehl in Ihrem Terminal aus:

```
curl -L https://nixos.org/nix/install | sh
```

Die Installation kann auch über die jeweilige Anwendungsverwaltung der Distribution erfolgen.

Eine Einzel- oder eine Mehrbenutzer-Installation ist möglich.

Siehe: https://nixos.org/manual/nix/stable/installation/installing-binary

Durchführung einer Einzelplatz-Installation von Nix.../tmp/nix-binary-tarball-unpack.A3weQFWCUQ/unpack/nix-2.13.3-x86_64-1inux/install

Das Verzeichnis /nix existiert, ist aber für Sie nicht beschreibbar. Dies könnte darauf hinweisen, dass ein anderer Benutzer bereits eine Einzelplatzinstallation von Nix auf diesem System durchgeführt hat.

Wenn Sie die Mehrbenutzerunterstützung aktivieren möchten, lesen Sie: 

URL??? Wenn du mit einer Einzelbenutzerinstallation fortfahren willst:

```
chown -R <user> /nix  \\ >>als root ausführen
```

Sicherheit

Nix hat zwei grundlegende Sicherheitsmodelle. Erstens kann es im „Einzelbenutzermodus“ verwendet werden, was dem entspricht, was die meisten anderen Paketverwaltungsprogramme tun: Es gibt einen einzigen Benutzer (normalerweise root), der alle Paketverwaltungsoperationen durchführt. Alle anderen Benutzer können dann die installierten Pakete verwenden, aber sie können selbst keine Paketverwaltungsoperationen durchführen.
Nix wurde im „Mehrbenutzermodus“ konfiguriert. In diesem Modell können alle Benutzer Operationen zur Paketverwaltung durchführen - zum Beispiel kann jeder Benutzer Software installieren, ohne Root-Rechte zu benötigen. Nix sorgt dafür, dass dies sicher ist. So ist es beispielsweise nicht möglich, dass ein Benutzer ein Paket, das von einem anderen Benutzer verwendet wird, mit einem Trojaner überschreibt.

Konfiguration

Um den Nix-Daemon beim Booten zu starten, aktiviere den nix-daemon.service. (Daemon - einen Prozess, der im Hintergrund abläuft und Dienste zur Verfügung stellt)

Füge die erforderlichen Benutzer zur Gruppe nix-users hinzu, um auf den Socket des Daemons zugreifen zu können.

Füge einen Kanal hinzu und aktualisiere ihn.

Die Liste der offiziellen NixOS-Kanäle findest du unter https://nixos.org/channels. Z.B.:

```
nix-channel --add https://nixos.org/channels/nixpkgs-unstable
nix-channel --update
```

Wie benutzt man den Nix-Paketmanager?

Profile

Im Gegensatz zu herkömmlichen Paketmanagern unterstützt Nix Profile. Ein Profil ist eine Umgebung in der bestimmte Pakete installiert sind. Profile ermöglichen es schnell zwischen unterschiedlichen Versionen installierter Software zu wechseln. Jeder Benutzer (auch Nicht-Administratoren) kann Profile anlegen und in diesen Software installieren.

Channels - Kanäle

Ein Kanal ist eine Git-Repository-Liste (verwaltetes Verzeichnis) von Paketen und deren Definitionen, die offiziell von Nix überprüft wurden. Dies ist vergleichbar mit dem Konzept der Repositories in Ubuntu/Debian/Arch. Es gibt mehrere Kanäle auf verschiedenen Ebenen der Verifizierung.

Typische Befehle:

Aktuelle Kanäle auflisten:

```
nix-channel --list
```

Neue Kanäle abonnieren:z.B.:

```
nix-channel --add https://nixos.org/channels/nixpkgs-unstable
```

Kanäle entfernen:

```
nix-channel --remove channel-alias
```

Kanäle aktualisieren:

```
nix-channel --update channel-alias
```

Alle Kanäle aktualisieren:

```
nix-channel --update
```

Suche nach Paketen

So suchst du nach einem Paket in der aktuellen Kanäle:

```
nix search <pkg-name>
```

Es wird eine Reihe von Anwendungen aufgelistet, die das Schlüsselwort im Format enthalten:

```
<nixpkgs.pkg-name>
```

Du kannst aber auch den Online-Nix-Store besuchen und nach dem Paket mit Anweisungen zu dessen Installation suchen: https://search.nixos.org/packages

Installieren von Paketen

Um ein beliebiges Paket aus dem aktuellen Profil zu installieren, verwende:

```
nix-env -iA <nixpkgs.pkg-name>
```

Das „-A“-Flag ermöglicht, einen bestimmten Kanal für den Download des Paketes herunterzuladen.

Wenn du vom Standardkanal aus installieren möchtest, verwende:

```
nix-env -i <pkg-name>
```

Pakete deinstallieren

Um ein beliebiges Paket im aktuellen Profil zu installieren, verwende:

```
nix-env -e <pkg-name>
```

Upgrade von Paketen

Um ein bestimmtes Paket zu aktualiduren, verwende:

```
nix-env -u subversion
```

Um alle Pakete des Profils zu aktualisieren, verwende:

```
nix-env -u
```

Rollbacks

Du kannst deinen letzten nix-env-Befehl zurücksetzen:

```
nix-env --rollback
```

Jedes Mal, wenn du eine nix-env-Operation durchführst, wird eine neue Benutzergeneration von der Benutzer-Umgebung erstellt. Wenn du zum Beispiel gimp installierst, wird eine neue Generation erstellt. Du kannst alle Generationen auflisten mit:

```
nix-env --list-generations
```

und wechsele zu einer bestimmten Generation mit:

```
nix-env --switch-generation <generation-number>
```

Entfernung von Müll

Du kannst ältere Generationen löschen:

```
nix-env --delete-generations old
```

Um bestimmte Generationen zu löschen, verwende:

```
nix-env --delete-generations 1 2
```

Hashes in Nix

Kryptographische Hashes spielen an vielen Stellen im Nix-Ökosystem eine wichtige Rolle. Wenn ein Hash irgendwo verwendet wird, sind zwei Kriterien entscheidend: der verwendete Algorithmus und die Kodierung (und in gewissem Maße auch das, was gehasht wird).

Unterstützte Algorithmen sind md5, sha1, sha256, sha512. Die ersten beiden sind veraltet und sollten nicht mehr verwendet werden, aber es kann sein, dass du in vorhandenem Code noch über sie stolpern.

Ein Hash - der einfach eine Folge von Bytes ist - wird normalerweise kodiert, damit er als String dargestellt werden kann. Gängige Kodierungen sind base16 (gemeinhin „hex“ genannt), base32 und base64. Beachte, dass base32 eine benutzerdefinierte Kodierung ist, die weder dokumentiert noch in irgendeiner Weise standardisiert ist! Wenn möglich, verwende die mitgelieferten Hash-Werkzeuge, um Hashes in diese Kodierung umzuwandeln (siehe unten). base32 wird von Nix an vielen Stellen verwendet, da es kürzer als hex ist, aber trotzdem sicher Teil eines Dateipfades sein kann (da es keine Schrägstriche enthält).

Quellen:

By Shobhit Singh / June 10, 2021 : https://www.linuxfordevices.com/tutorials/linux/nix-package-manager
Nixos : https://nixos.org/manual/nix/stable/introduction.html

Ergänzende zum weiteren Studium:

- https://de.wikibrief.org/wiki/Package_manager
- https://nixos.wiki/wiki/Nix_Package_Manager
- https://de.wikipedia.org/wiki/Nix_(Paketmanager)
- https://ariya.io/2020/05/nix-package-manager-on-ubuntu-or-debian
- https://christitus.com/nix-package-manager/
- https://julianhofer.eu/blog/01-silverblue-nix/

Wer Youtube braucht, diese fand ich OK:

- https://www.youtube.com/watch?v=CQMU3kSRhLU - So'n Typ im Internet - DE
- https://www.youtube.com/watch?v=Zxx9bnE7K7E - Pinguin TV - DE