Was sind Linux-Namespaces und wofür werden sie verwendet?

Linux-Namespaces sind die zugrunde liegende Technologie hinter Containertechnologien wie Docker. Sie sind eine Funktion des Linux-Kernels, die es dem System ermöglicht, die Ressourcen einzuschränken, die containerisierte Prozesse sehen, und die sicherstellt, dass keine von ihnen andere stören kann.

Was sind Namensräume?

Wenn Sie viele verschiedene Prozesse und Anwendungen auf einem einzigen Server ausführen, wie dies bei Bereitstellungstools wie Kubernetes der Fall ist, ist es vor allem aus Sicherheitsgründen wichtig, jeden Prozess zu isolieren.

Ein Container sollte nicht in der Lage sein, die Kontrolle über die Ressourcen eines anderen zu erlangen, denn wenn dieser Container dann kompromittiert wird, könnte dies das gesamte System kompromittieren. Diese Angriffsmethode ähnelt der Funktionsweise des CPU-Bugs Meltdown; verschiedene Threads eines Prozessors sollten voneinander isoliert werden. Ebenso sollten Prozesse, die auf unterschiedlichen virtuellen Systemen (Containern) laufen, von anderen Containern isoliert werden.

Namespaces erreichen diese Isolation auf Kernel-Ebene. Ähnlich wie die Anwendung chroot funktioniert, die einen Prozess in einem anderen Root-Verzeichnis einsperrt, trennen Namespaces andere Aspekte des Systems. Es stehen sieben Namespaces zur Verfügung:

- **Mount**, oder mnt. Ähnlich wie chroot partitioniert der Mount-Namespace das Dateisystem virtuell. Prozesse, die in separaten Mount-Namespaces ausgeführt werden, können nicht auf Dateien außerhalb ihres Mount-Punkts zugreifen. Da dies auf Kernel-Ebene geschieht, ist es viel sicherer, als das Root-Verzeichnis mit chroot zu ändern.
- **Prozess**, oder pid. Unter Linux spawnen die ersten Prozesse als Kinder von PID 1, die die Wurzel des Prozessbaums bildet. Der Prozessnamensraum schneidet einen Zweig des PID-Baums ab und erlaubt keinen Zugriff weiter oben im Zweig. Prozesse in untergeordneten Namespaces haben tatsächlich mehrere PIDs— die erste repräsentiert die globale PID, die vom Hauptsystem verwendet wird, und die zweite PID repräsentiert die PID innerhalb des untergeordneten Prozessbaums, der bei 1 neu gestartet wird.
- **Interprozesskommunikation** oder ipc. Dieser Namespace steuert, ob Prozesse direkt miteinander kommunizieren können.
- **Netzwerk**, oder netto. Dieser Namespace verwaltet, welche Netzwerkgeräte ein Prozess sehen kann. Dadurch wird jedoch nicht automatisch etwas für Sie eingerichtet. Sie müssen weiterhin virtuelle Netzwerkgeräte erstellen und die Verbindung zwischen globalen Netzwerkschnittstellen und untergeordneten Netzwerkschnittstellen verwalten. Containerisierungssoftware wie Docker hat dies bereits erkannt und kann das Netzwerk für Sie verwalten.
- **Benutzer**. Dieser Namespace ermöglicht dem Prozess “virtuelles Root” innerhalb ihres eigenen Namensraums, ohne tatsächlichen Root-Zugriff auf das übergeordnete System zu haben. Es partitioniert auch UID- und GID-Informationen, sodass untergeordnete Namespaces ihre eigenen Benutzerkonfigurationen haben können.
- **UTS**. Dieser Namespace kontrolliert Hostnamen- und Domaininformationen und lässt Prozesse glauben, dass sie auf Servern mit unterschiedlichen Namen ausgeführt werden.

- **Cgroup** ist ein weiteres Kernel-Feature, das Namespaces sehr ähnlich ist. Cgroups ermöglichen es dem System, Ressourcengrenzen (CPU, Arbeitsspeicher, Speicherplatz, Netzwerkverkehr usw.) für eine Gruppe von Prozessen zu definieren. Dies ist eine nützliche Funktion für containerisierte Apps, führt jedoch keine Art von “Informationsisolierung” wie Namensräume. Der Cgroup-Namespace ist eine separate Sache und steuert nur, welche Cgroups ein Prozess sehen kann, und weist ihn keiner bestimmten Cgroup zu.

Standardmäßig verwendet jeder von Ihnen ausgeführte Prozess die globalen Namespaces , und die meisten Prozesse auf Ihrem System tun dies auch, sofern nicht anders angegeben.

Arbeiten mit Namespaces

Sie können den lsns-Befehl (ls-namespaces) verwenden, um die aktuellen Namespaces anzuzeigen, die in Ihrem System aktiv sind. Dieser Befehl muss als Root ausgeführt werden, sonst ist die Liste möglicherweise unvollständig. die lsns-Ausgabe einer neuen Ubuntu-Installation. Jeder Namespace wird neben der Prozess-ID, dem Benutzer und dem Befehl aufgeführt, der ihn erstellt hat. Die sieben Namespaces, die aus /sbin/init mit PID 1 hervorgegangen sind, sind die sieben globalen Namespaces. Die einzigen anderen Namespaces sind mnt-Namespaces für System-Daemons sowie der Livepatch-Dienst von Canonical.

Wenn Sie mit Containern arbeiten, wäre diese Liste viel länger. Sie können diese Liste im JSON-Format mit dem Flag -J ausgeben, das Sie viel einfacher mit einer Skriptsprache verwenden können.

Sie können Ihren aktuellen Namespace mit dem Dienstprogramm nsenter ändern. Mit diesem Befehl können Sie die “Eingabe” der Namespace eines anderen Prozesses, normalerweise für Debugging-Zwecke. Es kann tatsächlich jeden Befehl in diesem Namespace ausführen, aber standardmäßig versucht es nur, eine Shell zu laden (normalerweise /bin/bash).

Sie geben eine Prozess-ID an und dann jeden Namensraum, den Sie eingeben möchten:

```
sudo nsenter -t PID –mount –net –pid
```

etc.

Wenn Sie beispielsweise versuchen, den Mount-Namespace für kdevtmpfs einzugeben, werden Sie in diesen Namespace geladen, schlagen jedoch anschließend fehl, weil /bin/bash nicht gefunden werden kann, was tatsächlich bedeutet, dass es funktioniert hat, weil das scheinbare Stammverzeichnis geändert wurde .

Wenn Ihr untergeordneter mnt-Namespace /bin/bash enthält, können Sie ihn eingeben und eine Shell laden. Dies kann manuell erfolgen, sollte jedoch über Bind-Mounts erfolgen, die den Verzeichnisbaum manipulieren und Dateien über mnt-Namespaces verknüpfen können. Dies kann zu einigen interessanten Anwendungsfällen führen, z. B. wenn zwei Prozesse unterschiedliche Inhalte aus derselben Datei lesen.

Um neue Namespaces zu erstellen, müssen Sie von einem bestehenden (normalerweise global) abzweigen und angeben, welche Namespaces Sie ändern möchten. Dies geschieht mit dem unshare-Befehl, der einen Befehl mit einem neuen Namespace “unshared” vom Master.

Um die Freigabe des Hostnamen-Namespace aufzuheben, verwenden Sie:

```
sudo unshare -u Befehl
```

Wenn der Befehl leer gelassen wird, führt “Unshare” standardmäßig die Bash aus. Dadurch wird ein neuer Namespace erstellt, der in der Ausgabe von lsns angezeigt wird:

Der Terminal-Multiplexer-Bildschirm wird hier verwendet, um die Bash im Hintergrund laufen zu lassen. Andernfalls würde der Namespace verschwinden, wenn der Prozess geschlossen wird.

Es sei denn, Sie führen eine sehr niedrige Programmierung durch. Sie müssen wahrscheinlich nicht selbst Namensräume berühren. Containerisierungsprogramme wie Docker verwalten die Details für Sie, und in den meisten Fällen, in denen Sie eine Prozessisolierung benötigen, sollten Sie einfach ein vorhandenes Tool verwenden. Es ist jedoch wichtig zu verstehen, wie Namespaces im Kontext der Containerisierung funktionieren, insbesondere wenn Sie eine Low-Level-Konfiguration Ihrer Docker-Container durchführen oder manuelles Debugging durchführen müssen.