Einfach Verzeichnisse synchronisieren | iX

Originalartikel

Backup

<html> Daten auf mehreren Rechnern synchron halten, Konfigurationsdateien über mehrere Server abgleichen, VM- oder Container-Images aktualisieren – nur eine kurze Reihe von Einsatzszenarien für Dateisynchronisierung. Geht es um schnelle, latenzfreie Verfügbarkeit der Daten auf einem Rechner, kann ein verteiltes Dateisystem diese Aufgabe erfüllen – das lässt sich jedoch nicht ohne Weiteres aufsetzen und gerade für kleine Umgebungen ist der Aufwand dafür eventuell nicht zu rechtfertigen. Alternativ bieten sich zentrale Dienste wie NFS an, die jedoch in der Regel nur lokal gut genug funktionieren. Modern wäre der Einsatz eines Cloud-Dienstes, da stellen sich allerdings Fragen bezüglich Zuverlässigkeit, Verfügbarkeit und Datenschutz beziehungsweise -sicherheit. Einfache Linux-Kommandozeilentools sind schnell eingerichtet und somit eine Alternative zu diesen Schwergewichten. Sie besitzen allerdings alle einen individuellen Schwerpunkt. Für mehr oder weniger intelligentes Kopieren – insbesondere übers Netzwerk – ist rsync bekannt. <pre id=„p_3“> rsync -avzhu –progress /source /destination </pre> aktualisiert /destination mit dem Inhalt aus /source, behält Nutzerinformation bei, komprimiert den Datenstrom und gibt den Kopierfortschritt an. Die Option –delete würde Dateien in /destination ohne Entsprechung im Quellpfad löschen. Die Rollen von source und destination sind allerdings fixiert und diese Hierarchie schränkt rsync in seiner Funktion als Synchronisationswerkzeug ein. Eine Erweiterung stellt unison dar, das ein Kommandozeilen- oder GTK+-UI zu rsync bietet und dessen Fähigkeiten um bidirektionale Synchronisation erweitert. Nutzer müssen bei Konflikten entscheiden, welcher Version der Vorzug zu geben ist. Ansonsten bringt das Programm beide Pfade zuerst auf den Maximalstand und prüft später auf Veränderungen. Es verfügt zudem über einen Batchmodus, in dem es Konflikte nach Änderungsdatum löst, sodass es im Hintergrund Pfade synchron halten kann. Um mehrere Rechner auf dem gleichen Stand zu halten, empfiehlt sich eine Sternkonfiguration mit einem dedizierten zentralen Knoten, von dem alle anderen Teilnehmer ihre Daten beziehen. <h3 class=„z“ id=„p_6“>Von der Kommandozeile in die Cloud</h3> In der Cloud kommt auch Technik zum Einsatz, die sich auf der Kommandozeile nutzen lässt. Nextcloud und ownCloud haben in ihren Clients ein solches Tool verbaut: csync. Trotz Versionsnummer 0.5 ist es stabil und verrichtet seine Aufgabe, bidirektionales Synchronisieren zweier Verzeichnisse, zuverlässig. Beim ersten Aufruf csync /path/1 /path/2 bringt csync beide Verzeichnisse auf den gleichen Stand und legt unter beiden Pfaden SQLite-Datenbanken an, in denen Informationen über die enthaltenen Dateien liegen. Bei Konflikten wählt csync stets die Datei jüngeren Datums. Unidirektional, aber automatisch funktioniert lsyncd. Es läuft als Hintergrundprozess und wartet auf Aktivierung per inotify oder fsevent. Alle paar Sekunden reagiert lsyncd, indem es standardmäßig die veränderten Dateien an einen zweiten Ort spiegelt, den der Anwender beim Starten angegeben hat. Üblicherweise verwendet das Tool rsync zum Übertragen der Daten. Alternativ lässt sich jedoch die Konfiguration verfeinern, sodass lsyncd nicht immer einfach Dateien repliziert. Es lassen sich im Prinzip beliebige Befehle mit den Dateisystemereignissen kombinieren. Denkbar sind somit etwa Skripte zur automatisierten Erzeugung von Thumbnail-Galerien für Bilderordner. Mit Augenmerk auf den Abgleich von Betriebssystemabbildern verrichtet casync seinen Dienst. Der Entwickler

Lennart Poettering sieht es als Symbiose aus rsync und git – damit fällt es auch in die Klasse der unidirektionalen Werkzeuge. Durch ein geschicktes Chunking minimiert es allerdings den Traffic effizienter als vergleichbare Werkzeuge. casync untersucht den Quellpfad und teilt ihn auf in sogenannte Chunks variabler, aber ähnlicher Größe. Die Aufteilung orientiert sich grob am Dateibaum, kann aber kleinere Dateien in einer Einheit zusammenfassen und große Dateien aufteilen. In einem zentralen Index liegen die Hashes der Blöcke, die bei der Synchronisation mit den Prüfsummen im Zielmedium verglichen werden. Auf Ebene der Chunks zu arbeiten ist auch CDN-freundlich, denn Content Delivery Networks funktionieren weder mit vielen kleinen noch mit einzelnen großen Dateien besonders effizient. Beinhaltet der Quellpfad zum Beispiel ein Dateisystemabbild, in dem ein Entwickler nur eine Bibliothek ausgetauscht hat, synchronisiert casync lediglich den relevanten Teil, während Alternativen wie OSTree oder verteilte Dateisysteme entweder auf der Ebene einzelner Dateien oder dem Image im Ganzen arbeiten. Zusätzlich lässt sich auf der Kommandozeile ein seed-Pfad angeben, dessen Inhalt casync als lokale Quelle für den Pfad verwendet. Möchte man etwa ein Linux-Image beziehen, kann man eine lokale Partition oder ein vorhandenes Image wählen. Damit sollte der Download des Basissystems überflüssig werden und lediglich applikationsspezifische Dateien werden heruntergeladen. (jab@ix.de) </html>