Engineer

Bash

die GNU-Shell, für die die meisten Skripte geschrieben sind

Was es ist

Bash, die Bourne-Again SHell, wird als GNU Bash verteilt und mit `bash` gestartet. Das Programm liest eingegebene oder in Dateien gespeicherte Befehle und führt sie aus. Es startet andere Programme, verbindet Ein- und Ausgabe und ergänzt Variablen, Schleifen, Tests und Funktionen, damit eine Befehlsfolge Entscheidungen treffen kann. Der Name spielt auf Stephen Bourne an, dessen `sh` mit der siebten Ausgabe von Research Unix bei Bell Labs erschien. Das Handbuch beschreibt Bash als weitgehend kompatibel damit, ergänzt um Merkmale der Korn- und C-Shell und mit dem Ziel, dem Shellteil von POSIX zu entsprechen. Die aktuelle Version ist 5.3, das Handbuch zu 5.3 stammt vom Mai 2025.

Eine Shell wird leicht mit ihrem Fenster verwechselt. Das Terminal zeichnet Zeichen und reicht Tastendrücke weiter. Die Shell ist das interpretierende Programm dahinter. Bash läuft interaktiv mit Eingabeaufforderung, Verlauf und Tabvervollständigung oder nicht interaktiv: Ein Skript wird unbeaufsichtigt von oben nach unten gelesen. Dort liegen die meisten Fallen.

Mehrere Programme können diese Rolle übernehmen. POSIX definiert die portable Sprache `sh`; das als `/bin/sh` installierte Programm hängt vom System ab. Unter Debian und Ubuntu ist es dash, eine kleine, von der Almquist-Shell abstammende POSIX-Shell. Debian Policy nennt bash, dash und ksh als die häufigsten Shells. Seit macOS 10.15 ist Zsh die voreingestellte Login-Shell neuer Konten, obwohl `/bin/bash` bleibt. Die erste Skriptzeile, der Shebang, wählt den Interpreter: `#!/bin/sh` verlangt die jeweilige POSIX-Shell, `#!/bin/bash` oder `#!/usr/bin/env bash` fordert Bash ausdrücklich an.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Bash?“ 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

Bash verbindet Programme auf Unix-Servern. Es ist auf fast jedem Linux-Rechner vorhanden, und die am Prompt getippten Befehle sind bereits Bash. Einige davon als Datei ergeben ein Skript für die vorhandenen Werkzeuge. Zum Verketten eines Datenbankdumps, seiner Kompression und einer externen Kopie sind wenige Sprachen ähnlich knapp, weil Pipes, Umleitungen und Exitstatus zu ihrem Grundwortschatz gehören.

Mit wachsendem Skript werden dieselben Eigenschaften teuer. Werte sind Zeichenketten, gerechnet wird nur mit ganzen Zahlen. Strukturiertes wie JSON, Datumsangaben und Dezimalbeträge muss an andere Werkzeuge gehen. Nicht durch Anführungszeichen geschützte Expansionen werden in Wörter zerlegt und anschließend mit Dateinamen abgeglichen. Leerzeichen oder Sternchen verändern so die Bedeutung des Befehls. Fehlerbehandlung muss eingeschaltet werden: Standardmäßig läuft ein Skript nach einem Fehler weiter. Auch `set -e` und `pipefail` haben Ausnahmen, die im Handbuch mehrere Absätze benötigen. Eine brauchbare Grenze lautet: Braucht ein Skript Datenstrukturen, Parser oder eine Testsuite, ist es zum Programm geworden. Eine Sprache wie Python ist dann günstiger zu pflegen.

Portabilität ist ebenfalls eine Entscheidung. Debian Policy beschränkt `/bin/sh`-Skripte auf POSIX und wenige Erweiterungen. Wer mehr braucht, muss die Shell in Zeile eins nennen, etwa `#!/bin/bash`. Zum Finden von Verstößen werden `checkbashisms` oder eine strengere Shell wie posh vorgeschlagen, im Zweifel `/bin/bash`. Der Standard bewegt sich langsam: Bash bietet `pipefail` seit 3.0, POSIX erst seit seiner Ausgabe von 2024.

Eine Shell verwandelt Text in Befehle und bildet überall dort eine Sicherheitsgrenze, wo fremder Text sie erreicht. Shellshock, im September 2014 als CVE-2014-6271 erfasst, zeigte deren Breite. Bis Version 4.3 führte Bash Befehle hinter einer Funktionsdefinition in einer Umgebungsvariable aus. Alles, was Angreifertext vor dem Bash-Start dorthin kopierte, wurde zum Codeausführungspfad: unter anderem Apache-CGI, OpenSSH ForceCommand und manche DHCP-Clients. Die erste Korrektur war unvollständig; der Rest erhielt CVE-2014-7169. Die alltägliche Lehre ist kleiner: Fremder Text erreicht einen Befehl als Argument, nie als Teil einer an `eval` oder `sh -c` übergebenen Zeichenkette.

Wie es in der Praxis aussieht

In der Arbeit des Unternehmens behält Bash die auf der Technikseite beschriebene Aufgabe: Python für Datenarbeit, Automatisierung und Dienste, Bash als lesbar geschriebene Verbindung dazwischen.

In den sechs ETL- und ELT-Pipelines, die Google BigQuery aus einem Warehouse hinter einem IPsec-VPN versorgten, enthielten die Downloads Parquet-, CSV- und .bak-Dateien. Bash und Python entpackten sie und ordneten sie nach Typ und Spaltenstruktur. bcp, psql und SSIS luden sie in Stagingschemas. gzip, Google Cloud Storage und bq trugen die Ergebnisse nach BigQuery. Das skriptgesteuerte Entpacken und Sortieren verkürzte einen manuellen Lauf von mindestens drei Stunden auf unter 20 Minuten. Daten konnten nun täglich statt wöchentlich synchronisiert werden.

Die hundert automatisierten kritischen Datensicherungen zeigen dieselbe Arbeitsteilung. Barman betreute PostgreSQL, Google Cloud die externen Kopien. Bash und Python steuerten und validierten das Ganze. Aufbewahrungsregeln hielten Kopien aktuell, Integritätsprüfungen bestätigten Qualität statt bloßer Existenz. Genau dafür eignet sich Bash: eine Folge bestehender Werkzeuge, bei der der Wert des Skripts darin liegt, die Rückgabe jedes Schritts zu prüfen.

Wo es schiefgeht

Ein Dateiname mit Leerzeichen wird zu zwei Argumenten. Ein nicht in Anführungszeichen gesetztes `$file` wird an Leerraum getrennt und als Glob expandiert. `report 2024.csv` erreicht `rm` als zwei Namen; ein versehentliches

`*` trifft ein ganzes Verzeichnis. Schützen Sie jede Expansion mit Anführungszeichen wie in `"$file"`, reichen Sie Argumentlisten als `"$@"` weiter, halten Sie Namenslisten in Arrays und lassen Sie ShellCheck den Rest finden.

Die Sicherungspipeline meldet Erfolg nach einem fehlgeschlagenen Dump. Der Exitstatus stammt vom letzten Befehl. In `pg_dump app | gzip > app.sql.gz` kann gzip nach einem gescheiterten Dump mit seinen wenigen empfangenen Daten erfolgreich enden. Aktivieren Sie `set -o pipefail`; damit liefert die Pipeline den letzten von null verschiedenen Status. Prüfen Sie zusätzlich mit `gzip -t` oder besser durch Wiederherstellung.

Ein Skript mit `set -e` setzt nach einem Fehler fort. Die Option wird in Tests von `if` und `while`, in jedem außer dem letzten Teil einer `&&`- oder `||`-Liste und in von dort aufgerufenen Funktionen ignoriert. Der folgende Dump kann deshalb scheitern, während das Skript trotzdem Erfolg ausgibt:

```
#!/usr/bin/env bash
set -euo pipefail

backup() {
  pg_dump app > /var/backups/app.sql
  gzip -f /var/backups/app.sql
}

if backup; then # set -e is ignored inside backup here
  echo "backup done"
fi
```

Verwandt ist `local out=$(cmd)`: Es liefert den Status von `local` und verbirgt einen Fehler in `cmd`. Fügen Sie innerhalb von Funktionen hinter jedem wichtigen Befehl `|| return 1` ein. Deklarieren Sie Variablen vor ihrer Zuweisung auf einer eigenen Zeile.

Was auf einem Rechner läuft, scheitert auf dem nächsten. Ein Skript mit `#!/bin/sh`, aber `[[`-Tests oder Arrays, trifft auf einem System Bash und auf dem nächsten dash. Dort ist `[[` unbekannt. Ein darauf bauendes `if` meldet einen Fehler und nimmt den else-Zweig; eine Arrayzuweisung beendet das Skript mit Syntaxfehler. Auf dem Mac meldet `/bin/bash` Version 3.2.57, älter als die in 4.0 eingeführten assoziativen Arrays und `mapfile`. `declare -A` scheitert dort als ungültige Option. Nennen Sie die tatsächlich benötigte Shell im Shebang und testen Sie die älteste unterstützte Version.

Eine Aufräumzeile löscht viel zu viel. Bei ungesetztem `build_dir` wird `rm -rf "$build_dir"/*` zu `rm -rf /*`. `set -u` macht eine ungesetzte Variable zum Fehler. `"${build_dir:?}"` stoppt ein nicht interaktives Skript sowohl bei fehlendem Wert als auch bei leerer Zeichenkette.

Weiterführende Lektüre

- GNU Bash Reference Manual — das Handbuch zu 5.3 mit Expansionen, Quoting, `set` und den genauen Ausnahmen von `set -e`.
- Shell Command Language — die portable `sh`-Sprache aus POSIX.1-2024 als Grenze jedes `#!/bin/sh`-Skripts.
- Debian Policy: Skripte — `/bin/sh`-Pflichten, ausdrückliche Bash-Anforderung und Hinweise zu `set -e`.
- Bash NEWS — die Versionsliste neuer Funktionen des Maintainers zur Prüfung von Mindestversionen.
- CVE-2014-6271 — der NVD-Eintrag zu Shellshock mit Angriffswegen und Folgekennung.

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 Bash?“ 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/bash/