trutz.io

SPICKZETTEL · PROVIDER-SOUVERÄNITÄT

Wer liefert deine Provider?

Lock-File lesen, Mirror bauen, offline verifizieren. Begleitmaterial zum Video, damit du dir den Bezugsweg deiner Provider selbst sicherst, statt ihn einer Registry zu überlassen.

DIE EINE REGEL, AUS DER ALLES FOLGT

Der Provider-Code ist oft identisch. Der Unterschied liegt im Vertriebsweg, und den kannst du dir selbst sichern.

SÄULE 1

Tiefes technisches Verständnis

Du liest das Lock-File, baust einen Mirror und verifizierst ihn offline. So bleibst du handlungsfähig, auch wenn eine Registry ihre Bedingungen ändert.

SÄULE 2

Open-Source-Lizenz

OpenTofu steht unter MPL 2.0 in der Linux Foundation. Der Bezugskanal gehört keiner einzelnen Firma, die ihn drosseln oder schließen kann.

01

Video 03:00

Lock-File lesen: der Kanal steht im Namen

- [ ] Öffne `.terraform.lock.hcl`. Der Schlüssel ist nicht `hcloud`, sondern voll qualifiziert. Die Registry steht mit im Namen.
- [ ] Beim Wechsel **ersetzt** OpenTofu den Eintrag, es stellt keinen zweiten daneben. Genau deshalb fällt der Wechsel des Bezugswegs kaum jemandem auf.
- [ ] Willst du beide Fassungen vergleichen, sichere das Lock-File **vor** dem Wechsel, als Kopie oder aus der Git-Historie.

```
# vorher, von Terraform erzeugt
provider "registry.terraform.io/…/hcloud" { … }
# nachher, von OpenTofu ersetzt
provider "registry.opentofu.org/…/hcloud" { … }
```

02

Video 04:30

Der Beweis: dasselbe Binary, zwei Kanäle

- [ ] Bilde die Prüfsumme beider heruntergeladener Binaries. Zwei Registries, dieselbe `sha256`: der Code ist Byte für Byte identisch.
- [ ] OpenTofu schreibt viele `h1`-Hashes (einen pro Plattform), Terraform nur einen. Die `zh`-Hashes sind bei beiden gleich, dieselben signierten Artefakte. Folge: OpenTofus Lock-File trägt über Plattformen, der Mac-gegen-Linux-CI-Ärger entfällt.

```
sha256sum terraform-provider-hcloud_v1.67.0
598a090eb8fb…  # registry.terraform.io
598a090eb8fb…  # registry.opentofu.org
```

03

Video 08:00

Mirror bauen, gleich für mehrere Plattformen

- [ ] Lege eine `.tofurc` an, die OpenTofu auf einen lokalen Ordner zeigen lässt statt aufs Netz.
- [ ] Baue den Mirror für alle Plattformen, die dein Team braucht, damit auch Kollegen auf Windows ihn nutzen können.

```
# .tofurc
provider_installation {
  filesystem_mirror {
    path    = "/root/mirror"
    include = ["registry.opentofu.org/…/hcloud"]
  }
}
# Mirror bauen, liegt dann im Volume:
tofu providers mirror ./mirror \
  -platform=linux_amd64 -platform=windows_amd64
```

04

Video 09:00

Offline verifizieren, die Probe aufs Exempel

- [ ] Starte einen zweiten Container mit demselben Volume, aber komplett ohne Netz: `--network none`.
- [ ] `tofu init` muss trotzdem sauber durchlaufen. Läuft es durch, kommen die Provider aus dem lokalen Mirror, und du bist vom Kanal unabhängig.

```
docker run --rm -it -v root:/root \
  --network none --entrypoint bash \
  ghcr.io/opentofu/opentofu:1.12.5
# im Container, ohne Internet:
tofu init
```

05

Video 06:30

Wer betreibt den Kanal?

REGISTRY.TERRAFORM.IO	REGISTRY.OPENTOFU.ORG
Betreiber: HashiCorp / IBM	Betreiber: Linux Foundation
Ein Unternehmen hält den Stift	Neutrale Stiftung, viele Beteiligte
Für fremde Software geschlossen	Verweist auf signierte GitHub-Releases

RÜCKVERSICHERUNG, KEIN ALLTAG

Den Mirror gibt es für Terraform genauso, er ist kein Alleinstellungsmerkmal. Der Unterschied ist die Wahrscheinlichkeit, dass du ihn brauchst: der OpenTofu-Kanal gehört keiner Firma, die ihn schließen kann. Der Mirror ist bei beiden der Feuerlöscher, bei OpenTofu ist die Brandgefahr kleiner.

Souveränität ist Technik plus Lizenz. Weitergeben erlaubt.

Dieser Spickzettel gehört zum Video über den Vertriebsweg von Terraform und OpenTofu. Jede Woche praxisnahe Antworten auf Docker- und Kubernetes-Probleme: der Newsletter.

trutz.io

Newsletter