Infrastructure as Code

Terraform & IaC

Was ist Infrastructure as Code

"[...] Infrastructure-as-Code (IaC) ist die Verwaltung von Infrastruktur (Netzwerken, virtuellen Computern, Lastenausgleichsmodulen und der Verbindungstopologie) in einem beschreibenden Modell. [...]"

Quelle

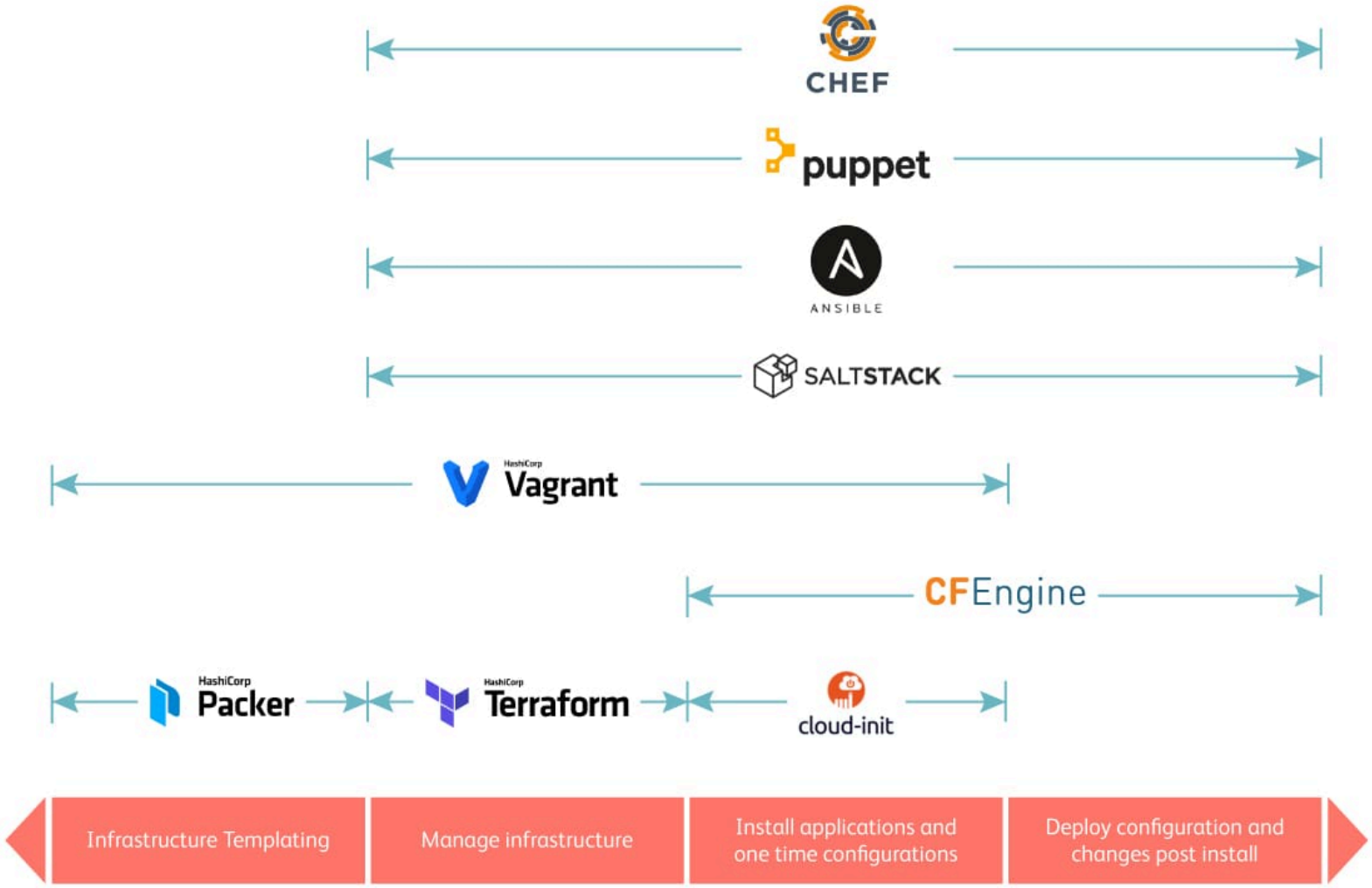
Was für Arten gibt es?

- Prozedurale Sprache - Wie erreiche ich den Zielzustand
- Deklarative Sprache - Was ist der Zielzustand

Deklarativ vs. prozedural

Deklarativ	Prozedural
Zielzustand is sichtbar	Zielzustand ist bedingt sichtbar
Aktueller Zustand ist sichtbar	Aktueller Zustand ist nicht sichtbar
Wiederverwendbar	Bedingt wiederverwendbar

IaC Bereiche



Vor- & Nachteile von deklarativem IaC

Vorteile	Nachteile
Transparente Infrastruktur => Risikovermeidung	Manuelle Konfigurationseingriffe können alles kaputt machen
Wiederholbar	Hoher Aufwand bei Konzeption & Umsetzung
Automatisierung der Infrastruktur	Know how über Cloudprovider APIs
Vorteile von Softwareentwicklung (Testbar, Versionierbar, Deployment Pipelines)	

Warum nutzen wir IaC?

- **Reproduzierbarkeit**: Umgebungen sind jederzeit neu erzeugbar (Dev/Test/Prod werden vergleichbarer)
- **Geschwindigkeit & Skalierung**: Provisioning und Änderungen laufen automatisiert und wiederholbar
- **Weniger Fehler**: weniger Klickpfade, weniger "vergessene" Einstellungen
- **Nachvollziehbarkeit**: Änderungen über Git/PRs (Review, Audit-Trail, Rollback)
- **Software-Prinzipien**: Tests, CI/CD, Security-Scans und Policy-as-Code sind möglich

Was gibt es für deklarative IaC Programme?

- AWS Cloud Formation
- Azure Resource Manager
- Google Cloud Deployment Manager
- Pulumi
- Terraform
- ...

Was ist Terraform

- Entwickelt von der Firma HashiCorp
- Released im Juli 2014 - 1.0 Release am 08.06.2021
- Deklarativer IaC
- Plattform unabhängig (Azure, AWS, vSphere)
- Unterstützt Hybrid Cloud Infrastruktur
- Unveränderbare Infrastruktur
- kein Agent
- kein Master Server

Funktionen von Terraform

- Integration von Plattformen über Provider
- Abhängigkeitsgraph
- Ausführungplan
- Inkrementelle Veränderungen

Warum nur via IaC ändern? (Config Drift)

Config Drift = Ist-Zustand weicht vom Code ab (z.B. Änderung im Portal/CLI/Hotfix).

- `terraform plan` zeigt plötzlich unerwartete Diffs
- Änderungen werden schwer reproduzierbar ("Snowflake"-Umgebungen)
- Debugging wird langsam: "Was ist wirklich live?"
- Risiko für Security/Compliance, weil Änderungen außerhalb von Reviews passieren

Prinzip: "Ändere die Quelle (Git), nicht das Ziel (Portal)."

Drift verhindern & erkennen

- **Verhindern**: Schreibrechte im Portal einschränken, Änderungen über Pipeline/PR erzwingen
- **Notfälle**: Break-glass nur temporär (z.B. PIM/JIT), danach Change in IaC nachziehen
- **Erkennen**: PR-Checks mit `terraform plan`, regelmäßige Drift-Checks (scheduled)
- **Beheben**: Drift bewusst in Code übernehmen *oder* per IaC wieder auf Soll-Zustand zurückführen

Terraform Code

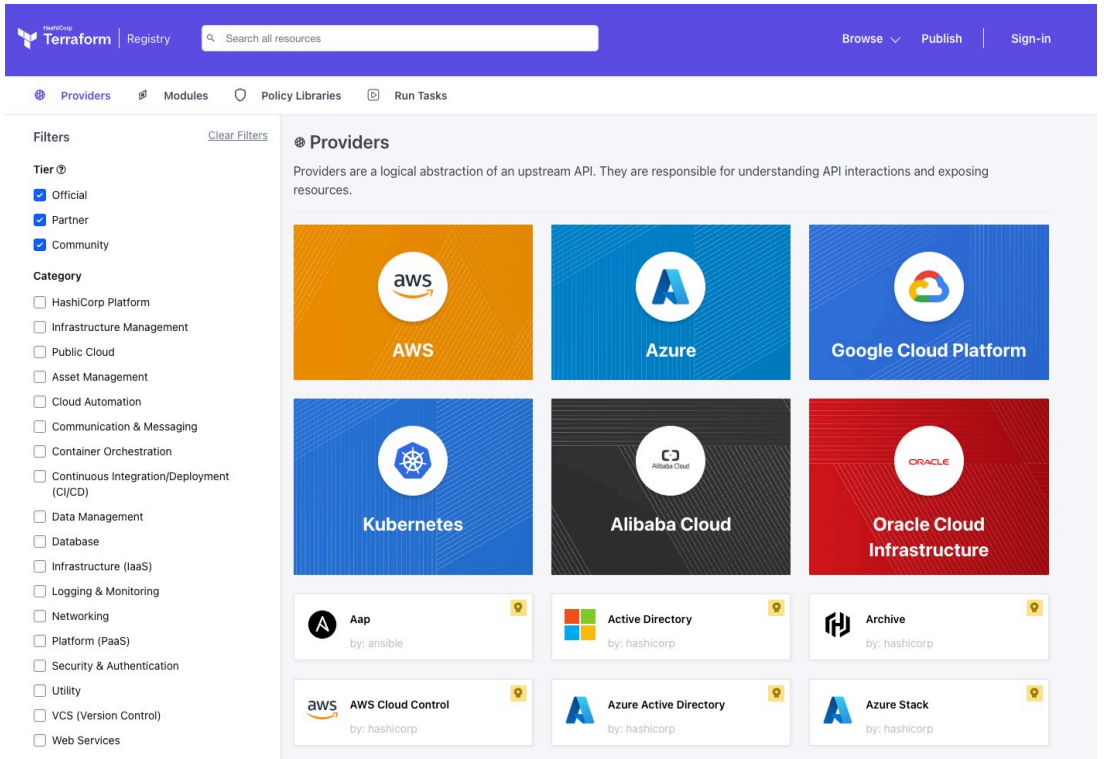
- HashiCorp Configuration Language (HCL)
- Domain Specific Language (DSL) für mehrere Produkte von HashiCorp
- Dateiendungen von Terraform: `.tf`, `.tfvars`, `.tfbackend`

Definition in Blöcken, z.B.:

```
terraform {}
provider {}
resource {}
data {}
locals {}
variable {}
output {}
```

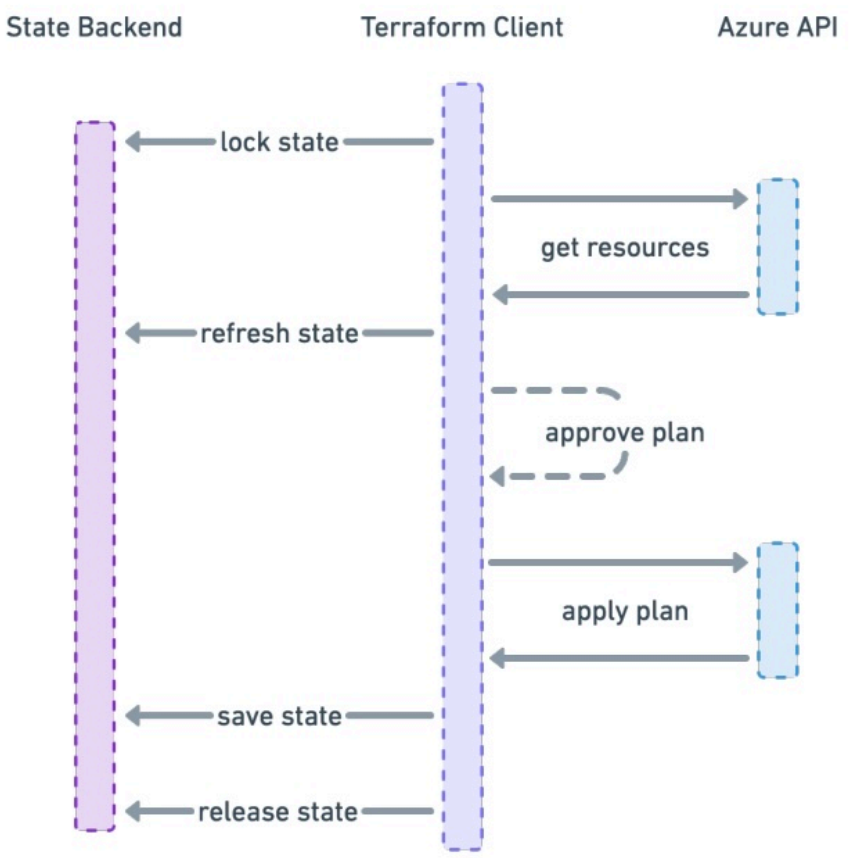
Terraform Registry

- Provider zur Anbindung an Cloud Provider
- Module zur Wiederverwendung von Code
- Für offizelle und Community Entwicklungen
- https://registry.terraform.io



Terraform State

- Speichert den aktuellen Zustand der Infrastruktur
- Standardmäßig lokal in der Datei `terraform.tfstate`
- Kann in einem Remote Backend gespeichert werden
- Locking des States bei Remote Backends
- Kann sensible Daten enthalten (z.B. Passwörter)



Variables & Locals

Variables

- Typisierung möglich: `string`, `number`, `bool`, `list`, `map`, *etc.*
- Standardwerte & Validierung möglich
- Können bei Modulen übergeben werden

```
variable "example"{
  type        = string
  description = "An example variable"
  default     = "foo"
}
```

Locals

- Lokale Variablen, nur im aktuellen Modul
- Können nicht von außen gesetzt werden
- Keine Typisierung, Validierung oder Standardwerte

```
locals {
  example = "foo"
}
```

Code Beispiele

Ende

Fragen ? -> Fragen

Quellen

- https://docs.microsoft.com/de-de/devops/deliver/what-is-infrastructure-as-code
- https://learn.microsoft.com/devops/deliver/what-is-infrastructure-as-code
- https://learn.microsoft.com/azure/cloud-adoption-framework/ready/considerations/infrastructure-as-code-updates
- https://www.computerweekly.com/de/ratgeber/Infrastructure-as-Code-Acht-beliebte-Tools-im-Vergleich
- https://www.redhat.com/de/topics/automation/what-is-infrastructure-as-code-iac