Wie wir mit Infrastructure-as-Data eine Plattform gebaut haben

Continuous Lifecycle / ContainerConf 2023

Simon Dreher & Alexander Huck

Team inovex
Karlsruhe · Köln · München · Hamburg
Berlin · Stuttgart · Pforzheim · Erlangen

inovex

About us

Alexander Huck

Cloud Platform Engineer & Neovim Aficionado

@alxndr13

alxndr13.xyz

Simon Dreher

Security & Cloud Platform Engineer

@SimonDreher

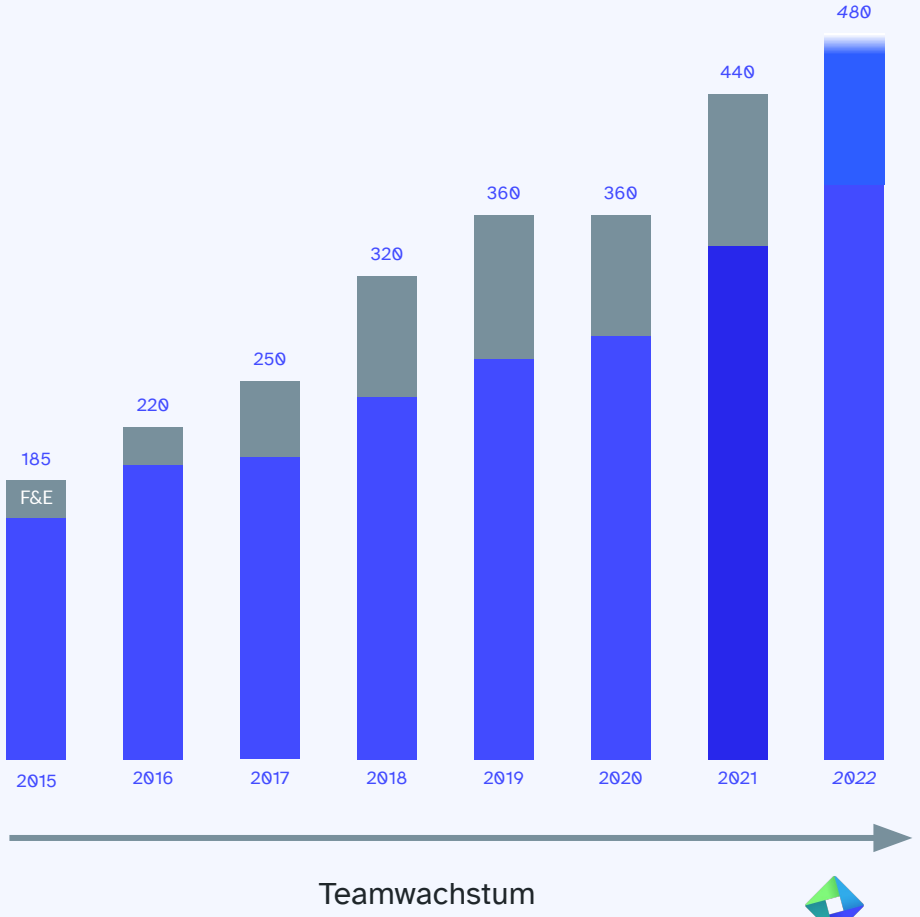
@DreherSimon@infosec.exchange

inovex

Unternehmensprofil

- Gründung: 1999
- inovex = Innovation + Exzellenz
- finanziell und inhaltlich unabhängig
- Umsatz 2022: ca. 45 Mio. €

- ausschließlich festangestellte Mitarbeiter:innen
- Wachstum durch Innovationen und erfolgreiche Projekte
- ca. 100 Mitarbeiter:innen für Forschung und Technologiebewertung



inovex

Standorte

inovex

markant

- Handelskooperation
- “Partner für Lieferanten und Händler”
- Services
  - Produktinformationen
  - Datenaustausch
  - Preisanalyse
  - Zahlungsservices / Zentralregulierung
  - ...

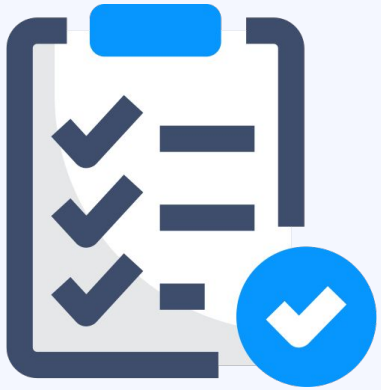
inovex

Challenge

inovex

Aufgabenstellung

- Ursprung: eigenes Rechenzentrum
- Cloud Erfahrung in einzelnen Teams gesammelt
- Neues digitales Produkt für B2B-Kunden
  - verschiedene Teams: n Services, Frontend, Billing, ...
  - Wiederholender Aufwand reduzieren



Aufgabenstellung - offene Fragen

- Verbindung von / nach on-premises?
- Verbindung von / zu Internet?
- Wie verbinden wir Services und Teams untereinander?
- Wie behalten wir den Überblick?
- Cloud Governance? Cloud Security?

inovex

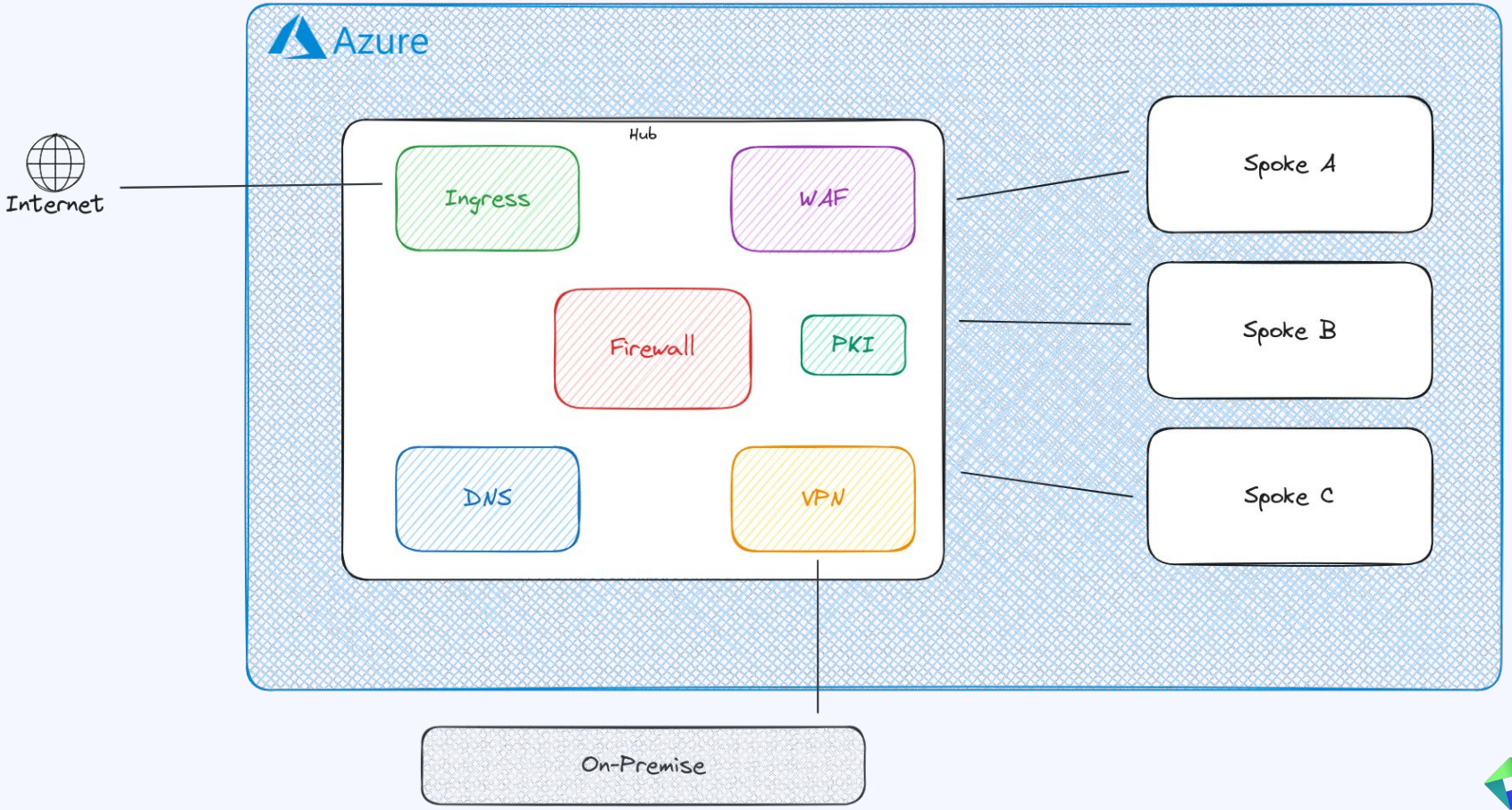
Aufgabenstellung - Constraints

- **IT-Security**: “Wenn ihr in die Cloud geht, dann darf es nur einen gesicherten zentralen Einstiegspunkt geben”
- **Compliance:** “Es gibt eine Cloud-Richtlinie - wäre schön wenn das alles umgesetzt würde.”

inovex

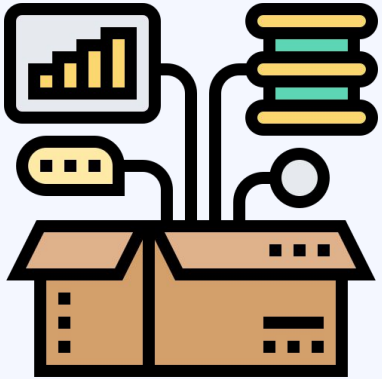
markhub

inovex


Azure
Internet
Hub
Ingress
WAF
Firewall
PKI
DNS
VPN
Spoke A
Spoke B
Spoke C
On-Premise

inovex

What’s in the box?



- **Private** Konnektivität zwischen Spokes
- Network Separation by default
- Public TLS Endpoint managed by us
- interne PKI für internen encrypted traffic, inkl. ACME Endpoint
- On-Premise Konnektivität
- Namensauflösung über alle Spokes und on-premise Grenzen

inovex

Wie deployen wir das?

inovex

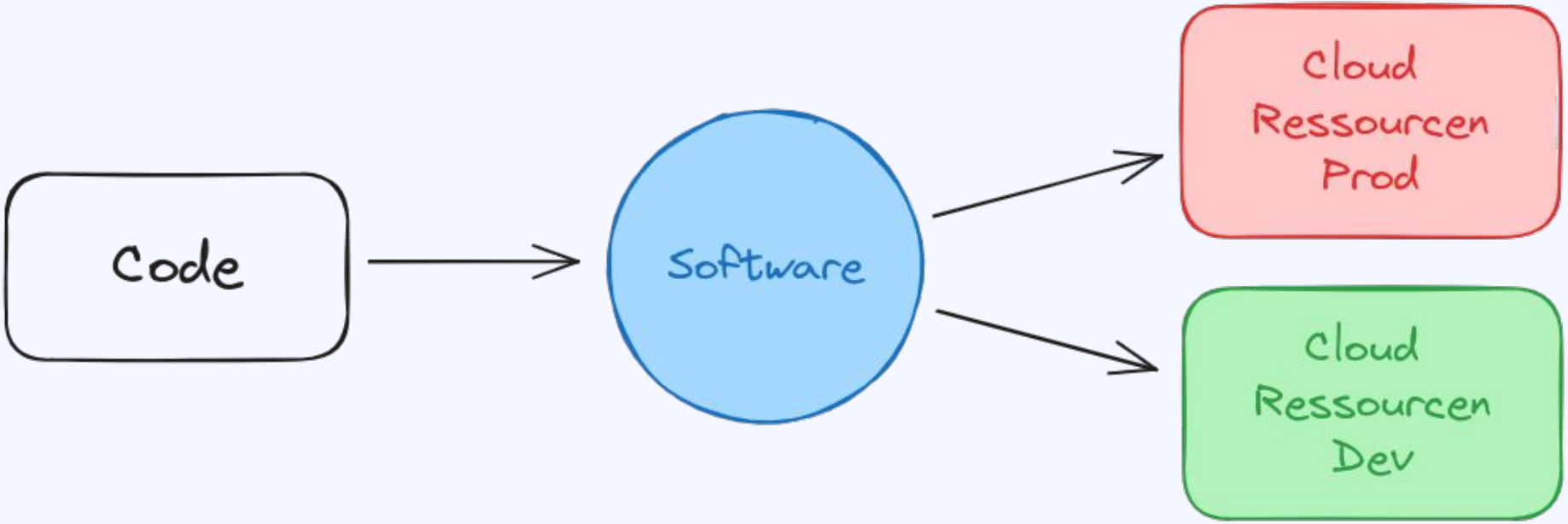
Azure Portal

Im Azure Portal klicken

Automatisieren

inovex

Infrastructure as Code



inovex

Imperativ

```
if !exists(network) {
    create network
    wait for network
}

if !exists(vm) {
    create vm in network
    …
}
…
```

Deklarativ

```
network a {

}

vm x {
    network: a
    ip: 13.3.3.7
}
```


inovex

Imperativ

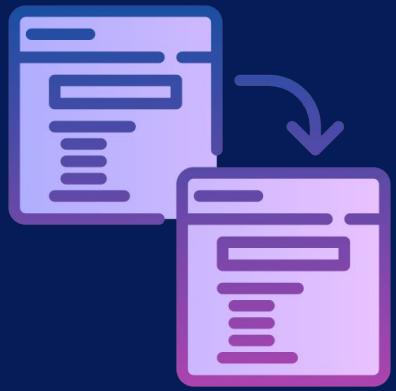
- Eher bekannt vom “normalen” Programmieren
- Einfacher - wenn man nicht alle Eventualitäten abdeckt
- Mehr Know-How benötigt

Deklarativ

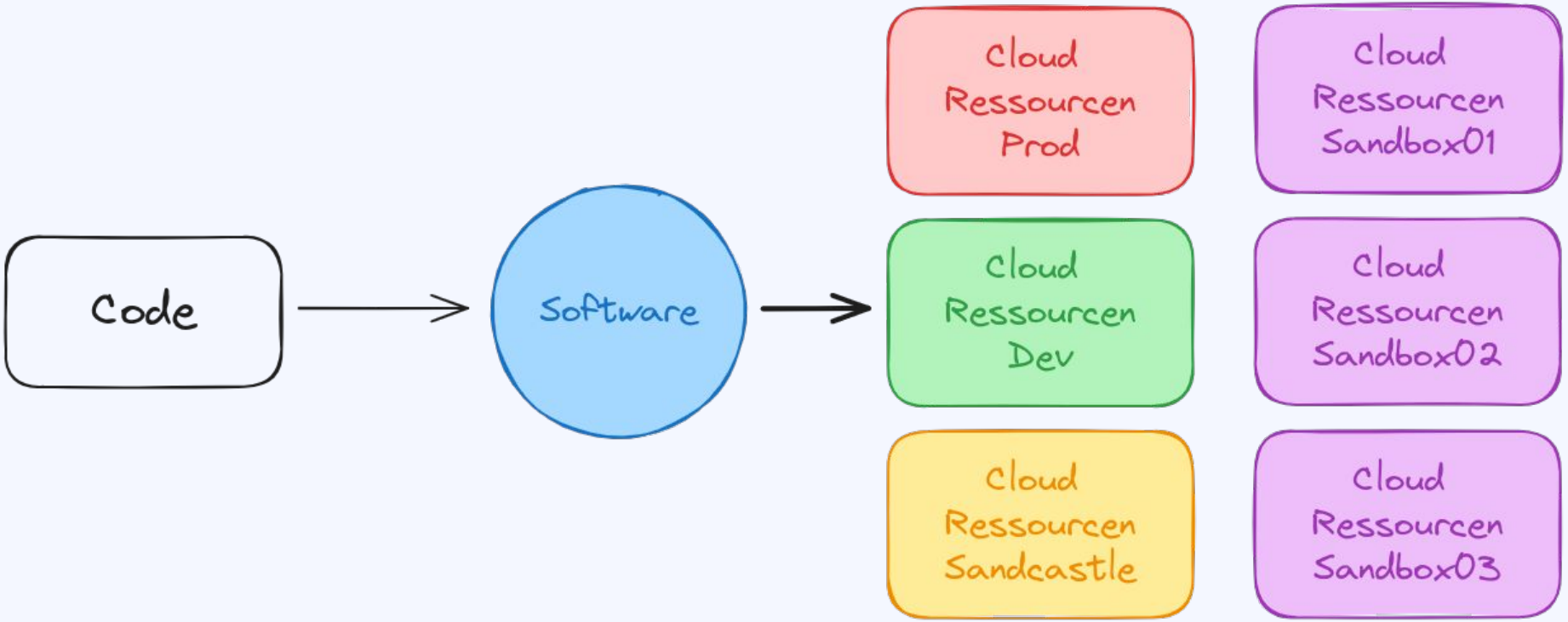
- Reproduzierbar
- Idempotent
- Code kann als “Dokumentation” dienen
- IaC + CI/CD = 💕

inovex

Herausforderung: viele ähnliche, aber nicht gleiche, Umgebungen



inovex


Code
Software
Cloud Ressourcen Prod
Cloud Ressourcen Dev
Cloud Ressourcen Sandcastle
Cloud Ressourcen Sandbox01
Cloud Ressourcen Sandbox02
Cloud Ressourcen Sandbox03

inovex

Mehrdimensionale Wiederholung

Spoke / Umgebung	prod	dev	…
Spoke A			
Spoke B			
Spoke C			
…			

inovex

Infrastructure as Data

inovex

Code:

“set of instructions”

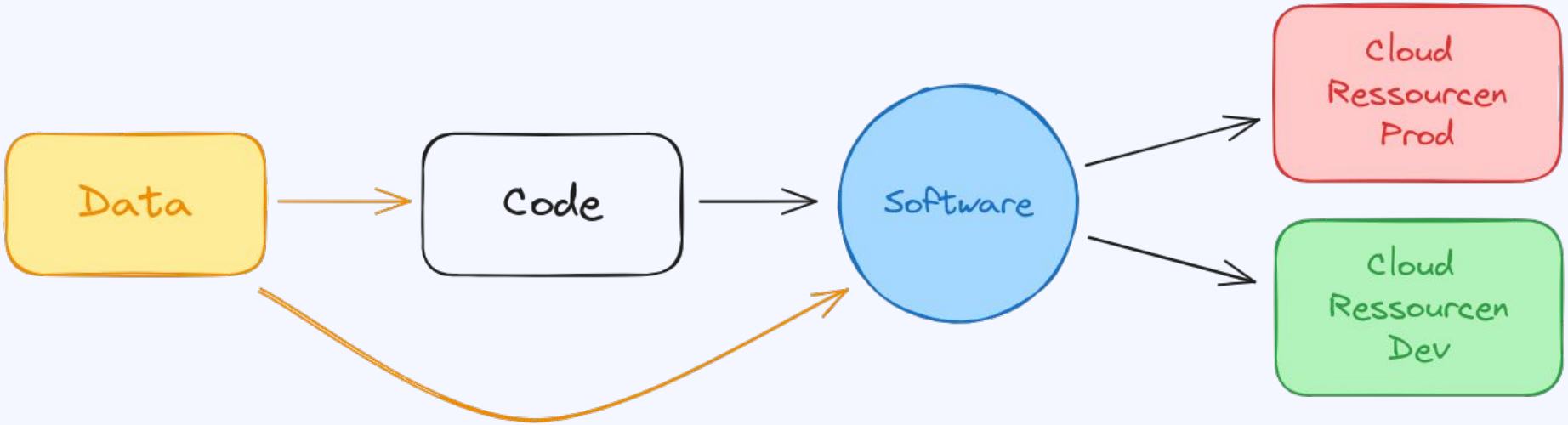
Data:

“collection of values that convey information”

*[…] “Infrastructure As Data”—describing what your systems look like in simple **machine-readable data formats**. Have **programs** execute those data formats and ensure your **infrastructure** matches.*

- *Michael DeHaan, 2013*

inovex


Data
Code
Software
Cloud Ressourcen Prod
Cloud Ressourcen Dev

inovex

```yaml
- id: "abcdef"
  name: "team1"
  subscription_id: "12345678-abcd-abcd-ef98-876543210"
  contact_email: "jordan.belfort@inovex.de"
  vnet_network_cidrs:
    - "10.204.2.0/24"
  members:
    - "12345678-abcd-abcd-ef98-876543210"
  policy_exemptions:
    - policy_assignment_name: "deny-paas-without-allowlist"
      policy_definition_reference_id: "Microsoft.ServiceBus/namespaces"
      reason: "Team 1 otherwise needs to add the full datacenter CIDR"
  services:
    - name: acme-service
      description: "The Dev Hubs ACME Service, providing certificates"
      fqdns:
        - "ca.acme.foo.bar"
      access_allowed_from_all_spokes: true
```


inovex

Vorteile

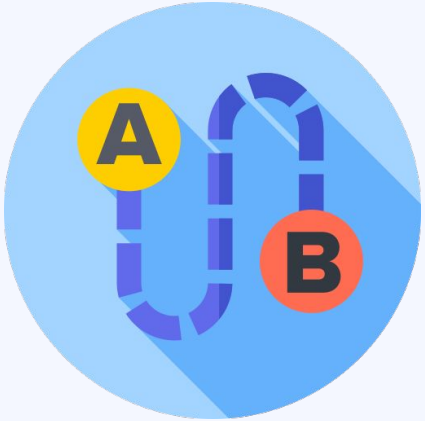
- Deklarativer Ansatz für den komplexen Teil “Code zu Realität”
- Freiheit wie Code generiert wird
- Trennung Daten und Code
  - Personelle Trennung
  - Daten validierbar

Nachteile

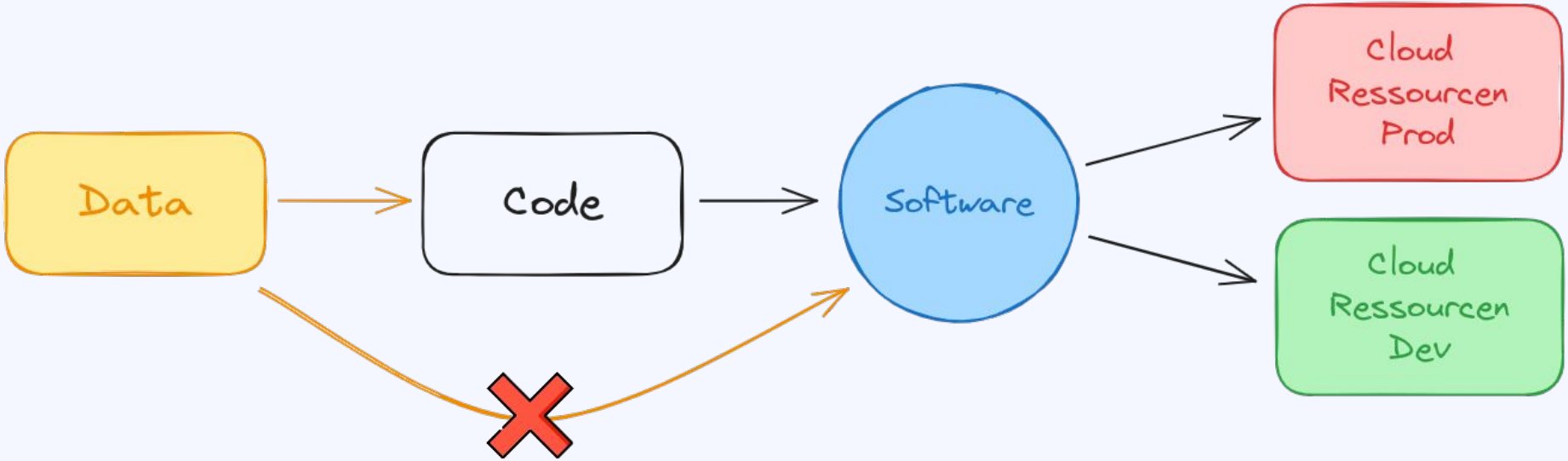
- Aufwand für Entwicklung und Wartung des Prozesses
- weiterer Abstraktionslayer

inovex

Wie kommt man nun von Daten zu Infrastruktur?



inovex


Data
Code
Software
Cloud Ressourcen Prod
Cloud Ressourcen Dev

inovex

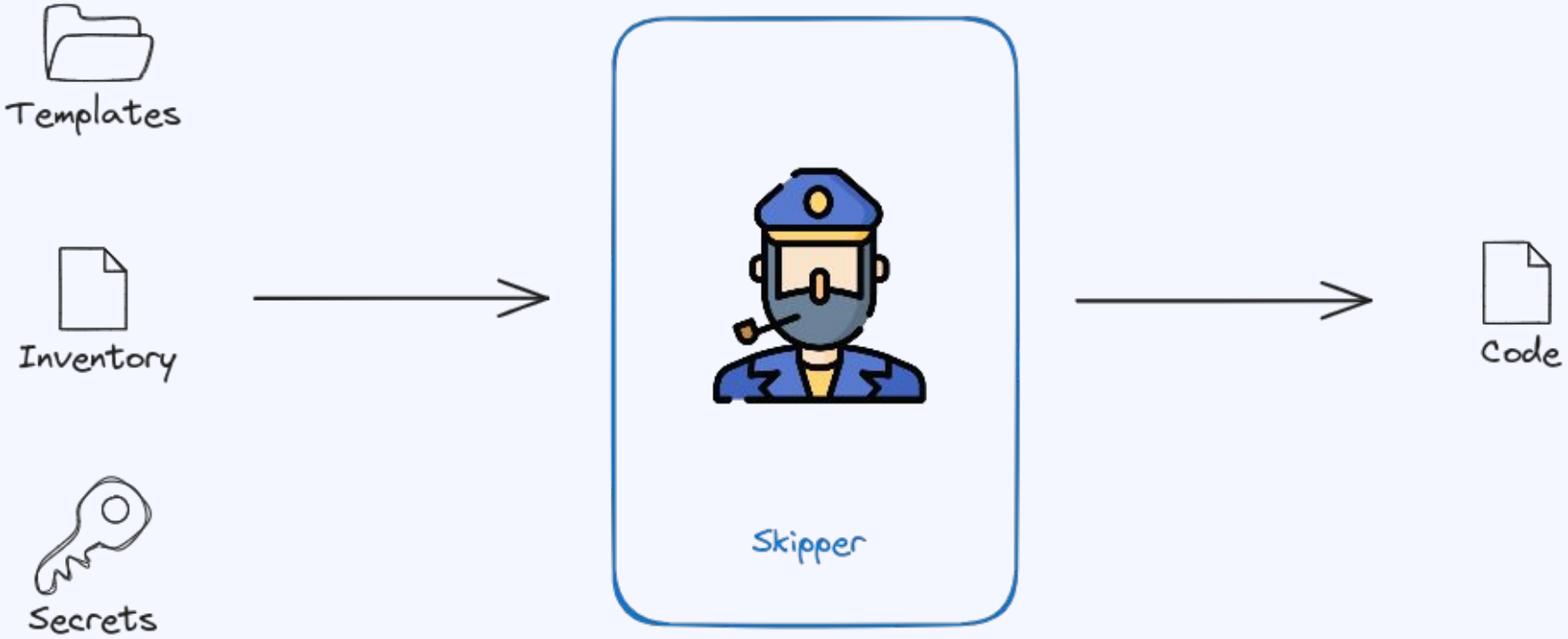
Konkrete Umsetzung

Skipper

Inventory based templated configuration library inspired by the kapitan project

https://github.com/lukasjarosch/skipper/

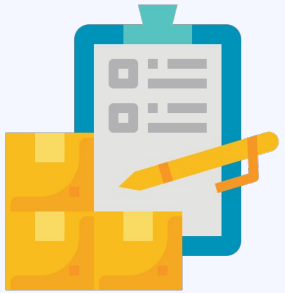
inovex



inovex

Inventory

- 💙 eines jeden Skipper-enabled Projekts
- Hier werden alle Informationen gespeichert bzw. laufen zusammen.
- “Single source of truth”
- Sammlung an YAML Files



inovex

Classes

- `Classes` erlauben einem das Inventory in mehrere Teile aufzusplitten
- Generell gerne genutzt um Defaults abzubilden
- Beispiele:
  - Eine `Class` für euren CoreDNS
- Kann in `Targets` importiert und verwendet werden

```yaml
coredns:
  version: "v0.0.14"
  instance_count: 2
  instance_sku: "Standard_B2s"
  fallback_dns:
    - "1.1.1.1"
  on_prem:
    domains:
      - "example.de"
      - "example.com"
    resolver_ip: "10.1.1.1"
```


inovex

Targets

- Ein `Target` repräsentiert eine “Instanz” eines Projektes (bspw. einer Plattform)
- Importiert `Classes` und kann deren Werte überschreiben
- Enthält alle Informationen für diese Instanz

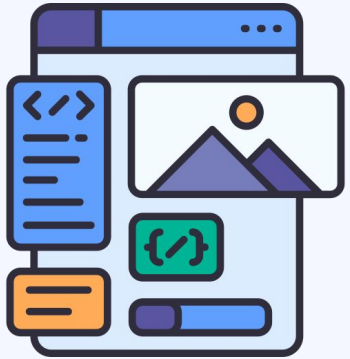
```
targets
├── dev.yaml
├── prod.yaml
├── sandbox01.yaml
├── sandbox02.yaml
├── sandbox03.yaml
└── sandcastle.yaml
```

```yaml
target:
  skipper:
    use:
      - acme
      - coredns
      - common
…
coredns:
  instance_count: 1
```


inovex

Templates

- `Templates` sind go-templates, welche mit den Informationen aus dem `Inventory` bestückt und anschließend gerendert wird
- Analoger Prozess zu `helm template`
- Was wir so damit rendern:
    - Makefiles, Bash Skripte, Terraform, Cloud-Init, Azure Policies, Tests, DNS Server Configs, Helm Values, etc.



Templates

```
resource "azurerm_firewall_policy_rule_collection_group" "default" {
  name               = "rcg-default"
  firewall_policy_id = azurerm_firewall_policy.default.id

  {{- if .Inventory.azure.s2s_vpn.enabled }}
  network_rule_collection {
    name     = "allow-to-all"
    action   = "Allow"

    rule {
      name                  = "from-vpn"
      protocols             = ["TCP", "UDP", "ICMP"]
      source_addresses      = [{{ .Inventory.vpn.on_prem_cidrs | tfStringArray }}]
      destination_addresses = ["10.0.0.0/8"]
      destination_ports     = ["*"]
    }
  }
  {{- end }}
```

Use Case: Secrets

Secrets

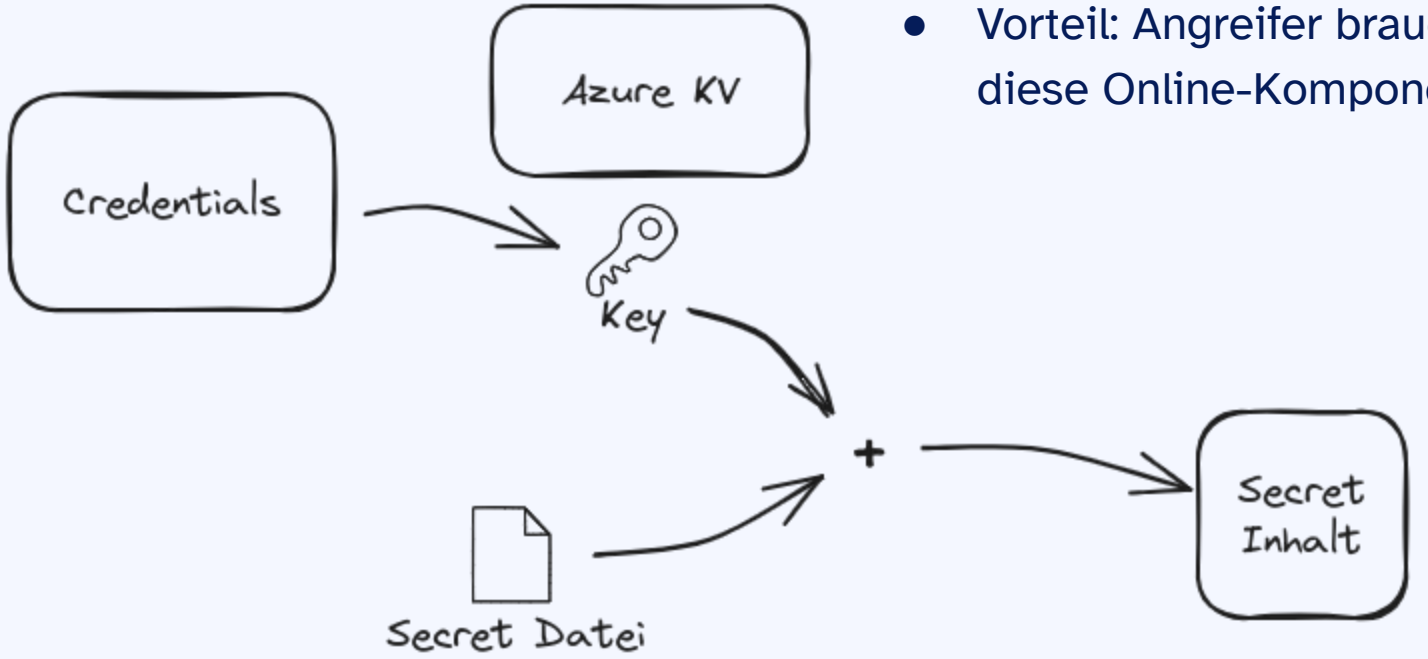
- `Secrets` sind in Dateien (verschlüsselt!)
  - Secret Driver: azurekv, AES
  - Versionierung
- Können im `Inventory` referenziert werden

```
1  data: Lz4kdkriufi/fk…
2  type: azurekv
```

```
1  service_principal:
2    client_id: "12345678-abcd-abcd-ef98-876543210"
3    client_secret: ?{azurekv:targets/dev/example}
```


inovex

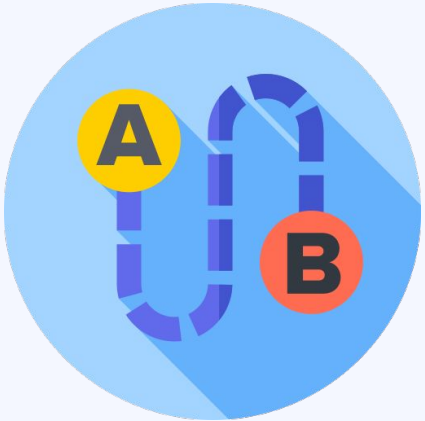
Wie komme ich an den Inhalt?



- Vorteil: Angreifer braucht auch diese Online-Komponente

inovex

Skipper ist ja aber nur eine Library..

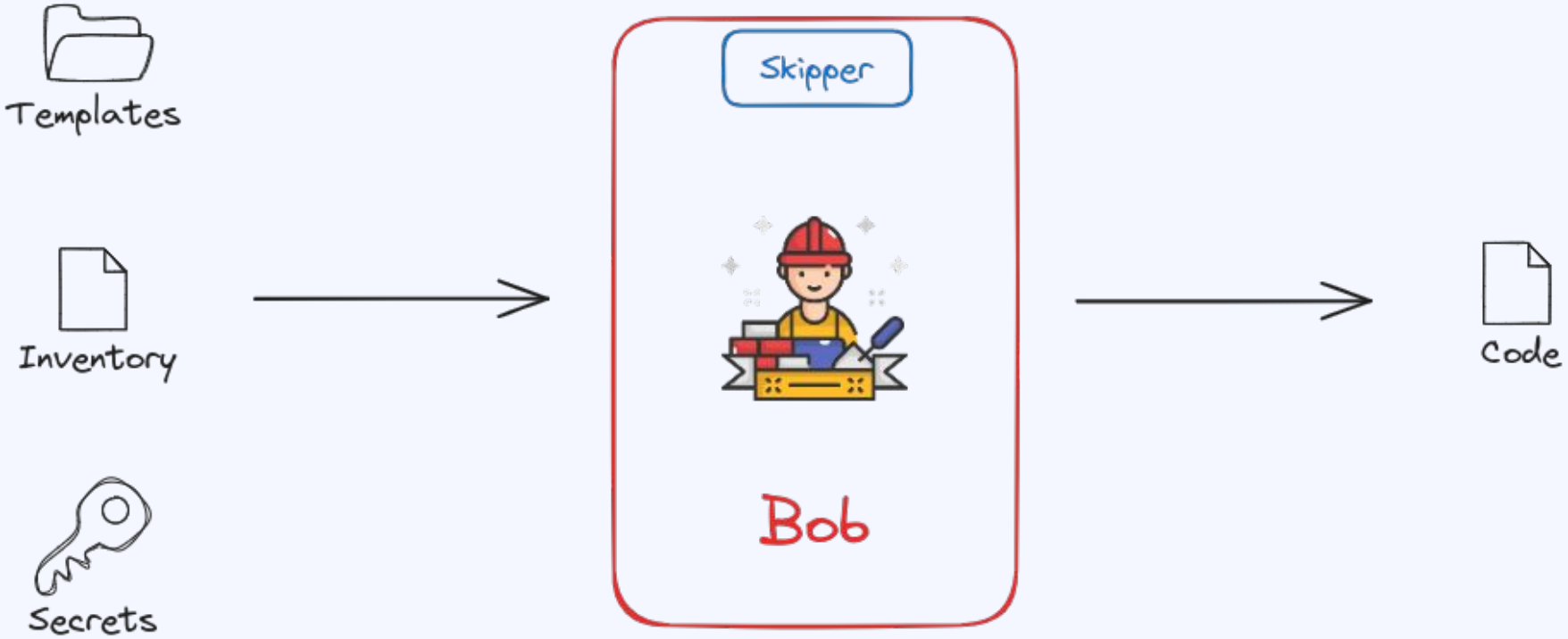


inovex

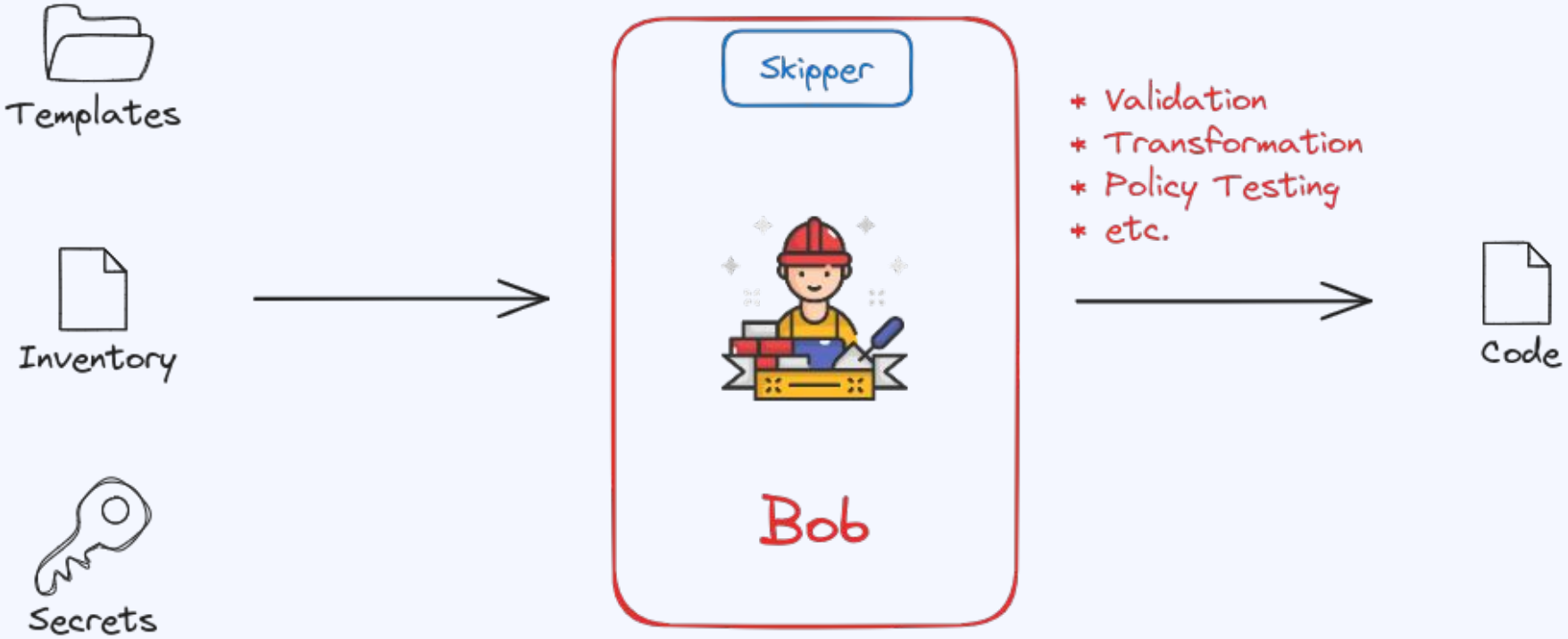
Bob

Our Infrastructure as Data Preprocessor

inovex



inovex


Templates
Inventory
Secrets
Skipper
Bob
* Validation
* Transformation
* Policy Testing
* etc.
Code

inovex

Use Case: Validierung

inovex

Validierungen

- Beispiele für Validierungen:
  - Hat die neue Spoke eine valide CIDR? Ist diese woanders schon vergeben?
  - Unique Spoke IDs?
  - Sind alle erforderlichen Felder im Inventar ausgefüllt?
  - Wurde eine valide Azure Location angegeben?

inovex

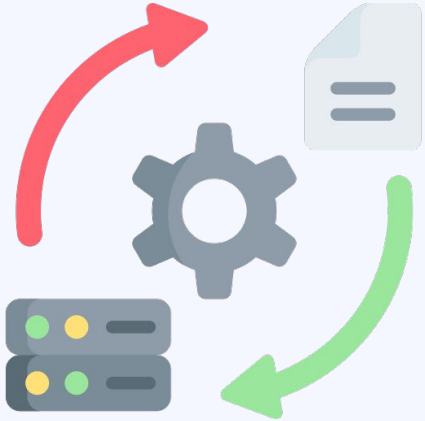
Validierungen

- Validierungen werden bei jedem `compile` durchgeführt
- Weil in einer “richtigen” Programmiersprache durchgeführt, können auch komplexe Validierungen schnell durchgeführt werden

```
2023/10/30 16:56:46 Errors occurred:
Inventory.spokes[2].vnet_network_ranges[0] failed validation for
field 'spokes[2].vnet_network_ranges[0]', param '', tag
'overlappingVnetNetworkCidrs' with value: '10.1.0.0/24'
```

Validierung - Code Beispiel

Use Case: Transformers



inovex

Beispiel: HAProxy Config

- In unserem Inventory pflegen wir eine Liste aller Public Entry Points für unseren Ingress inkl. des designierten Ziels
- *“Service YYY verwendet Port XXX”*

```
ingress:
  - fqdn: public-redis.${azure:dns:hub_zone}
    mode: tcp
    port: 8443
    service: apclws.redis
  - fqdn: servicebus1.${azure:dns:hub_zone}
    mode: tcp
    port: 8444
    service: apclws.sb1
```


inovex

Beispiel: HAProxy Config

- Für das HAProxy Config Rendering ist das aber doof, denn dort bräuchten wir die Daten anders herum
- ***“Port XXX zeigt auf Service YYY”*** anstelle *“Service YYY verwendet Port XXX”*

inovex

Beispiel: HAProxy Config

```
// TcpFrontends maps ports to a slice of frontend configurations.
type TcpFrontends map[uint][]FrontendConfig

// FrontendConfig is a representation of a TCP frontend used for HAProxy
configuration
type FrontendConfig struct {
     FQDN    string `yaml:"fqdn"`
     Service string `yaml:"service"`
}
```


inovex

Beispiel: HAProxy Config

```
// IngressToTcpFrontends transforms the ingress data structure to a structure
which is more easy to use for the haproxy config template
// It iterates over all TCP hosts and collects the frontend specifications per
port.
func IngressToTcpFrontends(ingress schema.Ingress) TcpFrontends {
     frontends := make(TcpFrontends)
     for _, host := range ingress.Hosts {
          if host.Mode != "tcp" {
               continue
          }
          frontends[host.Port] = append(frontends[host.Port], FrontendConfig{
               FQDN:    host.FQDN,
               Service: host.DefaultService,
          })
     }
     return frontends
}
```


inovex

Beispiel: HAProxy Config

```
{{- range $port, $hosts := .Inventory.transformed.haproxy.tcp_frontends }}
frontend tcp_{{ $port }} from tcp
  bind *:{{ $port }} ssl crt /etc/hapee-2.6/certs
  {{- range $index, $host := $hosts }}
  use_backend {{ $host.service }} if { ssl_fc_sni {{ $host.fqdn }} }
  {{- end }}
{{- end }}
```


inovex

Vor- und Nachteile

Nachteile

- Golang Know-How notwendig
- Wartung des Tools
- nach IaC rendern kann bei größeren Änderungen für Herausforderungen sorgen
- Kein kontinuierliches Reconcilen

inovex

Vorteile

- Pflege der Daten erfordert **kein** technisches Know-How
- **Volle** Flexibilität
- Golang-Templates bekannt von `Helm`
- Statisches Binary
- Secrets **sicher** und **versioniert** im Repo
- Separate Versionierung von Daten und Code
- Weniger Toil

inovex

Fazit

- Für unseren Case eine gute Option
- Mit Sicherheit nicht überall passend
- Trennung von Daten und Code öffnet Türen
  - Self Service MRs
  - Self Service APIs
- Infrastruktur + Software Development = a good collab

inovex

Danke an **inovex** für den Support. Danke an **markant** für die Möglichkeit das hier vorstellen zu dürfen. Danke an **flaticons.com** für die Bilder und Icons in den Slides. Danke an **ThePrimeagen** für den Neovim Content.

inovex