

ADC



**GitOps – Infrastructure as Code,
aber richtig**

Markus Lippert



- Business
Team & Technical Lead
bei COSMO CONSULT
- Social und Blog
lippertmarkus bei LinkedIn
Blog und BlueSky Account unter
lippertmarkus.com





Tobias Fenster



- Business
Managing Director bei 4PS und
Chief Engineer bei Hilti
- Community
MS Regional Director und
MVP für Azure und BizApps
Docker Captain
- Social, Blog und Podcast
tobiasfenster bei X und LinkedIn
tobiasfenster.io
„Window on Technology“ podcast

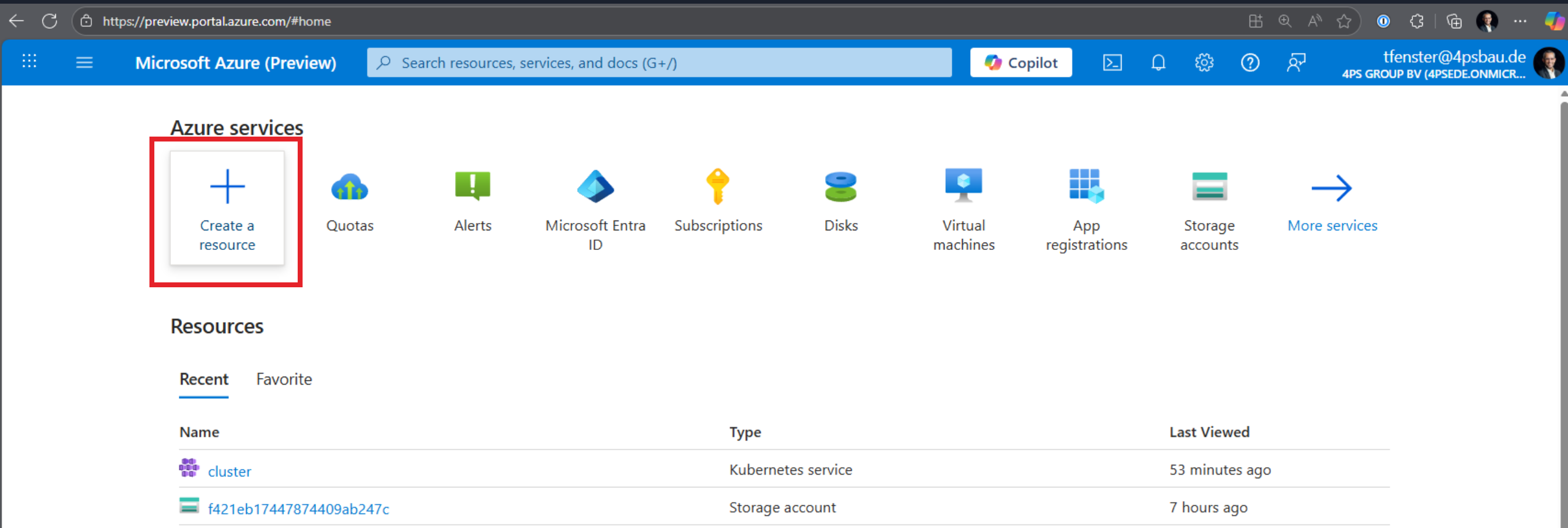


Agenda

- Einführung in das Thema und die Herausforderung
- Was ist GitOps und warum ist es hilfreich?
- Überblick über den Tech-Stack
- Deep-Dive und Live-Demo
- Best Practices und Lessons learned
- Q&A



Machst du noch ClickOps? Oder schon IaC?



https://preview.portal.azure.com/#home

Microsoft Azure (Preview) Search resources, services, and docs (G+/) Copilot



tfenster@4psbau.de
4PS GROUP BV (4PSEDE.ONMICR...)

Azure services

- Create a resource
- Quotas
- Alerts
- Microsoft Entra ID
- Subscriptions
- Disks
- Virtual machines
- App registrations
- Storage accounts
- More services

Resources

Recent Favorite

Name	Type	Last Viewed
 cluster	Kubernetes service	53 minutes ago
 f421eb17447874409ab247c	Storage account	7 hours ago



Machst du noch ClickOps? Oder schon IaC?

The screenshot shows the Microsoft Azure portal interface. At the top, there's a navigation bar with the Microsoft Azure (Preview) logo, a search bar, and a Copilot button. Below the navigation bar, the main heading is "Create a resource". Underneath, there are three buttons: "I want to duplicate an existing VM", "I need a new low-cost VM", and "Help me build a new Azure OpenAI application".

On the left side, there's a "Categories" list with items like Machine Learning, AI Apps and Agents, Analytics, Blockchain, Compute, Containers, Databases, Developer Tools, DevOps, and Identity. The "Virtual machine" category is highlighted with a red box.

In the center, there's a search bar for "Search services and marketplace" and a link to "Getting started? Try our Quickstart Center". Below this, there are two sections: "Popular Azure services" and "Popular Marketplace products".

The "Popular Azure services" section lists:

- Function App (Create)
- Web App (Create)
- Virtual network (Create)
- Key Vault (Create)
- Virtual machine (Create) - highlighted with a red box

The "Popular Marketplace products" section lists:

- Windows Server 2022 Datacenter: Azure Edition Hotpatch (Create | Learn more)
- Ubuntu Server 22.04 LTS (Create | Learn more)
- Windows 11 Enterprise N, version 22H2 (Create | Learn more)
- Free SQL Server License: SQL Server 2022 Developer on Windows Server 2022 (Create | Learn more)
- Red Hat Enterprise Linux10 (latest minor version) (Create | Learn more)

Machst du noch ClickOps? Oder schon IaC?

The screenshot shows the 'Create a virtual machine' wizard in the Microsoft Azure portal. The browser address bar shows the URL: `https://preview.portal.azure.com/#create/Microsoft.VirtualMachine`. The page header includes the Microsoft Azure (Preview) logo, a search bar, and the Copilot icon. The user's profile is identified as `tfenster@4psbau.de` from 4PS GROUP BV. The breadcrumb trail is `Home > Create a resource >`. The main heading is `Create a virtual machine`. The wizard has several tabs: `Basics` (selected), `Disks`, `Networking`, `Management`, `Monitoring`, `Advanced`, `Tags`, and `Review + create`. Below the tabs, there is a descriptive paragraph: `Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. Learn more`. The `Project details` section contains instructions: `Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.` It features two dropdown menus: `Subscription *` (set to `Construct DE - Test`) and `Resource group *` (set to `(New) Resource group`), with a `Create new` link below the second dropdown. The `Instance details` section includes a `Virtual machine name *` text input field and a `Region *` dropdown menu (set to `(Europe) Germany West Central`), with a `Deploy to an Azure Extended Zone` link below it.

Home > Create a resource >

Create a virtual machine

Basics Disks Networking Management Monitoring Advanced Tags Review + create

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * `Construct DE - Test`

Resource group * `(New) Resource group`
[Create new](#)

Instance details

Virtual machine name *

Region * `(Europe) Germany West Central`
[Deploy to an Azure Extended Zone](#)

Machst du noch ClickOps? Oder schon IaC?

Microsoft Azure (Preview) Search resources, services, and docs (G+/) Copilot tfenster@4psbau.de 4PS GROUP BV (4PSEDE.ONMICR...)

Home > ml-experimente

ml-experimente | Networking

Virtual machine

Search Feedback Attach network interface Detach network interface

ml-experimente772-46ce370b

IP configuration

Network Interface: ml-experimente772-46ce370b Effective security rules Troubleshoot VM connection issues Topology

Virtual network/subnet: vnet-germanywestcentral/snet-germanywestcentral-1 NIC Public IP: **4.182.232.69** NIC Private IP: **172.16.0.4**
Accelerated networking: **Enabled**

Inbound port rules Outbound port rules Application security groups Load balancing

Network security group **ml-experimente-nsg** (attached to network interface: **ml-experimente772-46ce370b**)
Impacts 0 subnets, 1 network interfaces

Add inbound port rule

Priority	Name	Port	Protocol	Source	Destination	Action
300	⚠ RDP	3389	TCP	Any	Any	✔ Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	✔ Allow

Machst du noch ClickOps? Oder schon IaC?

Probleme mit ClickOps

- Manuelles Arbeit
 - Anfällig für Fehler
 - Nicht zuverlässig reproduzierbar
 - Zu langsam
- Keine Versionsverwaltung
- Keine Historie darüber, wer was wann und warum getan hat

- → Kein professioneller Ansatz



Machst du noch ClickOps? Oder schon IaC?

```
resource virtualMachines_ml_experimente_name_resource 'Microsoft.Compute/virtualMachines@2024-11-01' = {
  name: virtualMachines_ml_experimente_name
  location: 'germanywestcentral'
  zones: [
    '1'
  ]
  properties: {
    hardwareProfile: {
      vmSize: 'Standard_D8a1ds_v6'
    }
    storageProfile: {
      imageReference: {
        publisher: 'microsoftwindowsdesktop'
        offer: 'windows-11'
        sku: 'win11-24h2-pro'
        version: 'latest'
      }
      osDisk: {
        osType: 'Windows'
        name: '${virtualMachines_ml_experimente_name}_OsDisk_1_13dfbcae423642d49e3dcb9b7f504a9d'
        createOption: 'FromImage'
        caching: 'ReadWrite'
      }
    }
  }
}
```



Machst du noch ClickOps? Oder schon IaC?

Probleme mit Infrastructure as Code (IaC)

- Versionierung und Historie (der Definition) sind jetzt möglich
- Aber:
 - Manuelle Eingriffe oder unbeabsichtigte automatisierte Änderungen können trotzdem erfolgen („Drift“)
 - Wird nicht kontinuierlich aktualisiert (sofern das IaC-Tool dies überhaupt kann), sondern nur auf Anforderung (Push)
 - Eingeschränkte Nachverfolgbarkeit
- → Ein großer Schritt in die richtige Richtung, aber auch noch große Probleme



Was ist GitOps und warum ist es hilfreich?

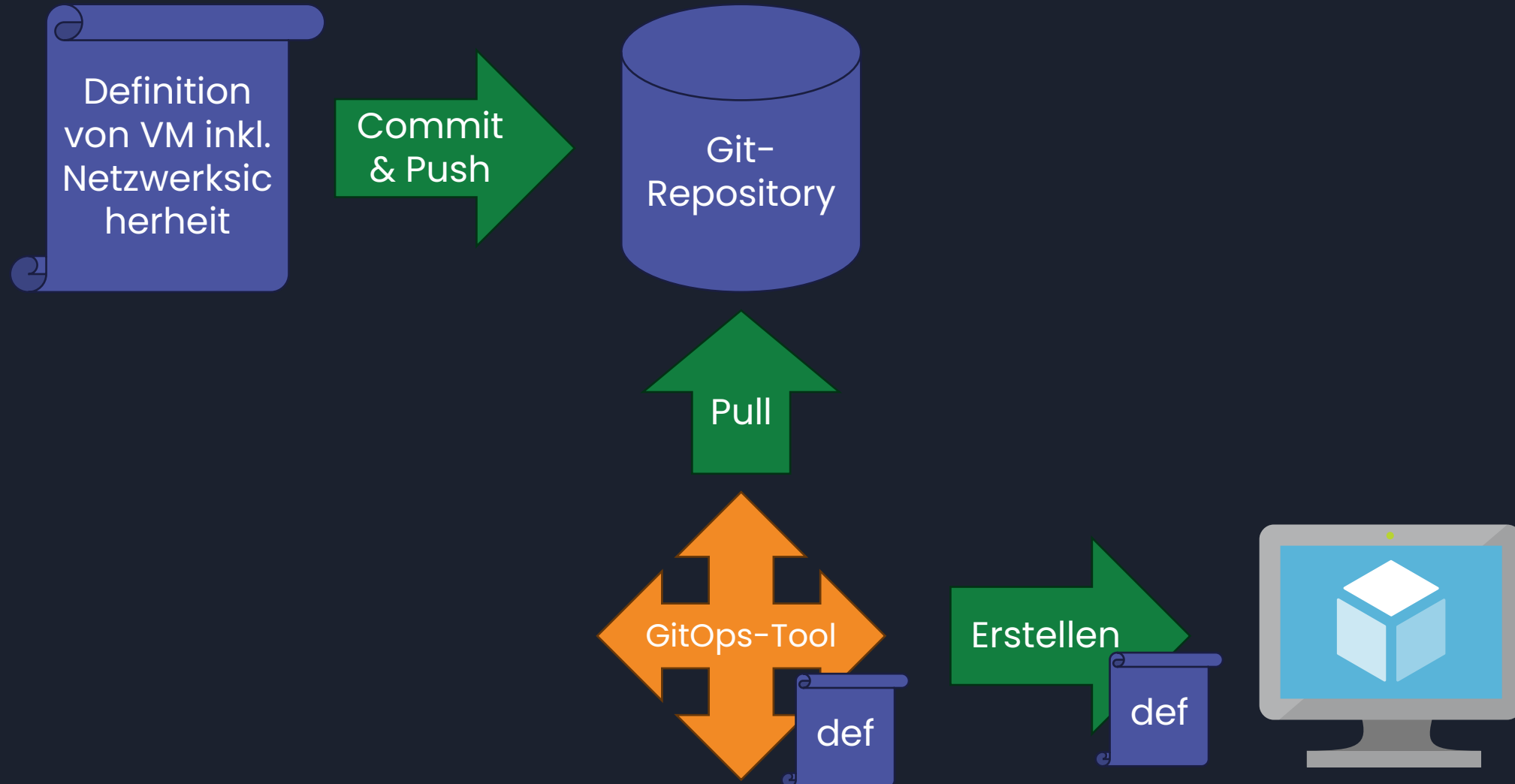
Grundlegende Konzepte

- Deklarativ: Man beschreibt, was man will, nicht wie man dorthin kommt
 - IaC kann auch imperativ sein, z. B. mit PowerShell oder anderen Skripten
- Git als "einzige Quelle der Wahrheit"
 - Keine lokale Ausführung von IaC-Definitionen mit Abhängigkeiten von oder Beeinflussung durch lokale Komponenten
 - Ermöglicht Sicherheits- und Compliance-Prüfungen auf PR-Ebene
- Pull-basiert
 - IaC ist typischerweise push-basiert: „Ich sage der Cloud, was zu tun ist“
 - GitOps ist pull-basiert: Ein System stellt sicher, dass die Realität dem gewünschten Zustand entspricht
- Identifiziert „Drift“ und löst ihn auf
 - Analysiert automatisch den Ressourcenstatus
 - Wendet bei Bedarf Änderungen an den Ressourcen an, damit diese der Definition entsprechen
 - Eventuelles Fehlschlagen kann überwacht werden
- Konsistenz über Umgebungen hinweg (z. B. Dev / Stage / Prod)
 - Gleicher Code → gleiche Konfiguration. Abweichungen leicht erkennbar



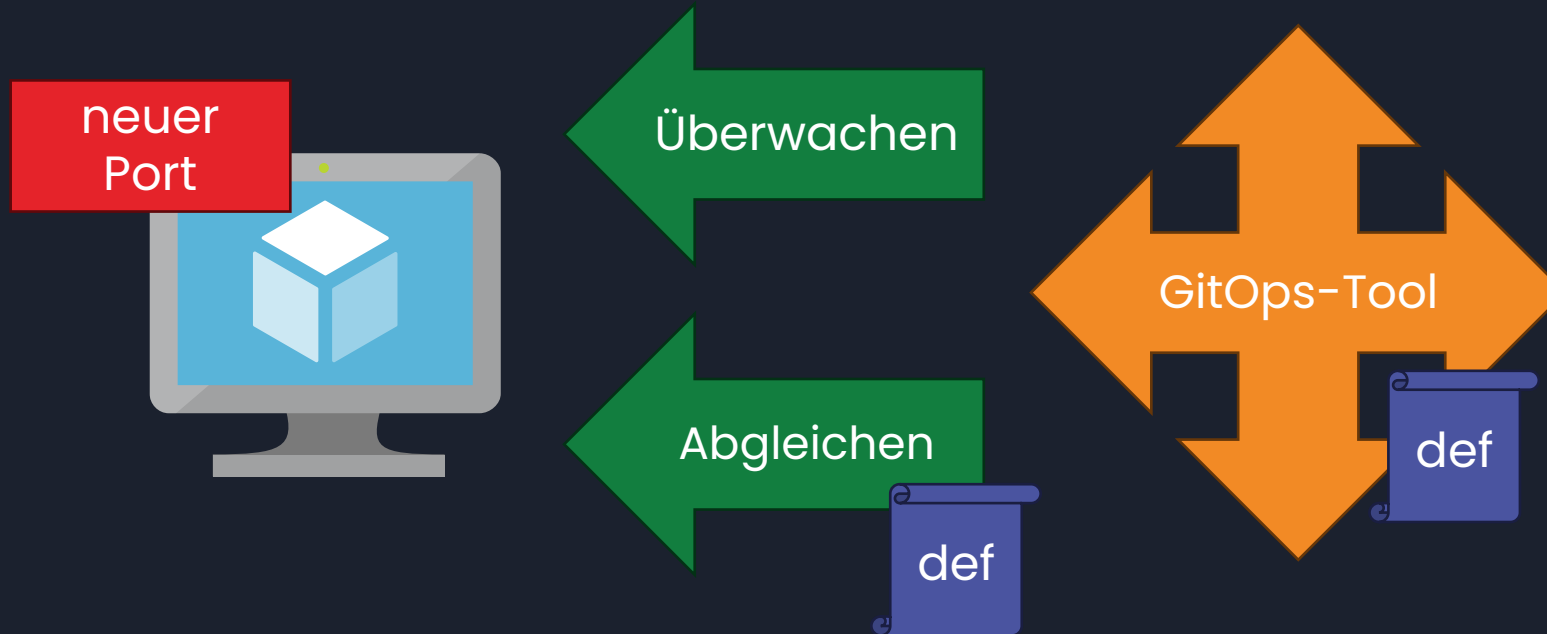
Was ist GitOps und warum ist es hilfreich?

Grundlegende Konzepte: Pull-basiert mit Git-Repository



Was ist GitOps und warum ist es hilfreich?

Grundlegende Konzepte: Drift und Abgleich



Der Tech-Stack im Überblick

- Kubernetes: De-facto-Standard-Orchestrierungs-Engine für Container-Workloads
 - Die Antwort auf „Ich möchte viele Container mit gegenseitigen Abhängigkeiten betreiben“
 - Von Google entwickelt, von vielen großen Tech-Unternehmen (Netflix, Shopify, Paypal, auch Microsoft) genutzt und mittlerweile auch bei kleineren Unternehmen sowie in anderen Szenarien verbreitet; mittlerweile Teil der CNCF
- Crossplane: Kubernetes-natives, GitOps-freundliches IaC-Tool zur Bereitstellung von Cloud-Ressourcen
 - Antwort auf „Ich möchte einen Kubernetes-Cluster und abhängige Ressourcen irgendwo in der Cloud bereitstellen“
 - Entwickelt von Upbound, mittlerweile Teil der CNCF
- Argo CD: GitOps-Tool zur Bereitstellung von Anwendungen auf Kubernetes
 - Antwort auf „Ich möchte einen Anwendungs-Container auf meinem Kubernetes-Cluster“
 - Entwickelt von Applatix, mittlerweile Teil der CNCF



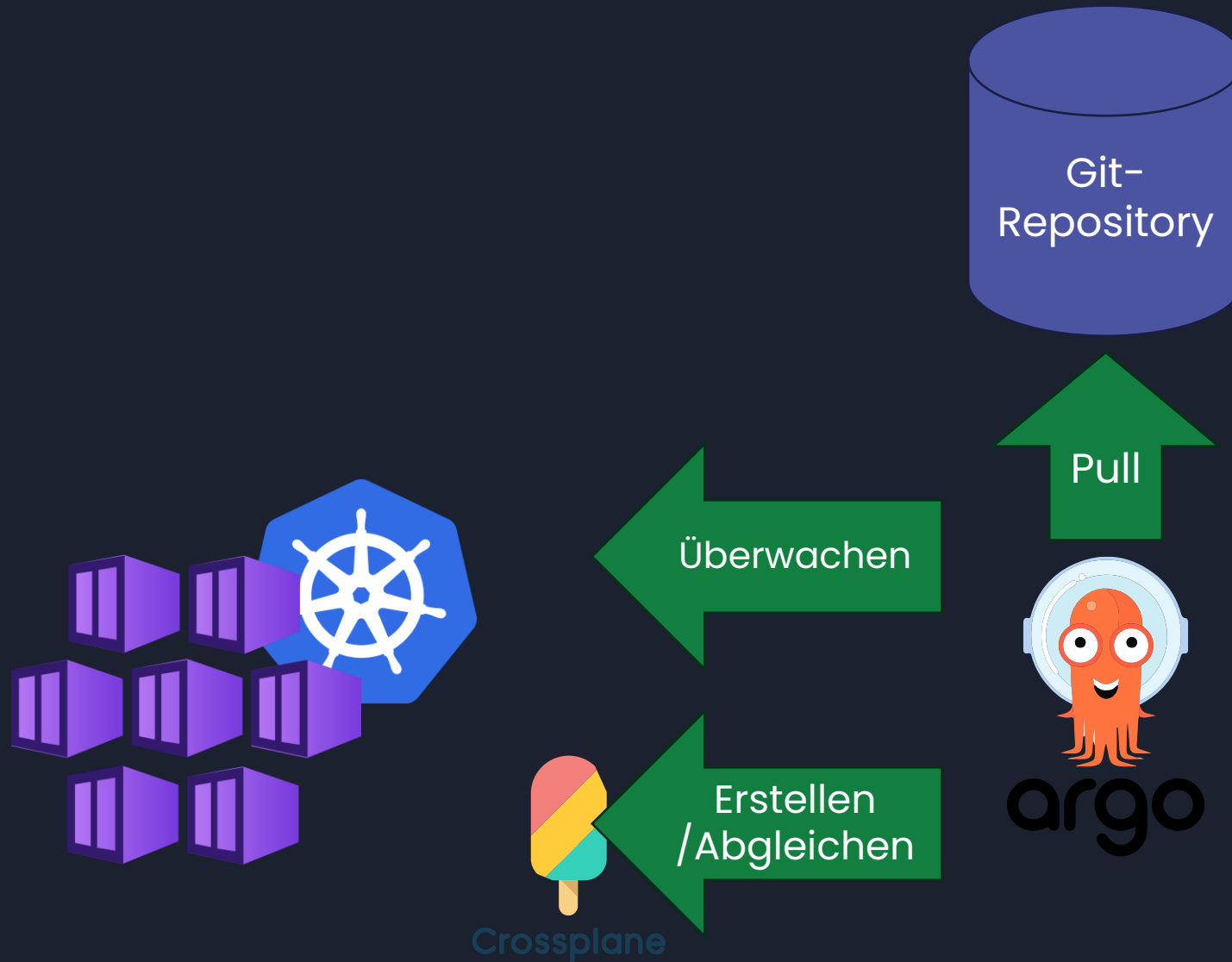
Crossplane



argo



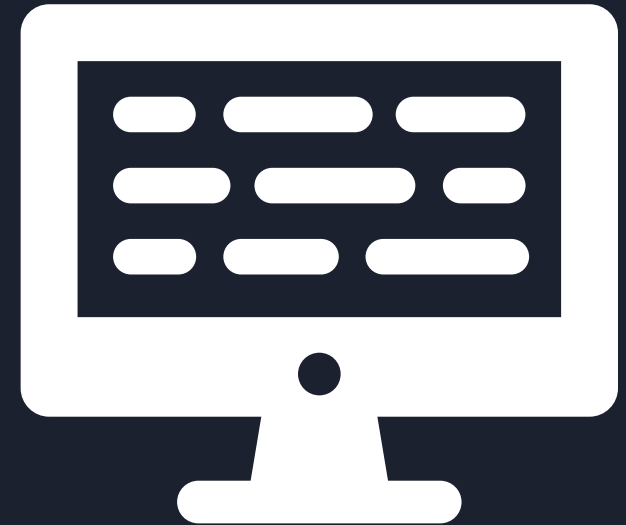
Der Tech-Stack im Überblick



Deep-Dive und Live-Demo

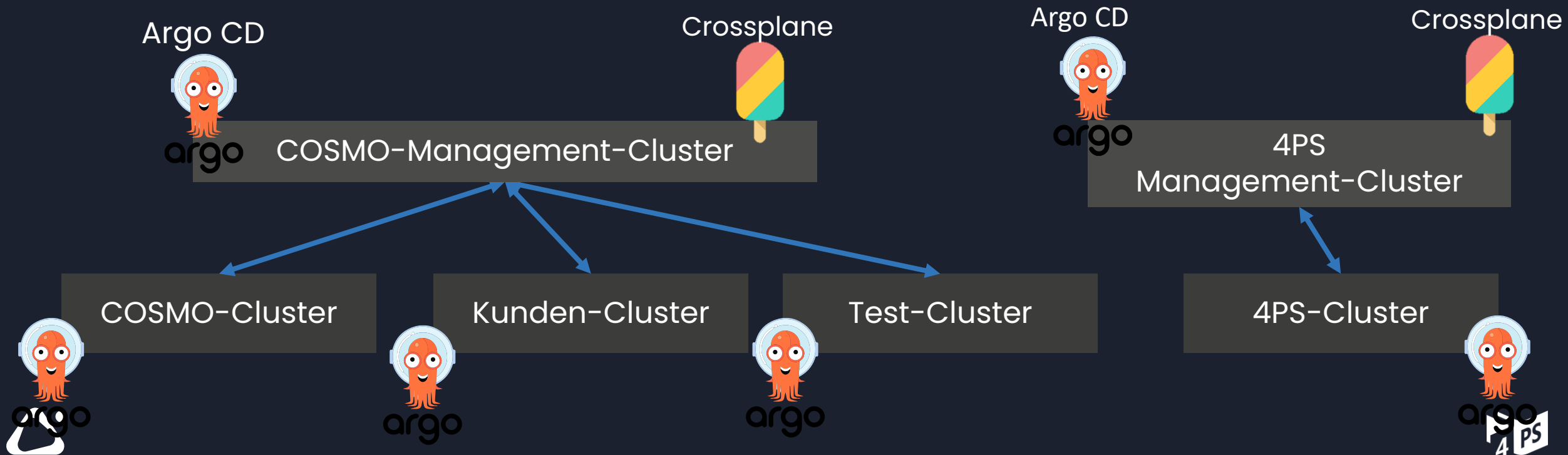
Wir schauen uns an:

- Infrastrukturänderungen mit dem GitOps-Ansatz durchführen
- Änderungen rückgängig machen
- Erkennung und Behandlung von Abweichungen ("Drift")
- Rückverfolgbarkeit und Überprüfbarkeit von Änderungen



Ein Beispiel für GitOps aus der Praxis von COSMO Alpaca

- COSMO Alpaca nutzt den Azure Kubernetes Service (AKS) zum Hosten verschiedener C#-Web-APIs und Windows-Anwendungscontainer
- 4 Cluster werden von COSMO über GitOps mit Argo CD und Crossplane verwaltet, 2 Cluster von 4PS
- Test-/Kunden-/COSMO-Cluster sind ähnlich; Änderungen werden über PRs durch diese Cluster („Stages“) ausgerollt



Ein Beispiel für GitOps aus der Praxis Infrastruktur

AKS-Cluster werden deklarativ definiert:

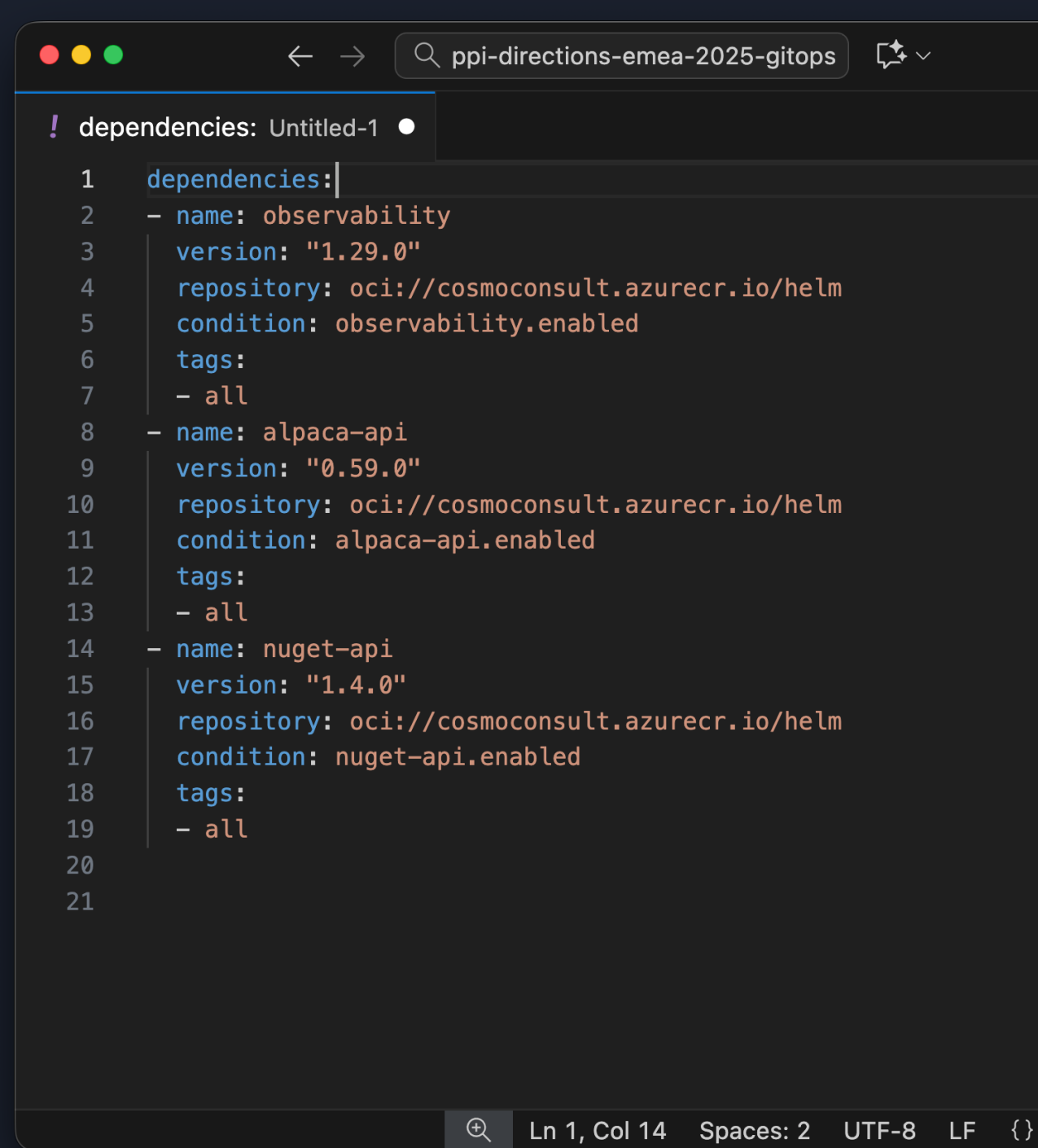
Node Size, Anzahl der Nodes, Kubernetes-
Version usw.

```
! cluster: Untitled-1 ●
1  cluster:
2    location: westeuropa
3    clusterKubernetesVersion: "1.32.4"
4
5    systemNodePool:
6      kubernetesVersion: "1.32.4"
7      size: Standard_D8s_v5
8
9    linuxNodePools:
10   - name: convert
11     kubernetesVersion: "1.32.4"
12     size: Standard_D16ds_v5
13     enableAutoScaling: true
14     maxCount: 3
15     nodeLabels:
16       ppi.cosmoconsult.com/node-type: convert
17     nodeTaints:
18       - "ppi.cosmoconsult.com/node-type=convert:NoSchedule"
19
20   windowsNodePools:
21     - name: win1
22       kubernetesVersion: "1.32.4"
23       enableAutoScaling: true
24       maxCount: 38
25       nodeLabels:
26         ppi.cosmoconsult.com/node-type: bcautoshtutdown
```



Ein Beispiel für GitOps aus der Praxis Apps

Jeder Cluster verfügt über ein eigenes Git-Repository, in dem festgelegt ist, welche Version der jeweiligen Anwendung auf dem jeweiligen Cluster bereitgestellt werden soll



```
! dependencies: Untitled-1 ●
1  dependencies:|
2  - name: observability
3    version: "1.29.0"
4    repository: oci://cosmoconsult.azurecr.io/helm
5    condition: observability.enabled
6    tags:
7      - all
8  - name: alpaca-api
9    version: "0.59.0"
10   repository: oci://cosmoconsult.azurecr.io/helm
11   condition: alpaca-api.enabled
12   tags:
13     - all
14  - name: nuget-api
15    version: "1.4.0"
16    repository: oci://cosmoconsult.azurecr.io/helm
17    condition: nuget-api.enabled
18    tags:
19      - all
20
21
```

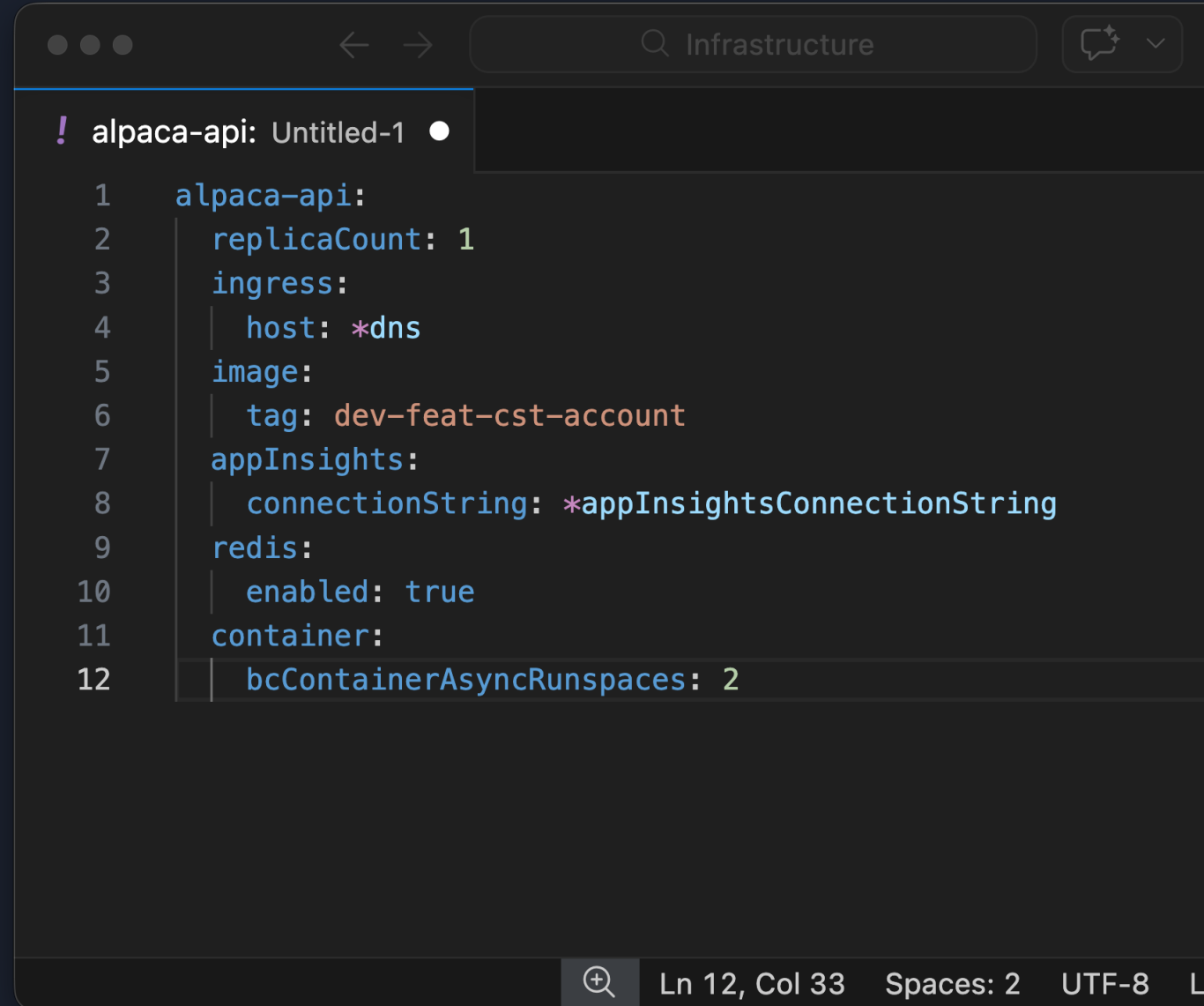
Ln 1, Col 14 Spaces: 2 UTF-8 LF {}



Ein Beispiel für GitOps aus der Praxis App-Konfiguration

In Git werden nicht nur App-Versionen, sondern auch Überschreibungen der App-Konfiguration definiert:

- Weniger Replikate im Testcluster
- Testen von Entwicklungsstadien in bestimmten Container Images
- Unterschiedliche Domänen/Connection Strings usw.



```
! alpaca-api: Untitled-1 ●
1  alpaca-api:
2    replicaCount: 1
3    ingress:
4      host: *dns
5    image:
6      tag: dev-feat-cst-account
7    appInsights:
8      connectionString: *appInsightsConnectionString
9    redis:
10     enabled: true
11    container:
12     bcContainerAsyncRunspaces: 2
```

The screenshot shows a code editor window titled 'alpaca-api: Untitled-1'. The editor displays a YAML configuration for an application named 'alpaca-api'. The configuration includes settings for replicaCount, ingress, image, appInsights, redis, and container. The status bar at the bottom indicates the cursor is at line 12, column 33, with 2 spaces, UTF-8 encoding, and a line length of L.



Ein Beispiel für GitOps aus der Praxis

Konfigurationen und Secrets

Separate GitOps-Repositorys für
Konfigurationen und verschlüsselte Secrets
werden in mehrere Cluster synchronisiert



The screenshot shows a GitHub repository page for 'alpac-configs-cosmo/licenses.yml'. The browser address bar shows the URL 'https://github.com/cosmoconsult/alpac-configs-cosmo/blob/main/licenses.yml'. The repository name 'alpac-configs-cosmo' is highlighted in the top navigation bar. The file 'licenses.yml' is selected in the file browser on the left. The main content area shows the code for 'licenses.yml' with a commit message 'update configs' by user 'lippertmarkus'. The code is a YAML file defining a spec with a template, data, metadata, annotations, and encrypted data. The metadata includes 'name: licenses' and 'creationTimestamp'. The annotations include 'sealedsecrets.bitnami.com/cluster'. The encrypted data is a list of secrets for various environments and clusters.

```
1 spec:
2   template:
3     data:
4     metadata:
5       name: licenses
6       creationTimestamp:
7     annotations:
8       sealedsecrets.bitnami.com/cluster:
9     encryptedData:
10    dbcllic-bc26-de-at: AgDMRTH70KjQcpBrMz
11    devlic-bc19-at: AgBHRBmHQDLidjA9ARsi1
12    devlic-bc19-ro-w1: AgC+9BcwrrIJ6H124M
13    dbcllic-bc18-se-de: AgC8ruikxda5Phc60L
14    dbcllic-bc24-ro-w1: AgDXBYLvQ0wnBRjLGF
15    dbcllic-bc-ro-w1: AgDXhnxwGXad06xxLXBF
16    dbcllic-bc-at: AgB6Ugb3XwHq8ieUKwsJIffj
17    dbcllic-bc-se: AgBnPUM51/eLkfro5hnLnjd
18    dbcllic-bc-de: AgA7rbrbox3GTx0+cQk5PJk
19    dbcllic-bc-fr: AgDNlovskLn3Jih/8fZSpnZ
20    devlic-bc-ro-w1: AgC4/Ttffmly6B+8T0Cw...
```

Ein Beispiel für GitOps aus der Praxis

Automatisierte Rollouts über PRs

Nach einer neuen App-Veröffentlichung werden automatisch PRs in den GitOps-Repositorys erstellt.

Der Rollout startet automatisch nach der Genehmigung eines PR.

The screenshot shows a pull request titled "chore: Update observability 1.29.0" in the Azure DevOps interface. The pull request is completed and was created by the "Infrastructure Build Service (cc-ppi)". The diff shows a change in the "Chart.yaml" file, where the version of the "observability" chart is updated from "1.28.0" to "1.29.0".

```
-----  
24     tags:  
25     - all  
26     - name: observability  
27     + version: "1.29.0"  
28     repository: oci://cosmoconsult.azurecr.io/helm  
29     condition: observability.enabled  
30     tags:  
-----
```



Ein Beispiel für GitOps aus der Praxis

Zusammenfassung

- Bei Alpaca speichern wir alle Definitionen in Git: Infrastruktur, Apps, App-Konfigurationen, Konfigurationsdateien und verschlüsselte Secrets
- Das Ausrollen von Änderungen durch mehrere Cluster („Stages“) und automatisierte Rollouts erfolgt durch die Genehmigung automatisch erstellter PRs
- Manuelle „große“ Änderungen wie das Ändern der Infrastruktur, das Hinzufügen neuer Apps oder das Ändern von Konfigurationen werden über PRs geprüft
- Der GitOps-Ansatz hat uns mehrfach dabei geholfen, fehlerhafte Änderungen schnell rückgängig zu machen



Best Practices & Lessons Learned

Wann sollte man GitOps einsetzen?

- **GitOps eignet sich gut für Cloud-native Szenarien**
 - Verwaltung von Kubernetes-Clustern und deren Anwendungen, Netzwerk, Speicher usw.
 - Infrastrukturmanagement über mehrere Cloud-Plattformen hinweg aus einer Hand
 - Pflege mehrerer ähnlicher Umgebungen (z. B. Entwicklung, Staging und Produktion)
 - Beispiel: ein Branch für jede Stage, Rollout von Änderungen durch jede Phase mittels einfacher Pull-Requests
- **Was vor der Nutzung von GitOps zu beachten ist**
 - Die Einrichtung guter GitOps-Tools, -Strukturen, deklarativer Definitionen und -Prozesse nimmt Zeit in Anspruch
 - Schnelle Experimente und die Ersteinrichtung sollten zunächst besser ohne GitOps durchgeführt und erst später in einer stabilen Phase integriert werden
 - Nicht alles, was man möglicherweise benötigt, ist deklarativ verfügbar



Best Practices & Lessons Learned

Zusammenfassung

- GitOps-Tools wie Argo CD: Stellen sicher, dass die Realität dem gewünschten Git-Zustand entspricht
 - Crossplane-Integration: sichere, konsistente, überprüfbare und selbstheilende Infrastruktur
 - Git-basierte Workflows: Commits, Pull-Requests und Reviews deklarativer Konfigurationen – wie bei Software
 - Ermöglicht die Verwendung von Copilot zum Schreiben von Deklarationen und die Nutzung von Tools wie GitHub Copilot Code Review zur Überprüfung
- Fang klein an: Pilotprojekt, minimale Repo-Strukturen und einfache Prozesse
- Schrittweise wachsen





Questions?



Thank you very much!
We look forward to your feedback!

**Scan
me!**

