GitOps – Infrastructure as Code, aber richtig

ppedv

Markus Lippert

- Business

  Team & Technical Lead
  bei COSMO CONSULT

- Social und Blog

  lippertmarkus bei LinkedIn

  Blog und BlueSky Account unter
  lippertmarkus.com

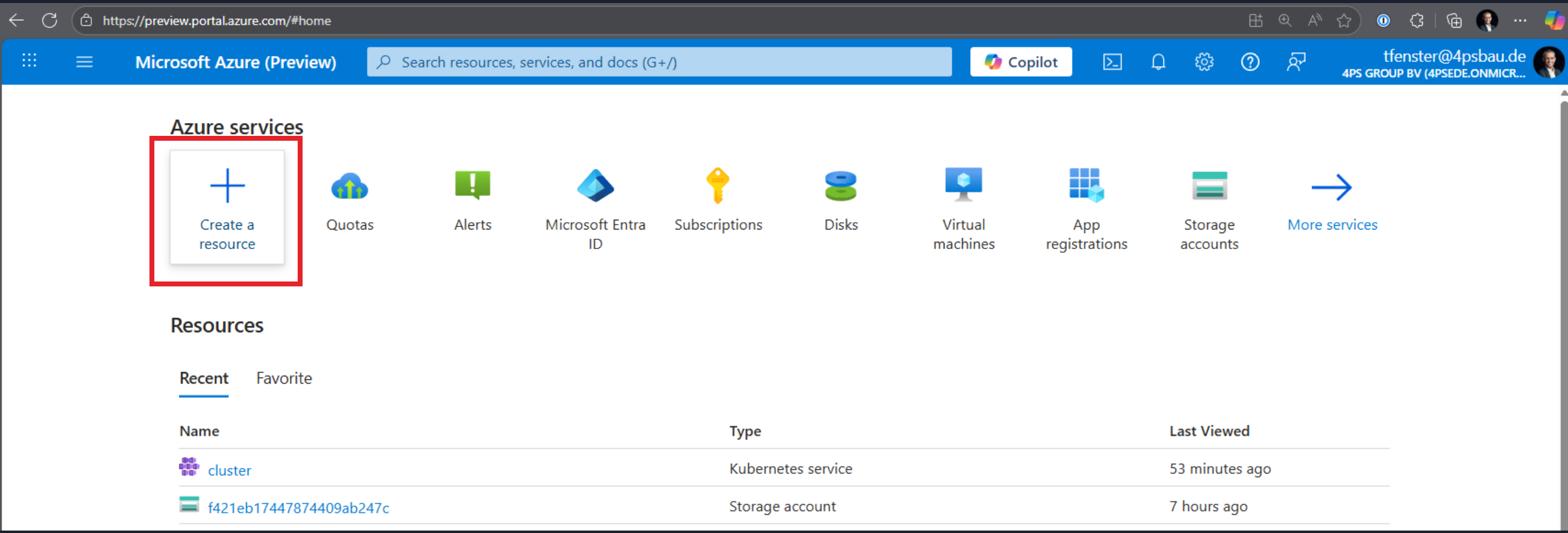
Tobias Fenster

- 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

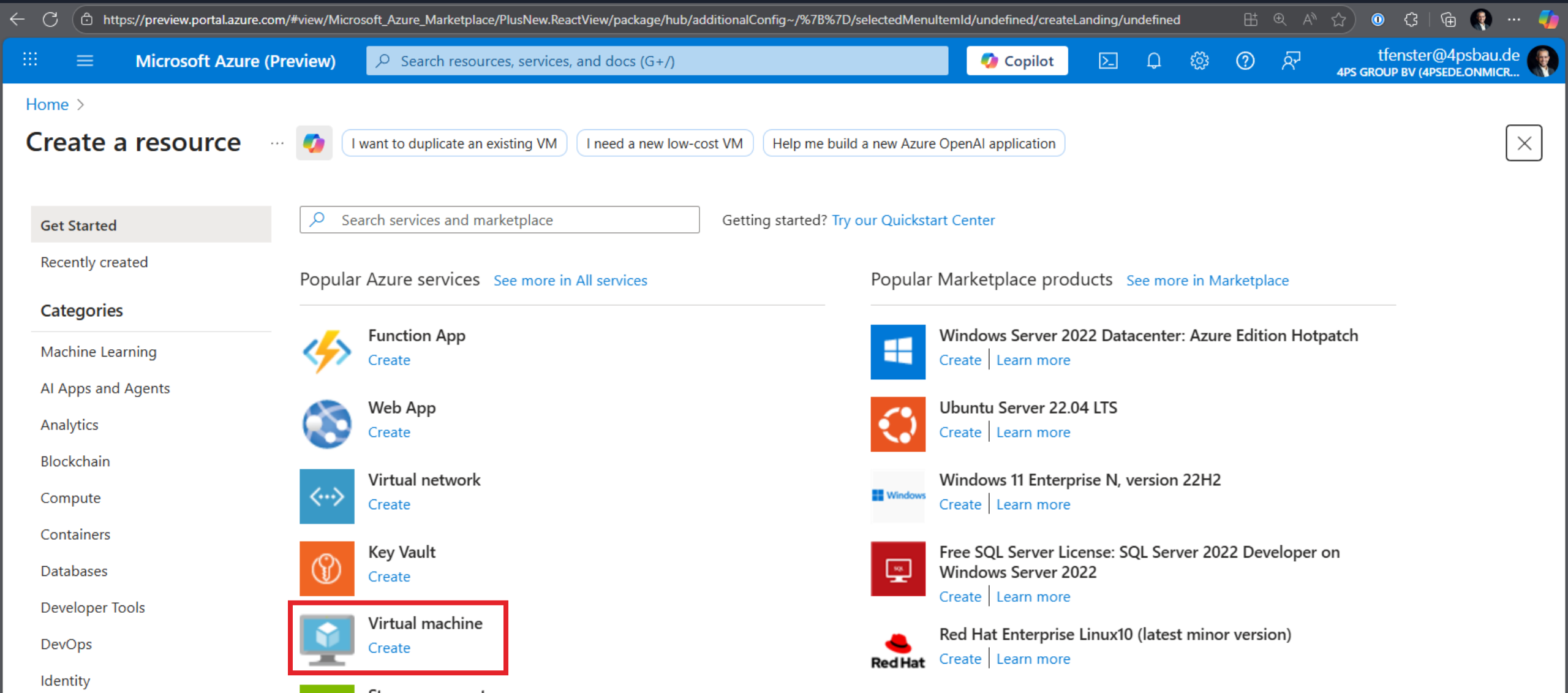
Agenda

- 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

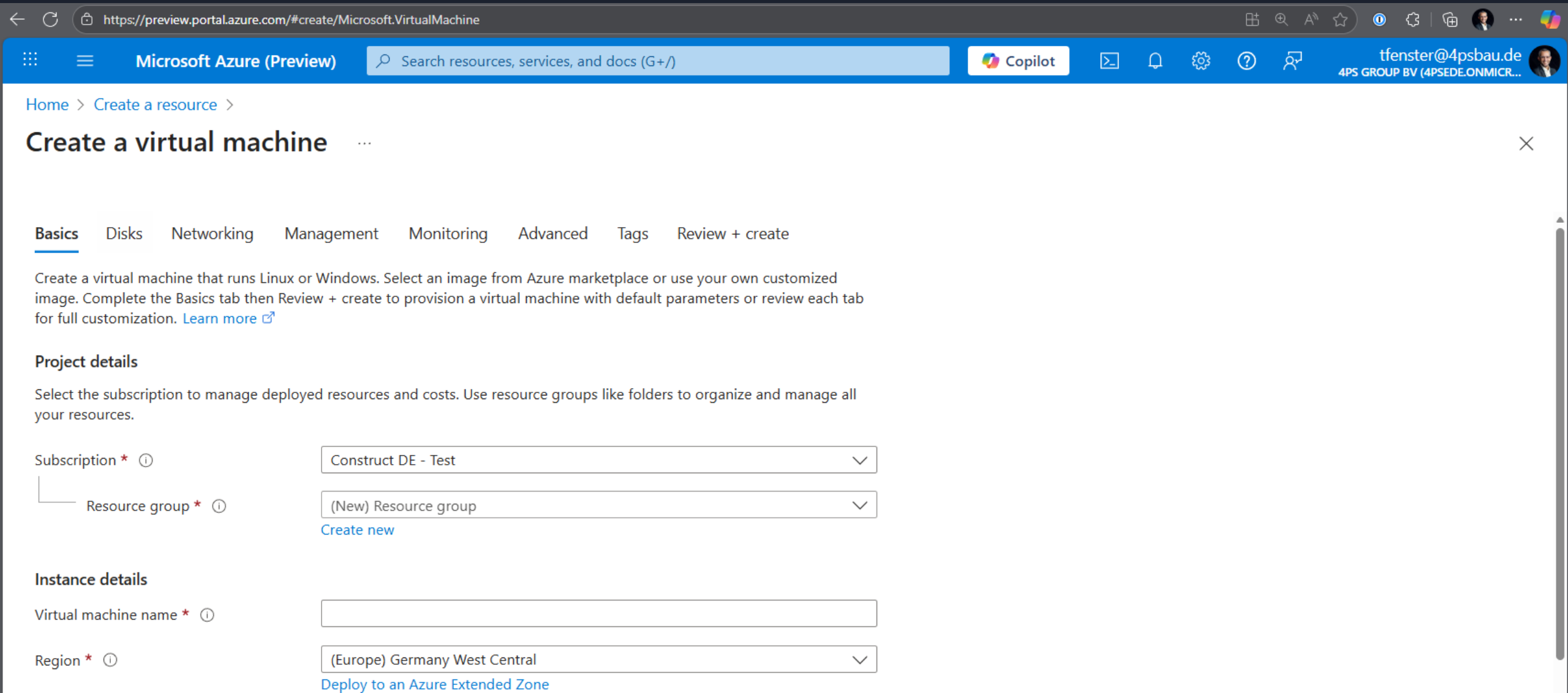
Machst du noch ClickOps? Oder schon IaC?



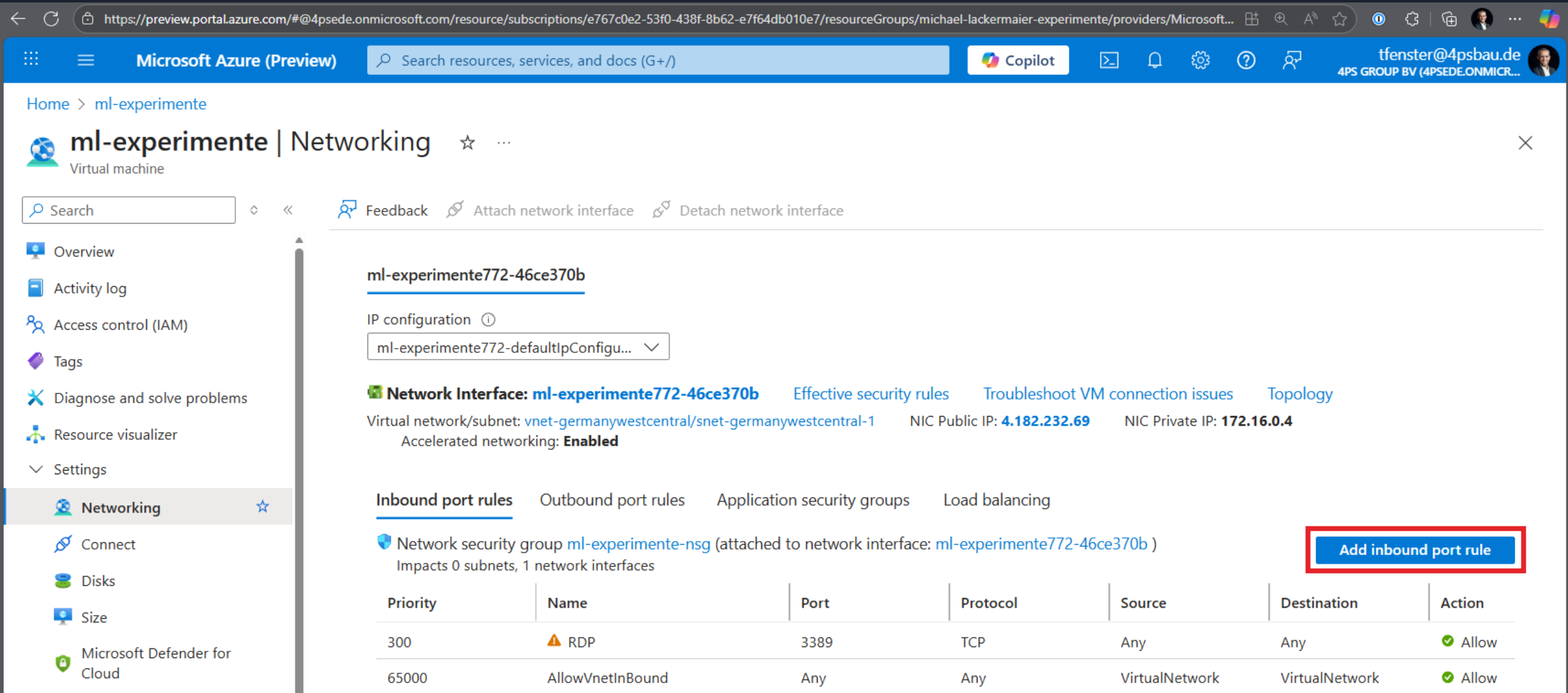
Machst du noch ClickOps? Oder schon IaC?



Machst du noch ClickOps? Oder schon IaC?



Machst du noch ClickOps? Oder schon IaC?



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_D8alds_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)

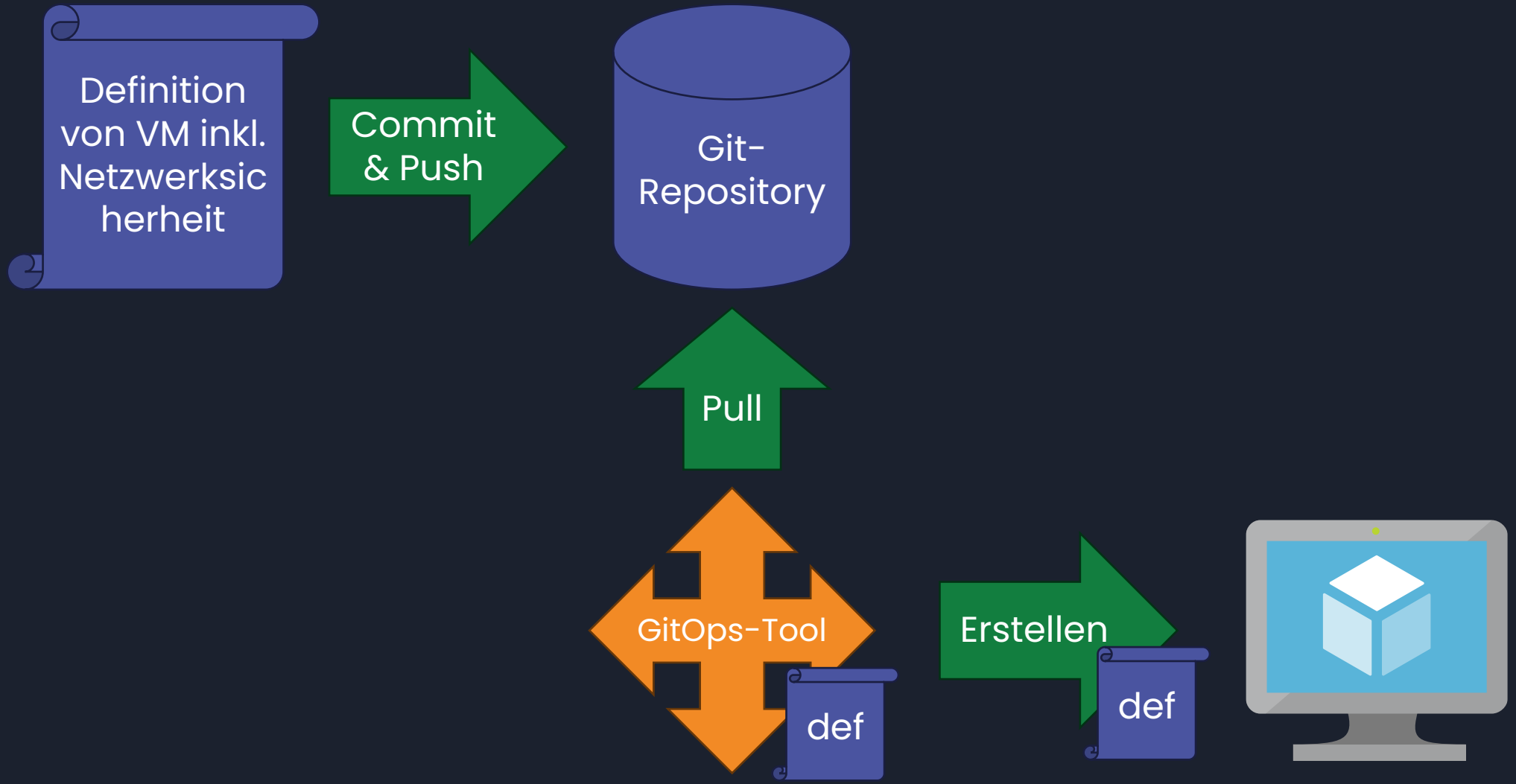
- Versionierung und Historie (der Definition) sind jetzt möglich
- Aber:
  - Manuelle Eingriffe oder unbeabsichtigte automatisierte Änderungen können trotzdem erfolgen („Drift“)
  - Wird nicht kontinuierlich aktualisiert (sofern das IaC-Tool dies überhaupt kann), sondern nur auf Anforderung (Push)
  - Eingeschränkte Nachverfolgbarkeit

- → Ein großer Schritt in die richtige Richtung, aber auch noch große Probleme

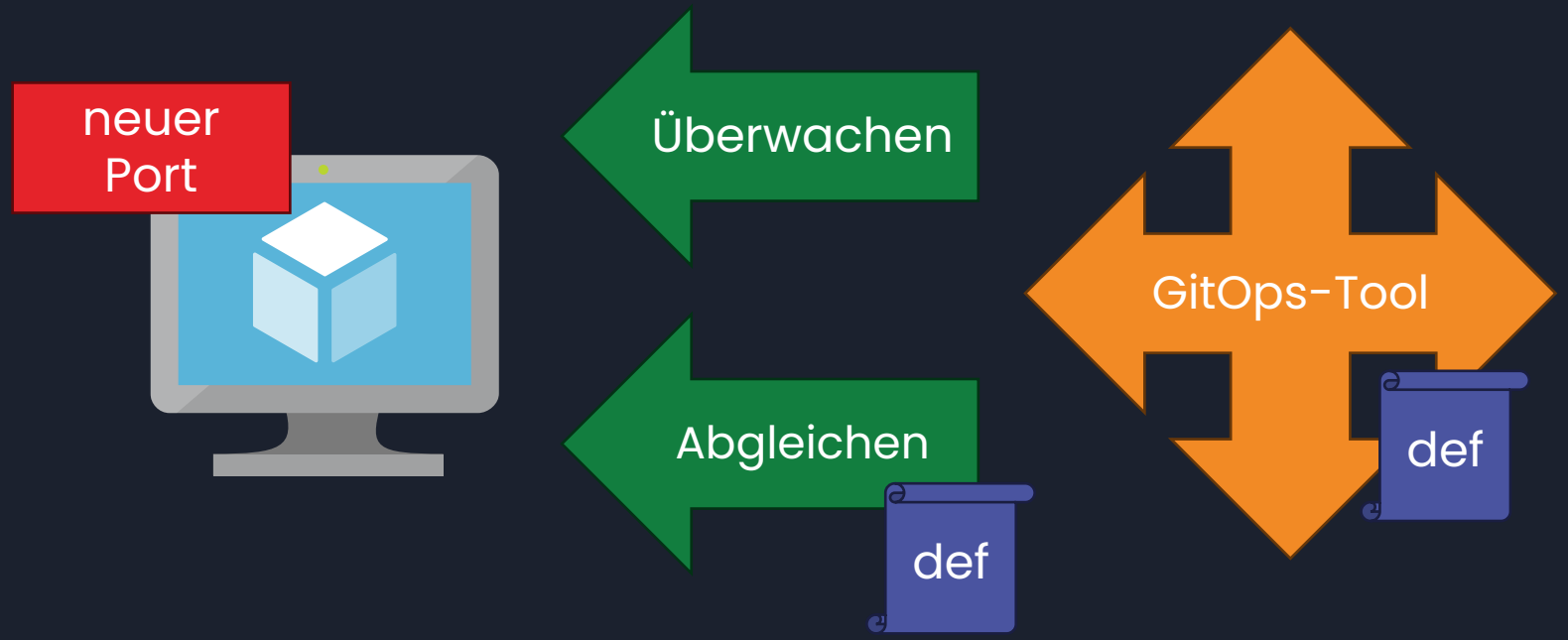
Was ist GitOps und warum ist es hilfreich?
Grundlegende Konzepte

- 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



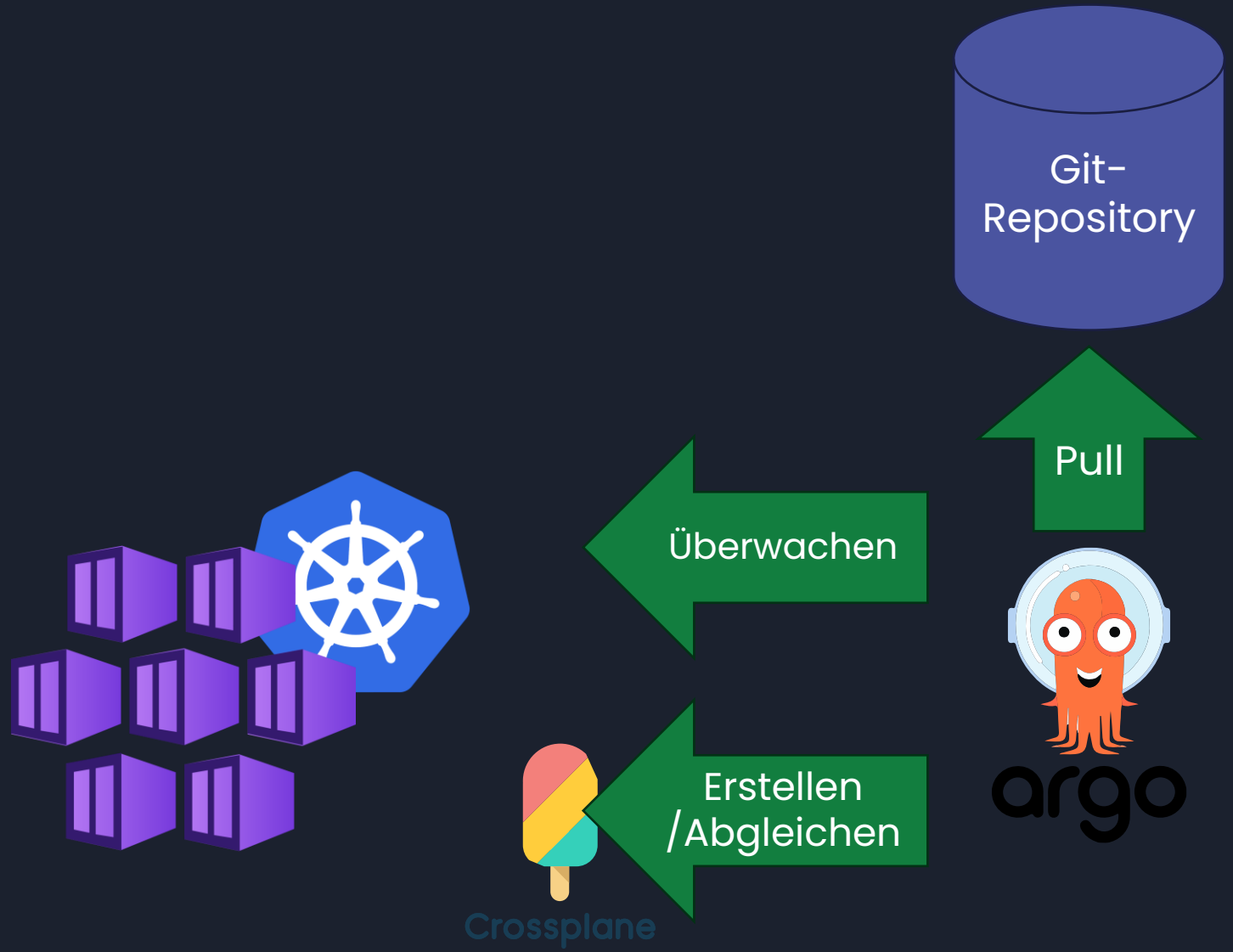
Der Tech-Stack im Überblick

- 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

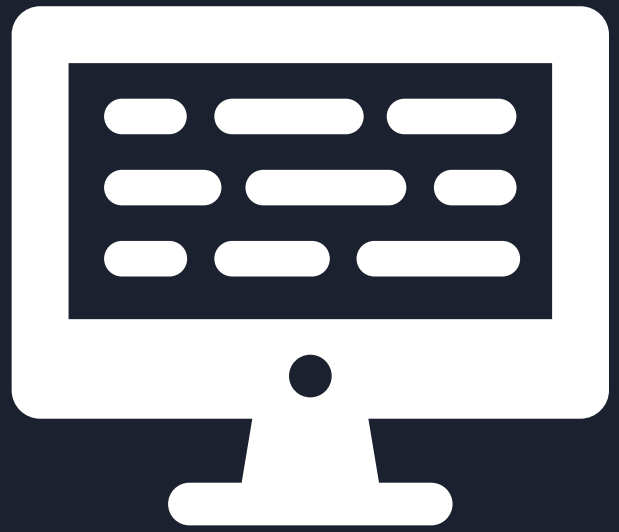
Der Tech-Stack im Überblick



Deep-Dive und Live-Demo

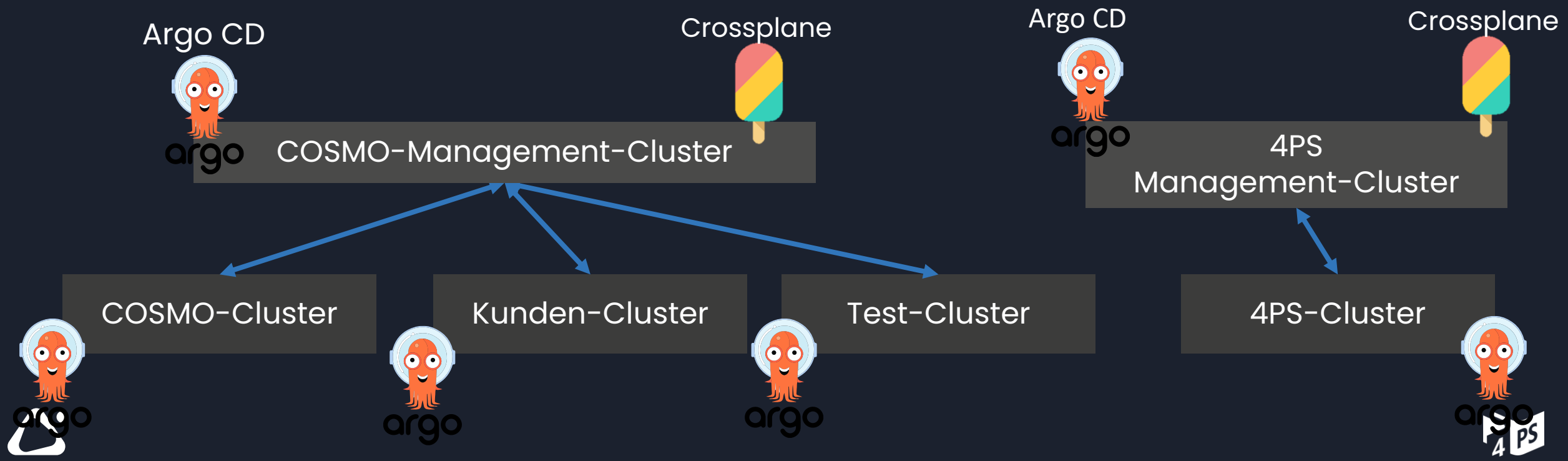
Wir schauen uns an:

- 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

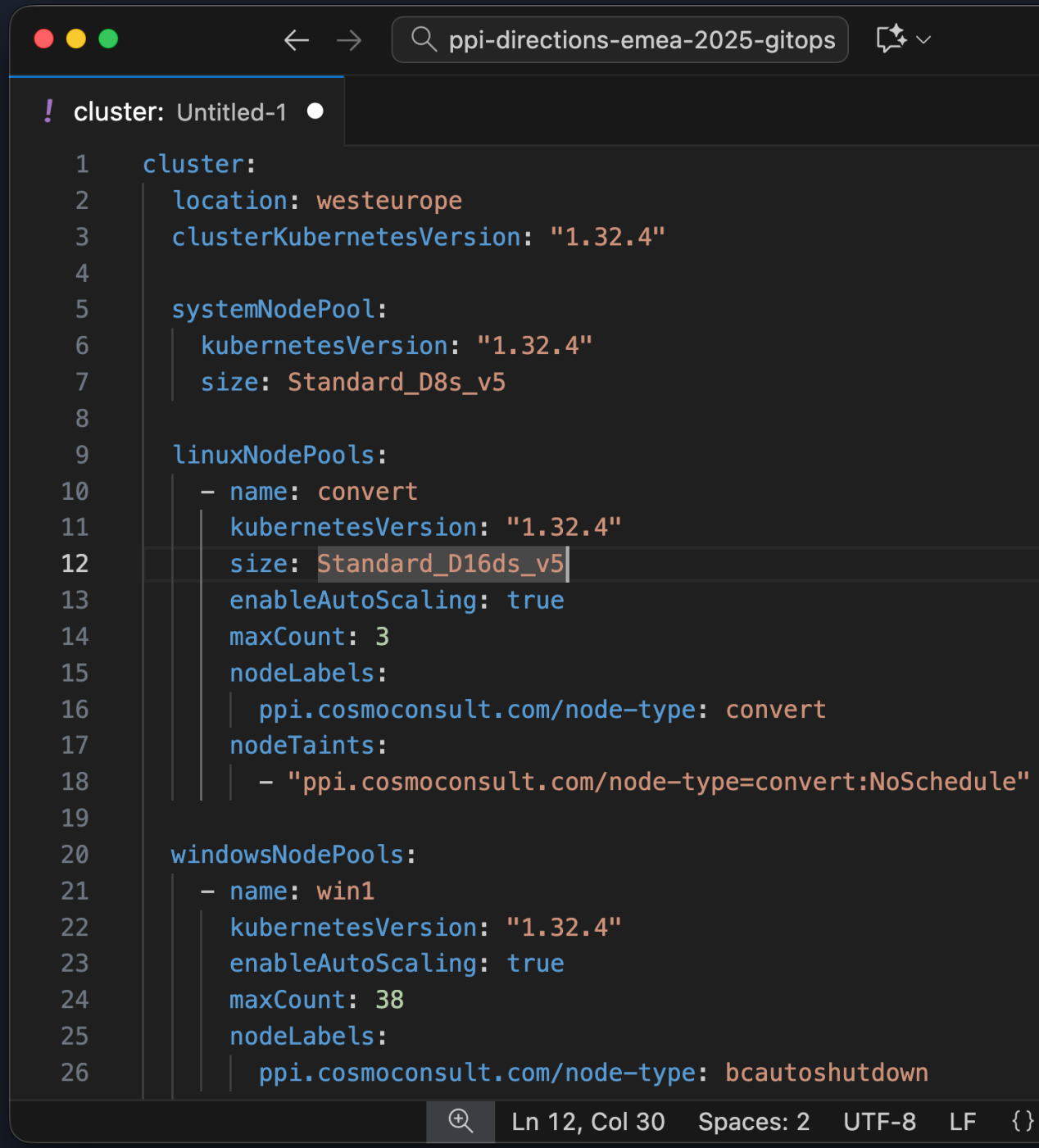
- COSMO Alpaca nutzt den Azure Kubernetes Service (AKS) zum Hosten verschiedener C#-Web-APIs und Windows-Anwendungscontainer
- 4 Cluster werden von COSMO über GitOps mit Argo CD und Crossplane verwaltet, 2 Cluster von 4PS
- Test-/Kunden-/COSMO-Cluster sind ähnlich; Änderungen werden über PRs durch diese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



Ein Beispiel für GitOps aus der Praxis
Infrastruktur

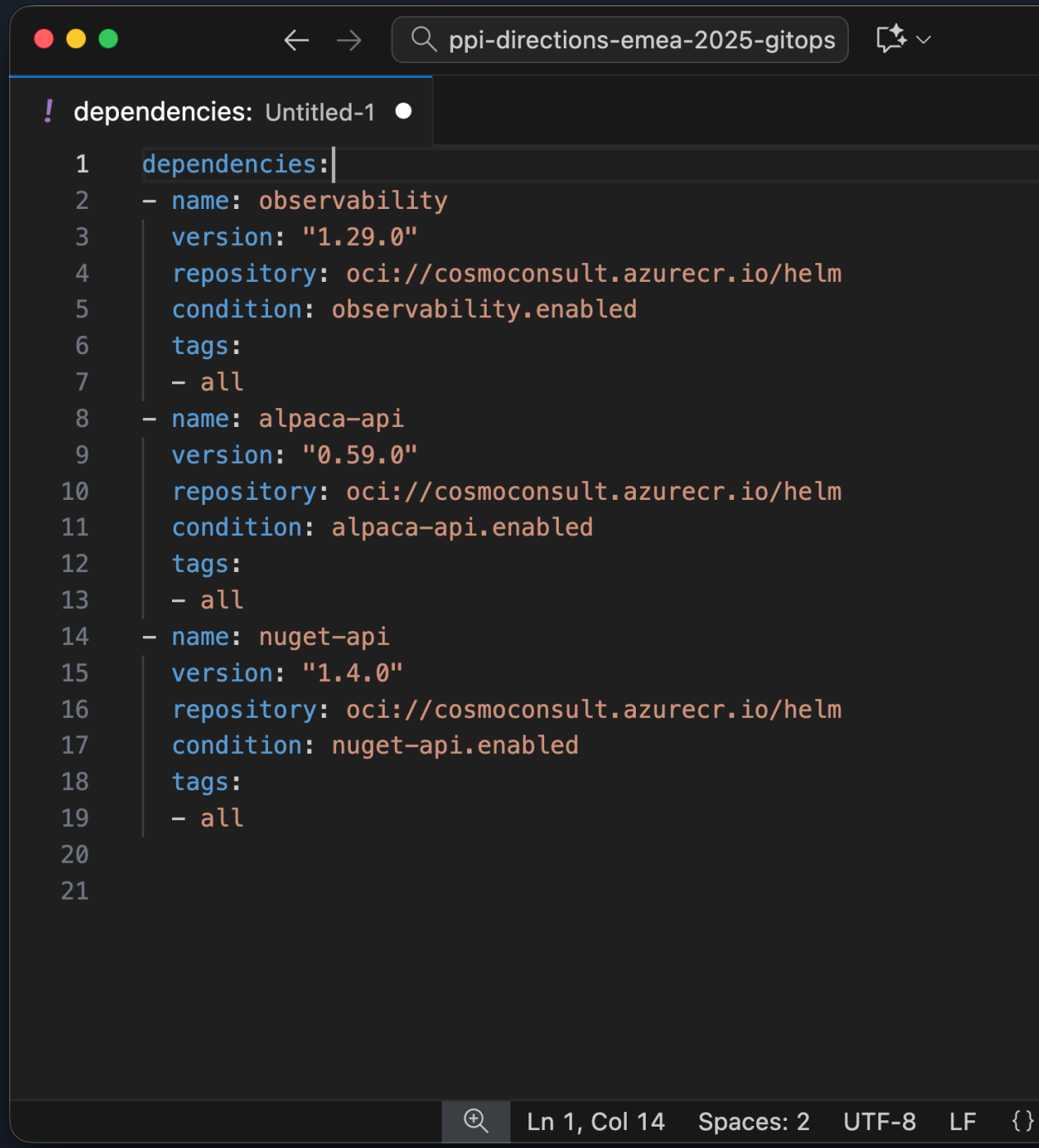
AKS-Cluster werden deklarativ definiert:

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



Ein Beispiel für GitOps aus der Praxis
Apps

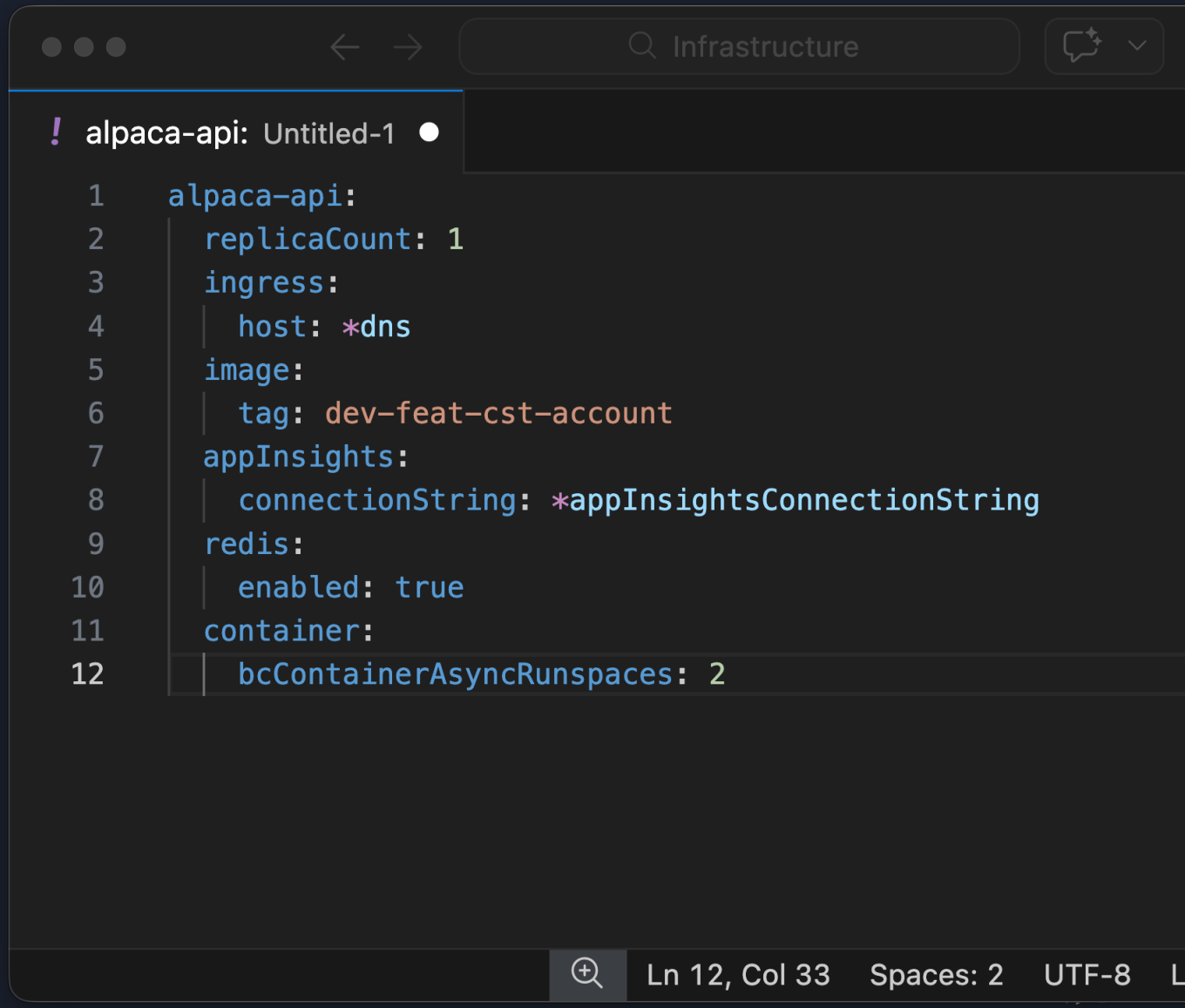
Jeder Cluster verfügt über ein eigenes Git-Repository, in dem festgelegt ist, welche Version der jeweiligen Anwendung auf dem jeweiligen Cluster bereitgestellt werden soll



Ein Beispiel für GitOps aus der Praxis
App-Konfiguration

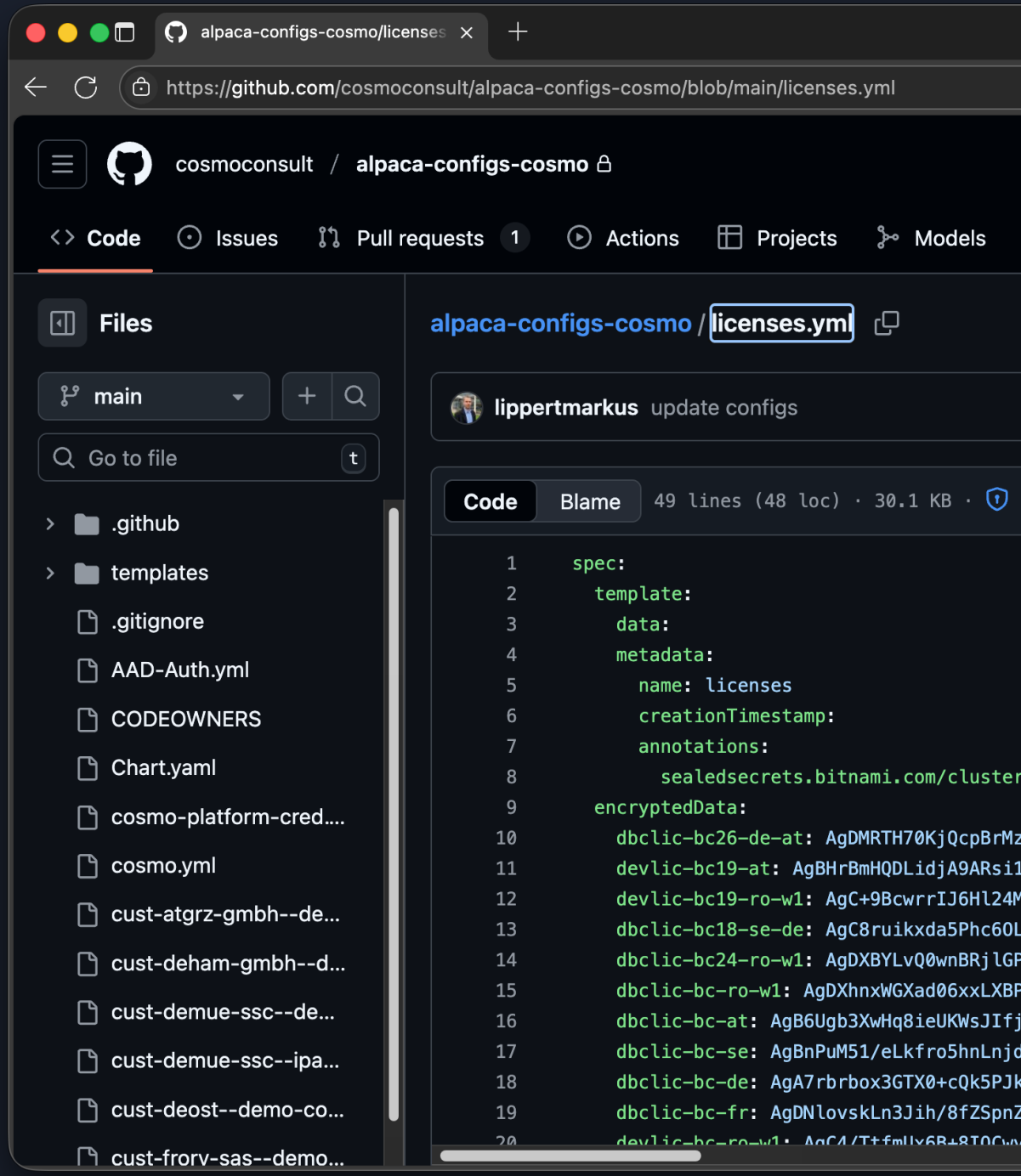
In Git werden nicht nur App-Versionen, sondern auch Überschreibungen der App-Konfiguration definiert:

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



Ein Beispiel für GitOps aus der Praxis
Konfigurationen und Secrets

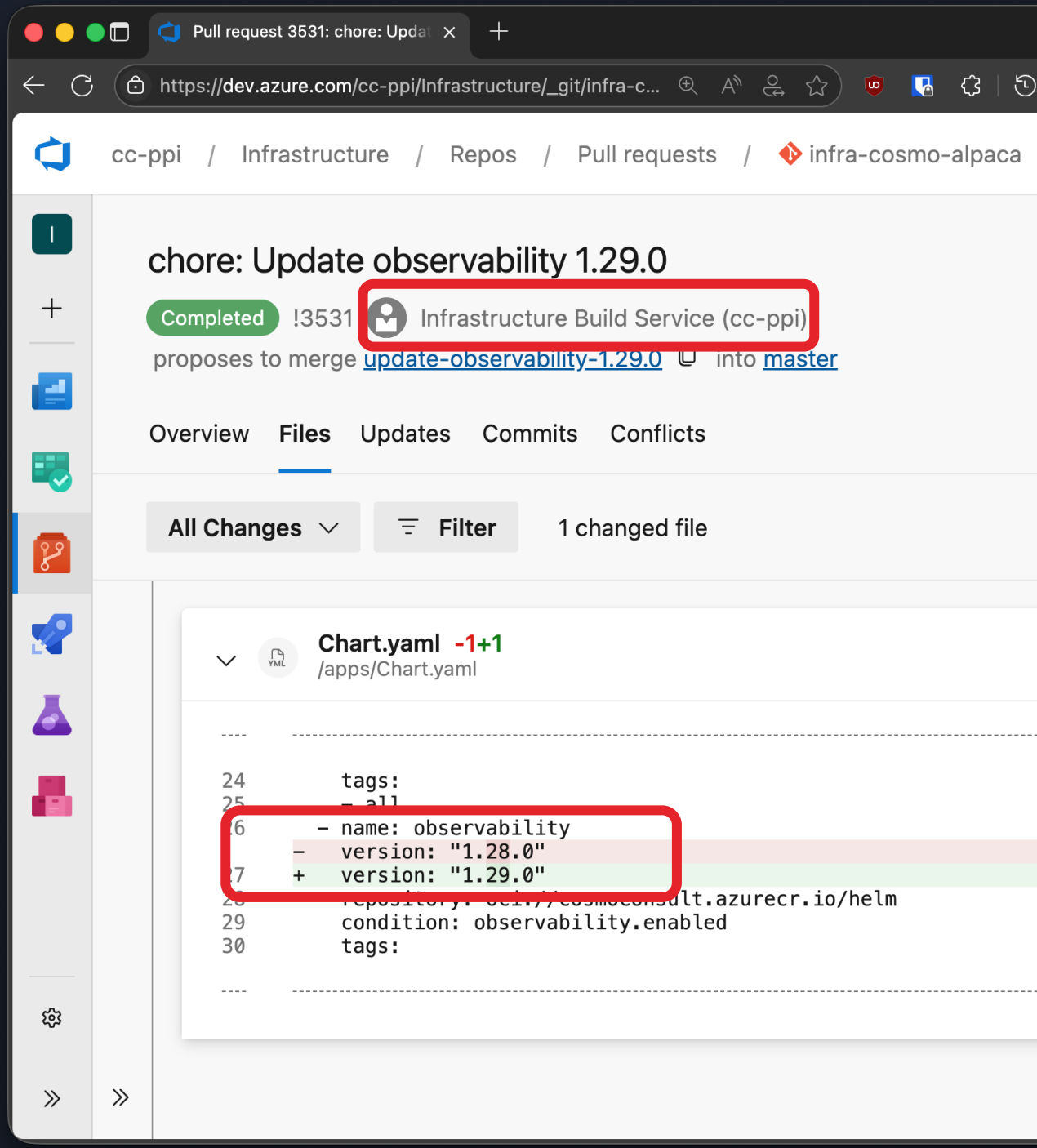
Separate GitOps-Repositorys für Konfigurationen und verschlüsselte Secrets werden in mehrere Cluster synchronisiert



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.



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!

ppedv