Building an Internal Developer Platform on Azure

Roberth Strand

Crayon Group | @roberthTweets

MVP Dagen

/whoami

Roberth Strand
Head of Platform Engineering, Crayon

Microsoft MVP
HashiCorp Ambassador

- Cloud Automator
- OSS & Cloud-Native advocate
  - CNCF
  - OpenGitOps
- Community contributor
  - Azure Cloud Native User Group
  - Norwegian PowerShell User Group
  - DevOpsDays Oslo co-organizer

Takk til våre sponsorer

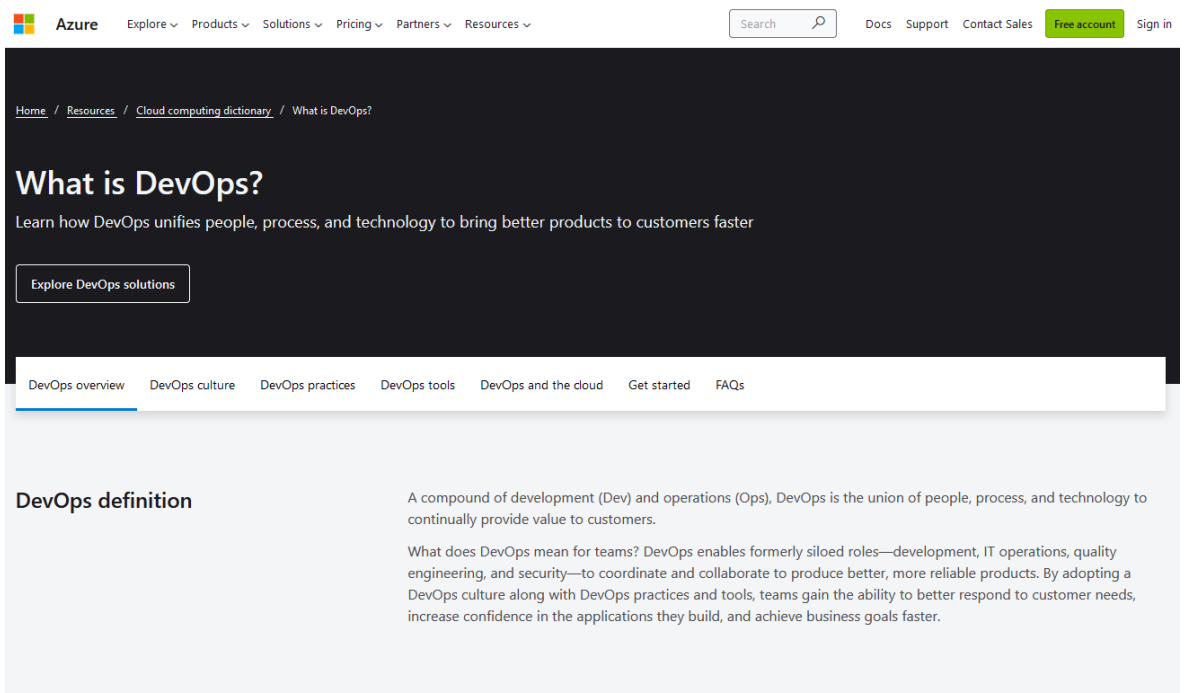
Topics

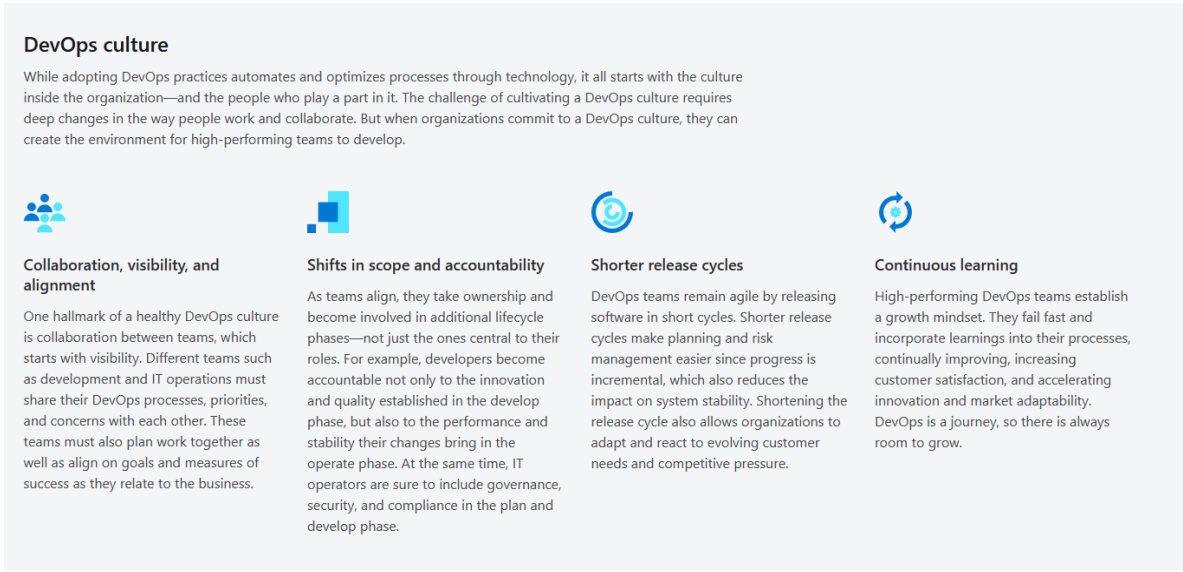
- DevOps
- Solving common IT Ops problems
- What is an IDP
- Strategies for implementing IDP

Let's talk about DevOps

A quick introduction, and what it means today

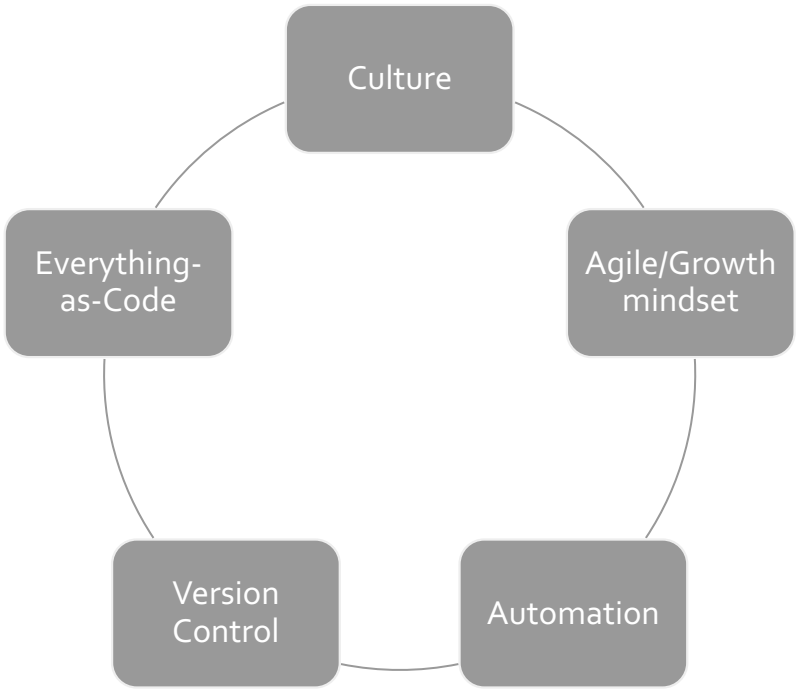
MVP-Dagen





Source: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-devops/

DevOps Principles & Practices



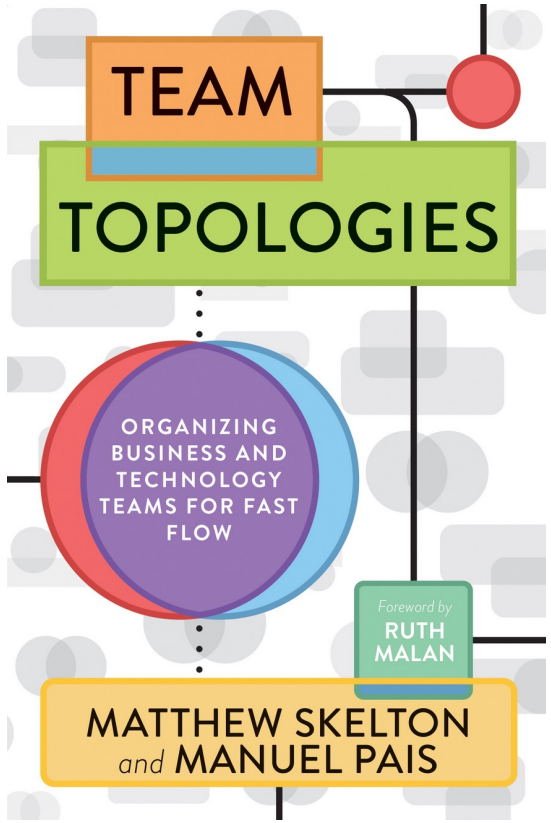
Team Topologies (2019)

Stream-aligned team

Enabling team

Complicated subsystem team

Platform team



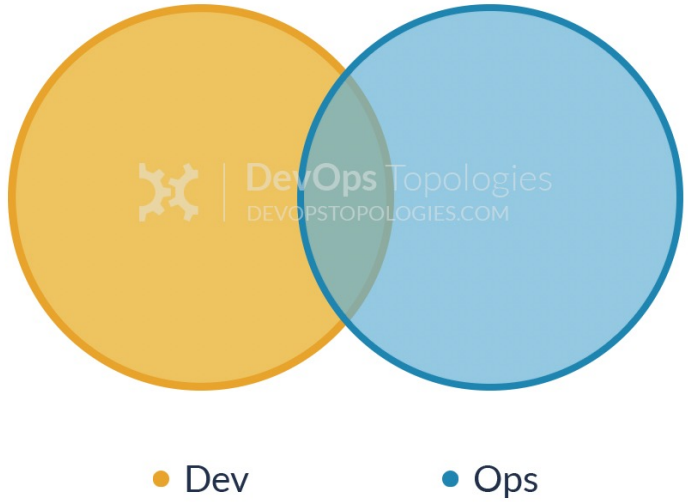
Visit teamtopologies.com

DevOps Topologies

Type 1: Dev and Ops Collaboration

This is the 'promised land' of DevOps: smooth collaboration between Dev teams and Ops teams, each specialising where needed, but also sharing where needed. There are likely many separate Dev teams, each working on a separate or semi-separate product stack.

My sense is that this Type 1 model needs quite substantial organisational change to establish it, and a good degree of competence higher up in the technical management team. Dev and Ops must have a clearly expressed and demonstrably effective shared goal ('Delivering Reliable, Frequent Changes', or whatever). Ops folks must be comfortable pairing with Devs and get to grips with test-driven coding and Git, and Devs must take operational features seriously and seek out Ops people for input into logging implementations, and so on, all of which needs quite a culture change from the recent past.



Type 1 suitability: an organisation with strong technical leadership.

Potential effectiveness: HIGH

Source: web.devopstopologies.com

DevOps Topologies

● Dev ● Ops

Type 2 suitability: organisations with a single main web-based product or service.

Potential effectiveness: HIGH

Source: web.devopstopologies.com

Type 2: Fully Shared Ops Responsibilities

Where operations people have been integrated in product development teams, we see a Type 2 topology. There is so little separation between Dev and Ops that all people are highly focused on a shared purpose; this is arguable a form of **Type 1 (Dev and Ops Collaboration)**, but it has some special features.

Organisations such as Netflix and Facebook with effectively a single web-based product have achieved this Type 2 topology, but I think it's probably not hugely applicable outside a narrow product focus, because the budgetary constraints and context-switching typically present in an organisation with multiple product streams will probably force Dev and Ops further apart (say, back to a **Type 1 model**). This topology might also be called 'NoOps', as there is no distinct or visible Operations team (although the Netflix NoOps might also be **Type 3 (Ops as IaaS)**).

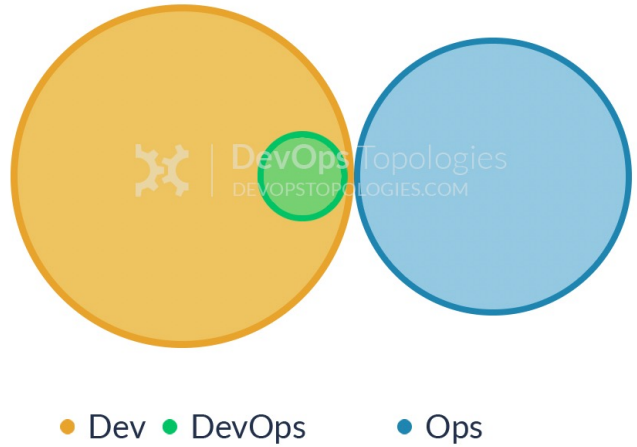
DevOps Topologies

Type 3: Ops as Infrastructure-as-a-Service (Platform)

For organisations with a fairly traditional IT Operations department which cannot or will not change rapidly [enough], and for organisations who run all their applications in the public cloud (Amazon EC2, Rackspace, Azure, etc.), it probably helps to treat Operations as a team who simply provides the elastic infrastructure on which applications are deployed and run; the internal Ops team is thus directly equivalent to Amazon EC2, or Infrastructure-as-a-Service.

A team (perhaps a virtual team) within Dev then acts as a source of expertise about operational features, metrics, monitoring, server provisioning, etc., and probably does most of the communication with the IaaS team. This team is still a Dev team, however, following standard practices like TDD, CI, iterative development, coaching, etc.

The IaaS topology trades some potential effectiveness (losing direct collaboration with Ops people) for easier implementation, possibly deriving value more quickly than by trying for **Type 1 (Dev and Ops Collaboration)** which could be attempted at a later date.



Type 3 suitability: organisations with several different products and services, with a traditional Ops department, or whose applications run entirely in the public cloud.

Potential effectiveness: MEDIUM

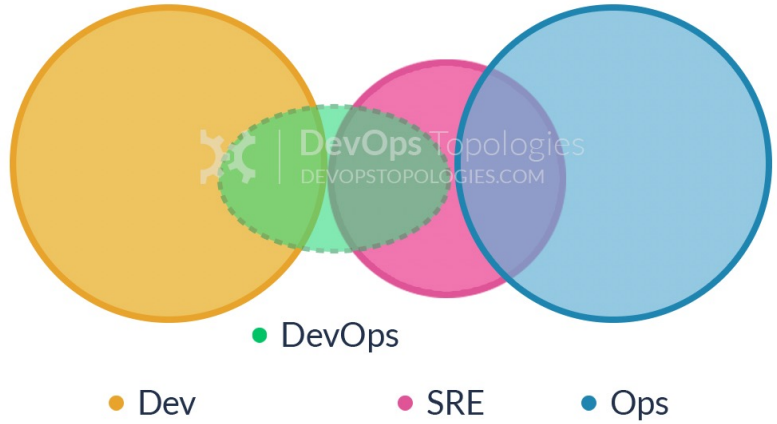
Source: web.devopstopologies.com

DevOps Topologies

Type 7: SRE Team (Google Model)

DevOps often recommends that Dev teams join the on-call rotation, but it's not essential. In fact, some organisations (including Google) run a different model, with an explicit 'hand-off' from Development to the team that runs the software, the Site Reliability Engineering (SRE) team. In this model, the Dev teams need to provide test evidence (logs, metrics, etc.) to the SRE team showing that their software is of a good enough standard to be supported by the SRE team.

Crucially, the SRE team can reject software that is operationally substandard, asking the Developers to improve the code before it is put into Production. Collaboration between Dev and SRE happens around operational criteria but once the SRE team is happy with the code, they (and not the Dev team) support it in Production.



Thanks to Kevin Hinde **(@kwdhinde)**

Type 7 suitability: Type 7 is suitable only for organisations with a high degree of engineering and organisational maturity. Beware of a return to **Anti-Type A** if the SRE/Ops team is told to "JFDI" deploy.

Potential effectiveness: LOW to HIGH

Source: web.devopstopologies.com

Solving modern IT Ops problems

Secure by default

Distributed ownership

Flexibility

Collaboration

Internal Developer Platform

Internal Developer Platform strategies

Build your own platform

- Automation layer
- Self-service layer

Use existing solutions

1. Backstage
2. Humanitec

Use the same operational model between Dev and Ops

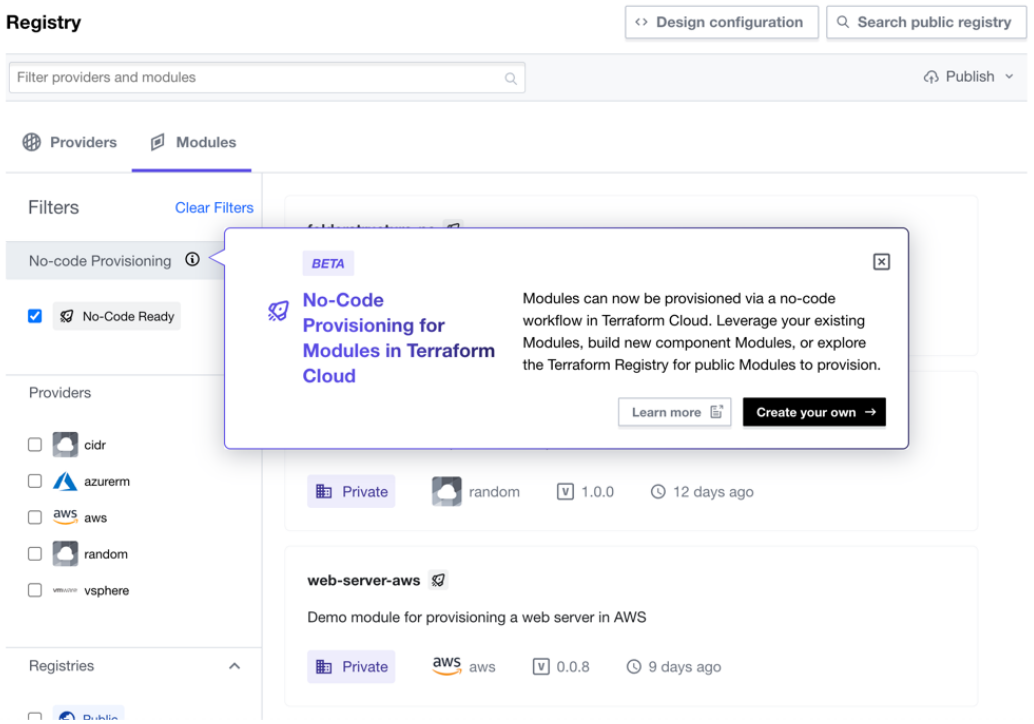
1. GitOps
2. Infrastructure-as-Code

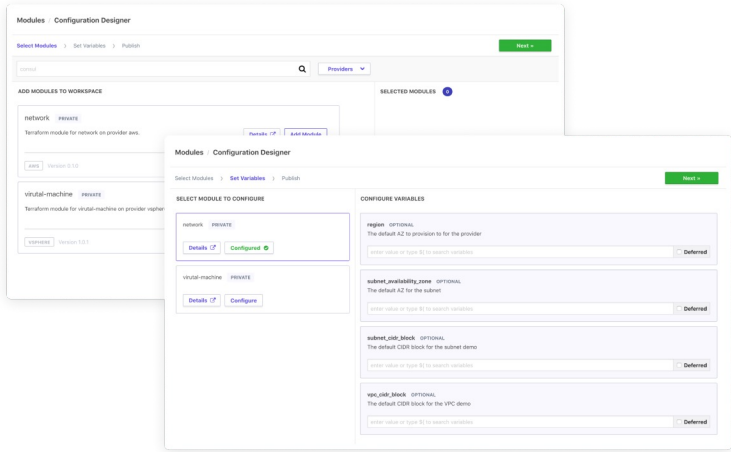
Terraform as shared deployment mechanic

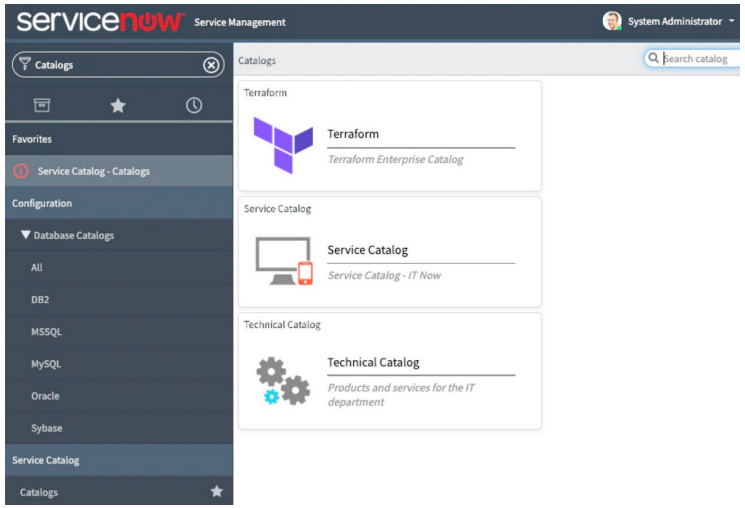
Source: https://www.hashicorp.com/products/terraform/self-service-infrastructure

Terraform Cloud self-service

NEW

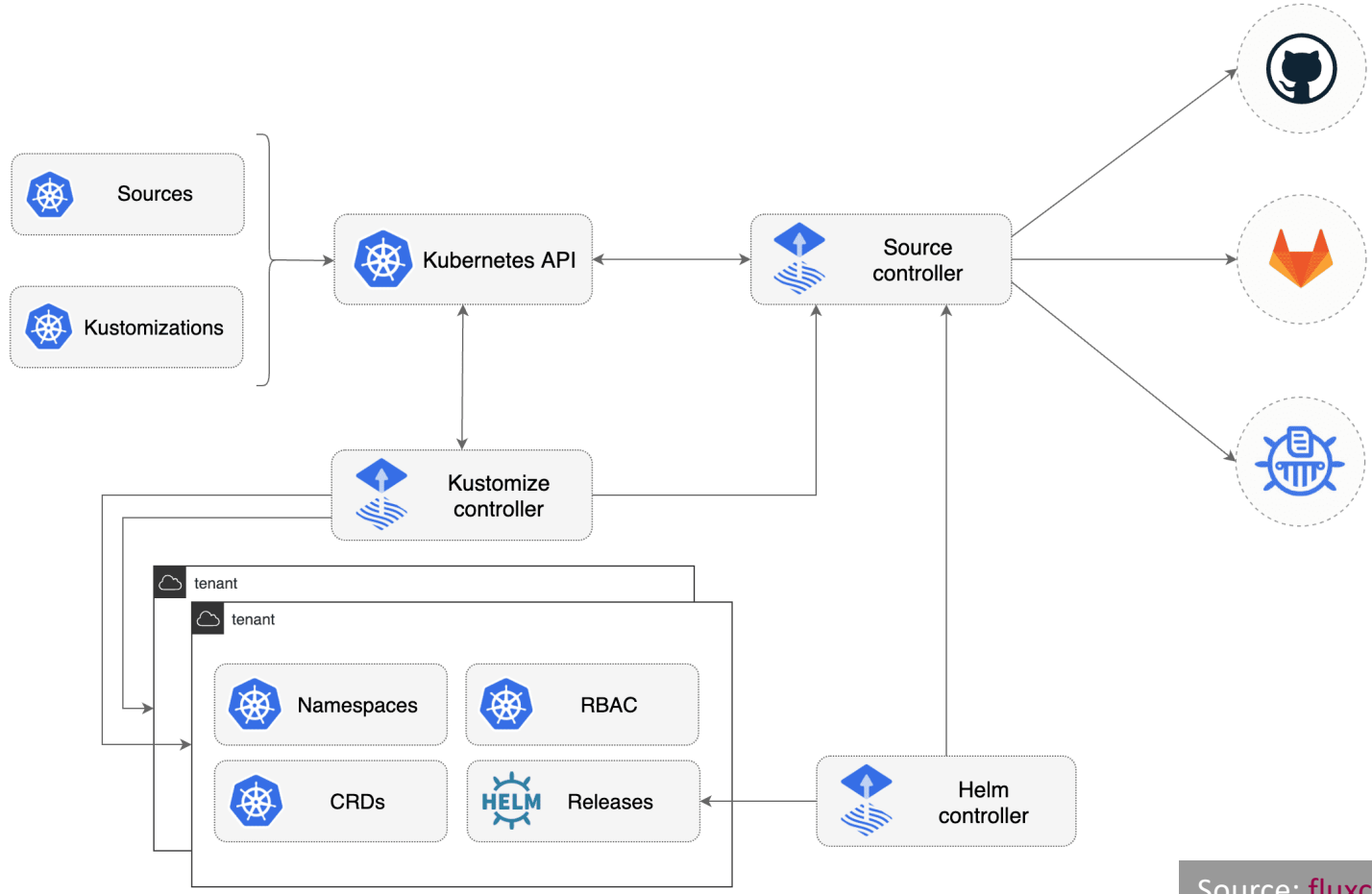






Source: https://www.hashicorp.com/

GitOps



Source: fluxcd.io

GitOps

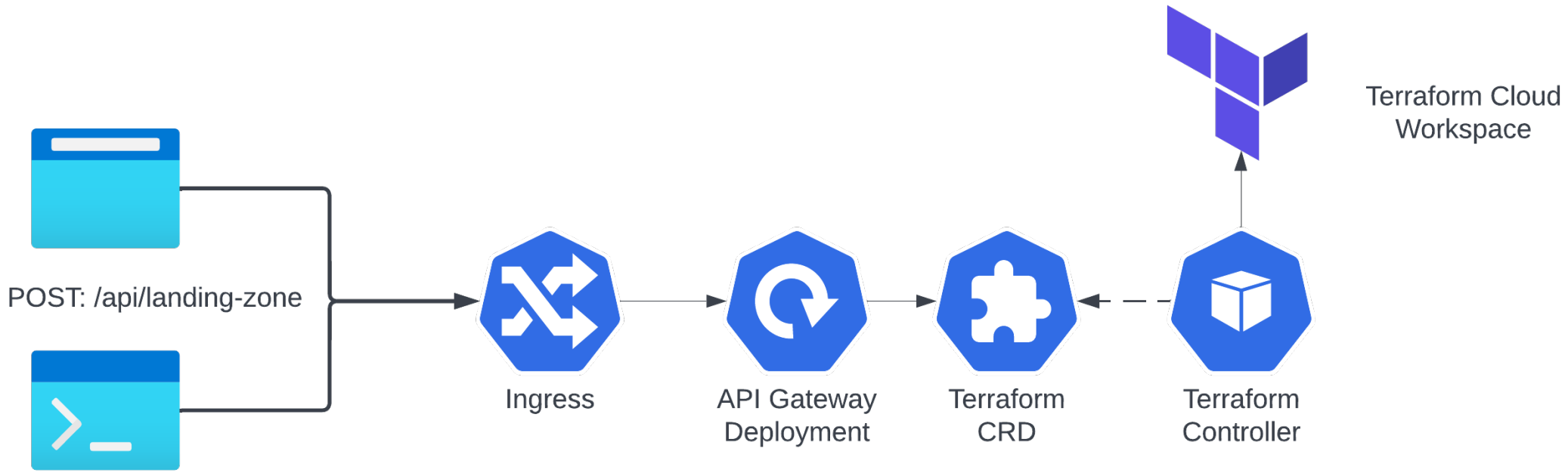
Highly effective continuous delivery

Can be combined with other Cloud-Native applications

- Progressive delivery
- Automatic certificate for web apps

Requires some Kubernetes skills

Self-service on using Cloud-Native technology




Tusen takk!
MVP-Dagen