

Building an Internal Developer Platform on Azure

Robert Strand

Crayon Group | @roberthTweets

MVP
Dagen



Robert 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



audiocodes

glasspaper

POINT:TAKEN

EPOS

aztek

Evidi

KPMG

CloudWay

spirhed

Noroff
School of technology
and digital media

blinQ

amesto
Fortytwo

SparebankenVest

ITstyring

INNOFACTOR

MVP-Dagen

Topics

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



Photo by [Kelly Sikkema](#) on [Unsplash](#)

Let's talk about DevOps

A quick introduction, and what it means today

What is DevOps?

Learn how DevOps unifies people, process, and technology to bring better products to customers faster

[Explore DevOps solutions](#)

DevOps overview DevOps culture DevOps practices DevOps tools DevOps and the cloud Get started FAQs

DevOps definition

A compound of development (Dev) and operations (Ops), DevOps is the union of people, process, and technology to continually provide value to customers.

What does DevOps mean for teams? DevOps enables formerly siloed roles—development, IT operations, quality engineering, and security—to coordinate and collaborate to produce better, more reliable products. By adopting a DevOps culture along with DevOps practices and tools, teams gain the ability to better respond to customer needs, increase confidence in the applications they build, and achieve business goals faster.

DevOps culture

While adopting DevOps practices automates and optimizes processes through technology, it all starts with the culture inside the organization—and the people who play a part in it. The challenge of cultivating a DevOps culture requires deep changes in the way people work and collaborate. But when organizations commit to a DevOps culture, they can create the environment for high-performing teams to develop.



Collaboration, visibility, and alignment

One hallmark of a healthy DevOps culture is collaboration between teams, which starts with visibility. Different teams such as development and IT operations must share their DevOps processes, priorities, and concerns with each other. These teams must also plan work together as well as align on goals and measures of success as they relate to the business.



Shifts in scope and accountability

As teams align, they take ownership and become involved in additional lifecycle phases—not just the ones central to their roles. For example, developers become accountable not only to the innovation and quality established in the develop phase, but also to the performance and stability their changes bring in the operate phase. At the same time, IT operators are sure to include governance, security, and compliance in the plan and develop phase.



Shorter release cycles

DevOps teams remain agile by releasing software in short cycles. Shorter release cycles make planning and risk management easier since progress is incremental, which also reduces the impact on system stability. Shortening the release cycle also allows organizations to adapt and react to evolving customer needs and competitive pressure.

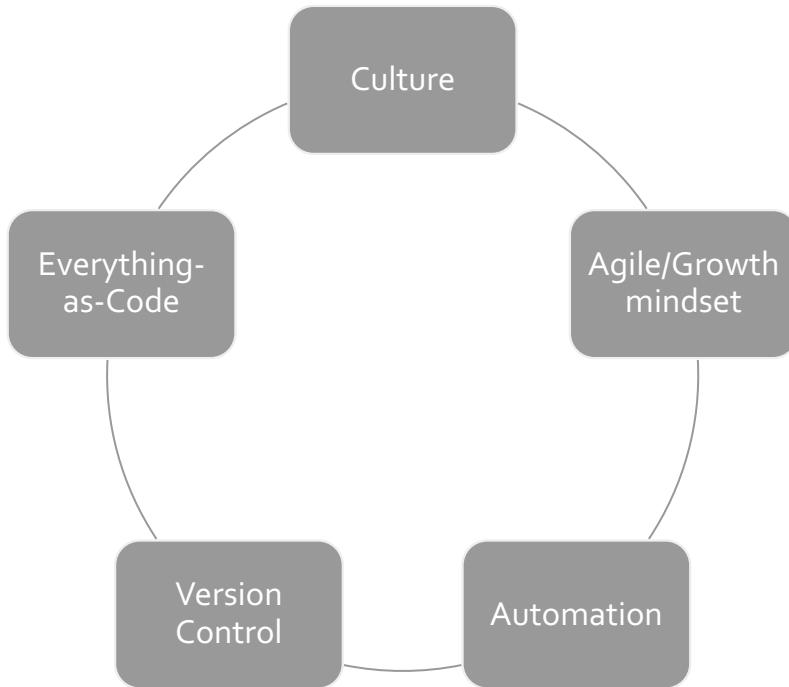


Continuous learning

High-performing DevOps teams establish a growth mindset. They fail fast and incorporate learnings into their processes, continually improving, increasing customer satisfaction, and accelerating innovation and market adaptability. DevOps is a journey, so there is always room to grow.

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

DevOps Principles & Practices



Team Topologies (2019)

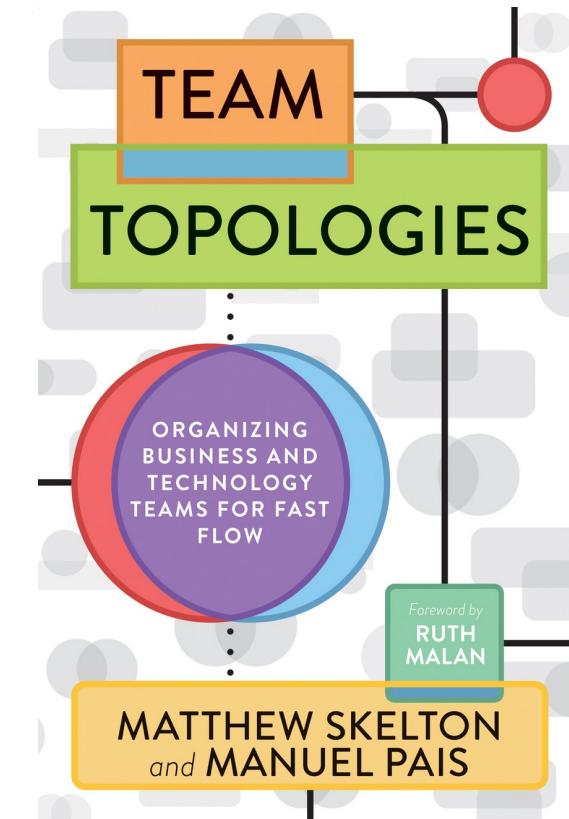
Stream-aligned team

Enabling team

Complicated subsystem team

Platform team

Visit teamtopologies.com

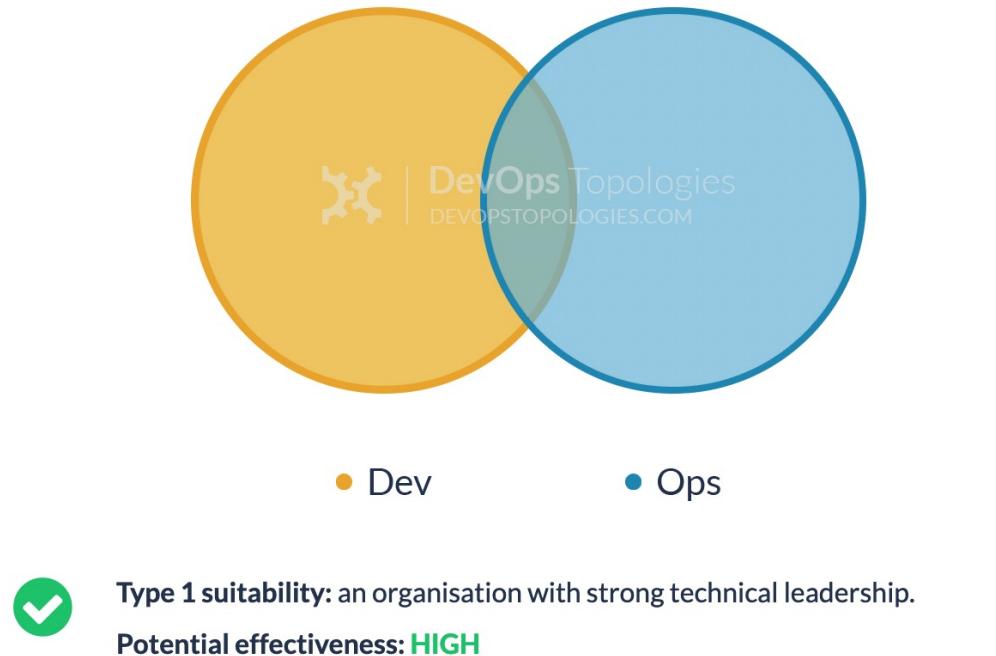


DevOps Topologies

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.



Source: web.devopstopologies.com

DevOps Topologies



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 arguably 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)**).

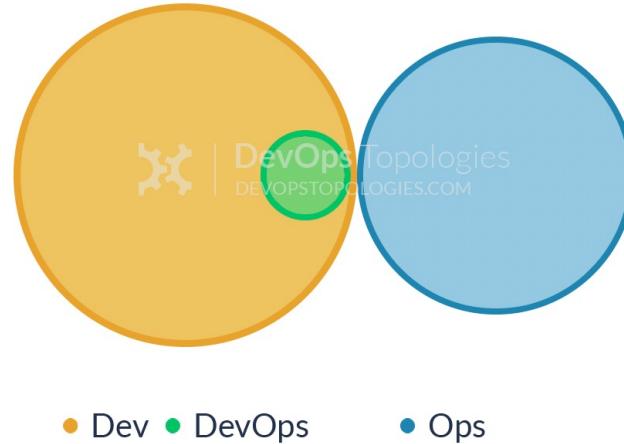
DevOps Topologies

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

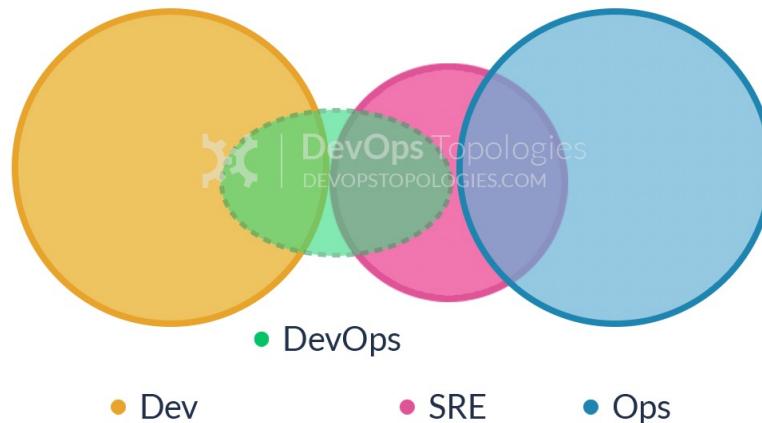
Source: web.devopstopologies.com

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



Registry

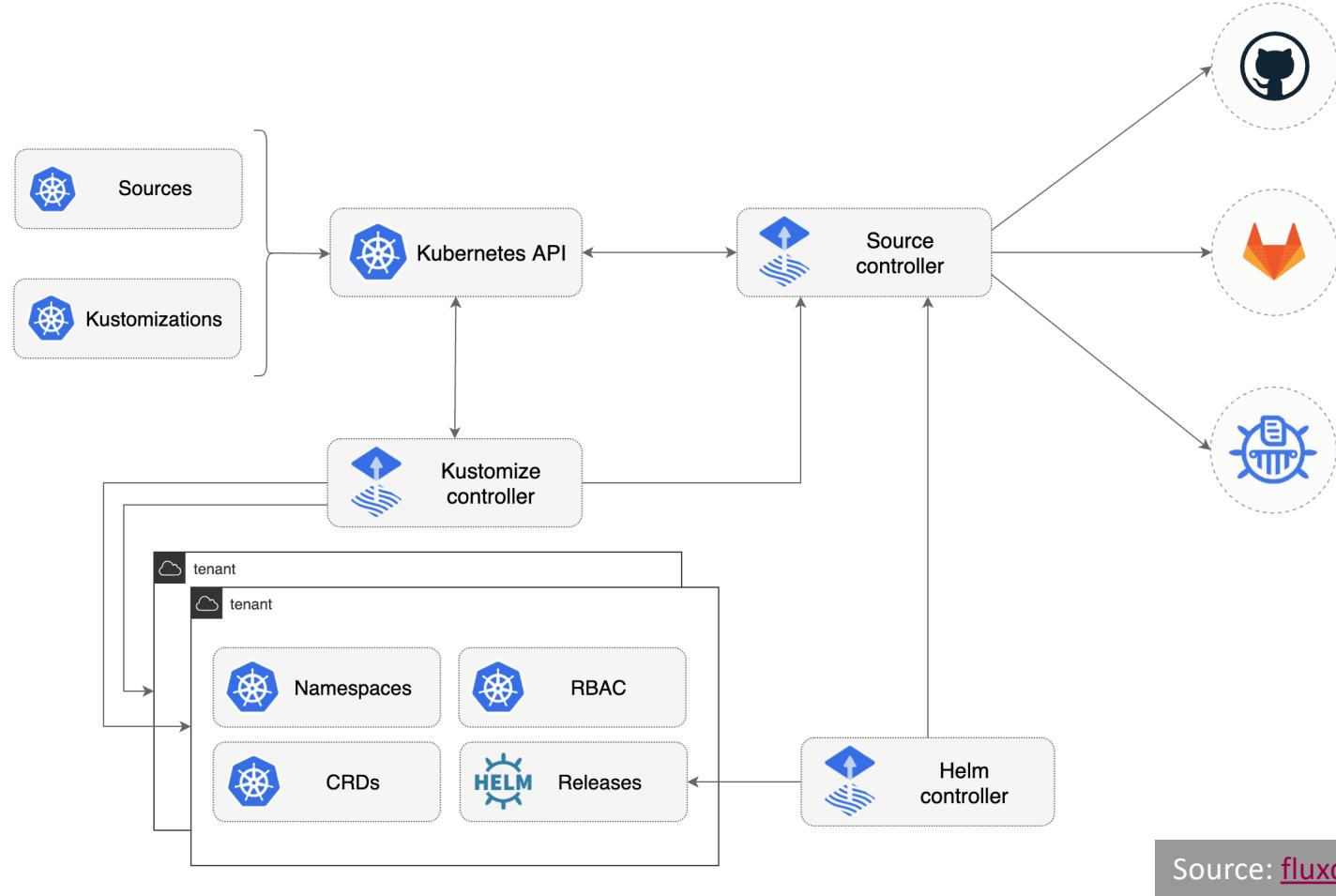
The screenshot shows the Terraform Cloud Registry interface. At the top, there are buttons for 'Design configuration' and 'Search public registry'. Below that is a search bar labeled 'Filter providers and modules'. Under the 'Providers' tab, there's a 'No-code Provisioning' section with a 'BETA' badge, a checkbox for 'No-Code Ready', and a call-to-action button 'Create your own →'. The main list displays several modules: 'network' (version 2.1.0), 'virtual-machine' (version 1.0.1), 'web-server-aws' (version 0.0.8), and 'random'. Each item has a status indicator (e.g., 'Private', 'aws'), a version number, and a timestamp ('12 days ago' or '9 days ago').

The screenshot shows the Terraform Cloud Configuration Designer. It starts with a 'Select Modules' step where 'network' and 'virtual-machine' are selected. The next step, 'ADD MODULES TO WORKSPACE', shows the 'network' module being added. The final step, 'CONFIGURE VARIABLES', shows configuration fields for 'region', 'subnet_availability_zone', 'subnet_cidr_block', and 'vpc_cidr_block'.

The screenshot shows the ServiceNow Service Management Catalog. On the left, the navigation menu includes 'Catalogs', 'Favorites', 'Service Catalog - Catalogs', 'Configuration', 'Database Catalogs', and 'Service Catalog'. The main area displays the 'Terraform Enterprise Catalog' under the 'Service Catalog' category. Other visible catalog items include 'Service Catalog - IT Now' and 'Technical Catalog'.

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

GitOps



GitOps

Highly effective continuous delivery

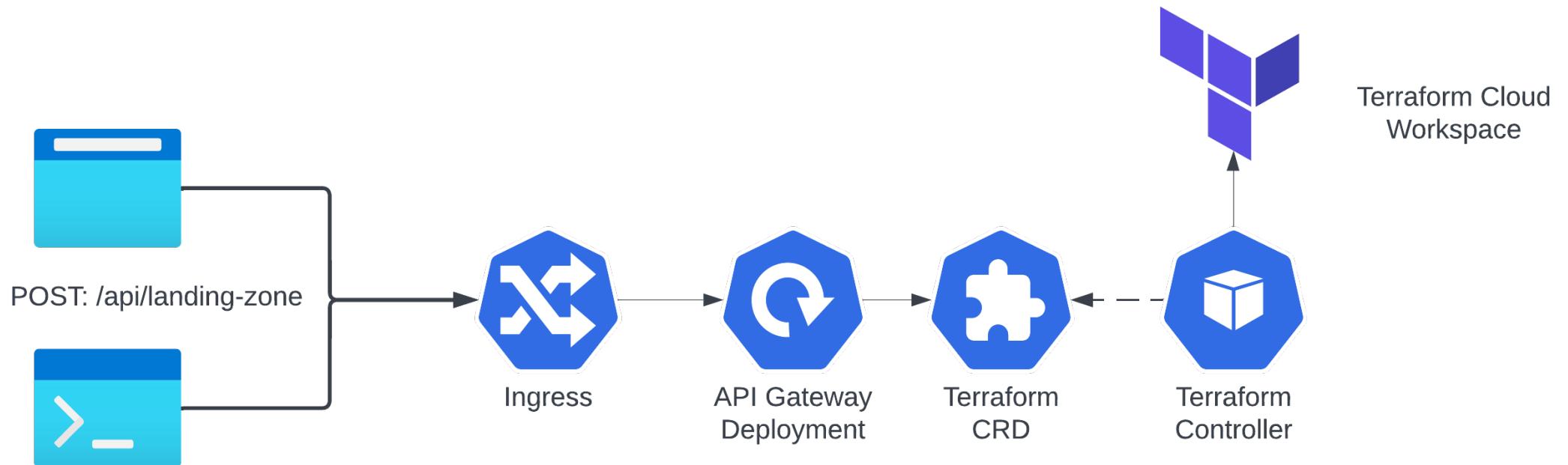
Can be combined with other Cloud-Native applications

Progressive delivery

Automatic certificate for web apps

Requires some Kubernetes skills

Self-service on using Cloud-Native technology



???



A photograph of a large audience seated in rows, viewed through a blue-tinted glass window. The people are mostly men, wearing casual attire. The room has a modern design with a whiteboard and a door in the background.

Tusen takk!
MVP-Dagen