



# Cluster API: A Unified Approach to Cluster Lifecycle Management Across Diverse Environments

———— CND Bergen | October 2025 ————



**Simon Weald**

Site Reliability Engineer

# Outline

**Intro and overview of Cluster API**

**Why Cluster API?**

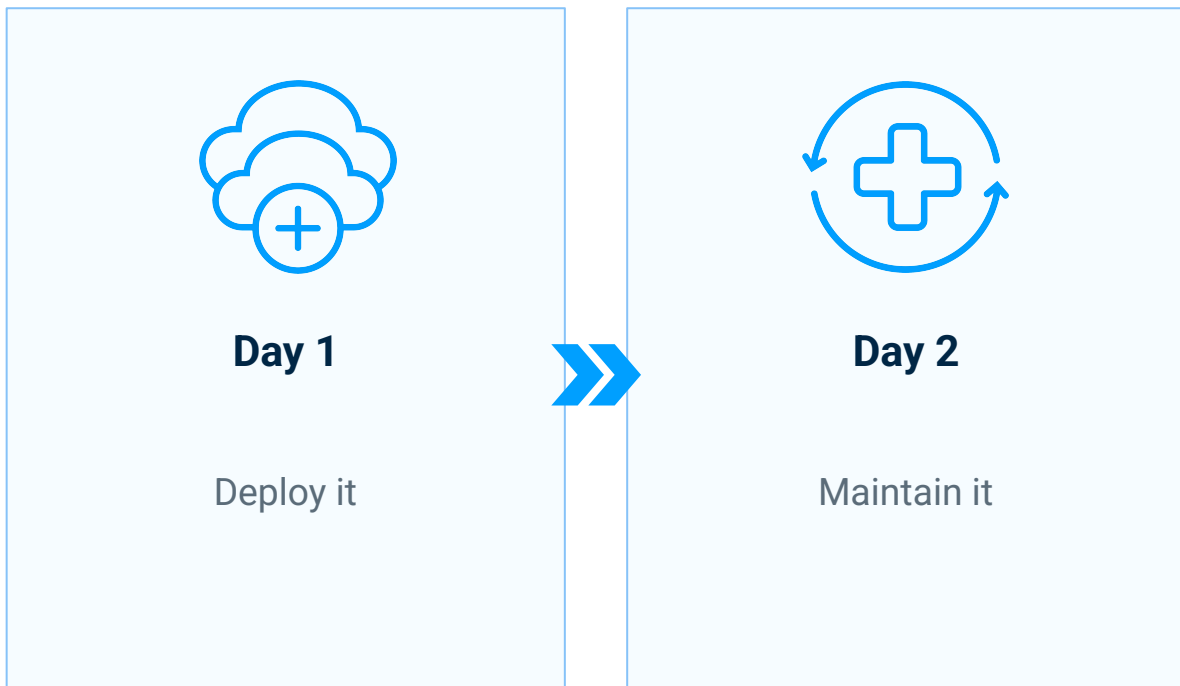
**Cluster API in practise**

**Demo**

**Wrap-up**

# Overview

# The challenge: lifecycle management



# Cluster creation isn't easy

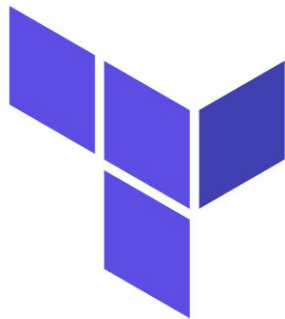
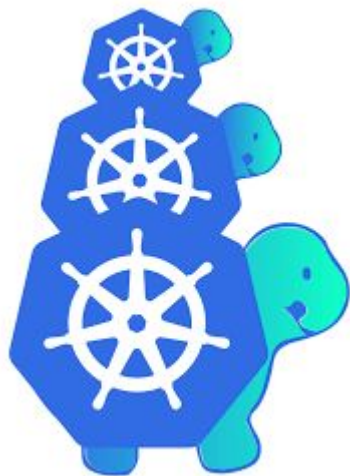
- Networking
- Identity management
- Compute
- Certificate management
- Bootstrapping of core cluster components

# Cluster maintenance also isn't easy

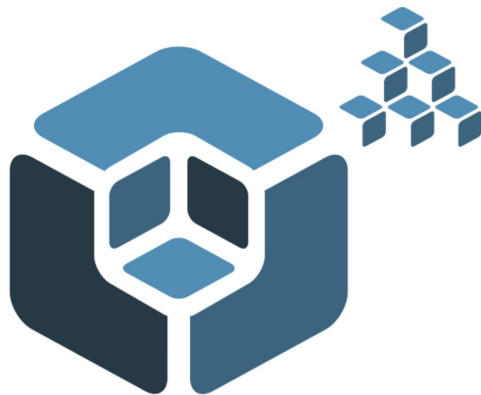
- Security and vulnerabilities
  - Releases fix vulnerabilities
- Stability and performance
  - Updates patch bugs and improve performance
- Compliance and support limitations
  - Cloud providers often only support the latest three minor versions
- Scaling
  - Clusters may need more resources
- Fault recovery
  - Incidents require remediation

# Tools

There are **lots** of options



HashiCorp  
**Terraform**



**KUBESPRAY**



**K3S**



**kops**



# Why Cluster API?



# Manual cluster creation

- Creates snowflakes
  - Clusters are unique and unreproducible
  - Differences happen between environments
- Causes config drift
  - Ad-hoc changes cause deviations
  - You cannot reconcile the desired state vs the actual state
- Requires manual operations
  - Upgrades aren't smooth
  - Scaling is often slow with a human in the pipeline



# What is Cluster API?



The Cluster API project uses Kubernetes-style APIs and patterns to automate cluster lifecycle management

*SIG Cluster Lifecycle, <https://cluster-api.sigs.k8s.io/>*

# Automated cluster creation

- Automated by controllers
  - Provisioning, configuration and teardown
  - No manual intervention or fragile custom scripts
- Config is declarative
  - The entire cluster is defined in YAML
  - Infra is now immutable and reproducible
- GitOps workflows
  - Desired state can now be stored in Git
  - Single source of truth
  - Versioned



# Basic concepts

- Management Cluster
  - A Kubernetes cluster which manages the lifecycle of Workload Clusters
- Workload Cluster
  - A Kubernetes cluster whose lifecycle is managed by a Management Cluster
- Infrastructure Controller
  - Provisions infrastructure required by the cluster
- ControlPlane Controller
  - Manages the control plane
- Bootstrap Controller
  - Turns the infrastructure into a cluster

# Cluster

- [Cluster](#) is the top-level resource
- Represents the logical cluster
- Provider agnostic
- References:
  - The [InfrastructureCluster](#) (e.g. [AWSCluster](#), [ProxmoxCluster](#))
  - The [ControlPlane](#) resource

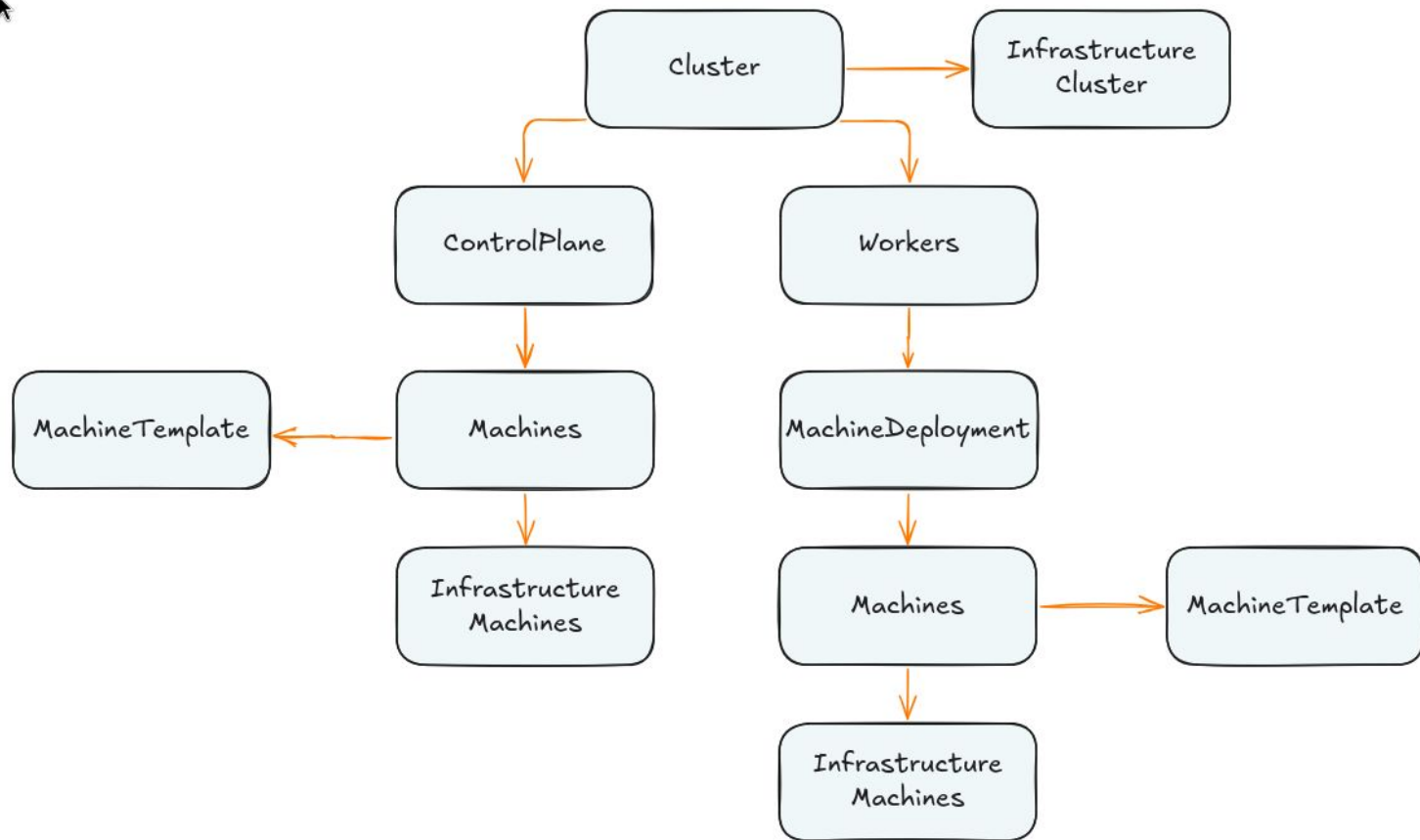
# Control plane

- Managed by a control plane provider
- Responsible for:
  - Managing the control plane [Machines](#)
  - Managing the services running on the control plane nodes
  - Rolling upgrades and scaling
- References a [MachineTemplate](#) for the underlying infrastructure

# Worker resources

- Managed by [MachineDeployments](#) and [MachineSets](#)
- [MachineDeployment](#)
  - Defines desired worker count, update strategy etc
  - References a [MachineTemplate](#) for the underlying infrastructure
- [MachineSet](#)
  - Handles replica management of worker [Machines](#)
  - Creates [Machine](#) objects (actual VMs)

# Putting it all together





# Lifecycle management is now easy



# Cluster API in practise

# Potential pitfalls

- Version skew can be a problem
- All providers are not equal
- Multiple controllers can make debugging difficult
  - No single pane of glass to identify issues
- [MachineTemplates](#) are immutable
- Management Clusters are a single point of failure

# Demo



# Wrap-up

# Wrap-up

- Cluster lifecycle management is complex
  - Creation, upgrades, scaling, and recovery all have traps
- Manual approaches lead to drift, unreproducibility, and operational toil.
- Cluster API abstracts complexity with declarative, provider-agnostic automation.
- It standardises lifecycle operations through controllers, YAML, and GitOps workflows.
- There are still pitfalls
  - Version skew, provider maturity, controller sprawl, and management cluster fragility
- Cluster API doesn't remove complexity - it organises it.

# Resources

## Slides and resources:

<https://github.com/a7d-corp/talks>

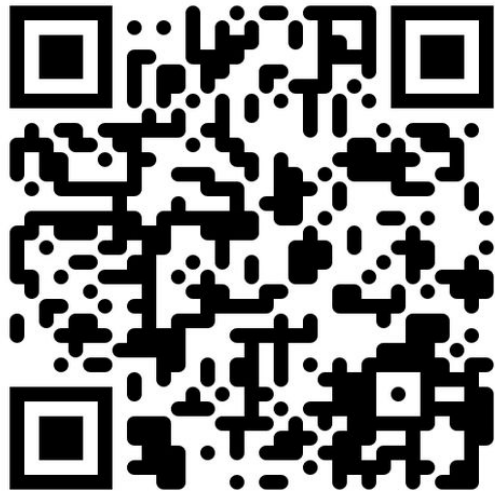
## Cluster API book:

<https://cluster-api.sigs.k8s.io/>

## Thoughts and feedback:

<https://simonweald.com>

[simon@simonweald.com](mailto:simon@simonweald.com)







# Thank you