



KubeCon



CloudNativeCon

Europe 2026

#KubeCon #CloudNativeCon

# In-place Updates with Cluster API

**The Sweet Spot between Immutable and  
Mutable Infrastructure**





KubeCon



CloudNativeCon

Europe 2026



## Stefan Büringer

Principal Engineer  
VMware by Broadcom  
Cluster API maintainer



## Fabrizio Pandini

Principal Engineer  
VMware by Broadcom  
Cluster API maintainer



KubeCon



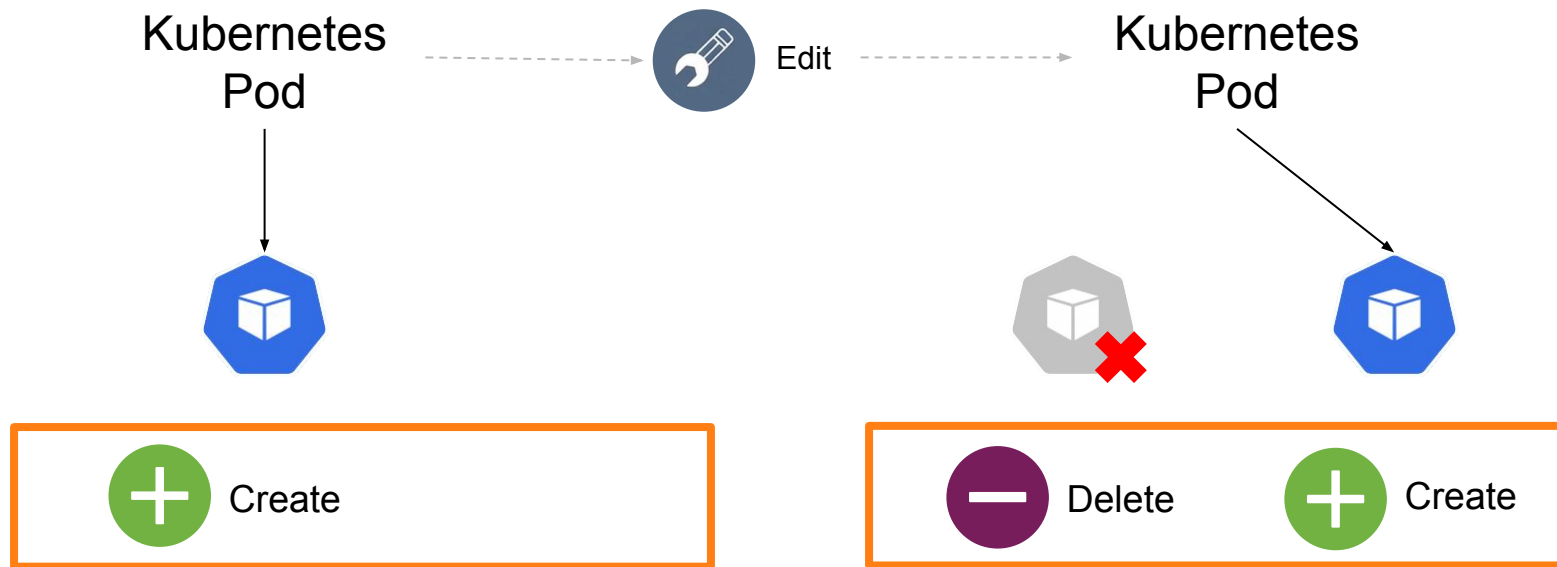
CloudNativeCon

Europe 2026

# Introduction



# Immutability 101



**In Cluster API Machines are treated like Pods in Kubernetes**

**Create, delete, no in-place updates**

# Traits of an Immutable System

Simple, Robust



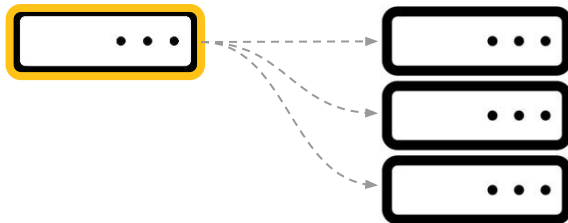
Create



Delete

The system is based on  
only two primitives

Predictable, Reliable



Every Pod/Machine is equal to the  
corresponding “Trusted Image”

**No more configuration drift surprises!**  
**(Snowflakes don't scale well)**

# Lessons Learned: The Reality is More Complex

## Friction at infrastructure level



Performance of creating infrastructure varies significantly

## Friction at application level

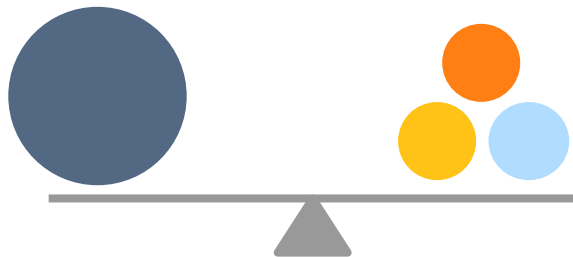


Not all applications can handle drain nicely

**Kubernetes is improving in this area (e.g. in-place Pod resize)  
What about Cluster API?**

# The Sweet Spot between Immutable and Mutable Infrastructure

Cluster API  
is immutable  
“by design”



Immutable rollout improvements:

- Avoid unnecessary rollouts
- Reduce Pod churn during rollouts
- Skip worker Machine rollouts when upgrading multiple minors

**In-place updates:**

- Perform in-place updates on existing Machines



KubeCon



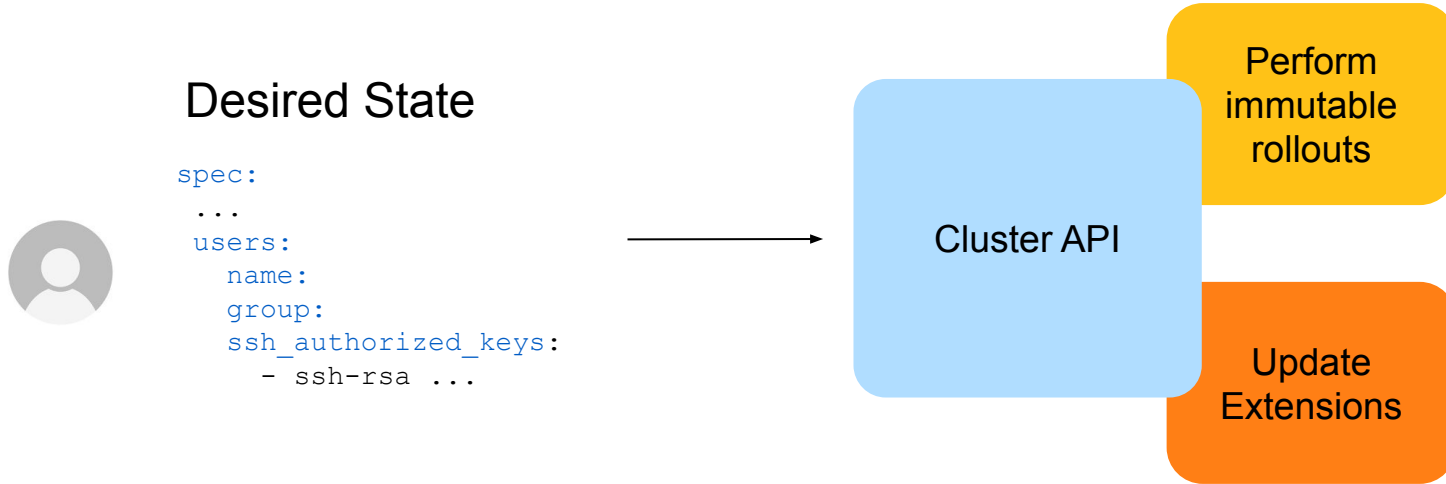
CloudNativeCon

Europe 2026

# Principles of In-place Updates In Cluster API



# User experience should remain the same



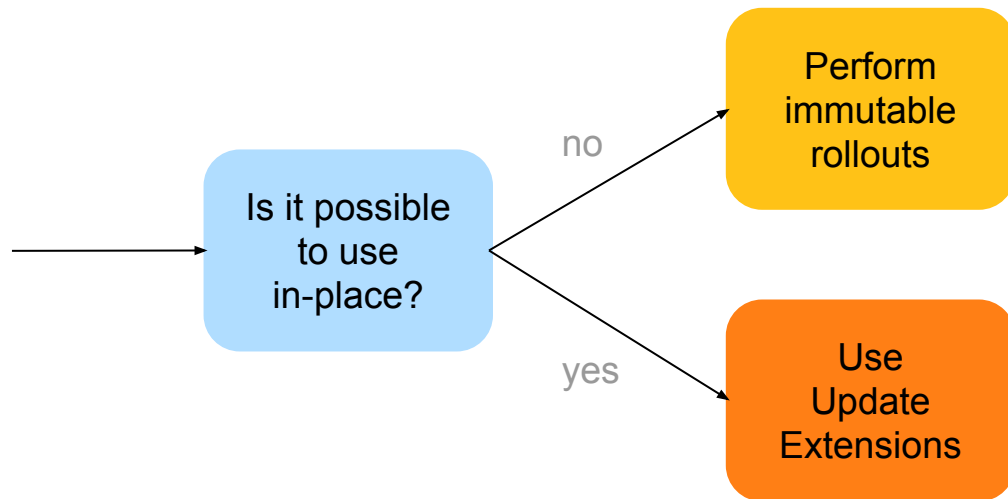
**Users should focus on desired state**  
**Cluster API is responsible to determine how to achieve it**

**In-place updates can be enabled (or disabled) at any time  
and without any change in user experience**

# Availability Constraints Also Apply to In-place Updates

## Desired State

```
spec:  
  ...  
  replicas: 3  
  strategy:  
    type: RollingUpdate  
    rollingUpdate:  
      maxSurge: 1  
      maxUnavailable: 0
```



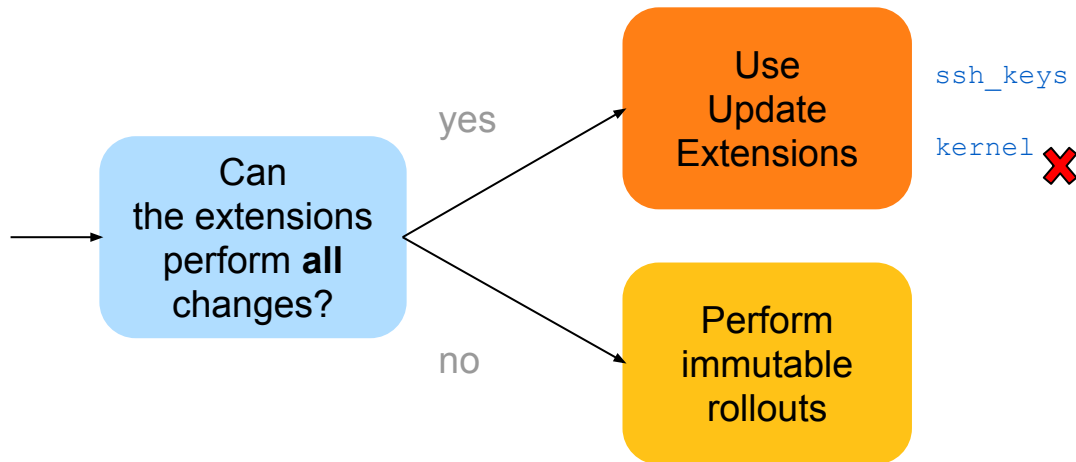
**In-place updates might lead to Machine unavailability**

**Cluster API will perform in-place updates only if this approach does not violate constraints defined in desired state**

# Minimise Rollouts while Reaching the Desired State

## Desired State

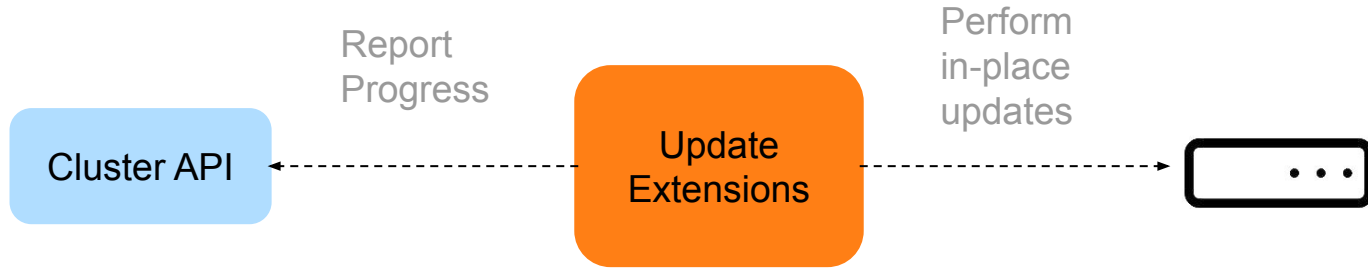
```
spec:
  ...
  users:
    name:
    group:
    ssh_authorized_keys:
      - ssh-rsa ...
  os:
    type: ubuntu
    kernel: 6.19
```



**Cluster API assumes that Update Extensions are capable to perform only a limited set of changes in-place**

**The system adapts to the capabilities of the Update Extensions**

# No Human Intervention During In-place Updates



**Update extensions should perform only carefully validated, repeatable in-place updates**

**Cluster API does not have opinions on how updates should be performed**



KubeCon



CloudNativeCon

Europe 2026

# In-place Updates for Control Plane Machines



# Immutable Control Plane Rollouts

replicas: 3, maxSurge: 1 (maxUnavailable: 0)

→ because maxUnavailable is 0 Cluster API ensures that there are always 3 available replicas



replicas: 3, maxSurge: 0 (maxUnavailable: 1)

→ because maxUnavailable is 1 Cluster API only ensures that there are always 2 available replicas



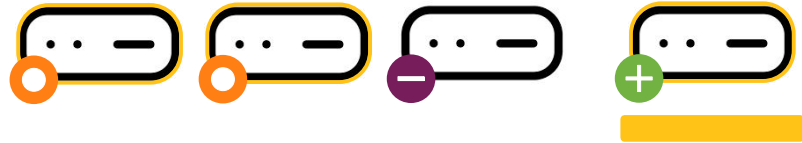
# Control Plane Rollouts with In-place Updates

When is it possible to use in-place?

- There is more than one replica
- The control plane is in a healthy state

replicas: 3, maxSurge: 1 (**maxUnavailable: 0**)

→ because maxUnavailable is 0 Cluster API ensures that there are always 3 available replicas



Create an additional replica, update two in-place and then delete one

# Control Plane Rollouts with In-place Updates

When is it possible to use in-place?

- There is more than one replica
- The control plane is in a healthy state

replicas: 3, maxSurge: 0 (**maxUnavailable: 1**)

→ because maxUnavailable is 1 Cluster API only ensures that there are always 2 available replicas



All replicas are updated in-place



KubeCon



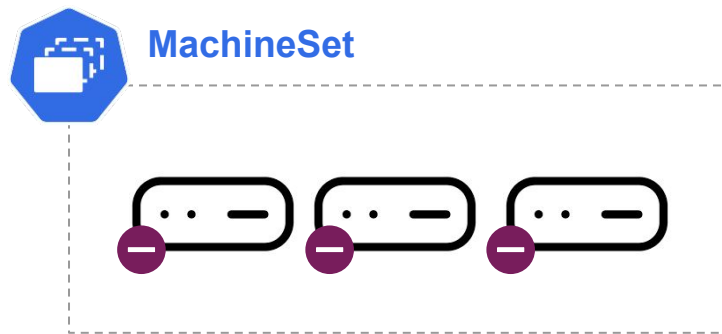
CloudNativeCon

Europe 2026

# In-place Updates for Worker Machines

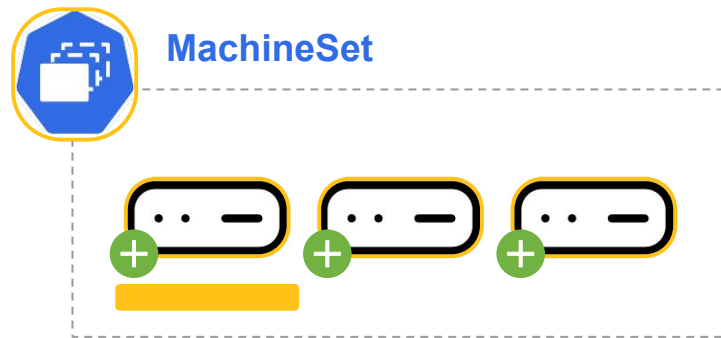


# Immutable MachineDeployment Rollouts



maxUnavailable: 0

**Defines how much  
disruption Cluster API  
should tolerate at any time**



maxSurge: 1

**Defines how much additional  
Machines Cluster API can create  
during rollouts**

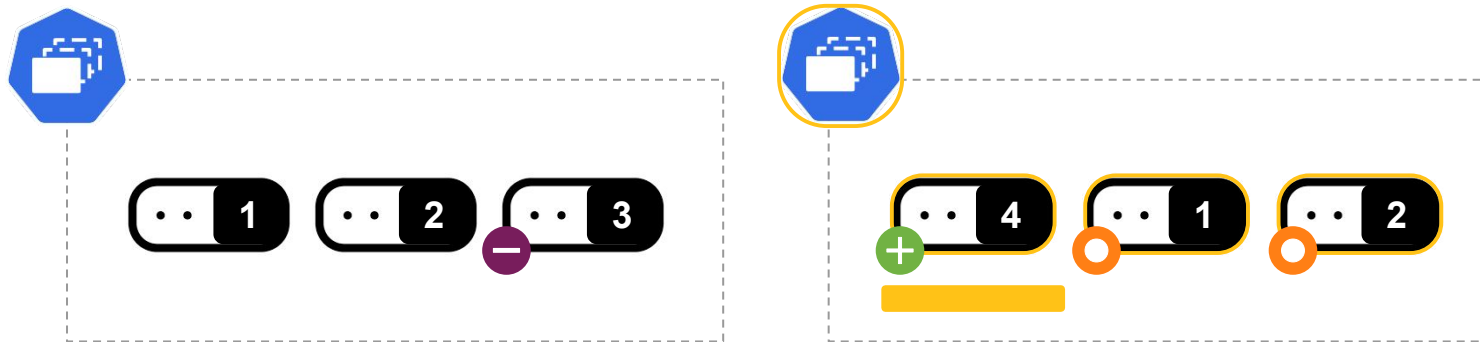
# MachineDeployment Rollouts with In-place Updates

When it is possible  
use in-place?

- Rollout strategy type is RollingUpdate

Replicas: 3, maxSurge: 1, **maxUnavailable: 0**

→ because maxUnavailable is 0 Cluster API ensures that there are always 3 available replicas



**Create an additional replica, update two in-place and then delete one**  
Only one slot from maxSurge is used

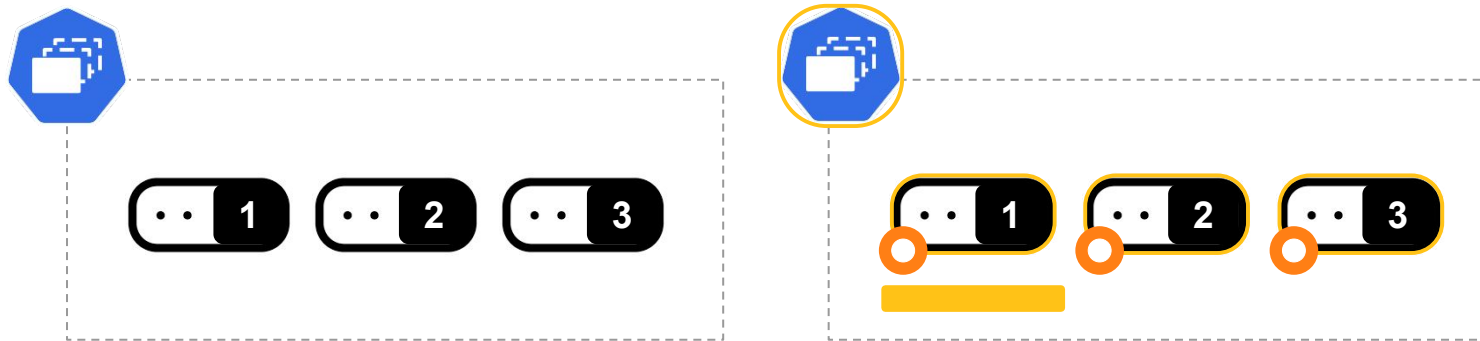
# MachineDeployment Rollouts with In-place Updates

When it is possible  
use in-place?

- Rollout strategy type is RollingUpdate

Replicas: 3, maxSurge: 0, **maxUnavailable: 1**

→ because maxUnavailable is 1 Cluster API only ensures that there are always 2 available replicas



**All replicas are updated in-place**  
No slots from maxSurge are used



KubeCon



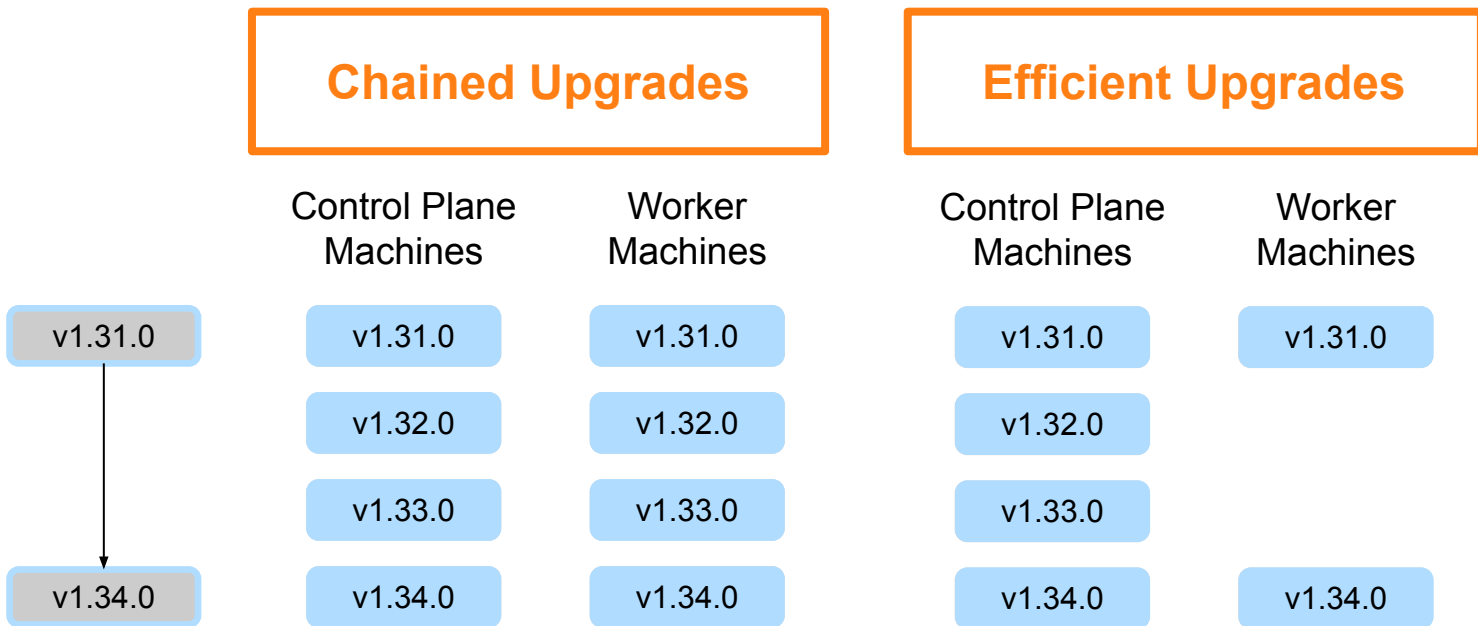
CloudNativeCon

Europe 2026

# Bonus slide



# Skip Worker Rollouts When Upgrading Multiple Minors



**Efficient Upgrades reduces the number of worker rollouts by 66%**



KubeCon



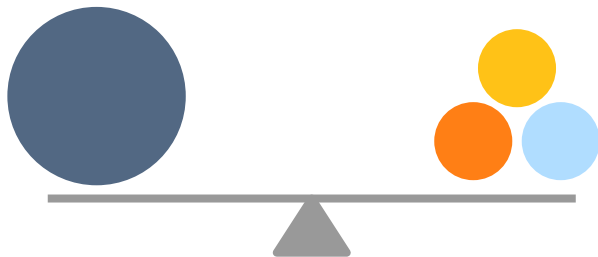
CloudNativeCon

Europe 2026

# Wrapping up



# The Sweet Spot between Immutable and Mutable Infrastructure



- **Immutability is and always will be at the core of Cluster API**
- Cluster API will continue to make immutable rollouts faster and more efficient
- **Cluster API now supports In-place updates on existing Machines**  
via Update Extensions

# Two cents from the Cluster API maintainers

- **In-place updates should not be used as a “workaround” to avoid fixing foundational problems in your platform**
  - If your application cannot handle drain nicely you must look into it and try to fix it.  
Drain is a foundational primitive of Kubernetes (trying to avoid drain is fighting Kubernetes)
  - If your infrastructure is slow you must look into it and try to improve speed.  
You cannot avoid creating or deleting Machines (this is Kubernetes, not a mainframe!)
- **In-place updates are best suited for rolling out changes that don't require a Node drain or Pod restarts**
  - If the application will be disrupted anyway, just do an immutable rollout



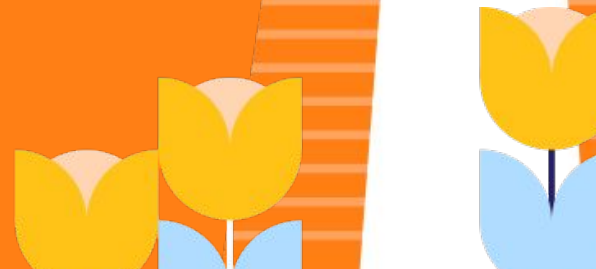
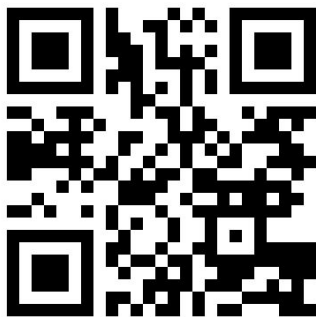
KubeCon



CloudNativeCon

Europe 2026

# Thank you!





KubeCon



CloudNativeCon

Europe 2026

