How to Migrate 700 Kubernetes Clusters to Cluster API with Zero Downtime

Type: Operations | Content Experience Level: Intermediate (Mid-level experience)

Mercedes-Benz Tech Innovation

How to Migrate 700 Kubernetes Clusters to Cluster API with Zero Downtime

Type: Operations | Content Experience Level: Intermediate (Mid-level experience)

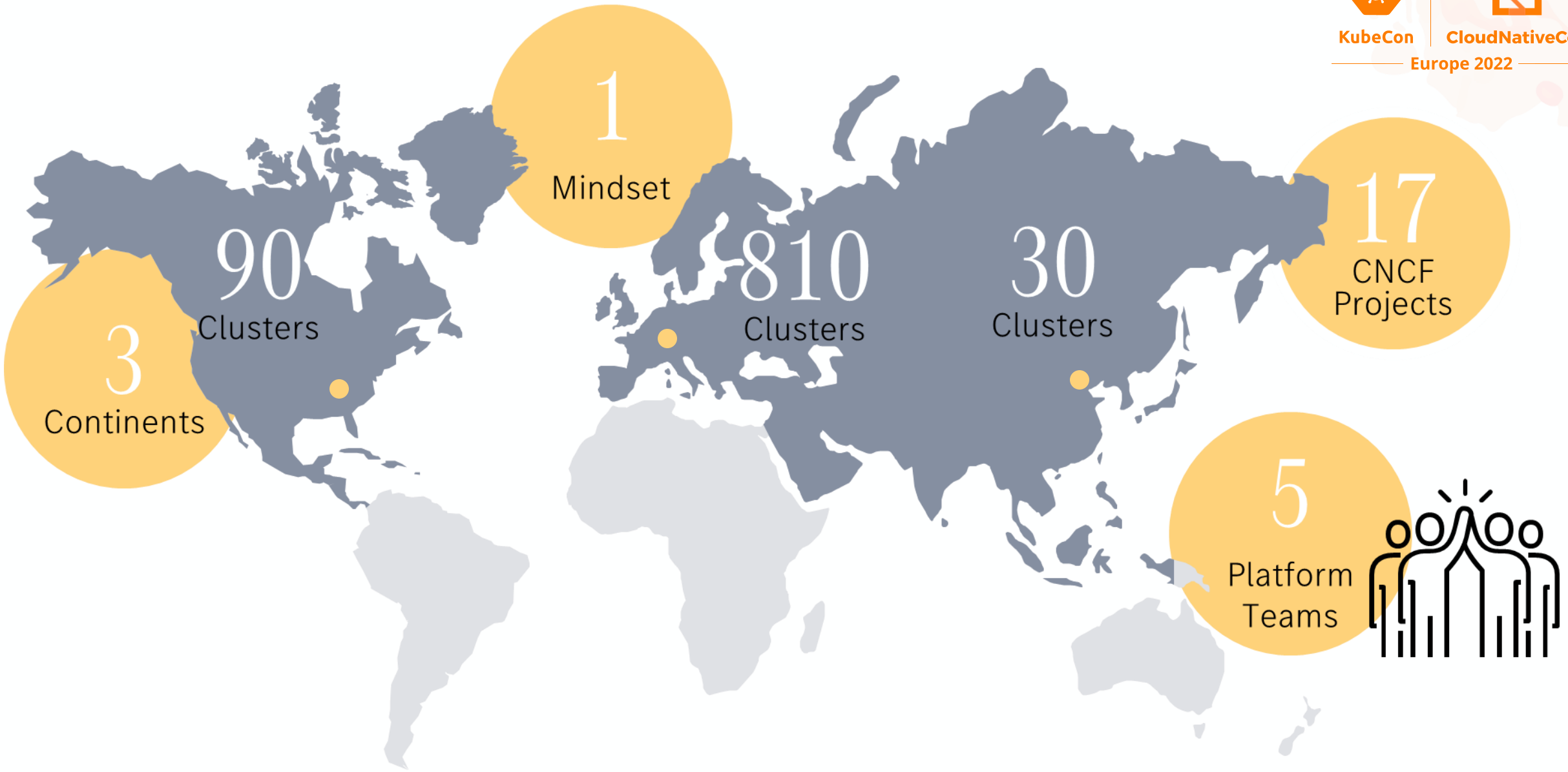
Sean Schneeweiss
Software Engineer

Tobias Giese
Software Engineer

Mercedes-Benz Tech Innovation

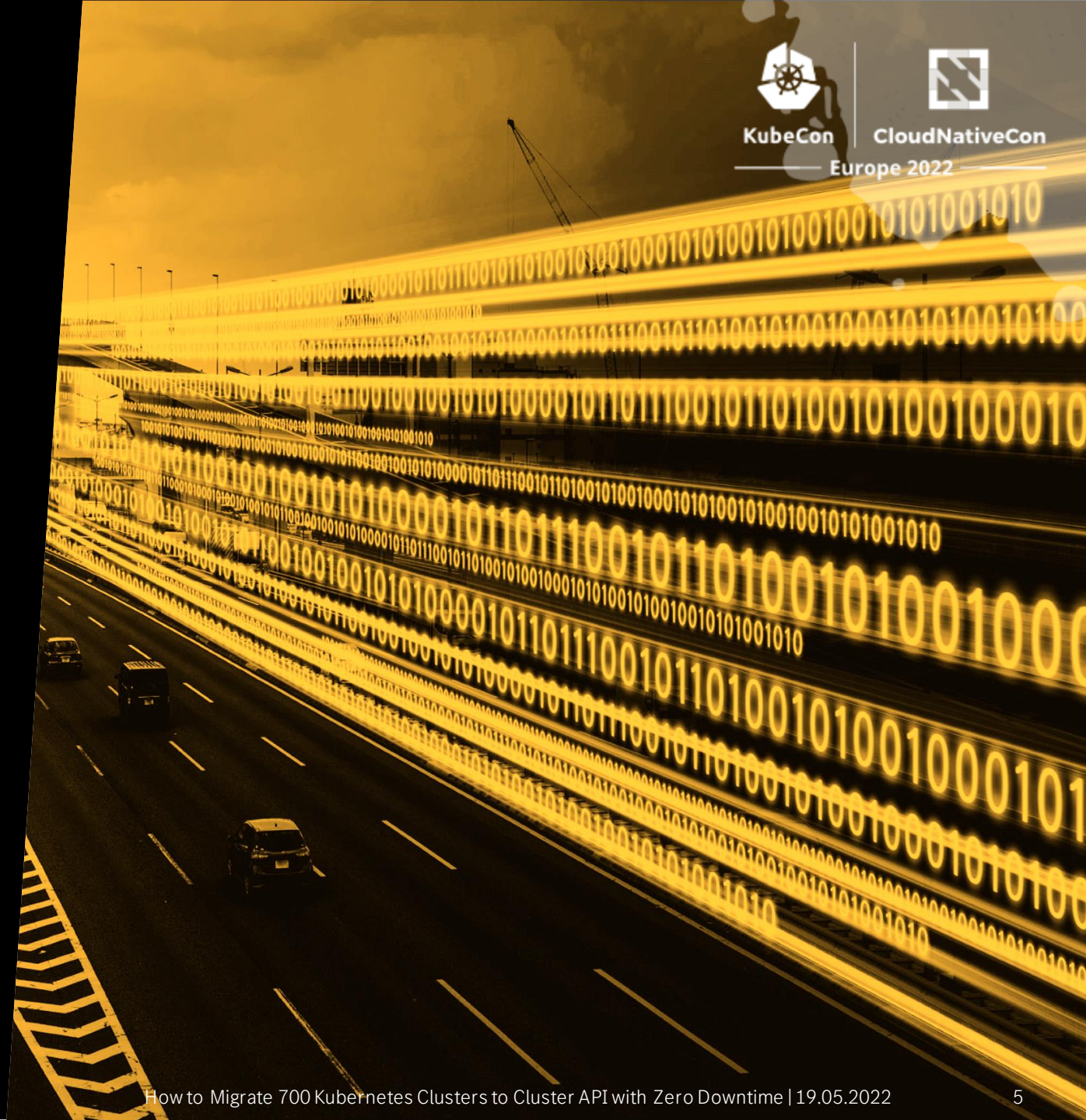
Mercedes-Benz Tech Innovation

Formerly known as Daimler TSS


1
Mindset
90
Clusters
810
Clusters
30
Clusters
17
CNCF
Projects
3
Continents
5
Platform
Teams

Agenda

1. Set the Stage
2. Legacy Provisioning
3. Migration to Cluster API
4. Lessons Learned
5. Next Steps
6. Get Involved



„Cluster API is a Kubernetes sub-project
focused on providing declarative APIs and tooling
to simplify provisioning, upgrading,
and operating multiple Kubernetes clusters.“

KubeCon CloudNativeCon Europe 2022

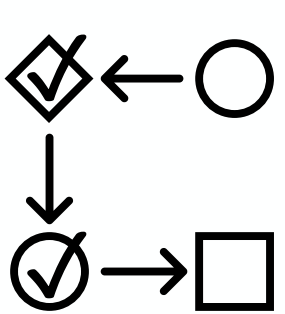
Mercedes-Benz Tech Innovation

1 Set the Stage

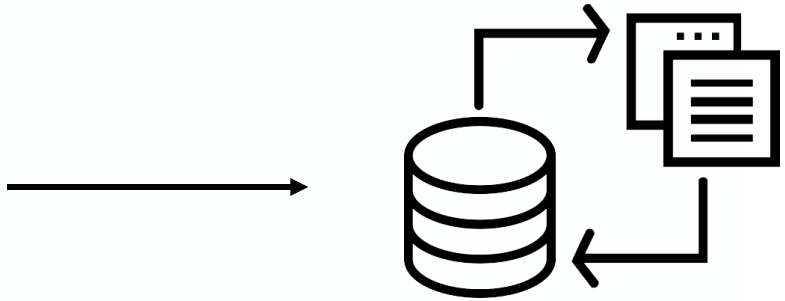
How to Migrate

Kubernetes Clusters

Zero Downtime



Legacy pipelines



Manage clusters
the Kubernetes style

1 Set the Stage

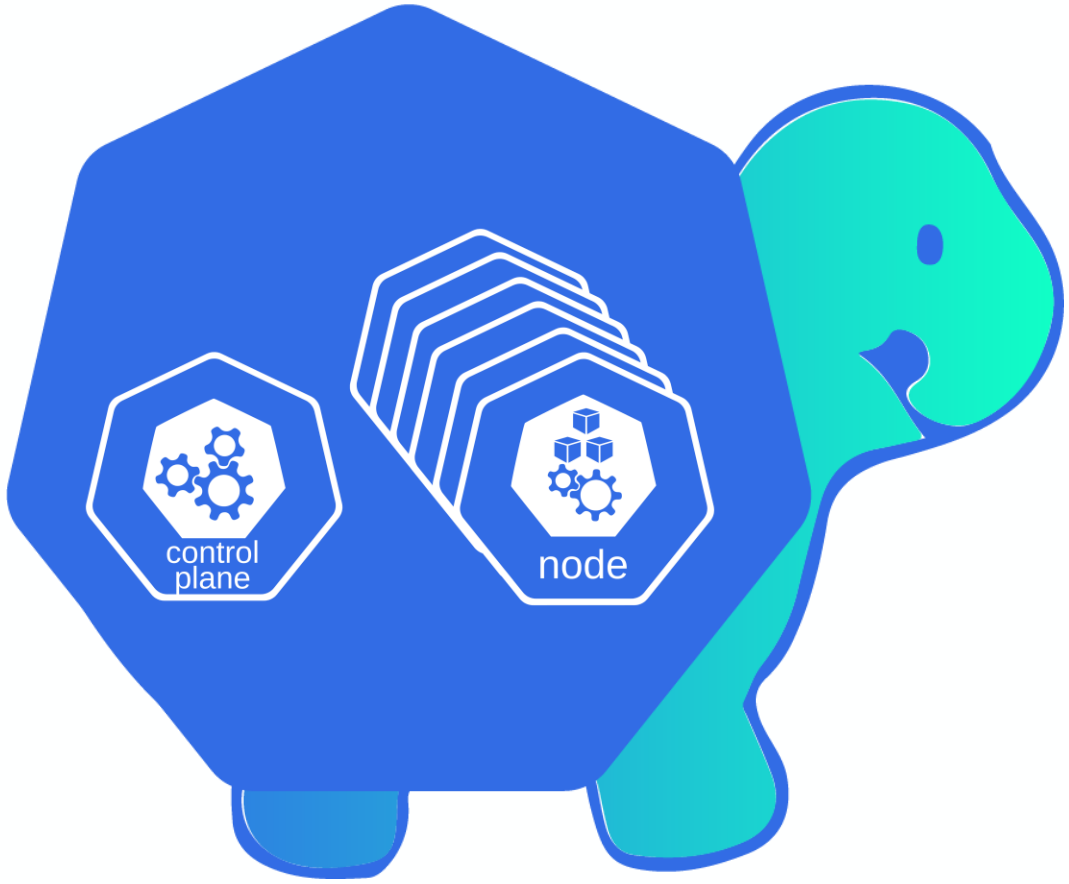
How to Migrate

Kubernetes Clusters

Zero Downtime

700 Workload Clusters

More than 200 clusters controlled by our largest Management Cluster



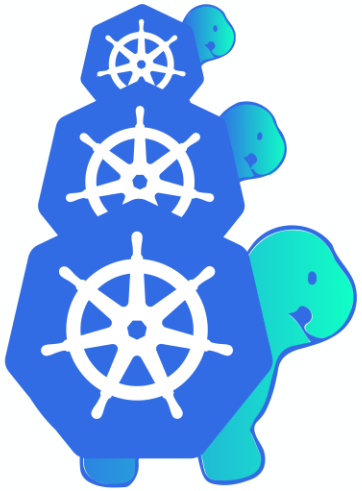
1 Set the Stage

KubeCon CloudNativeCon Europe 2022

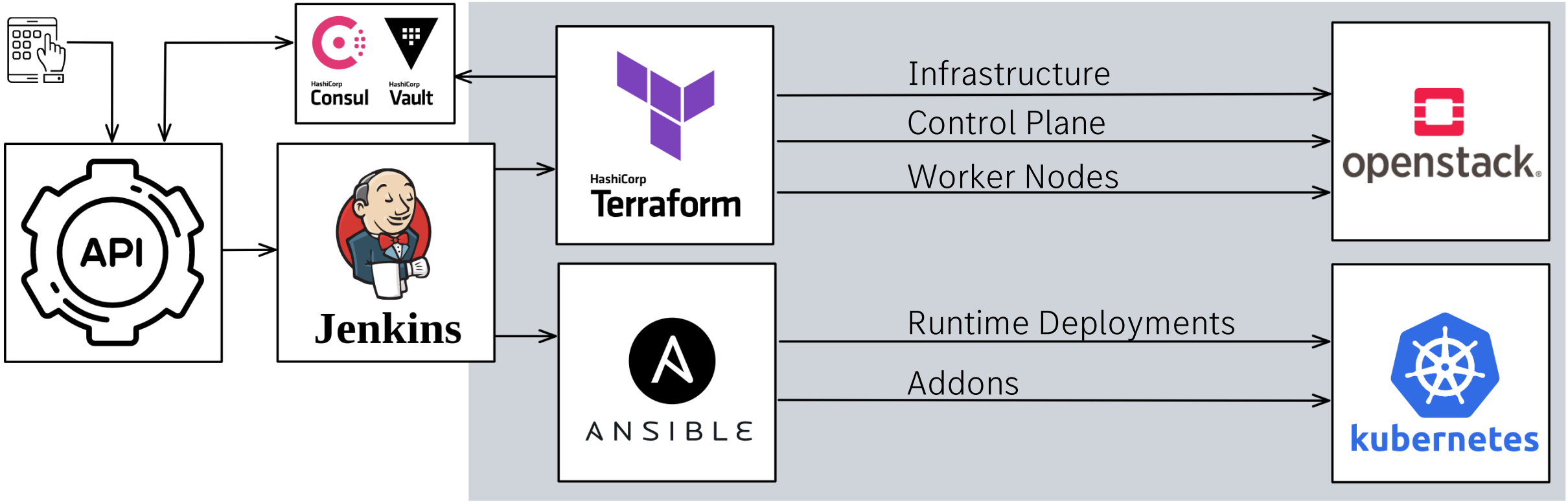
How to Migrate
Kubernetes Clusters
Zero Downtime

Users do not have to redeploy

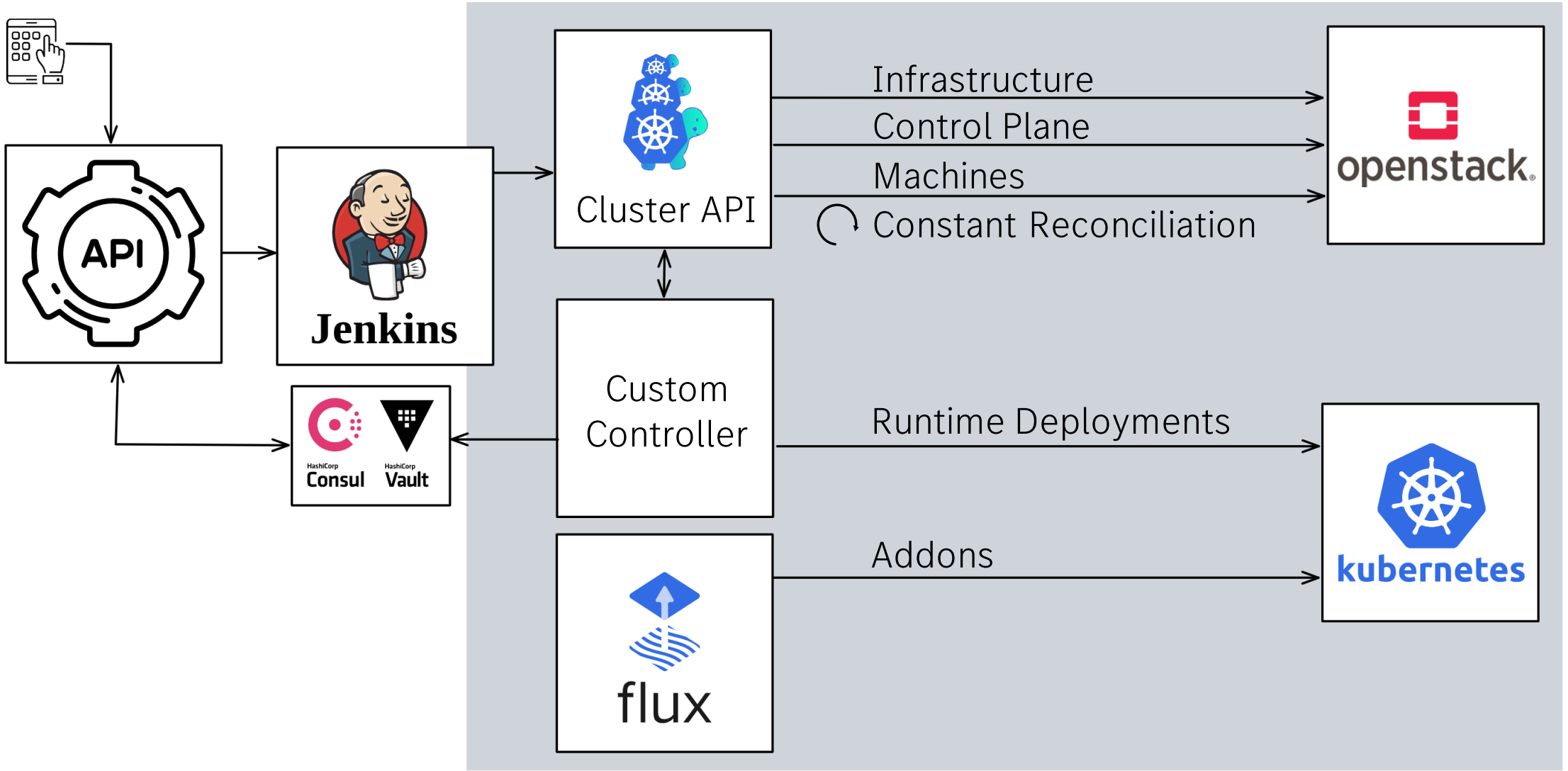
Control Plane and Worker Nodes are always available



2 Legacy Provisioning



3 Migration to Cluster API



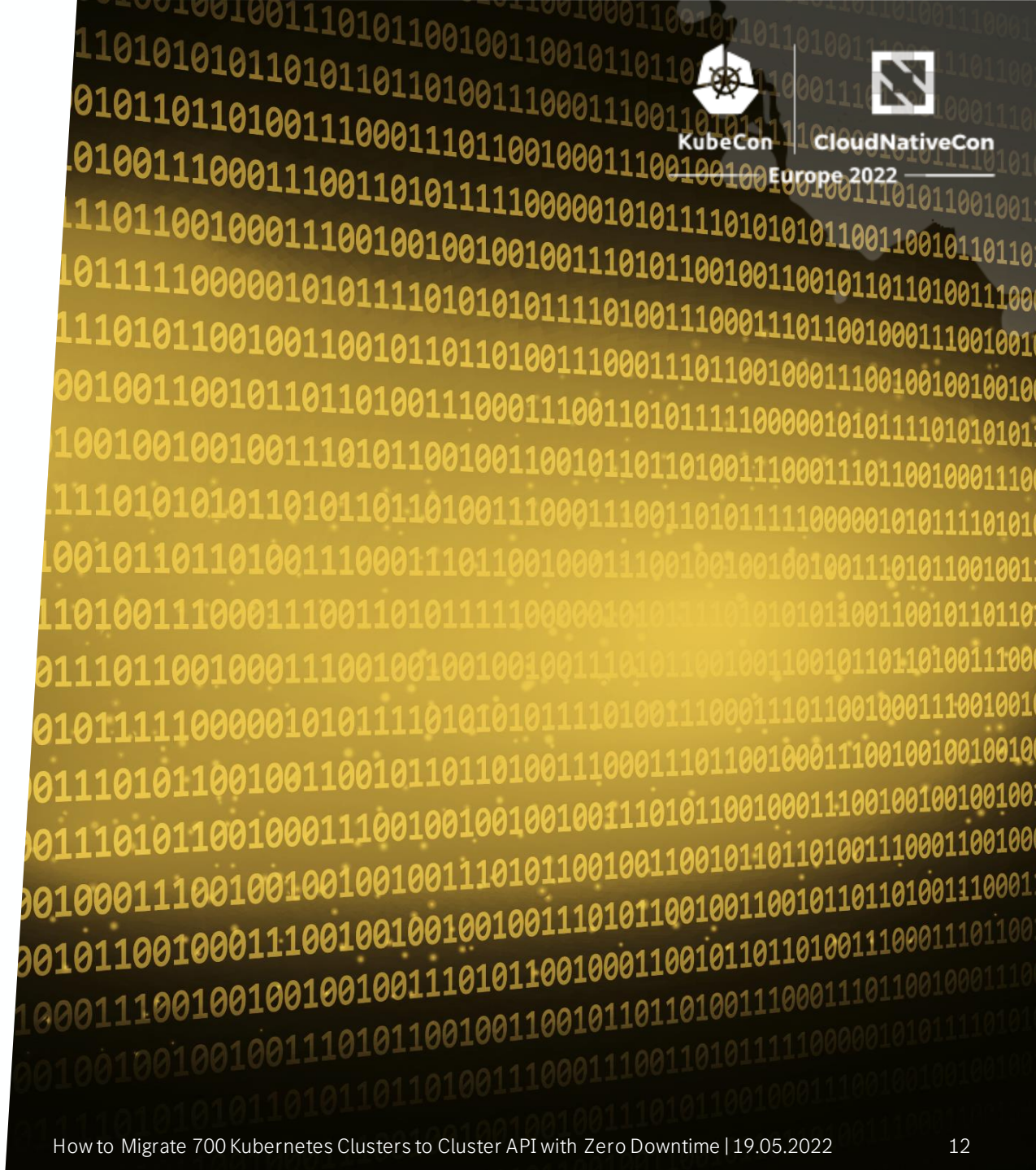
3 Migration to Cluster API

Cluster API should adopt all provider infrastructure specific objects to perform a rolling update

Users should be able to reach the Kubernetes API all the time

User workload must remain running and untouched

Users should not notice the migration



3 Migration to Cluster API - #0 Preparation

Cluster API should adopt already existing infrastructure objects

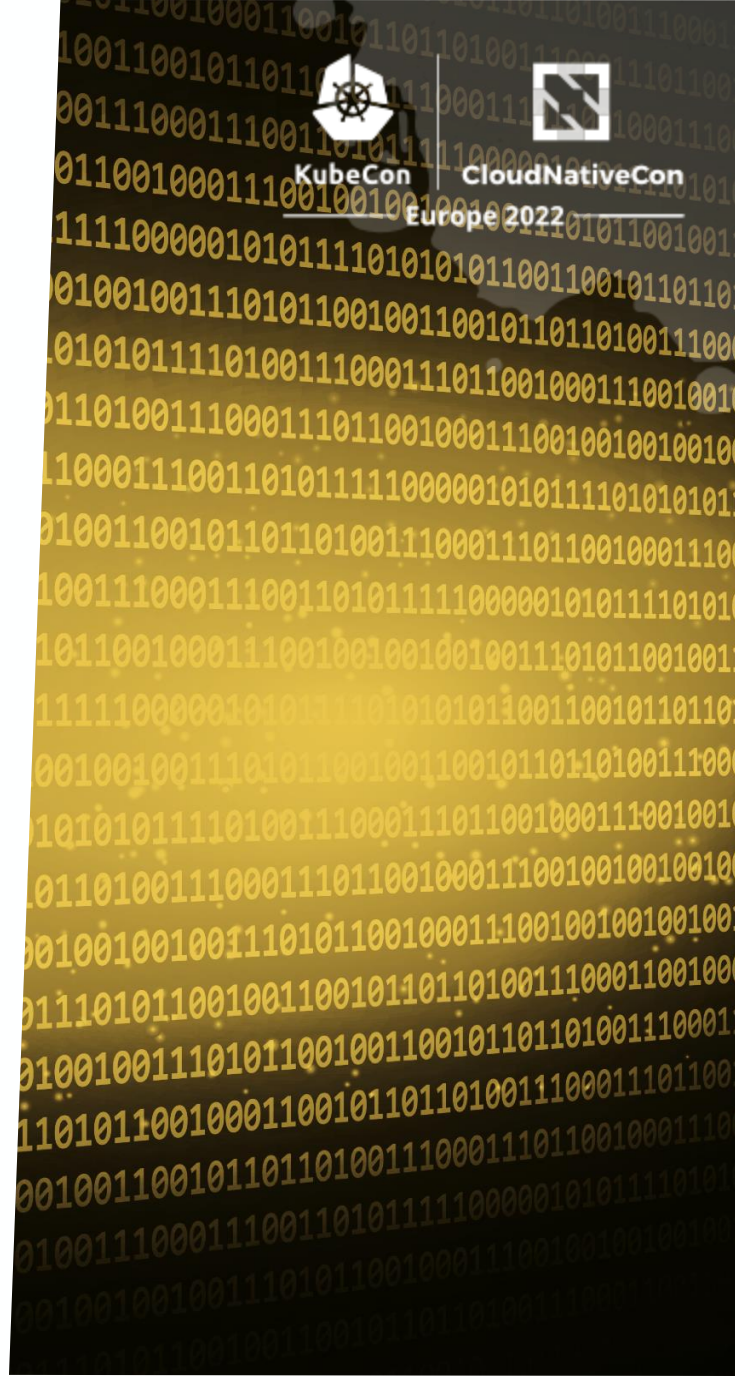
OpenStack specific objects, like:

- Router, Network, Subnet
- Load Balancer, Pool, Member, Listener, Health Monitor

Example names:

Router: k8s-clusterapi-cluster-namespace-my-cluster

Load Balancer: k8s-clusterapi-cluster-namespace-my-cluster-kubeapi



3 Migration to Cluster API

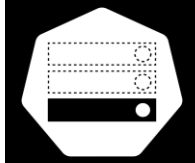
1 Migrate the Infrastructure

Cluster

Infra Cluster

2 Migrate the Worker Nodes

Machine Deployment



Machine Set

Machines & Infra Machines

3 Migrate the Control Plane

Kubeadm Control Plane

Machines & Infra Machines for Control Plane

3 Migration to Cluster API - #1 Infrastructure

KubeCon CloudNativeCon Europe 2022

1 Migrate the Infrastructure

Cluster

Infra Cluster

2 Migrate the Worker Nodes

Machine Deployment

Machine Set

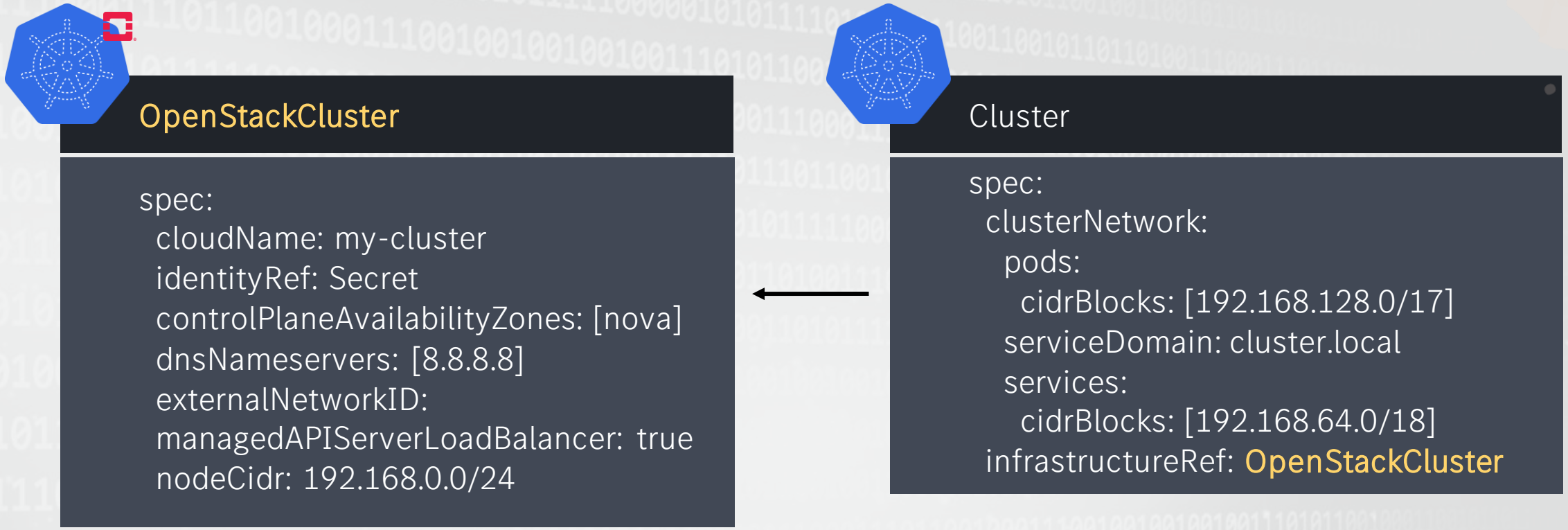
Machines & Infra Machines

3 Migrate the Control Plane

Kubeadm Control Plane

Machines & Infra Machines for Control Plane

3 Migration to Cluster API - #1 Infrastructure



Cluster API now reconciles the network, API load balancer, and firewall rules

3 Migration to Cluster API - #2 Worker Nodes

1

Migrate the Infrastructure

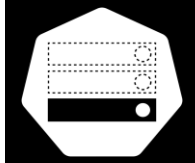
Cluster

Infra Cluster

2

Migrate the Worker Nodes

Machine Deployment



Machine Set

Machines & Infra Machines

3

Migrate the Control Plane

Kubeadm Control Plane

Machines & Infra Machines for Control Plane

3 Migration to Cluster API - #2 Worker Nodes

Add "fake" KubeadmControlPlane to allow the reconciliation.

Machine for Control Plane

```
metadata:
  name: control-plane-dummy
  annotations:
    cluster.x-k8s.io/paused: true
  labels:
    cluster.x-k8s.io/control-plane: true
    cluster.x-k8s.io/cluster-name: my-cluster
spec:
  clusterName: my-cluster
  bootstrap:
    dataSecretName: my-cluster-dummy
```

KubeadmControlPlane

```
metadata:
  name: my-cluster-dummy
  annotations:
    cluster.x-k8s.io/paused: true
spec:
  version: v0.0.0
  machineTemplate:
    infrastructureRef:
      apiVersion: infrastructure/v1alpha4
      kind: OpenStackMachineTemplate
      name: dummy
```

3 Migration to Cluster API - #2 Worker Nodes

Patch KubeadmControlPlane & Cluster status fields and condition.

Cluster

```
status:
  controlPlaneReady: true
  conditions:
  - type: ControlPlaneInitialized
    status: "True"
```

KubeadmControlPlane

```
status:
  initialized: true
  ready: true
```

Now we have a "fake" control plane and can continue to migrate the Worker Nodes.

3 Migration to Cluster API - #2 Worker Nodes

OpenStackMachines

```
spec:
  cloudName: my-cluster
  image: migration
  providerId: ${providerID}
  instanceId: ${instanceID}
```

Machines

```
spec:
  clusterName: my-cluster
  providerId: ${providerID}
  infrastructureRef: OpenStackMachine
  bootstrap:
    dataSecretName: secret-dummy
```

${providerID} and ${instanceID} must match the exact IDs of the instance.

Create an OpenStackMachine & Machine for each Worker Node of the cluster.

Secret

```
name: secret-dummy
```

3 Migration to Cluster API - #2 Worker Nodes

OpenStackMachines

Machines

MachineSet

```
metadata:
  labels: migration-labels
spec:
  clusterName: my-cluster
  replicas: 3
  template:
    labels: migration-labels
  selector:
    matchLabels: migration-labels
```

migration-labels

```
cluster.x-k8s.io/cluster-name: my-cluster
cluster.x-k8s.io/deployment-name: my-cluster-md
machine-template-hash: migration
```

Apply migration-labels to *metadata* of OpenStackMachines, Machines, and MachineSet.

3 Migration to Cluster API - #2 Worker Nodes

KubeadmConfigTemplate

OpenStackMachineTemplate

migration-labels

```
cluster.x-k8s.io/cluster-name: my-cluster
cluster.x-k8s.io/deployment-name: my-cluster-md
machine-template-hash: migration
```

MachineDeployment

```
metadata:
  name: my-cluster-md
spec:
  clusterName: my-cluster
  replicas: 3
  template:
    spec:
      version: v1.20.5
      bootstrap:
        configRef: KubeadmConfigTemplate
      infrastructureRef: OpenStackMachineTemplate
```

3 Migration to Cluster API - #2 Worker Nodes

legacy

```
$ kubectl get nodes
NAME                                STATUS   ROLES                  AGE   VERSION
my-cluster-legacy-control-plane01   Ready    control-plane,master   16m   v1.19.3
my-cluster-legacy-node01            Ready    <none>                 26m   v1.19.3
my-cluster-legacy-node02            Ready    <none>                 27m   v1.19.3
my-cluster-legacy-node03            Ready    <none>                 27m   v1.19.3
```

after migration

```
$ kubectl get nodes
NAME                                STATUS   ROLES                  AGE   VERSION
my-cluster-legacy-control-plane01   Ready    control-plane,master   36m   v1.20.5
my-cluster-md-6b6c846486-276zl      Ready    <none>                 29m   v1.20.5
my-cluster-md-6b6c846486-jp7x8      Ready    <none>                 28m   v1.20.5
my-cluster-md-6b6c846486-z7ckq      Ready    <none>                 29m   v1.20.5
```

3 Migration to Cluster API - #3 KubeadmControlPlane

1

Migrate the Infrastructure

Cluster

Infra Cluster

2

Migrate the Worker Nodes

Machine Deployment

Machine Set

Machines & Infra Machines

3

Migrate the Control Plane

Kubeadm Control Plane

Machines & Infra Machines for Control Plane

3 Migration to Cluster API - #3 KubeadmControlPlane

KubeadmControlPlane

```
metadata:
  annotations:
    cluster.x-k8s.io/paused: true
  name: my-cluster
spec:
  version: v1.21.3
  kubeadmConfigSpec: {...}
  machineTemplate:
    infrastructureRef: {...}
```

Create new KubeadmControlPlane (KCP) with real data.

The *dummy* KCP will be deleted afterwards.

Cluster API Webhook denies changes to some fields of the KCP spec, thus a new one is the easiest way.

3 Migration to Cluster API - #3 KubeadmControlPlane

Secret

name: secret-dummy

KubeadmConfig

name: kubeadmconfig-dummy

Machines for Control Plane

```
metadata:
  annotations:
    cluster.x-k8s.io/paused: true
  labels:
    cluster.x-k8s.io/control-plane: true
    cluster.x-k8s.io/cluster-name: my-cluster
spec:
  clusterName: my-cluster
  providerId: ${providerID}
  bootstrap:
    dataSecretName: secret-dummy
    configRef: kubeadmconfig-dummy
```

Loop over legacy Control Plane Nodes of the cluster and create Cluster API objects for the migration.

3 Migration to Cluster API - #3 KubeadmControlPlane

Secret

KubeadmConfig

Machines for Control Plane

OpenStackMachines for Control Plane

```
metadata:
  ownerReferences:
  - kind: Machine
  - kind: KubeadmControlPlane
  labels:
    cluster.x-k8s.io/control-plane:  true
    cluster.x-k8s.io/cluster-name:  my-cluster
spec:
  cloudName: my-cluster
  image: migration
  providerId: ${providerID}
  instanceId: ${instanceID}
```

${providerID} and ${instanceID} must match the exact IDs of the instance.

3 Migration to Cluster API - #3 KubeadmControlPlane

Cluster

```
spec:
  controlPlaneRef:
    name: my-cluster
```

KubeadmControlPlane my-cluster

Unpause all created **non-dummy** resources...

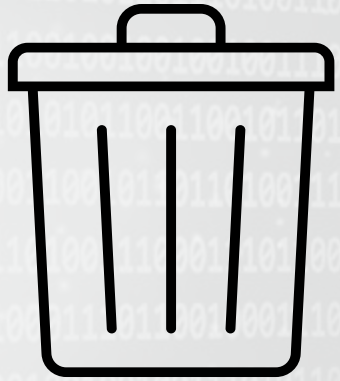
... and Cluster API will do the rest.

```
$ kubectl get nodes
NAME                                    STATUS     ROLES                  AGE   VERSION
my-cluster-control-plane-8cffd47b9-kvqcv NotReady   control-plane,master   23s   v1.21.3
my-cluster-kube-control-plane01         Ready      control-plane,master   36m   v1.21.3
my-cluster-md-cc8c675cb-lq8j8           Ready      <none>                 28m   v1.21.3
my-cluster-md-cc8c675cb-r2pgw           Ready      <none>                 28m   v1.21.3
my-cluster-md-cc8c675cb-wxhdz           Ready      <none>                 28m   v1.21.3
```

3 Migration to Cluster API - #3 KubeadmControlPlane

- Secret **dummy**
- KubeadmConfig **dummy**
- KubeadmControlPlane **dummy**
- Machine for Control Plane **dummy**

Delete all **dummy** leftovers, not needed anymore



3 Migration to Cluster API - 700 Clusters Migrated

1 Migrate the Infrastructure

2 Migrate the Worker Nodes

3 Migrate the Control Plane

```
$ kubectl get nodes
NAME                                      STATUS     ROLES                  AGE   VERSION
my-cluster-control-plane-8cffd47b9-kvqcv  Ready      control-plane,master   3m    v1.21.3
my-cluster-md-cc8c675cb-lq8j8             Ready      <none>                 31m   v1.21.3
my-cluster-md-cc8c675cb-r2pgw             Ready      <none>                 31m   v1.21.3
my-cluster-md-cc8c675cb-wxhdz             Ready      <none>                 31m   v1.21.3
```

4 Lessons Learned

Pod Disruption Budgets (PDB) allows to drain workload nodes safely without application downtime.

Users can secure their critical application by their own.

Set the Node Drain Timeout to zero seconds to never violate PDBs.

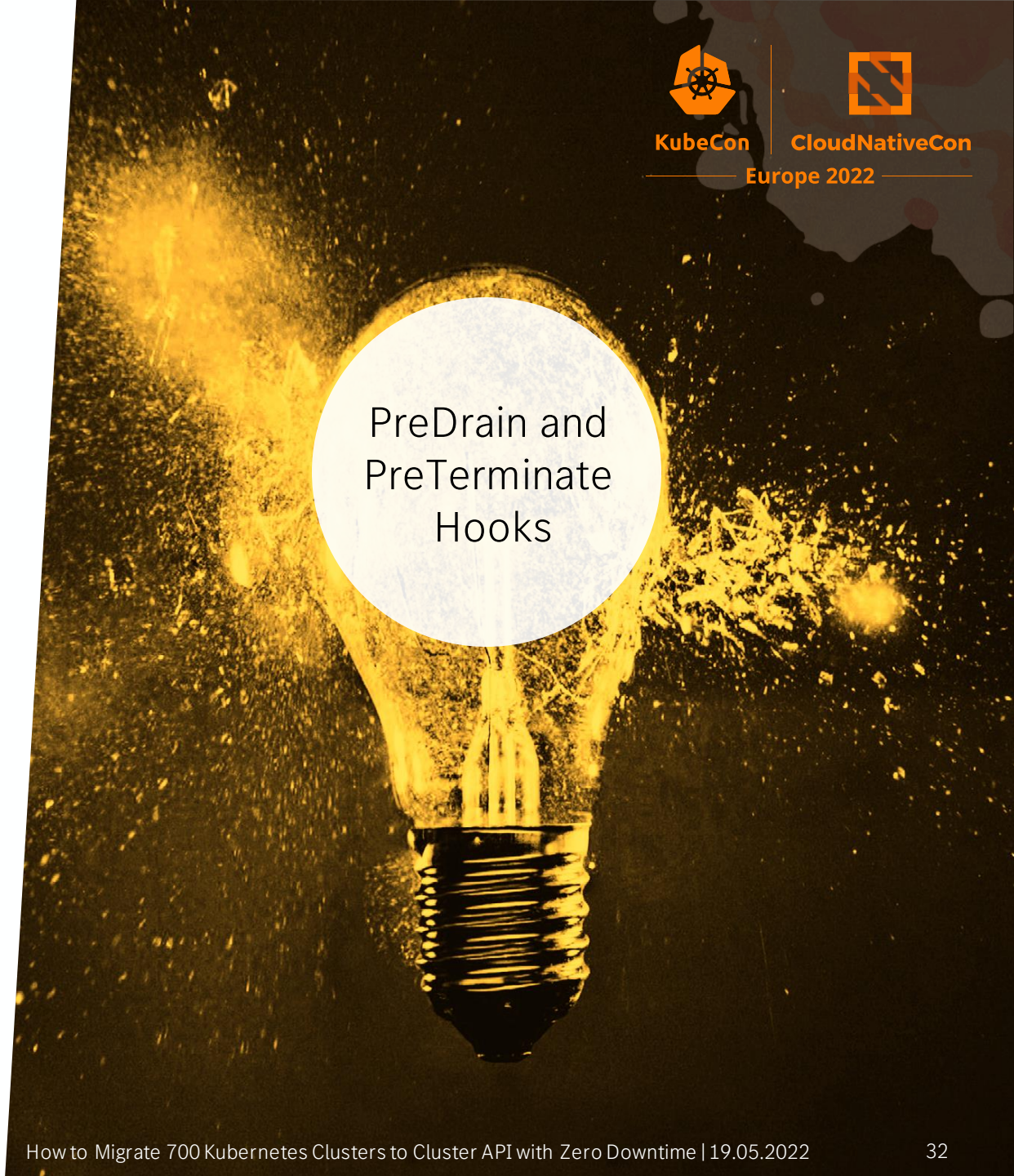
Users are notified if a Deployment is stuck due to a PDB.

4 Lessons Learned

Pre-Drain Annotation
pre-drain.delete.hook.machine.cluster.x-k8s.io

Custom controller prevents scheduling Pods on Nodes that will be deleted during update.

We do not enforce draining, to ensure that PDBs can block the drain.

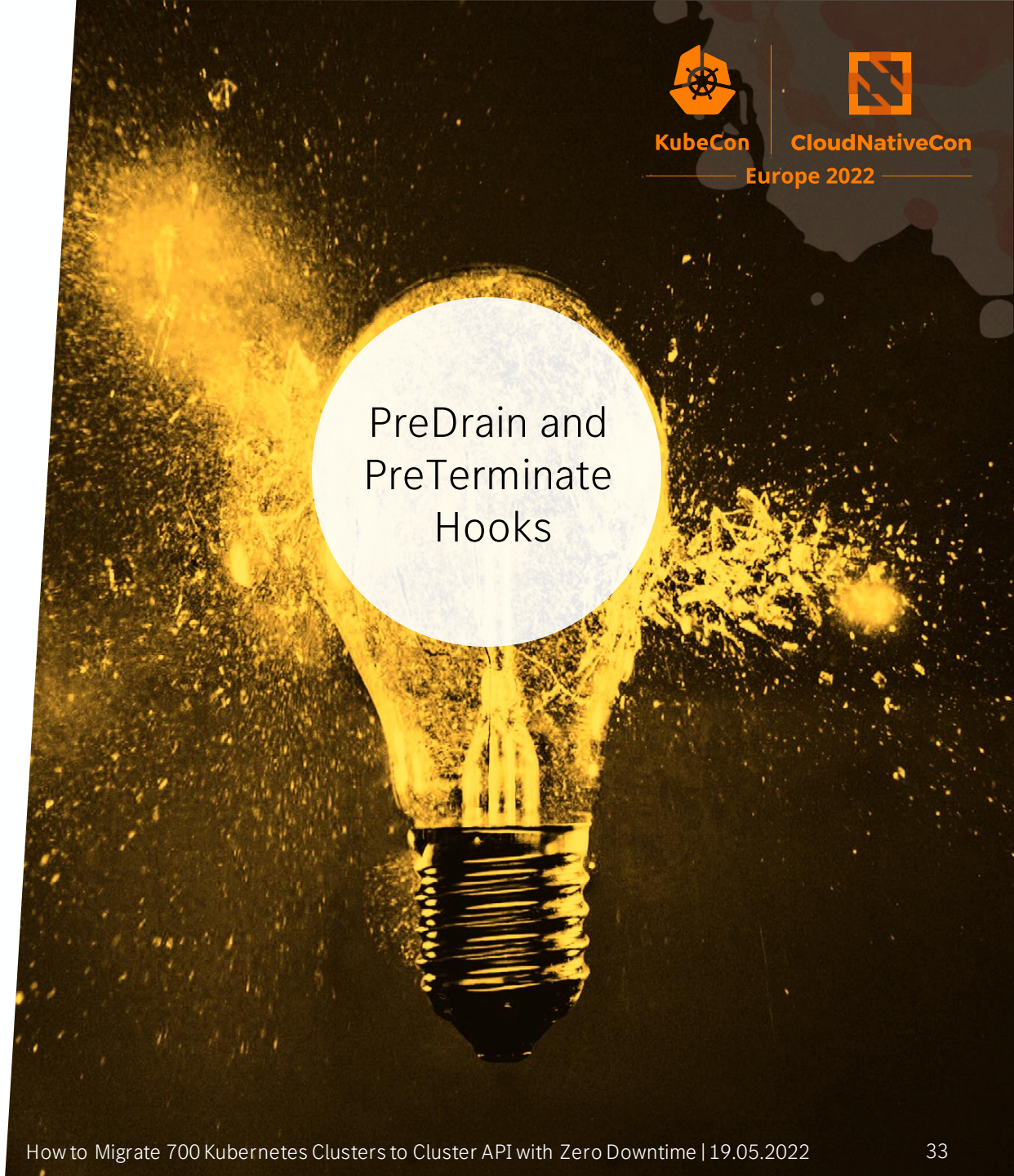


4 Lessons Learned

Pre-Terminate Annotation
pre-terminate.delete.hook.machine.cluster.x-k8s.io

Custom controller detaches remaining volumes, removes members from load balancer, …

Custom controller patches the attached volumes in the Node status, otherwise Cluster API controller is stuck between draining and terminating.



4 Lessons Learned

Prevent snowflakes, all clusters must look the same. In this case fully automatic nightly build testing can simulate a real cluster migration.

Nightly builds of:

- Migration from legacy to Cluster API managed clusters
- Creation of new clusters via Cluster API

4 Lessons Learned

Prevent forceful deletion of Cluster API resources via kubectl by removing finalizers.

In exceptional cases, the controllers may not notice the deletion.

Workaround by restarting Pods.

Fixed by controller-runtime PR #1640.

5 Next Steps

Replace Ansible by Flux

Migrate Add-On management from legacy provisioning to Flux

Adopt New Cluster API Functionality

ClusterClass managed topologies

MachineHealthCheck

Integrate Runtime SDK

Contribute Cluster API Metrics Exporter

Cluster API state metrics for observability of Cluster API related objects

Public Clouds

Provide fully managed Kubernetes on public clouds

Next integration: AWS

Same experience as on-premises

6 Get Involved

GitHub

- mercedes-benz
- kubernetes-sigs/cluster-api
- kubernetes-sigs/cluster-api-provider-openstack

Slack

- #cluster-api
- #cluster-api-openstack

Let's Drive Innovation!

KubeCon CloudNativeCon Europe 2022

Thank you!

Q&A

Mercedes-Benz Tech Innovation GmbH
Wilhelm-Runge-Street 11 | 89081 Ulm
techinnovation@mercedes-benz.com
www.mercedes-benz-techinnovation.com

Mercedes-Benz Tech Innovation GmbH
Wilhelm-Runge-Street 11, 89081 Ulm
Phone +49 731 505-06 / techinnovation@mercedes-benz.com / www.mercedes-benz-techinnovation.com
Domicile and Court Registry: Ulm / HRB-No.: 3844 / Management: Daniel Geisel (CEO), Isabelle Krautwald

Mercedes-Benz Tech Innovation