OpenSlice
by ETSI

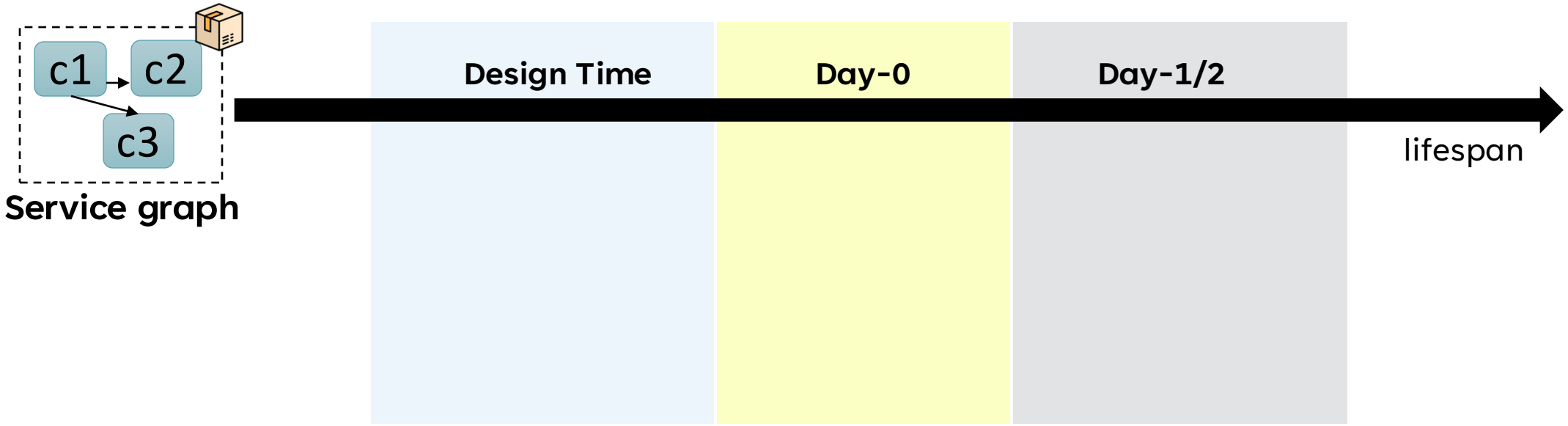
Zero-touch end-user service update mechanisms via an open orchestration platform

Anastasios Poimenidis, Vasilios Katopodis, Labros Papadopoulos, Alexandros Valantasis, Georgios P. Katsikas

Joint ETSI SDG Ecosystem Day

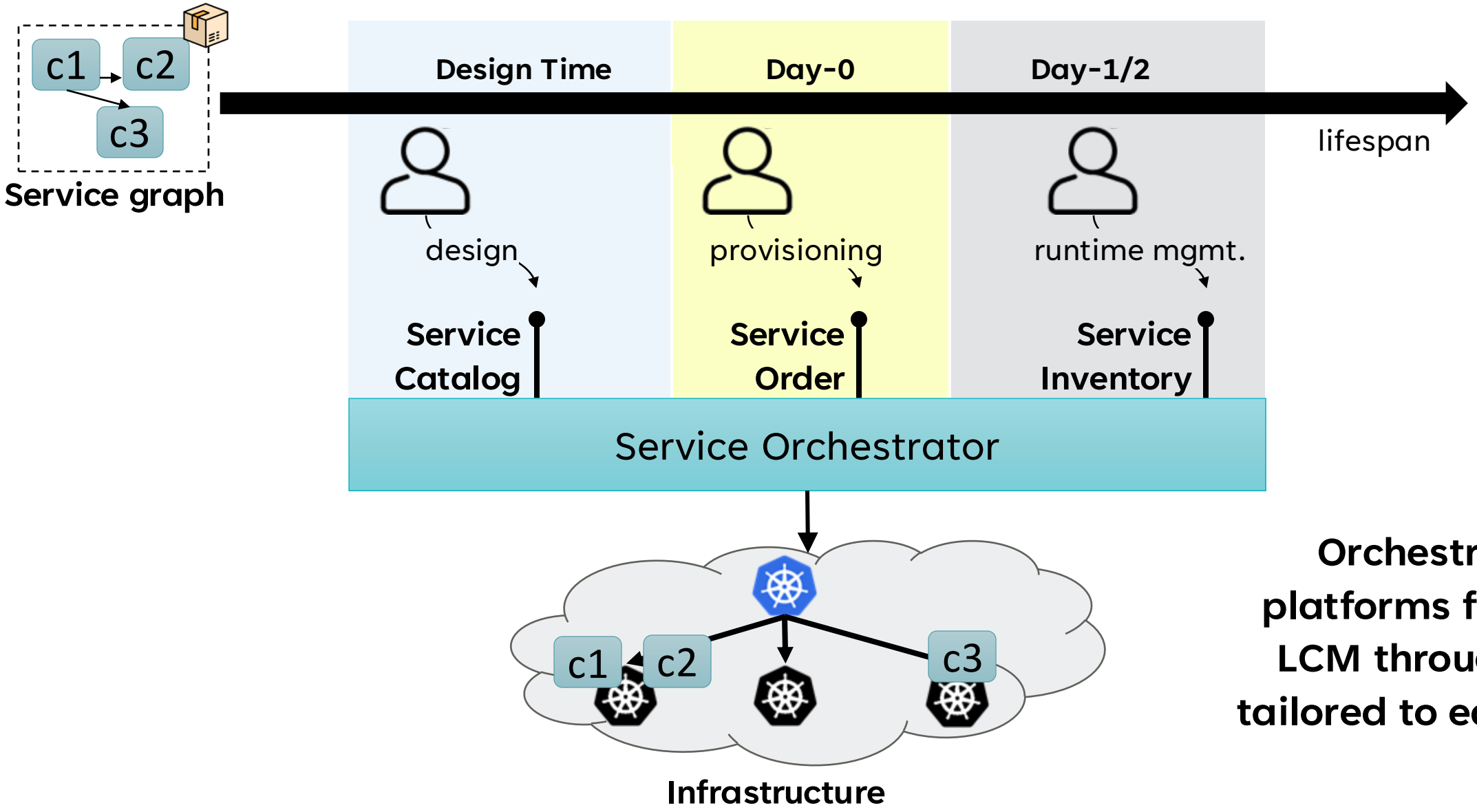
26/11/2025

Introduction



Service lifecycle undergoes many stages

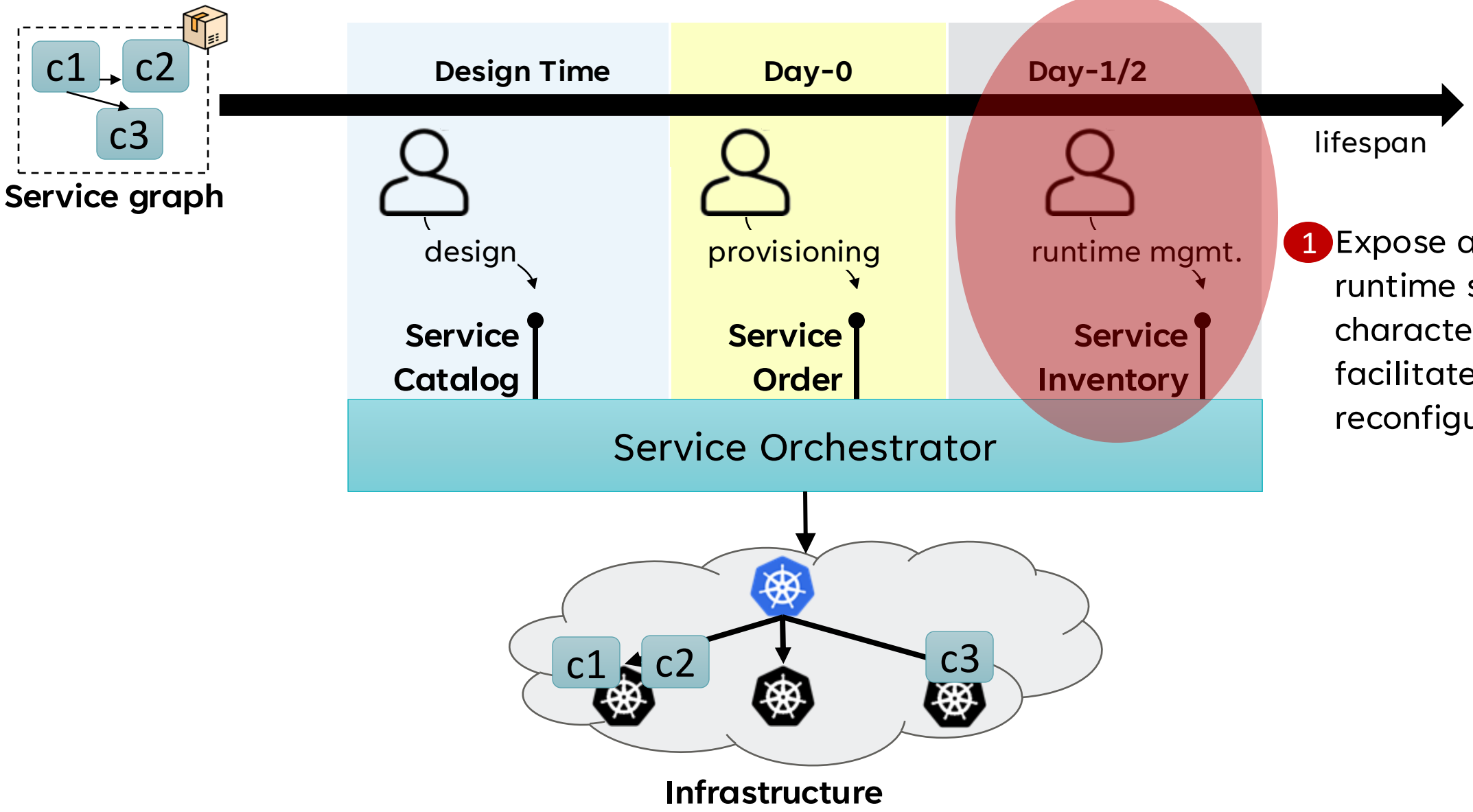
Introduction



Orchestration platforms facilitate LCM through APIs tailored to each stage

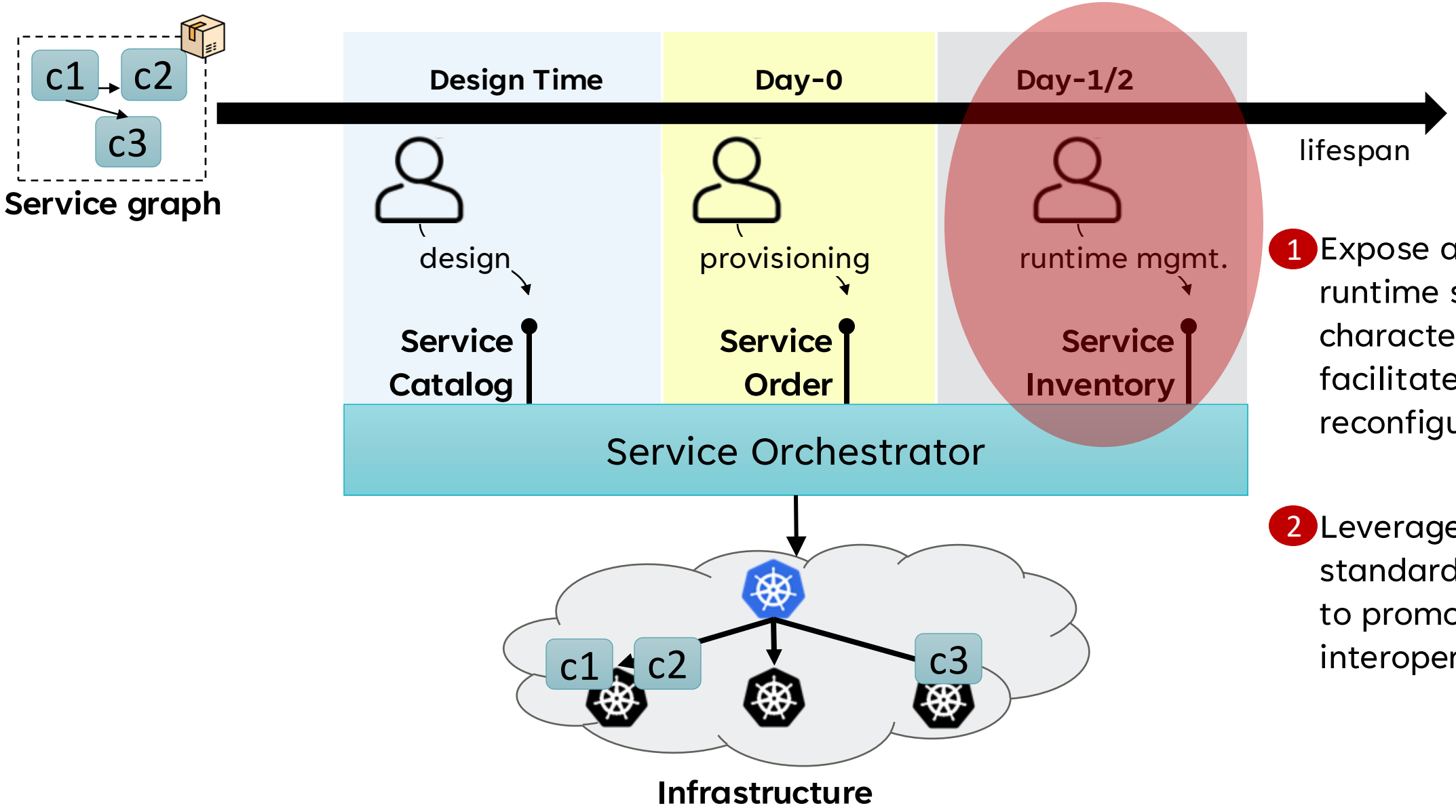
Objective

Objective



1 Expose a rich set of runtime service characteristics to facilitate service reconfiguration

Objective

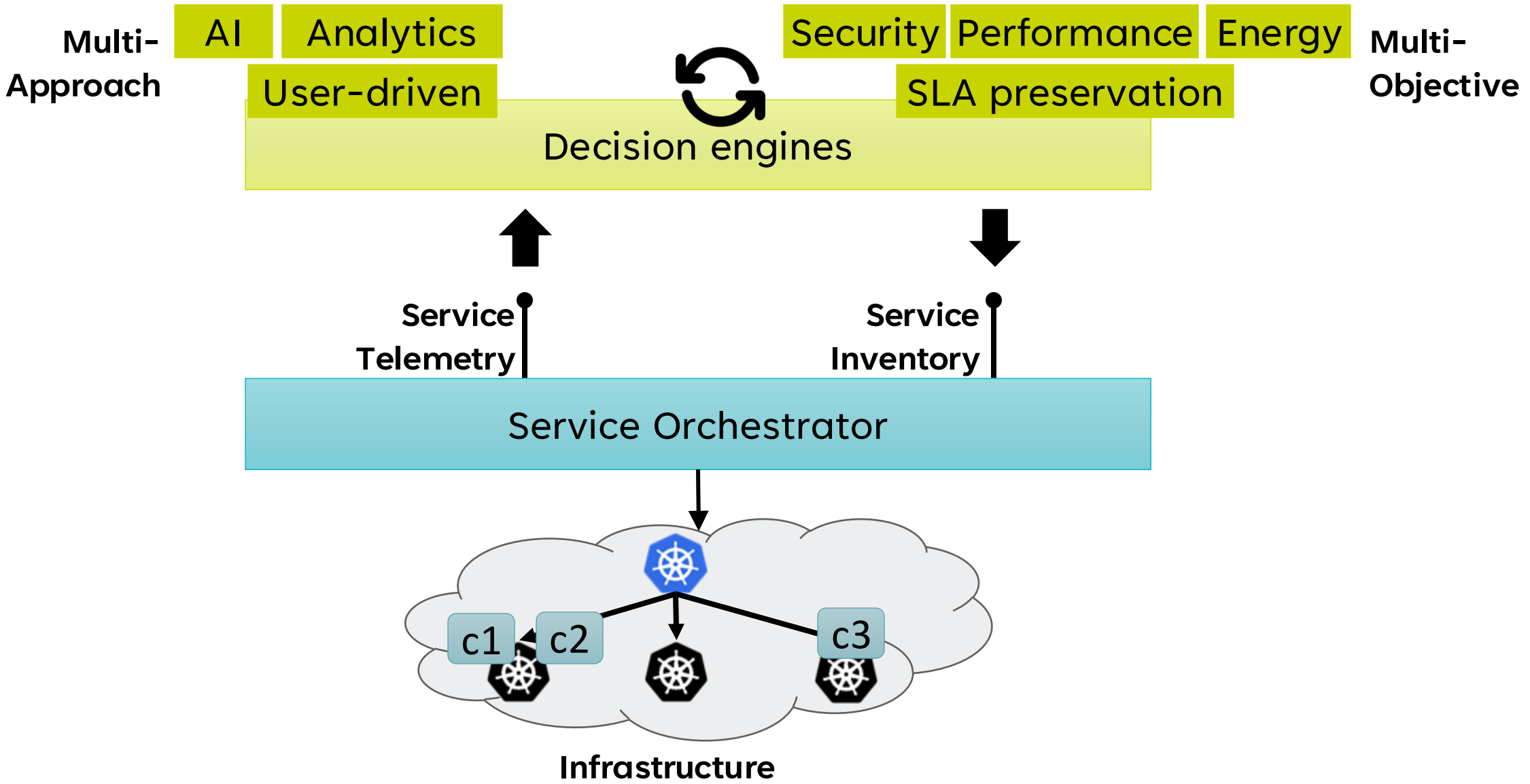


1. Expose a rich set of runtime service characteristics to facilitate service reconfiguration

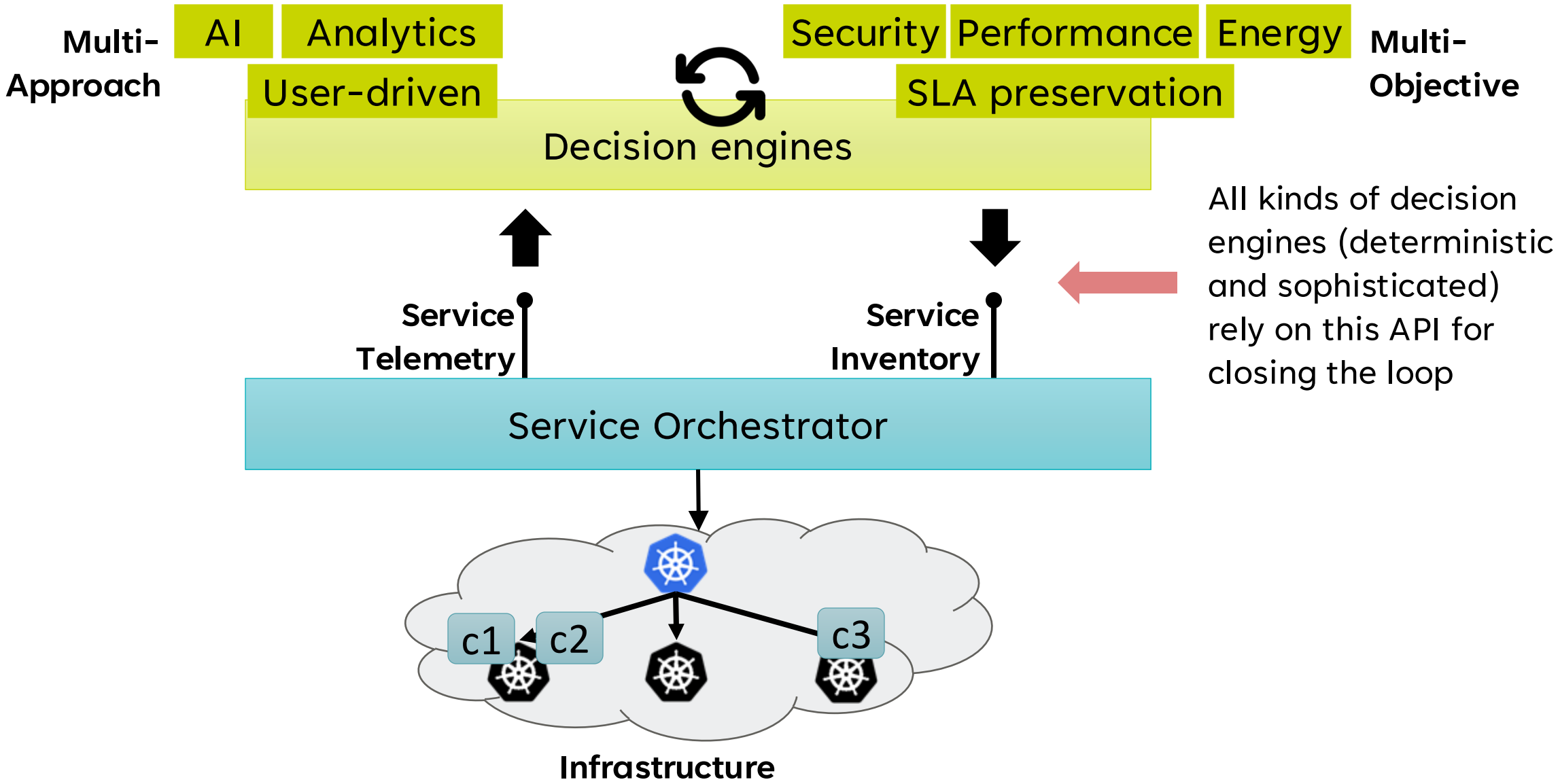
2. Leverage standardized APIs to promote interoperability

Relevance

Relevance


Multi-
Approach
AI
Analytics
User-driven
Security
Performance
Energy
SLA preservation
Multi-
Objective
Decision engines
Service
Telemetry
Service
Inventory
Service Orchestrator
c1
c2
c3
Infrastructure

Relevance



All kinds of decision engines (deterministic and sophisticated) rely on this API for closing the loop

Scenarios

Multi-purpose real-time service updates via an open orchestration platform

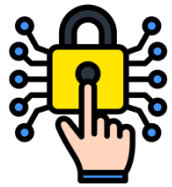
Scenarios

1. Real-time updates for service scaling

   # of replicas, CPU/memory limits, etc.

2. Real-time updates for service security

   Security patch roll-out



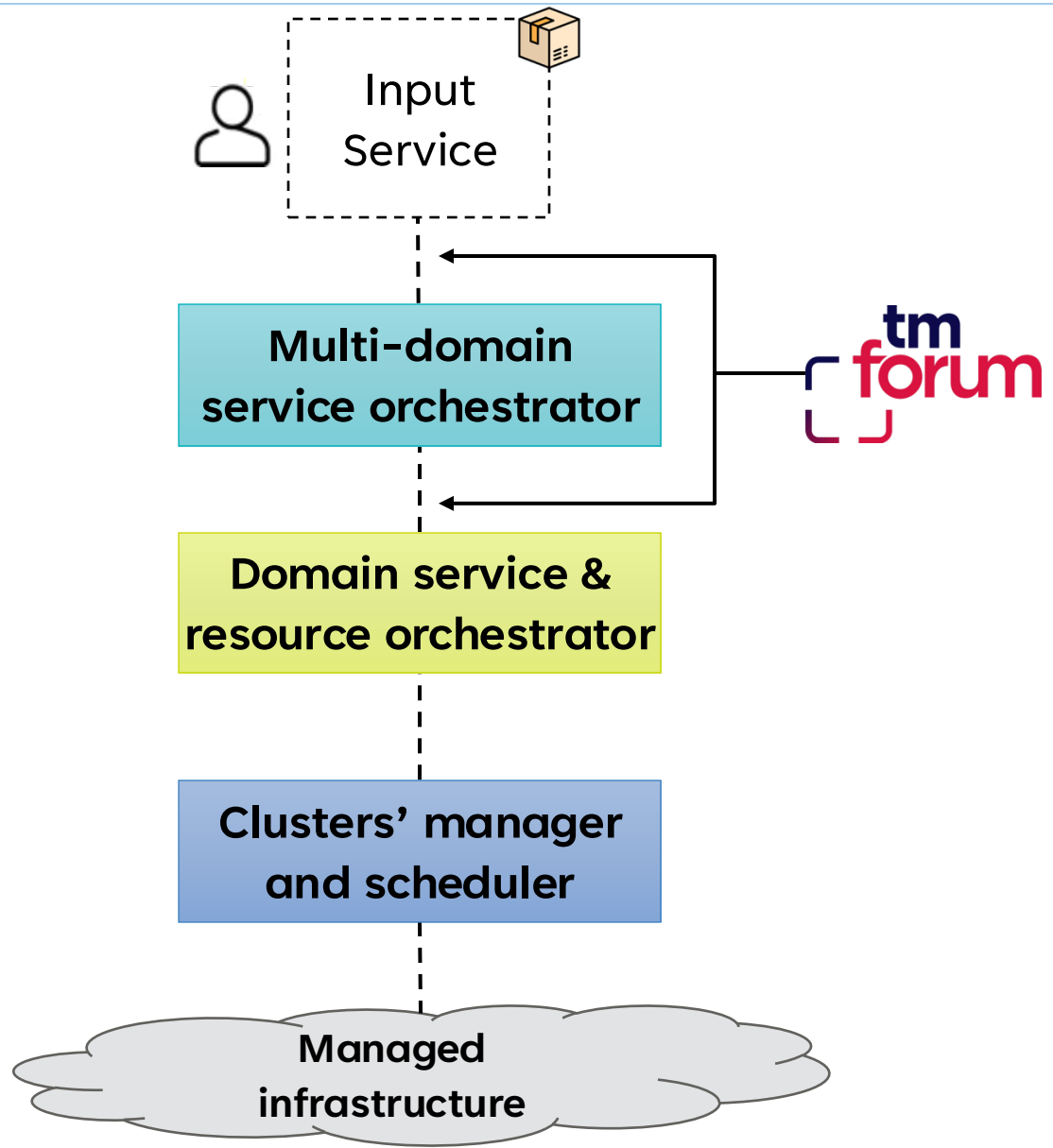
3. Real-time updates for service re-location

   Label-based placement of service components across multiple clusters

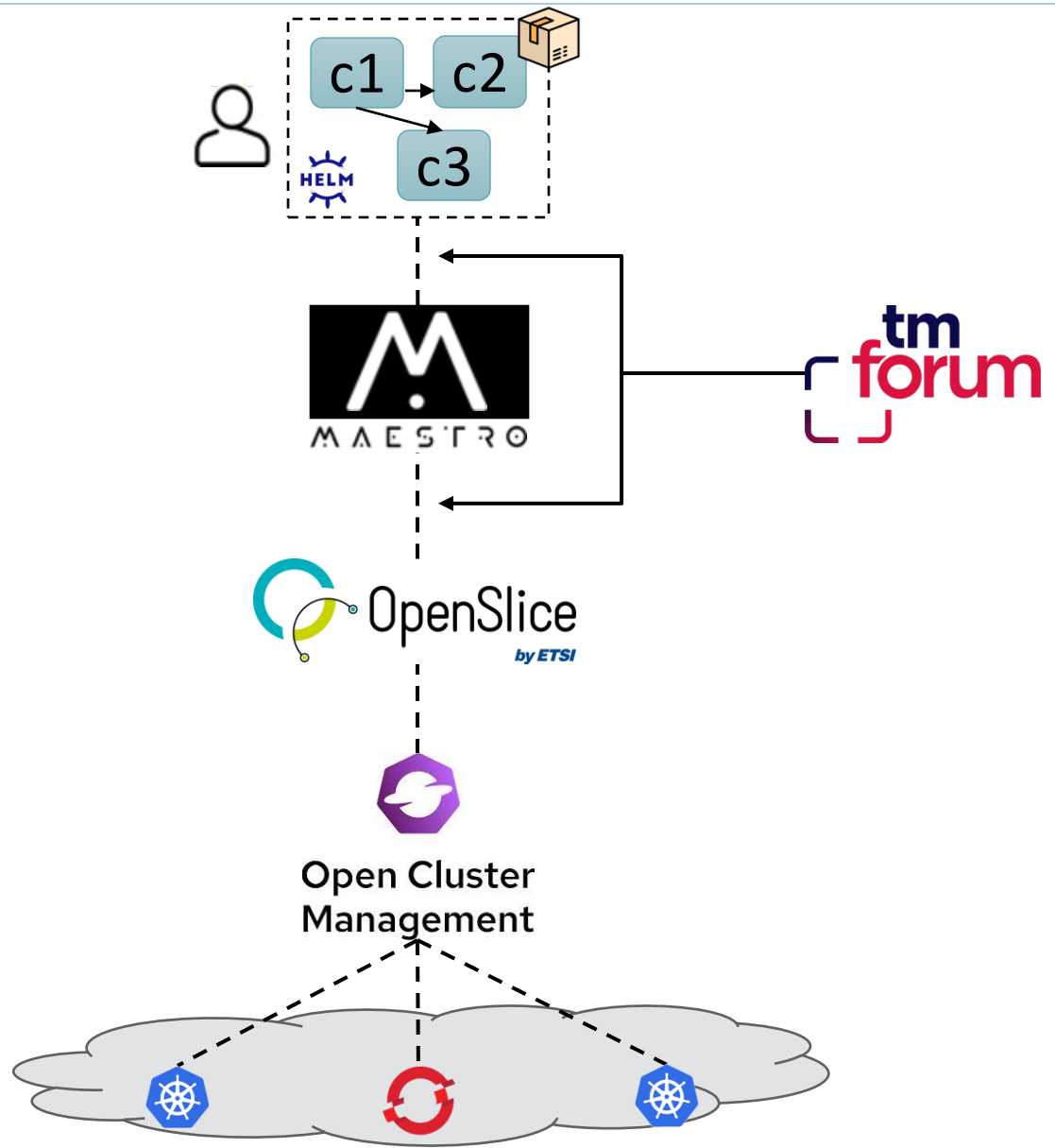
Setup

Platform setup and assumptions taken

Setup (abstract)



Setup (tech. specific)



Multi-component Service graph

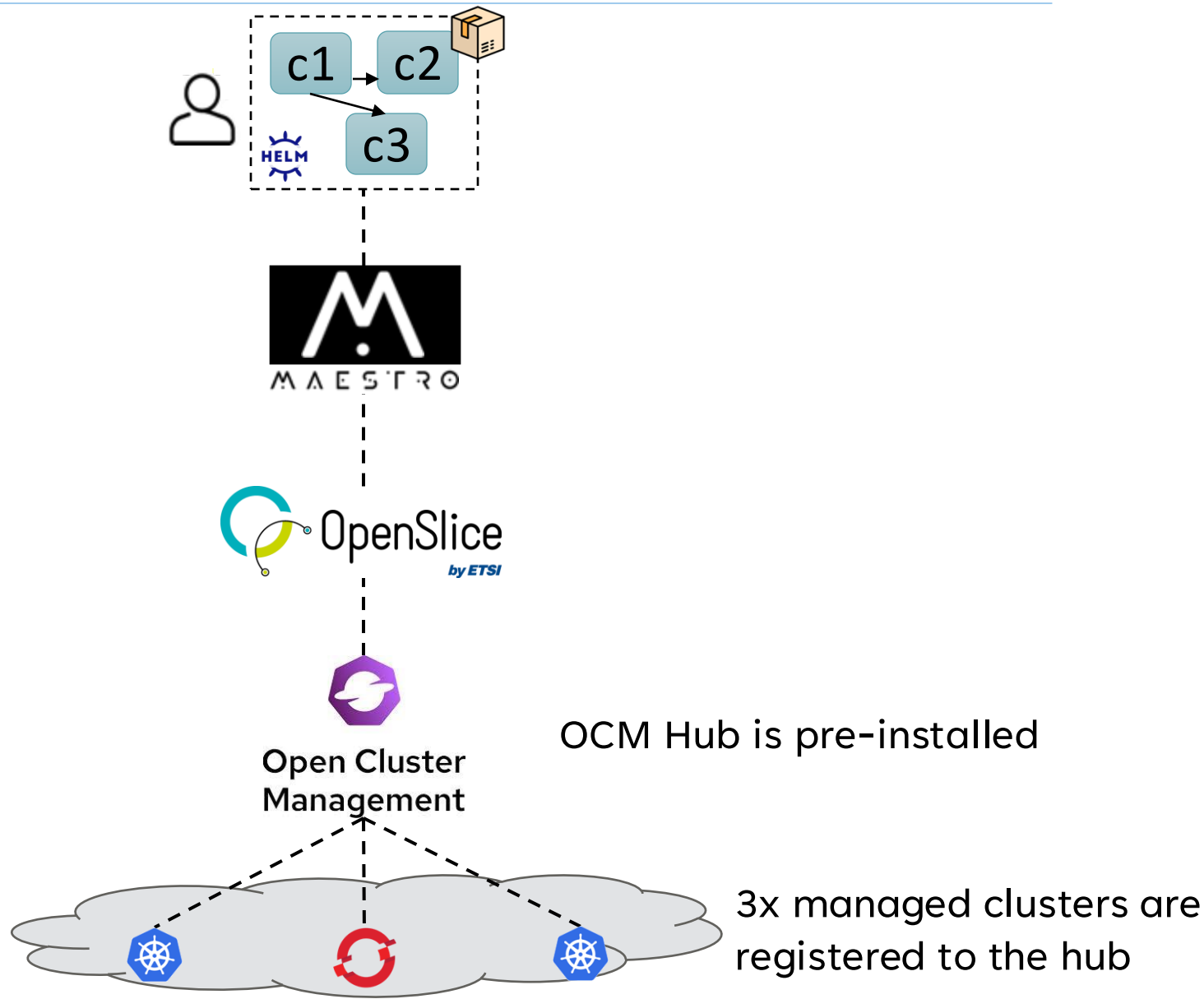
Multi-domain service orchestrator

Domain service & resource orchestrator

Clusters' manager and scheduler

Managed infrastructure

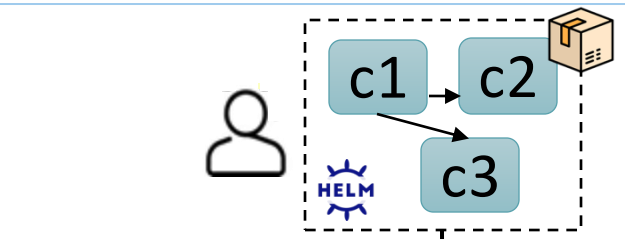
Assumptions before we start

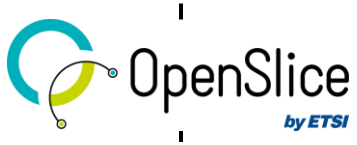


OCM Hub is pre-installed

3x managed clusters are registered to the hub

Assumptions before we start

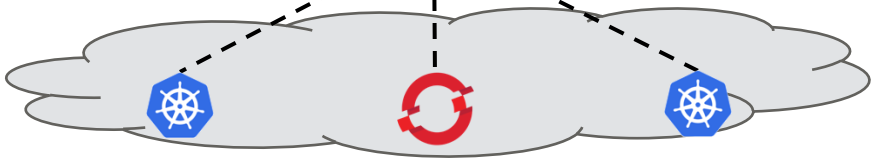




OpenSlice exposes the OCM Hub as a service

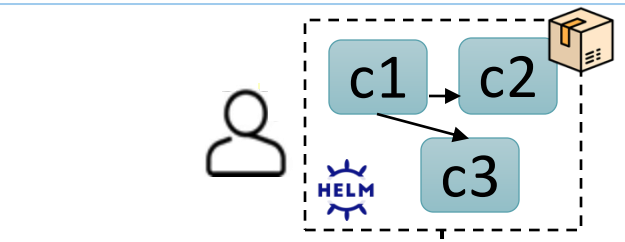
Open Cluster Management

OCM Hub is pre-installed

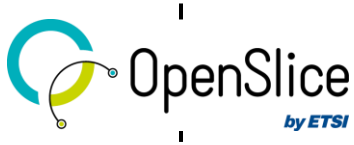


3x managed clusters are registered to the hub

Assumptions before we start



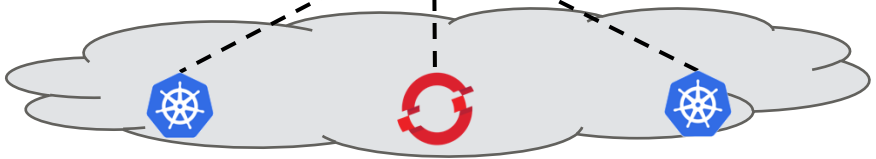
Maestro has already “peered” with OpenSlice



OpenSlice exposes the OCM Hub as a service

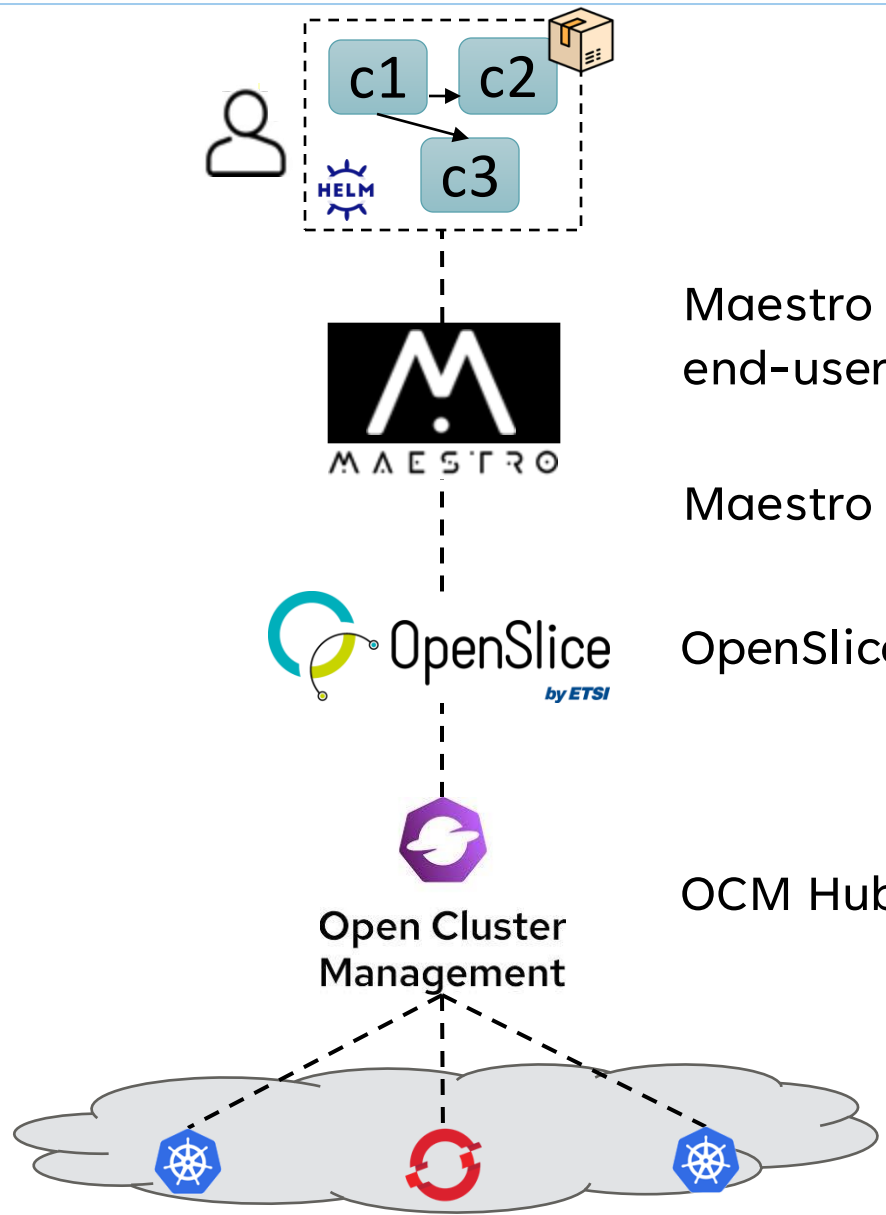
Open Cluster Management

OCM Hub is pre-installed



3x managed clusters are registered to the hub

Assumptions before we start



Maestro catalog contains both OCM and end-user service specifications

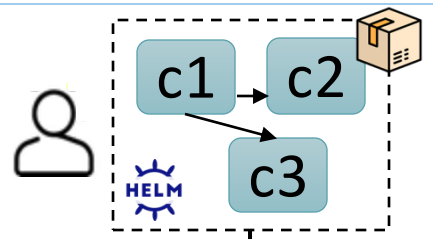
Maestro has already “peered” with OpenSlice

OpenSlice exposes the OCM Hub as a service

OCM Hub is pre-installed

3x managed clusters are registered to the hub

Assumptions before we start

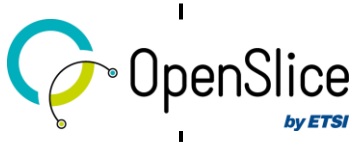


Input service packages are stored in a registry

Registry credentials shared with Maestro

Maestro catalog contains both OCM and end-user service specifications

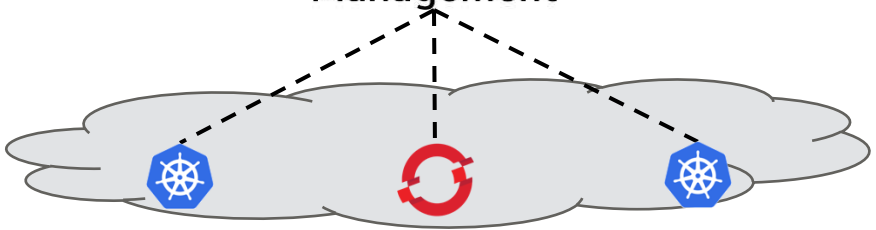
Maestro has already “peered” with OpenSlice



OpenSlice exposes the OCM Hub as a service

Open Cluster Management

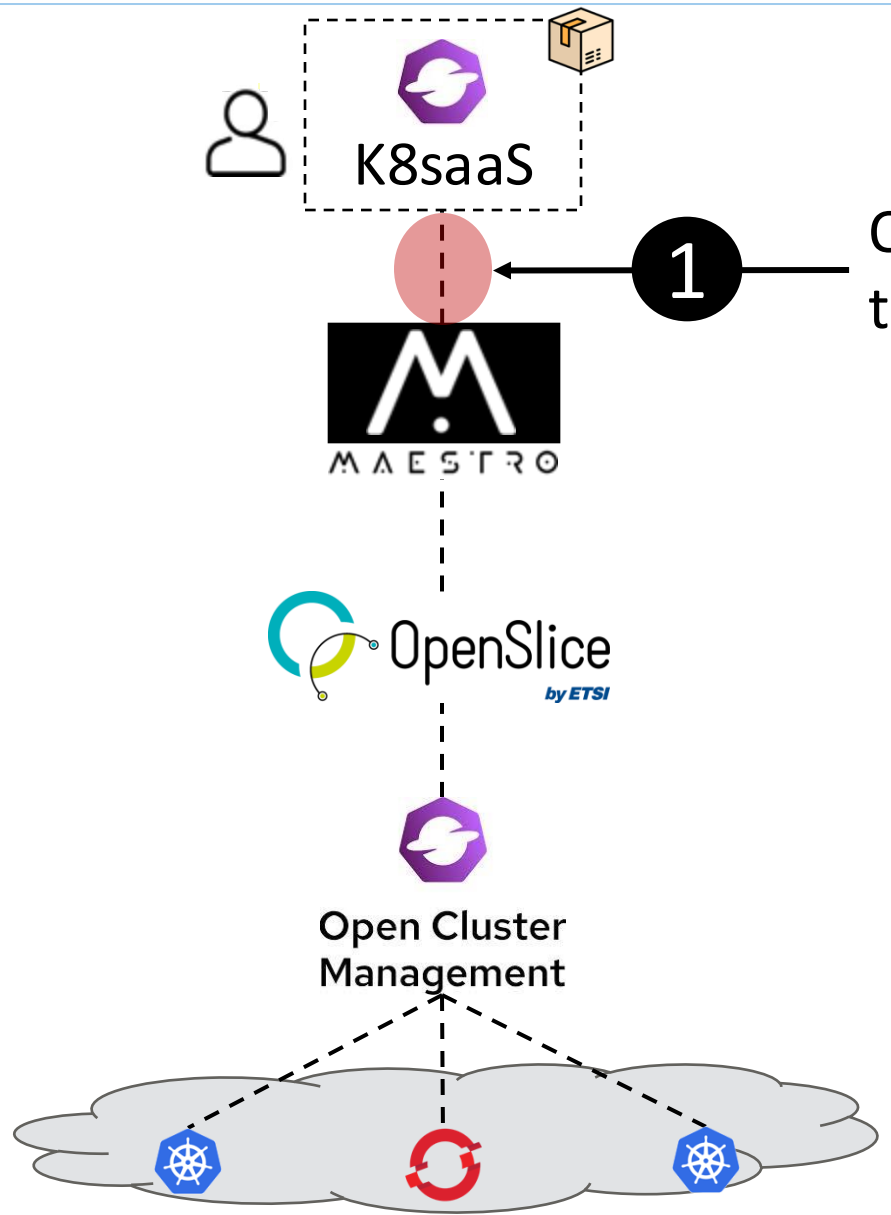
OCM Hub is pre-installed



3x managed clusters are registered to the hub

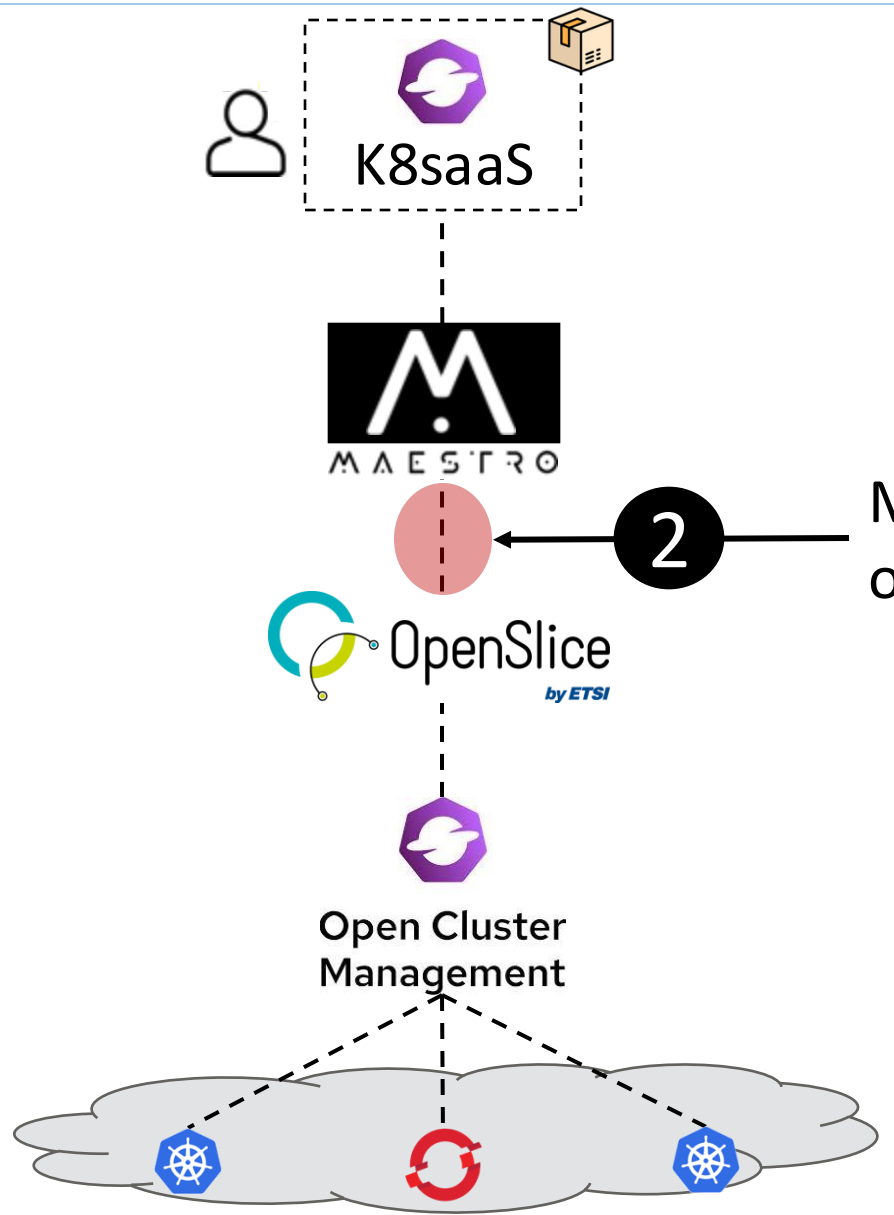
Service and Resource provisioning

Preparatory Steps – Resource allocation



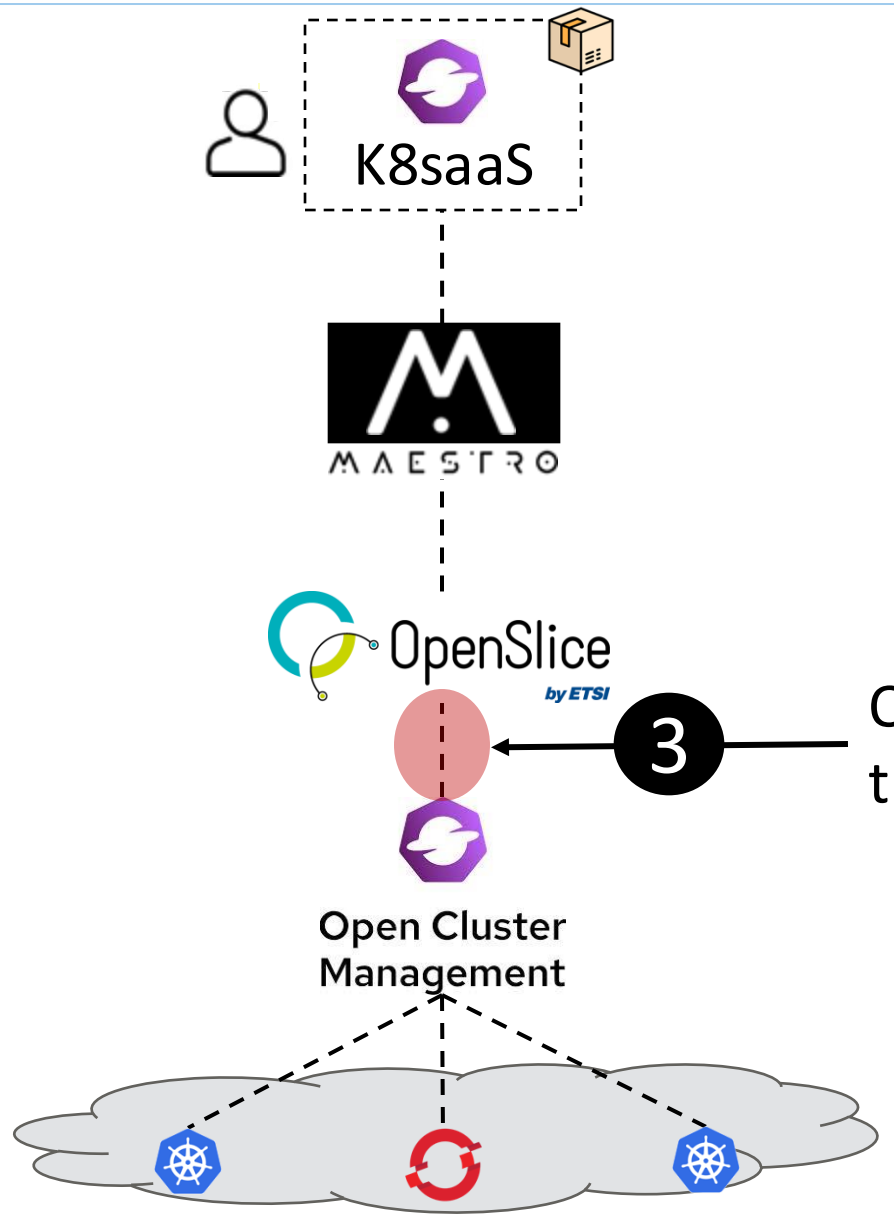
Order the OCM Hub service from the Maestro service catalog

Preparatory Steps – Resource allocation



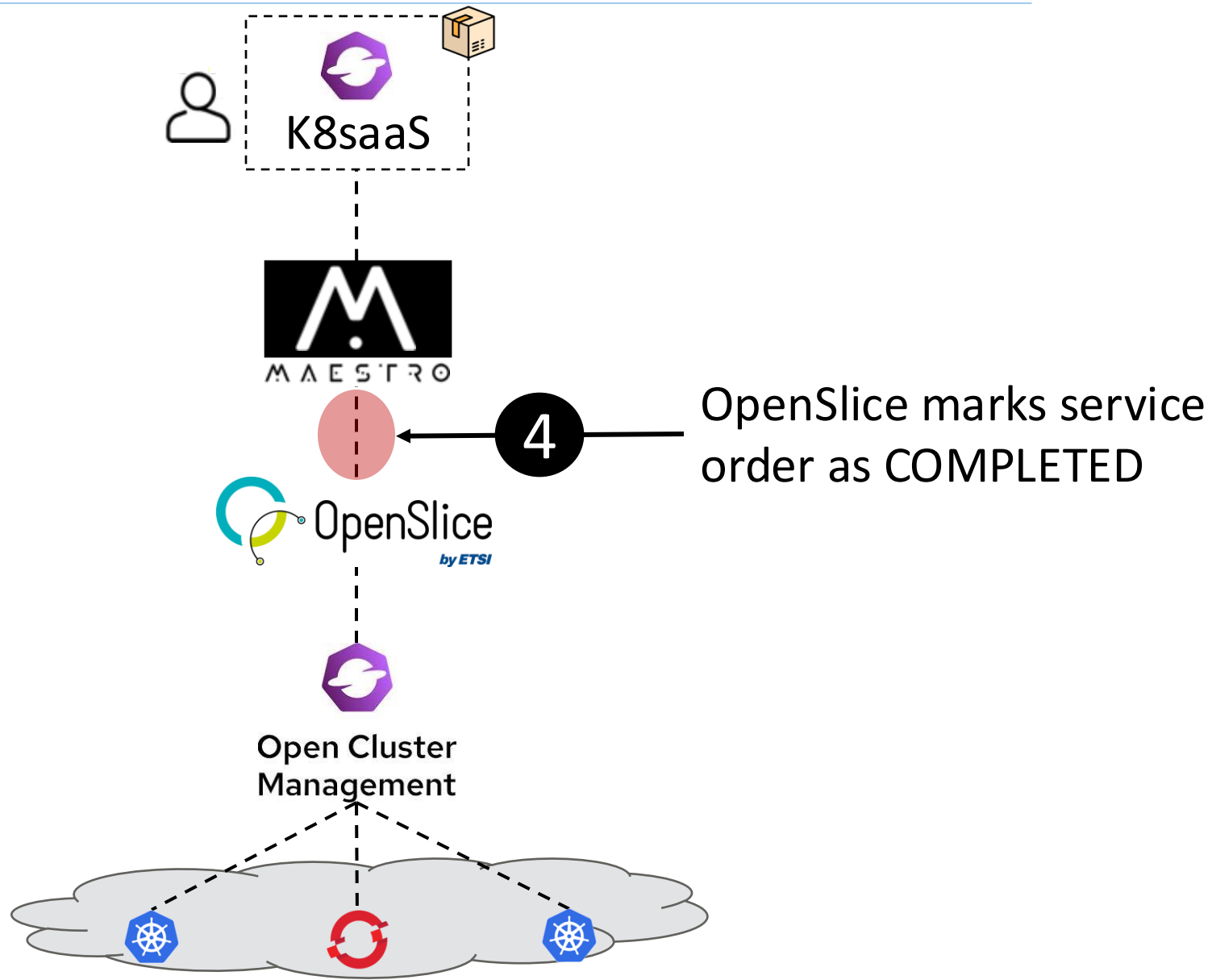
Maestro transfers service order to OpenSlice

Preparatory Steps – Resource allocation

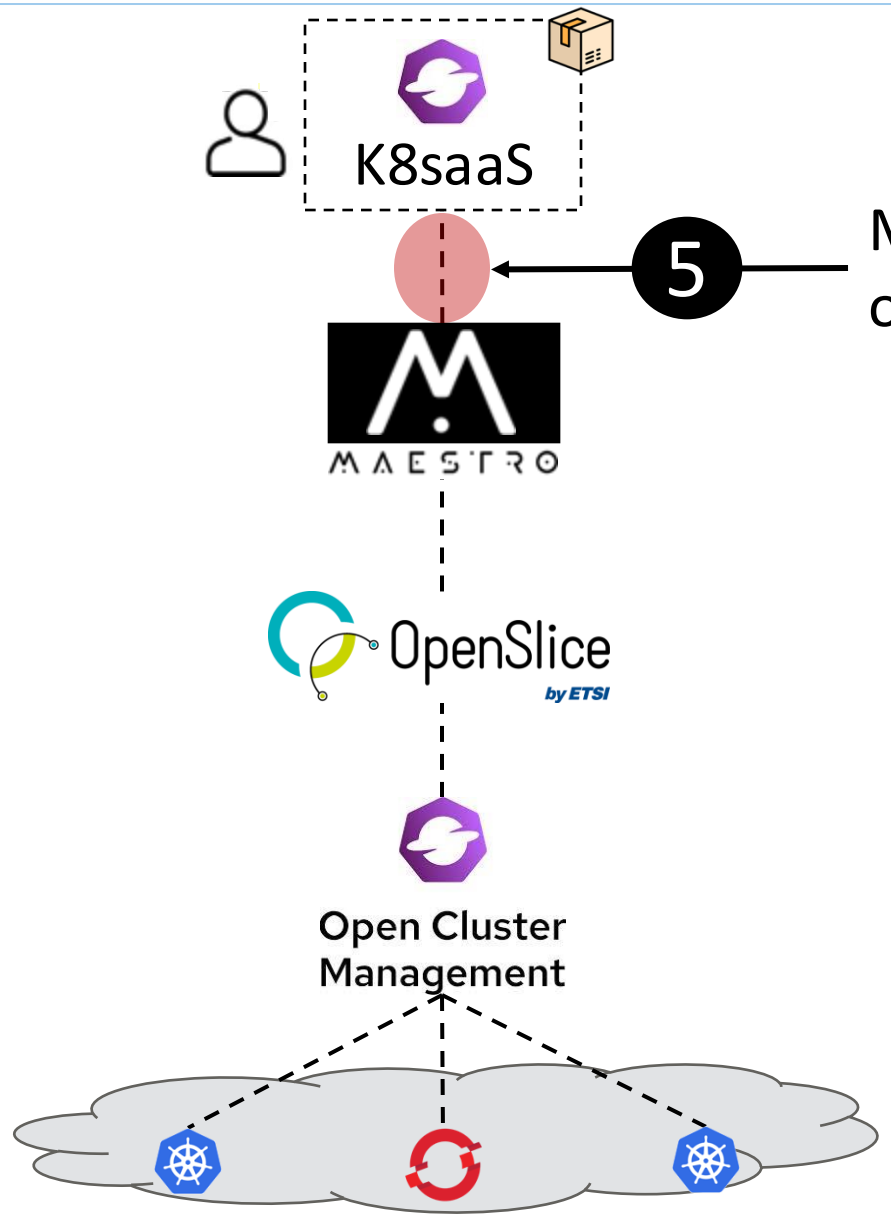


OpenSlice retrieves and exposes the OCM Hub's kubeconfig

Preparatory Steps – Resource allocation

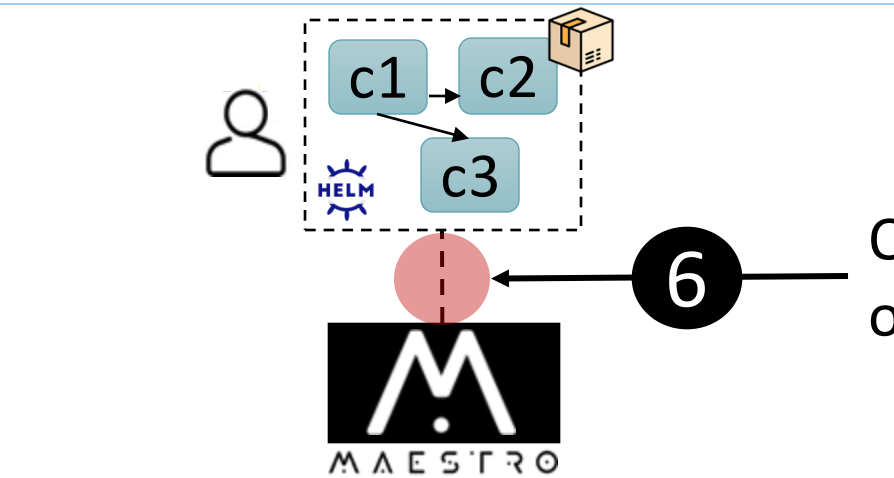


Preparatory Steps – Resource allocation



Maestro marks platform service order as COMPLETED

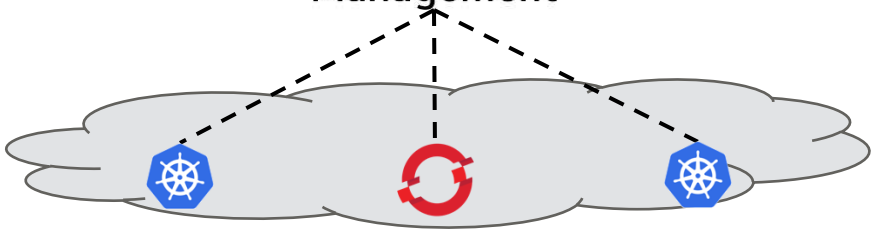
Preparatory Steps – Resource allocation



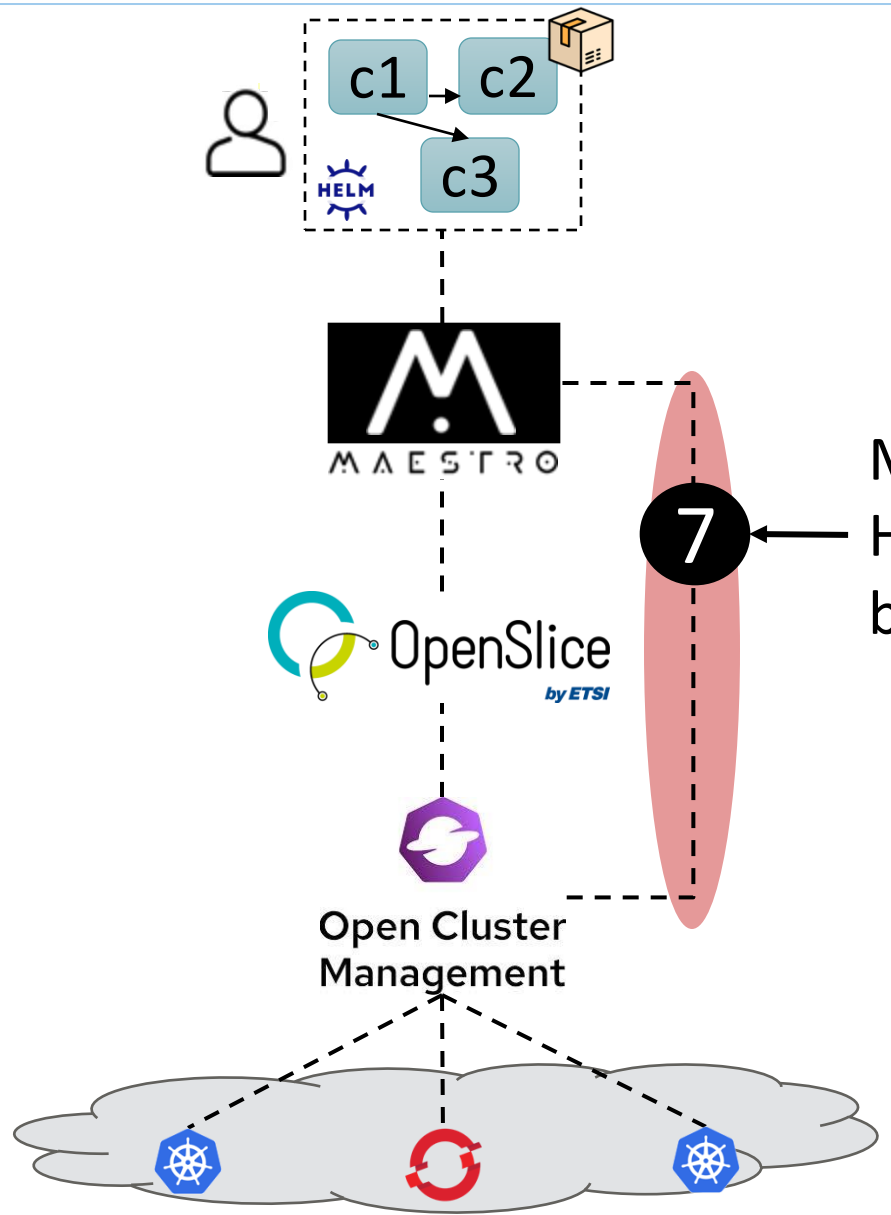
6

Order a 3-component service on top of OCM

Open Cluster Management



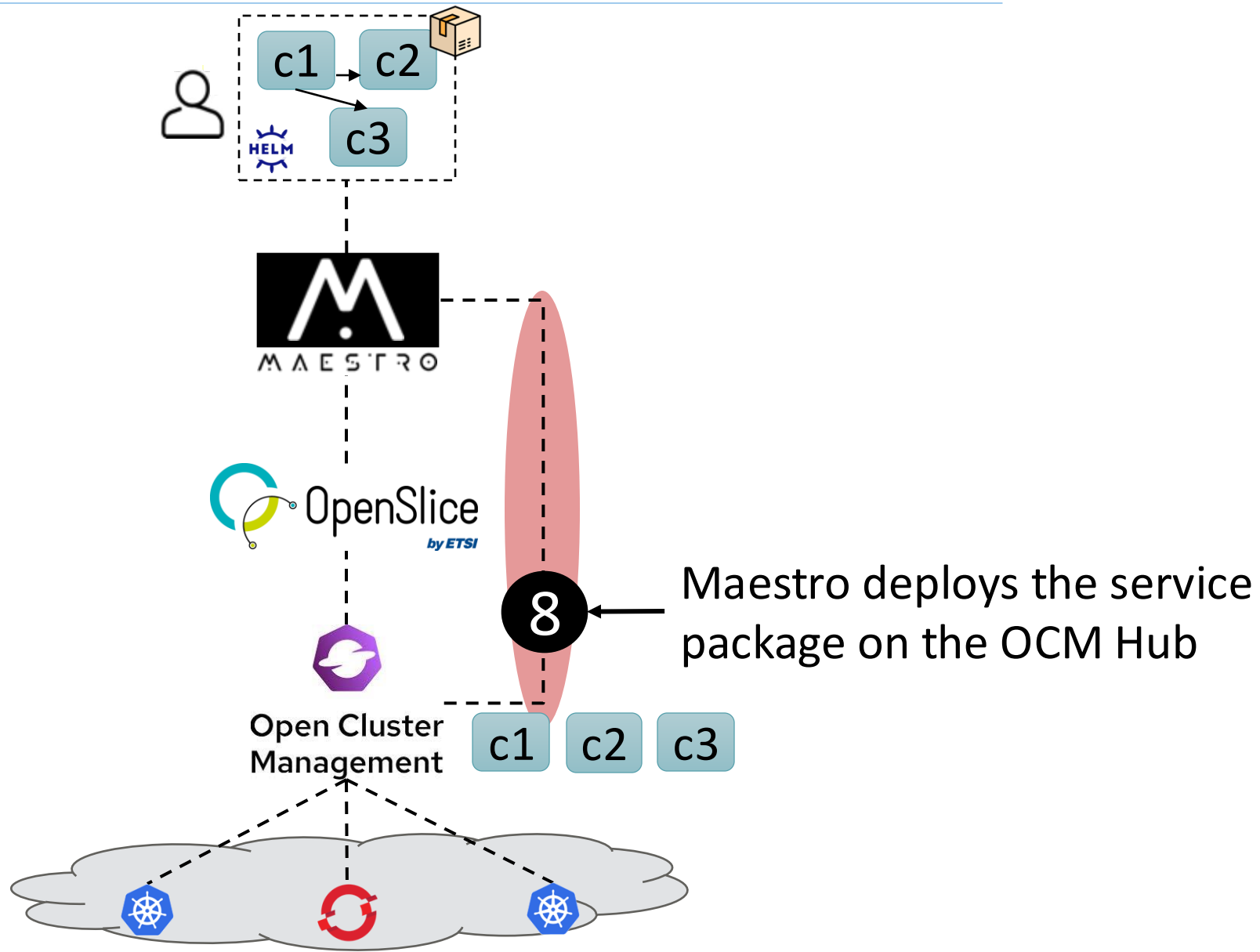
Preparatory Steps – Resource allocation



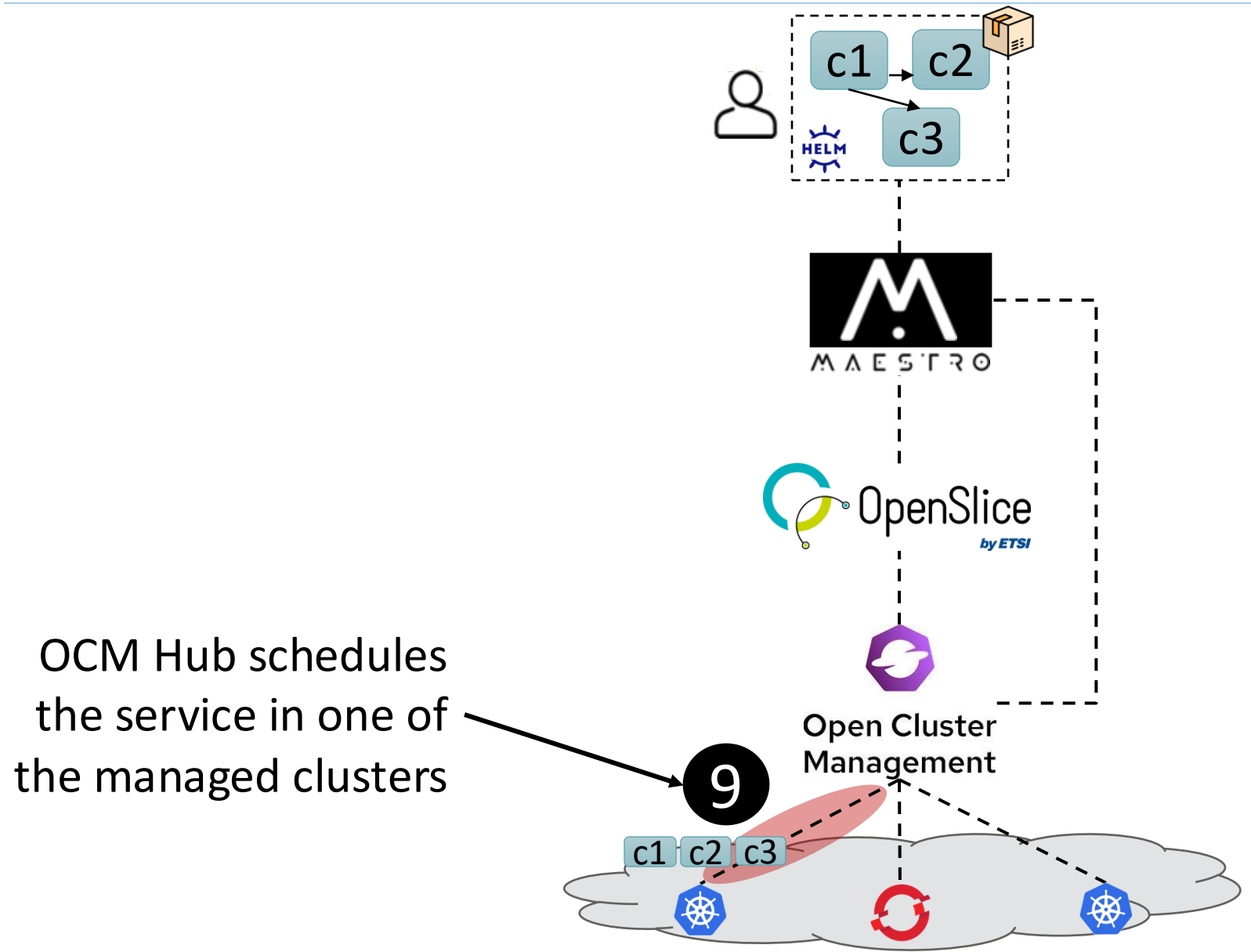
7

Maestro connects to the OCM Hub using the kubeconfig shared by OpenSlice

Preparatory Steps – Resource allocation

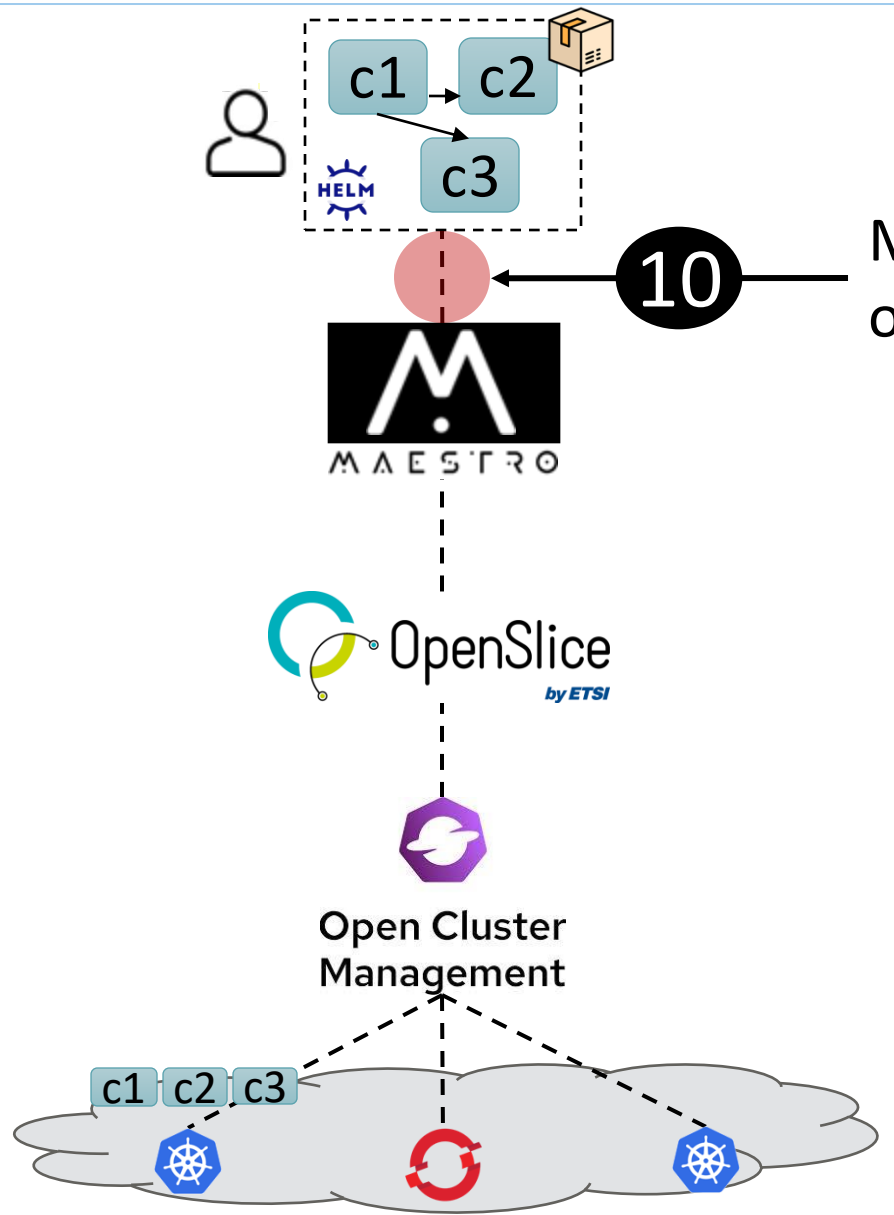


Preparatory Steps – Resource allocation



OCM Hub schedules the service in one of the managed clusters

Preparatory Steps – Resource allocation



Maestro marks end-user service order as COMPLETED

Preparatory Steps – Resource allocation demo

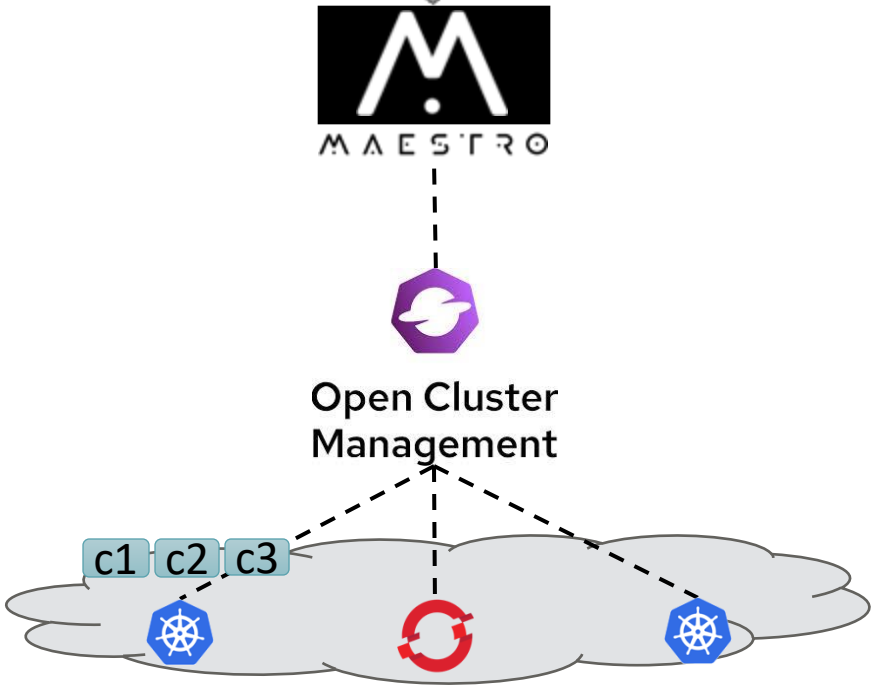
Scenario #1

Real-time updates for service scaling

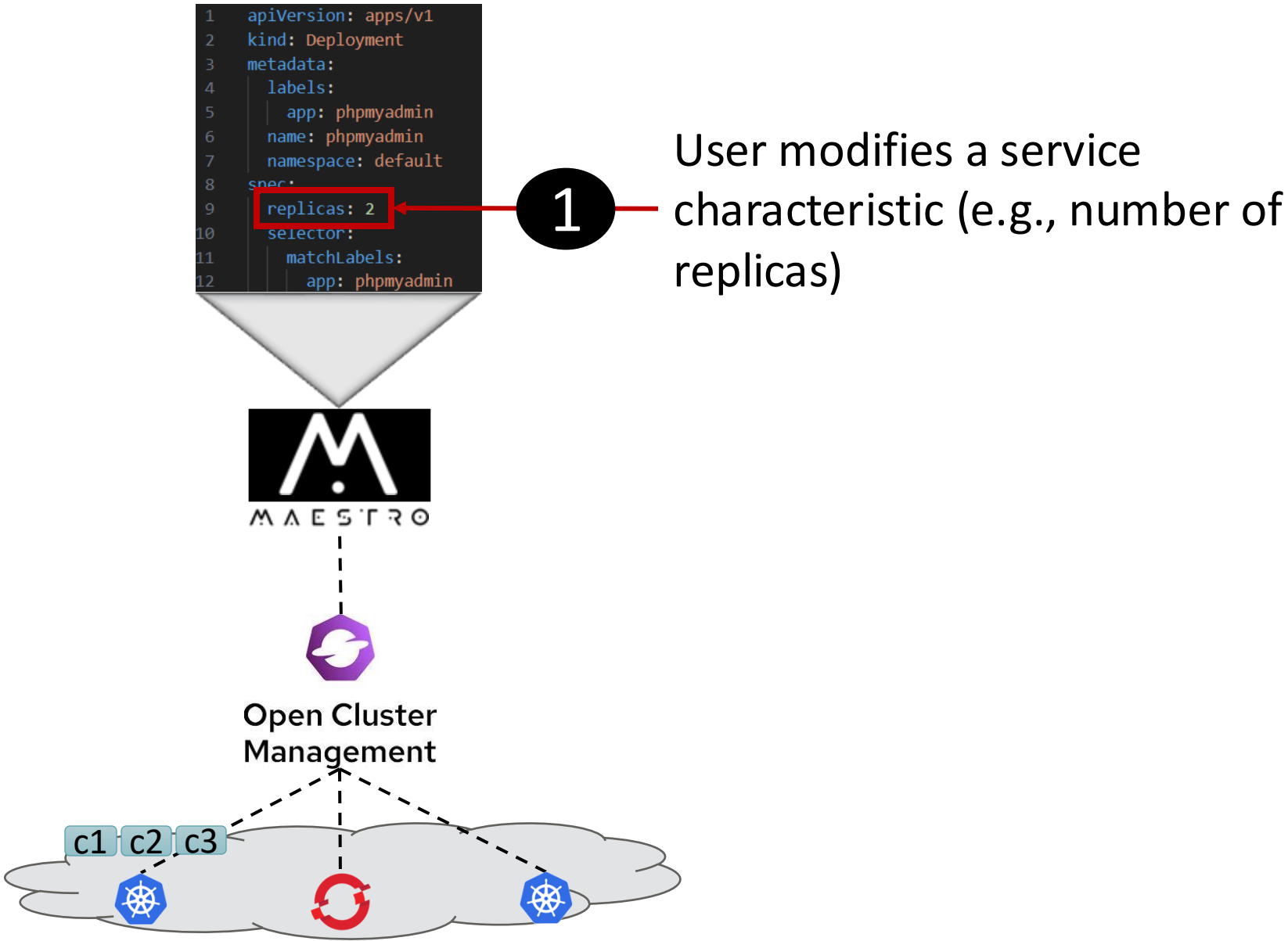
Scenario #1 – Service update for scaling

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: phpmyadmin
  name: phpmyadmin
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: phpmyadmin
```

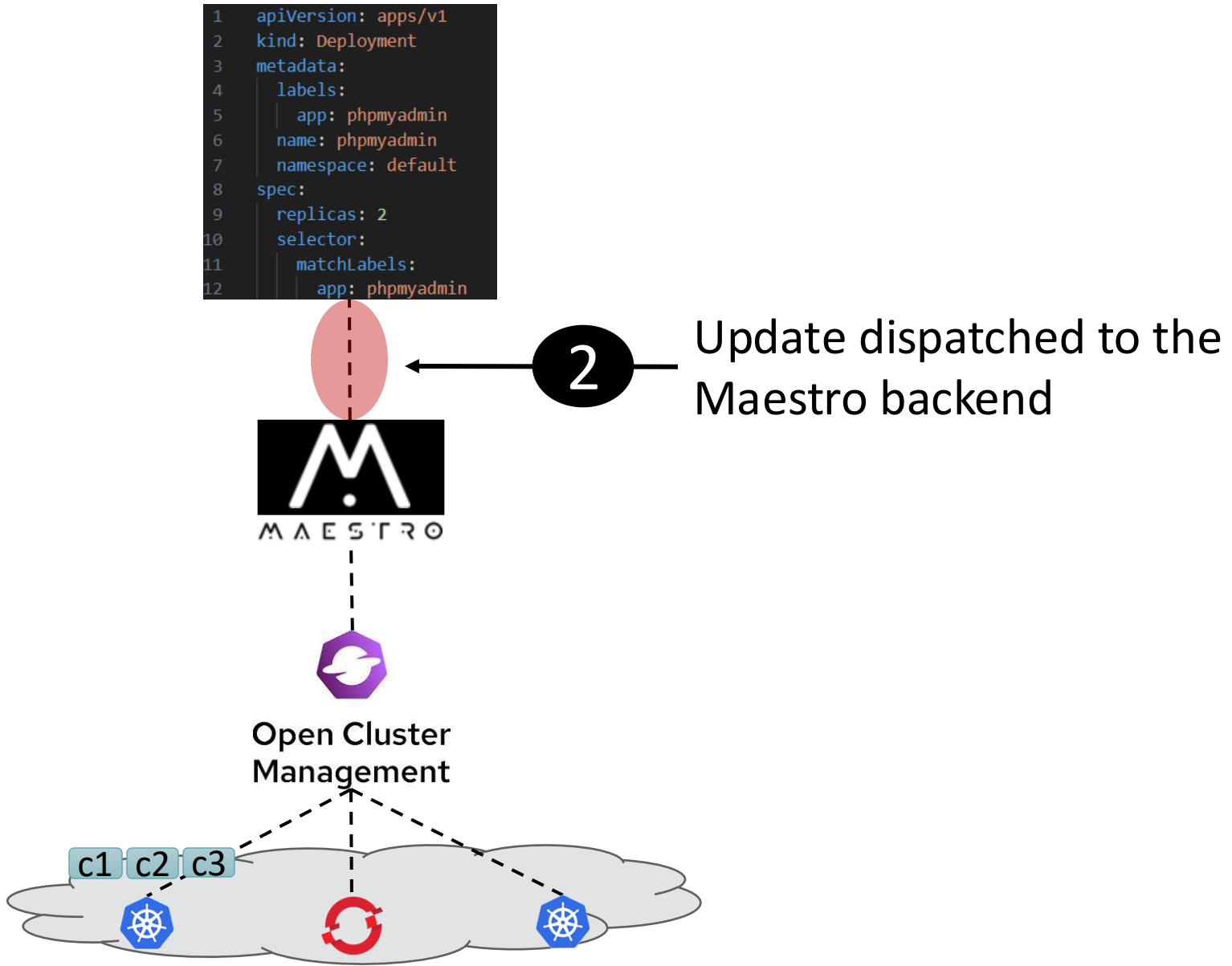
← Maestro exposes service configuration knobs via the portal



Scenario #1 – Service update for scaling



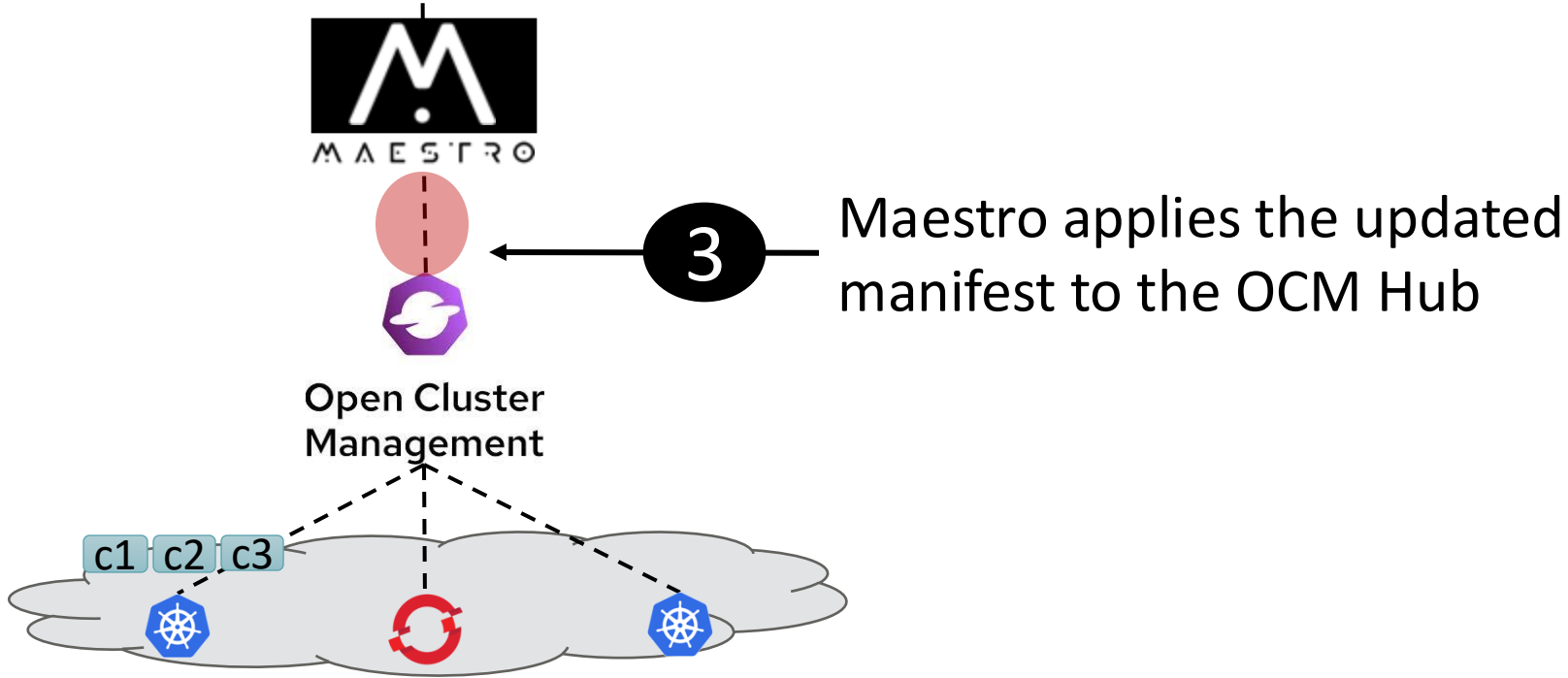
Scenario #1 – Service update for scaling



Scenario #1 – Service update for scaling

OpenSlice

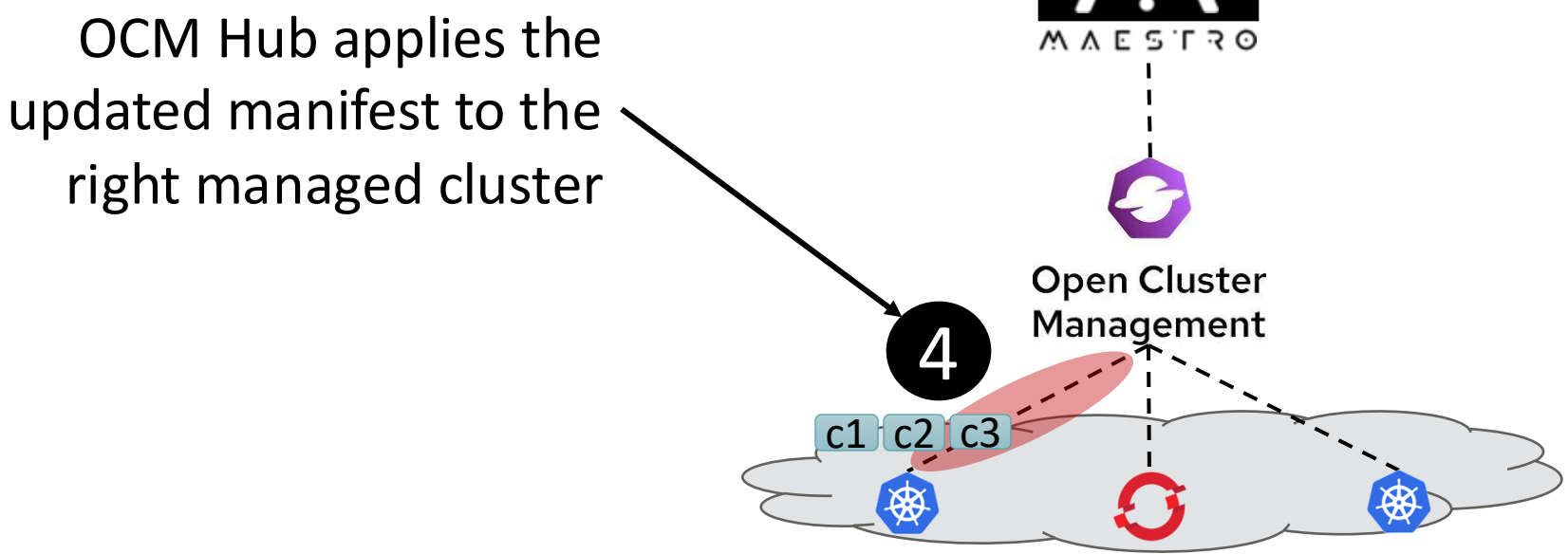
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: phpmyadmin
  name: phpmyadmin
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: phpmyadmin
```



Scenario #1 – Service update for scaling

OpenSlice

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: phpmyadmin
  name: phpmyadmin
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: phpmyadmin
```



OCM Hub applies the updated manifest to the right managed cluster

Scenario #1 – Service update for scaling

OpenSlice

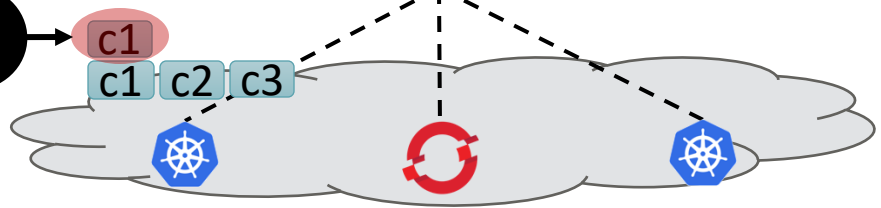
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: phpmyadmin
  name: phpmyadmin
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: phpmyadmin
```

MAESTRO

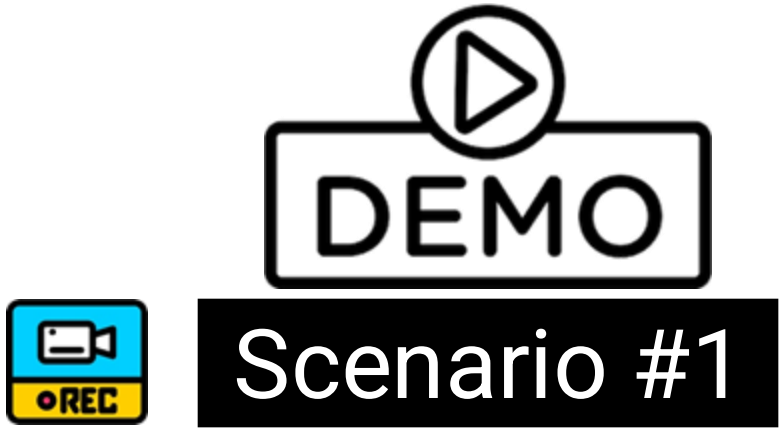
Open Cluster Management

A replica instance appears

5



Scenario #1 – Demo

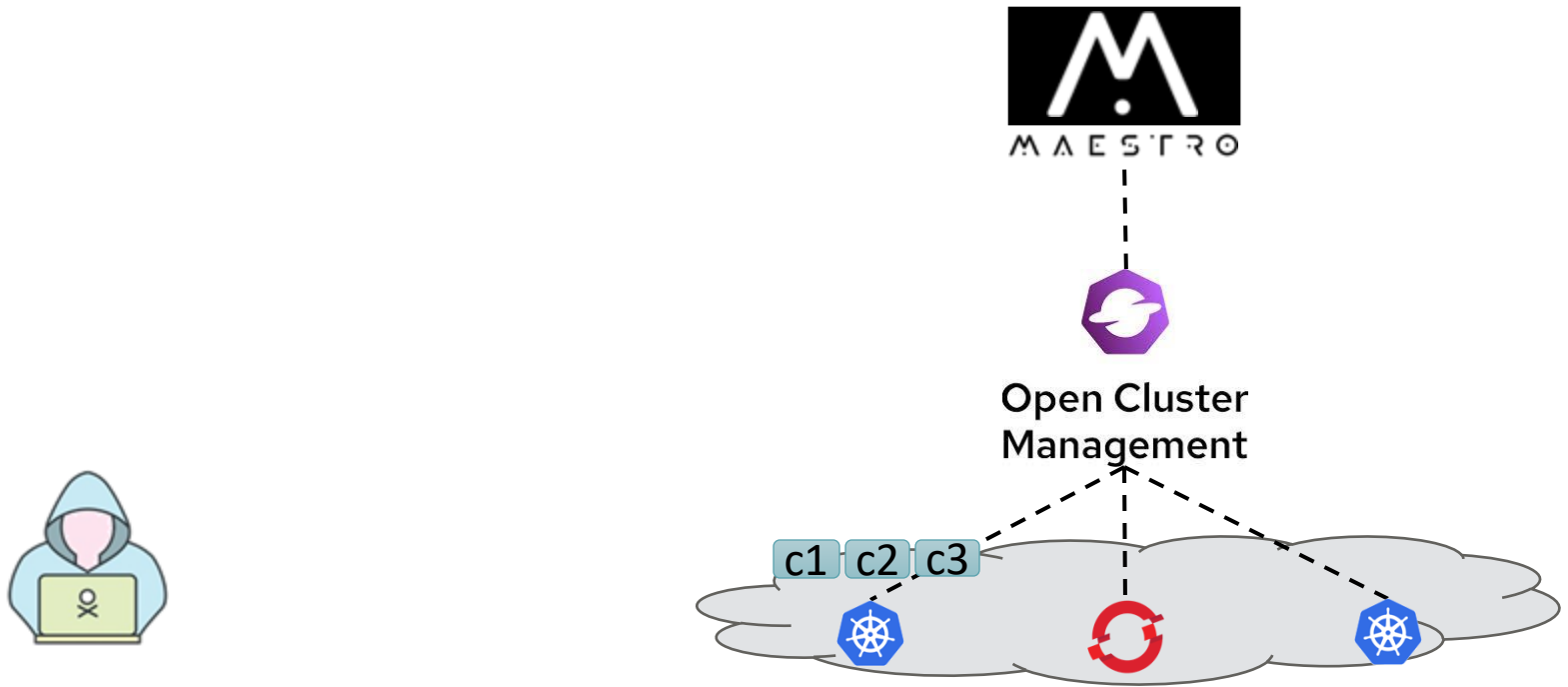


Scenario #2

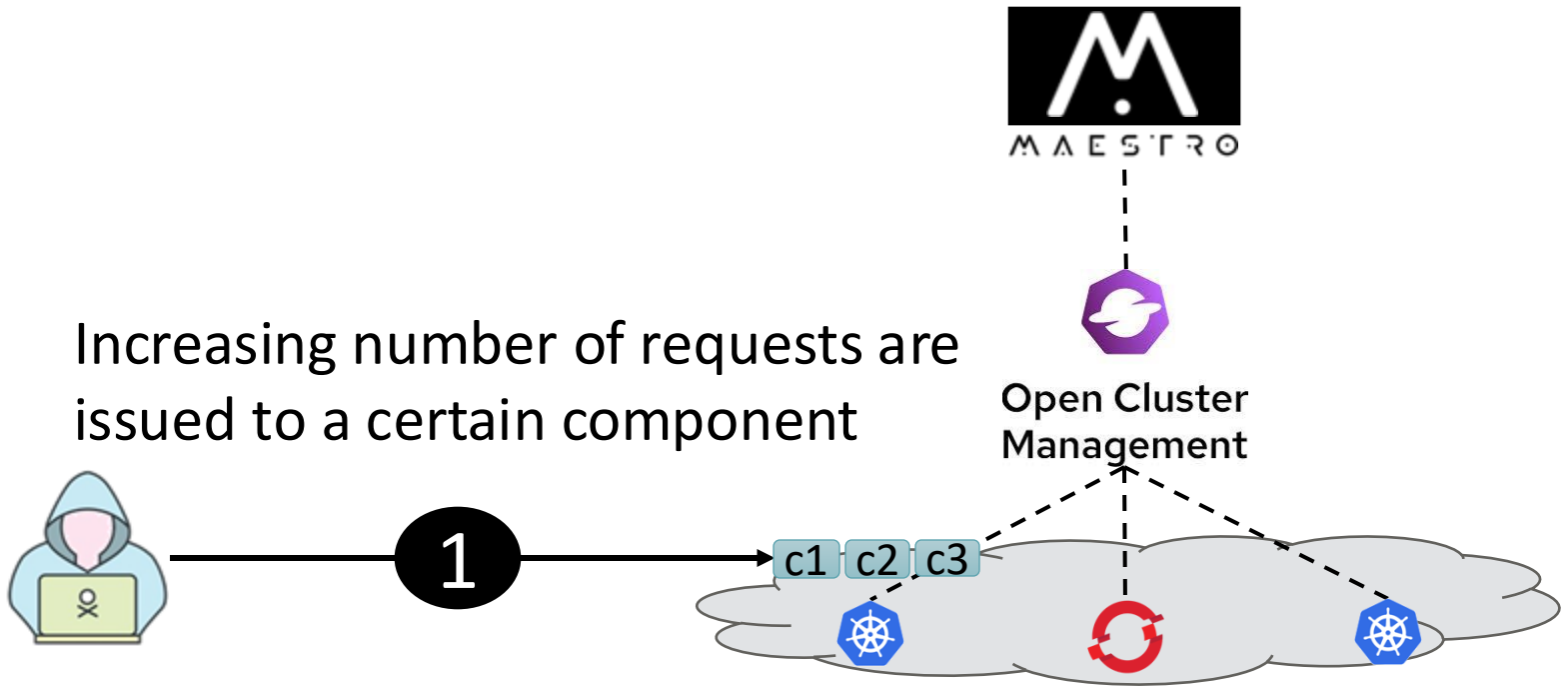
Real-time updates for service security

Scenario #2 – Realtime service security updates

This scenario emulates a potential attack to the service from an external entity

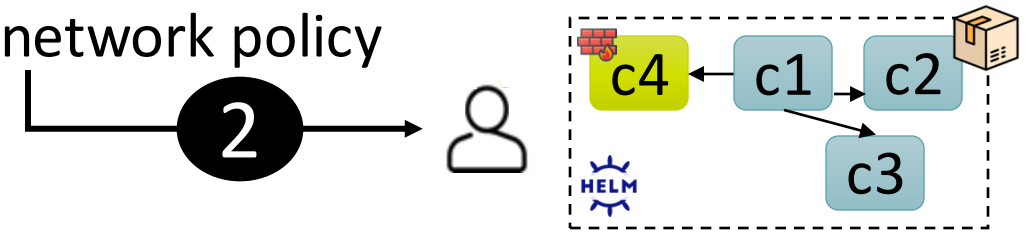


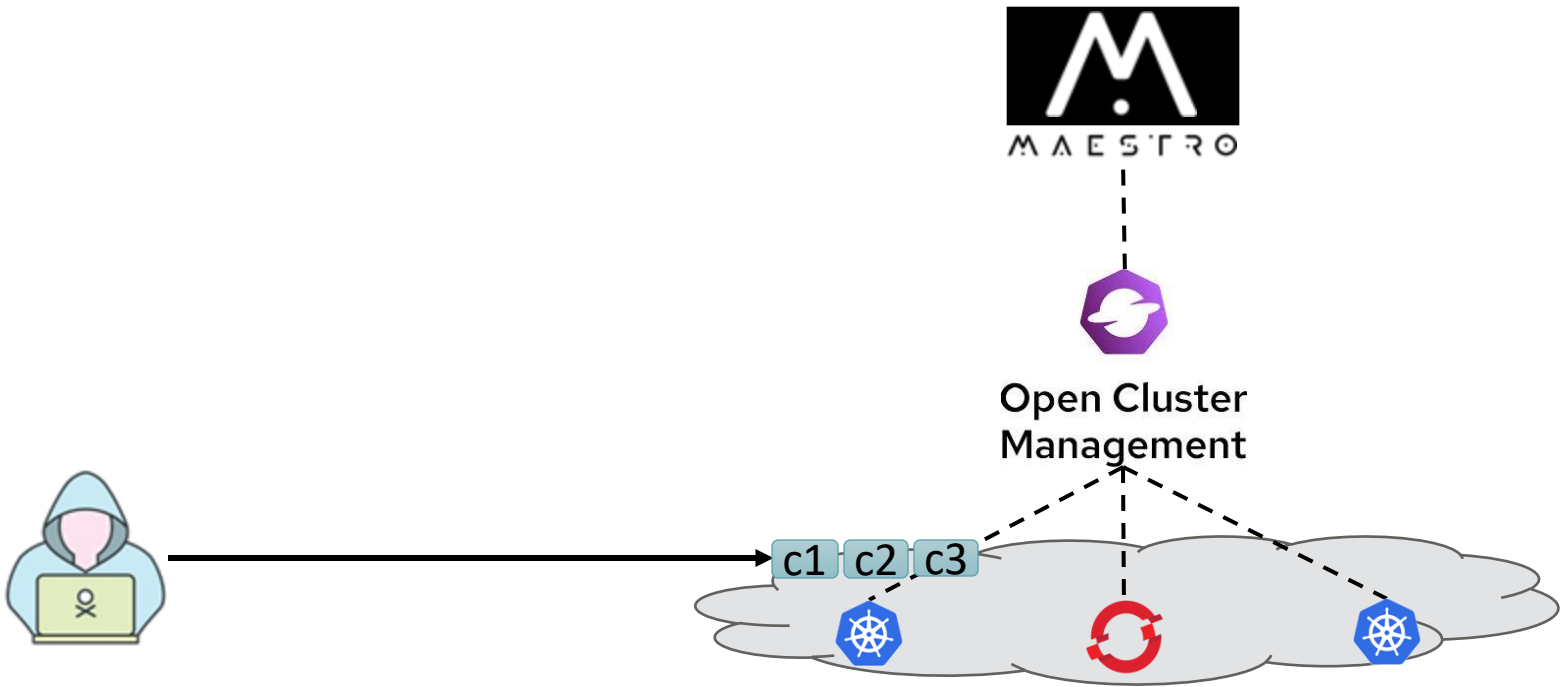
Scenario #2 – Realtime service security updates



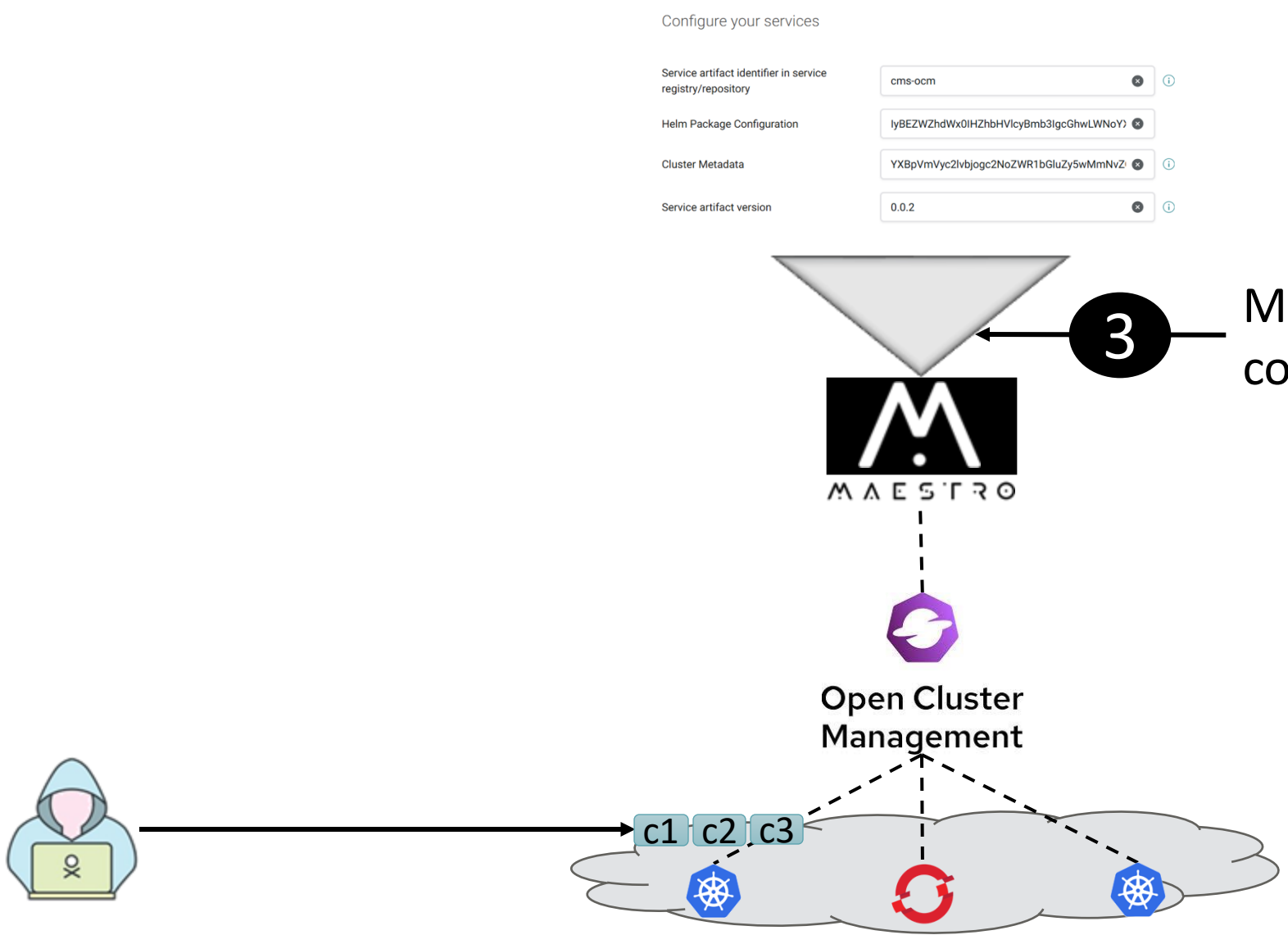
Scenario #2 – Realtime service security updates

Service designer creates a security patch to fortify the service using a network policy



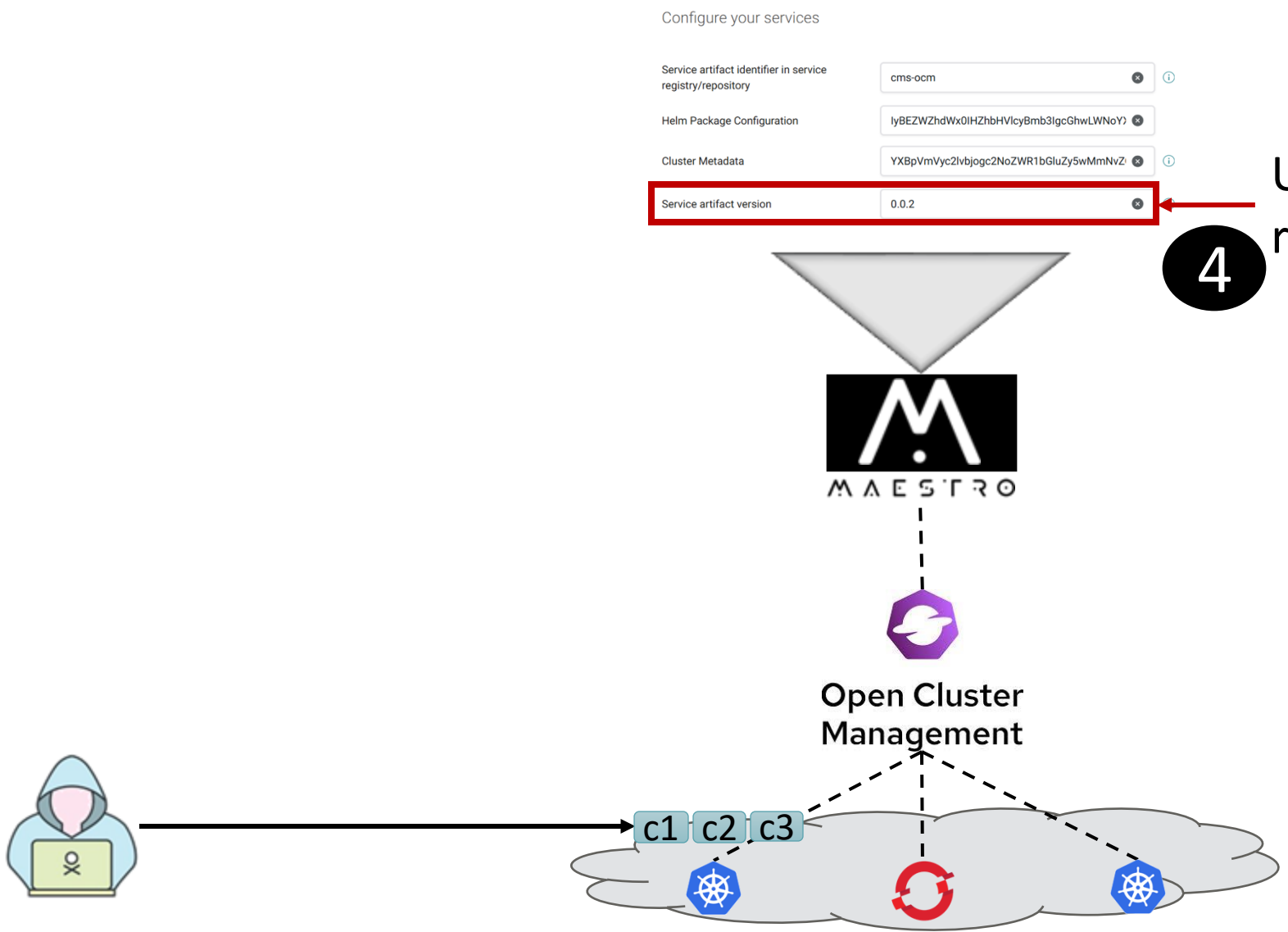


Scenario #2 – Realtime service security updates



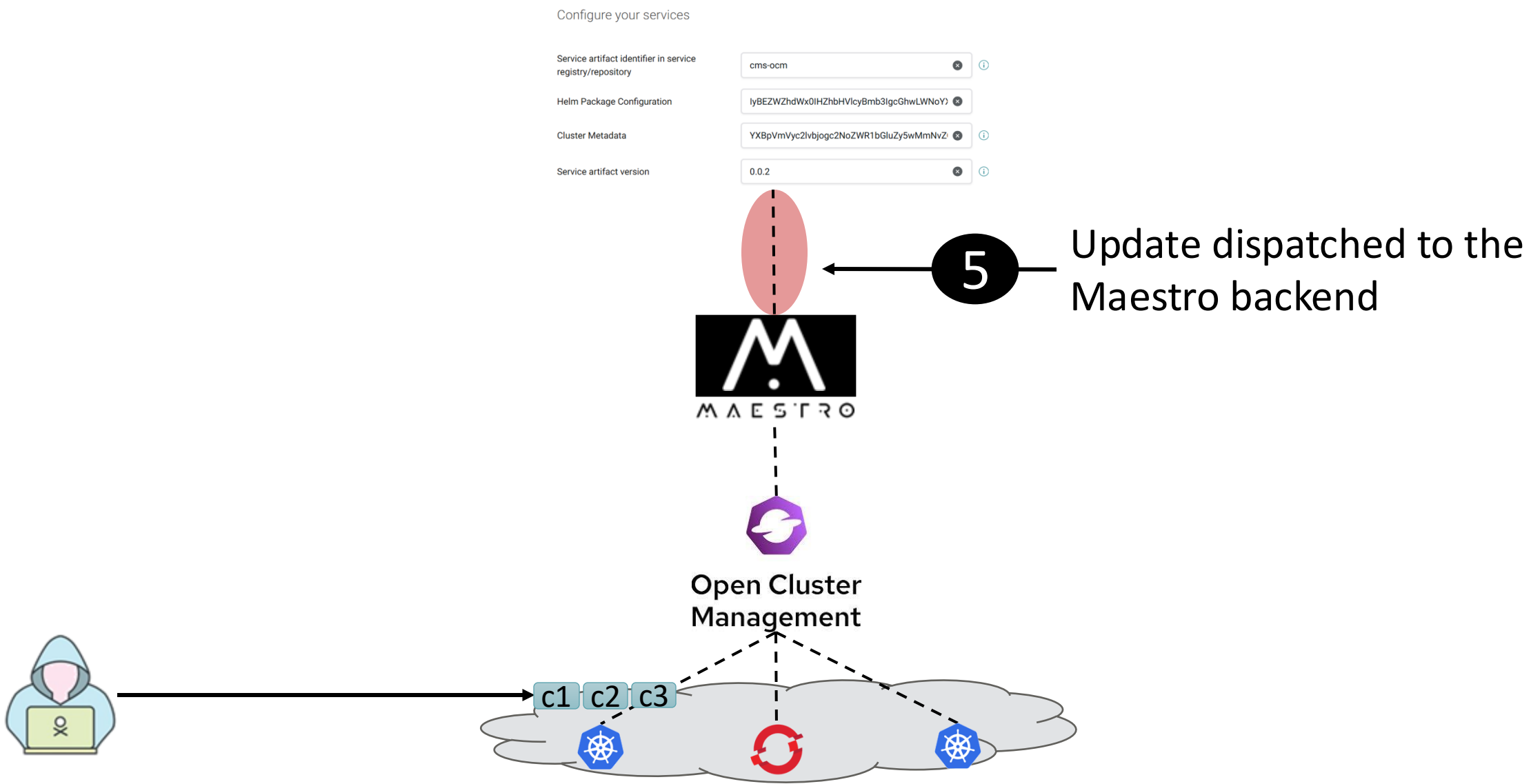
Maestro exposes service configuration knobs via the portal

Scenario #2 – Realtime service security updates

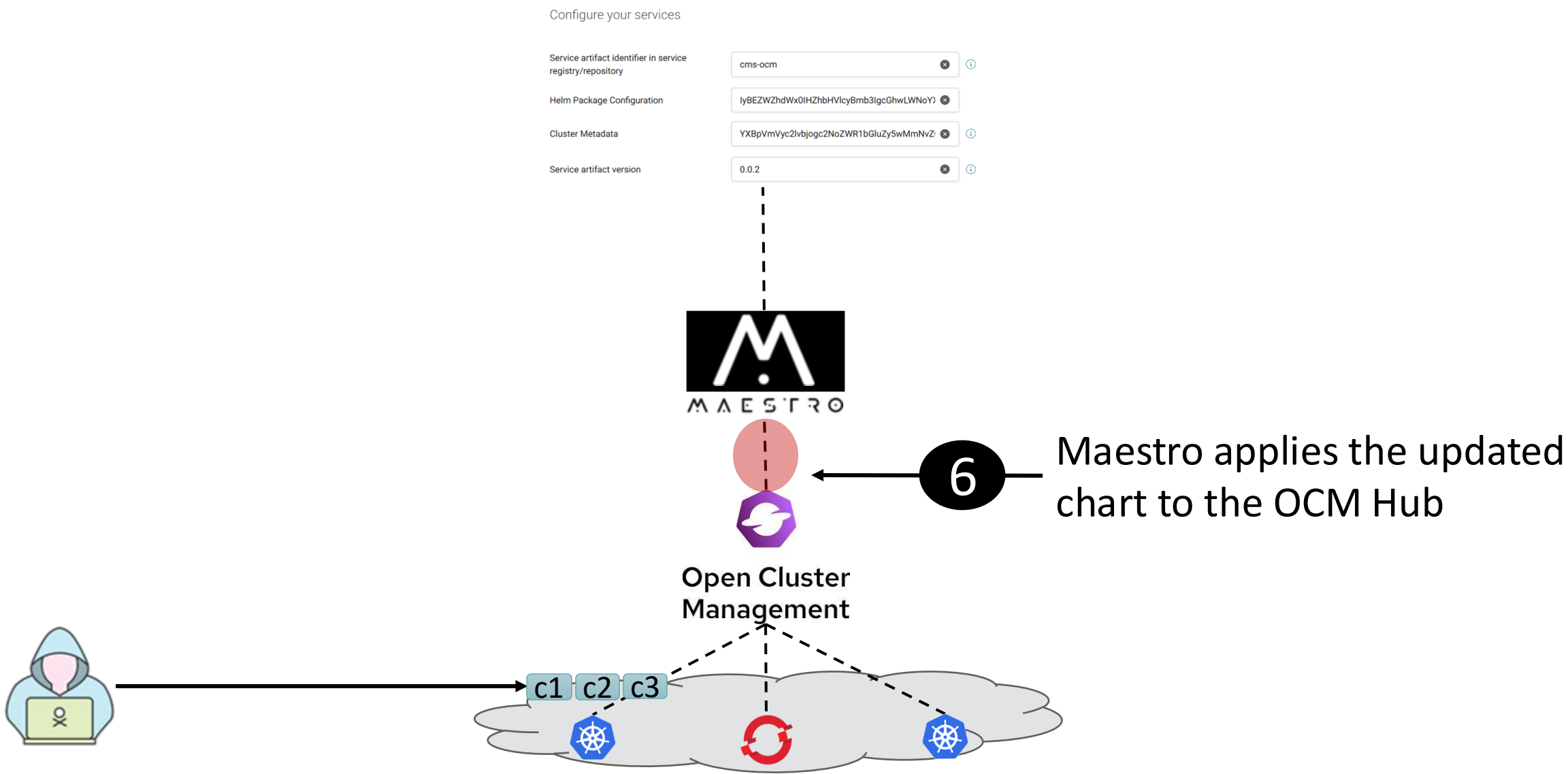


User bumps service version up to roll-out a new release on the fly

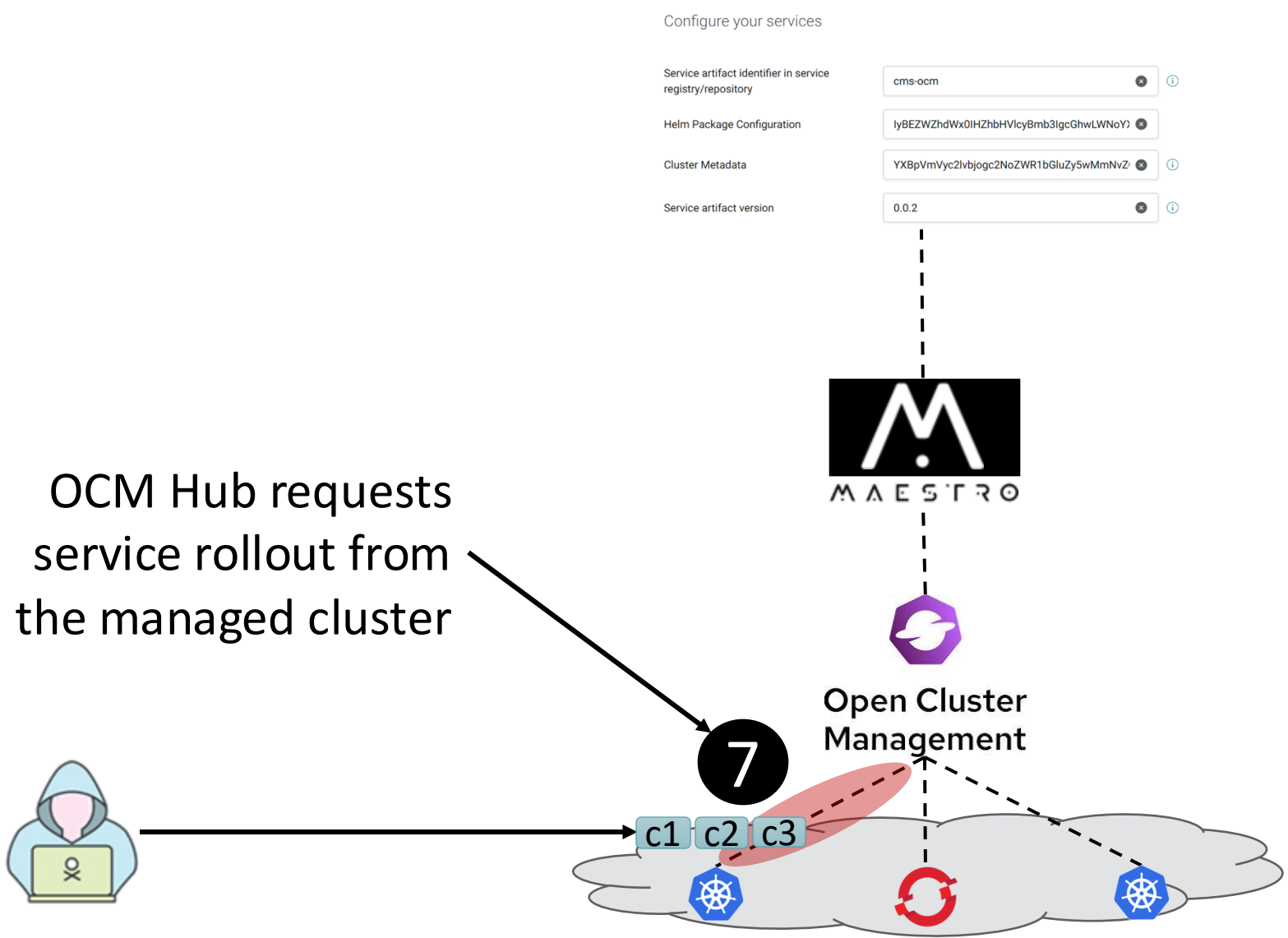
Scenario #2 – Realtime service security updates



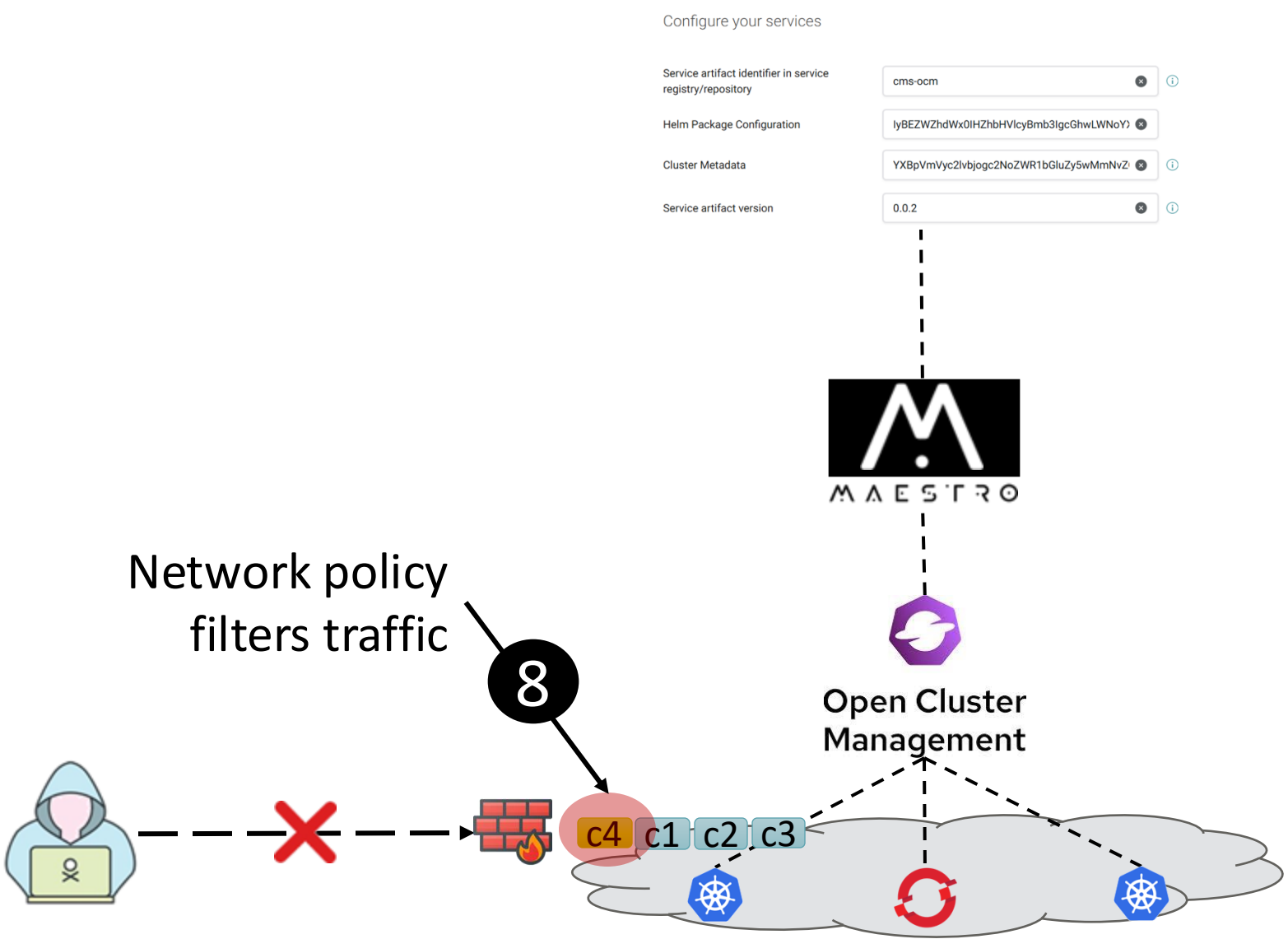
Scenario #2 – Realtime service security updates



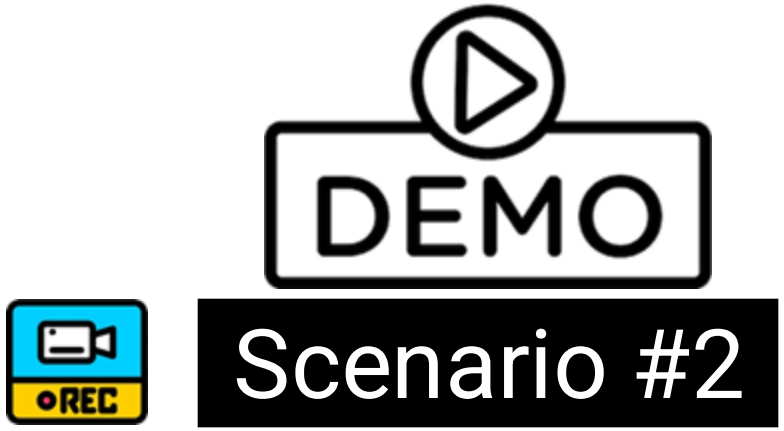
Scenario #2 – Realtime service security updates



Scenario #2 – Realtime service security updates



Scenario #2 – Demo

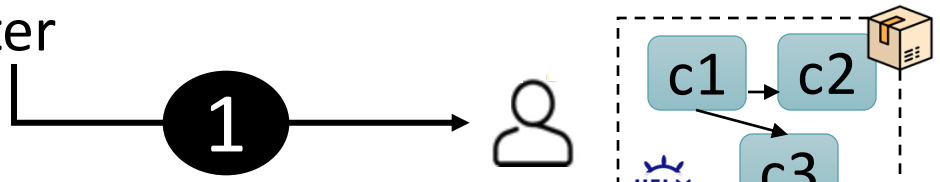


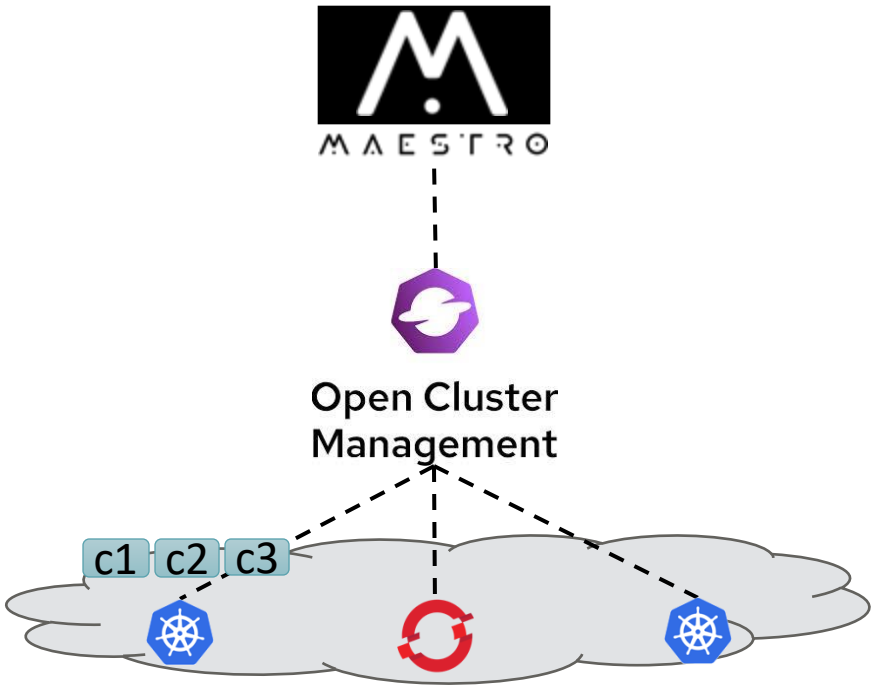
Scenario #3

Real-time updates for service re-location

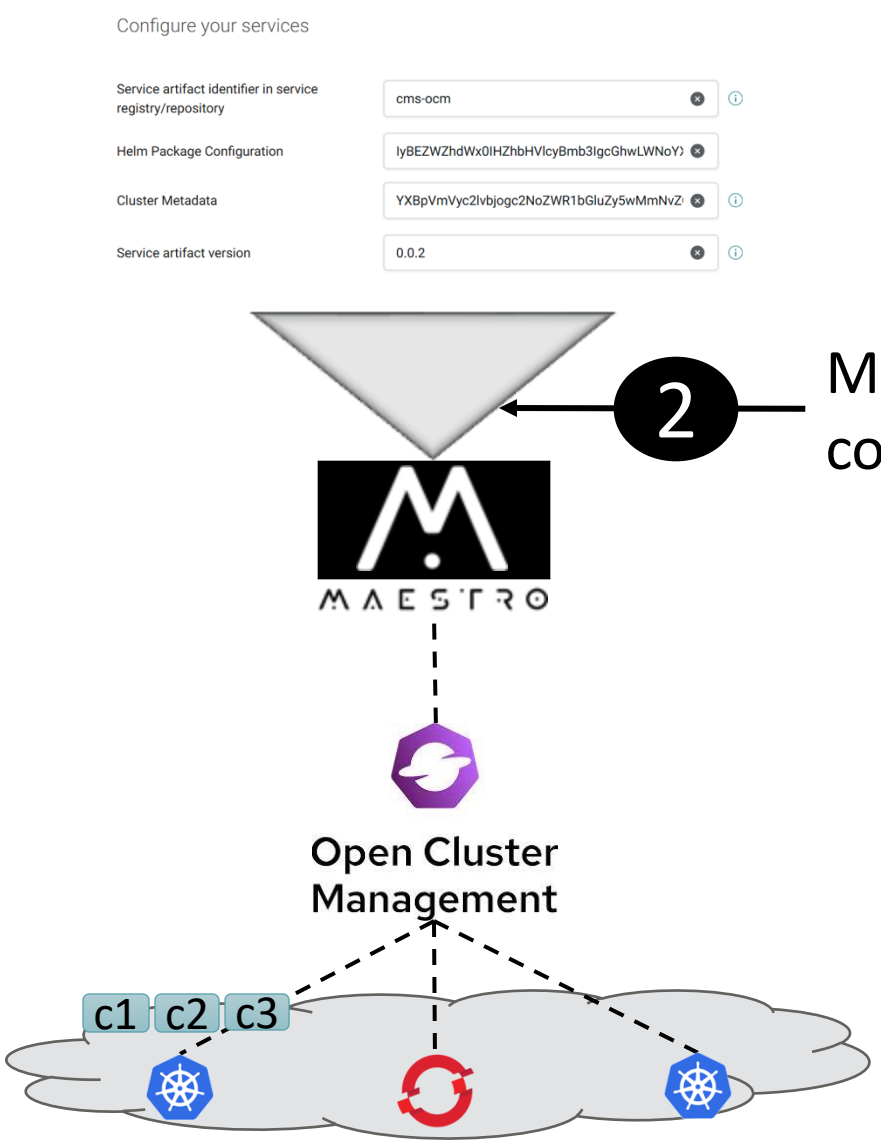
Scenario #3 – Realtime service re-scheduling

Service designer wants to move two of the service components to another cluster



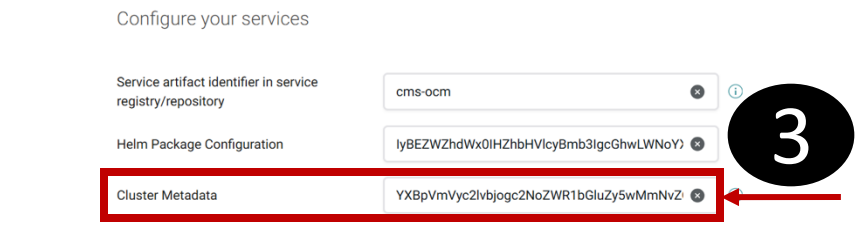


Scenario #3 – Realtime service re-scheduling

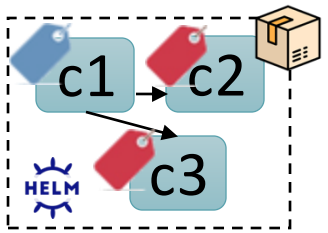


Maestro exposes service configuration knobs via the portal

Scenario #3 – Realtime service re-scheduling

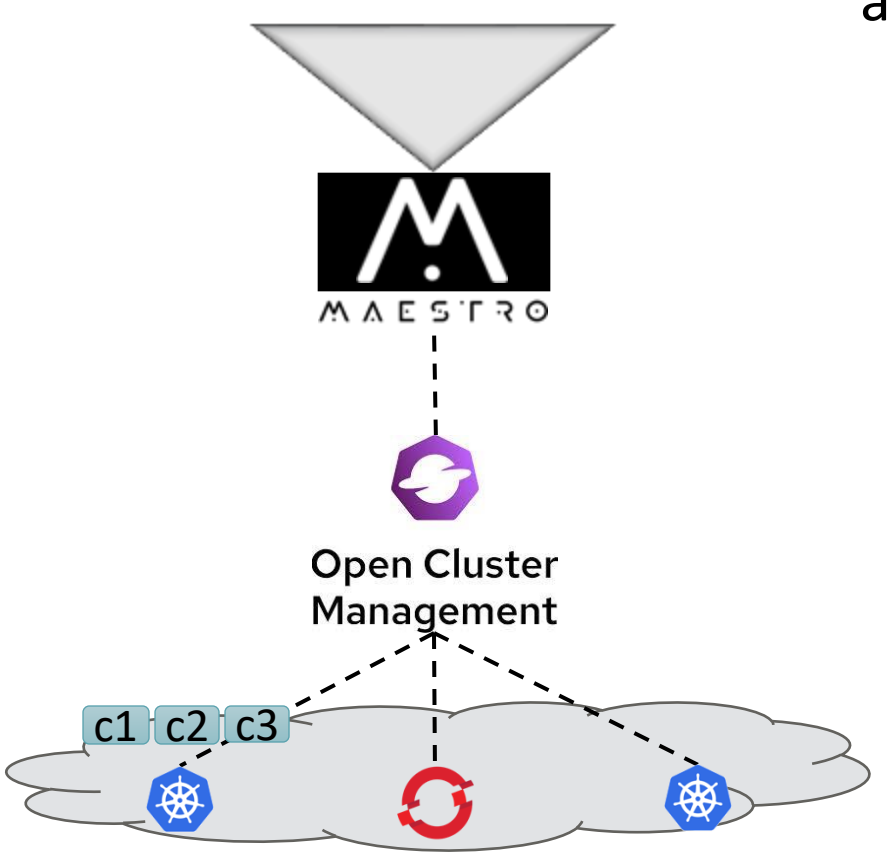


3 User modifies cluster metadata accordingly

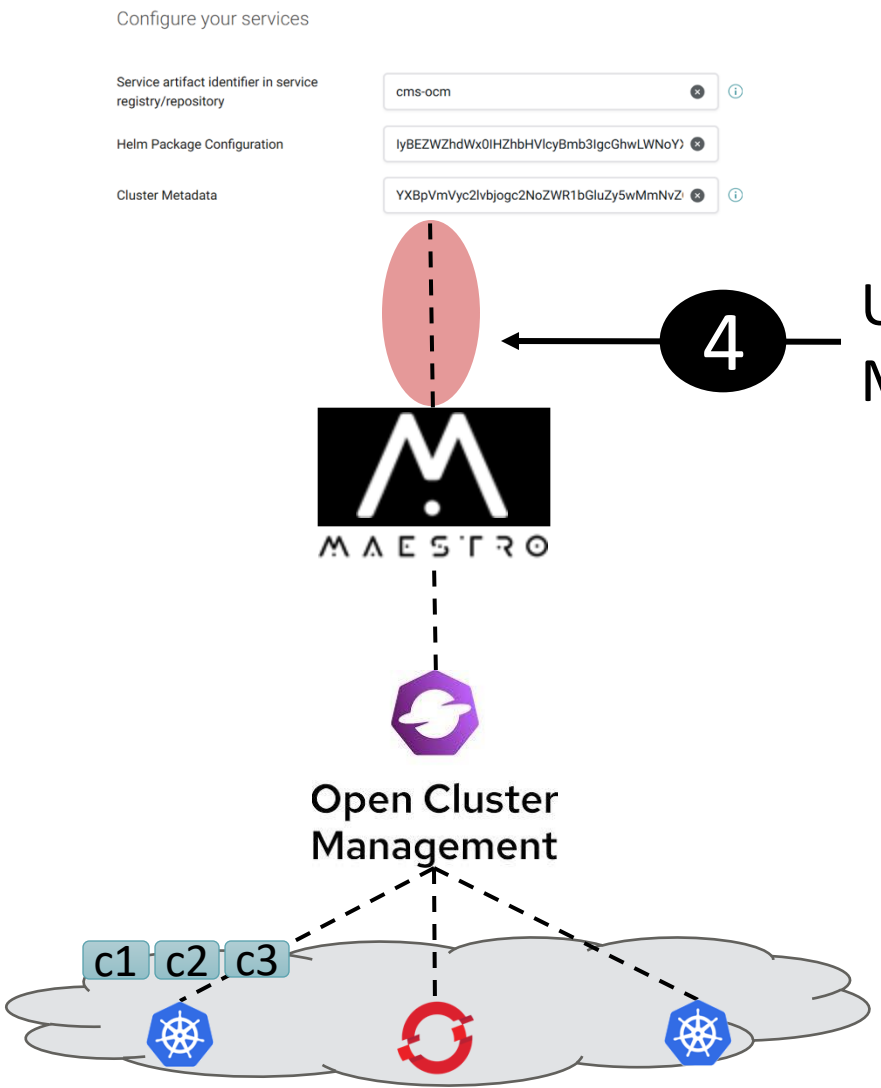


Label: kubernetes

Label: openshift

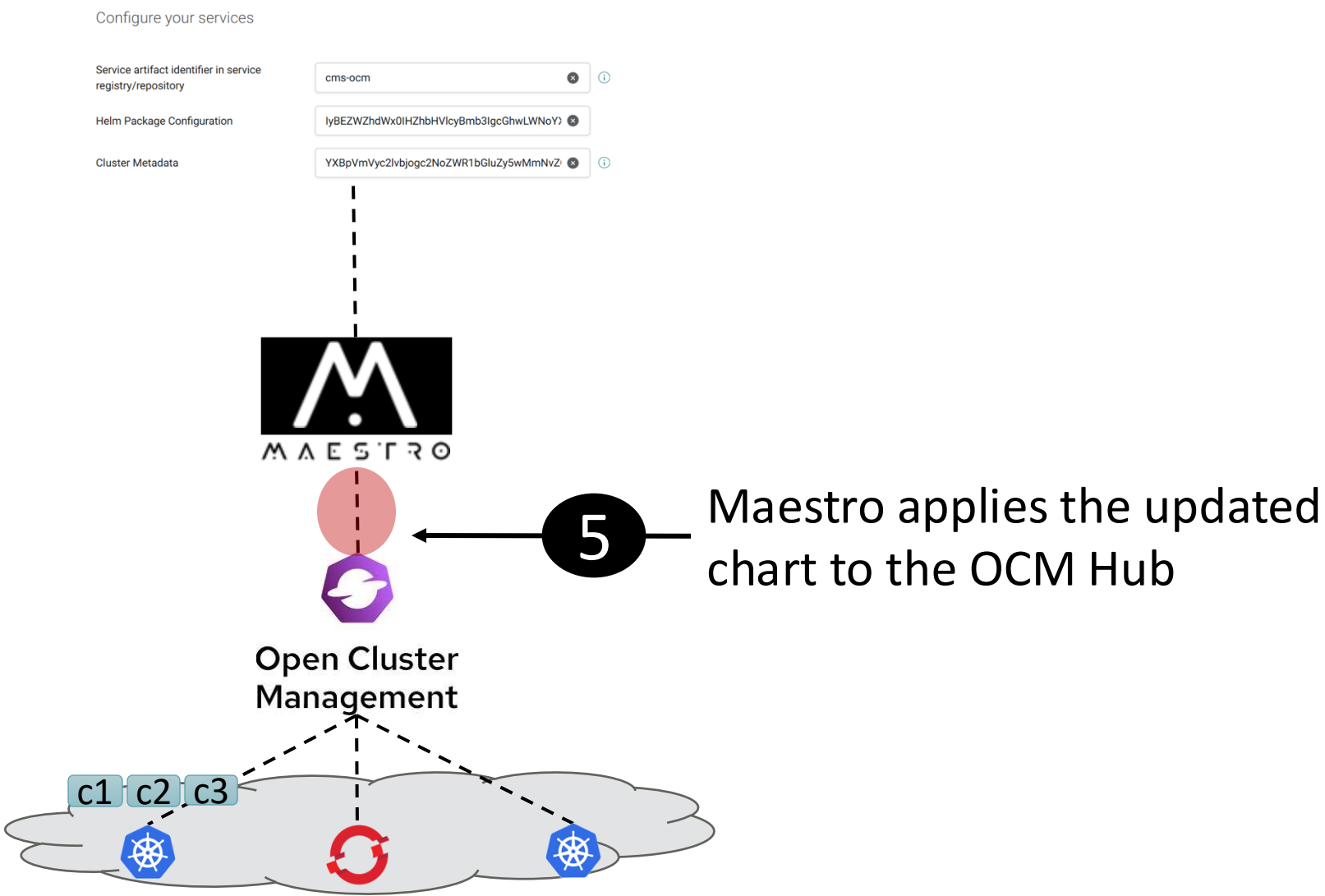


Scenario #3 – Realtime service re-scheduling



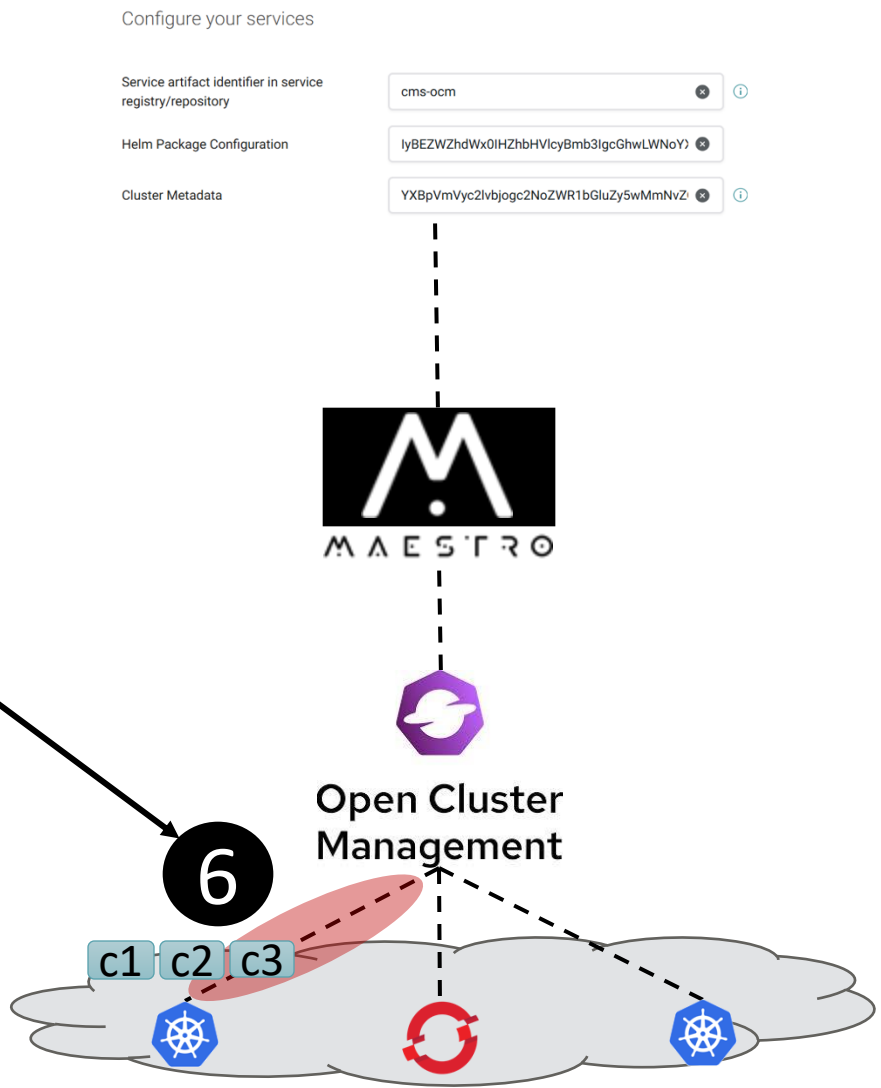
Update dispatched to the Maestro backend

Scenario #3 – Realtime service re-scheduling



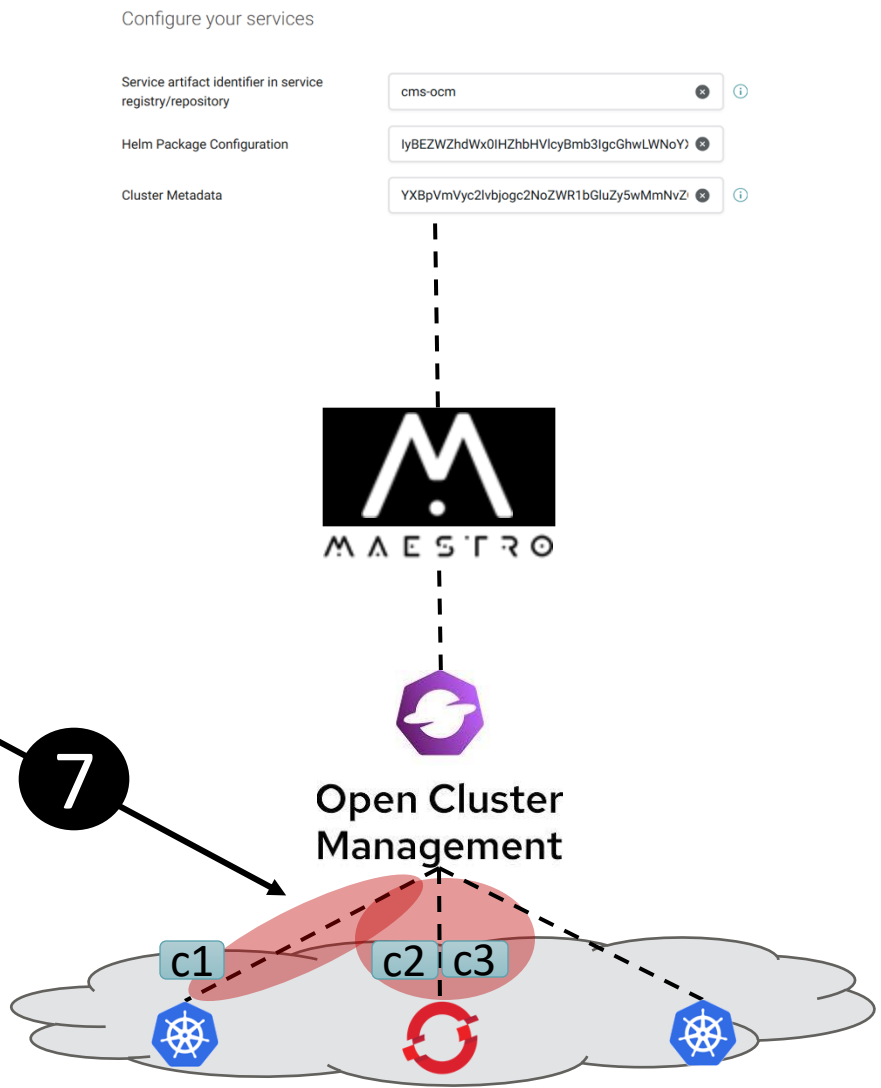
Scenario #3 – Realtime service re-scheduling

OCM Hub requests service re-location from the managed cluster

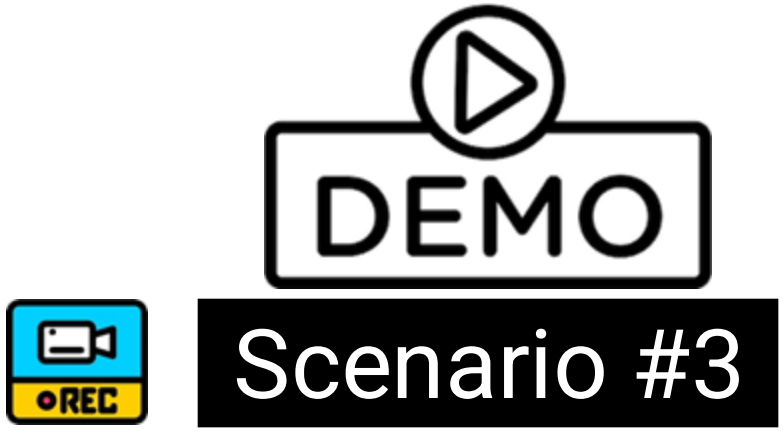


Scenario #3 – Realtime service re-scheduling

2x service components re-scheduled on the OpenShift cluster



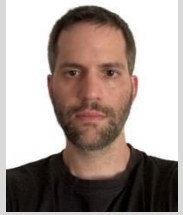
Scenario #3 – Demo



Team

Team behind the demo

Anastasios Poimenidis
Senior Software Engineer



Vasilios Katopodis
Senior Software Engineer

Labros Papadopoulos
Software Engineer

Alexandros Valantasis
Tech. Management - PM

Vasilios Charlaftis
Software Engineer

Padelis Malekas
Software Engineer

Georgios P. Katsikas
NSS Technical Leader - PM

Nikos Psaromanolakis
Tech. Management - PM

Open-source activities

Open-source activities

Open-source component	Partner
Multi-domain service orchestrator https://maestro-mkdocs.readthedocs.io/ (Soon under ETSI OSL)	UBITECH ubiquitous solutions
Operations Support System (OSS) for Network-as-a-Service https://osl.etsi.org/	University of Patras; ONIT Emerging Networks & Applications
Open Cluster Management (OCM) https://open-cluster-management.io/	Red Hat

OpenSlice

Projects behind this work

Acknowledgements

HORIZON-JU-SNS-2022 **ACROSS** project with grant agreement number 101097122

Programming Platform for intelligent COllaborative DEployments over heterogeneous edge-IoT environments (**P2CODE**) under GA No. 101093069

Joint technical effort among

OpenSlice

Thank you!