



# Oakestra : A Hierarchical Orchestration Framework for Edge Computing

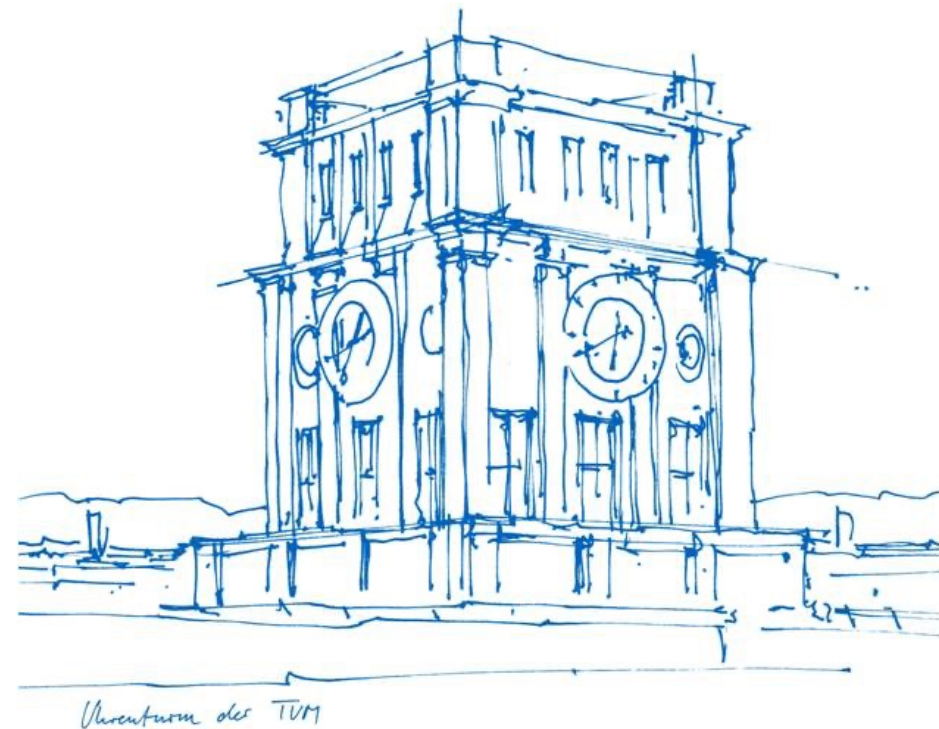
**Giovanni Bartolomeo, Mehdi Yosofie, Simon Bärle,  
Oliver Haluszczynski, Nitinder Mohan, Jörg Ott**

Chair of Connected Mobility

Technical University of Munich, Germany

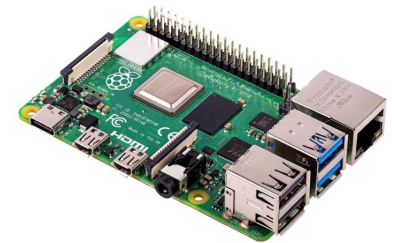


[oakestra.io](https://oakestra.io)



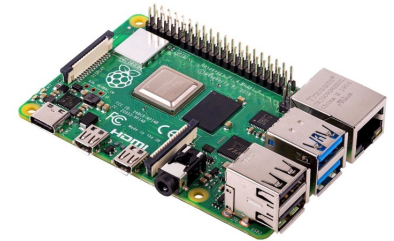
# Edge Computing (can be ...)

- Constrained small footprint hardware



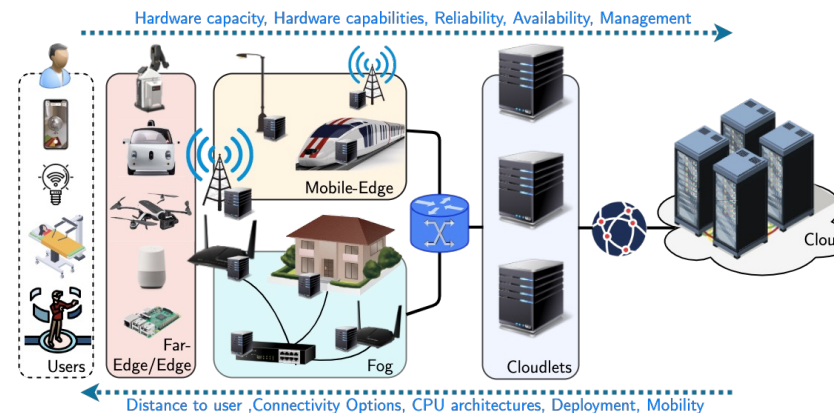
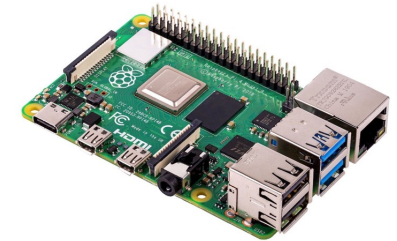
# Edge Computing (can be ...)

- Constrained small footprint hardware
- Heterogeneous and specialized
  - CPU/GPU architectures (x86/ARM/...)
  - Networking (WiFi/ethernet/cellular)
  - Connectivity (public/private)



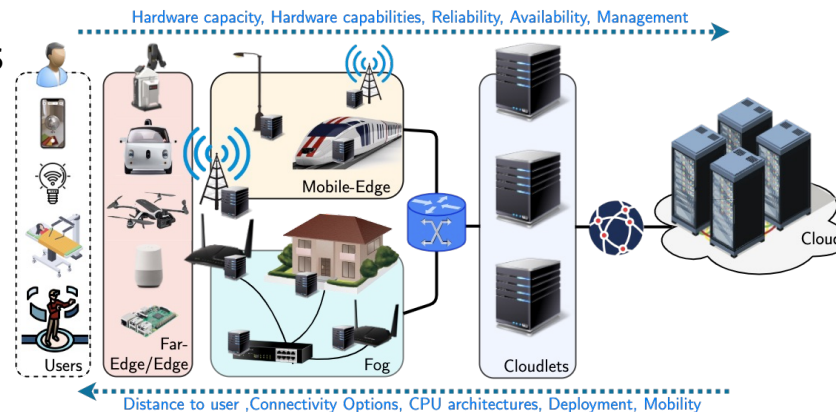
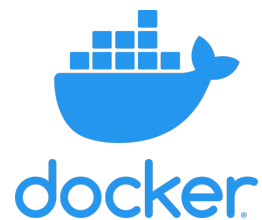
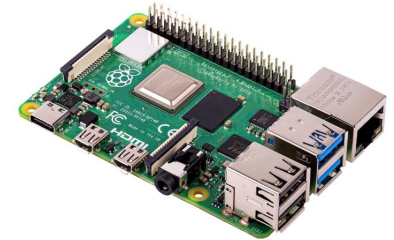
# Edge Computing (can be ...)

- Constrained small footprint hardware
- Heterogeneous and specialized
  - CPU/GPU architectures (x86/ARM/...)
  - Networking (WiFi/ethernet/cellular)
  - Connectivity (public/private)
- Owned by different operators in different environments



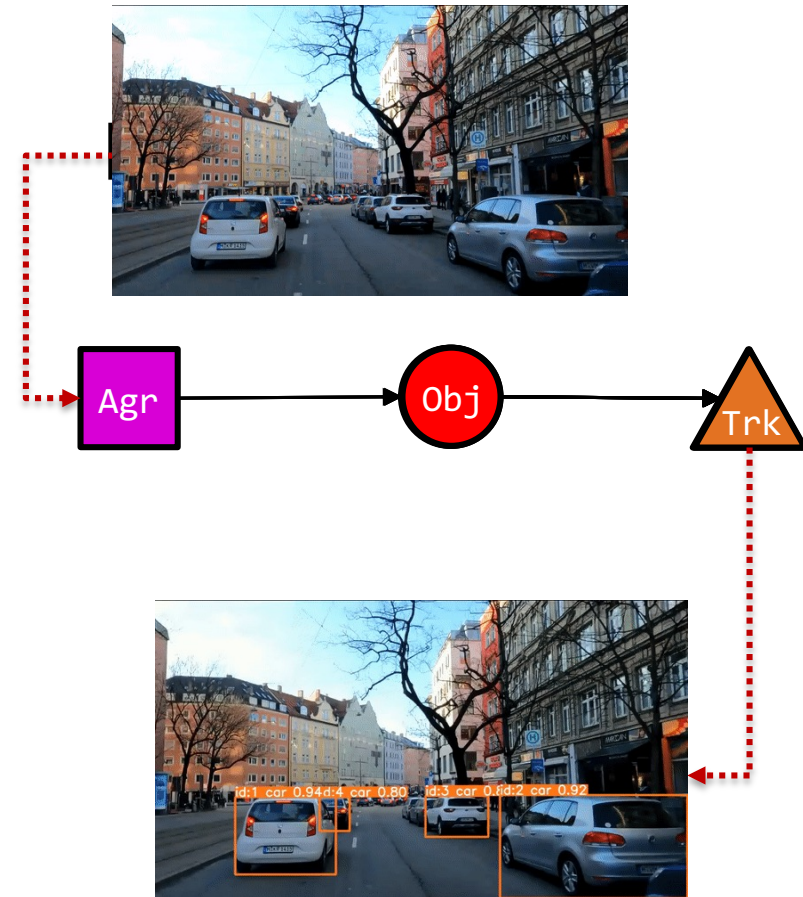
# Edge Computing (can be ...)

- Constrained small footprint hardware
- Heterogeneous and specialized
  - CPU/GPU architectures (x86/ARM/...)
  - Networking (WiFi/ethernet/cellular)
  - Connectivity (public/private)
- Owned by different operators in different environments
- Support different virtualization/runtimes



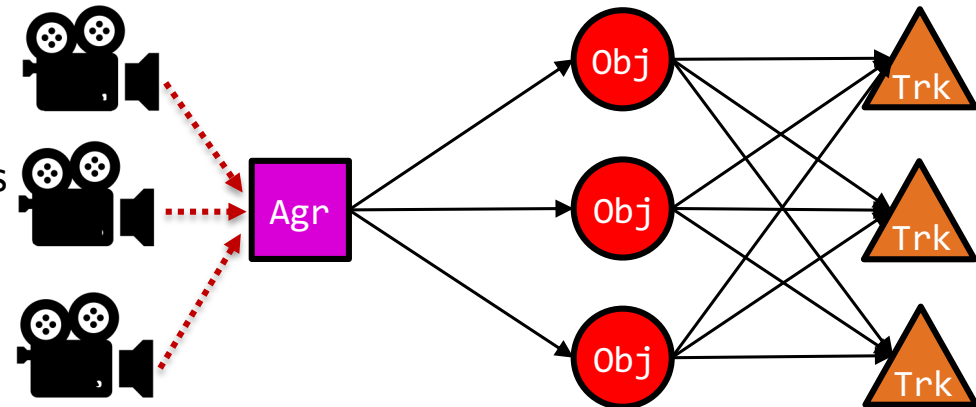
# Applications leveraging Edge Computing

- Applications are complex and composed of many microservices
- Management and coordination of microservices across the available resources



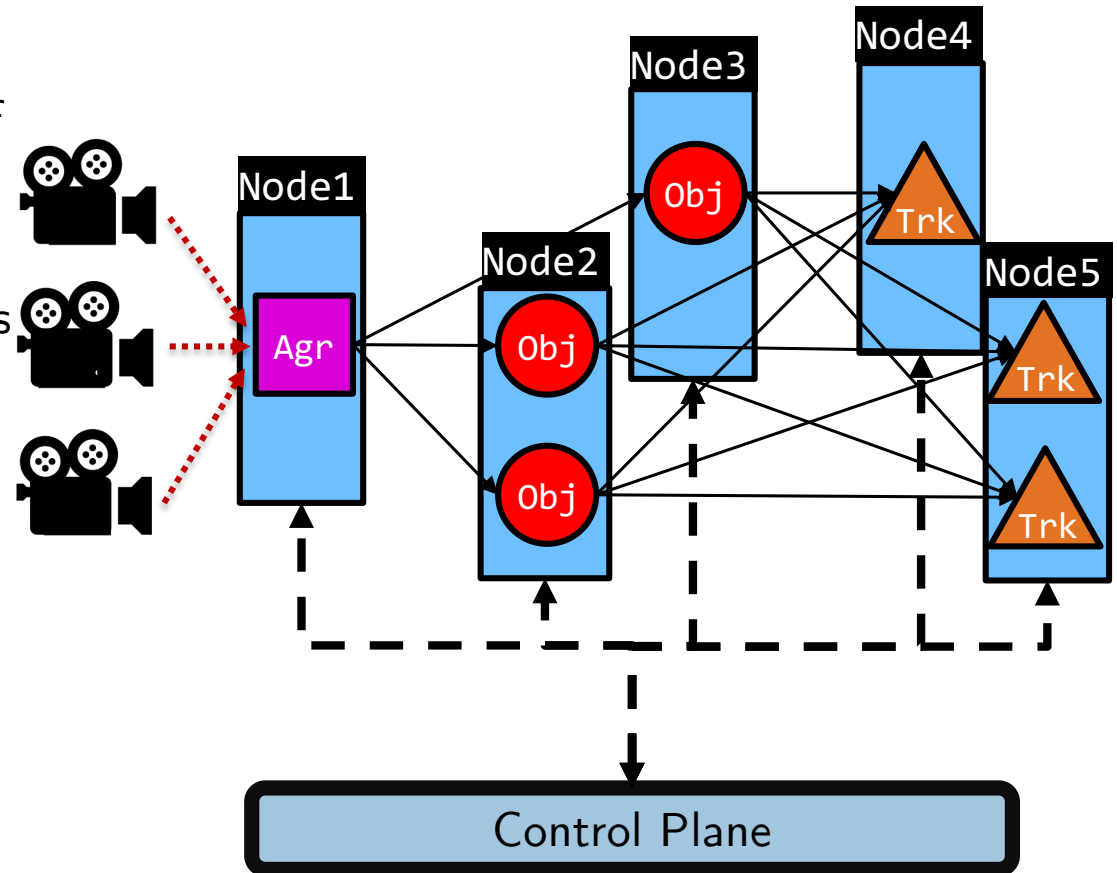
# Service Orchestration

- Applications are complex and composed of many microservices
- Management and coordination of microservices across the available resources
  - Resources and services monitoring
  - Replicas scale-up/down
  - Workload migration
  - Services networking



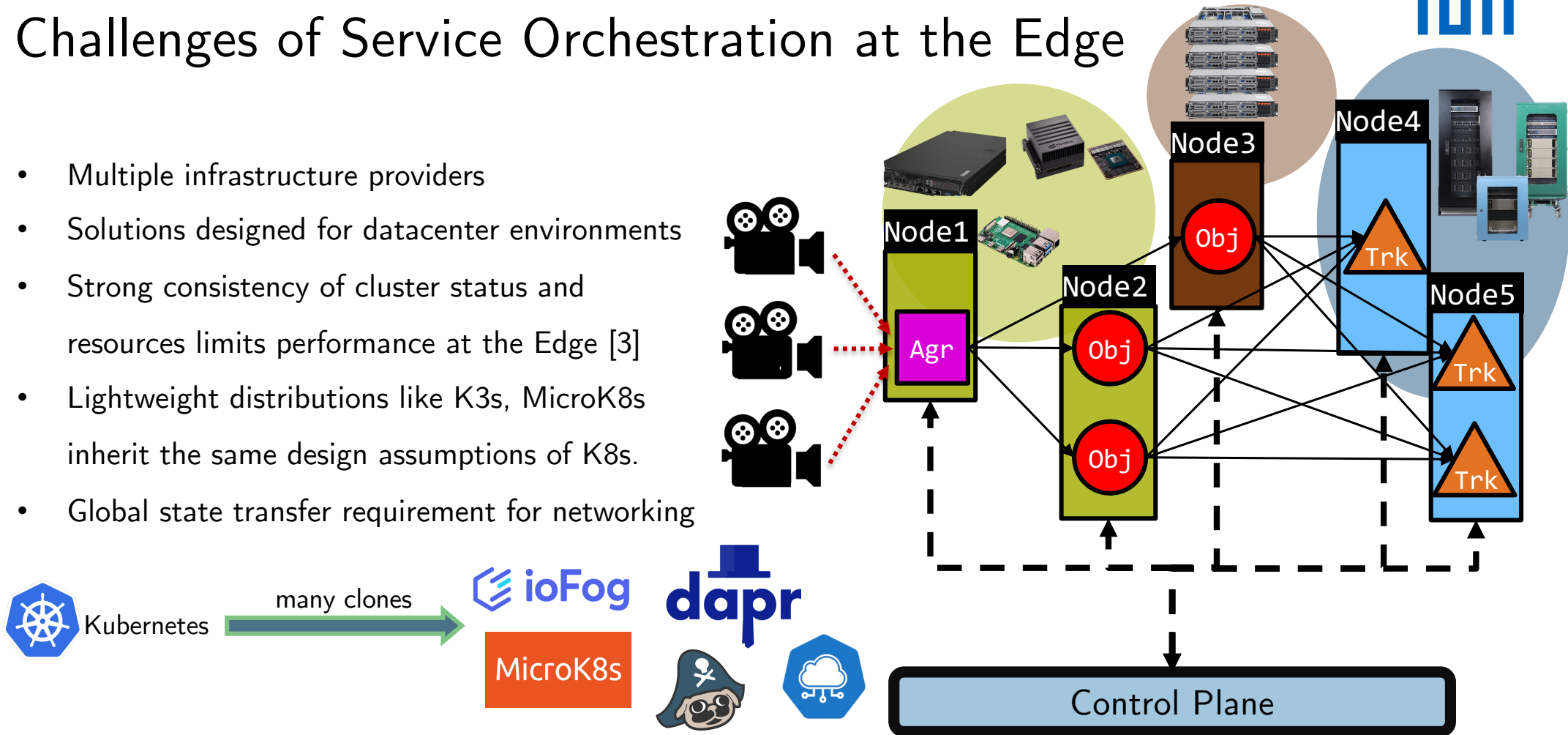
# Service Orchestration

- Applications are complex and composed of many microservices
- Management and coordination of microservices across the available resources
  - Resources and services monitoring
  - Replicas scale-up/down
  - Workload migration
  - Services networking
  - ... and more
- Control plane + Nodes
- Kubernetes (K8s) family



# Challenges of Service Orchestration at the Edge

- Multiple infrastructure providers
- Solutions designed for datacenter environments
- Strong consistency of cluster status and resources limits performance at the Edge [3]
- Lightweight distributions like K3s, MicroK8s inherit the same design assumptions of K8s.
- Global state transfer requirement for networking



[1] Andrew Jeffery, Heidi Howard, and Richard Mortier. 2021. Rearchitecting Kubernetes for the Edge. 4th ACM EdgeSys (2021)

**Hierarchical orchestration**

Consolidation of multiple edge providers

Lightweight Implementation

Semantic overlay networking

**Resource aggregation**

# Oakestra

Site-to-Site tunneling

Multi-virtualization support

Deployment across geography

**Delegated service scheduling**

Fine-grained extensible SLA primitives



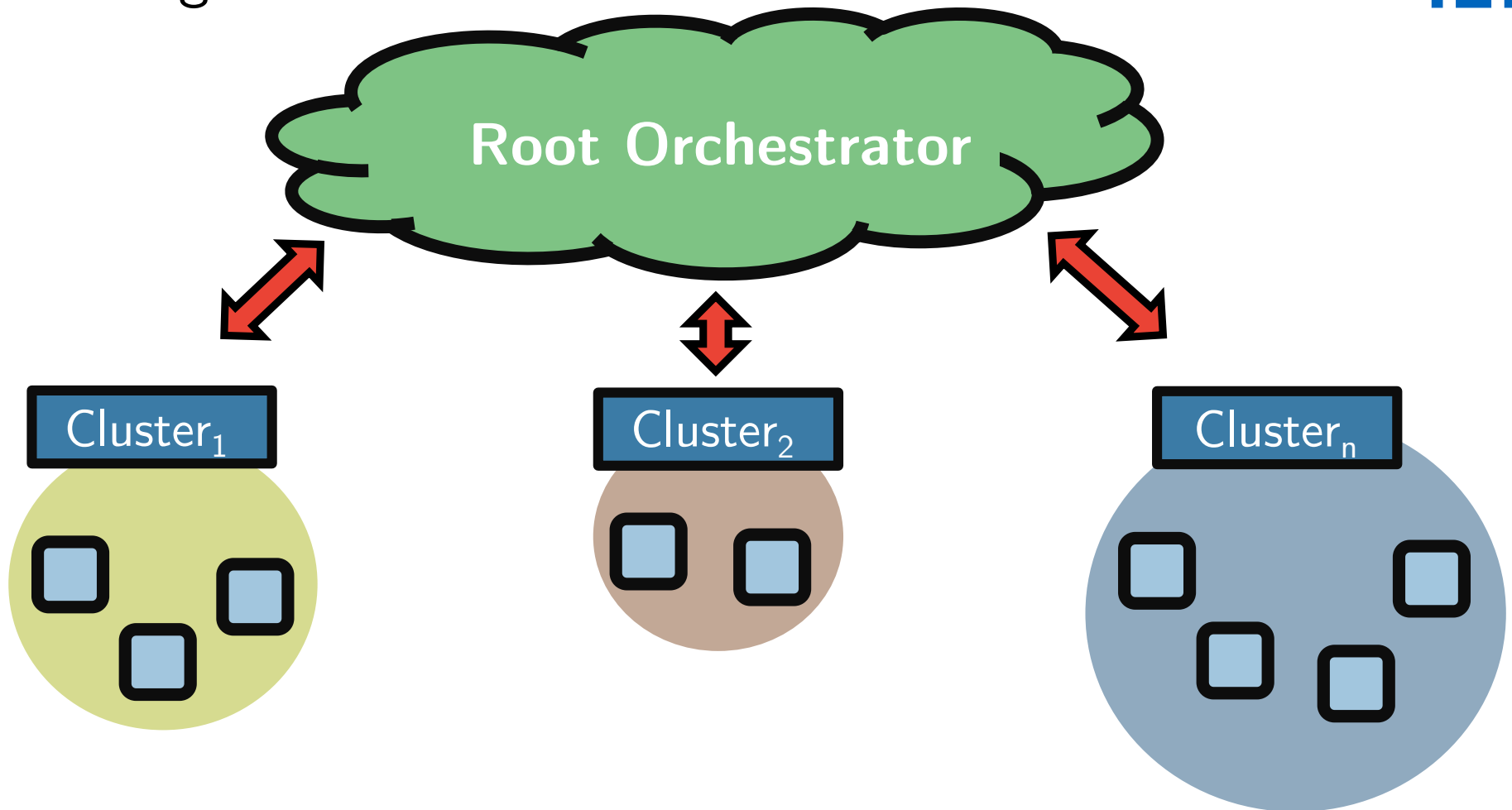
[oakestra.io](https://oakestra.io)



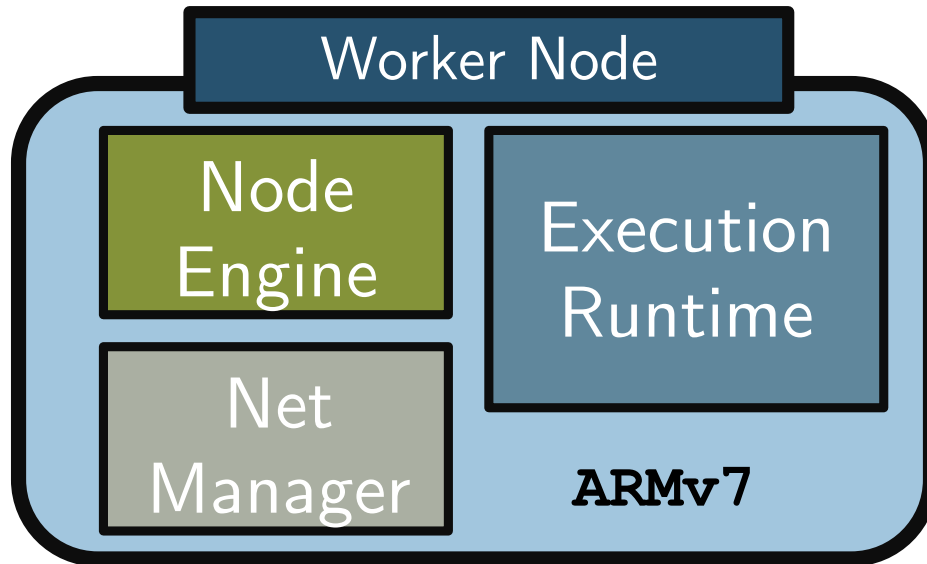
@oakestra



Oakestra

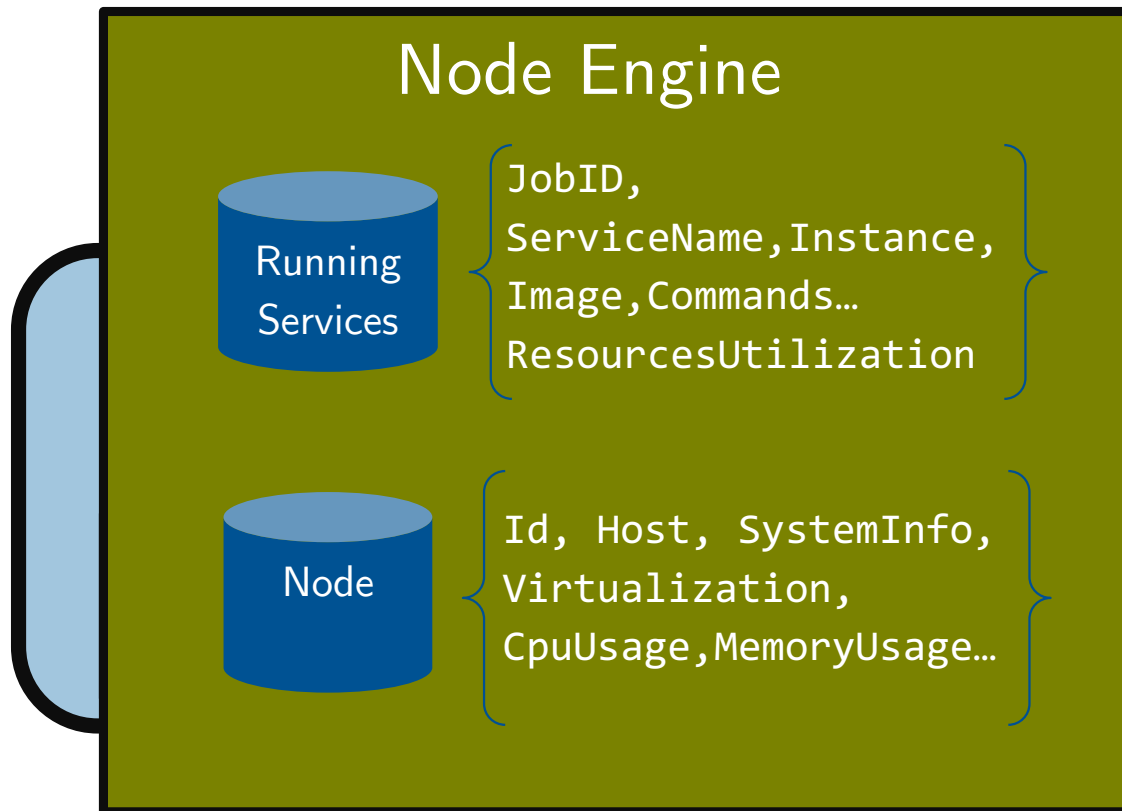


# Worker Node



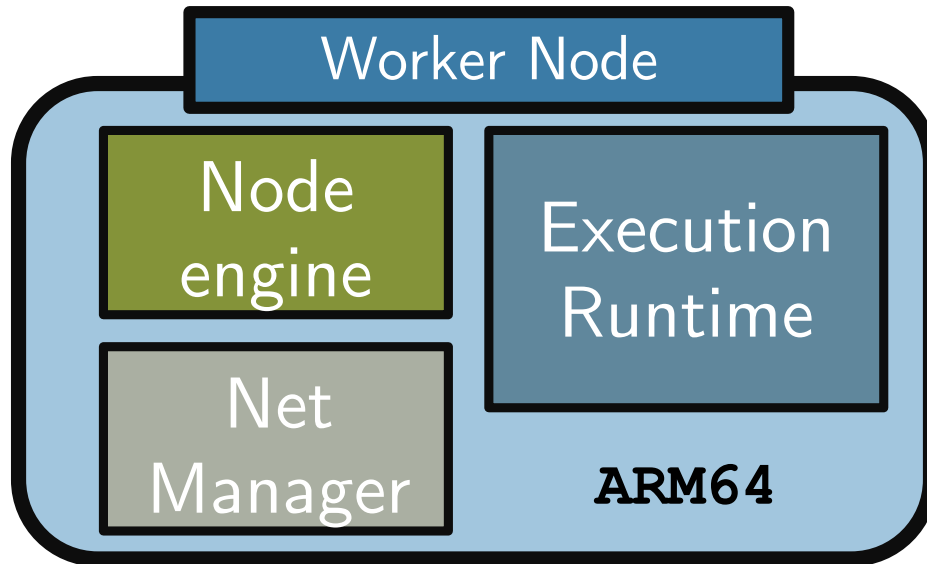
- Multiple architectures
- Multiple execution runtimes
  - Default: containerd
- Distributed networking management
- Resource/service monitoring

# Worker Node

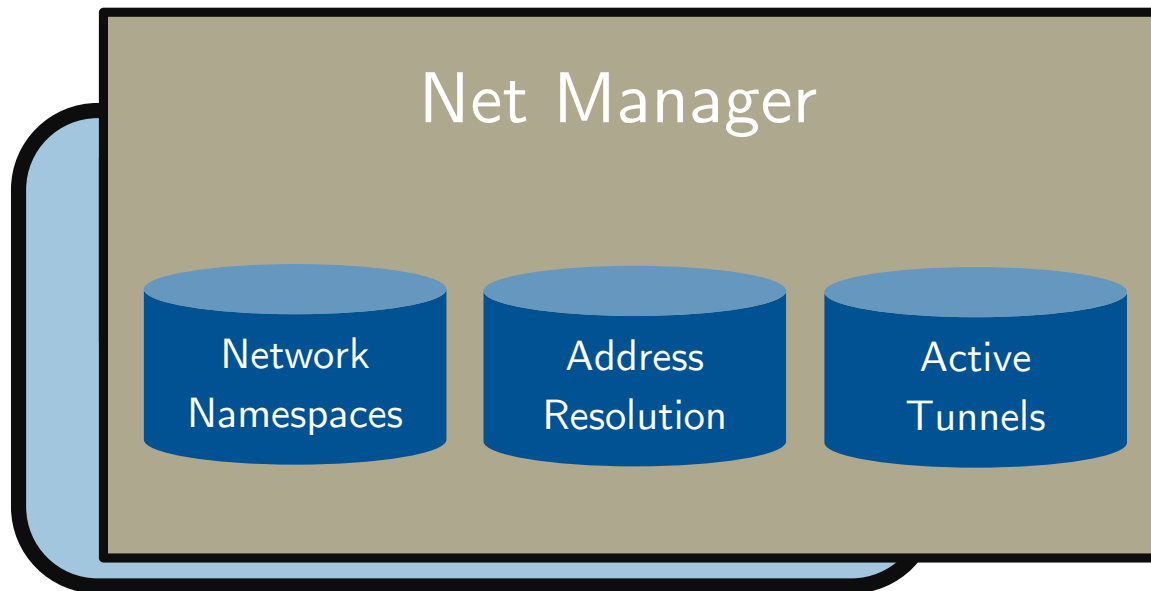


- Deployed service instances
- Service's resources utilization
- Node's system and real time info
- Periodical cluster updates

# Worker Node

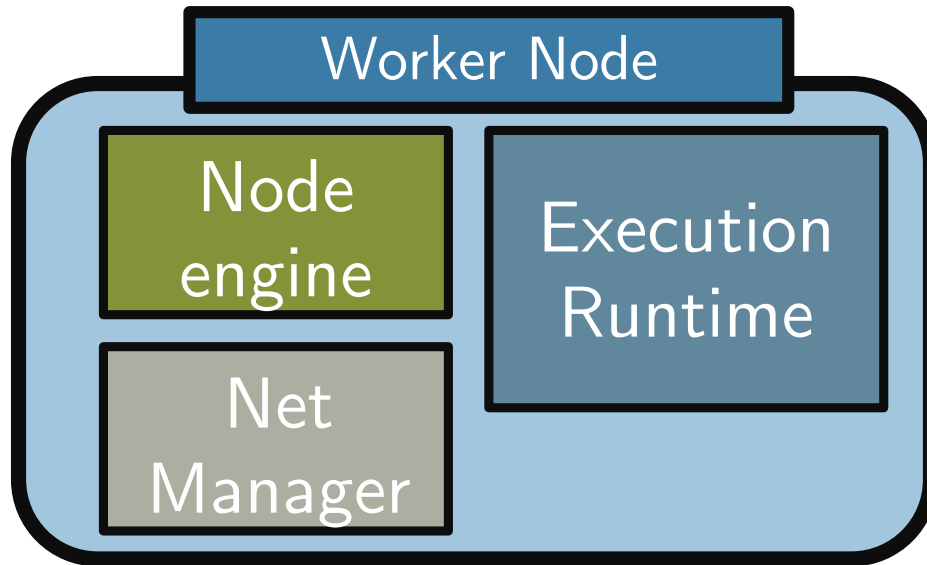


# Worker Node

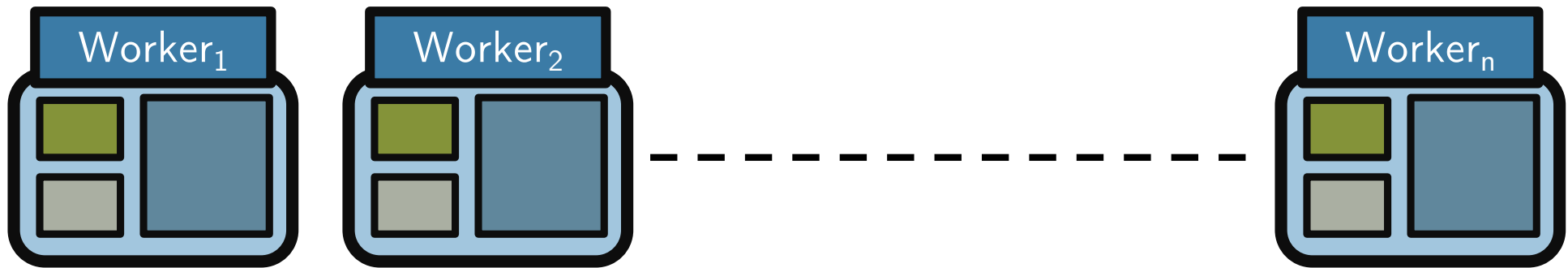


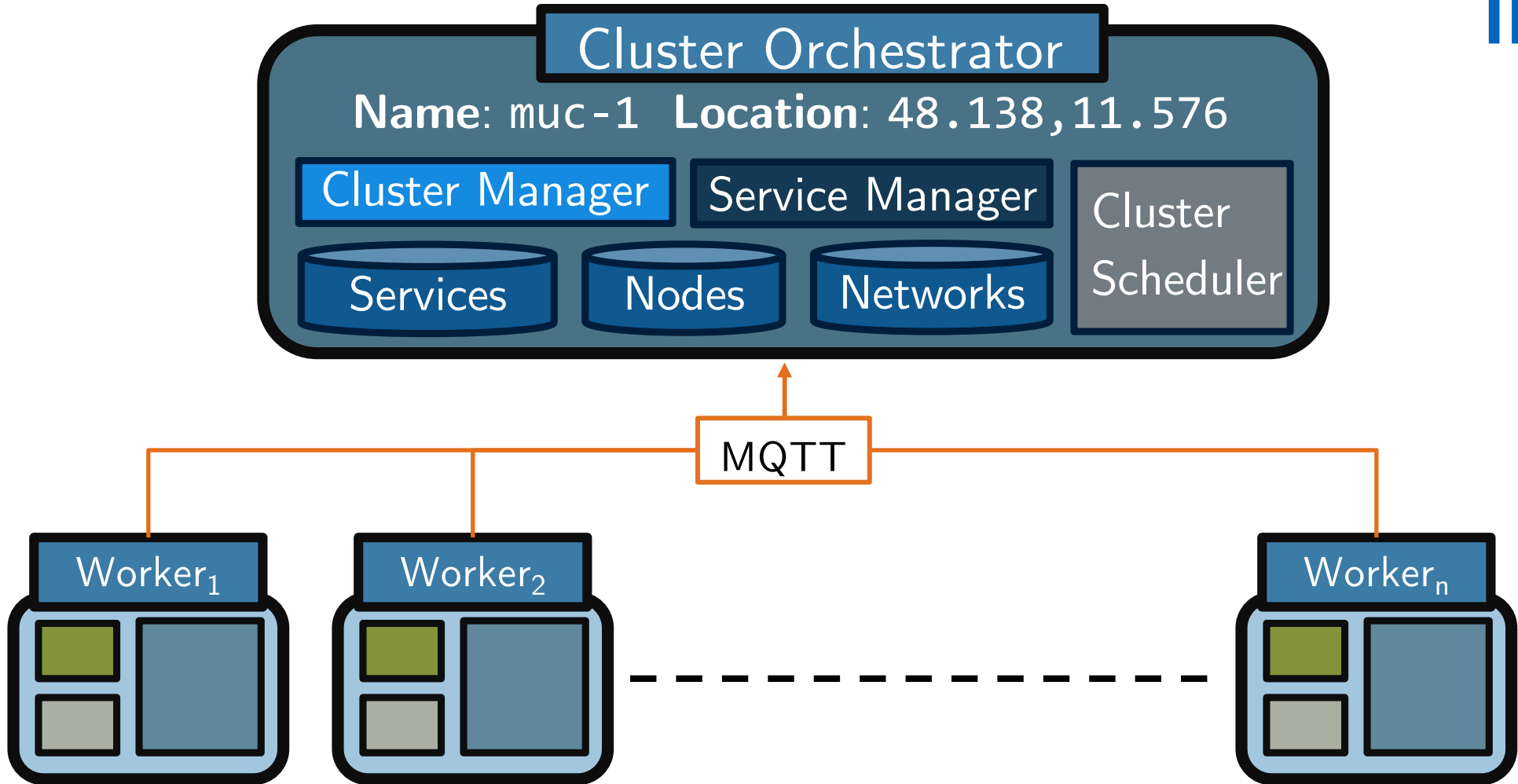
- Autonomously manages service addressing and traffic tunneling
- Creates the network namespaces for the services
- More details later...

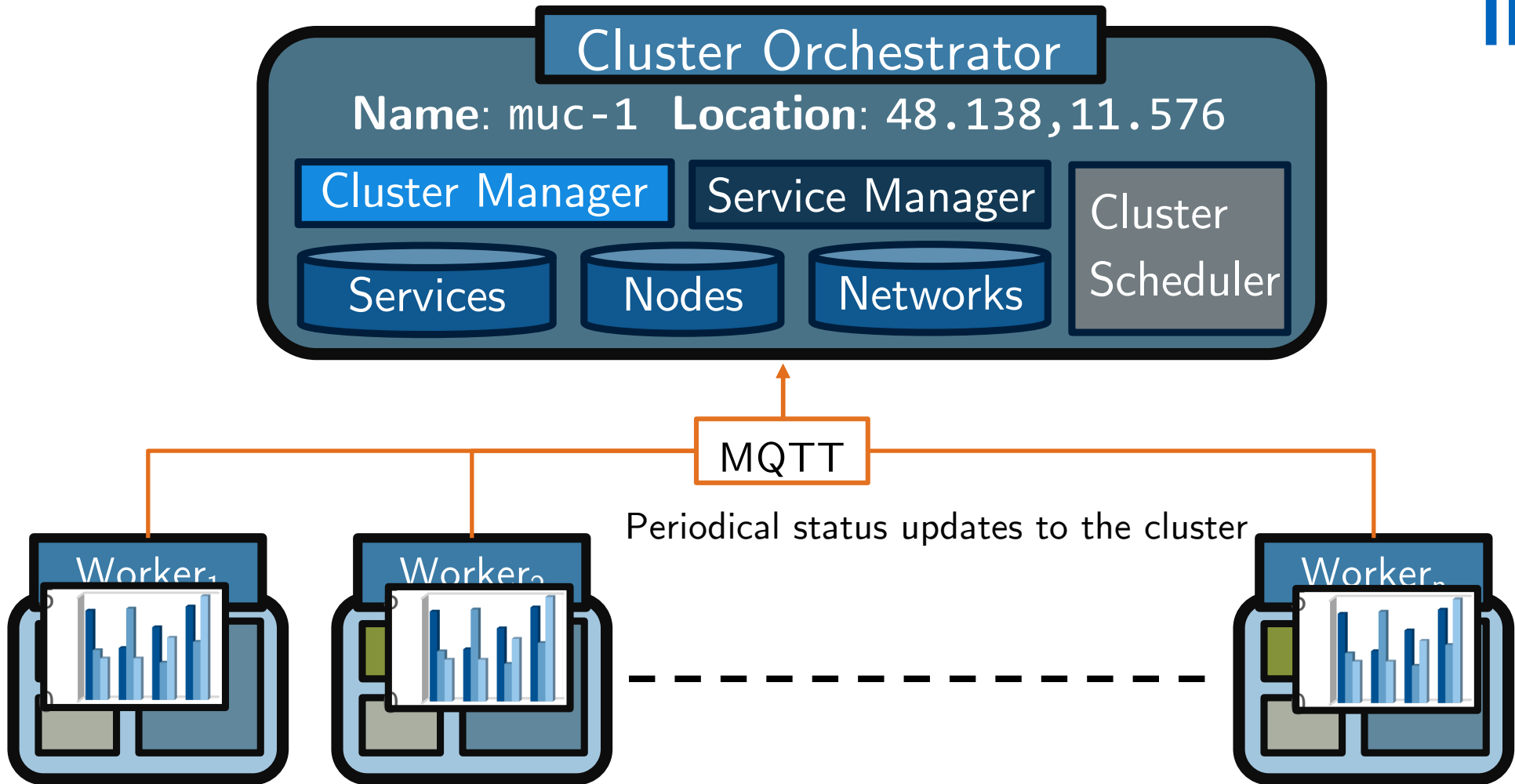
# Worker Node

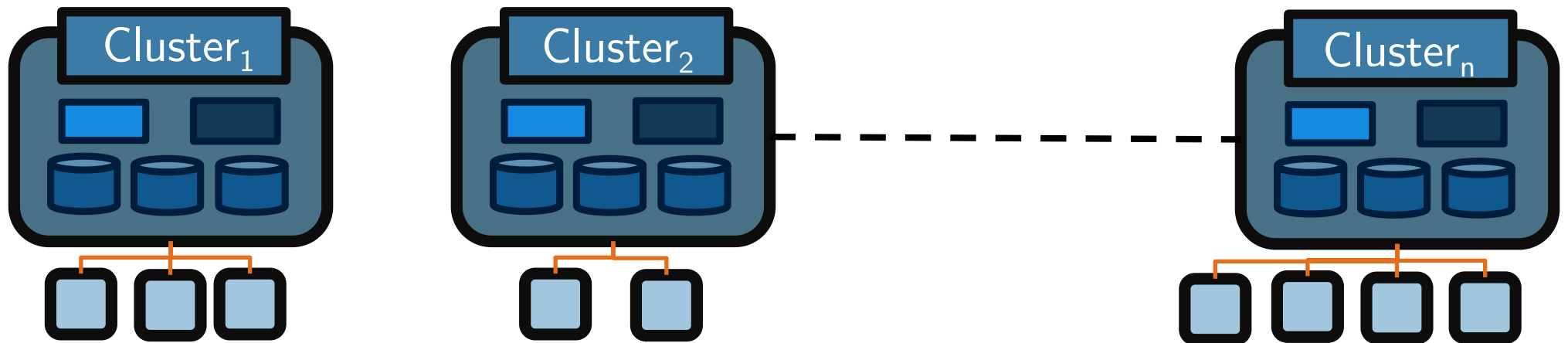


A cluster can be composed of multiple nodes

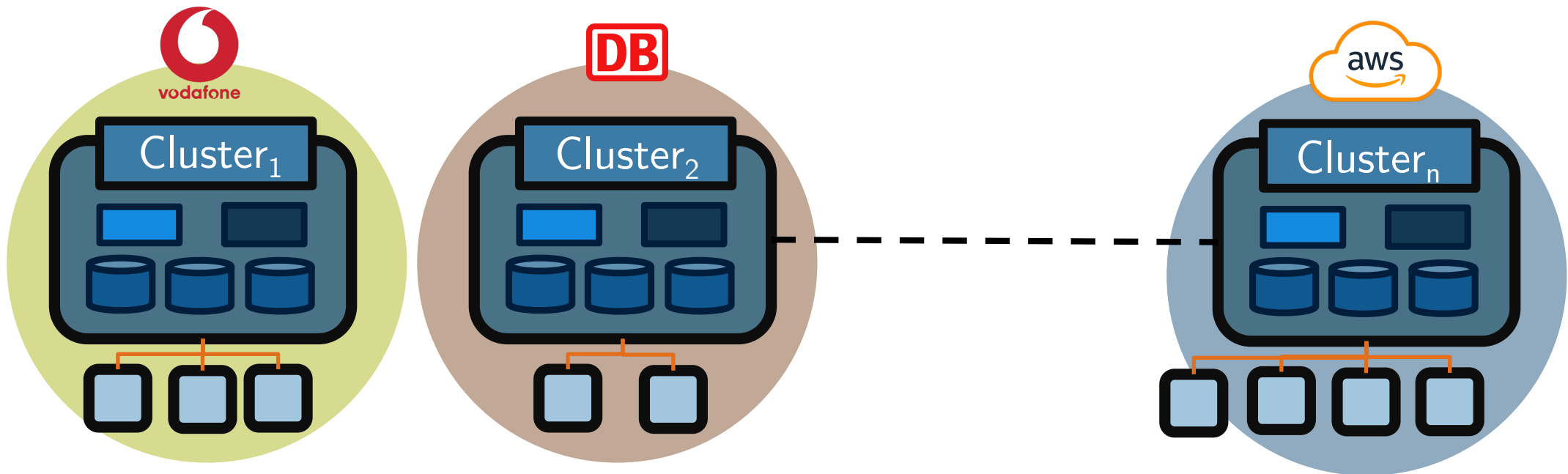


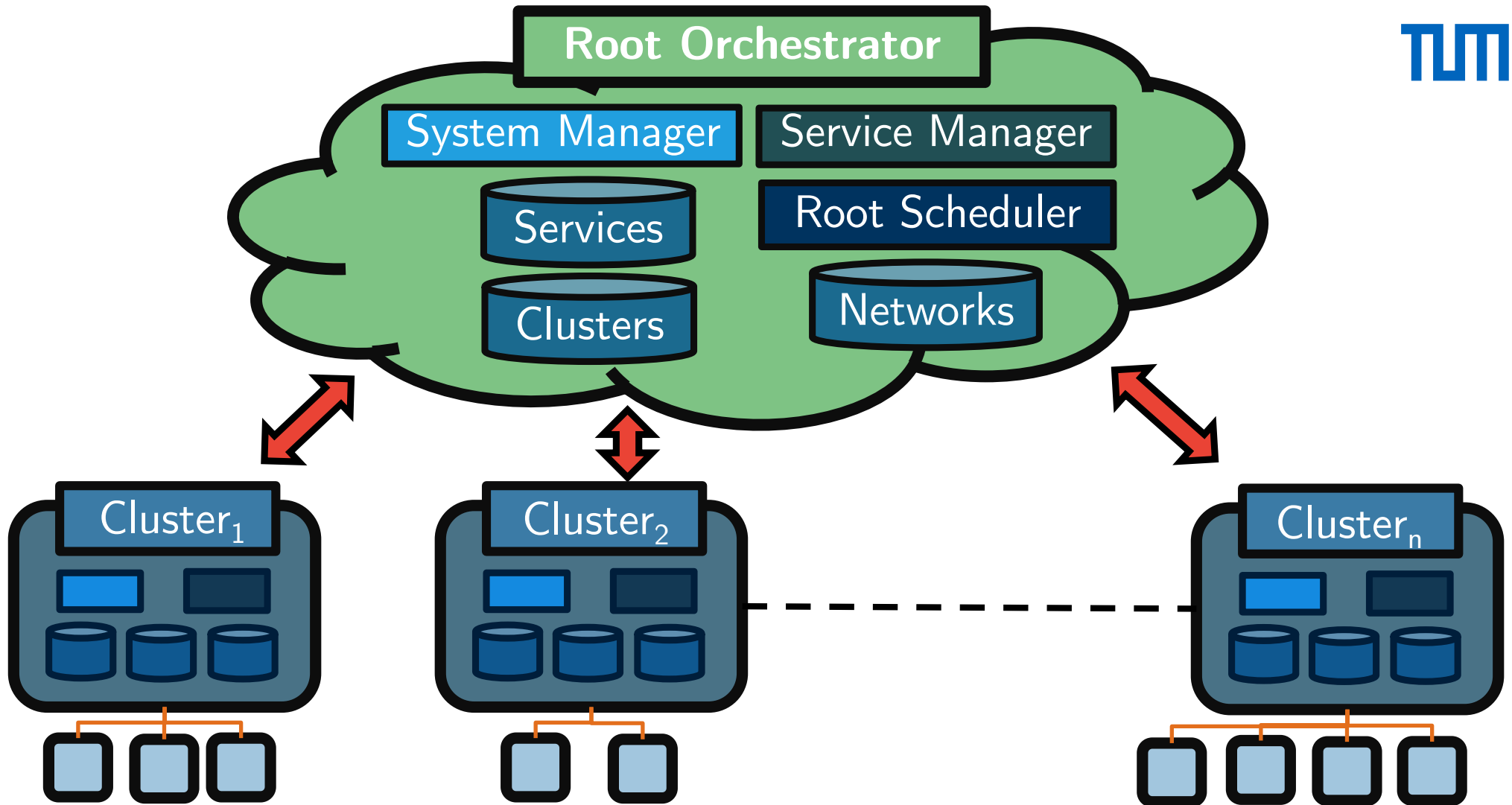


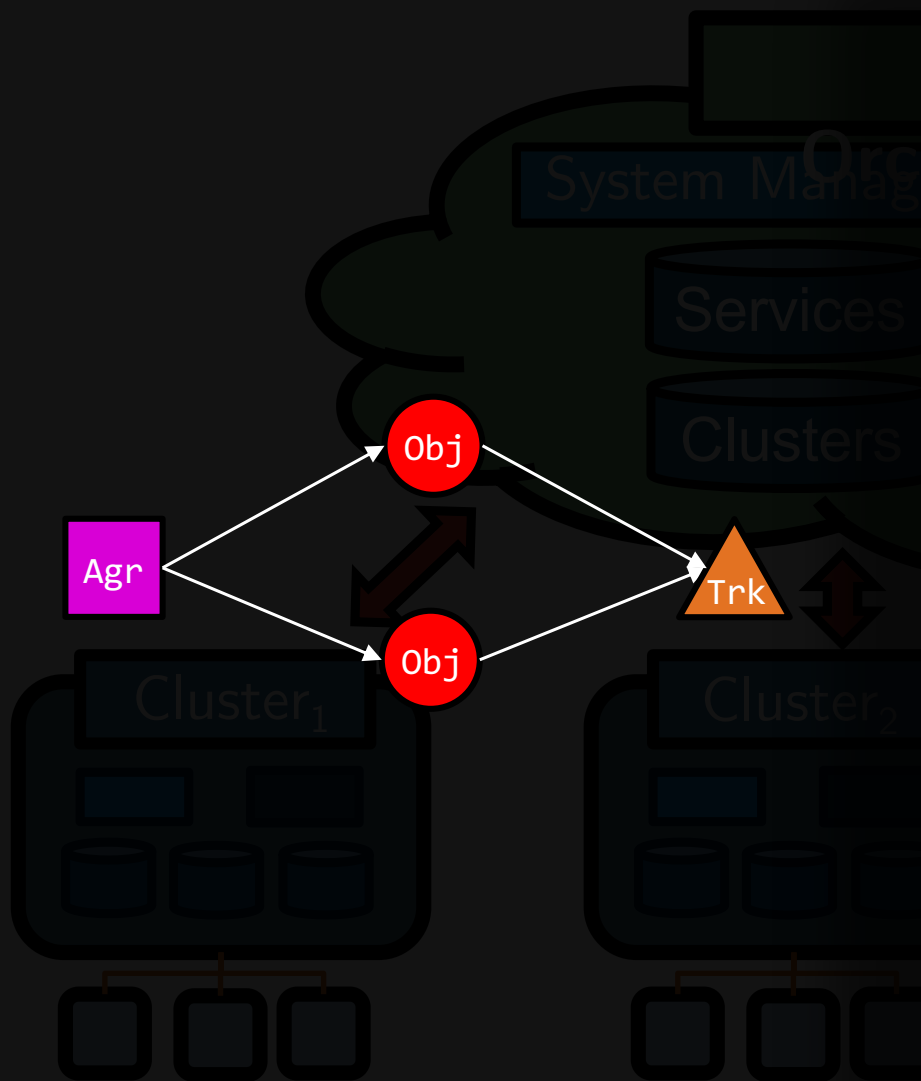




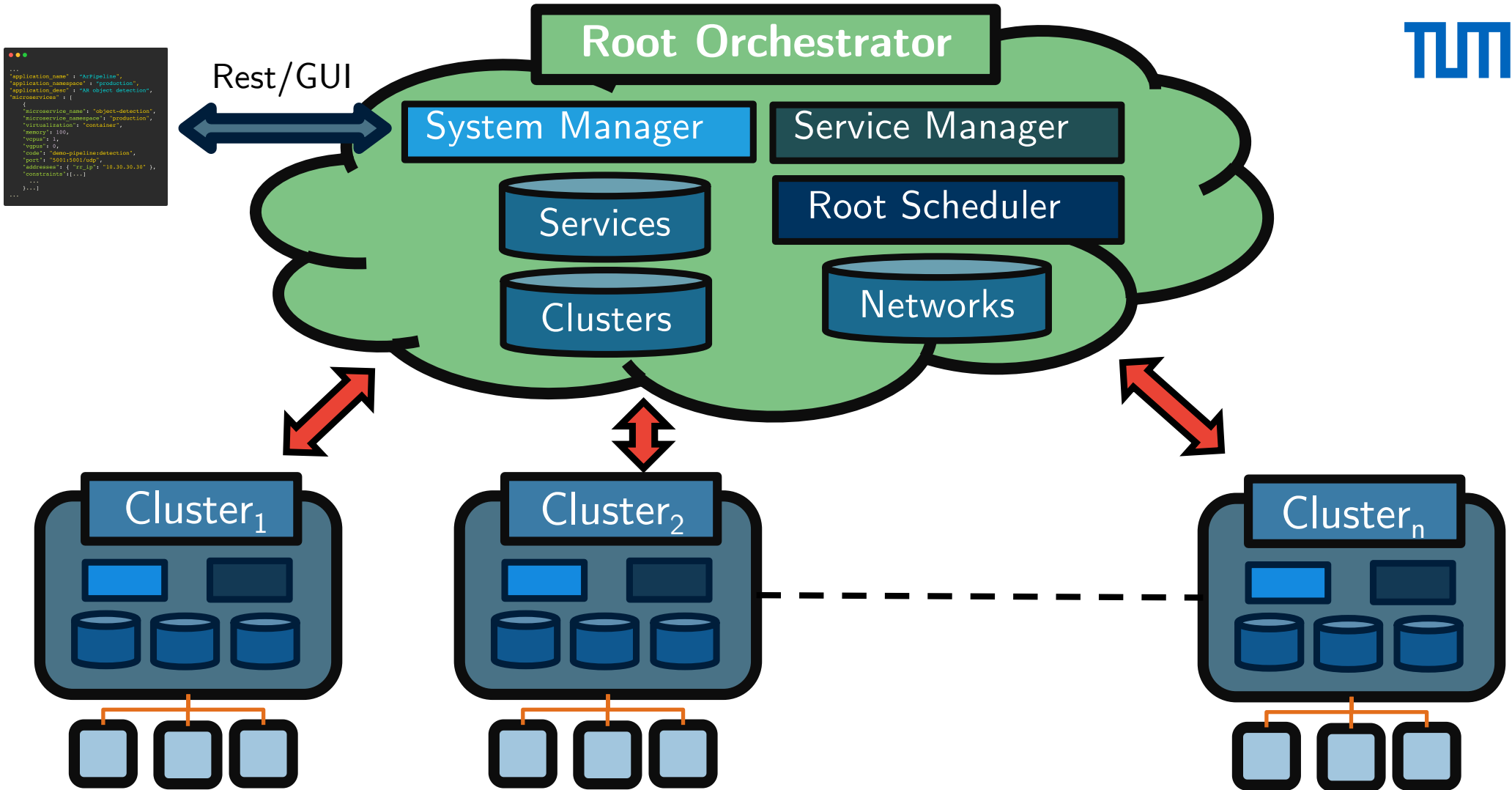
- Different clusters can be administrated by different providers
- Resource aggregation preserves minute details about internal infrastructure make-up



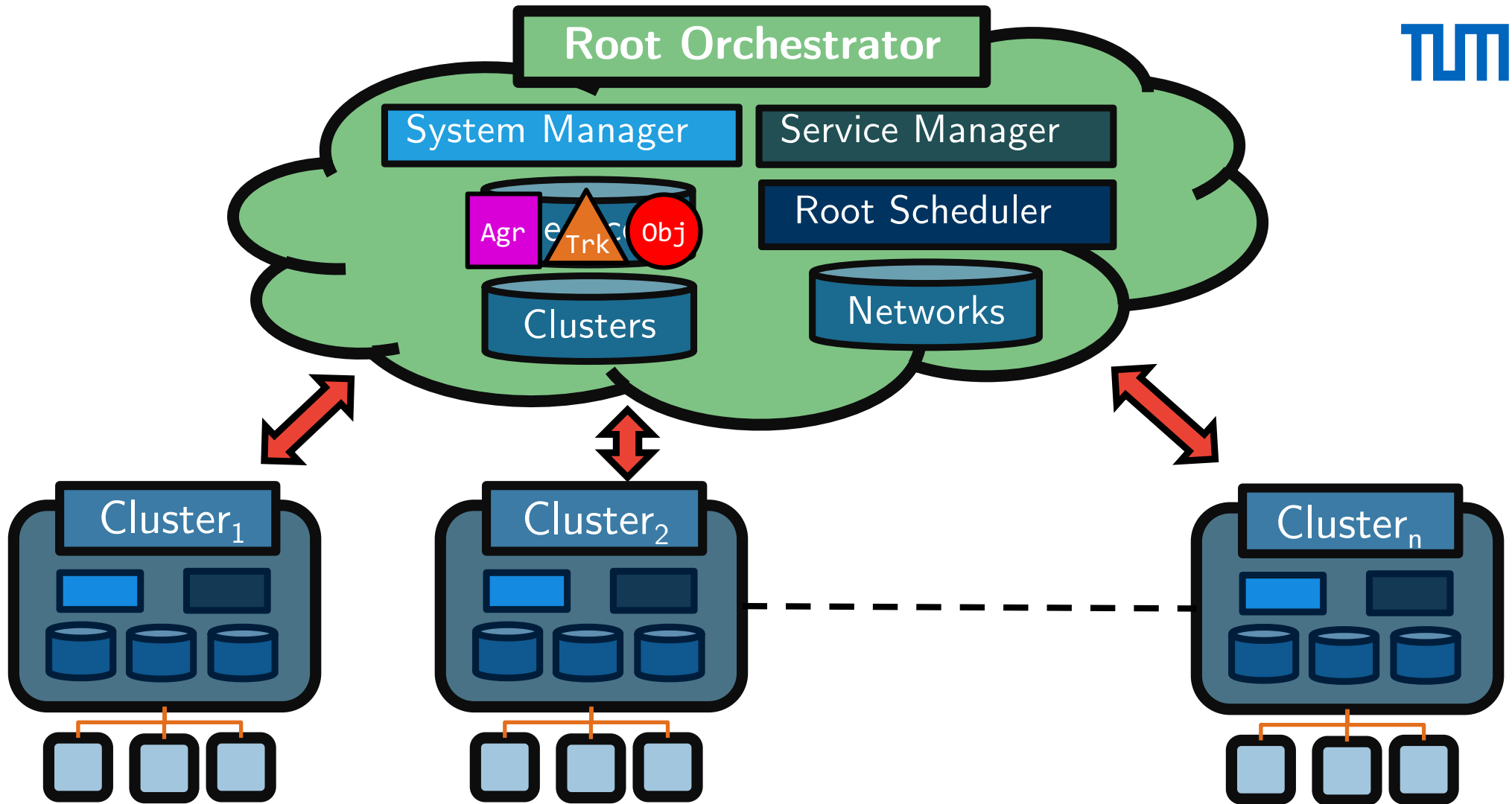


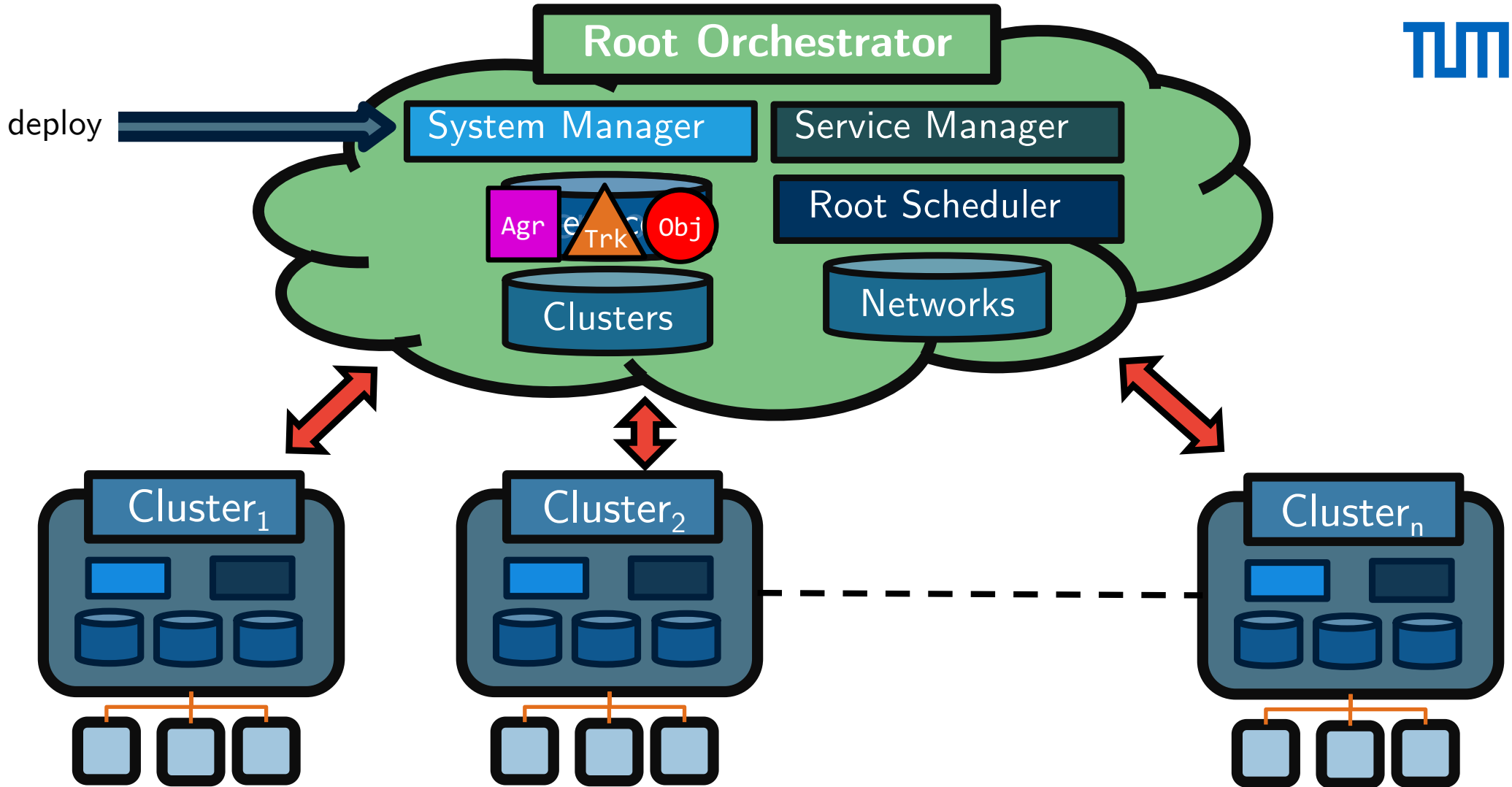


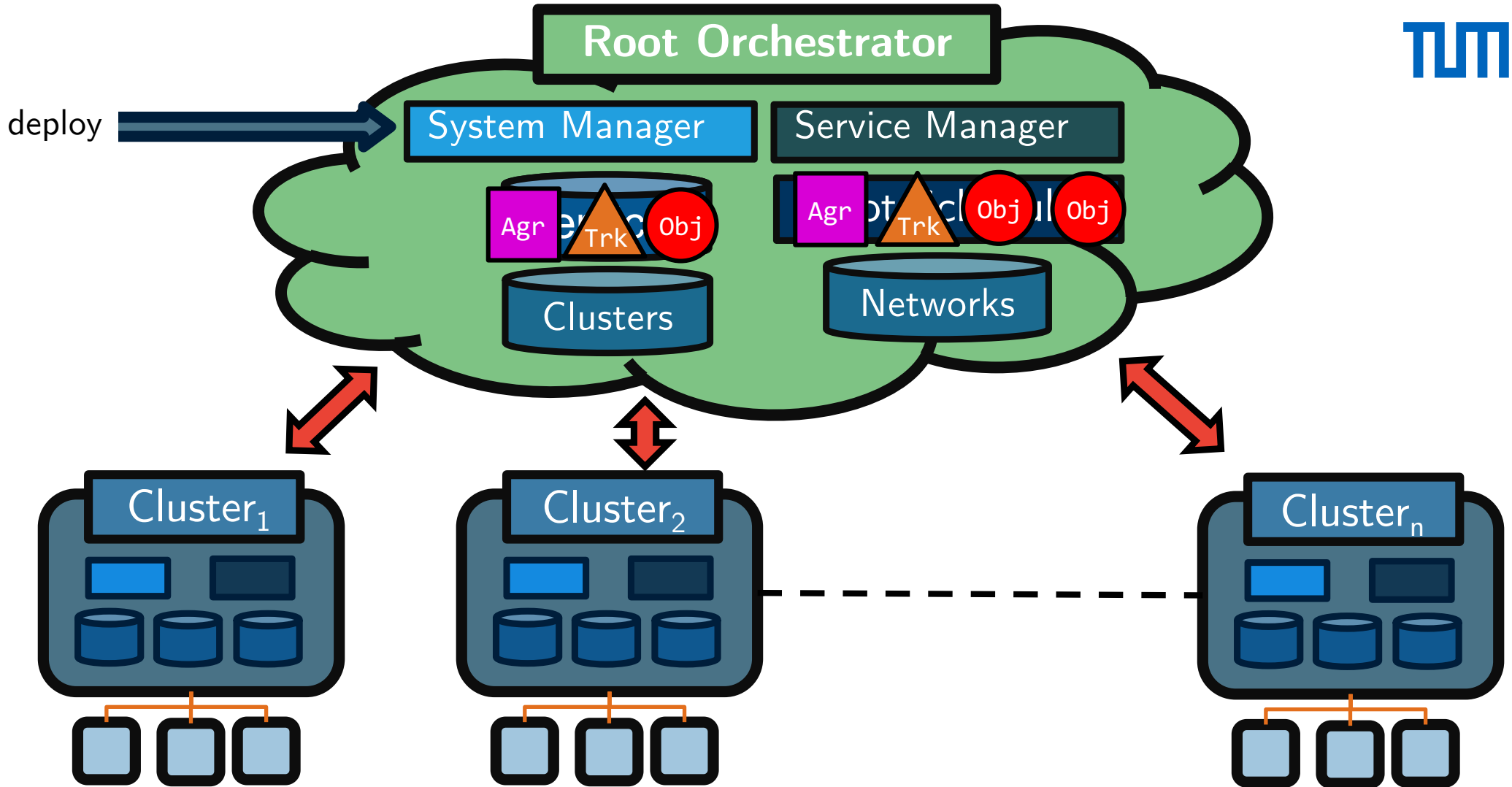
```
...  
"application_name" : "ArPipeline",  
"application_namespace" : "production",  
"application_desc" : "AR object detection",  
"microservices" : [  
  {  
    "microservice_name": "object-detection",  
    "microservice_namespace": "production",  
    "virtualization": "container",  
    "memory": 100,  
    "vcpus": 1,  
    "vgpus": 1,  
    "code": "demo-pipeline:detection",  
    "port": "5001:5001/udp",  
    "addresses": { "rr_ip": "10.30.30.30" },  
    "constraints":[...]  
    ...  
  }...]  
...
```

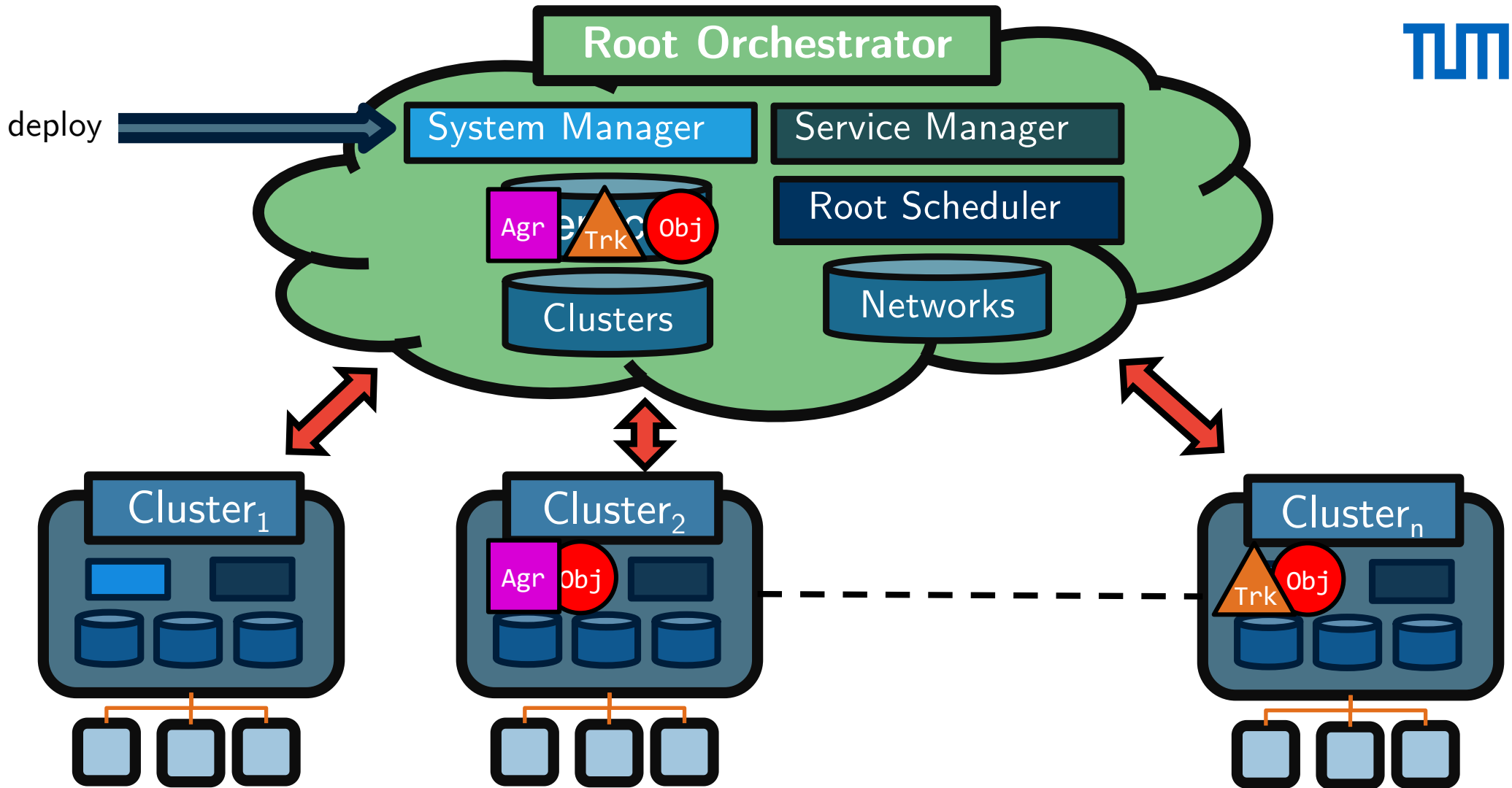


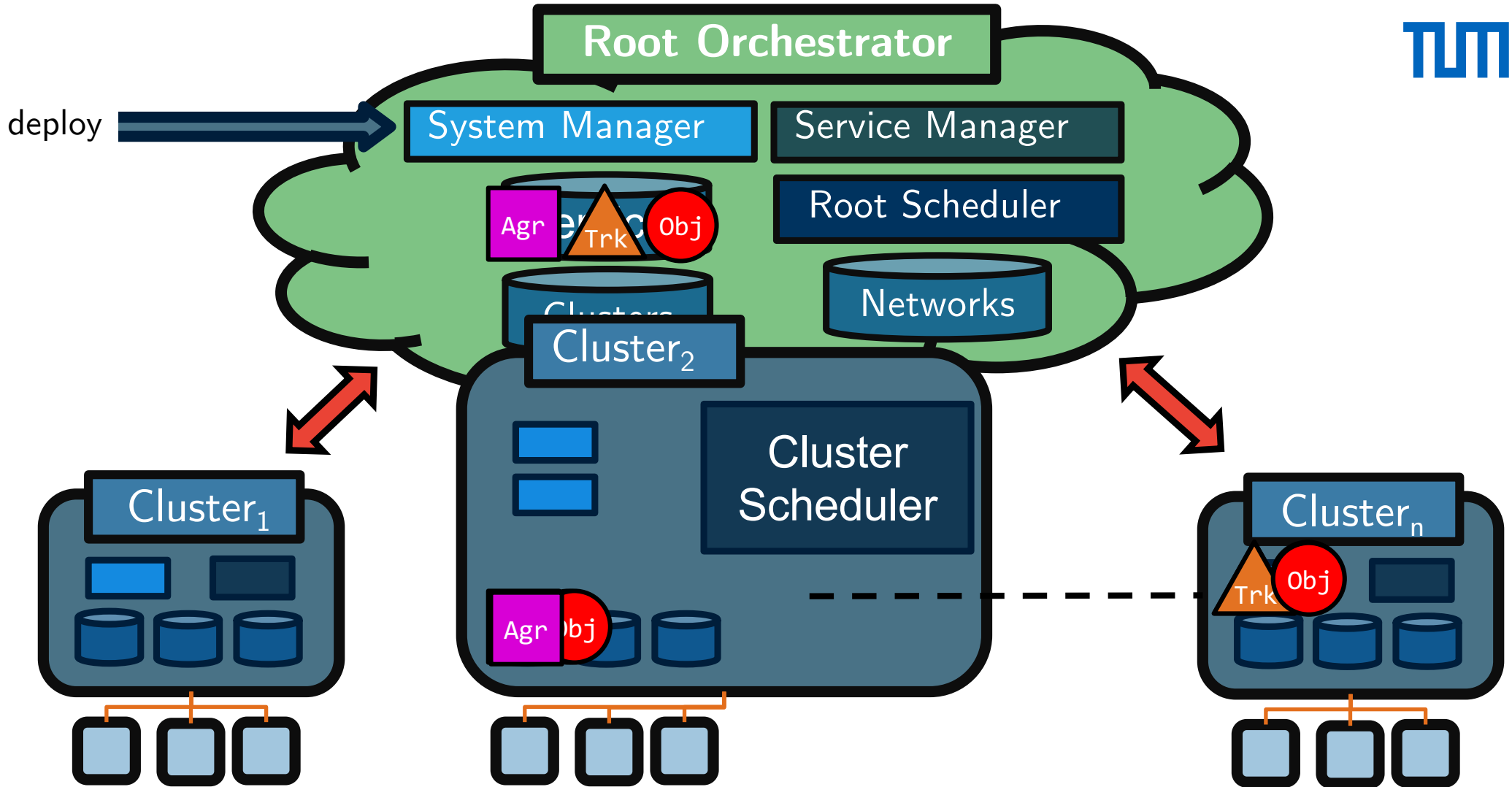


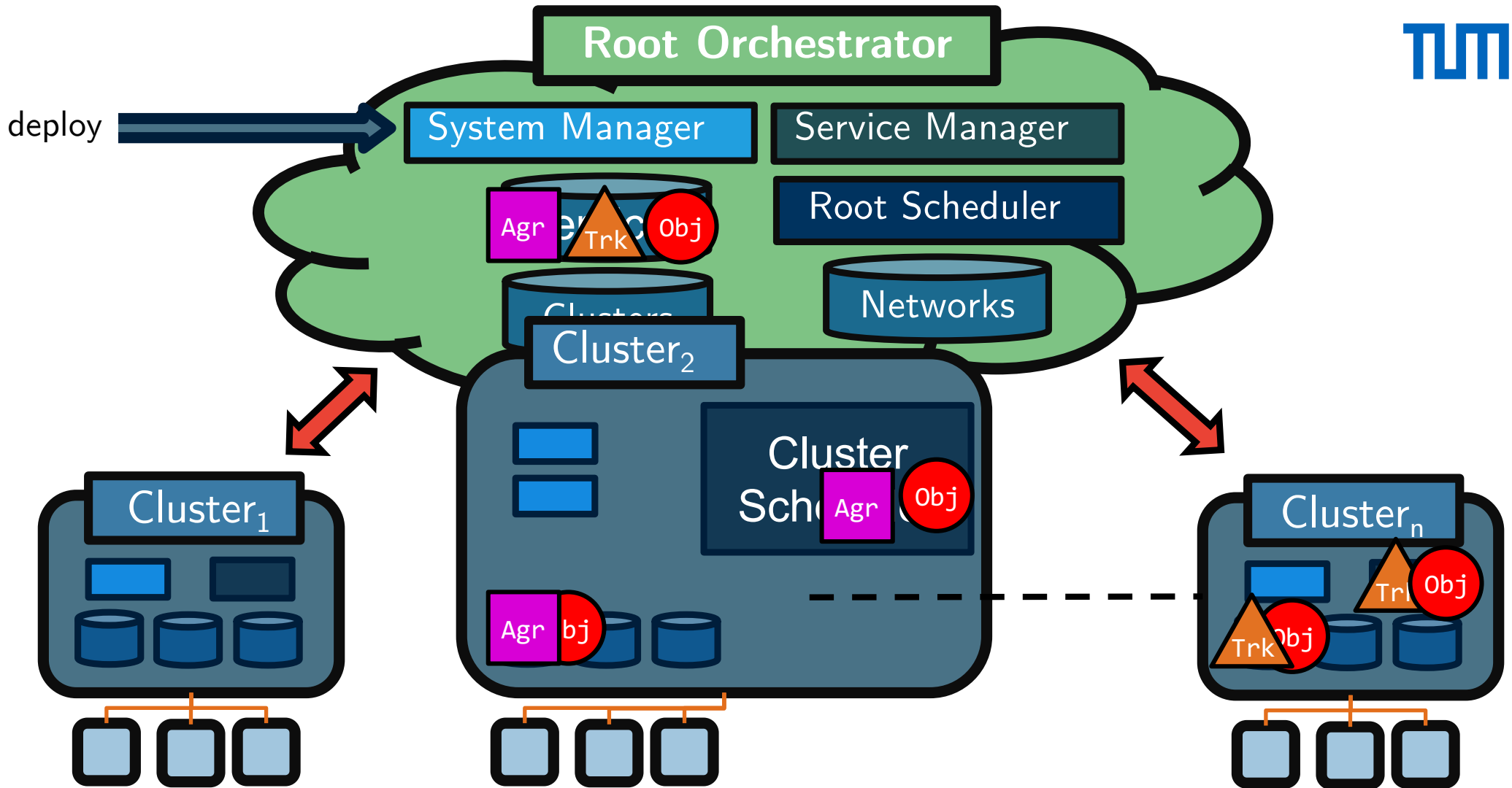


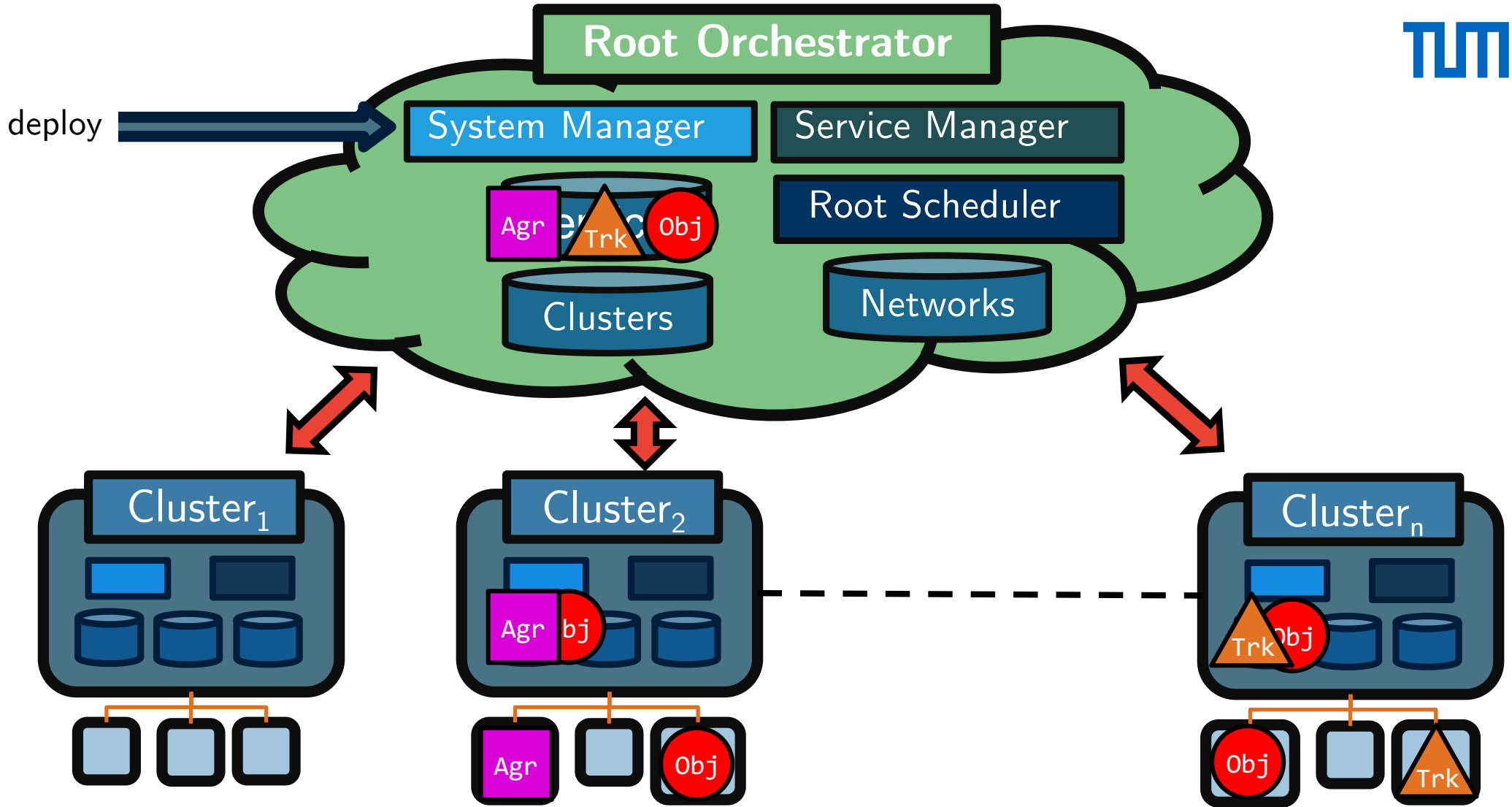








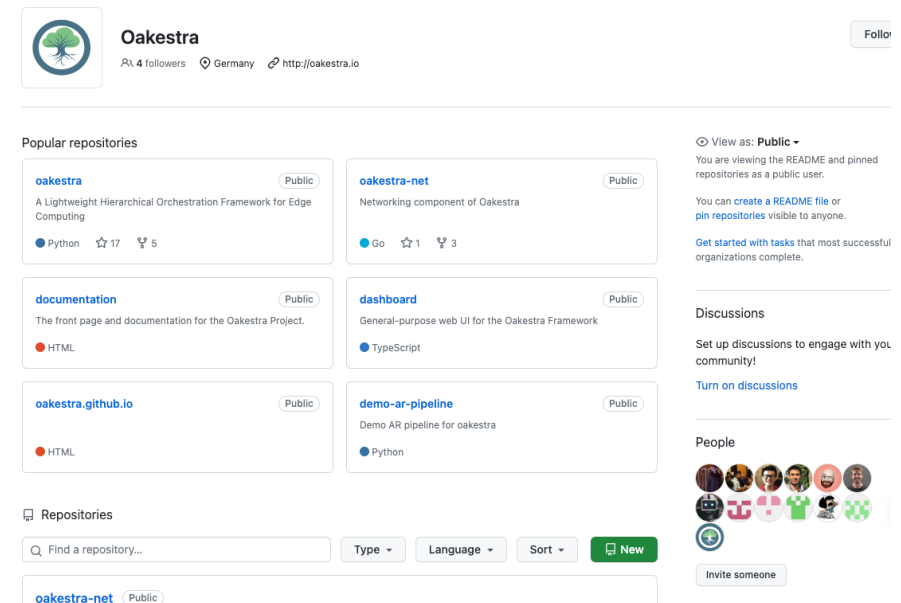




# Implementation

Support for **heterogeneity** and **hardware constraints** by implementation

- Open source
- Currently some 19,000 LOC
- Mainly Python and GoLang
- Modular and extensible
- Lightweight to embrace low end edge devices
- Support for Linux and Docker container-ized services; unikernels in progress
- Performance compares favorably to k8s, k3s, MicroK8s



The screenshot shows the GitHub organization page for Oakestra. At the top, the organization name "Oakestra" is displayed with a tree logo, 4 followers, location in Germany, and the website URL http://oakestra.io. Below this, a "Popular repositories" section lists several repositories:

- oakestra**: A Lightweight Hierarchical Orchestration Framework for Edge Computing. Language: Python. 17 stars, 5 forks.
- oakestra-net**: Networking component of Oakestra. Language: Go. 1 star, 3 forks.
- documentation**: The front page and documentation for the Oakestra Project. Language: HTML.
- dashboard**: General-purpose web UI for the Oakestra Framework. Language: TypeScript.
- oakestra.github.io**: Language: HTML.
- demo-ar-pipeline**: Demo AR pipeline for oakestra. Language: Python.

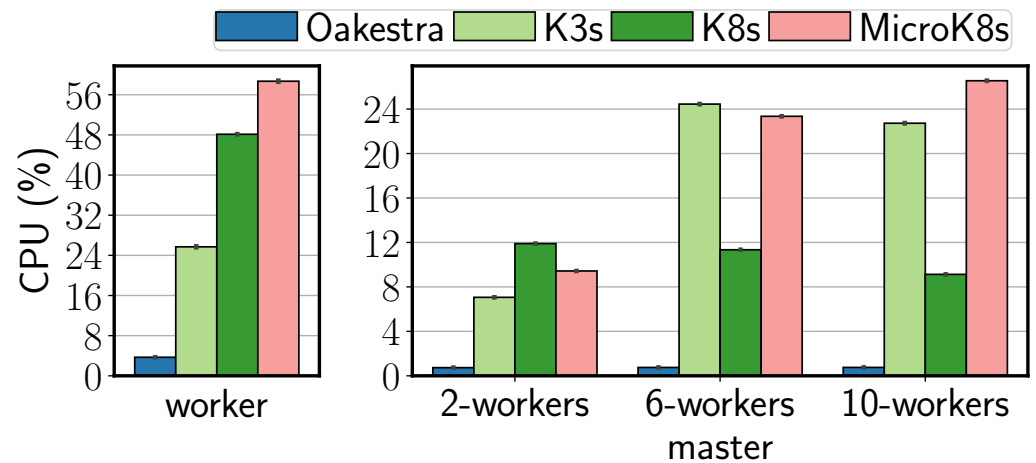
Below the popular repositories, there is a "Repositories" section with a search bar and filters for Type, Language, and Sort. A single repository, **oakestra-net**, is listed. On the right side, there are sections for "Discussions" (with a link to "Turn on discussions") and "People" (showing a grid of user avatars).

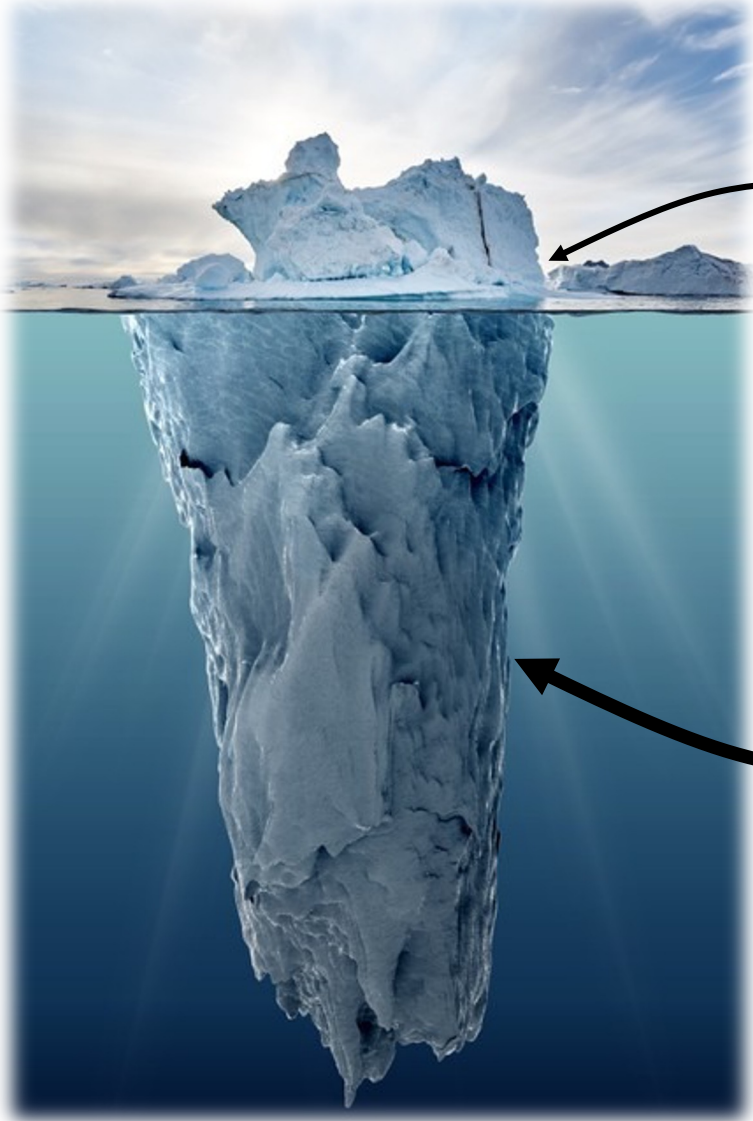
# System Performance



## Constrained Hardware

- 6x CPU% reduction at worker level compared to K3s
- 10x CPU% reduction at master level compared to K3s





# Questions?



This presentation



## Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing

Giovanni Bartolomeo<sup>2</sup> Mehdi Yosofe<sup>2</sup> Simon Bäurle<sup>2</sup> Oliver Haluszczynski<sup>2</sup>  
Nitinder Mohan<sup>2</sup> Jörg Ott<sup>2</sup>  
Technical University of Munich, Germany

{firstname.lastname@tum.de}

<sup>2</sup> Corresponding Authors

 @oakestra

 Oakestra



[oakestra.io](https://oakestra.io)

### Abstract

Edge computing seeks to enable applications with strict latency requirements by utilizing resources deployed in diverse, dynamic, and possibly constrained environments closer to the users. Existing state-of-the-art orchestration frameworks (e.g. Kubernetes) perform poorly at the edge since they were designed for reliable, low latency, high bandwidth cloud environments. We present Oakestra, a hierarchical, lightweight, flexible, and scalable orchestration framework for edge computing. Through its novel federated three-tier resource management, delegated task scheduling, and semantic overlay networking, Oakestra can flexibly consolidate multiple infrastructure providers and support applications over dynamic variations at the edge. Our comprehensive evaluation against the state-of-the-art demonstrates the significant benefits of Oakestra as it achieves approximately tenfold reduction in resource usage through reduced management overhead and 10% application performance improvement due to lightweight operation over constrained hardware.

### 1 Introduction

Within almost a decade since its inception, edge computing has found a wide range of use cases in industry and research, especially for supporting latency-critical services like AR/VR [24], live video analytics [14], etc. [46]. However, despite significant interest, there have only been a handful of real-world demonstrations of edge so far [49]. Reasons for this are manifold and may include, on the technical side, the following. Firstly, resources at the edge are far less capable and

Secondly, the majority of popularly used orchestration frameworks, e.g., Kubernetes [34], K3s [29], KubeFed [35], etc., are off-shoot branches of solutions that were inherently designed to perform well in managed datacenter networks. Such frameworks make strong assumptions about the underlying infrastructure's (especially the network's) consistent reliability and reachability, which does not necessarily hold at the edge where resources are more dispersed. For example, recent investigations into Kubernetes' operations uncovered that its reliance on maintaining strong consistency in the datastore via etcd along with its limited scalability results in severe availability and efficiency issues in edge-like environments [37]. Moreover, the core components of such frameworks incorporate many heavyweight operations – limiting their use on constrained hardware. Furthermore, almost none of the existing solutions can currently support the edge's heterogeneity in hardware, networking, and resource availability.

In this work, we present Oakestra, a flexible, hierarchical orchestration framework that overcomes the many challenges just mentioned. Conceptually, Oakestra allows multiple operators over vast geographical regions to contribute their resources to a federated infrastructure – reducing the investment to achieve a dense computing fabric at the edge. Furthermore, Oakestra's implementation is lightweight and extensible, allowing it to manage effectively constrained and heterogeneous edge infrastructures. Specifically, our contributions are:

(1) We consolidate edge infrastructures in a logical three-tier hierarchy. With a root orchestrator managing many resource clusters, each controlled by a cluster orchestrator, we enable infrastructure federation. The cluster orchestrator exercises

Bartolomeo *et al.*: Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing, USENIX ATC 2023.