In the name of god


# container orchestration

## software architectures

Professor: Dr Mehrdad Ashtiani

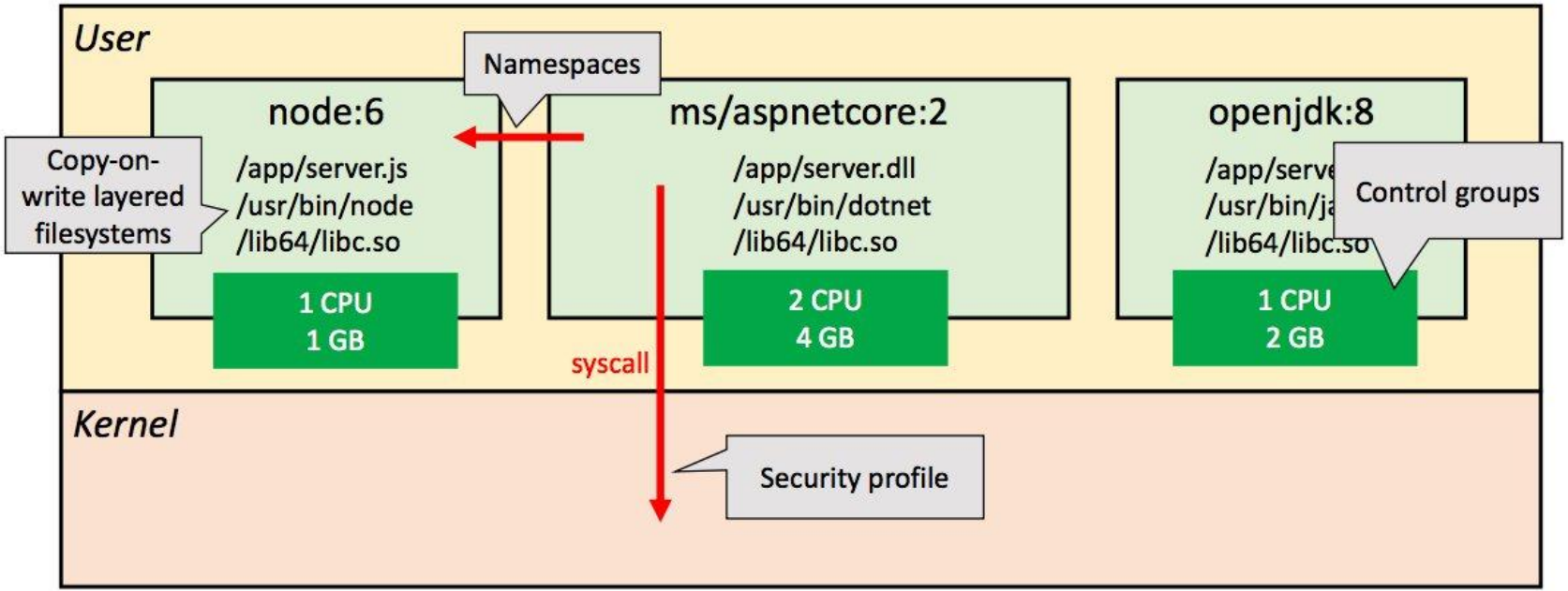
Mehran Salmani - Mostafa Mohammad Ali Ebrahim

# table of contents

# What are containers?

- isolated user space instances running on a host
- so the term OS-level virtualization is used for container virtualization
- they can only run on similar guest operating systems as the underlying host
- less overhead, more portable, more scalable, fast start up, ...

# Linux containers




Linux container building blocks

Image from sasha goldshtein dotNext talk

# container orchestration

automatic process of managing or scheduling the work of individual containers for applications based on microservices within multiple clusters.

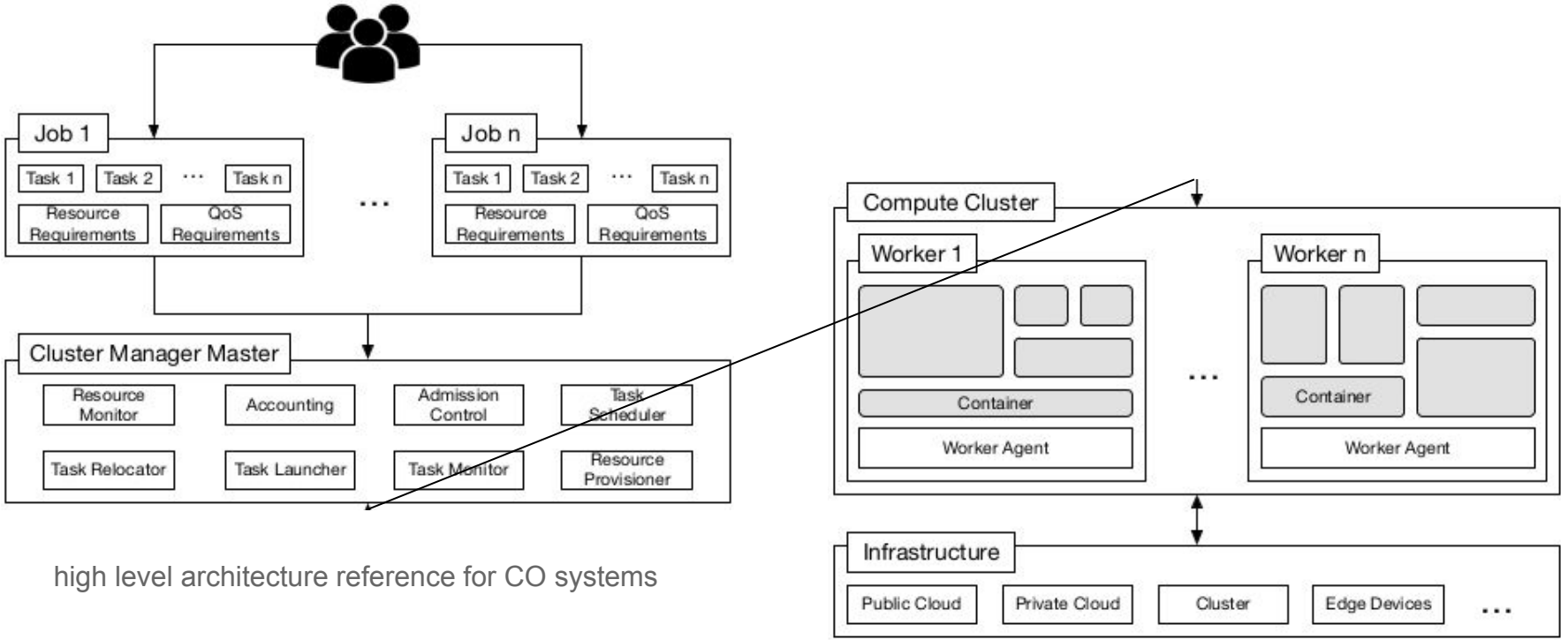
# Why to use container orchestration?

- scheduling of containers
- deployments of containers
- availability and Health monitoring of containers
- scaling of containers

# Why to use container orchestration? (contd)

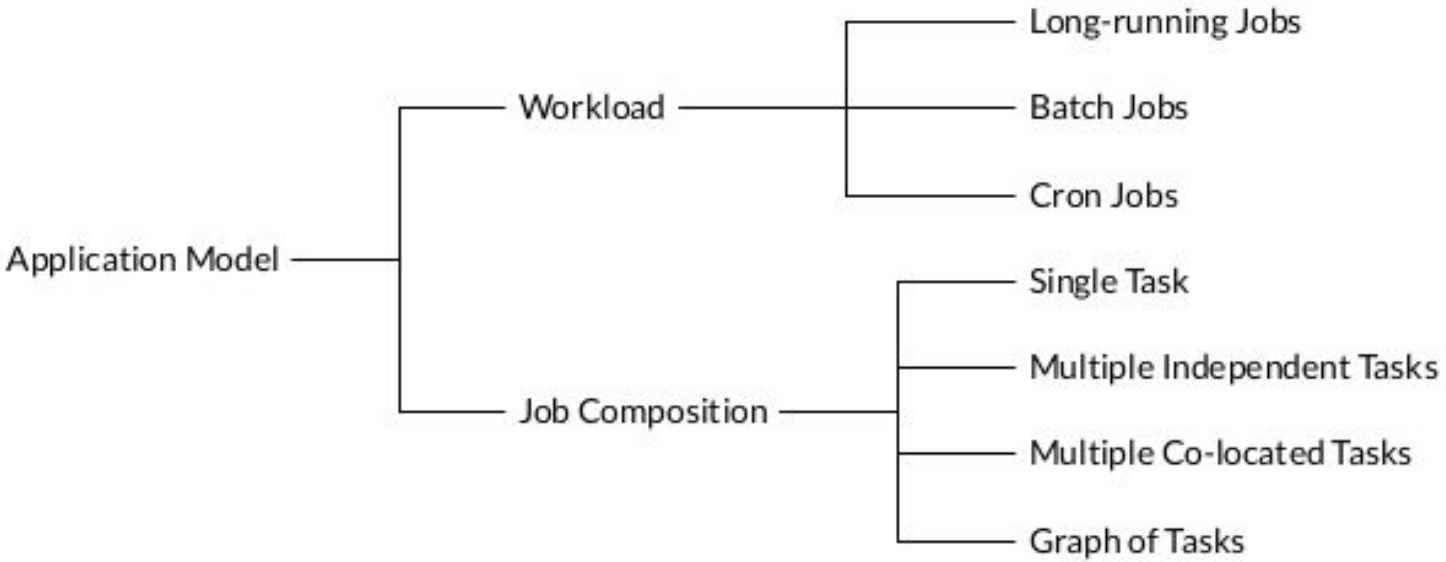
- allocation of resources between containers
- load balancing and service discovery of containers
- securing the interactions between containers.

# container orchestration system architecture




high level architecture reference for CO systems

# application models




application model classification

# scheduling: architecture

- Centralized
  - Pros: better optimization decisions
  - Cons: single point of failure - scalability/load issues
- Decentralized (either monolithic or modular)
  - Pros: scalable
  - Cons: needs request partitioning and state management policies

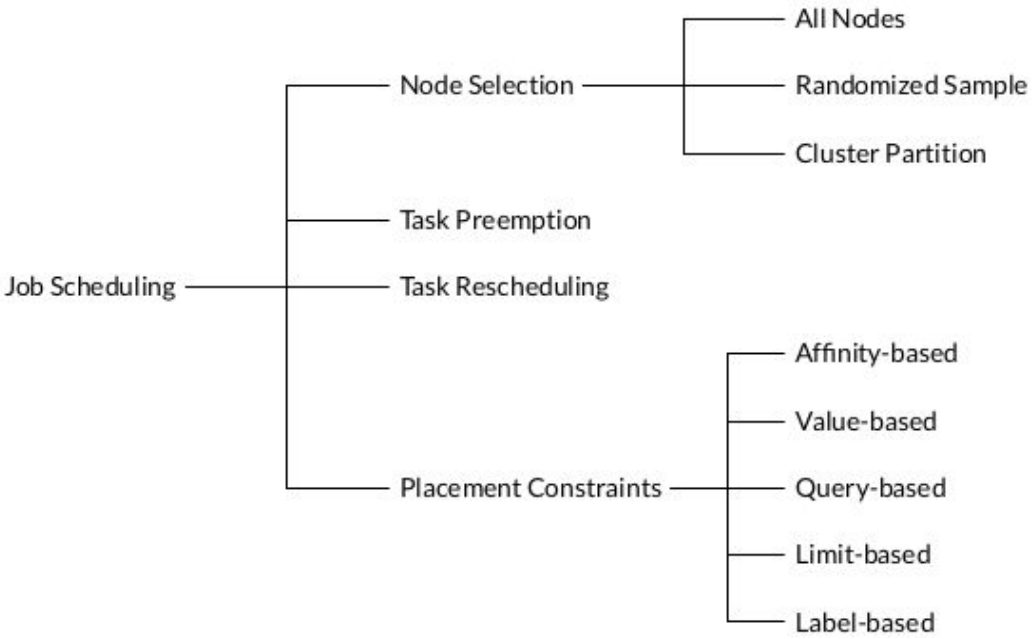
# scheduling: state management in decentralized

- Shared state
  - optimistic parallelism
  - requires concurrency control
- Partial view
  - no conflicts between schedulers

# scheduling: two level architecture

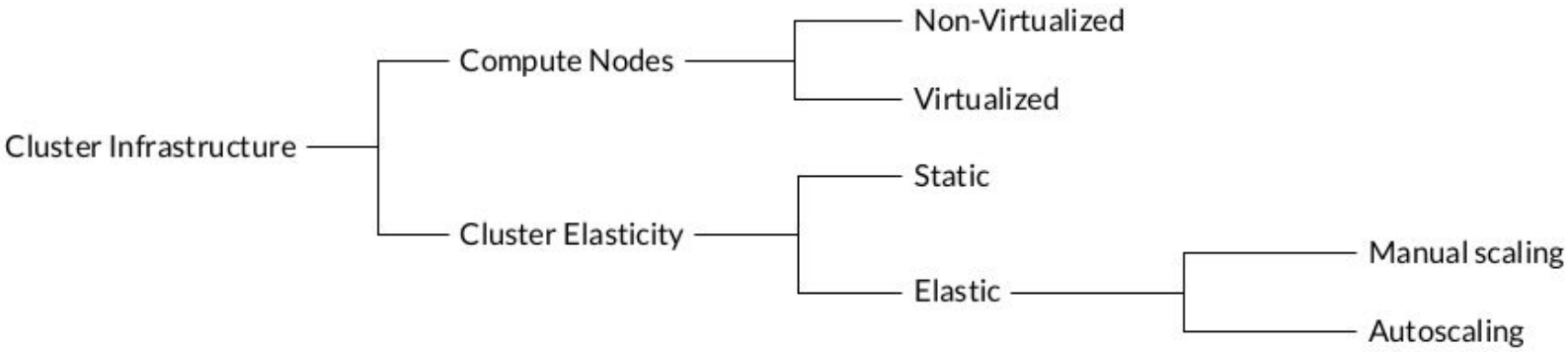
- Scheduler framework layer
  - responsible for placement decisions
- Resource management layer
  - responsible for managing cluster resources
  - offer-based or request based

# scheduling: job scheduling



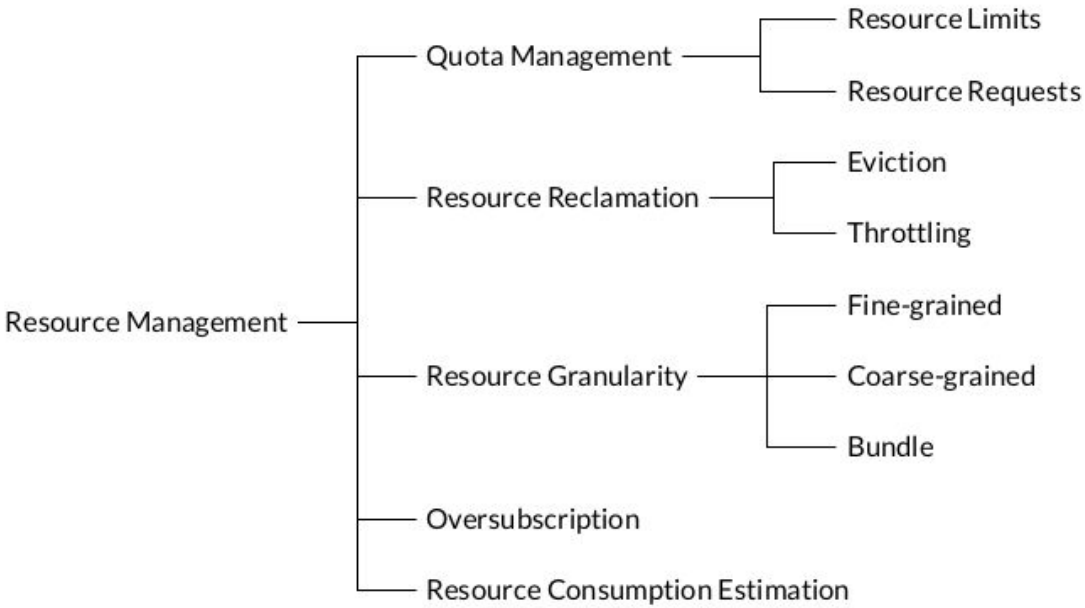

job scheduling taxonomy

# cluster infrastructure



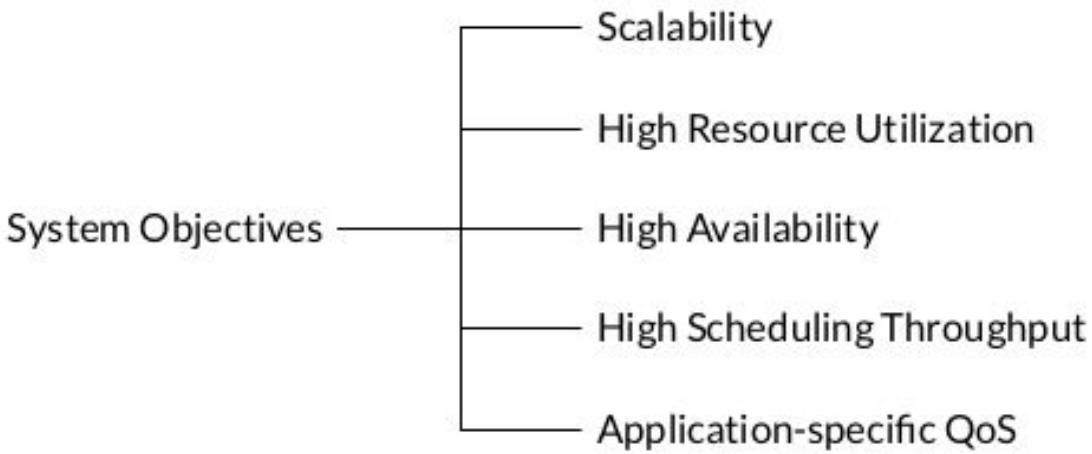

cluster infrastructure taxonomy

# resource management




resource management taxonomy

# system objectives




Orchestration system objectives

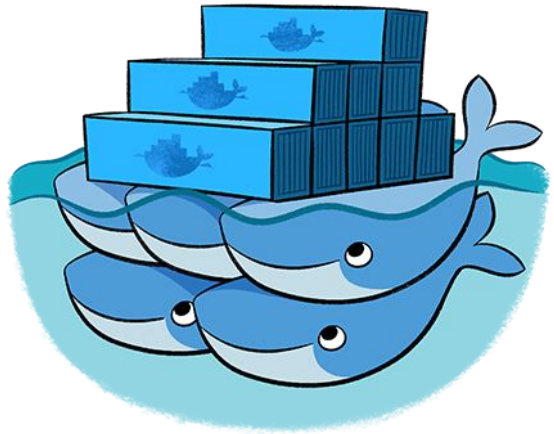
# table of contents (part 2)

# Kubernetes

- Developed by Google (Offshoot of Borge)
- Popularity
- Main Architecture Components
  - Cluster
  - Kubernetes master
  - Kubelet
  - Pods
  - Deployments, replicas, and ReplicaSets

# Docker Swarm

- Fully Integrated
- Less Extensible and Complex
- Main Architecture Components
  - Swarm
  - Service
  - Manager Node
  - Worker Node
  - Task



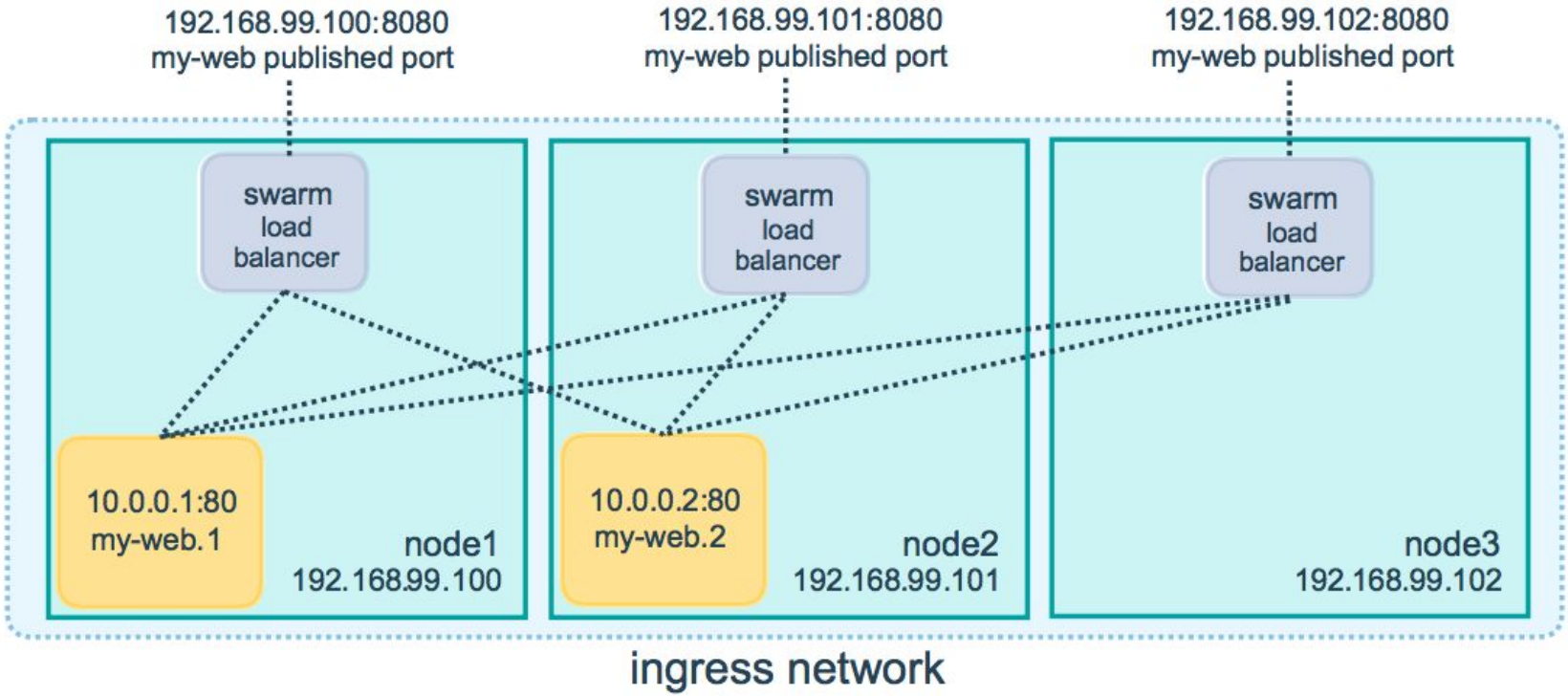
# Apache Mesos (and Marathon)

- Older Than Kubernetes
- Lightweight Interface
- Marathon a “production-grade”
- Main Architecture Components
  - Master daemon and agent daemon
  - Framework
  - Offer
  - Task

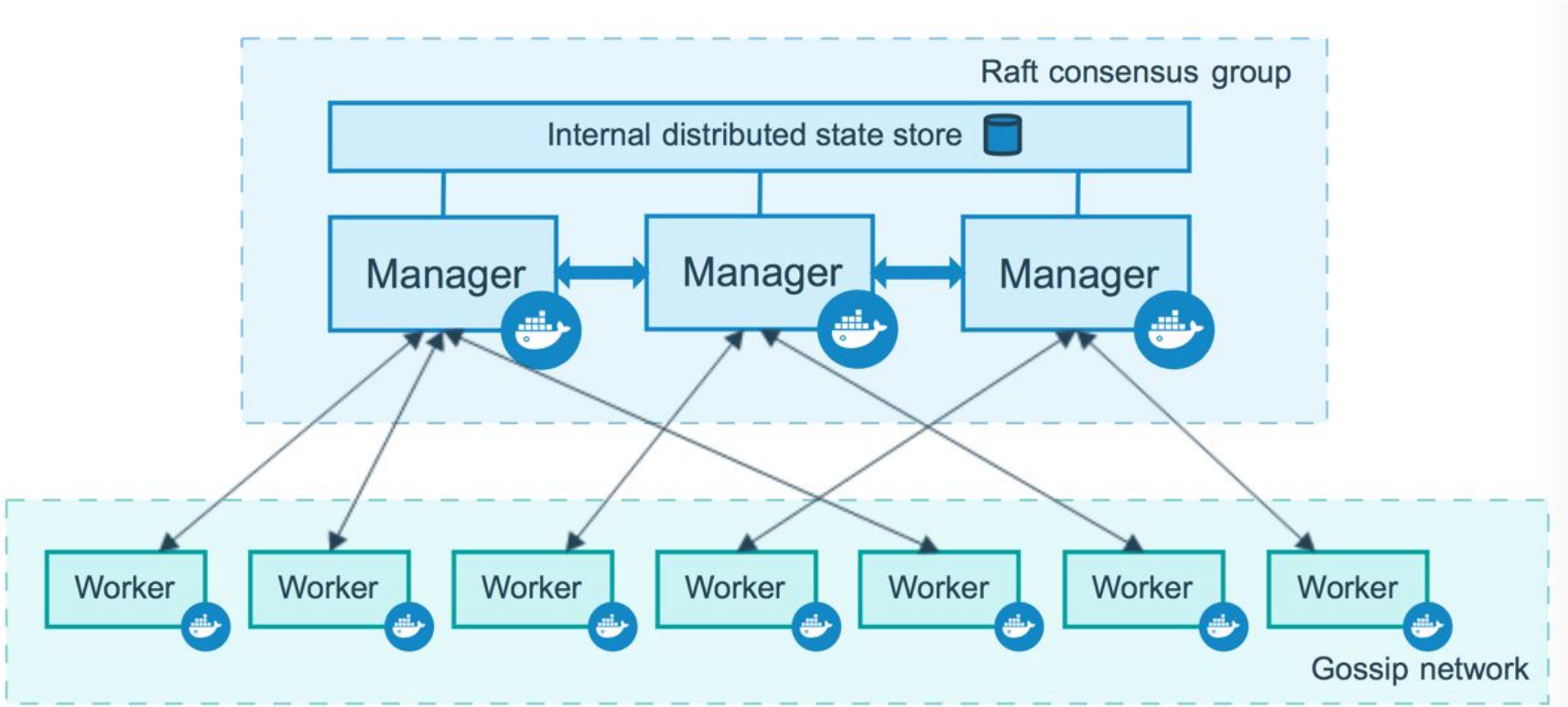
# key concepts

- What is a swarm?
  - The cluster management and orchestration
  - Consists of multiple nodes: Managers and Worker
  - Task vs standalone container
  - Difference
- Nodes
  - Manager nodes and worker nodes
- Services and tasks
- Load balancing
  - ingress load balancing and internal load balancing

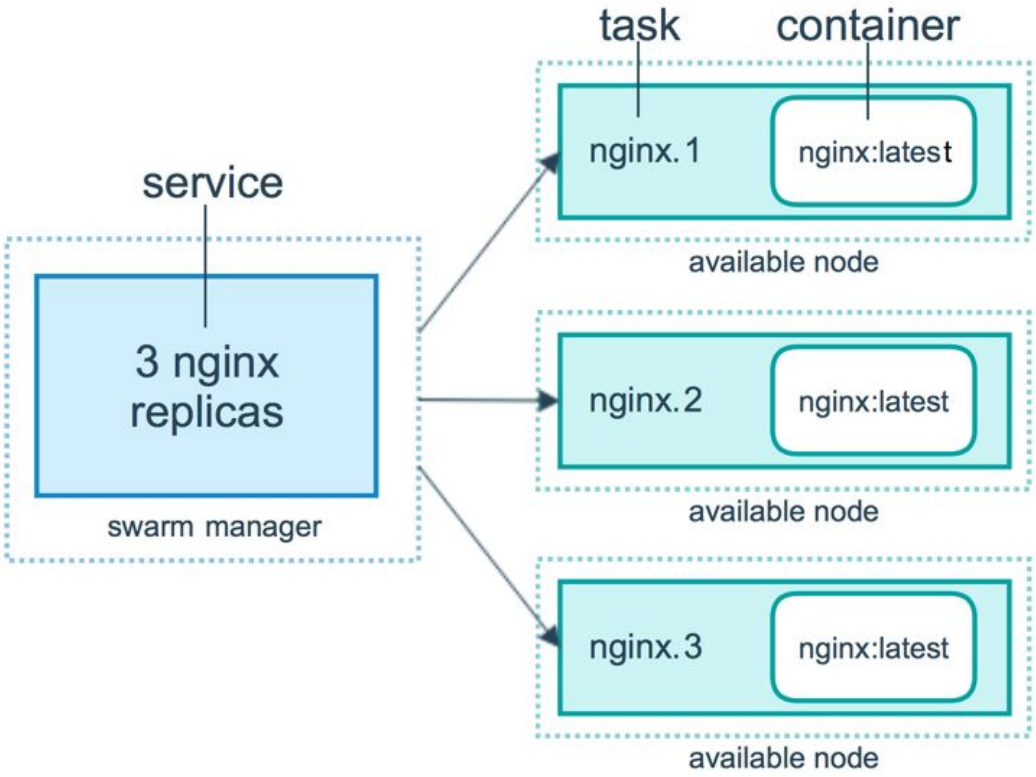
# routing mesh



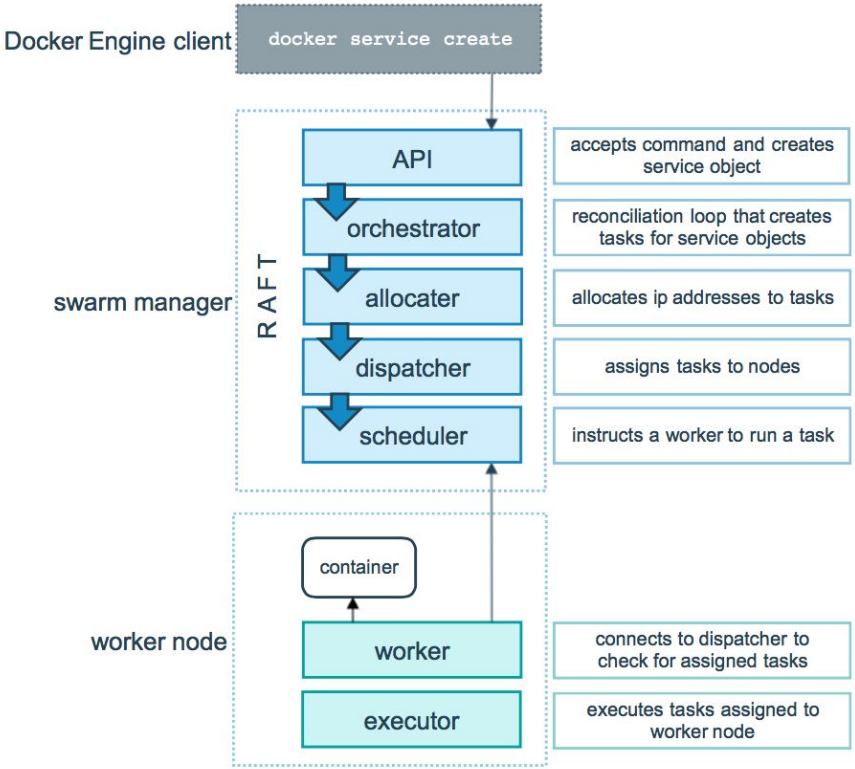
# How nodes work



# How services work

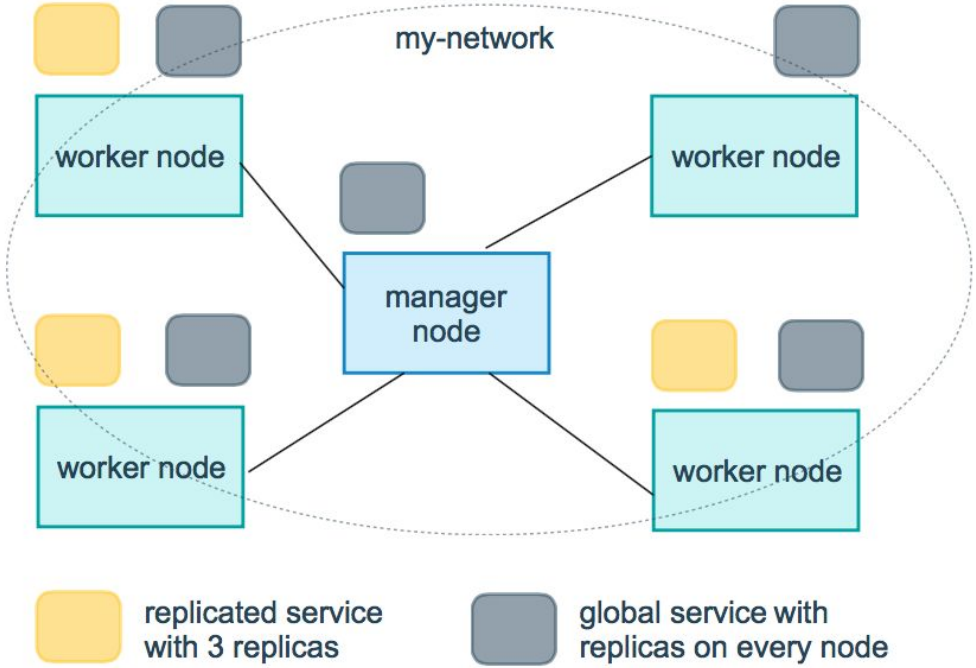


# How services work (cont'd)




Tasks and scheduling

# How services work (cont'd)

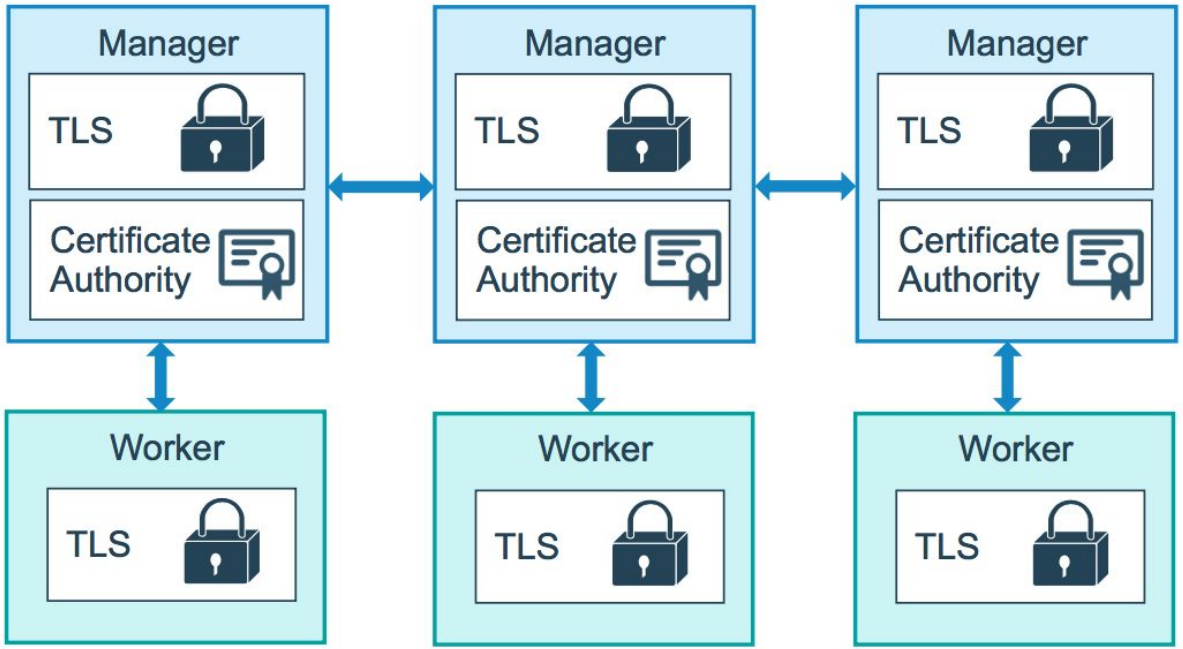



Replicated and global services

# scheduling

- Spread
- Binpack
- Random
- Customize

# security


Manager
TLS
Certificate Authority
Manager
TLS
Certificate Authority
Manager
TLS
Certificate Authority
Worker
TLS
Worker
TLS
Worker
TLS

# pros & cons

- pros
  - Specializes
  - Lightweight & Flexible
  - Scalable
  - Learning curve
- cons
  - Failures of nodes
  - Specializes
  - Third-party tool

# comparison

System classification for the application model

| System | Workload | Job Composition |
|---|---|---|
| Kubernetes | All | Co-located tasks |
| Swarm | Long-running jobs | Co-located tasks |
| Mesos | All | Single task |
| Marathon | Long-running jobs | Co-located tasks |

# comparison (cont'd)

System classification for job scheduling

| System | Architecture | Node Selection | Preemption | Rescheduling | Placement Constraints |
|---|---|---|---|---|---|
| Kubernetes | Decentralized monolithic | All nodes | - | - | Label and affinity-based |
| Swarm | Decentralized monolithic | All nodes | - | + | Label and affinity-based |
| Mesos | Two-level offerbased | N/A | - | - | N/A |
| Marathon | Two-level offerbased | All nodes | - | + | Value and querybased |

# comparison (cont’d)

System classification for cluster infrastructure

| System | Cluster Elasticity | Cluster Infrastructure |
|---|---|---|
| Kubernetes | Elastic, manual and autoscaling | Virtualized, non-virtualized |
| Swarm | Elastic, manual scaling | Virtualized, non-virtualized |
| Mesos | Elastic, manual scaling | Virtualized, non-virtualized |
| Marathon | Elastic, manual scaling | Virtualized, non-virtualized |

# comparison (cont'd)

System classification for resource management

| System | Quota Management | Resource Reclamation | Resource Granularity | Oversubscription | Resource Estimation |
|---|---|---|---|---|---|
| Kubernetes | Limits, requests | Eviction, throttling | Fine-grained | + | - |
| Swarm | Requests | Eviction | Fine-grained | - | - |
| Mesos | Requests | Eviction, throttling | Fine-grained | + | - |
| Marathon | Requests | Eviction, throttling | Fine-grained | - | - |

# comparison (cont'd)

System classification for system objectives

| System | Scalability | High Availability | High Utilization | High Throughput | Application QoS |
|---|---|---|---|---|---|
| Kubernetes | + | - | - | - | - |
| Swarm | + | + | + | + | - |
| Mesos | + | + | - | - | - |
| Marathon | + | + | - | - | - |

# So how do I choose?

- Architecture
- Scale
- Learning curve
- Third-party tool

# references

Rodriguez, Maria A., and Rajkumar Buyya. "Container‐based cluster orchestration systems: A taxonomy and future directions." *Software: Practice and Experience* 49.5 (2019): 698-719.

"Container Orchestration Definition". *Avinetworks*, https://avinetworks.com/glossary/container-orchestration/.

"What Is Container Orchestration?". *Newrelic*, 2018, https://blog.newrelic.com/engineering/container-orchestration-explained/.

"Swarm Mode Overview". *Docker*, https://docs.docker.com/engine/swarm/.

# Thanks