# Deployment and Operations for Software Engineers 2$^{nd}$ Ed

## Chapter 5—Container Orchestration

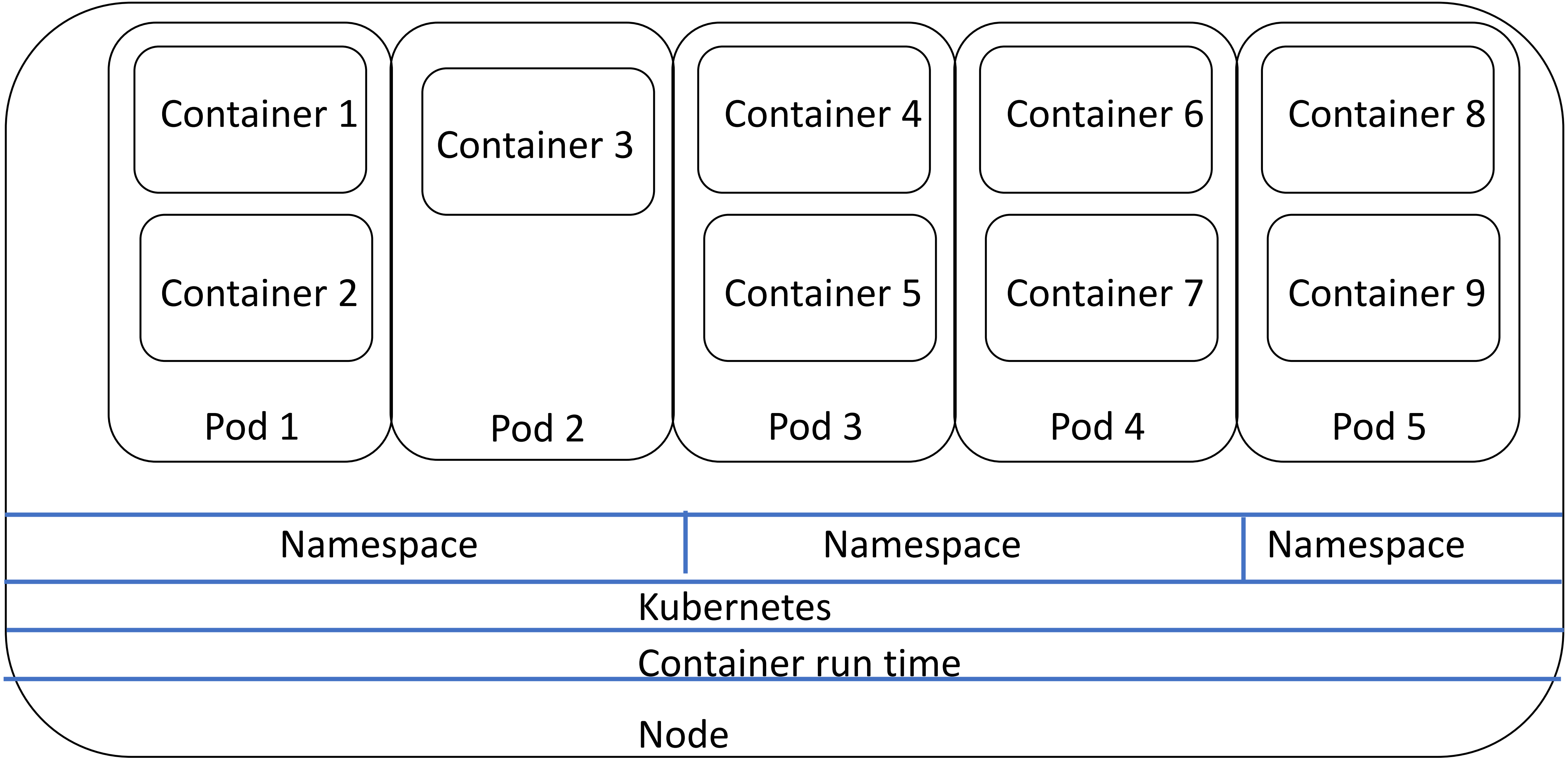
# Outline

- **Pods**
- Orchestration and service meshes

# Reducing communication times

- Sending a message from one container to another can be time consuming since it involves placing the message on the network.
- If the two containers are on the same host, the message can be sent without using the network.
- A pod is a construct from Kubernetes that ensures that two containers are deployed onto the same host.
- Pods are divided into namespaces for isolation purposes.

# Kubernetes architecture



# Single container in a pod

- The pod has an IP addresses and a collection of ports.
- A message is sent to the IP and a port.
- The message is passed to the container in the pod.

# Multiple containers in a pod

- The pod has an IP address and a collection of ports.
- Each container listens on one port—distinct from the port of the other containers in the pod.
- A message from outside the pod is delivered in the normal fashion to a container in the pod.
- A message from one container inside the pod to another container inside the pod can be sent to localhost to avoid overhead. Then it is delivered to the container listening on the destination port.

# Container lifetime

- Containers in the same pod are allocated and deallocated together. That is, they have the same lifetime.
- Two containers should be placed in the same pod if they are tightly coupled—their actions are interdependent.

# Discussion questions

1. Give examples of two containers that should be in the same pod.
2. Suppose one container in a pod fails. What happens to the other containers in that pod?

# Outline

- Pods
- **Orchestration and service meshes**

# Orchestration system

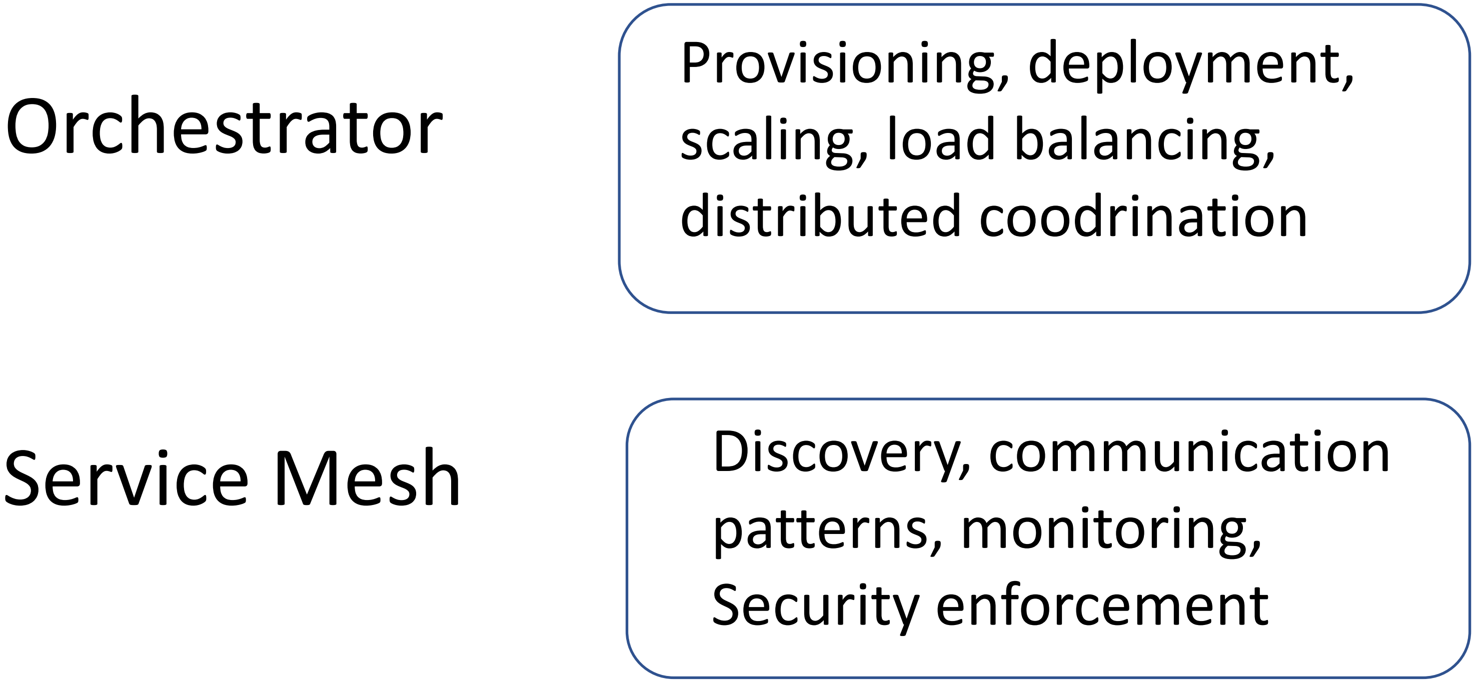
- An orchestration system manages a container's lifecycle, including provisioning, deployment, scaling, and load balancing.
- Common orchestration systems are Kubernetes, variants of Kubernetes by Google, Amazon, Microsoft, and Red Hat, and Docker Swarm.
- Istio is a common service mesh

# Kubernetes specification

- In YAML
- Pod name
- Containers included in the pod
- Number of instances of the pod to create
- Autoscaling rules
- …

# Orchestrators and Service Meshes

- Orchestrators frequently work in conjunction with a service mesh.
- Division of responsibilities shown is notional. Some functions may exist in either the orchestrator or the service mesh.



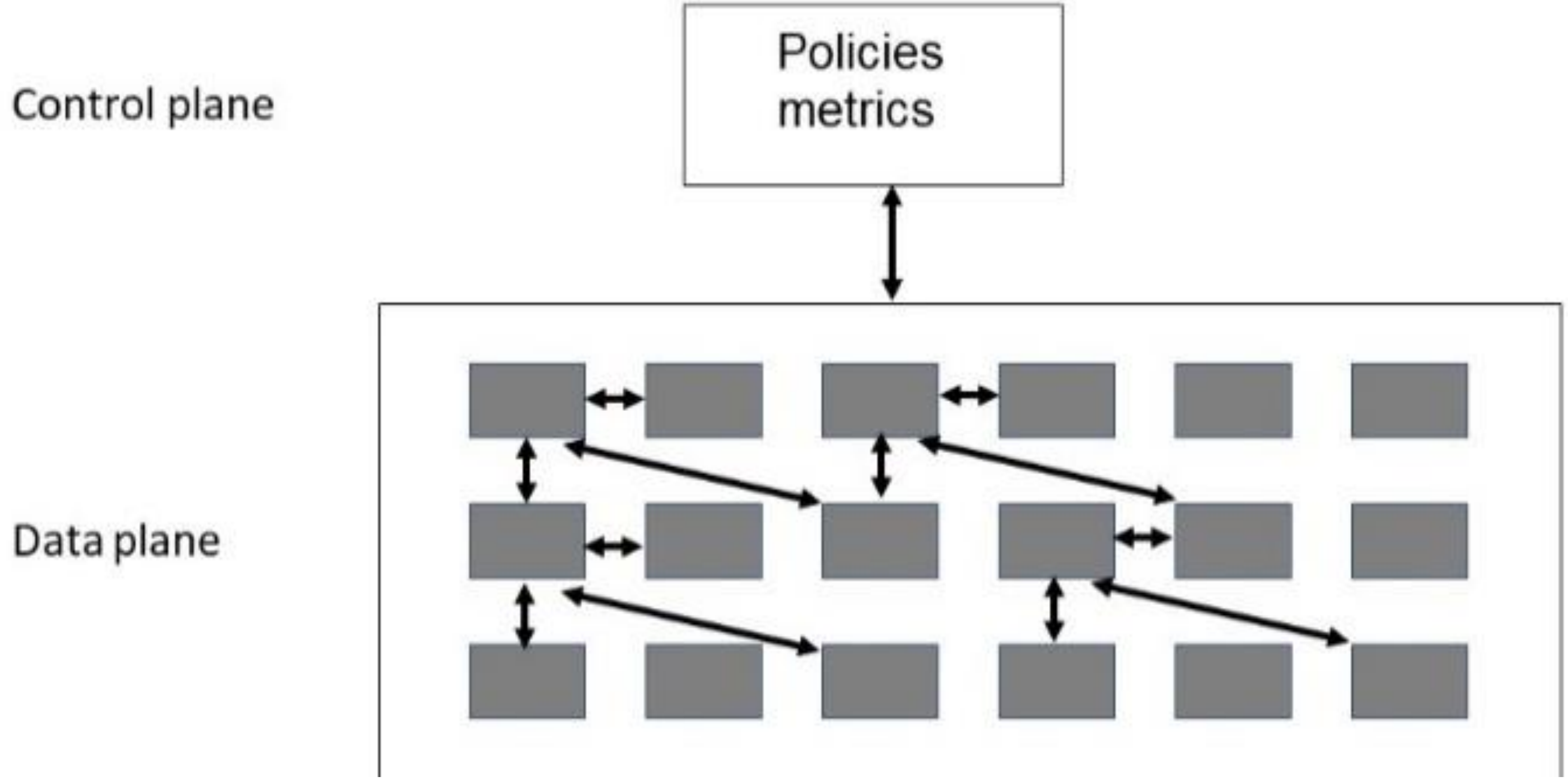
Orchestrator

Provisioning, deployment, scaling, load balancing, distributed coodrination

Service Mesh

Discovery, communication patterns, monitoring, Security enforcement

# Basic architecture of a service mesh



# Control plane and data plane

- The control plane is where management functions are performed
- The data plane is where communication is performed.
- Discovery service is packaged in pod with service mesh.
- Multiple instances of discovery service- one for each pod
- The discovery service will include IP address of containers in a particular context.

# Context

- Context rules are specified in the policy section of the control plane
- Context includes
  - Instances of containers within same pod or same host.
  - Containers participating in canary or A/B testing
  - Shadowing for availability
  - Access control

# Discussion questions

1. How are orchestration rules specified in Kubernetes?
2. How is a discovery container populated?