Designing Multi-container Pods

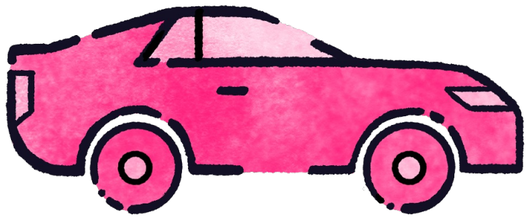
Patrick Rusch
Senior Consultant / IT Instructor

Why Use Multi-container Pods?

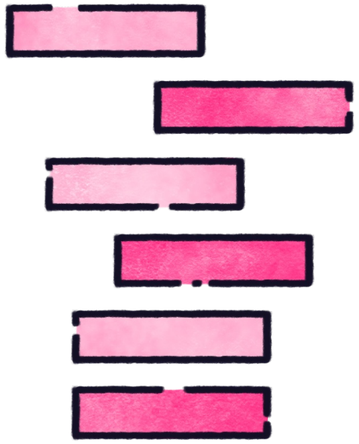
Shared Resources in a Pod

- Shared network (IP, ports)
- Shared volumes
- Co-scheduled on the same node

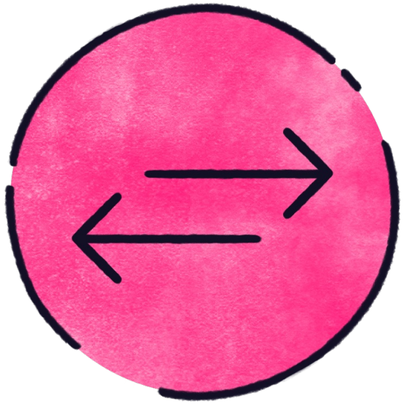
When to Use Multi-container Pods



Sidecars



Adapters



Ambassadors

Init containers

When Not to Use Them

If containers are independent

If they don’t need to share lifecycle

If they scale differently

Sidecar Pattern: Common Use Cases

What Is a Sidecar?

Helper container inside the same Pod

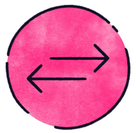
Enhances or supports the main app

Shares storage and network

Sidecar Use Cases

Logs

Metrics



Proxy

Config sync

Caching / preprocessing

Benefits of the Pattern

- Clean separation of concerns
- Reusable infrastructure components
- App remains unchanged

Init Containers: Setup before Main Container Runs

What Are Init Containers?

Run once, before main container

Sequential execution

Do setup, then exit

Common Use Cases

Service check

Config fetch

Certificate validation

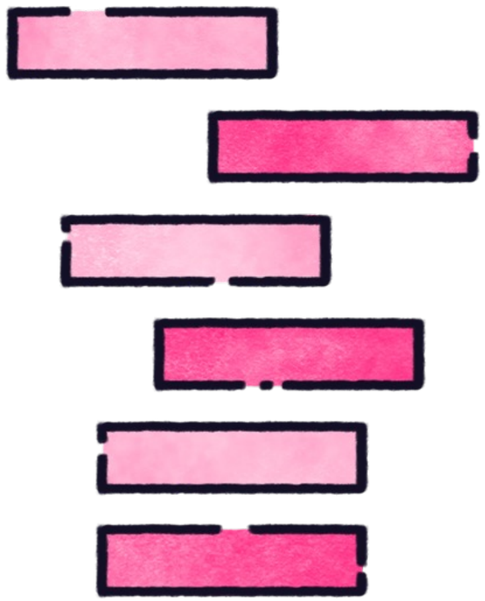
Resource wait

Benefits

- Clean separation of setup logic
- Increased reliability
- Dev and ops collaboration

Other Patterns: Adapter and Ambassador

Adapter Pattern

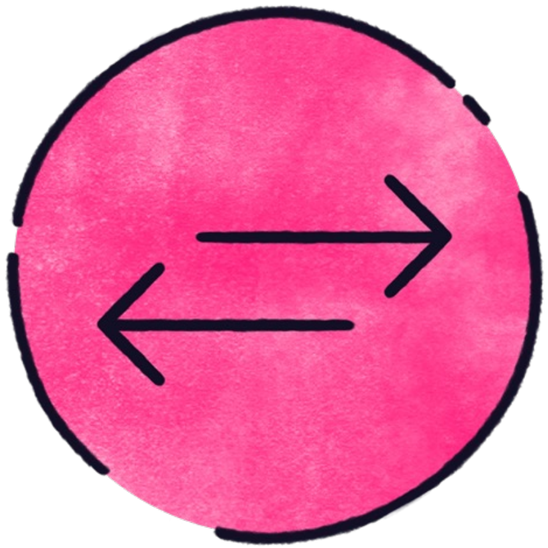


Transforms app output

Useful for log or data format conversion

Shares volume with main container

Ambassador Pattern



Proxies traffic to external services

Adds behavior like retries or TLS

App talks to localhost

When to Use

- Can’t modify app
- Need to translate formats
- Want to proxy or wrap behavior

Demo: Multi-container Pod with Init and Sidecar