119:- Diff b/w side car container & co-located container in K8s

pod:-

It is the smallest deployable unit, often comprising multiple containers that share the same n/w namespace and storage volumes.

With this context, 2 common patterns emerge.

1. SideCar Container 2. Co-located Containers

while both reside within the same pod, they serve distant roles & have different lifecycle behaviors.

SideCar Container

It is a Secondary Container that runs alongside the main application Container within the same pod

purpose:-

To enhance or extend the functionality of the main application without modifying its code

Usecase:-

* logging * monitoring * proxying & * data synchronization

starting with K8s v1.29, sidecar containers are implemented as restartable init containers by setting restartPolicy: Always and are placed under the init Containers field.

⇒ This approach ensures that sidecar containers start before and are terminated after the main application containers & continue running throughout the pod's lifecycle

Eg:- Application with a logging sidecar

```
apiVersion : v1
Kind: Pod
metadata:
  name: app-with-logging-sidecar
 Spec:
  Volumes:
    - name: shared-logs
      emptyDir: {}
   init Containers:
     - name: log-shipper
       image: alpine: latest
       command: ['sh', '-c', 'tail -F /opt/logs.txt']
       volumeMounts:
         - name: shared-logs
           mountPath: /opt
       restartPolicy: Always
     Containers:
      - name: main-app
        image: alpine: latest
        command: ['sh', '-c', 'while true; do echo "logging" >>
                          /opt/logs.txt; sleep 1; done']

        volumeMounts:
          - name: shared-logs
            mountPath : /opt
```

In this eg, log-shipper side car container tails the log file generated by the main-app container.

The shared volume shared-logs facilitates communication b/w the two containers.

It helps the main container by doing extra work

share resources like logs

Eg:- chef + dishwasher

Co-located Containers

⇒It refers to multiple containers running within the same pod that collaborate to achieve a common goal.

⇒That is the new term now used to distinguish what was previously referred to as a sidecar container before the K8s V.129 updates

⇒Unlike sidecar containers, co-located containers often share equal responsibility in the app's functionality.

⇒They can start, & stop independently & may not have a defined, startup order

Eg:- webServer with content Generator

```
apiVersion: v1
kind: Pod
metadata:
  name: web-server-with-helper
spec:
  volumes:
    -name: shared-content
     emptyDir: {}
  containers:
    - name: content-generator
      image: busy-box
      command: ['sh', '-c', 'echo "Hello from Helper!" >
                /output/index.html && sleep 3600']

      volumeMounts:
        -name: shared-content
         mountPath: /output

  -name: webserver
   image: nginx
   volumeMounts:
     - name : shared-content
       mountPath: /usr/share/nginx/html
```

Here, the content-generator container creates an HTML file that the web server container serves using NGINX.

Both containers are integral to app's functionality & operate collaboratively.

Key Difference

Feature Side Car Container Co-located Container Primary
Role Enhances or extend main application functionality
Collaborate equally in application logic Startup Order Starts
before main container (if using init container)
No Defined startup order Lifecycle Runs alongside main container;
Can be restarted independently Independent lifecycle Typical
Use Cases Logging, monitoring, proxying, data synchronization
Multi-process applications, helpers Implementation (post v1.29)
defined as init container with restartPolicy: Always Defined in
containers field.

Co-located container

⇒ Just lives together (share same pod)

⇒ May not share anything

Side car = Best helper friend

Co-located = Just room mates