
KubeCon
CloudNativeCon
North America 2023


KubeCon | CloudNativeCon

North America 2023

Sidecar Containers are Built-in to Kubernetes: What, How, and Why Now?

Sergey Kanzhelev (Google)
Todd Neal (AWS)

Sidecars in Kubernetes

Sergey Kanzhelev (Google)

Todd Neal (AWS)

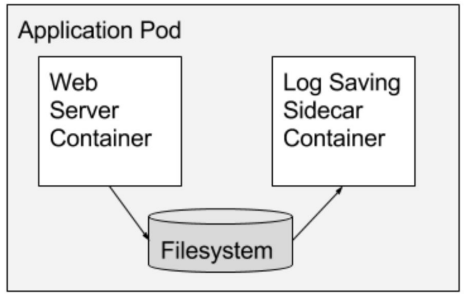
What are sidecars?

Sidecar containers extend and enhance the "main" container, they take existing containers and make them better.

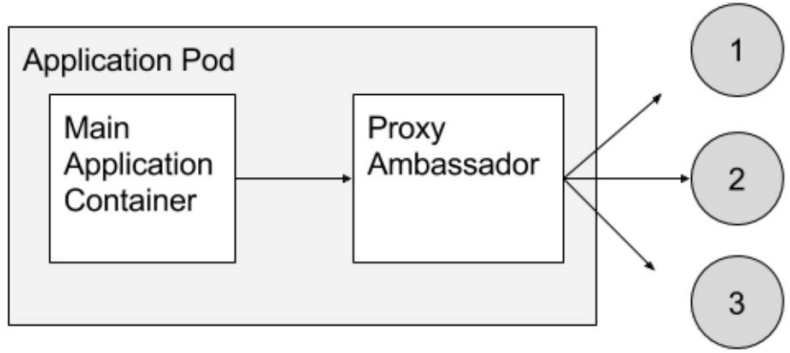
- *kubernetes.io Blog Post (June 2015)*

Examples:

- Telemetry
- Networking
- Security (certificate refresh)
- Data access



Telemetry

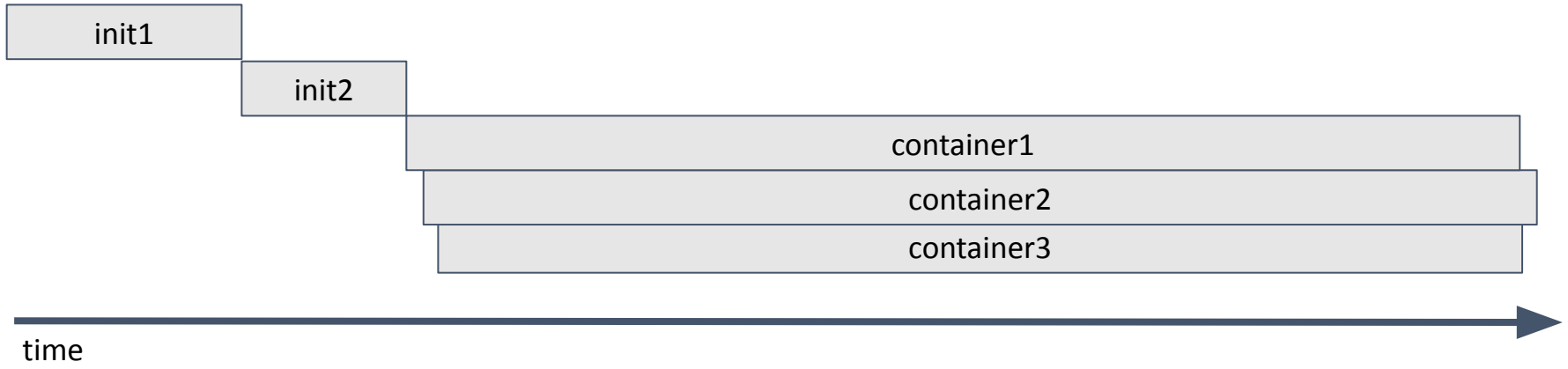


Networking

What we have today?

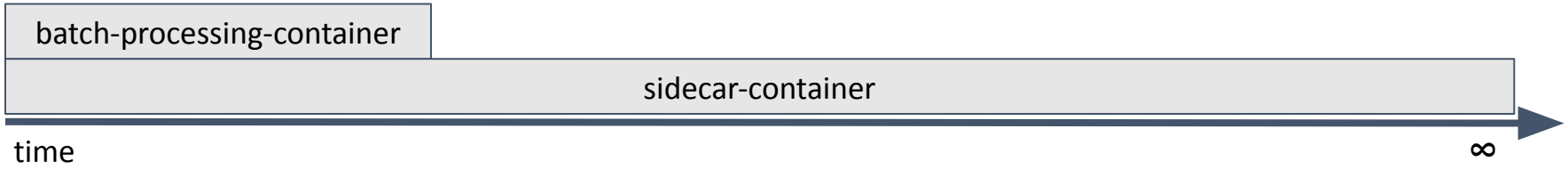
- Init Containers
  - Ordered Startup & Termination
  - Run to completion prior to subsequent init containers and any main container

- Main Containers
  - Unordered Startup & Termination
  - Share the lifetime of the pod



What about Jobs?

- Job pods typically have a finite lifetime and finish when the main containers successfully terminate
- If your “sidecar” is a main container, your job pod doesn’t terminate
- If the “sidecar” crashed, it is not being restarted again



Why now?

- Kubernetes is used widely
- SIG Node state:
  - Graduated a lot of beta features, went thru major dockershim deprecation
  - A lot of experimentation!
    - 40 feature gates
      - 5 GA
      - 16 beta (3 in supported versions)
      - 19 alpha, many draft keps

SIG Node directions:

- Support new workloads
- Better understand the hardware

Why now?

Sidecar feature is improving the support of new workloads

- Batch/jobs support
- AI/ML is in high demand
- Telemetry and mesh sidecars need more guarantees

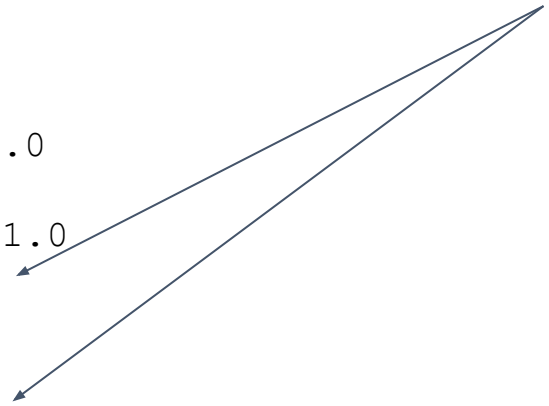
What does the sidecar feature bring?

- Ordered startup & termination
- Run for the lifetime of the pod, including restarting if they crash
- Won’t block pod completion

How are the implemented?

How?

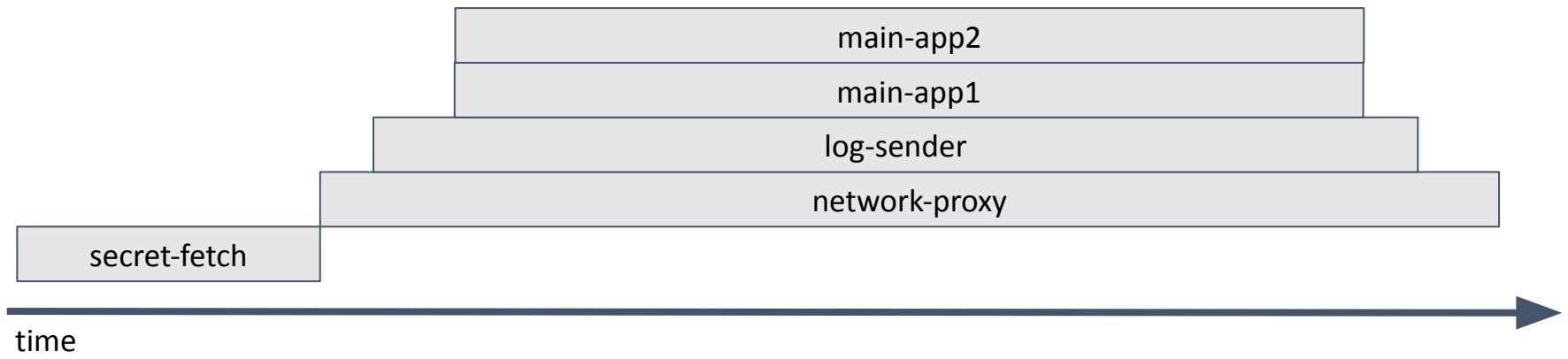

```yaml
apiVersion: v1
kind: Pod
spec:
 initContainers:
 - name: secret-fetch
   image: secret-fetch:1.0
 - name: network-proxy
   image: network-proxy:1.0
   restartPolicy: Always
 - name: log-sender
   image: log-sender:1.0
   restartPolicy: Always
 containers:
 - name: main-app1
   image: main-app1:1.0
 - name: main-app2
   image: main-app2:1.0
 ...
```



"Sidecars" are an init container with a `restartPolicy` of `Always`.

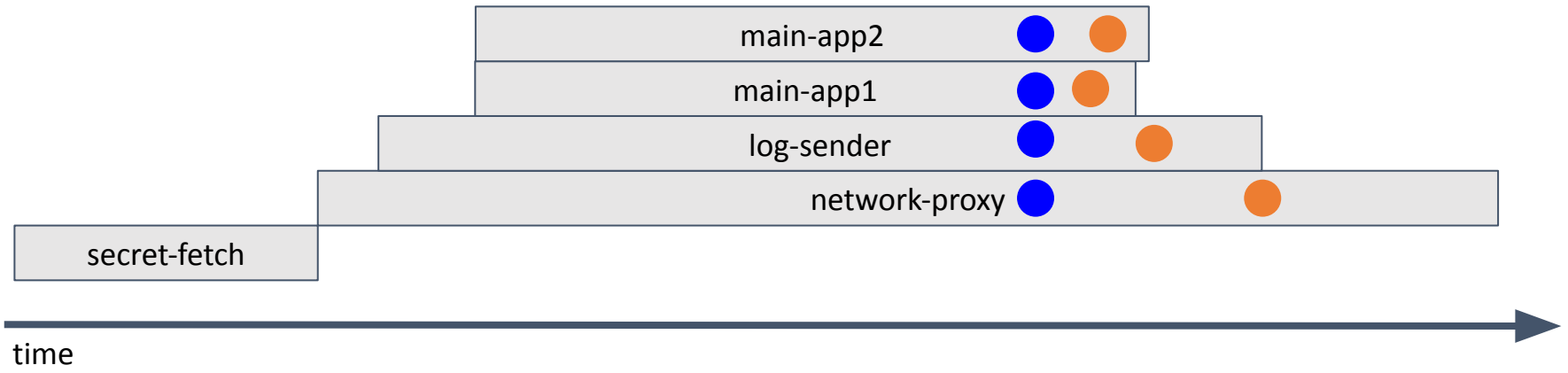
How?


```yaml
apiVersion: v1
kind: Pod
spec:
 initContainers:
 - name: secret-fetch
   image: secret-fetch:1.0
 - name: network-proxy
   image: network-proxy:1.0
   restartPolicy: Always
 - name: log-sender
   image: log-sender:1.0
   restartPolicy: Always
 containers:
 - name: main-app1
   image: main-app1:1.0
 - name: main-app2
   image: main-app2:1.0
 ...
```



Termination Ordering for Sidecars

- PreStop -> Signals pod termination has started
- SIGTERM -> Signals that preceding sidecars/main containers have exited

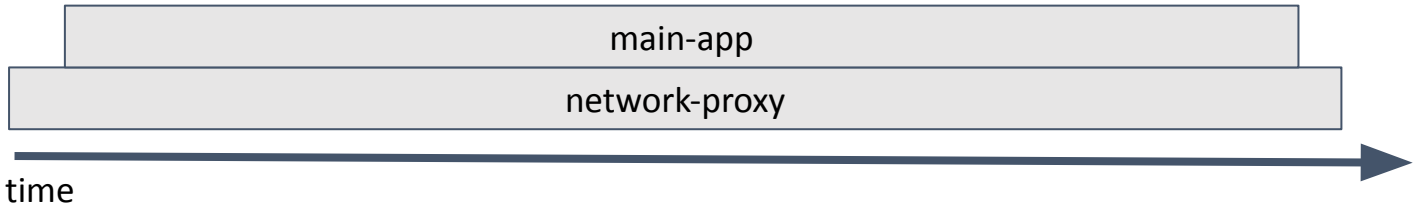


Sidecar Best/Worst Practices

Best: Define or recommend requests and limits for the sidecar & account for the increase in your planning.

Worst: Who needs requests/limits?

```yaml
apiVersion: v1
kind: Pod
spec:
 initContainers:
 - name: network-proxy
   image: network-proxy:1.0
   restartPolicy: Always
   resources:
     requests:
       cpu: 1
 containers:
 - name: main-app1
   image: main-app1:1.0
   resources:
     requests:
       cpu: 1
```

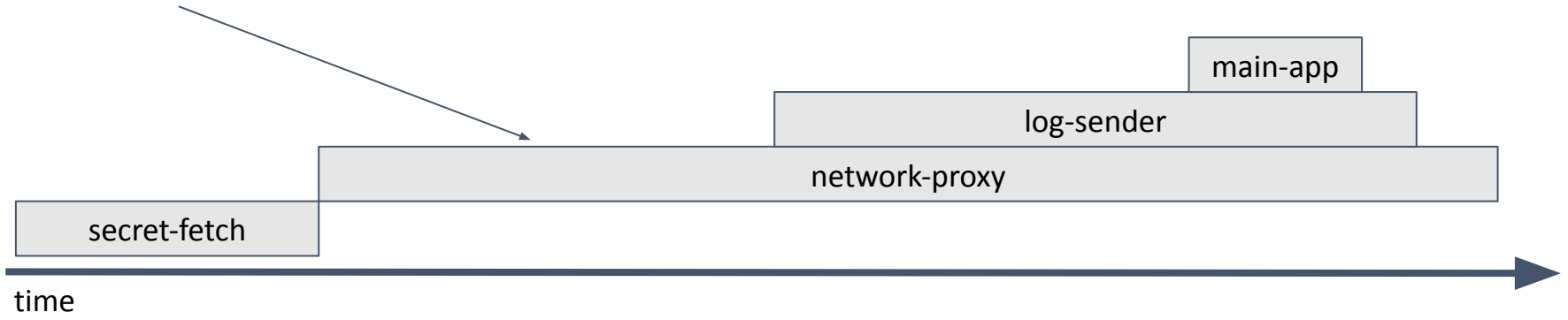


Sidecar Best/Worst Practices

Best: Minimize the time spent in startupProbe or postStart hook

Worst: Take as long as you want, the main containers will start eventually….

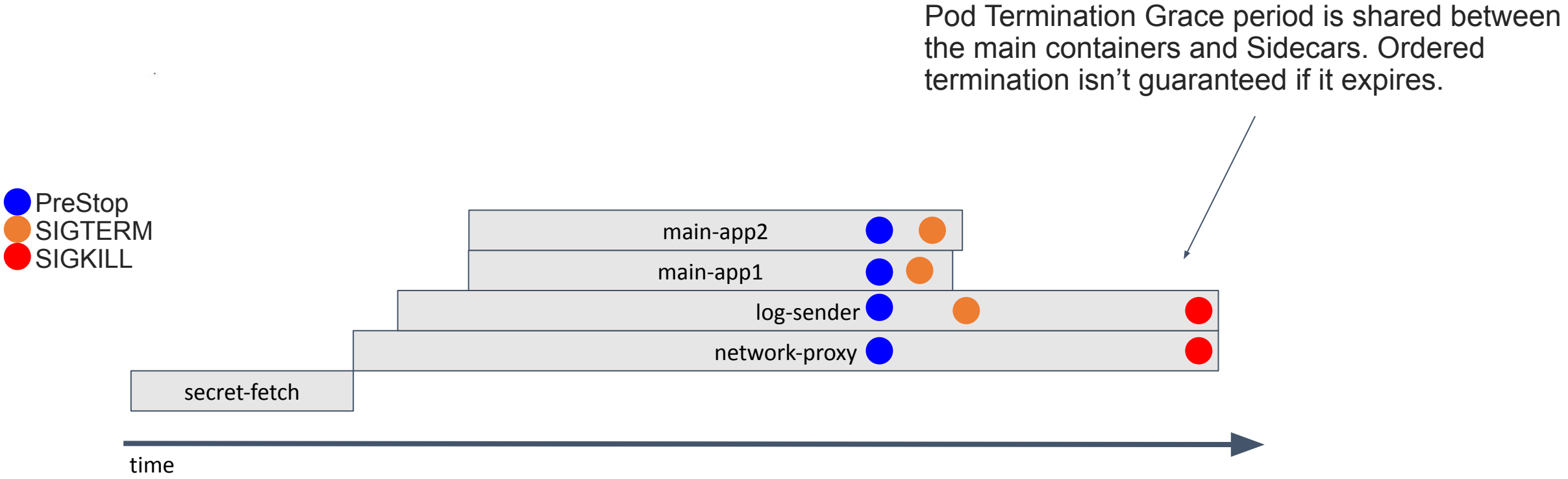
Sidecars start serially, so minimize this time for faster overall Pod startup.



Sidecar Best/Worst Practices

Best: Exit preStop hook ASAP, and shutdown fully on SIGTERM.

Worst: Take as long as you want, who needs termination ordering?



Don’t Forget!

- Any code that sums Pod resource requests is wrong if it doesn’t consider Sidecars
  - e.g. That reporting tool you wrote three years ago…

- You may need to recompile mutating admission webhooks that will silently drop the new restartPolicy field


KubeCon | CloudNativeCon

North America 2023

What’s next?

State of Sidecars

- 1.28
  - Alpha Status
  - Ordered sidecar startup
  - Sidecars restart if they crash
  - No ordered sidecar termination
- 1.29
  - Beta Status
  - Adds ordered sidecar termination
  - Better integration with resource managers
- Adoption:
  - Istio: https://istio.io/latest/blog/2023/native-sidecars/
  - GCS fuse (talk: https://sched.co/1Rj4I):
- Upcoming:
  - Restart sidecars during pod termination if they crash

More goodness

- Security boundaries between the sidecar and main containers
- Different resource usage patterns - e.g. dedicated CPUs for the main containers and shared pool for the sidecars
- Crashloop backoff configuration for sidecar containers
- OOMkill and other liveness cross-dependencies between containers and sidecars
- More lifecycle control patterns

Where we will NOT go…

… anytime soon:

- Pod is a single scheduleable unit. We do not plan to make Containers even more flexible - i.e. conditional enablement of a container inside the Pod
- Full-blown systemd clone inside the Pod. We evaluated the need for things like PartOf or BindTo and decided it is too much
- Split Pod into multiple units and/or provide “Sidecar Pods”
- Define resource usage as a percentage of other containers resource usage

Please scan the QR Code above
to leave feedback on this session