KubeCon | CloudNativeCon

Europe 2022

WELCOME TO VALENCIA

Debugging With Ephemeral Containers

Aaron Alpar, Kasten by Veeam

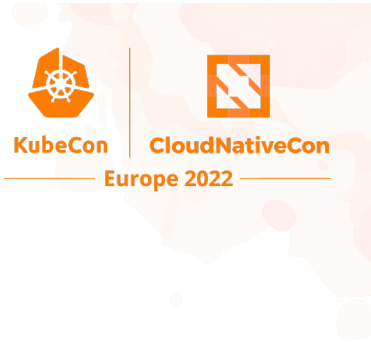
Ephemeral Containers

Debugging with Ephemeral Containers

KASTEN by Veeam

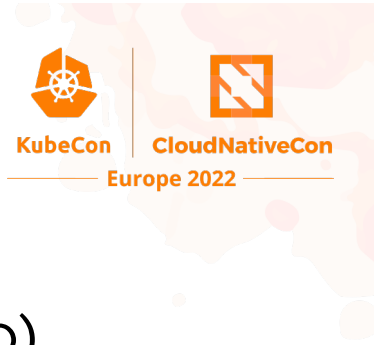
Aaron Alpar
MTS, *Kasten by Veeam*

About this Presentation



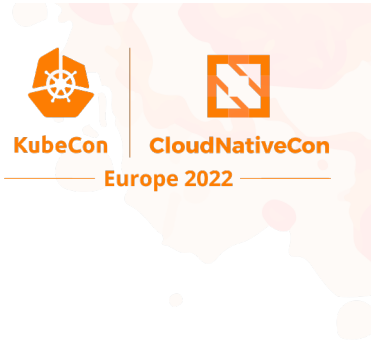
- Introduction
- Ephemeral Containers
- Setup and Tools
- Debugging containers in K8s
- Linux Namespaces
- New Pod properties
- Creating Ephemeral Containers with custom `securityContext`
- Distroless

Kind



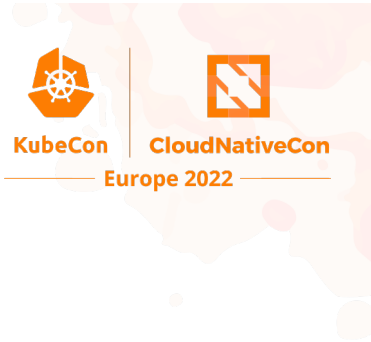
- Use Kind in this presentation (https://kind.sigs.k8s.io)
- Local Kubernetes environment
- Used for Kubernetes automated testing
- Uses Docker

Postgres



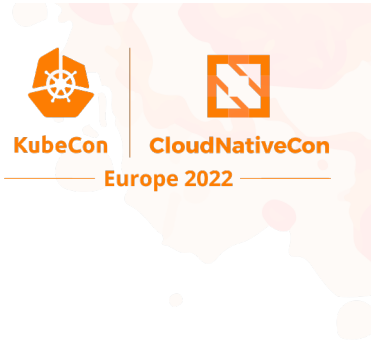
- Open source database
- Good target for examples

Setup: Kind



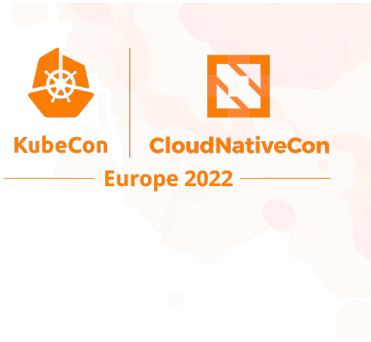

```
$ kind create cluster
Creating cluster "kind" ...
 ✓ Ensuring node image (kindest/node:v1.23.4)
 ✓ Preparing nodes
...

Not sure what to do next? 😅  Check out
https://kind.sigs.k8s.io/docs/user/quick-start/

$
```

Setup: Postgres



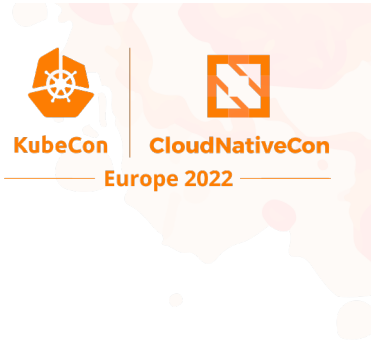

```
$ helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories

$ kubectl create namespace pg
namespace/pg created

$ helm install pg bitnami/postgresql --namespace="pg"
NAME: pg
LAST DEPLOYED: Tue Apr 12 09:35:44 2022
...
    PGPASSWORD="$POSTGRES_PASSWORD" psql --host 127.0.0.1 -U postgres -d
postgres -p 5432
```

Setup: Check versions



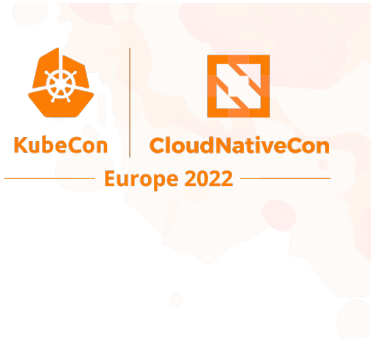

```
$ kubectl version --short
...
Client Version: v1.24.0
Kustomize Version: v4.5.4
Server Version: v1.23.4
```

Setup



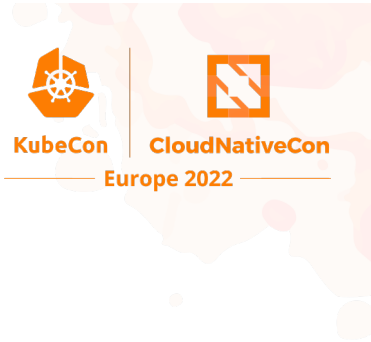

```
$ kubectl get pods -n pg
NAME                READY   STATUS    RESTARTS   AGE
pg-postgresql-0     1/1     Running   0          30s
```

Ephemeral Containers



- Dynamically allocated containers
- Useful for debugging
- Beta in 1.23

Debug without Ephemeral Containers



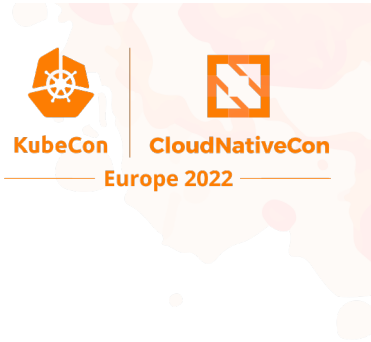
- Ephemeral containers start with debugging

Debug without Ephemeral Containers

- Ephemeral containers start with debugging
- Debug by attaching (exec)

```
kubectl exec -it -n pg pg-postgresql-0 \
  -c postgresql -- /bin/sh
```

Debug without Ephemeral Containers



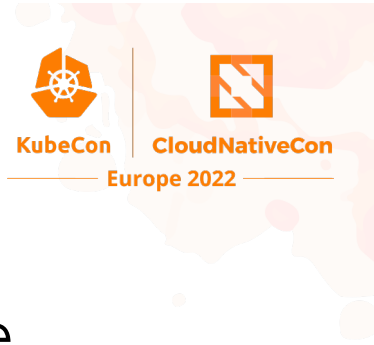
- Ephemeral containers start with debugging
- Debug by attaching (exec)

```
kubectl exec -it -n pg pg-postgresql-0 \
  -c postgresql -- /bin/sh
```

- Debug by copy

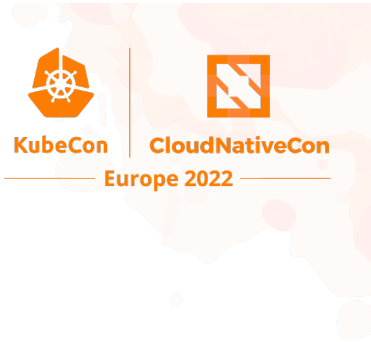
```
kubectl debug -it -n pg pg-postgresql-0 \
  --copy-to pg-postgresql-debug \
  --container debug-postgresql \
  --image busybox -- /bin/sh
```

Debug without Ephemeral Containers



- Debug by **attaching** limited by the tools in the image
- Debug by **copy** requires a pod restart
- Debugging workloads requires modifying number of replicas

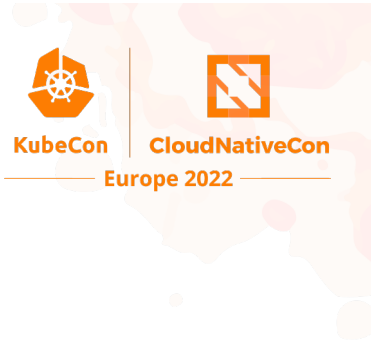
Debug with Ephemeral Containers



- Debug by ephemeral container

```
kubectl debug -it -n pg pg-postgresql-0 \
  --image busybox -- /bin/sh
```

Debug with Ephemeral Containers

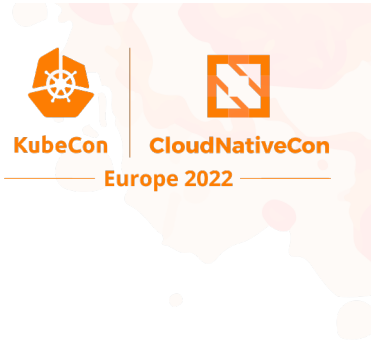


- Debug by ephemeral container

```
kubectl debug -it -n pg pg-postgresql-0 \
  --image busybox -- /bin/sh
```

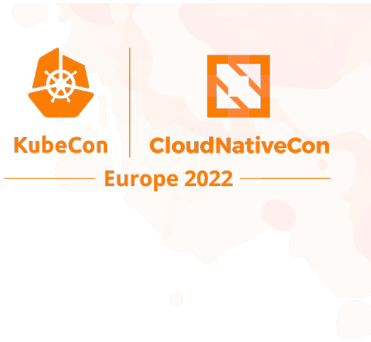
- Allow sharing of container resources
- Ideal for complicated debugging of failing containers

Debug with Ephemeral Containers



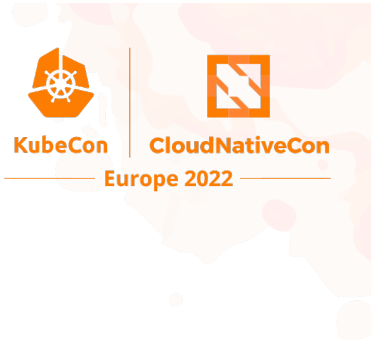

```
$ kubectl debug \
    -n pg \
    -c debug-postgresql \
    -it \
    --image=busybox \
    pod/pg-postgresql-0 \
    -- /bin/sh
If you don't see a command prompt, try pressing
enter.

# exit
```

Ephemeral Containers spec:



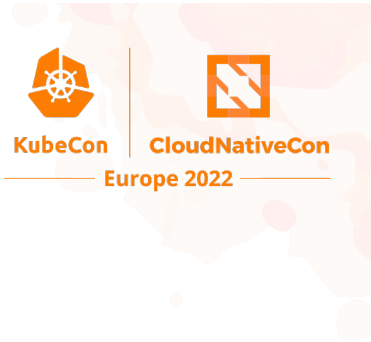

```
$ kubectl get pod -n pg pg-postgresql-0 -o yaml

apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2021-10-17T19:26:40Z"
...
spec:
  containers:
     ...
  ephemeralContainers:
  - image: busybox
    imagePullPolicy: Always
    name: debug-postgresql
...
status:
...
```

Ephemeral Containers spec:



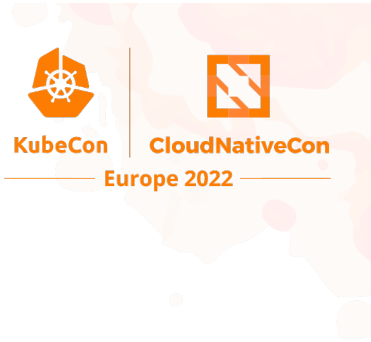

```
...
spec:
  containers:
      ...
  ephemeralContainers:
  - image: busybox
    imagePullPolicy: Always
    name: debug-postgresql
...
status:
...
```

Ephemeral Containers `status`:



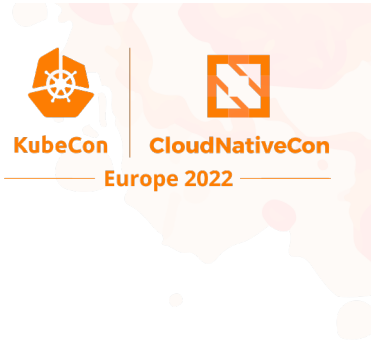

```
$ kubectl get pod -n pg pg-postgresql-0 -o yaml

apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2021-10-17T19:26:40Z"
...
spec:
  ...
status:
  containerStatuses:
  ...
  ephemeralContainerStatuses:
  - containerID: docker://85a4eb6742bf ...
    image: busybox:latest
    imageID: docker-pullable://busybox@sha25 ...
...
```

Ephemeral Containers `status:`




```
...
spec:
  ...
status:
  containerStatuses:
    ...
  ephemeralContainerStatuses:
  - containerID: docker://85a4eb6742bf ...
    image: busybox:latest
    imageID: docker-pullable://busybox@sha25 ...
...
```

What is a Container?



- What is a Container?

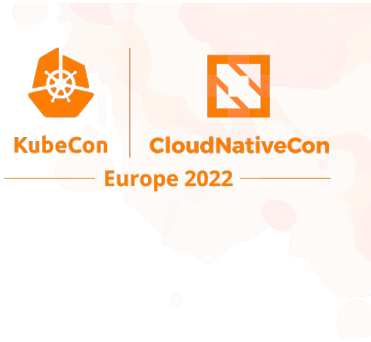
What is a Container?

- What is a Container?
- Containers can be thought of as Sandboxes
- Once in a Sandbox, cannot see what's outside of the Sandbox
- Controlling what a container can see is what isolates one Container from another

What is a Container?

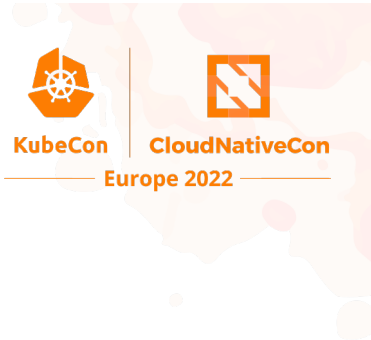
- What is a Container?
- Containers can be thought of as Sandboxes
- Once in a Sandbox, cannot see what's outside of the Sandbox
- Controlling what a container can see is what isolates one Container from another
- **Linux Namespaces in the Node make this possible**

Linux Namespaces



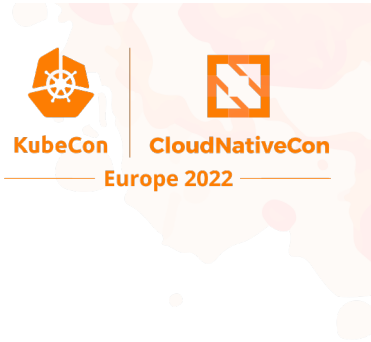
- Linux Namespaces control Container isolation
- Linux Namespaces contain Container resources
  - processes, mounts, …
- Linux Namespaces can be copied from Container to Container

Linux Namespaces



- There are many
  - `mnt`, `pid`, `net`, `ipc`, `uts`, `uid`, `cgroup`, `time`, ...
- Each namespace is identified by an `inode`

Linux Namespaces

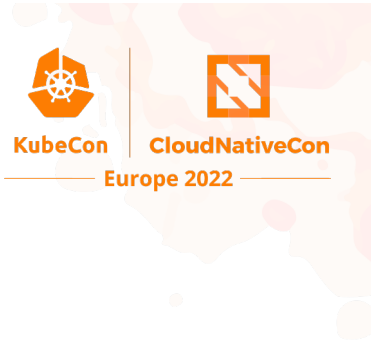


- There are many
  - `mnt`, `pid`, `net`, `ipc`, `uts`, `uid`, `cgroup`, `time`, ...
- Each namespace is identified by an `inode`
- `pid`
  - Processes
- `mnt`
  - File system mounts
- `net`
  - Network interfaces

Linux Namespaces on node (docker)

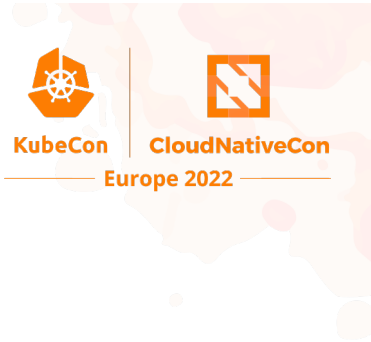

```
$ docker ps # get docker container for node
CONTAINER ID  IMAGE                  COMMAND ...                  NAMES
abf011764870  kindest/node:v1.23.4  "/usr/local/bin/entr…" ...  kind-control-
plane

$ docker exec -it kind-control-plane /bin/bash # exec into node

# ls -l /proc/1/ns/* # get namespaces for PID 1
lrwxrwxrwx 1 root root 0 May  7 20:12 /proc/1/ns/cgroup -> cgroup:[4026532459]
lrwxrwxrwx 1 root root 0 May  7 20:12 /proc/1/ns/ipc -> ipc:[4026532372]
lrwxrwxrwx 1 root root 0 May  7 20:12 /proc/1/ns/mnt -> mnt:[4026532370]
lrwxrwxrwx 1 root root 0 May  7 20:12 /proc/1/ns/net -> net:[4026532376]
lrwxrwxrwx 1 root root 0 May  7 20:00 /proc/1/ns/pid -> pid:[4026532374]
...
```

Linux Namespaces on node (docker)



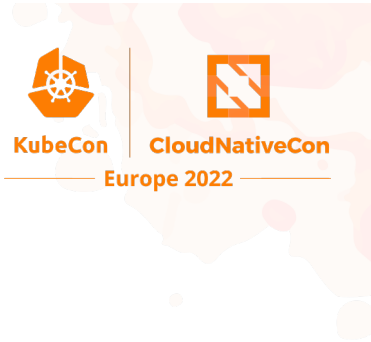

```
# # get namespaces for PID 1
# ls -l /proc/1/ns/*
lrwx ... /proc/1/ns/cgroup -> cgroup:[4026532459]
lrwx ... /proc/1/ns/ipc -> ipc:[4026532372]
lrwx ... /proc/1/ns/mnt -> mnt:[4026532370]
lrwx ... /proc/1/ns/net -> net:[4026532376]
lrwx ... /proc/1/ns/pid -> pid:[4026532374]
```

Linux Namespaces on node (docker)



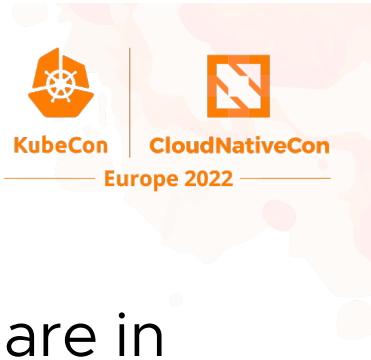

```
# pgrep -o postgres # get postgres PID
1935

# ps -f -p $(pgrep -o postgres) # get postgres process info
UID          PID    PPID  C STIME          TIME CMD
1001        1935    1805  0 00:54      00:00:00 /opt/bitnami/postgresql ...

# ls -l /proc/$(pgrep -o postgres)/ns/* # list postgres namespaces
lrwxrwxrwx 1 1001 root 0 May  7 20:03 /proc/1946/ns/cgroup -> cgroup:[4026532921]
lrwxrwxrwx 1 1001 root 0 May  7 20:03 /proc/1946/ns/ipc -> ipc:[4026532917]
lrwxrwxrwx 1 1001 root 0 May  7 20:03 /proc/1946/ns/mnt -> mnt:[4026532919]
lrwxrwxrwx 1 1001 root 0 May  7 20:03 /proc/1946/ns/net -> net:[4026532837]
lrwxrwxrwx 1 1001 root 0 May  7 20:03 /proc/1946/ns/pid -> pid:[4026532920]
...
```

Linux Namespaces on node (docker)




```
# # list postgres namespaces
# ls -l /proc/$(pgrep -o postgres)/ns/*
lrwx ... /proc/1946/ns/cgroup -> cgroup:[4026532921]
lrwx ... /proc/1946/ns/ipc -> ipc:[4026532917]
lrwx ... /proc/1946/ns/mnt -> mnt:[4026532919]
lrwx ... /proc/1946/ns/net -> net:[4026532837]
lrwx .../proc/1946/ns/pid -> pid:[4026532920]
```

nsenter



- What you can see depends on what namespace you are in
- nsenter allows execution of a command within a namespace
- The command will only see what the namespace allows

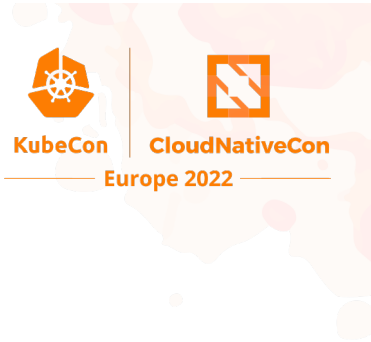
nsenter


```
# nsenter --net=/proc/$(pgrep -o postgres)/ns/net \
    ss --tcp -l
State  Recv-Q  Send-Q  Local Address:Port  Peer
Address:Port
LISTEN 0       224     0.0.0.0:postgresql  0.0.0.0:*
LISTEN 0       224     [::]:postgresql     [::]:*
```

nsenter

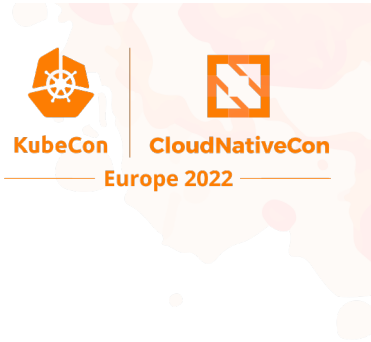

```
# nsenter --target $(pgrep -o postgres) --all ps -e
    PID TTY          TIME CMD
      1 ?        00:00:01 postgres
    137 ?        00:00:00 postgres
    138 ?        00:00:00 postgres
    139 ?        00:00:00 postgres
    140 ?        00:00:00 postgres
    141 ?        00:00:00 postgres
    142 ?        00:00:00 postgres
  22340 ?        00:00:00 ps
```

Namespaces on Pod Container (K8s)



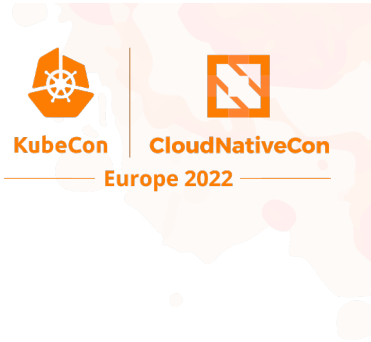

```
$ kubectl exec -it -n pg pg-postgresql-0 -- ps -ef

UID   PID  PPID  C STIME TTY    TIME CMD
1001    1     0  0 17:41 ?      00:00:00 /opt/bitnami/postgresql ...
1001  137     1  0 17:41 ?      00:00:00 postgres: checkpointer
1001  138     1  0 17:41 ?      00:00:00 postgres: background writer
1001  139     1  0 17:41 ?      00:00:00 postgres: walwriter
1001  140     1  0 17:41 ?      00:00:00 postgres: autovacuum la ...
1001  141     1  0 17:41 ?      00:00:00 postgres: stats collector
1001  142     1  0 17:41 ?      00:00:00 postgres: logical repli ...
1001 1180     0  0 17:52 pts/0  00:00:00 ps -ef
```

Namespaces using nsenter



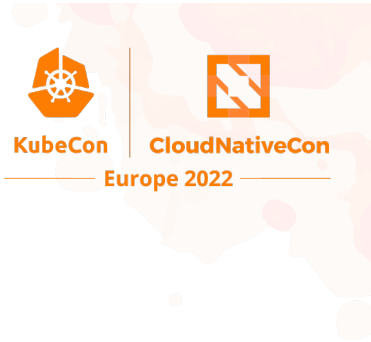

```
$ docker exec kind-control-plane /bin/bash -c -- \
  'nsenter --target $(pgrep -o postgres) --all /bin/ps -ef'
UID   PID  PPID  C STIME TTY    TIME CMD
1001    1     0  0 17:41 ?      00:00:00 /opt/bitnami/postgresql ...
1001  137     1  0 17:41 ?      00:00:00 postgres: checkpointer
1001  138     1  0 17:41 ?      00:00:00 postgres: background writer
1001  139     1  0 17:41 ?      00:00:00 postgres: walwriter
1001  140     1  0 17:41 ?      00:00:00 postgres: autovacuum la ...
1001  141     1  0 17:41 ?      00:00:00 postgres: stats collector
1001  142     1  0 17:41 ?      00:00:00 postgres: logical repli ...
1001  682     0  0 17:46 pts/1  00:00:00 /bin/sh
root 1073     0  0 17:50 pts/1  00:00:00 ps -ef
```

Ephemeral Containers: debug



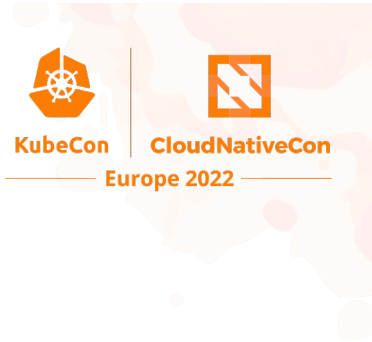
- Network namespace shared by default

Ephemeral Containers



- Network namespace shared by default
- Other namespaces can be shared

Namespaces when debug by attaching



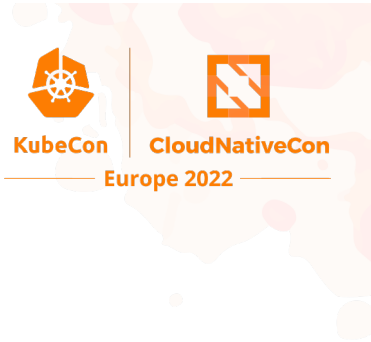

```
$ kubectl exec -it -n pg pg-postgresql-0 -c postgresql -- /bin/sh
```

```
# ls -l /proc/self/ns/*
1 1001  root  0 May  7 20:27 /proc/self/ns/cgroup -> cgroup:[4026532921]
1 1001  root  0 May  7 20:27 /proc/self/ns/ipc -> ipc:[4026532917]
1 1001  root  0 May  7 20:27 /proc/self/ns/mnt -> mnt:[4026532919]
1 1001  root  0 May  7 20:27 /proc/self/ns/net -> net:[4026532837]
1 1001  root  0 May  7 20:27 /proc/self/ns/pid -> pid:[4026532920]
1 1001  root  0 May  7 20:27 /proc/self/ns/pid_for_children -> pid:[4026532920]
1 1001  root  0 May  7 20:27 /proc/self/ns/time -> time:[4026531834]
1 1001  root  0 May  7 20:27 /proc/self/ns/time_for_children -> time:[4026531834]
1 1001  root  0 May  7 20:27 /proc/self/ns/user -> user:[4026531837]
1 1001  root  0 May  7 20:27 /proc/self/ns/uts -> uts:[4026532916]
```

Namespaces when using Ephemeral Container...

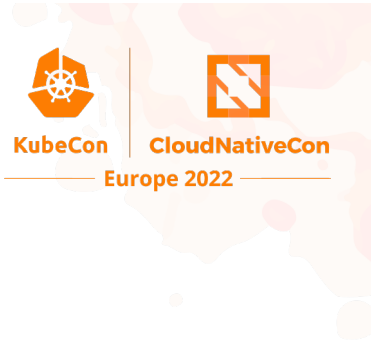

```
$ kubectl debug -it -n pg pg-postgresql-0 --image busybox -- /bin/sh

Defaulting debug container name to debugger-n8bvb
If you don't see a command prompt, try pressing enter.

# ls -l /proc/self/ns/*
1 root  root  0 May  7 20:28 /proc/self/ns/cgroup -> cgroup:[4026532924]
1 root  root  0 May  7 20:28 /proc/self/ns/ipc -> ipc:[4026532917]
1 root  root  0 May  7 20:28 /proc/self/ns/mnt -> mnt:[4026532922]
1 root  root  0 May  7 20:28 /proc/self/ns/net -> net:[4026532837]
1 root  root  0 May  7 20:28 /proc/self/ns/pid -> pid:[4026532923]
1 root  root  0 May  7 20:28 /proc/self/ns/pid_for_children -> pid:[4026532923]
1 root  root  0 May  7 20:28 /proc/self/ns/time -> time:[4026531834]
1 root  root  0 May  7 20:28 /proc/self/ns/time_for_children -> time:[4026531834]
1 root  root  0 May  7 20:28 /proc/self/ns/user -> user:[4026531837]
1 root  root  0 May  7 20:28 /proc/self/ns/uts -> uts:[4026532916]
```

Share namespaces of target container ...



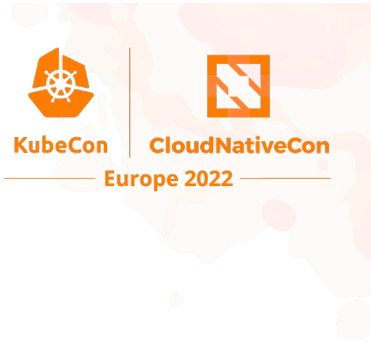

```
$ kubectl debug -it -n pg pg-postgresql-0 --image busybox \
    --target postgresql -- /bin/sh
Defaulting debug container name to debugger-vfcxs
If you don't see a command prompt, try pressing enter.

# ls -l /proc/self/ns/*
1 root  root  0 May  7 20:31 /proc/self/ns/cgroup -> cgroup:[4026532923]
1 root  root  0 May  7 20:31 /proc/self/ns/ipc -> ipc:[4026532917]
1 root  root  0 May  7 20:31 /proc/self/ns/mnt -> mnt:[4026532922]
1 root  root  0 May  7 20:31 /proc/self/ns/net -> net:[4026532837]
1 root  root  0 May  7 20:31 /proc/self/ns/pid -> pid:[4026532920]
1 root  root  0 May  7 20:31 /proc/self/ns/pid_for_children -> pid:[4026532920]
1 root  root  0 May  7 20:31 /proc/self/ns/time -> time:[4026531834]
1 root  root  0 May  7 20:31 /proc/self/ns/time_for_children -> time:[4026531834]
1 root  root  0 May  7 20:31 /proc/self/ns/user -> user:[4026531837]
1 root  root  0 May  7 20:31 /proc/self/ns/uts -> uts:[4026532916]
```

Ephemeral Container vs `kubectl exec`



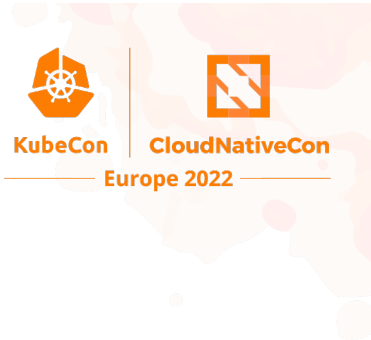
- Share PID namespace
- Share Network Namespace
- Share IPC, UTS

Ephemeral Containers



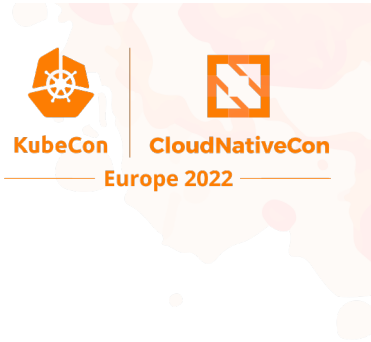
- Ephemeral containers are a sub-resource

Sub-Resource



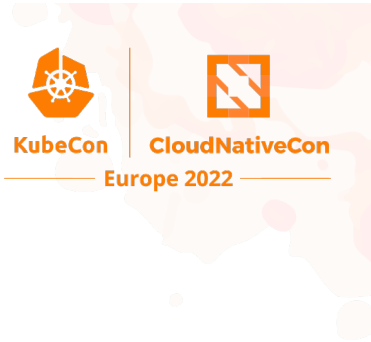
- Ephemeral containers are a sub-resource, "`ephemeralContainer`"
- Added to the pod spec `ephemeralContainers` property for each invocation
- Container exit status in the `ephemeralContainerStatuses` property
- Unique names must be used

Sub-Resource




```
$ kubectl get pod -n pg pg-postgresql-0 -o json | \
    jq '{"ephemeralContainers": [(.spec.ephemeralContainers[].name)], \
         "ephemeralContainerStatuses":
[(.status.ephemeralContainerStatuses[].name)]}'
{
  "ephemeralContainers": [
    "debug-postgresql",
    "debugger-n8bvb",
    "debugger-vfcxs"
  ],
  "ephemeralContainerStatuses": [
    "debug-postgresql",
    "debugger-n8bvb",
    "debugger-vfcxs"
  ]
}
```

Custom API calls

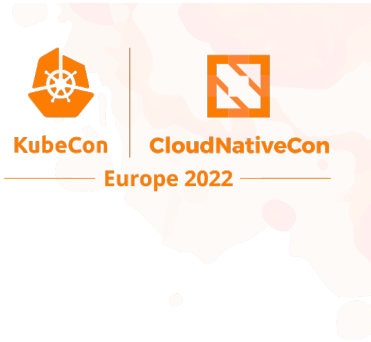


- Properties not available from `kubectl`
- Can mount container volumes
- Privileged containers
  - Set `securityContext`
- Cannot remove or change `ephemeralContainers` or `ephemeralContainerStatuses` property

Ephemeral Containers

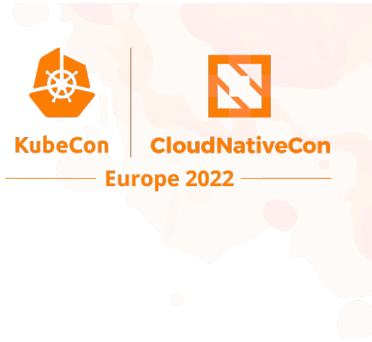

```
$ kubectl proxy &
Starting to serve on 127.0.0.1:8001

$ curl -v -XPATCH -H "Content-Type: application/json-patch+json" \
  'http://127.0.0.1:8001/api/v1/namespaces/pg/pods/pg-postgresql-0/ephemeralcontainers' \
  --data-binary @- << EOF
[{
  "op": "add", "path": "/spec/ephemeralContainers/-",
  "value": {
    "command":[ "/bin/sh" ],
    "stdin": true, "tty": true,
    "image": "busybox",
    "name": "debug-container-1",
    "volumeMounts": [{
      "mountPath": "/mnt",
      "name": "data" }],
    "targetContainerName": "postgresql" }}]
EOF
```

Ephemeral Containers




```
$ kubectl proxy &
Starting to serve on 127.0.0.1:8001

$ curl -v -XPATCH -H "Content-Type: application/json-patch+json" \
  'http://127.0.0.1:8001/api/v1/namespaces/pg/pods/pg-postgresql-0/ephemeralcontainers' \
  --data-binary @- << EOF
[{
  "op": "add", "path": "/spec/ephemeralContainers/-",
  "value": {
    "command":[ "/bin/sh" ],
    "stdin": true, "tty": true,
    "image": "busybox",
    "name": "debug-container-1",
    "volumeMounts": [{
      "mountPath": "/mnt",
      "name": "data" }],
    "targetContainerName": "postgresql" }}]
EOF
```

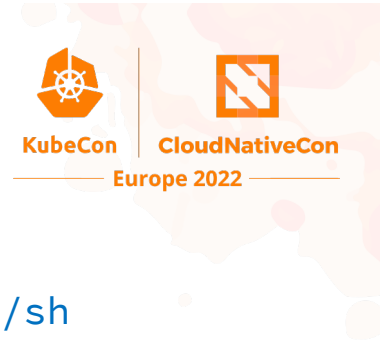
Namespaces when debug by attaching




```
$ kubectl exec -it -n pg pg-postgresql-0 -c postgresql -- /bin/sh
```

```
# ls -l /proc/self/ns/*
1 1001  root  0 May  7 20:27 /proc/self/ns/cgroup -> cgroup:[4026532921]
1 1001  root  0 May  7 20:27 /proc/self/ns/ipc -> ipc:[4026532917]
1 1001  root  0 May  7 20:27 /proc/self/ns/mnt -> mnt:[4026532919]
1 1001  root  0 May  7 20:27 /proc/self/ns/net -> net:[4026532837]
1 1001  root  0 May  7 20:27 /proc/self/ns/pid -> pid:[4026532920]
1 1001  root  0 May  7 20:27 /proc/self/ns/pid_for_children -> pid:[4026532920]
1 1001  root  0 May  7 20:27 /proc/self/ns/time -> time:[4026531834]
1 1001  root  0 May  7 20:27 /proc/self/ns/time_for_children -> time:[4026531834]
1 1001  root  0 May  7 20:27 /proc/self/ns/user -> user:[4026531837]
1 1001  root  0 May  7 20:27 /proc/self/ns/uts -> uts:[4026532916]
```

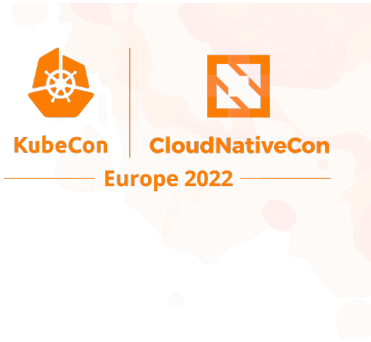
Ephemeral Containers




```
$ kubectl exec -it -n pg pg-postgresql-0 -c debug-container-1 -- /bin/sh
```

```
# ls -l /proc/self/ns/*
1 root  root  0 May  7 20:55 /proc/self/ns/cgroup -> cgroup:[4026532923]
1 root  root  0 May  7 20:55 /proc/self/ns/ipc -> ipc:[4026532917]
1 root  root  0 May  7 20:55 /proc/self/ns/mnt -> mnt:[4026532922]
1 root  root  0 May  7 20:55 /proc/self/ns/net -> net:[4026532837]
1 root  root  0 May  7 20:55 /proc/self/ns/pid -> pid:[4026532920]
1 root  root  0 May  7 20:55 /proc/self/ns/pid_for_children -> pid:[4026532920]
1 root  root  0 May  7 20:55 /proc/self/ns/time -> time:[4026531834]
1 root  root  0 May  7 20:55 /proc/self/ns/time_for_children -> time:[4026531834]
1 root  root  0 May  7 20:55 /proc/self/ns/user -> user:[4026531837]
1 root  root  0 May  7 20:55 /proc/self/ns/uts -> uts:[4026532916]
```

Ephemeral Containers

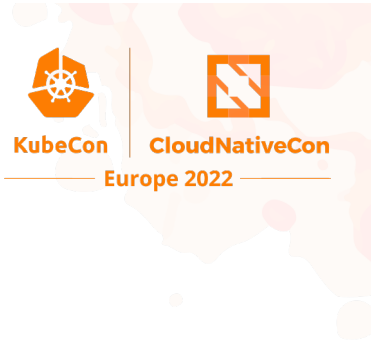



```
# df # can mount pod volumes ("volumes" property in pod)
Filesystem           1K-blocks      Used Available Use% Mounted on
overlay              123329088  61869768  55151536  53% /
tmpfs                    65536         0     65536   0% /dev
/dev/vda1            123329088  61869768  55151536  53% /mnt
/dev/vda1            123329088  61869768  55151536  53% /etc/hosts
...

# ls -1 /mnt/data
PG_VERSION
base
global
...
postgresql.auto.conf
postmaster.opts
postmaster.pid
```

Netshoot

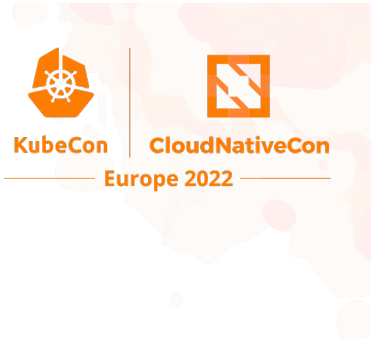
- Comprehensive toolset for diagnosis of network problems
- System level diagnosis tools
  - `strace`
  - `ltrace`
  - `tcpdump`
  - `...`

Ephemeral Containers



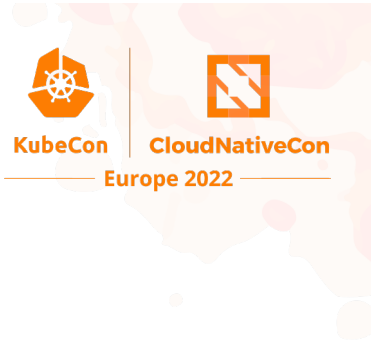

```
$ curl -v -XPATCH -H "Content-Type: application/json-patch+json" \
  'http://127.0.0.1:8001/api/v1/namespaces/pg/pods/pg-postgresql-0/ephemeralcontainers' \
  --data-binary @- << EOF
[{
  "op": "add", "path": "/spec/ephemeralContainers/-",
  "value": {
    "command":[ "/bin/sh" ],
    "stdin": true, "tty": true,
    "image": "nicolaka/netshoot",
    "name": "debug-container-2",
    "securityContext": { "privileged": true },
    "volumeMounts": [{
      "mountPath": "/mnt",
      "name": "data" }],
    "targetContainerName": "postgresql" }}]
EOF
```

Ephemeral Containers




```
$ curl -v -XPATCH -H "Content-Type: application/json-patch+json" \
  'http://127.0.0.1:8001/api/v1/namespaces/pg/pods/pg-postgresql-0/ephemeralcontainers' \
  --data-binary @- << EOF
[{
  "op": "add", "path": "/spec/ephemeralContainers/-",
  "value": {
    "command":[ "/bin/sh" ],
    "stdin": true, "tty": true,
    "image": "nicolaka/netshoot",
    "name": "debug-container-2",
    "securityContext": { "privileged": true },
    "volumeMounts": [{
      "mountPath": "/mnt",
      "name": "data" }],
    "targetContainerName": "postgresql" }}]
EOF
```

Ephemeral Containers



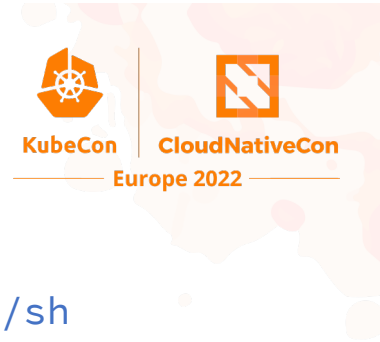

```
# strace -p $(pgrep postgresql) 2>&1 | head
strace: Process 1 attached
select(8, [5 6 7], NULL, NULL, {tv_sec=55, tv_usec=50862 ...
rt_sigprocmask(SIG_SETMASK, ~[ILL TRAP ABRT BUS FPE SEGV ...
accept(5, {sa_family=AF_INET, sin_port=htons(36538), sin_...
getsockname(9, {sa_family=AF_INET, sin_port=htons(5432), ...

# exit
```

Namespaces when debug by attaching (slide 40)


```
$ kubectl exec -it -n pg pg-postgresql-0 -c postgresql -- /bin/sh
```

```
# ls -l /proc/self/ns/*
1 1001  root  0 May  7 20:27 /proc/self/ns/cgroup -> cgroup:[4026532921]
1 1001  root  0 May  7 20:27 /proc/self/ns/ipc -> ipc:[4026532917]
1 1001  root  0 May  7 20:27 /proc/self/ns/mnt -> mnt:[4026532919]
1 1001  root  0 May  7 20:27 /proc/self/ns/net -> net:[4026532837]
1 1001  root  0 May  7 20:27 /proc/self/ns/pid -> pid:[4026532920]
1 1001  root  0 May  7 20:27 /proc/self/ns/pid_for_children -> pid:[4026532920]
1 1001  root  0 May  7 20:27 /proc/self/ns/time -> time:[4026531834]
1 1001  root  0 May  7 20:27 /proc/self/ns/time_for_children -> time:[4026531834]
1 1001  root  0 May  7 20:27 /proc/self/ns/user -> user:[4026531837]
1 1001  root  0 May  7 20:27 /proc/self/ns/uts -> uts:[4026532916]
```

Ephemeral Containers



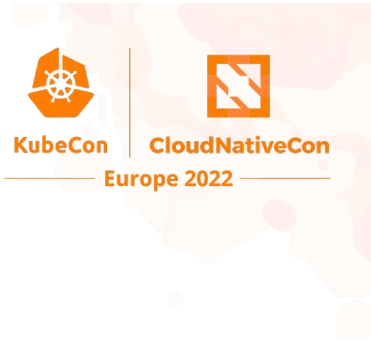

```
$ kubectl exec -it -n pg pg-postgresql-0 -c debug-container-2 -- /bin/sh
```

```
# ls -l /proc/self/ns/*
1 root  root  0 May  7 21:01 /proc/self/ns/cgroup -> cgroup:[4026532459]
1 root  root  0 May  7 21:01 /proc/self/ns/ipc -> ipc:[4026532917]
1 root  root  0 May  7 21:01 /proc/self/ns/mnt -> mnt:[4026532924]
1 root  root  0 May  7 21:01 /proc/self/ns/net -> net:[4026532837]
1 root  root  0 May  7 21:01 /proc/self/ns/pid -> pid:[4026532920]
1 root  root  0 May  7 21:01 /proc/self/ns/pid_for_children -> pid:[4026532920]
1 root  root  0 May  7 21:01 /proc/self/ns/time -> time:[4026531834]
1 root  root  0 May  7 21:01 /proc/self/ns/time_for_children -> time:[4026531834]
1 root  root  0 May  7 21:01 /proc/self/ns/user -> user:[4026531837]
1 root  root  0 May  7 21:01 /proc/self/ns/uts -> uts:[4026532916]
```

Linux Namespaces on node (docker) (slide 30)

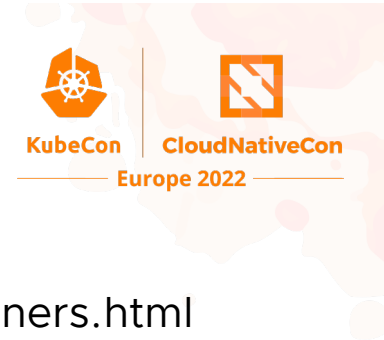

```
# # get namespaces for PID 1
# ls -l /proc/1/ns/*
lrwxr ... /proc/1/ns/cgroup -> cgroup:[4026532459]
lrwxr ... /proc/1/ns/ipc -> ipc:[4026532372]
lrwxr ... /proc/1/ns/mnt -> mnt:[4026532370]
lrwxr ... /proc/1/ns/net -> net:[4026532376]
lrwxr ... /proc/1/ns/pid -> pid:[4026532374]
```

Distroless Containers



- Separate production release from tools
- Prod and tools projects become version independent
  - Can select tools appropriate to task
  - No longer dependent on versions in container
- Distroless containers are faster to build
  - Fewer layers

Notes



https://opensource.googleblog.com/2022/01/Introducing+Ephemeral+Containers.html
https://www.ianlewis.org/en/what-are-kubernetes-pods-anyway
https://icicimov.github.io/blog/virtualization/Linux-Container-Basics/
https://bit.ly/38Nvd6P
https://en.wikipedia.org/wiki/Linux_namespaces
https://en.wikipedia.org/wiki/Cgroups
https://github.com/GoogleContainerTools/distroless
https://kind.sigs.k8s.io
https://github.com/nicolaka/netshoot
https://github.com/eldadru/ksniff