KubeCon CloudNativeCon North America 2026

#KubeCon #CloudNativeCon

When the Clock Lies

How an 8-Minute Time Skew Locked a Kubernetes Cluster for a Day



WHO AM I

Akhil Mohan

@akhilerm

Software Engineer @ VMware by Broadcom

Maintainer @ containerd, k8s publishing-bot

A storage driver pod stuck for 19 hours

```
$ kubectl get pods -A | grep -Ev 'Run|Completed'
vmware-system-csi  vsphere-csi-node-ltf2p  1/3  CreateContainerError  3 (19h ago)  2d19h
```

```
$ kubectl logs vsphere-csi-node-ltf2p -c node-driver-registrar
I0304 09:52:02.769181  main.go:172] "Attempting to open a gRPC connection" csiAddress="/csi/csi.sock"
I0304 09:52:12.769775  connection.go:253] "Still connecting" address="unix:///csi/csi.sock"
I0304 09:52:22.769670  connection.go:253] "Still connecting" address="unix:///csi/csi.sock"
I0304 09:43:39.782750  connection.go:253] "Still connecting" address="unix:///csi/csi.sock"
E0304 09:43:39.783123  main.go:176] "Error connecting to CSI driver" err="context deadline exceeded"
```

```
Warning  Failed  3m13s (x5327 over 19h)  kubelet  Error: failed to reserve container name
"node-driver-registrar_vsphere-csi-node-ltf2p_vmware-system-csi_<uid>_2": name "…_2" is reserved for "65cfb29e…"
```

What happened?

```
$ crictl ps -a | grep vsphere-csi-node-ltf2p
2cffdb3c46b4c  24 hours ago  Running  liveness-probe         1  vsphere-csi-node-ltf2p
2f93f91a450ad  24 hours ago  Exited   vsphere-csi-node       1  vsphere-csi-node-ltf2p
d0ad1a479518f  24 hours ago  Exited   node-driver-registrar  1  vsphere-csi-node-ltf2p
df1324b5c5ce2  24 hours ago  Running  vsphere-csi-node       2  vsphere-csi-node-ltf2p
65cfb29edbdb1  24 hours ago  Running  node-driver-registrar  2  vsphere-csi-node-ltf2p
```

What happened?

```
$ crictl ps -a | grep vsphere-csi-node-ltf2p
2cffdb3c46b4c  24 hours ago  Running  liveness-probe          1  vsphere-csi-node-ltf2p
2f93f91a450ad  24 hours ago  Exited   vsphere-csi-node        1  vsphere-csi-node-ltf2p
d0ad1a479518f  24 hours ago  Exited   node-driver-registrar   1  vsphere-csi-node-ltf2p
df1324b5c5ce2  24 hours ago  Running  vsphere-csi-node        2  vsphere-csi-node-ltf2p
65cfb29edbdb1  24 hours ago  Running  node-driver-registrar   2  vsphere-csi-node-ltf2p
```

```
Warning  Failed  3m13s (x5327 over 19h)  kubelet  Error: failed to reserve container name
"node-driver-registrar_vsphere-csi-node-ltf2p_vmware-system-csi_<uid>_2": name "…_2" is reserved for "65cfb29e…"
```

PROBLEM SETUP

The Environment

CLUSTER	A VKS guest cluster on ESXi VMs , containerd 2.2 , Kubernetes 1.35
TRIGGER	Power failed, the ESXi host rebooted ; 13 VMs ; six clusters
SPREAD	Kubernetes 1.33 - 1.36 ; Photon 5 and Ubuntu 24.04
OUTCOME	One cluster stayed **stuck**
STARTED AS	CSI Node Daemon issue, everything else recovered

Mitigations that MIGHT work

1. restart kubelet and containerd
2. crictl rm <container ID>
3. kubectl delete pod
4. delete the node

How kubelet names a container

```
<container>_<pod>_<namespace>_<pod-uid>_<attempt>
```

annotation io.kubernetes.container.restartCount

Where does *attempt* come from?

```
restartCount := 0
containerStatus := podStatus.FindContainerStatusByName(container.Name)
if containerStatus ≠ nil {
    restartCount = containerStatus.RestartCount + 1
}
```

pkg/kubelet/kuberuntime/kuberuntime_container.go

```
new instance attempt = new instance restartCount = previous instance restartCount + 1
```

How a pod reaches the API server

- **ClusterIP** exists as iptables/nftables rules written by **kube-proxy**
- Every client authenticates with a **projected service-account token**

TOO EARLY

```
dial tcp 10.96.0.1:443: i/o timeout
```

UNAUTHENTICATED

```
401 Unauthorized
```

Journey starts…

The CSI driver

Three containers, one socket

node-driver-registrar · liveness-probe

```
I0304 09:52:12.769775  connection.go:253] "Still connecting" address="unix:///csi/csi.sock"
E0304 09:43:39.783123  main.go:176] "Error connecting to CSI driver" err="context deadline exceeded"
```

Three containers, one socket

node-driver-registrar · liveness-probe

```
I0304 09:52:12.769775  connection.go:253] "Still connecting" address="unix:///csi/csi.sock"
E0304 09:43:39.783123  main.go:176] "Error connecting to CSI driver" err="context deadline exceeded"
```

vsphere-csi-node

```
failed to fetch configmap … Get "https://10.96.0.1:443/api/v1/namespaces/vmware-system-csi/configmaps/…":
    dial tcp 10.96.0.1:443: i/o timeout
Failed to initialize the orchestrator
```

Three containers, one socket

node-driver-registrar · liveness-probe

```
I0304 09:52:12.769775  connection.go:253] "Still connecting" address="unix:///csi/csi.sock"
E0304 09:43:39.783123  main.go:176] "Error connecting to CSI driver" err="context deadline exceeded"
```

vsphere-csi-node

```
failed to fetch configmap … Get "https://10.96.0.1:443/api/v1/namespaces/vmware-system-csi/configmaps/…":
    dial tcp 10.96.0.1:443: i/o timeout
Failed to initialize the orchestrator
```

kube-proxy

```
E0304 09:52:03.085453 reflector.go:204] "Failed to watch" err="failed to list *v1.Node: Unauthorized"
E0304 09:49:01.444028 reflector.go:204] "Failed to watch" err="failed to list *v1.EndpointSlice: Unauthorized"
```

CSI, just the symptom

CONCLUSION

- *containerd having stale container name reservation*

→ SUSPICION MOVES TO CONTAINERD · UNAUTHORIZED PARKED

Container runtime

Is the reservation stale?

```
$ crictl ps -a | grep 65cfb29e
65cfb29edbdb1  24 hours ago  Running  node-driver-registrar  2  vsphere-csi-node-ltf2p
```

The id in the error **exists**.

Is the reservation stale?

```
$ crictl ps -a | grep 65cfb29e
65cfb29edbdb1   24 hours ago   Running   node-driver-registrar   2   vsphere-csi-node-ltf2p
```

The id in the error **exists**.

```
$ crictl inspect 65cfb29edbdb1 | jq '.status.state, .info.config.metadata.attempt'
"CONTAINER_RUNNING"
2
```

CONTAINER_RUNNING, attempt **2**.

Is the reservation stale?

```
$ crictl inspect 65cfb29edbdb1 | jq '.status.state, .info.config.metadata.attempt'
"CONTAINER_RUNNING"
2
```

```
$ crictl ps -a | grep vsphere-csi-node-ltf2p
2cffdb3c46b4c  24 hours ago  Running  liveness-probe         1  vsphere-csi-node-ltf2p
2f93f91a450ad  24 hours ago  Exited   vsphere-csi-node       1  vsphere-csi-node-ltf2p
d0ad1a479518f  24 hours ago  Exited   node-driver-registrar  1  vsphere-csi-node-ltf2p
df1324b5c5ce2  24 hours ago  Running  vsphere-csi-node       2  vsphere-csi-node-ltf2p
65cfb29edbdb1  24 hours ago  Running  node-driver-registrar  2  vsphere-csi-node-ltf2p
```

containerd.log

```
CreateContainer within sandbox "85cf5329…" for name:"node-driver-registrar" attempt:2 failed
error="failed to reserve container name "node-driver-registrar_vsphere-csi-node-ltf2p_vmware-system-csi_<uid>_2":
       name "…_2" is reserved for "65cfb29edbdb…""
```

containerd is the messenger

CONCLUSION

containerd worked as expected.

OPEN QUESTIONS

A Why did the containers crash after boot at all?

B Why does kubelet want attempt 2 again?

→ SUSPICION MOVES TO KUBE-PROXY AND THE API SERVER

kube-apiserver

Why was kube-proxy Unauthorized?

kube-apiserver

```
E0304 09:48:13.410388  authentication.go:75] "Unable to authenticate the request"
  err="[invalid bearer token, service account token is not valid yet]"
```

Why was kube-proxy Unauthorized?

kube-apiserver

```
E0304 09:48:13.410388  authentication.go:75] "Unable to authenticate the request"
  err="[invalid bearer token, service account token is not valid yet]"
```

Not valid yet.

Whose clock?

MINTING APISERVER

stamps **nbf** from **its** clock



`token · nbf = iat`

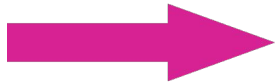
VALIDATING APISERVER

checks it against **its own** clock

Whose clock?

MINTING APISERVER · cp-A

on the rebooted host



token · nbf = iat

VALIDATING · cp-B, cp-C

not valid yet

journald: the smoking line

```
$ dmesg
[   45.466264] systemd-journald[359]: Time jumped backwards, rotating.
```

journald: the smoking line

```
$ dmesg
[   45.466264] systemd-journald[359]: Time jumped backwards, rotating.
```

45 seconds after boot.

AGAIN

The log you already saw

```
$ kubectl logs vsphere-csi-node-ltf2p -c node-driver-registrar
I0304 09:52:02.769181  main.go:172] "Attempting to open a gRPC connection" csiAddress="/csi/csi.sock"
I0304 09:52:12.769775  connection.go:253] "Still connecting" address="unix:///csi/csi.sock"
I0304 09:52:22.769670  connection.go:253] "Still connecting" address="unix:///csi/csi.sock"
I0304 09:43:39.782750  connection.go:253] "Still connecting" address="unix:///csi/csi.sock"
E0304 09:43:39.783123  main.go:176] "Error connecting to CSI driver" err="context deadline exceeded"
```

09:52:02 → 09:52:12 → 09:52:22 → 09:43:39

AGAIN

The log you already saw

```
$ kubectl logs vsphere-csi-node-ltf2p -c node-driver-registrar
I0304 09:52:02.769181  main.go:172] "Attempting to open a gRPC connection" csiAddress="/csi/csi.sock"
I0304 09:52:12.769775  connection.go:253] "Still connecting" address="unix:///csi/csi.sock"
I0304 09:52:22.769670  connection.go:253] "Still connecting" address="unix:///csi/csi.sock"
I0304 09:43:39.782750  connection.go:253] "Still connecting" address="unix:///csi/csi.sock"
E0304 09:43:39.783123  main.go:176] "Error connecting to CSI driver" err="context deadline exceeded"
```

−533 s

/etc/chrony.conf

```
makestep 1.0 3
```

RULE	Step if offset > 1 s during the first 3 updates; slew afterwards.
WHY A STEP	Slewing 533 s at the default rate ≈ 1 h 47 min; Hence step

Who started first?

`kubelet.service`	active at **09:51:56** - 3 s after boot
`chrony-wait.service`	ran **33.084 s**
`ordering`	kubelet.service has **no After= / Requires=** on it

Who started first?

`kubelet.service`	active at **09:51:56** - 3 s after boot
`chrony-wait.service`	ran **33.084 s**
`ordering`	kubelet.service has **no After= / Requires=** on it

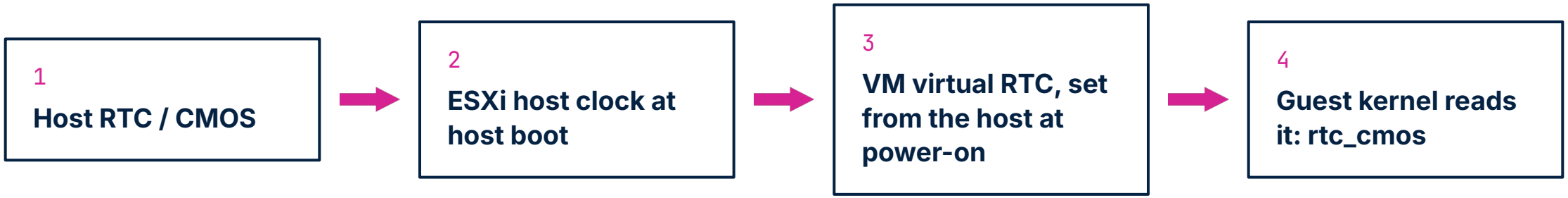
```
kubelet.service:    After=network-online.target
containerd.service: After=network.target dbus.service
```

Where the 8 minutes came from

```
[    0.783479] rtc_cmos 00:01: setting system clock to 2026-03-04T09:51:41 UTC (1772617901)
[   45.466264] systemd-journald[359]: Time jumped backwards, rotating.
```

Where the 8 minutes came from

```
[    0.783479] rtc_cmos 00:01: setting system clock to 2026-03-04T09:51:41 UTC (1772617901)
[   45.466264] systemd-journald[359]: Time jumped backwards, rotating.
```



Service-account tokens

```
// JWT payload
{
  "nbf": not before,
  "iat": issued at,
  "exp": expiry
}
```

Service-account tokens

```
// JWT payload
{
  "nbf": not before,
  "iat": issued at,
  "exp": expiry
}
```

PAYLOAD	3 timestamps
SPEC	RFC 7519 (JWT) - leeway depends on the implementation.
VALIDATE	go-jose DefaultLeeway = **1 * time.Minute** — never overridden by Kubernetes.

Mystery A, on a timeline

WRONG CLOCK · ON THE VM

REAL TIME

09:51:53
boot

09:52:01
tokens minted, kube-proxy and CSI: iat = nbf

09:52:03
CSI driver dials the ClusterIP

09:52:04
kube-proxy's first rule sync

Mystery A, on a timeline

09:52:3x → 09:43:39 · chrony steps -533 s · the CSI containers exit

WRONG CLOCK · ON THE VM

REAL TIME

09:51:53
boot

09:52:01
tokens minted, kube-proxy and CSI: iat = nbf

09:52:03
CSI driver dials the ClusterIP

09:52:04
kube-proxy's first rule sync

Mystery A, on a timeline

09:52:3x → 09:43:39 · chrony steps -533 s · the CSI containers exit

WRONG CLOCK · ON THE VM

09:51:53
boot

09:52:01
tokens minted, kube-proxy and CSI: iat = nbf

09:52:03
CSI driver dials the ClusterIP

09:52:04
kube-proxy's first rule sync

REAL TIME

09:43:52 – 09:50:58
21 attempts - through a **working** ClusterIP. The driver's own token is from the future.

21 × 401

Mystery A, on a timeline

09:52:3x → 09:43:39 · chrony steps -533 s · the CSI containers exit

WRONG CLOCK · ON THE VM

09:51:53
boot

09:52:01
tokens minted, kube-proxy and CSI: iat = nbf

09:52:03
CSI driver dials the ClusterIP

09:52:04
kube-proxy's first rule sync

REAL TIME

09:43:52 – 09:50:58
21 attempts - through a **working** ClusterIP. The driver's own token is from the future.

09:51:01
real time reaches **nbf - 60 s**

09:51:09.612
200 OK · socket appears · 3/3 Running

21 × 401

SUSPECT 2 - MYSTERY A, CLOSED

Every layer behaved as designed

CONCLUSION

→ *Host reboot*

→ *Host RTC ahead*

→ *Guest boots 8 min in the future*

→ *Tokens minted from the future*

→ *Locked out until real time reaches nbf - 60 s*

→ *All pods start Running*

*… yet the CSI pod stayed **CreateContainerError** for **19 hours***

→ MYSTERY B · SUSPICION MOVES TO KUBELET

Almost there…

What does kubelet think the latest container is?

```
$ kubectl get pod vsphere-csi-node-ltf2p -o yaml
restartCount: 1
lastState.terminated: { startedAt: 09:52:02, finishedAt: 09:43:39 }
state.waiting.reason: CreateContainerError
```

What does kubelet think the latest container is?

```
$ kubectl get pod vsphere-csi-node-ltf2p -o yaml
restartCount: 1
lastState.terminated: { startedAt: 09:52:02, finishedAt: 09:43:39 }   # finished before it started
state.waiting.reason: CreateContainerError
```

FindContainerStatusByName returns the first match

- Iterate **ContainerStatuses**, return first with a matching name.
- “Latest” means “first in the list”.

```
for _, containerStatus := range podStatus.ContainerStatuses {
    if containerStatus.Name == containerName {
        return containerStatus
    }
}
```

pkg/kubelet/container/runtime.go

FindContainerStatusByName returns the first match

- Iterate **ContainerStatuses**, return first with a matching name.
- “Latest” means “first in the list”.

```
for _, containerStatus := range podStatus.ContainerStatuses {
    if containerStatus.Name == containerName {
        return containerStatus
    }
}
```

pkg/kubelet/container/runtime.go

Who sorts the list?

Found the culprit

The sort

```
func (c containerStatusByCreated) Less(i, j int) bool {
    return c[i].CreatedAt.After(c[j].CreatedAt)
}
```

pkg/kubelet/kuberuntime/helpers.go

The sort

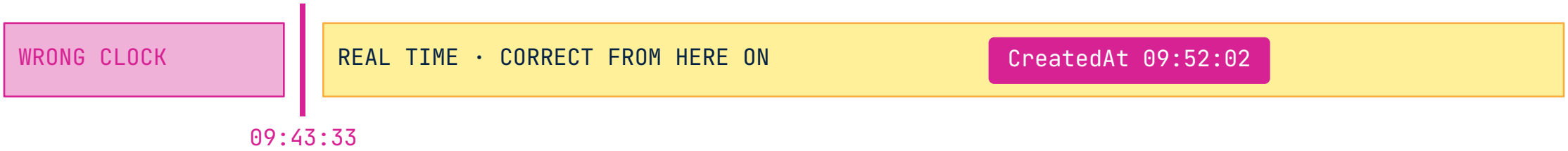
```
func (c containerStatusByCreated) Less(i, j int) bool {
    return c[i].CreatedAt.After(c[j].CreatedAt)
}
```

pkg/kubelet/kuberuntime/helpers.go

```
status := containerstore.Status{CreatedAt: time.Now().UnixNano()}
// containerd · internal/cri/server/container_create.go:419
```

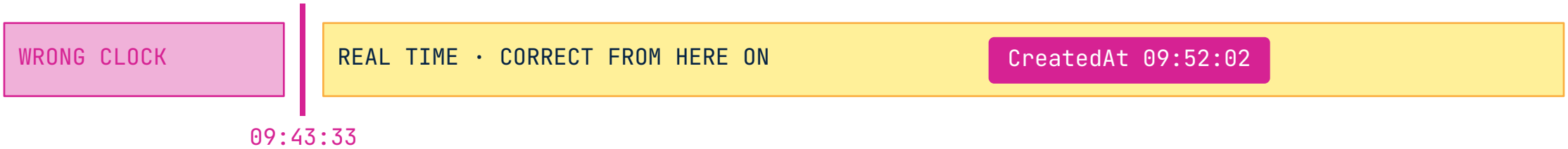
```
CreatedAt: time.Unix(0, status.CreatedAt)
// kubelet · pkg/kubelet/kuberuntime/kuberuntime_container.go:750
```

The loop - on a correct clock



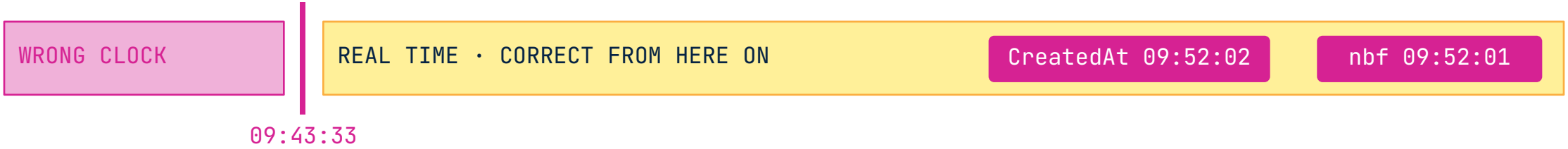
1	09:52:02	attempt 1 created on the future clock - crashes at 09:43:39 real time.

The loop - on a correct clock



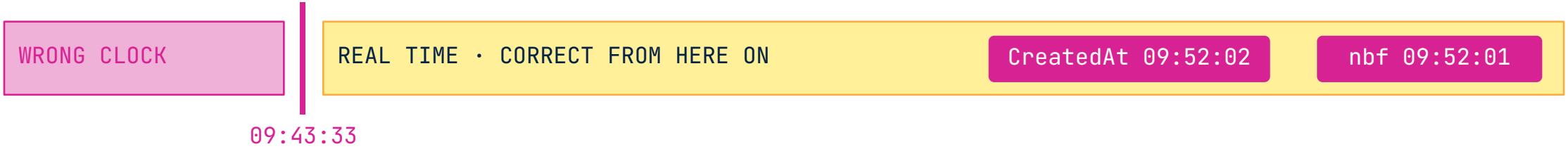
1	09:52:02	attempt 1 created on the future clock - crashes at 09:43:39 real time.
2	09:43:52	kubelet creates attempt 2. CreatedAt 09:43:52 < 09:52:02 → attempt 2 sorts as **older**.

The loop - on a correct clock



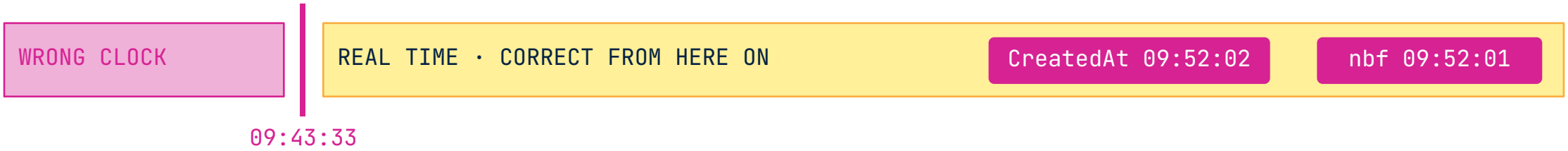
1	09:52:02	attempt 1 created on the future clock - crashes at 09:43:39 real time.
2	09:43:52	kubelet creates attempt 2. CreatedAt 09:43:52 < 09:52:02 → attempt 2 sorts as **older**.
3	+ 1 s	attempt 2 crashes: future-dated token. Sync: “latest” = attempt 1 → RestartCount + 1 = 2 → CreateContainer(..._2) → **reserved**.

The loop - on a correct clock



1	09:52:02	attempt 1 created on the future clock - crashes at 09:43:39 real time.
2	09:43:52	kubelet creates attempt 2. CreatedAt 09:43:52 < 09:52:02 → attempt 2 sorts as **older**.
3	+ 1 s	attempt 2 crashes: future-dated token. Sync: “latest” = attempt 1 → RestartCount + 1 = 2 → CreateContainer(..._2) → **reserved**.
4	~ 1 s	PLEG deletor: keeps attempt 1, removes attempt 2, create again. Repeats **22 times** in 7 minutes.

The loop - on a correct clock



1	09:52:02	attempt 1 created on the future clock - crashes at 09:43:39 real time.
2	09:43:52	kubelet creates attempt 2. CreatedAt 09:43:52 < 09:52:02 → attempt 2 sorts as **older**.
3	+ 1 s	attempt 2 crashes: future-dated token. Sync: “latest” = attempt 1 → RestartCount + 1 = 2 → CreateContainer(..._2) → **reserved**.
4	~ 1 s	PLEG deletor: keeps attempt 1, removes attempt 2, create again. Repeats **22 times** in 7 minutes.
5	09:51:09	attempt 2 stays Running - and the loop **freezes.** Last successful create: **53 seconds** before real time crossed 09:52:02.

Four layers, one host reboot

CONCLUSION

→ *Containers sorted by **CreatedAt***

→ *sync picks the “latest” status by it*

→ *RestartCount + 1 trusts that pick*

→ *the PLEG deletor keeps the “newest” corpse by it*

\+ a **60-second** token leeway that put attempt 2 inside the skew window

= a non-recoverable loop

→ THE FIX

Single line of systemd

```
 [Unit]
 Wants=network-online.target
-After=network-online.target
+After=network-online.target chrony-wait.service
```

Three options, one survivor

`Requires=chrony-wait`	REJECTED

Three options, one survivor

`Requires=chrony-wait`	REJECTED
`Wants=chrony-wait`	REJECTED

Three options, one survivor

`Requires=chrony-wait`	REJECTED
`Wants=chrony-wait`	REJECTED
`After=chrony-wait only`	CHOSEN

chrony-wait.service

```
Type=oneshot
ExecStart=/usr/bin/chronyc waitsync 0 0.1 0.0 1
TimeoutStartSec=180
RemainAfterExit=yes
```

UPSTREAM · kubernetes/kubernetes

Past Attempts !

When the time of a node changes to earlier, the pod may be "CreateContainerError" and can't recover itself #83050

Closed

[kubelet] time shift and `failed to reserve container name` #134153

Closed

sort pod sandbox by attempt number #83701

Closed

Conversation Commits 1 Checks Files changed

order sandbox by attempt or create time #130551

Merged

Conversation Commits 1 Checks Files changed

Fix time shift handling in kubelet pod cache #134204

Closed

Conversation Commits 2 Checks Files changed

Should also order containerStatus by attempt or create time #138174

Open #138191

fix: Should also order containerStatus by attempt or create time #138191

Open

Conversation Commits 1 Checks Files changed

fix(kubelet): order API container statuses by restart count #143483

Open

Conversation Commits 1 Checks Files changed

Kubelet can report an old container instance in Pod status after a clock rollback #143482

Open #143483

Few questions that came across.

? Why only the CSI node pods?

hostNetwork: start at boot, need the API at once, crash fast.

? Why stuck only once it was healthy?

A running attempt 2 is never deleted or garbage-collected.

? Was storage broken for 19 hours?

No. Serving RPCs. Only kubelet’s view of the pod was wrong, and daemonset never became ready

? Why only this cluster, of six?

Its attempt 2 landed inside the 61-s window - 53 s in. The others landed past the stamp.

TAKEAWAYS

What you can do?

- chrony-wait / systemd-time-wait-sync **enabled**
- kubelet.service; After=chrony-wait.service; Decide **fail-open** vs **fail-closed** deliberately.

Thank you.

Akhil Mohan · @akhilerm