
Allet juht, Atze!
Kubernetes & Chill
with automated infrastructure
KCD Berlin | 2022-06-30

Hi, I'm Danielle

Danielle Tal
Program Manager,
Microsoft

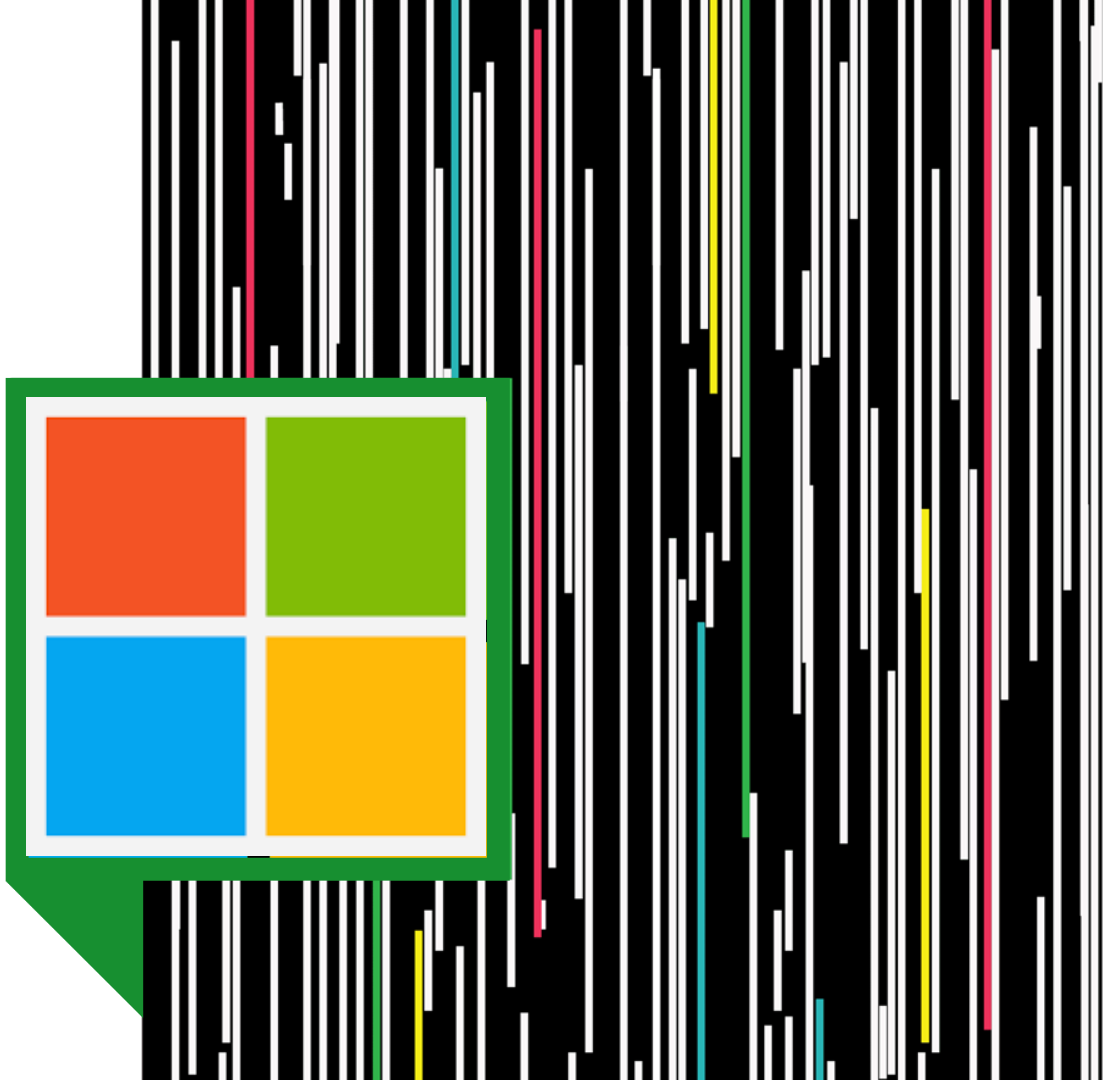
Github: miao0miao
Email: danittal@microsoft.com

Hi, I'm Thilo

Thilo Fromm

Engineering manager, Microsoft

Github: t-lo
Mastodon: @thilo@fromm.social
Twitter: ThiloFM
Email: thilofromm@microsoft.com



Fei|er|abend

1) Free time following a day of work.

"*To enjoy one's Feierabend.*"

2) End of work / closing time.

"*I'm calling Feierabend, heading home!*"

3) Vigorous shut-down of an ongoing activity.

"*Stop it, no more, that's it – Feierabend!*"

Container-optimised Linux History

Container-optimised Linux History

CoreOS Container Linux (Oct. 2013 – May 2020)

(derived from Chromium OS, which was derived from Gentoo)

Container-optimised Linux History

CoreOS Container Linux (Oct. 2013 – May 2020)

(derived from Chromium OS, which was derived from Gentoo)

Fedora CoreOS / Red Hat CoreOS (June 2018)

(based on Fedora, limited compatibility to CoreOS)

Container-optimised Linux History

CoreOS Container Linux (Oct. 2013 – May 2020)

(derived from Chromium OS, which was derived from Gentoo)

Fedora CoreOS / Red Hat CoreOS (June 2018)

(based on Fedora, limited compatibility to CoreOS)

Flatcar Container Linux (Nov. 2019)

(Friendly fork, fully compatible to CoreOS)

Container-optimised Linux?

Container-optimised Linux?

Your OS is infrastructure

Should be boring (not exciting), just work

Let you focus on your business logic

If a light switch needs constant attention,

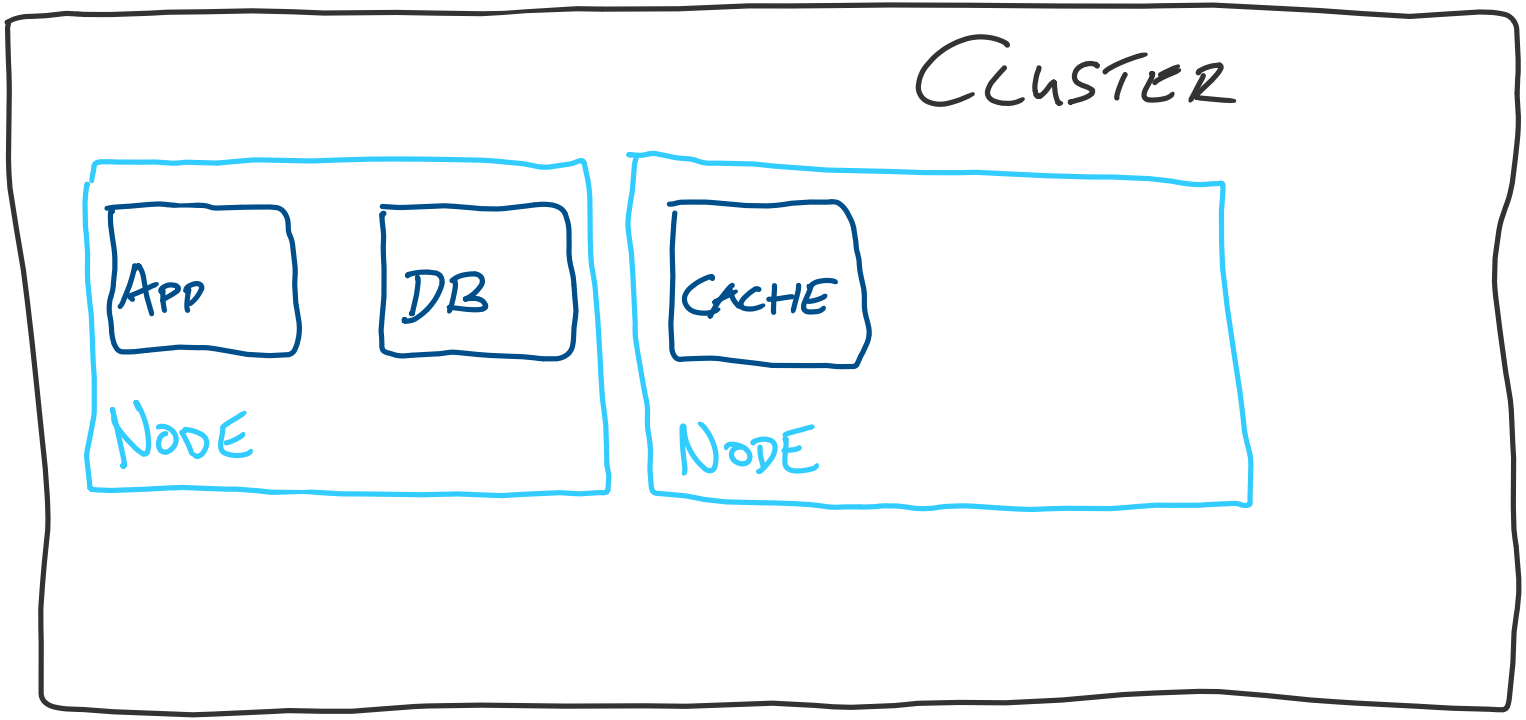
are you ever going to enjoy your fancy home cinema?

Container isolation applied to the OS?

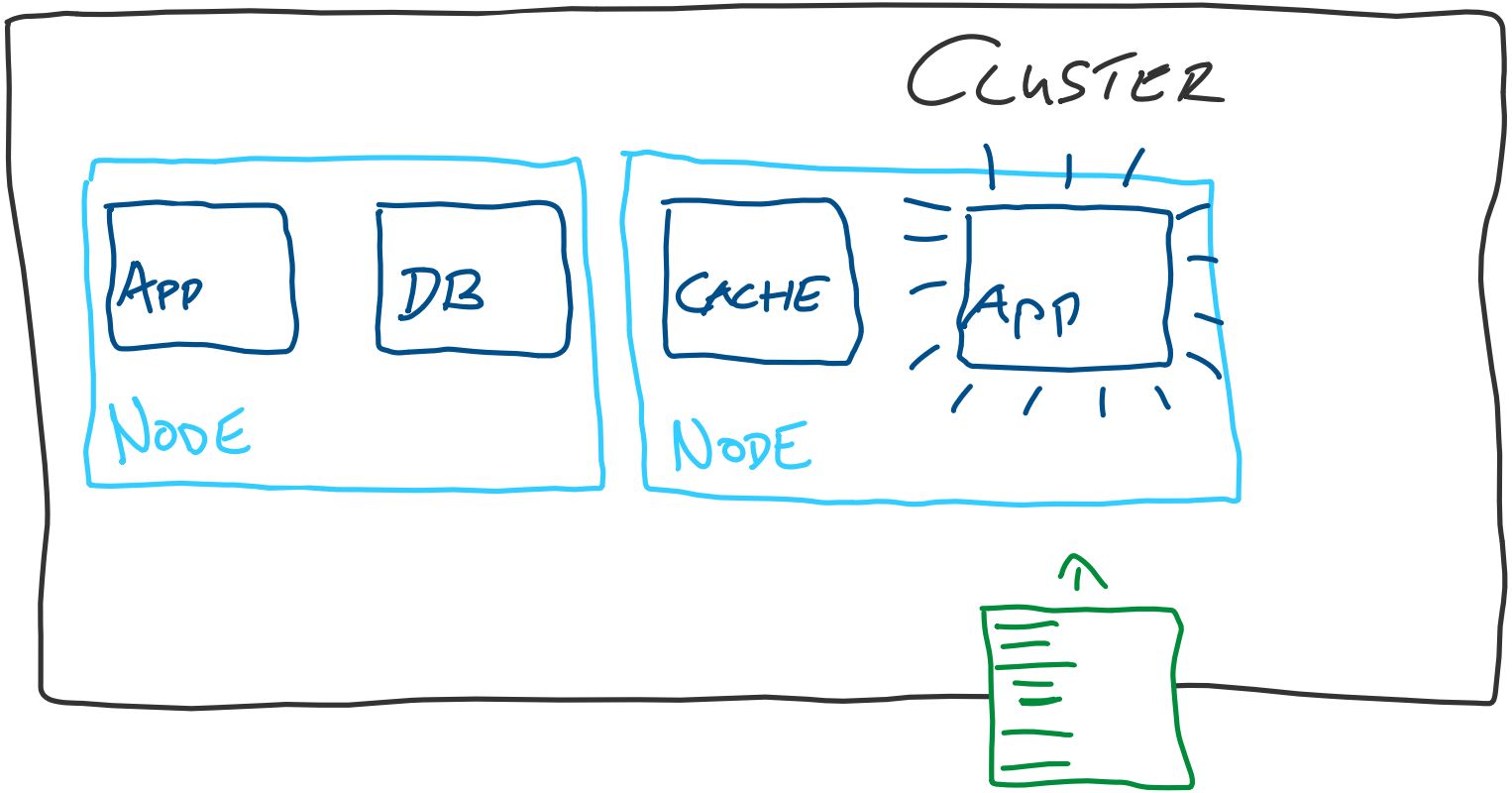
We isolate apps from each other, and from the OS.

Why not use that to make the OS yet another app?

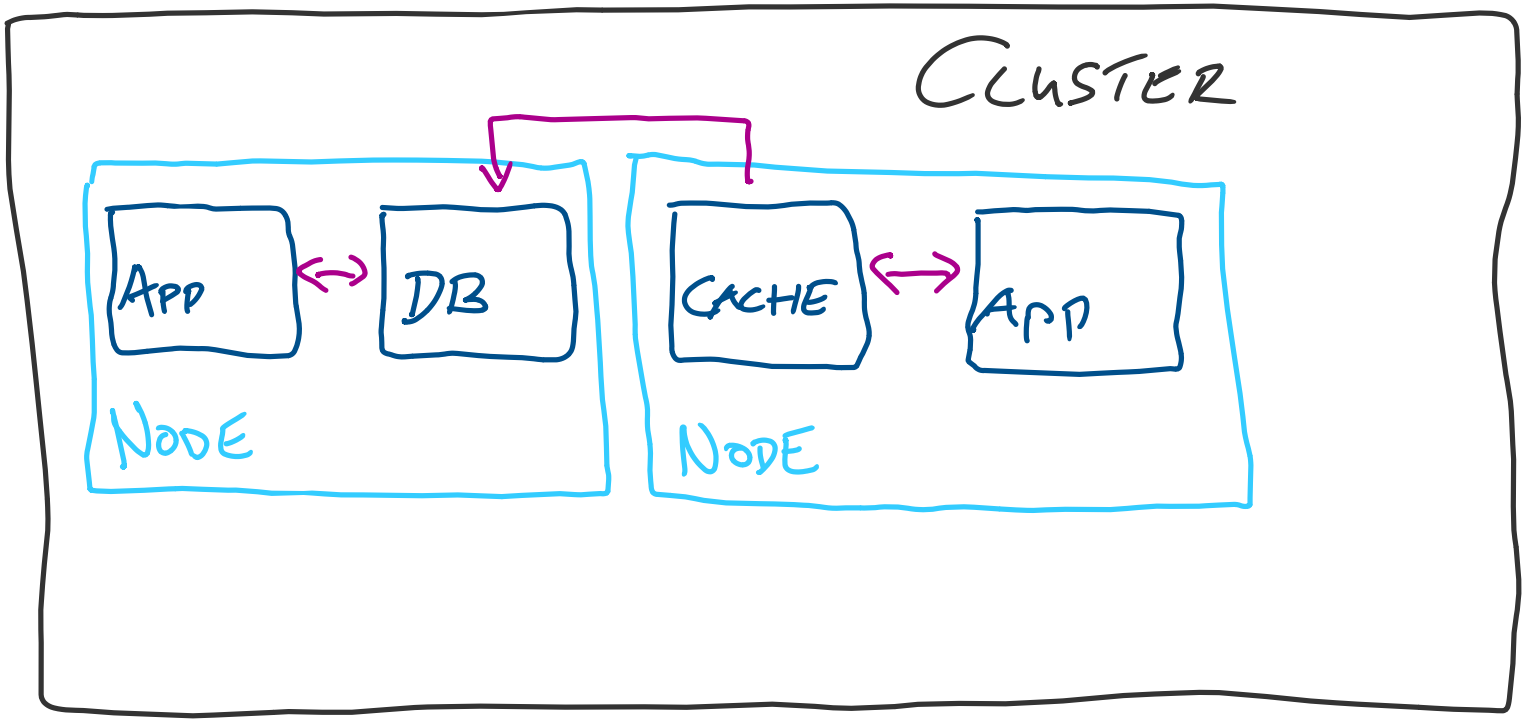
Container-optimised Linux?



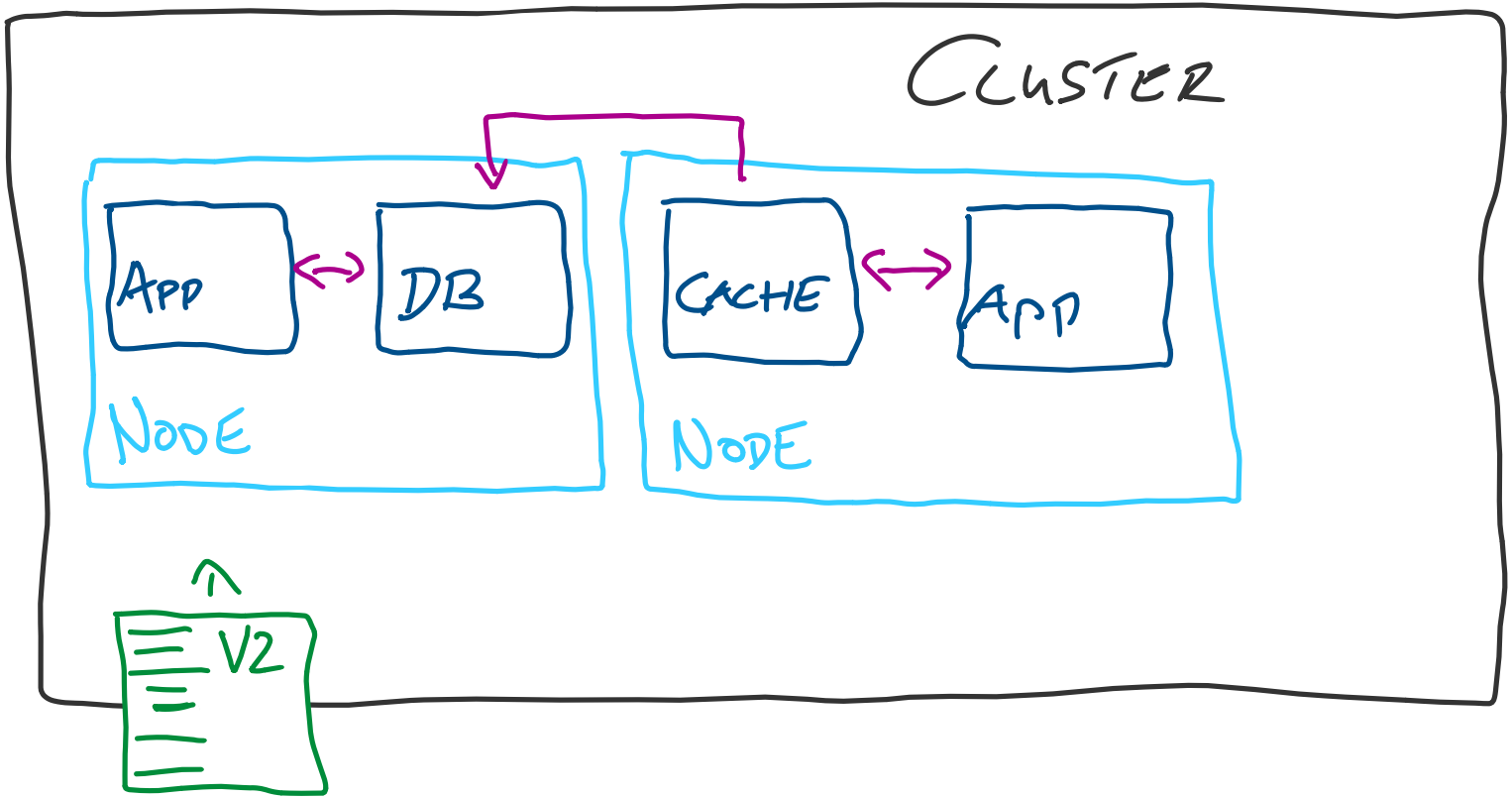
Container-optimised Linux?



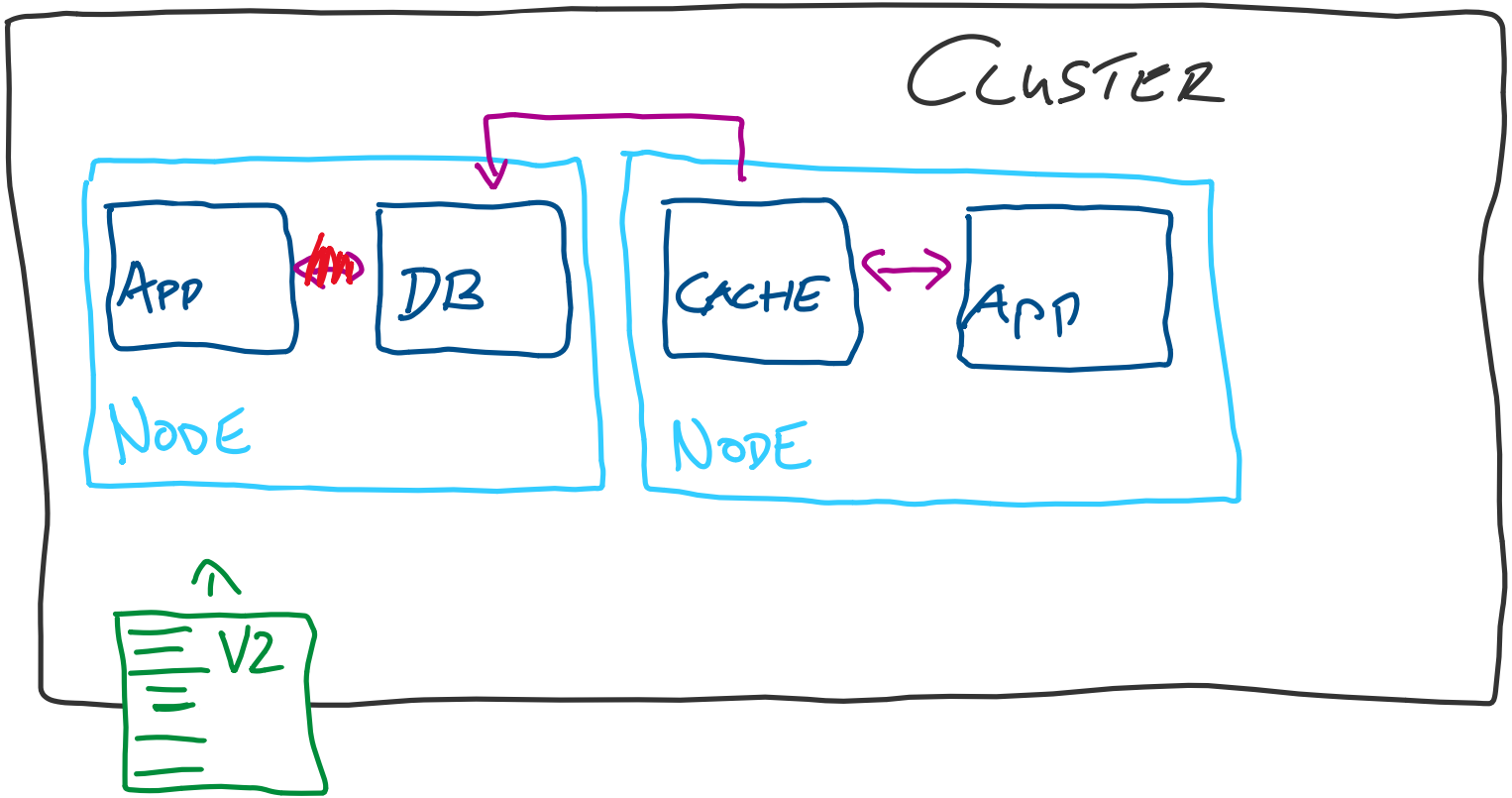
Container-optimised Linux?



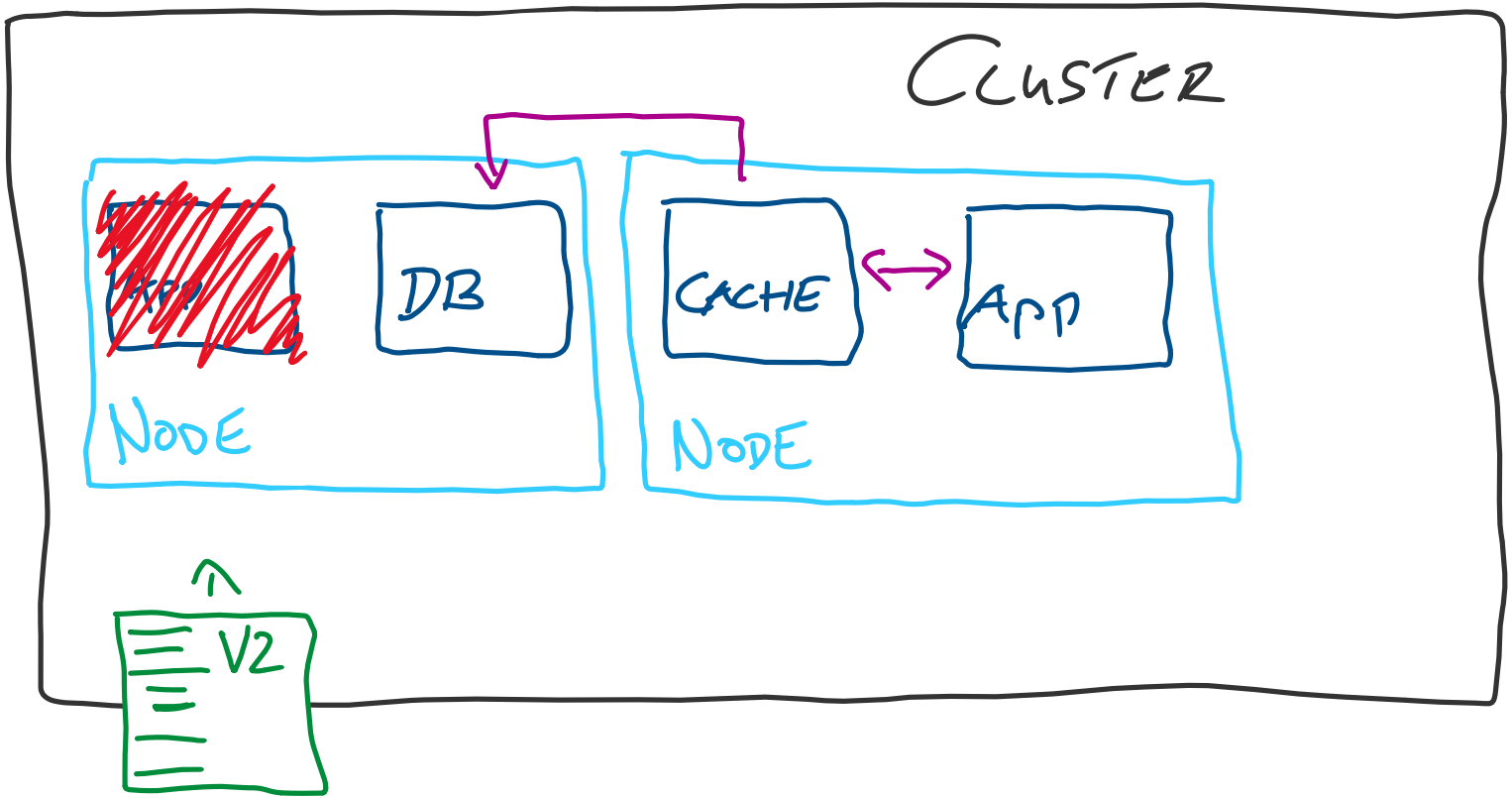
Container-optimised Linux?



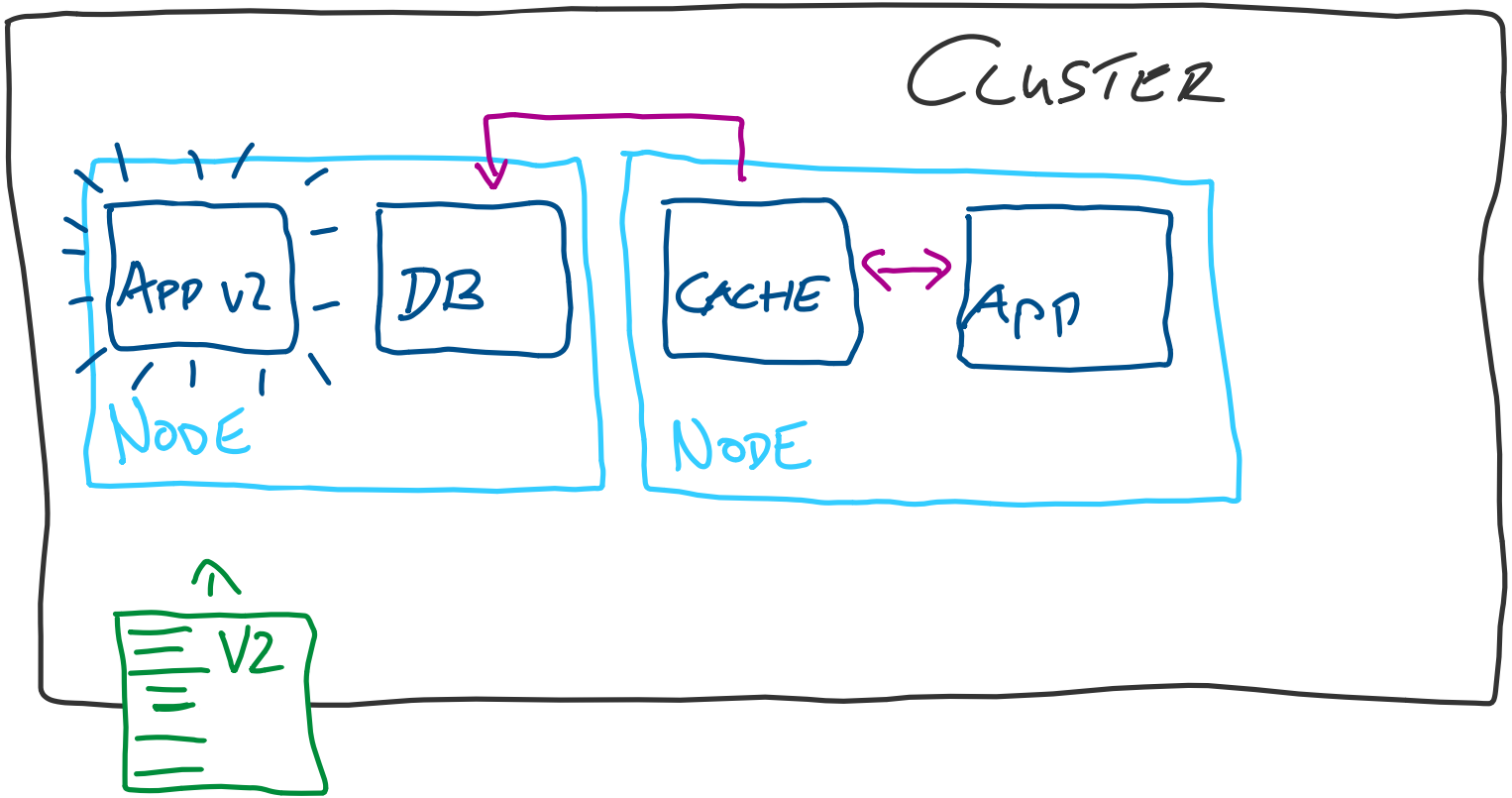
Container-optimised Linux?



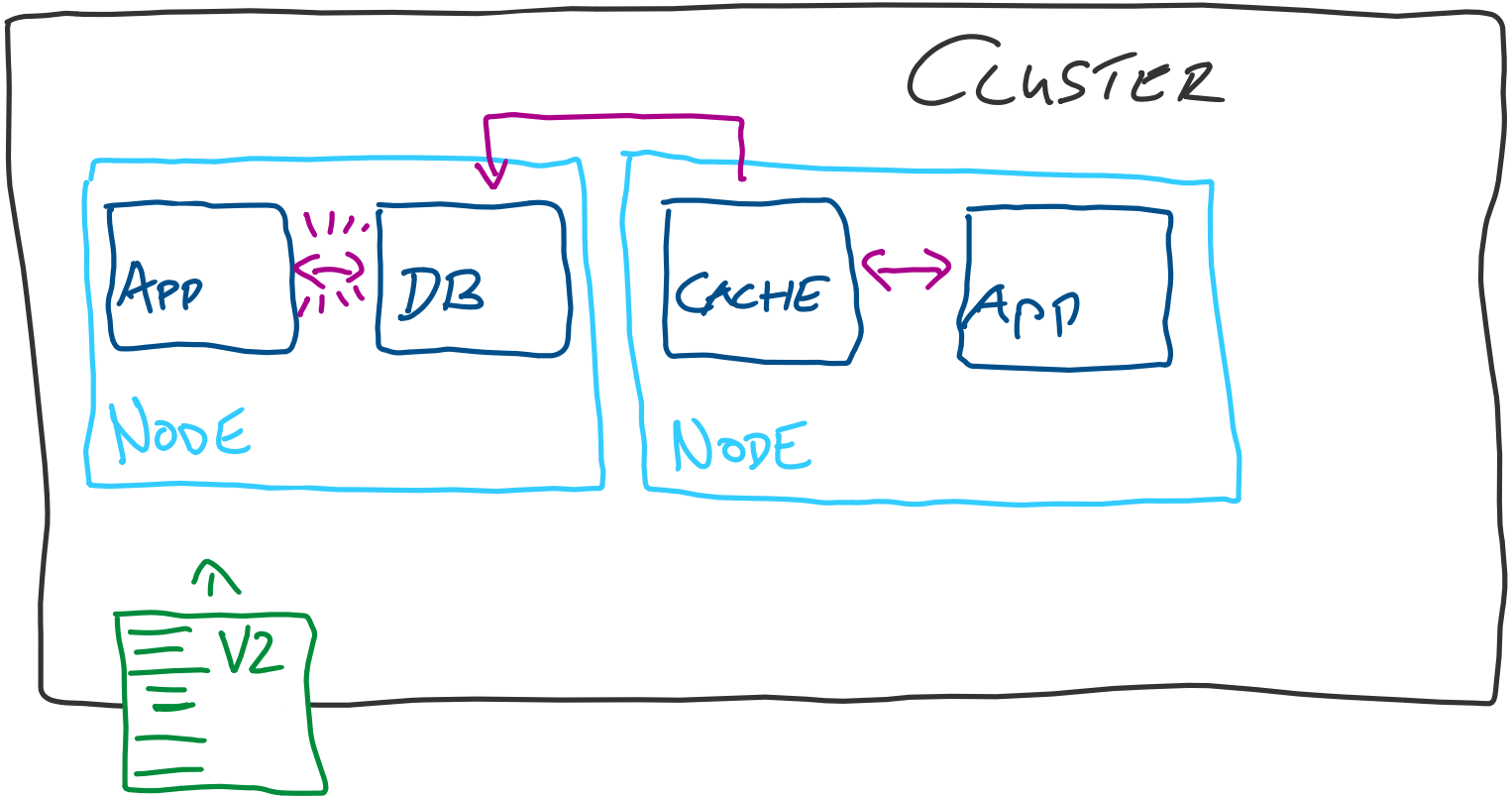
Container-optimised Linux?



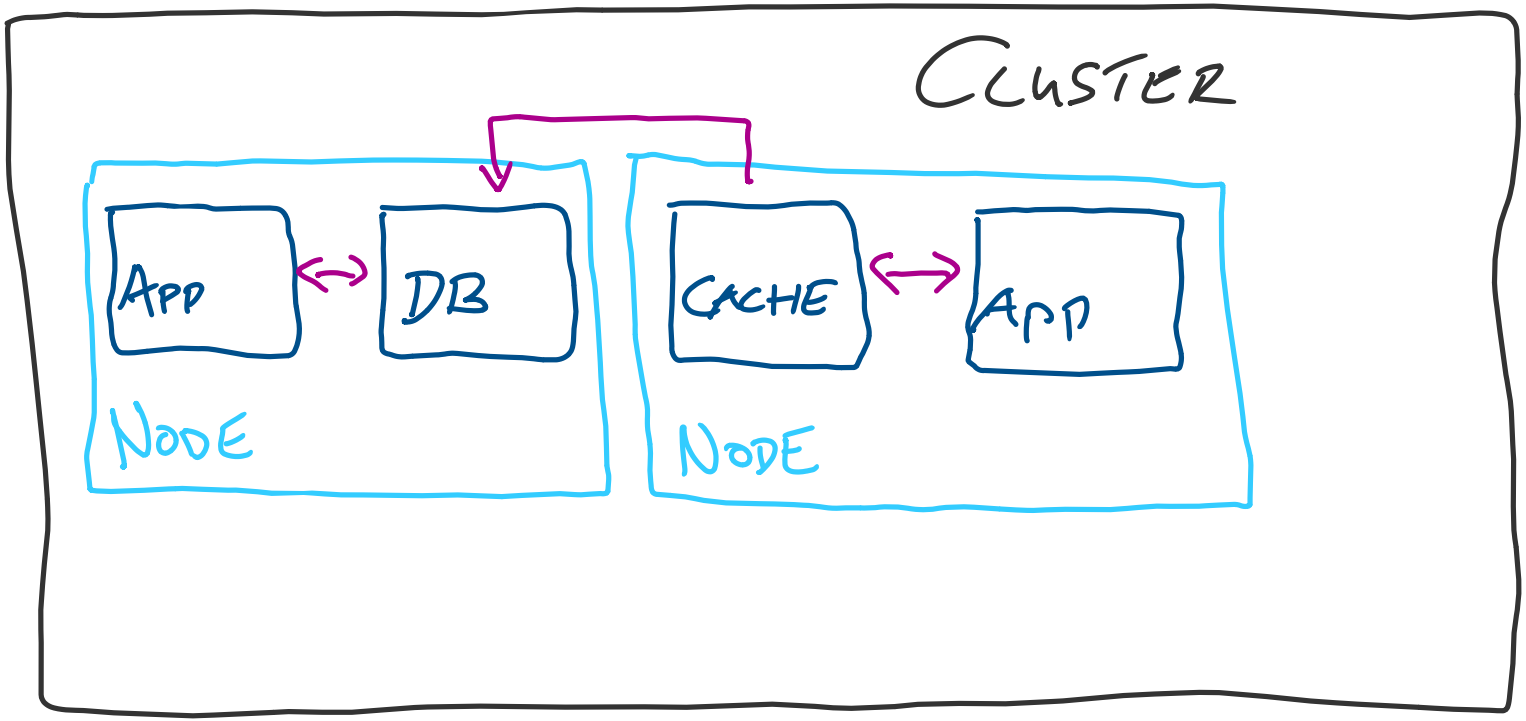
Container-optimised Linux?



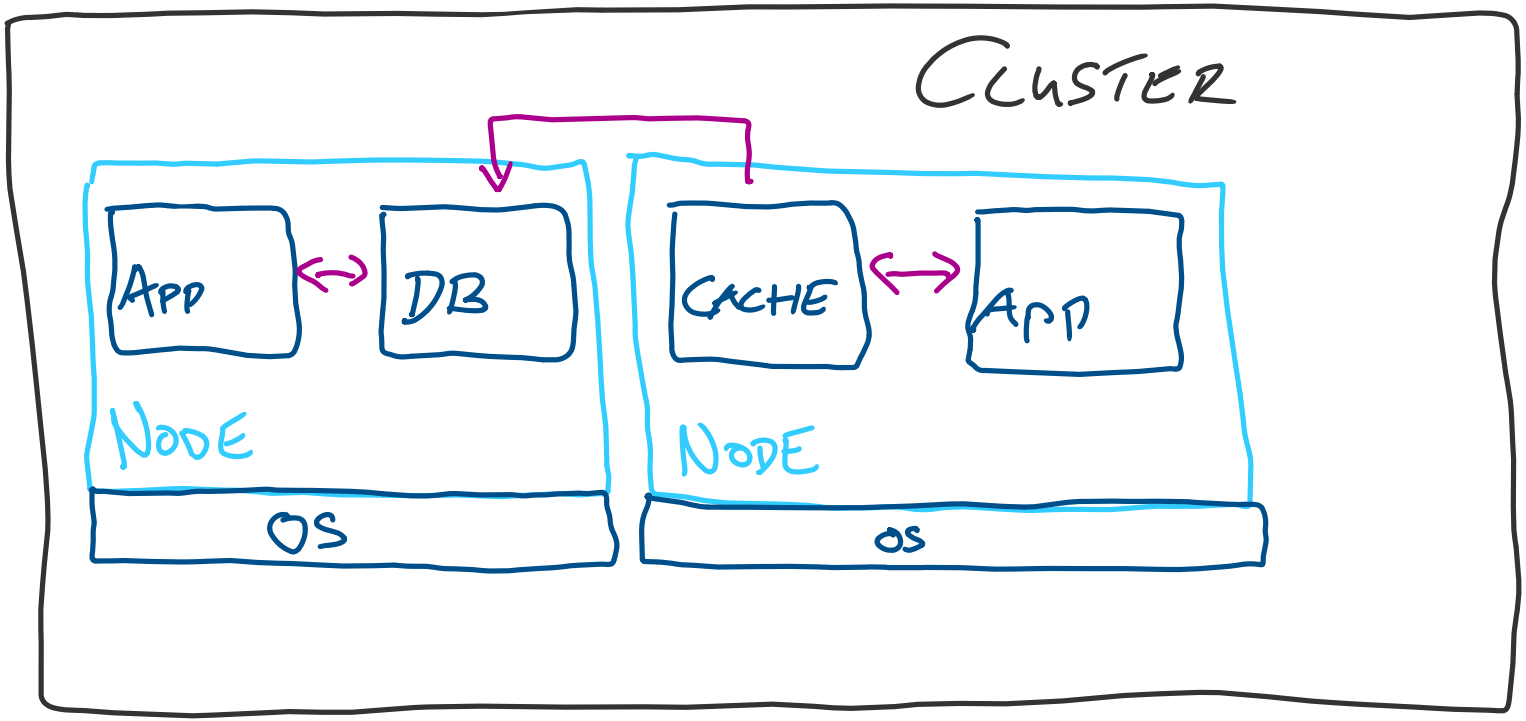
Container-optimised Linux?



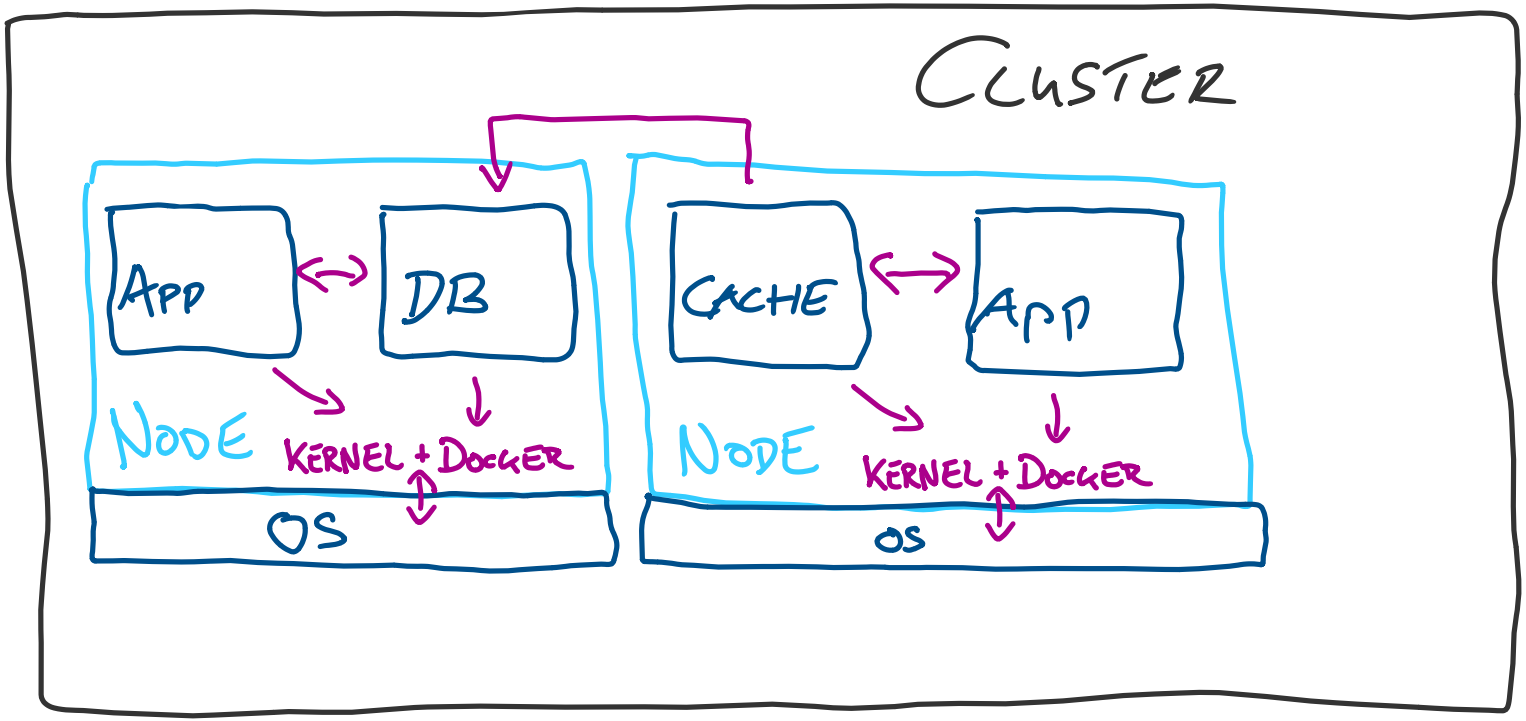
Container-optimised Linux?



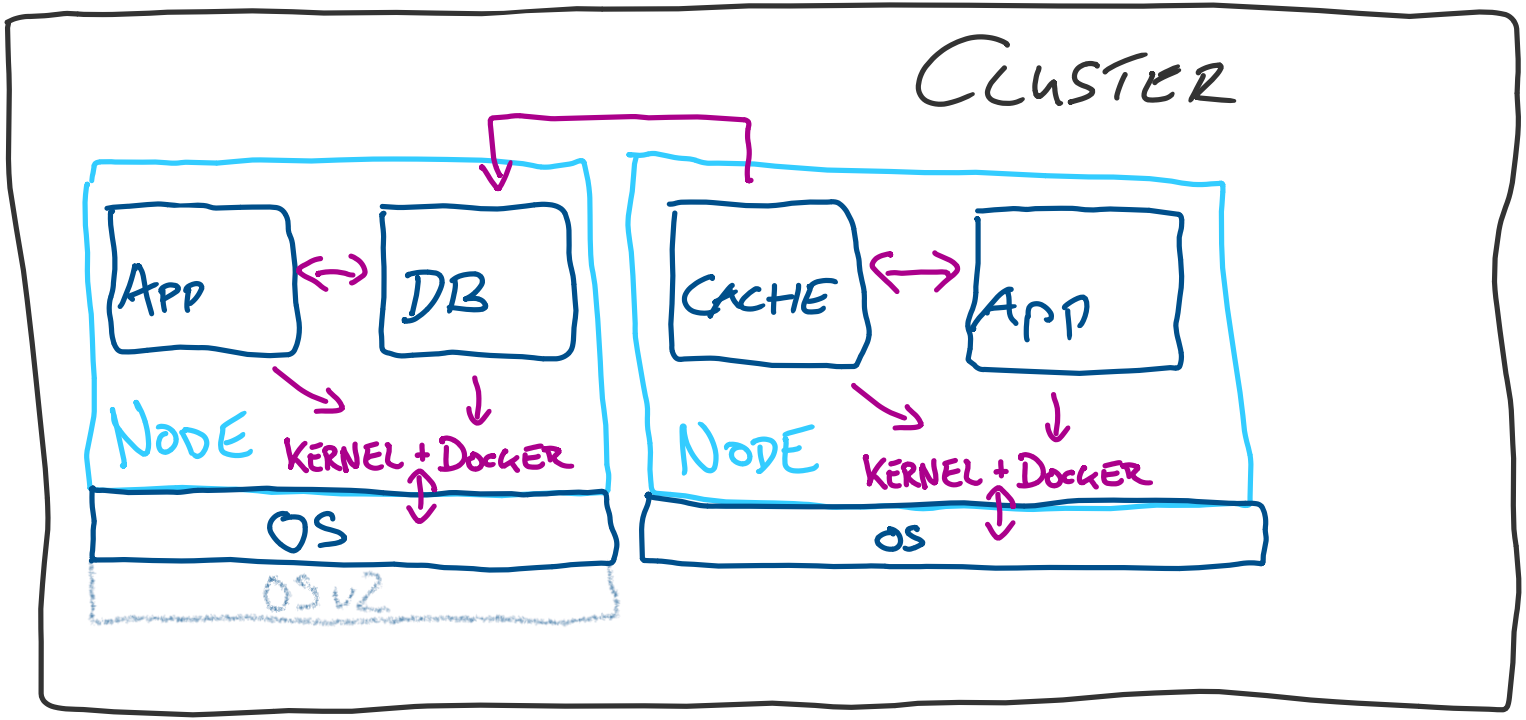
Container-optimised Linux?



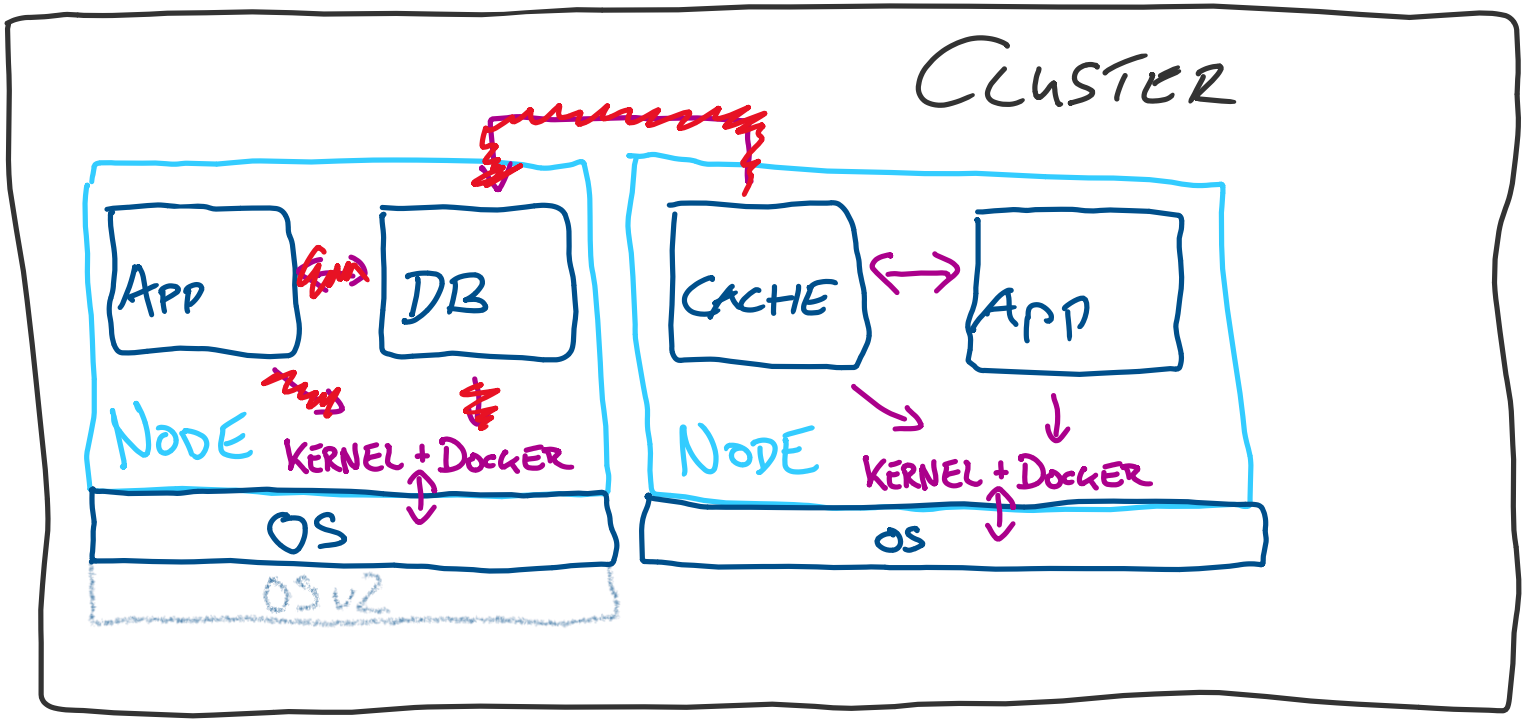
Container-optimised Linux?



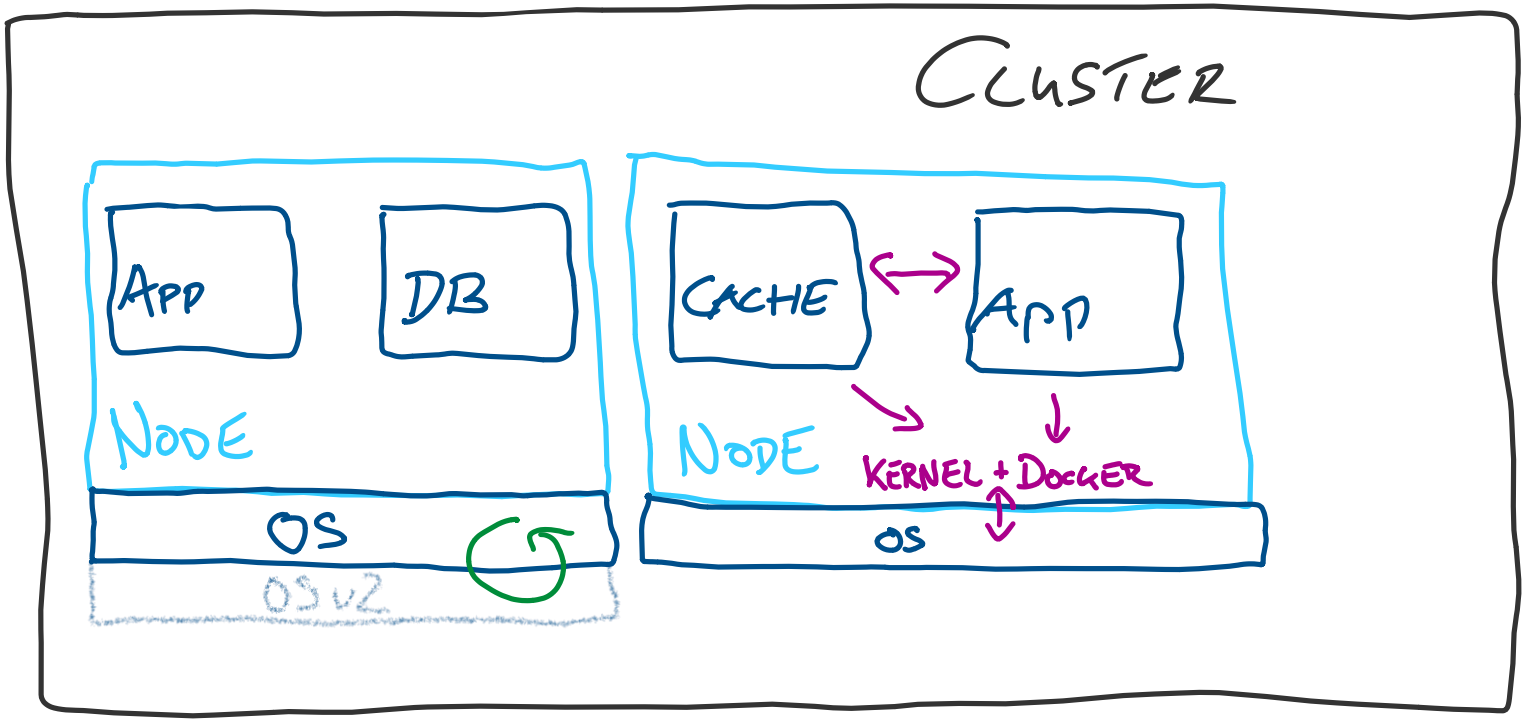
Container-optimised Linux?



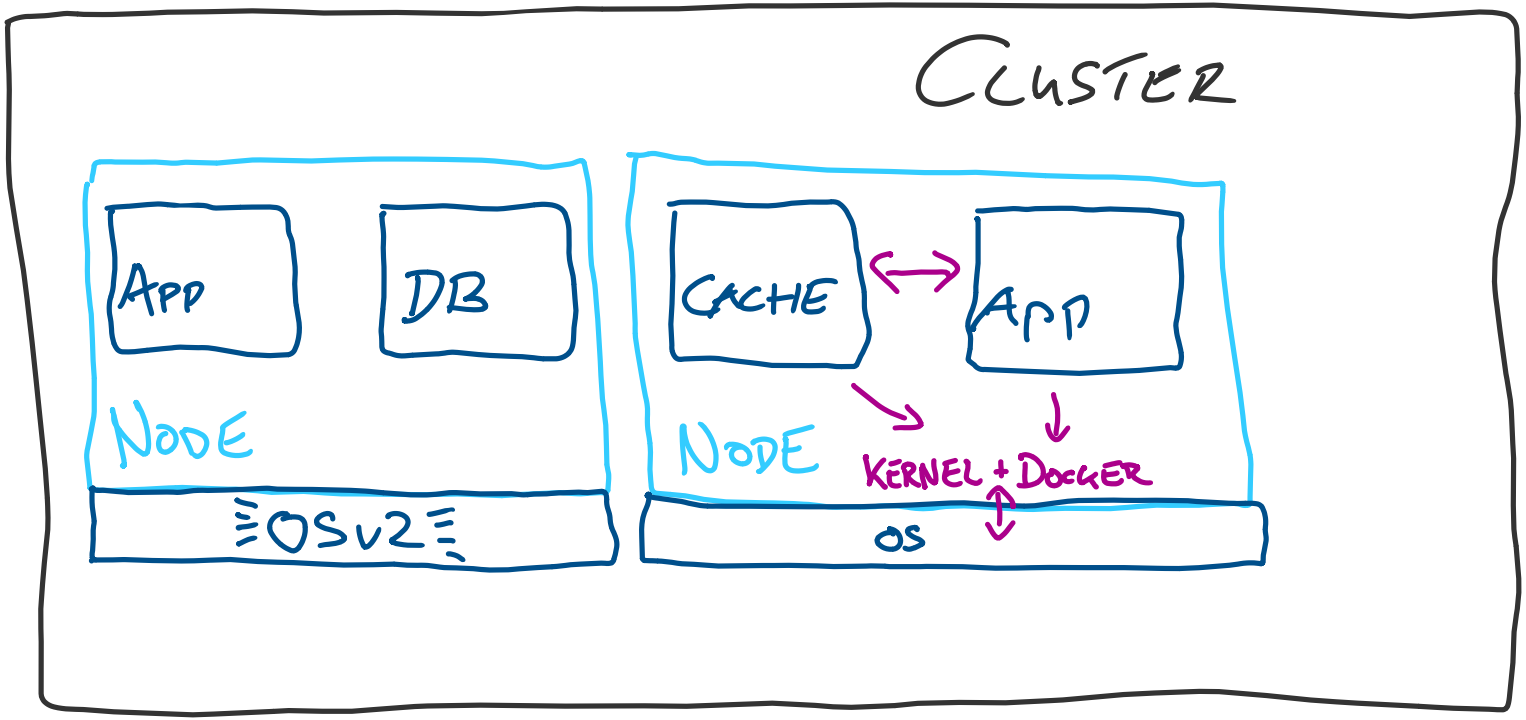
Container-optimised Linux?



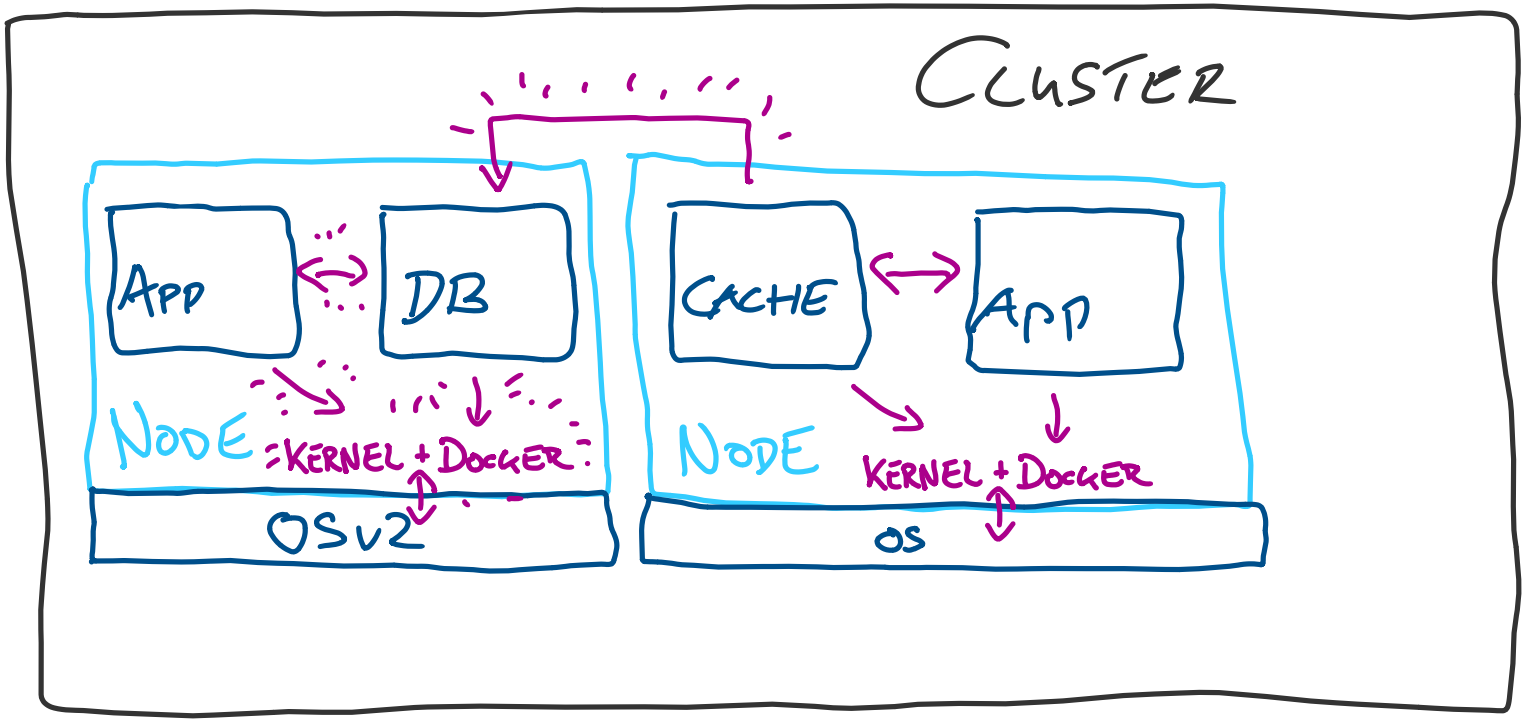
Container-optimised Linux?



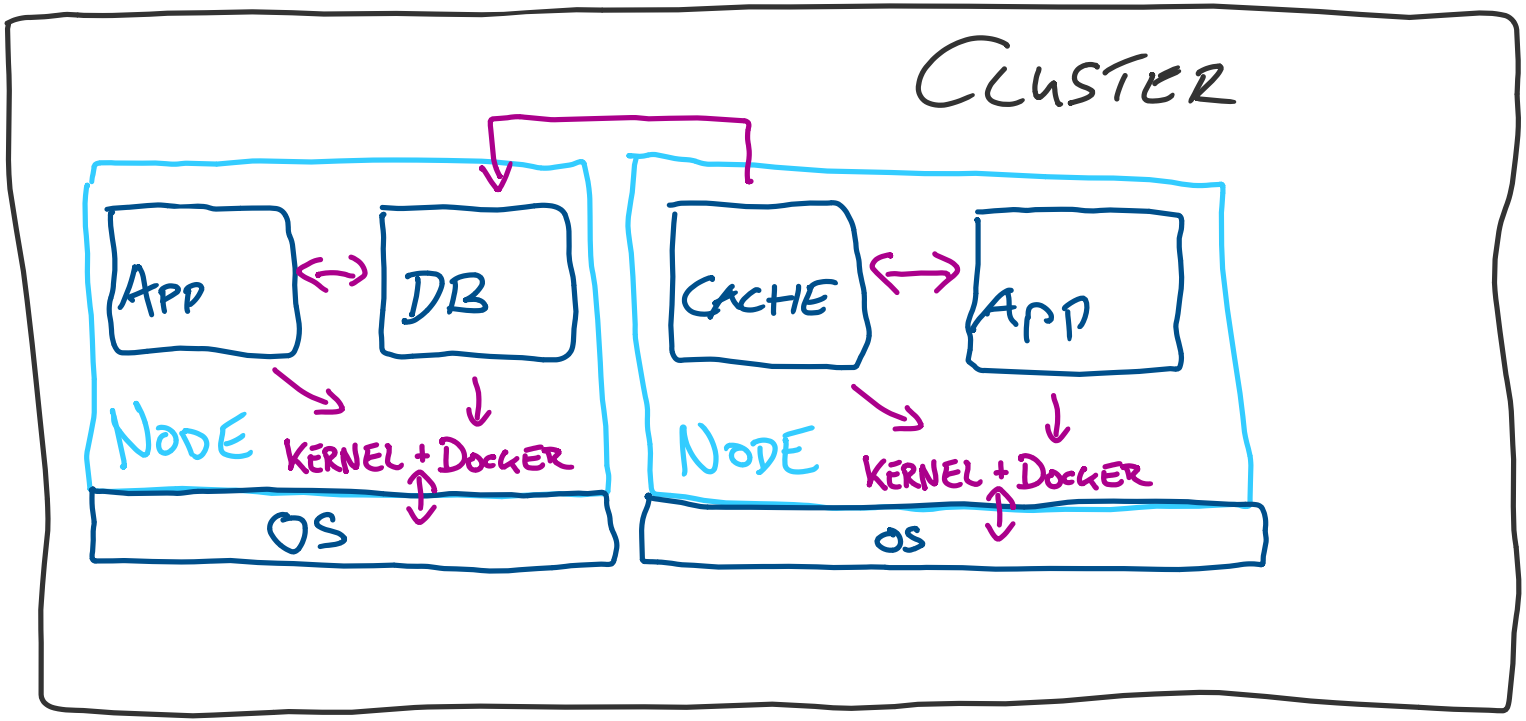
Container-optimised Linux?



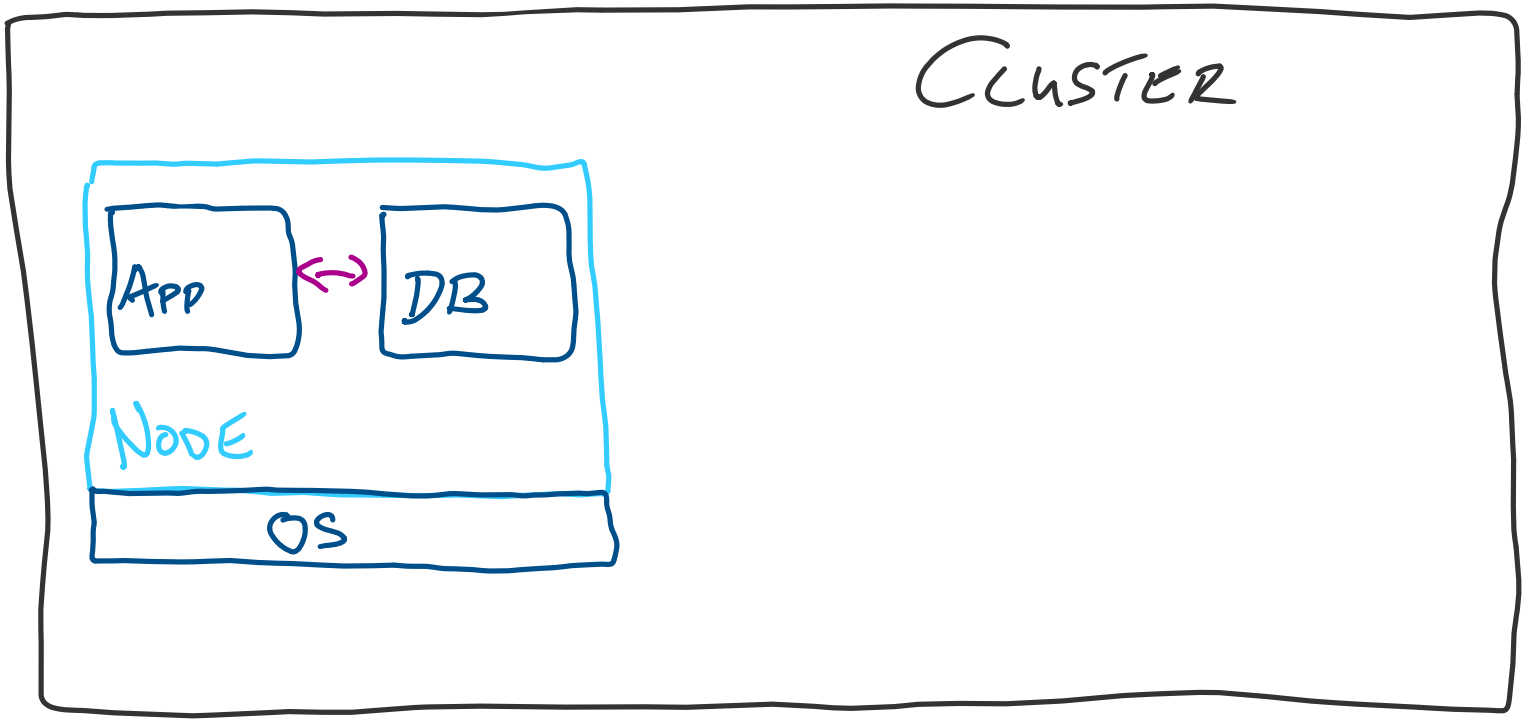
Container-optimised Linux?



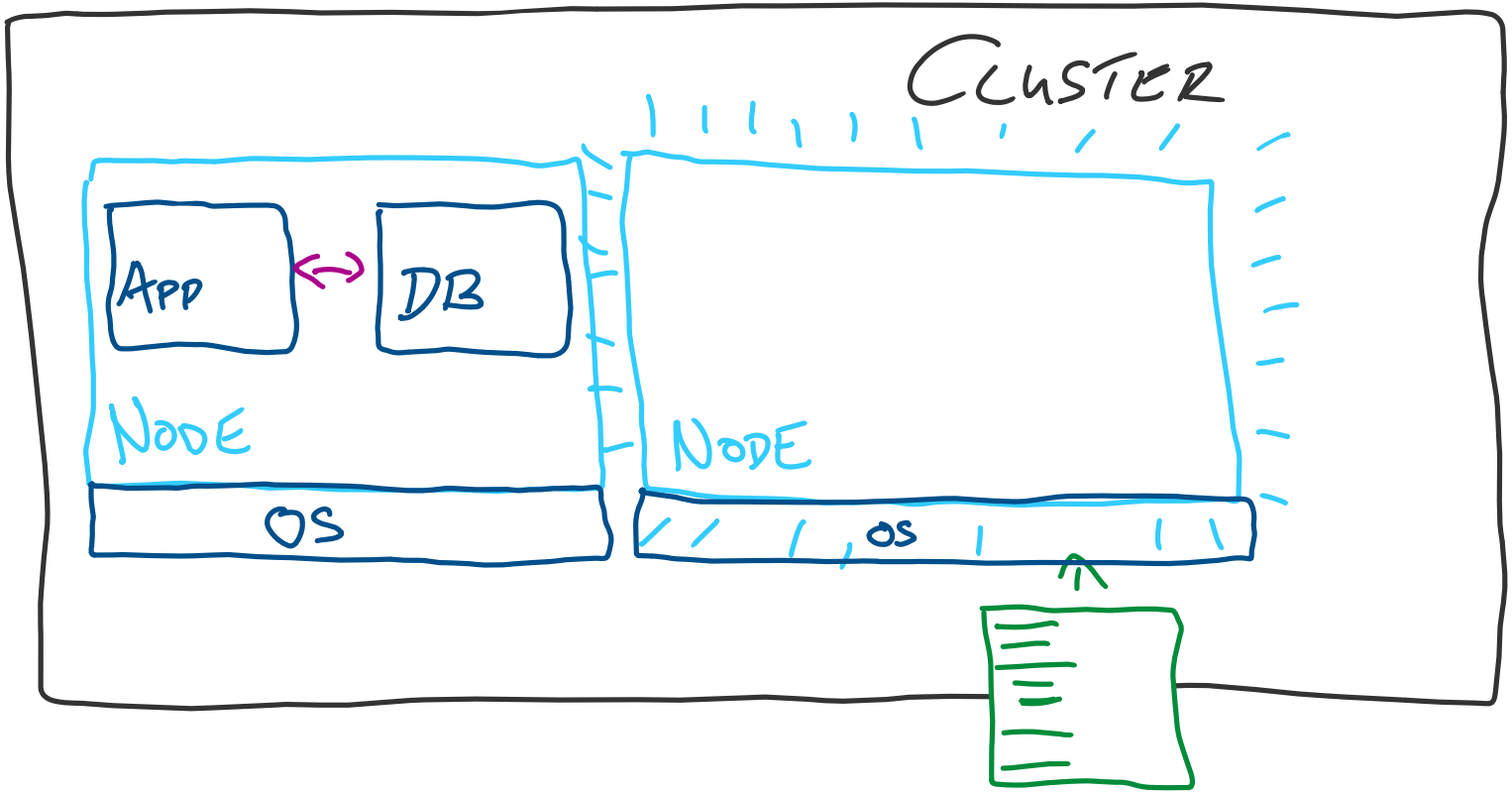
Container-optimised Linux?



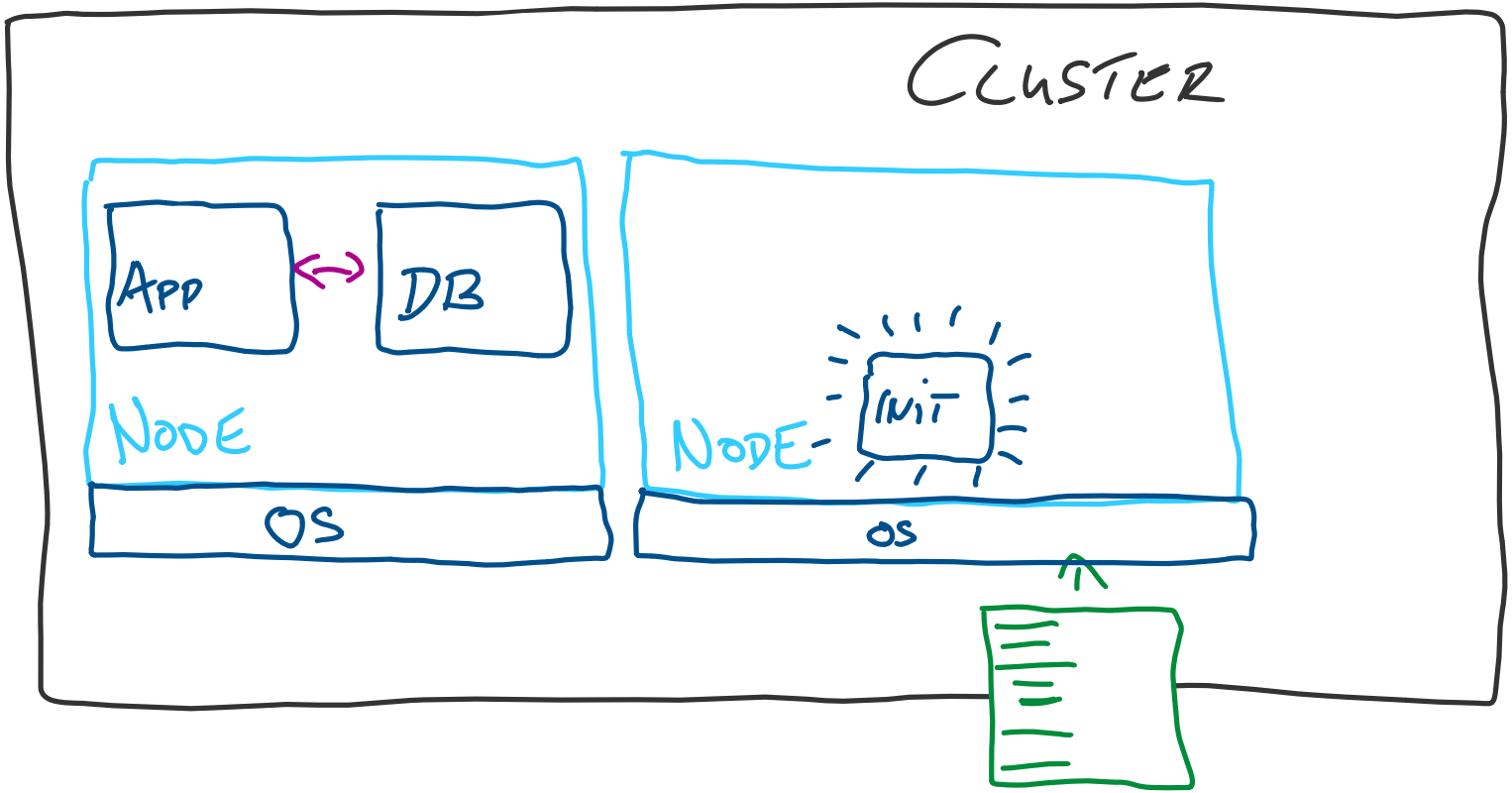
Container-optimised Linux?



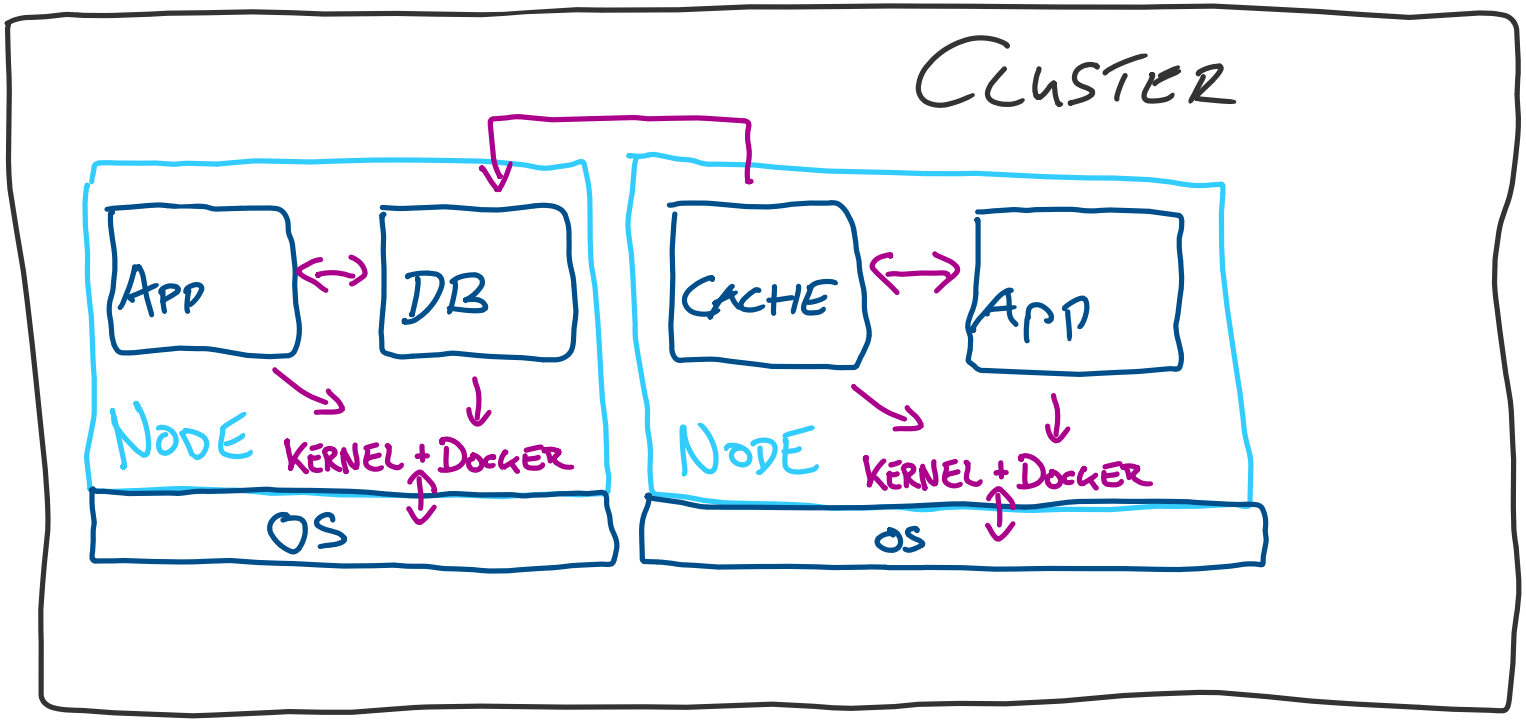
Container-optimised Linux?



Container-optimised Linux?



Container-optimised Linux?



Container-optimised Linux?

Isolated, no shared dependencies

Well-defined, limited interface to OS + "plumbing"

Declarative, automatable provisioning

Configure, apply at provisioning time. No dynamics.

Automated updates, controlled roll-out

Stage, switch, activate (or roll back)

==> Your OS as a stateless app.

Isolation

No inter-dependencies between OS and Apps

No OS <-> App shared libs. Just the Kernel ABI.

Clear and concise API/contract OS <--> Apps

No service dependencies. Just Docker/Containerd.

Updates and roll-backs with little to no side effects

Well-testable and canary friendly.

Provisioning automation

Declarative Configuration with Ignition

Applied once, at provisioning time.

Fully automatable

Thorough set of options, GO bindings, etc.

Easy to integrate with provisioning orchestration

Terraform provider, CAPI, etc.

```
passwd:
  users:
    - name: caddy
      no_create_home: true
      groups: [ docker ]

storage:
  files:
    - path: /srv/www/html/index.html
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        inline: |
          <html><body align="center">
          <h1>Hello KCD!</h1>
          <img src="kcd.png" alt="KCD logo" />
          </body></html>
    - path: /srv/www/html/kcd.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: kcd.png

systemd:
  units:
    - name: kcd-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=KCD example static web server
        After=docker.service
        Requires=docker.service
        [Service]
        User=caddy
        TimeoutStartSec=0
        ExecStartPre=-/usr/bin/docker rm --force caddy
        ExecStart=docker run -i -p 80:80 \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop nginx1
        Restart=always
        RestartSec=5s
        [Install]
        WantedBy=multi-user.target
```

```
passwd:
  users:
    - name: caddy
      no_create_home: true
      groups: [ docker ]
```


```
storage:
  files:
    - path: /srv/www/html/index.html
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        inline: |
          <html><body align="center">
          <h1>Hello KCD!</h1>
          <img src="kcd.png" alt="KCD logo" />
          </body></html>
    - path: /srv/www/html/kcd.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: kcd.png
```

```
systemd:
  units:
    - name: kcd-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=KCD example static web server
        After=docker.service
        Requires=docker.service
        [Service]
        User=caddy
        TimeoutStartSec=0
        ExecStartPre=-/usr/bin/docker rm --force caddy
        ExecStart=docker run -i -p 80:80 \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop nginx1
        Restart=always
        RestartSec=5s
        [Install]
        WantedBy=multi-user.target
```

```
passwd:
  users:
    - name: caddy
      no_create_home: true
      groups: [ docker ]

storage:
  files:
    - path: /srv/www/html/index.html
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        inline: |
          <html><body align="center">
          <h1>Hello KCD!</h1>
          <img src="kcd.png" alt="KCD logo" />
          </body></html>
    - path: /srv/www/html/kcd.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: kcd.png
```

```
systemd:
  units:
    - name: kcd-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=KCD example static web server
        After=docker.service
        Requires=docker.service
        [Service]
        User=caddy
        TimeoutStartSec=0
        ExecStartPre=-/usr/bin/docker rm --force caddy
        ExecStart=docker run -i -p 80:80 \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop nginx1
        Restart=always
        RestartSec=5s
        [Install]
        WantedBy=multi-user.target
```


```
passwd:
  users:
    - name: caddy
      no_create_home: true
      groups: [ docker ]

storage:
  files:
    - path: /srv/www/html/index.html
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        inline: |
          <html><body align="center">
          <h1>Hello KCD!</h1>
          <img src="kcd.png" alt="KCD logo" />
          </body></html>
    - path: /srv/www/html/kcd.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: kcd.png

systemd:
  units:
    - name: kcd-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=KCD example static web server
        After=docker.service
        Requires=docker.service
        [Service]
        User=caddy
        TimeoutStartSec=0
        ExecStartPre=-/usr/bin/docker rm --force caddy
        ExecStart=docker run -i -p 80:80 \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop nginx1
        Restart=always
        RestartSec=5s
        [Install]
        WantedBy=multi-user.target
```

```
passwd:
  users:
    - name: caddy
      no_create_home: true
      groups: [ docker ]

storage:
  files:
    - path: /srv/www/html/index.html
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        inline: |
          <html><body align="center">
          <h1>Hello KCD!</h1>
          <img src="kcd.png" alt="KCD logo" />
          </body></html>
    - path: /srv/www/html/kcd.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: kcd.png

systemd:
  units:
    - name: kcd-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=KCD example static web server
        After=docker.service
        Requires=docker.service
        [Service]
        User=caddy
        TimeoutStartSec=0
        ExecStartPre=-/usr/bin/docker rm --force caddy
        ExecStart=docker run -i -p 80:80 \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop nginx1
        Restart=always
        RestartSec=5s
        [Install]
        WantedBy=multi-user.target
```

```
passwd:
  users:
    - name: caddy
      no_create_home: true
      groups: [ docker ]

storage:
  files:
    - path: /srv/www/html/index.html
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        inline: |
          <html><body align="center">
          <h1>Hello KCD!</h1>
          <img src="kcd.png" alt="KCD logo" />
          </body></html>
    - path: /srv/www/html/kcd.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: kcd.png

systemd:
  units:
    - name: kcd-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=KCD example static web server
        After=docker.service
        Requires=docker.service
        [Service]
        User=caddy
        TimeoutStartSec=0
        ExecStartPre=-/usr/bin/docker rm --force caddy
        ExecStart=docker run -i -p 80:80 \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop nginx1
        Restart=always
        RestartSec=5s
        [Install]
        WantedBy=multi-user.target
```

```
passwd:
  users:
    - name: caddy
      no_create_home: true
      groups: [ docker ]

storage:
  files:
    - path: /srv/www/html/index.html
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        inline: |
          <html><body align="center">
          <h1>Hello KCD!</h1>
          <img src="kcd.png" alt="KCD logo" />
          </body></html>
    - path: /srv/www/html/kcd.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: kcd.png

systemd:
  units:
    - name: kcd-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=KCD example static web server
        After=docker.service
        Requires=docker.service
        [Service]
        User=caddy
        TimeoutStartSec=0
        ExecStartPre=-/usr/bin/docker rm --force caddy
        ExecStart=docker run -i -p 80:80 \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop nginx1
        Restart=always
        RestartSec=5s
        [Install]
        WantedBy=multi-user.target
```

Image-based OS

Stateless installations

OS is replaceable without side effects

Immutable (read-only) OS partition

No changes to OS binaries, accidentally or otherwise.

Verifiable / attestable

OS never changes, build / release signature remains valid

Updates

- Atomic updates
  - *Staged on inactive OS partition in the background*
  - *Applied at reboot*
- Automated, configurable roll-back
  - *Define required services and timeouts*
- Controlled roll-out
  - *Cluster-wide reboot coordination*
  - *Roll-back support*

Updates – release stabilisation

Alpha –> Beta -> Stable (->LTS)

Alpha is for dev, Beta for canaries, and Stable

Extensive scenario testing for each release

Complex - Kubernetes workloads, CNI, etc.

All releases always pass full test suite

Beta canaries ensure Stable runs fine with your workloads

Scaling out

(Community) LTS

When you're so big that any maintenance is costly.

FOSS update server (Nebraska)

Self-hosted updates, fine-grained roll-out, fleet mgmt.

User-centric, community-driven, fully open source

Speak up, join in, participate!

Community driven

Single vendor past

Back in the day right after our friendly fork

Freed ourselves, embracing the community

Re-orientation of project stewardship

Microsoft supports the community

And publicly commits to this

MS committed to support and to foster community

https://azure.microsoft.com/en-us/blog/microsoft-acquires-kinvolk-to-accelerate-containeroptimized-innovation/

"Flatcar Container Linux has a sizeable community of users on Azure, as well as other clouds, and on-premises. We know the CoreOS community has been on a winding journey over the years—we want to assure the Flatcar community that Microsoft and the Kinvolk team will continue to collaborate with the larger Flatcar community on the evolution of Flatcar Container Linux. Microsoft is committed to Flatcar Container Linux community development and will invest in working with the Flatcar community to create a growth path forward together. We'll have our first meeting with the community within the coming weeks and invite anyone interested to attend and join the conversation."

MS committed to support and to foster community

https://azure.microsoft.com/en-us/blog/microsoft-acquires-kinvolk-to-accelerate-containeroptimized-innovation/

"Flatcar Container Linux has a sizeable community of users on Azure, as well as other clouds, and on-premises. We know the CoreOS community has been on a winding journey over the years—we want to assure the Flatcar community that Microsoft and the Kinvolk team will continue to collaborate with the larger Flatcar community on the evolution of Flatcar Container Linux. Microsoft is committed to Flatcar Container Linux community development and will invest in working with the Flatcar community to create a growth path forward together. We'll have our first meeting with the community within the coming weeks and invite anyone interested to attend and join the conversation."

Work in the open

Day-to-day interactions

Matrix, Slack – line work and releases brawl

Planning, short and long term

Office hours, release planning, project boards

Bug smashing and docs writing days

Live streamed! Next one is 8th of July.

What's going on? – Boards

Implementation

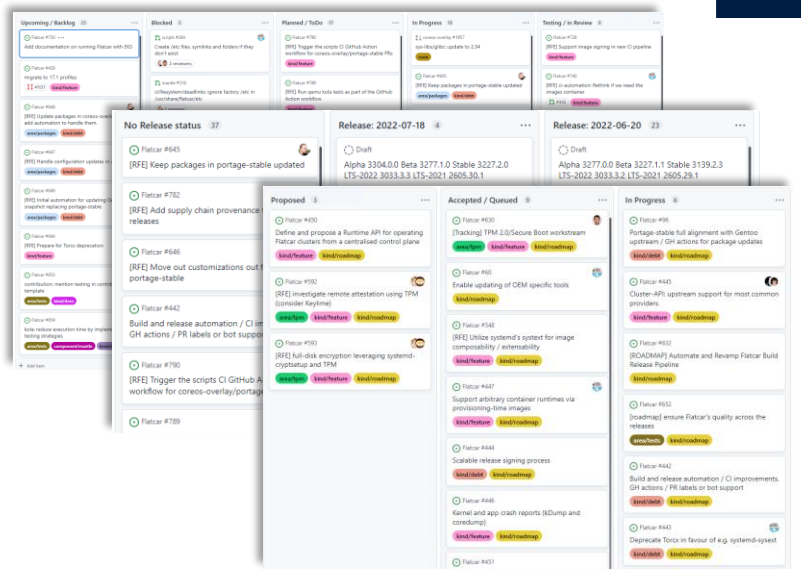
day-to-day kanban

Release

Wrap up implementations

Roadmap

Epics with subtasks



Contribute to other projects

Gentoo

package updates, stabilisation, bug fixes

Ignition, Butane

Extensions and bug fixes

Linux Kernel, Cilium, Falco

Bug fixes and interop

Contribute

Report bugs and issues, give feedback

We want your opinion

Bug smashing / doc writing days

Next bug smash is on the 8th of July, live streamed

Package updates / add tools to OS image

Become a Flatcar dev!

Stay in touch

Office hours

Every second Tuesday of the month, 5:30 CE[S]T

Matrix

Our day-to-day comms

Slack

Same

Demo time

We'll use the web server set-up shown earlier

(Almost. Update auto-reboot will be disabled.)

We will also update the OS.

Straightforward. Boring. Your Feierabend is safe.

The demo sources are available

https://github.com/flatcar-linux/flatcar-demos/tree/main/kcd-berlin-2022-06-30

Thank you

Look us up – https://www.flatcar.org
Join our office hours - https://github.com/flatcar-linux/Flatcar/#monthly-office-hours-and-release-planning
Contribute – https://github.com/flatcar-linux/Flatcar
Chat with us – https://app.element.io/#/room/#flatcar:matrix.org
– https://kubernetes.slack.com/archives/C03GQ8B5XNJ