
Zero Touch Kubernetes
Vollautomatisierte Infrastruktur mit Flatcar Container Linux
FrOSCon
Free and Open Source Software Conference
2022-08-20

Hallo!

Ich bin Thilo.

Thilo Fromm

Engineering manager, Microsoft

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

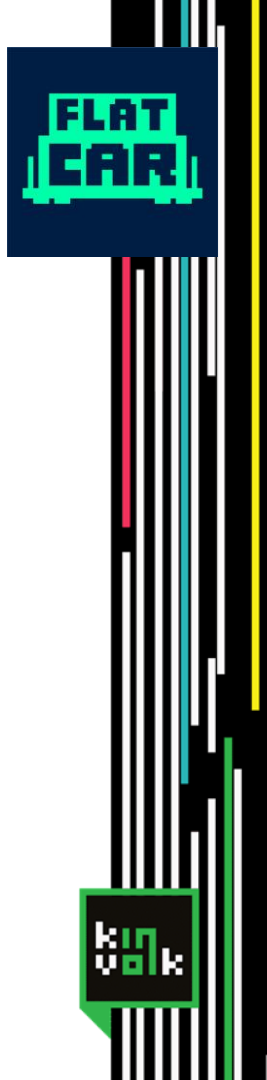
Und Ihr?

Cluster-Admins / Verteilte Applikationen?

Und Ihr?

Cluster-Admins / Verteilte Applikationen?

Container-Applikationen/Workloads?



Und Ihr?

Cluster-Admins / Verteilte Applikationen?

Container-Applikationen/Workloads?

Kubernetes?

Container-optimiertes Linux - Geschichte

Container-optimiertes Linux

CoreOS Container Linux (Okt. 2013 – Mai 2020)

(Basiert auf Chromium OS, welches auf Gentoo basiert)

Container-optimiertes Linux

CoreOS Container Linux (Okt. 2013 – Mai 2020)

(Basiert auf Chromium OS, welches auf Gentoo basiert)

Fedora CoreOS / Red Hat CoreOS (seit Juni 2018)

(basiert auf Fedora, eingeschränkt kompatibel zu CoreOS)

Container-optimiertes Linux

CoreOS Container Linux (Okt. 2013 – Mai 2020)

(Basiert auf Chromium OS, welches auf Gentoo basiert)

→ Fedora CoreOS / Red Hat CoreOS (seit Juni 2018)

(basiert auf Fedora, eingeschränkt kompatibel zu CoreOS)

→ Flatcar Container Linux (seit Nov. 2019)

(“Friendly fork”, vollständig mit CoreOS kompatibel)

Container-optimiertes Linux?

Container-optimiertes Linux?

Betriebssystem als Infrastruktur

Langweilig (nicht aufregend), “tut einfach”

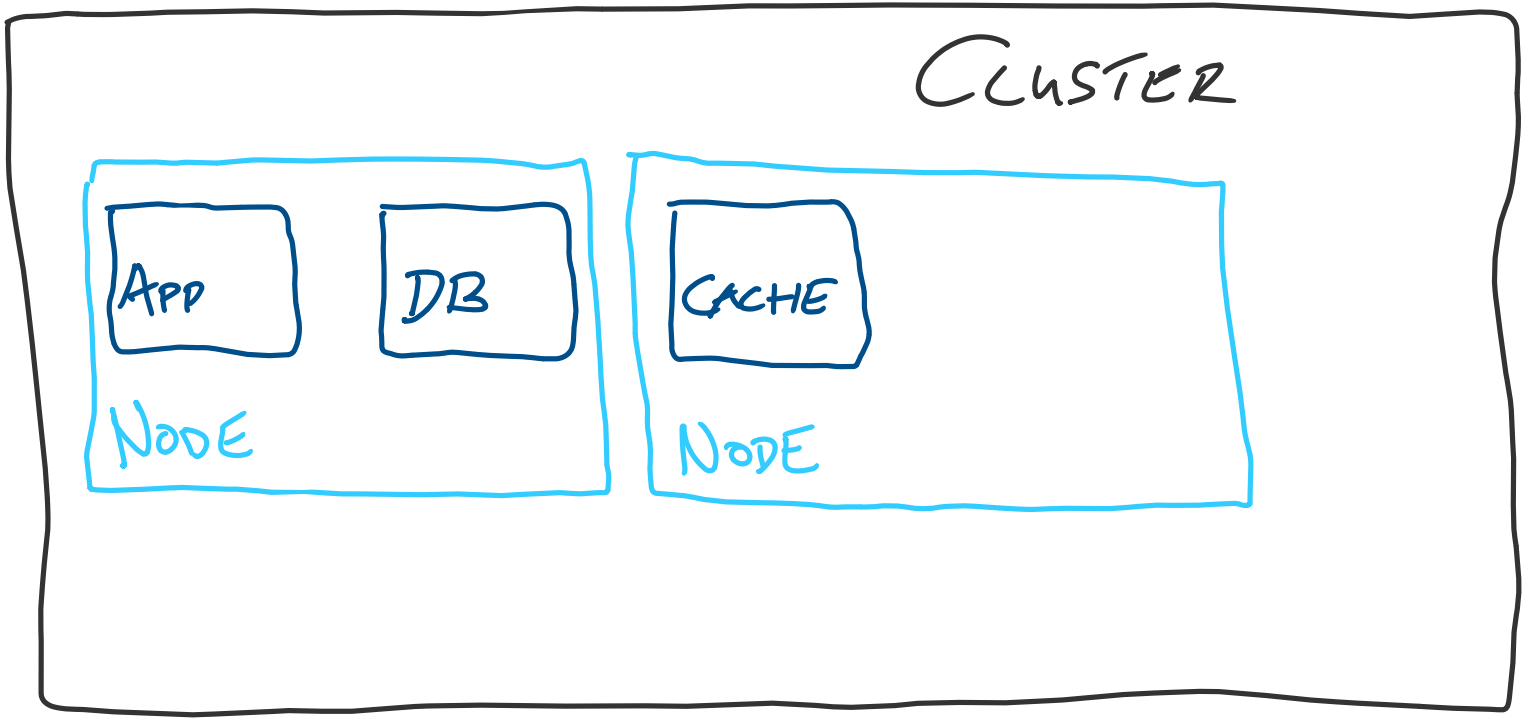
Mehr Zeit für Applikationen / Business Logic

Wenn Dein Lichtschalter ständig Aufmerksamkeit braucht,
kann das das die Freude am Heimkino trüben

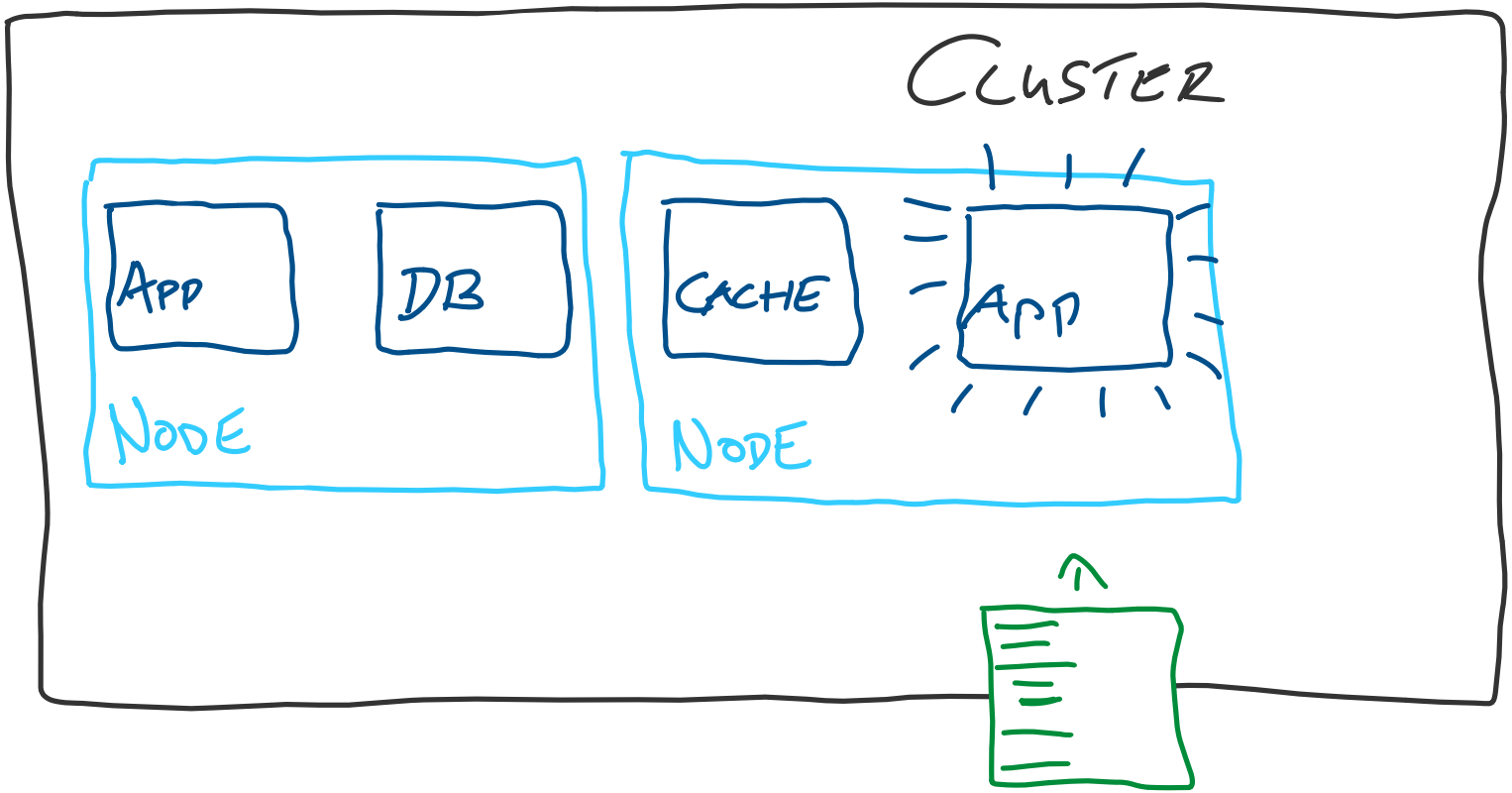
Container-Isolation – auch vom Betriebssystem

Container-Applikationen sind voneinander isoliert.
Isoliert sie das nicht auch vom Betriebssystem?

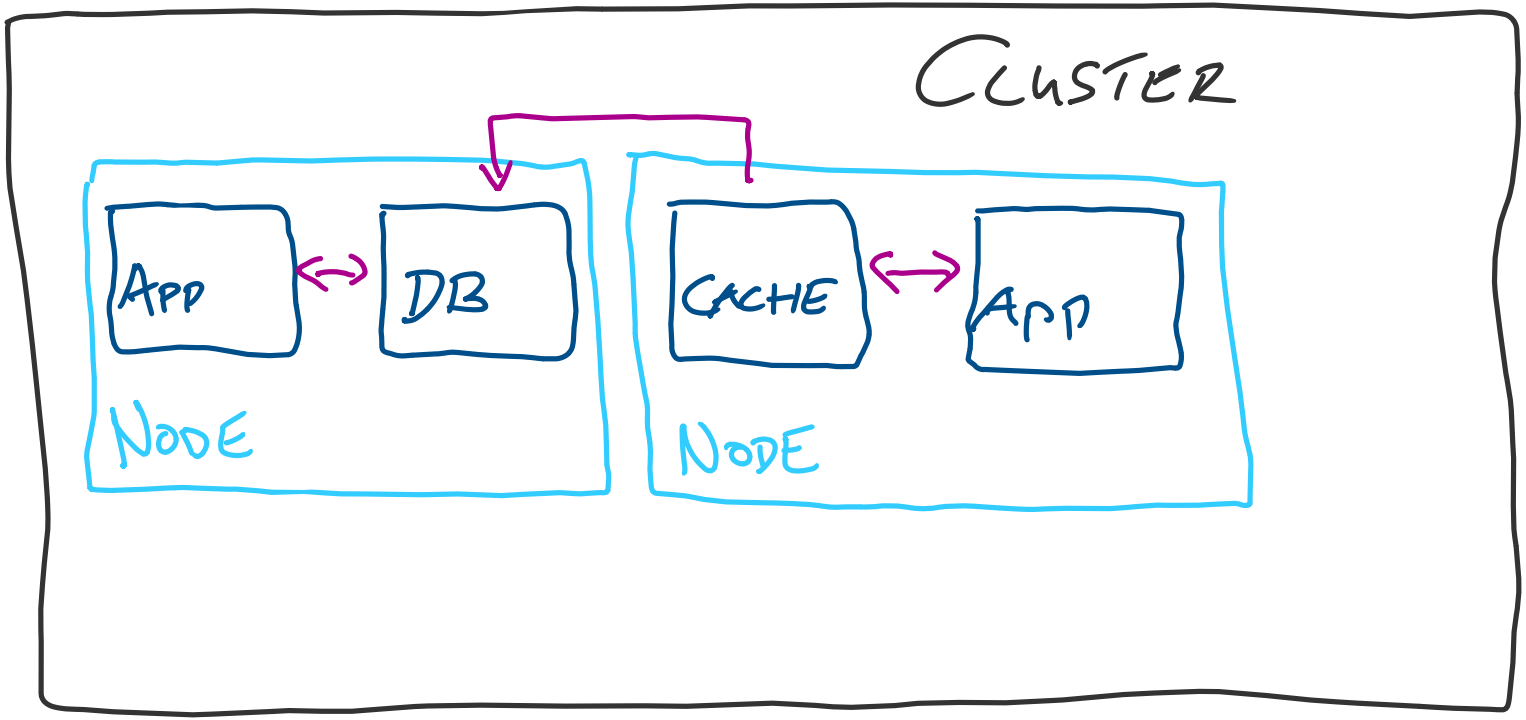
Container-optimiertes Linux?



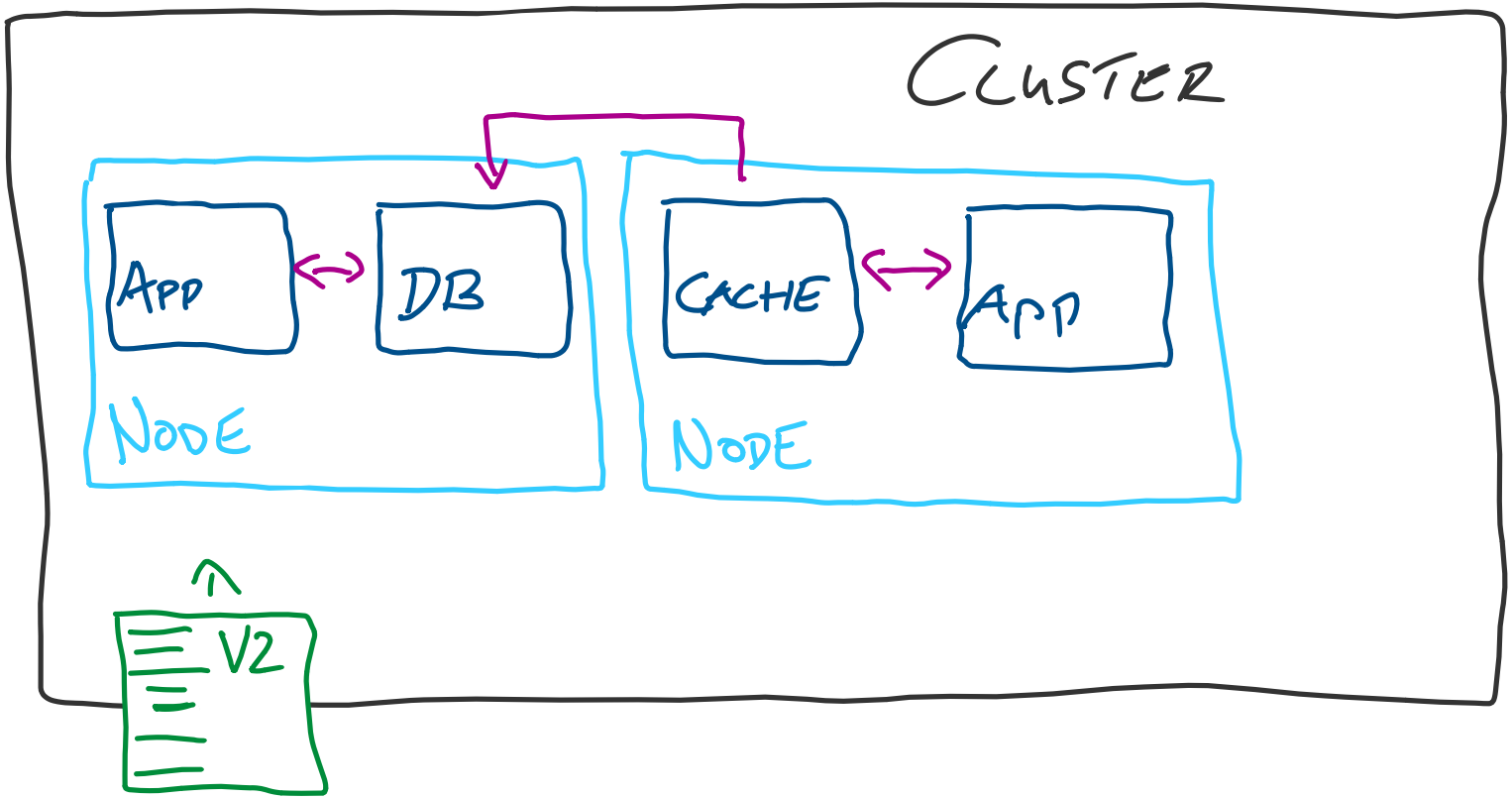
Container-optimiertes Linux?



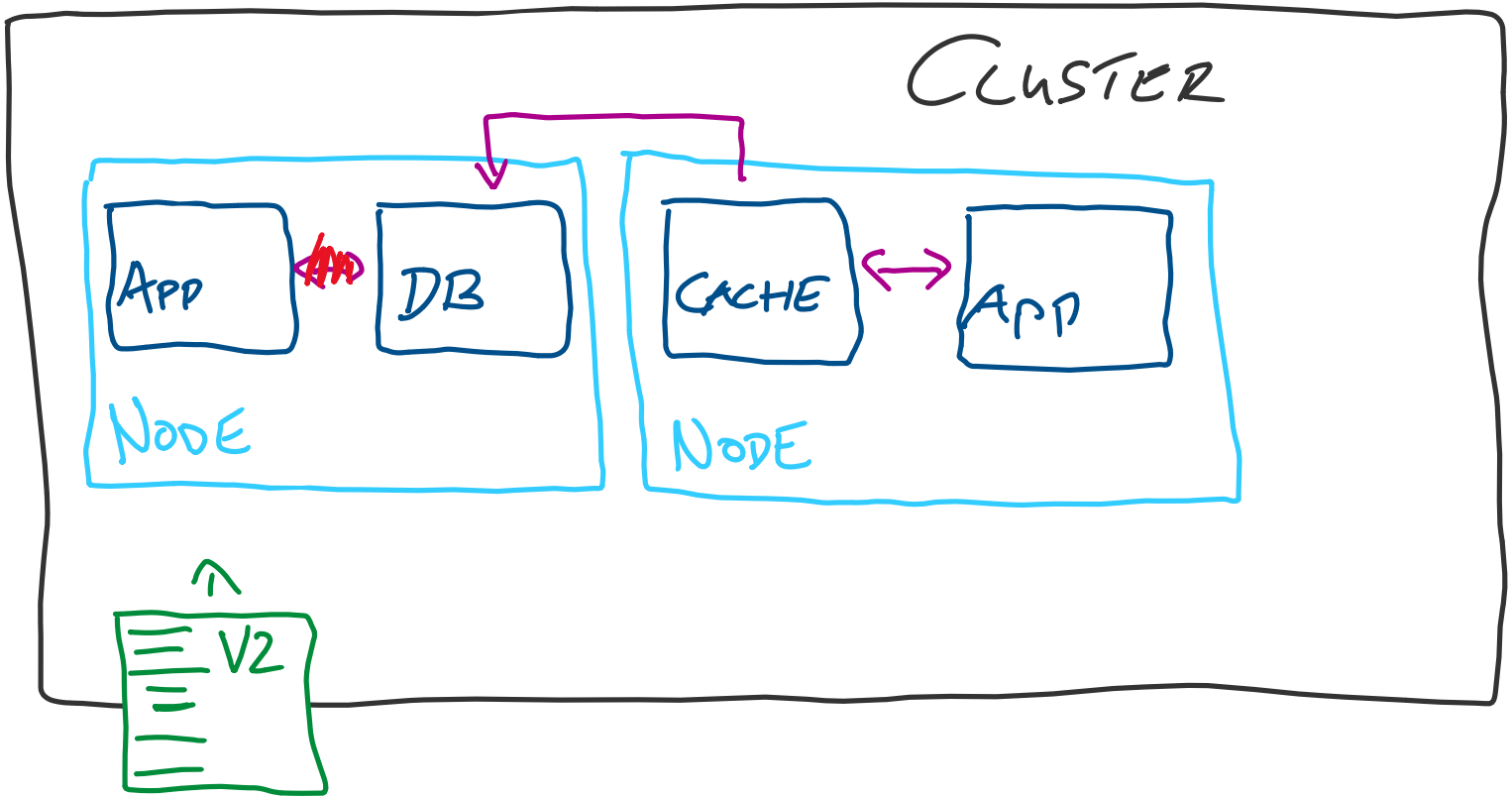
Container-optimiertes Linux?



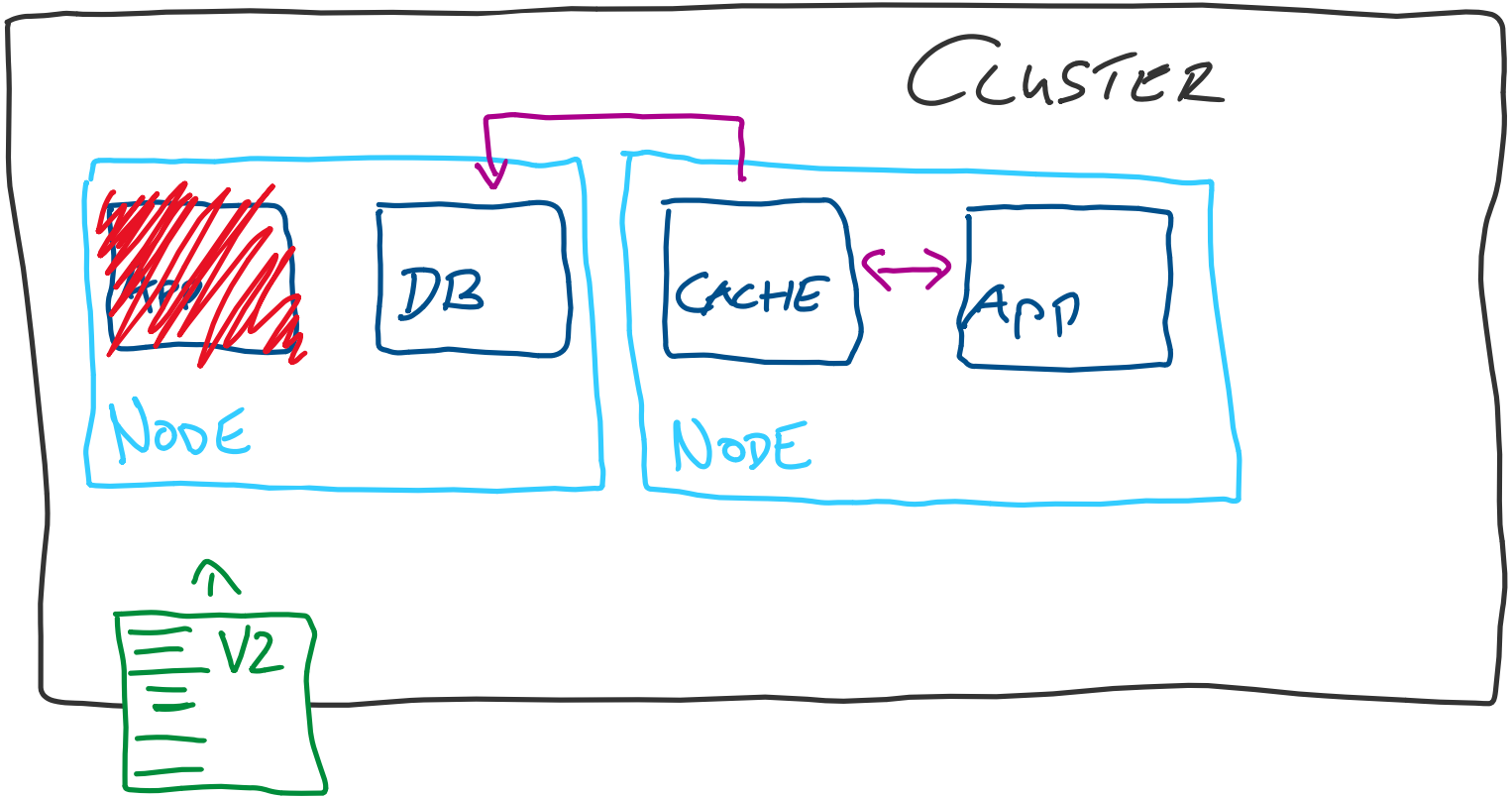
Container-optimiertes Linux?



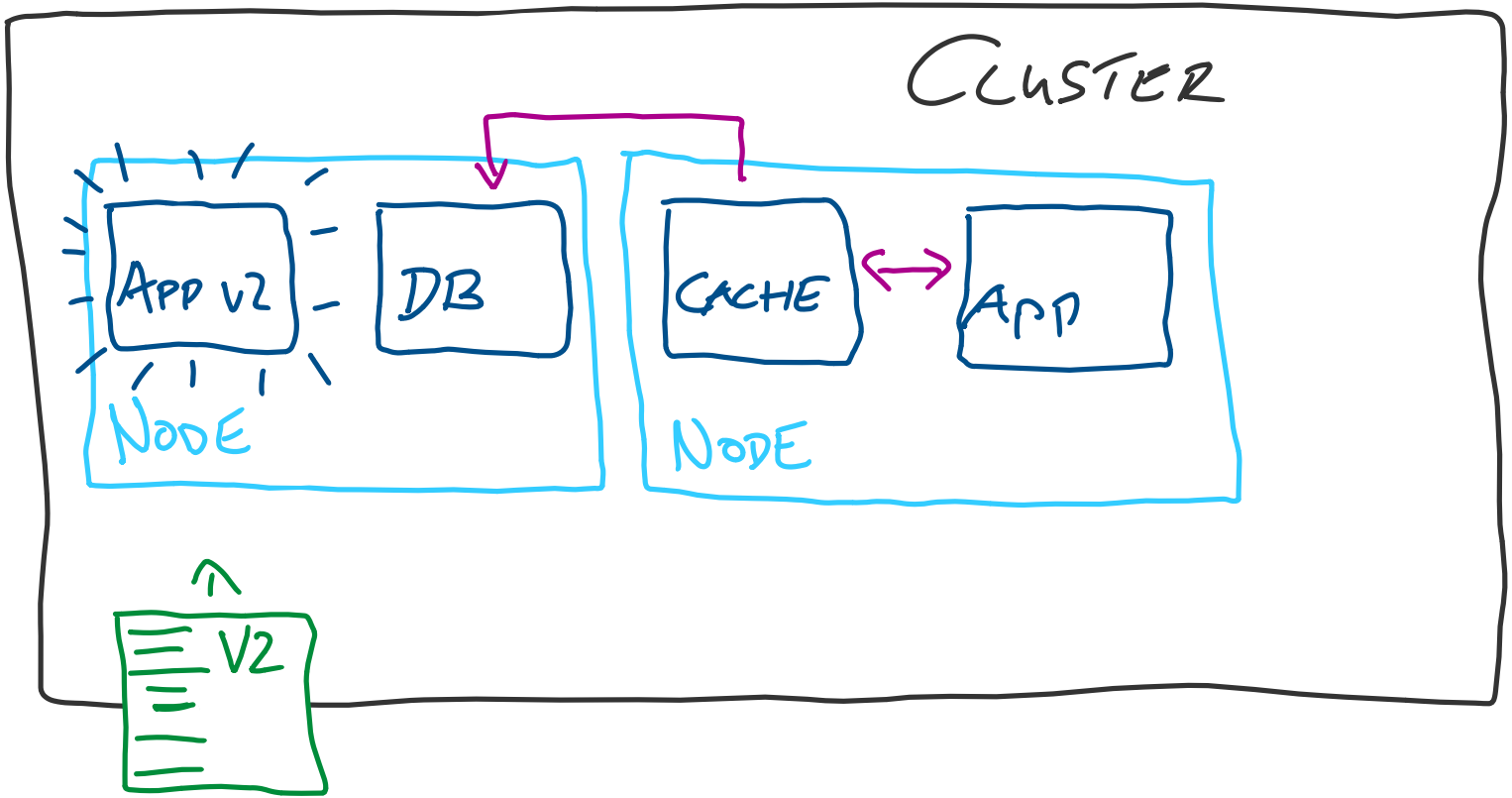
Container-optimiertes Linux?



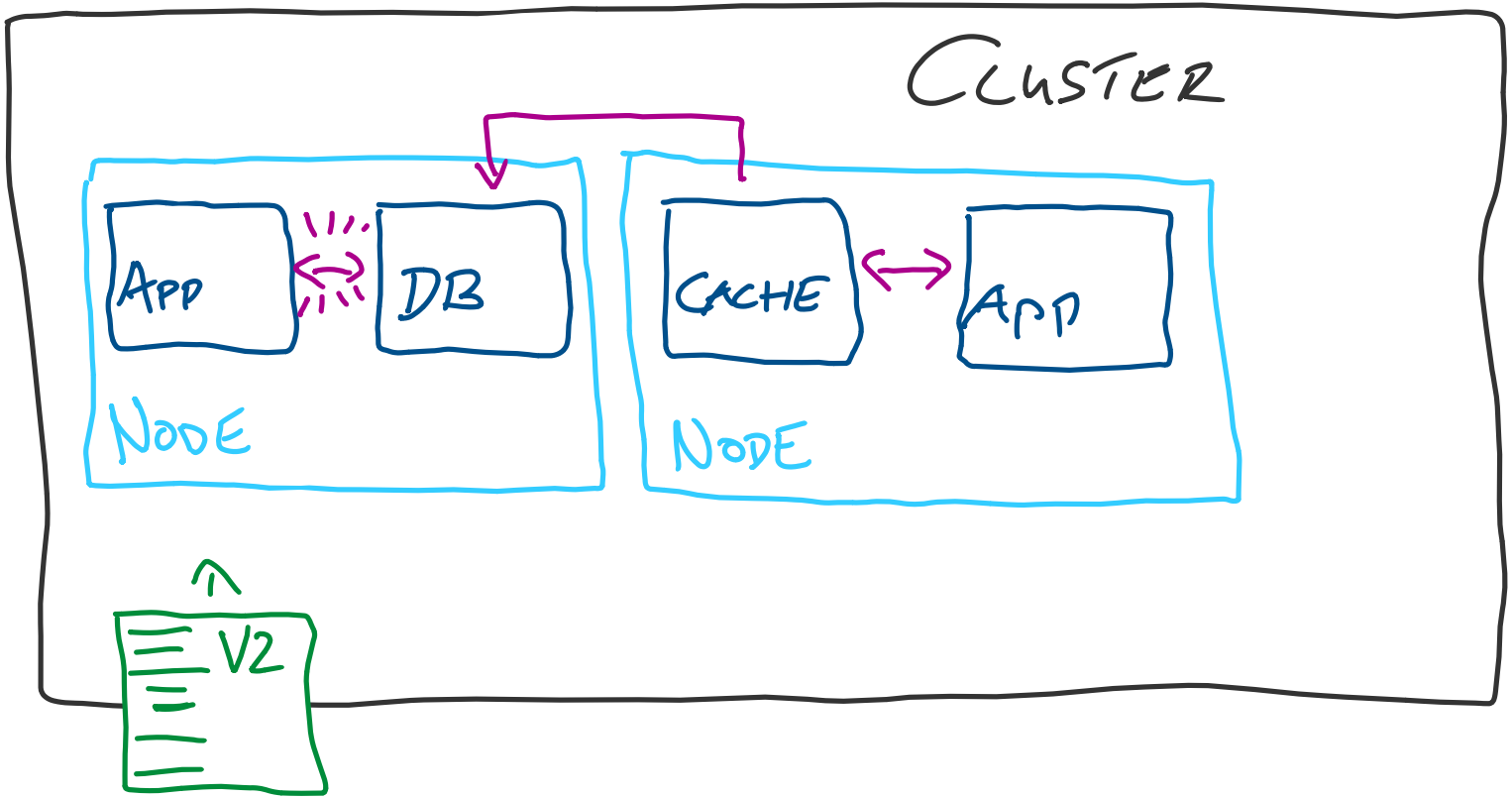
Container-optimiertes Linux?



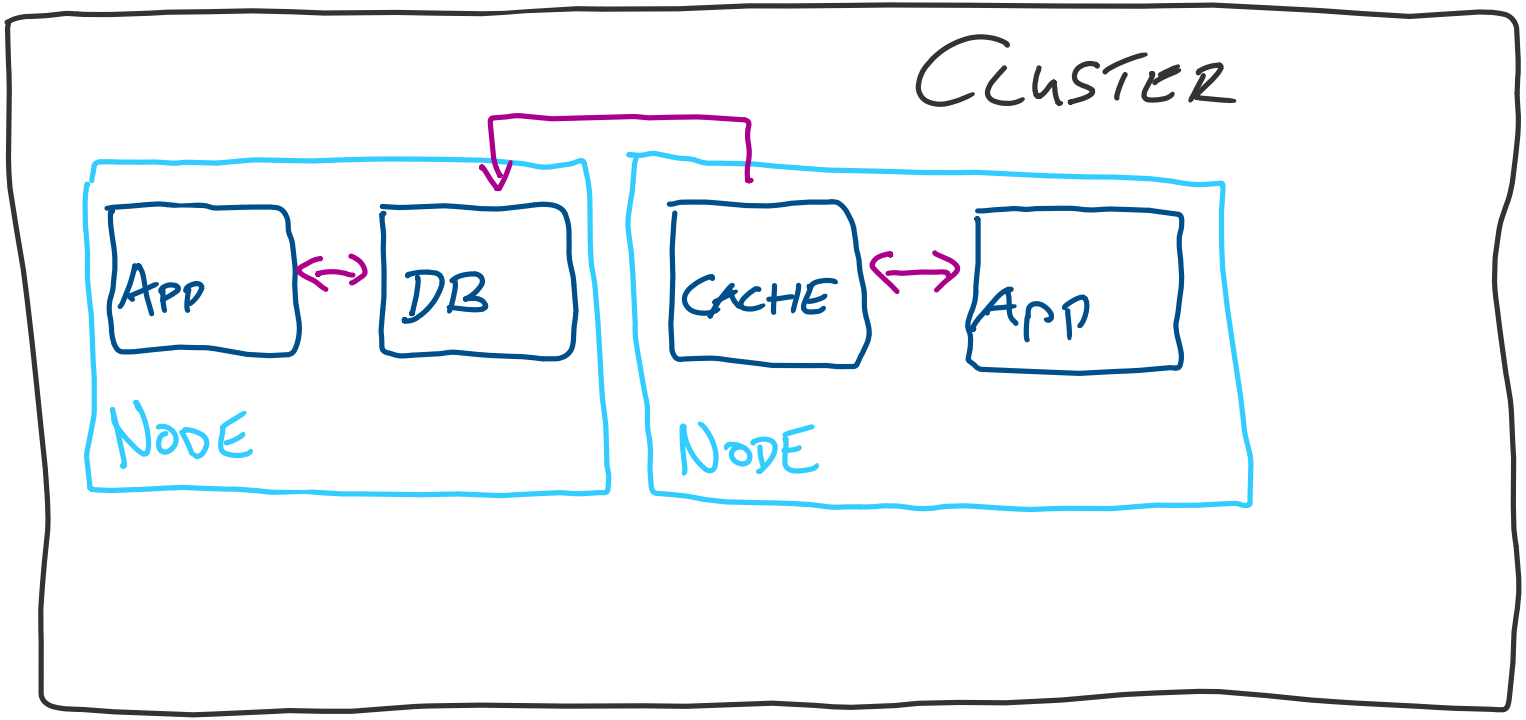
Container-optimiertes Linux?



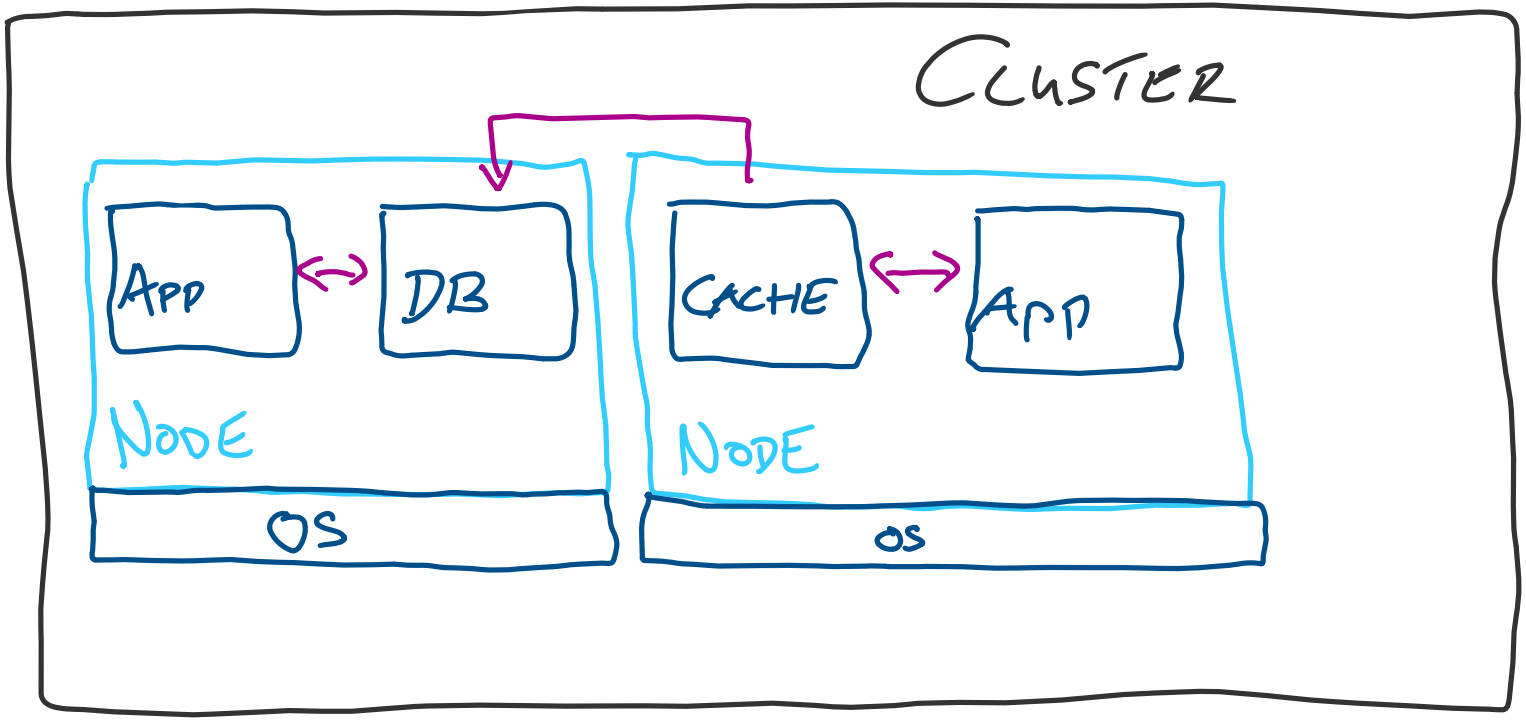
Container-optimiertes Linux?



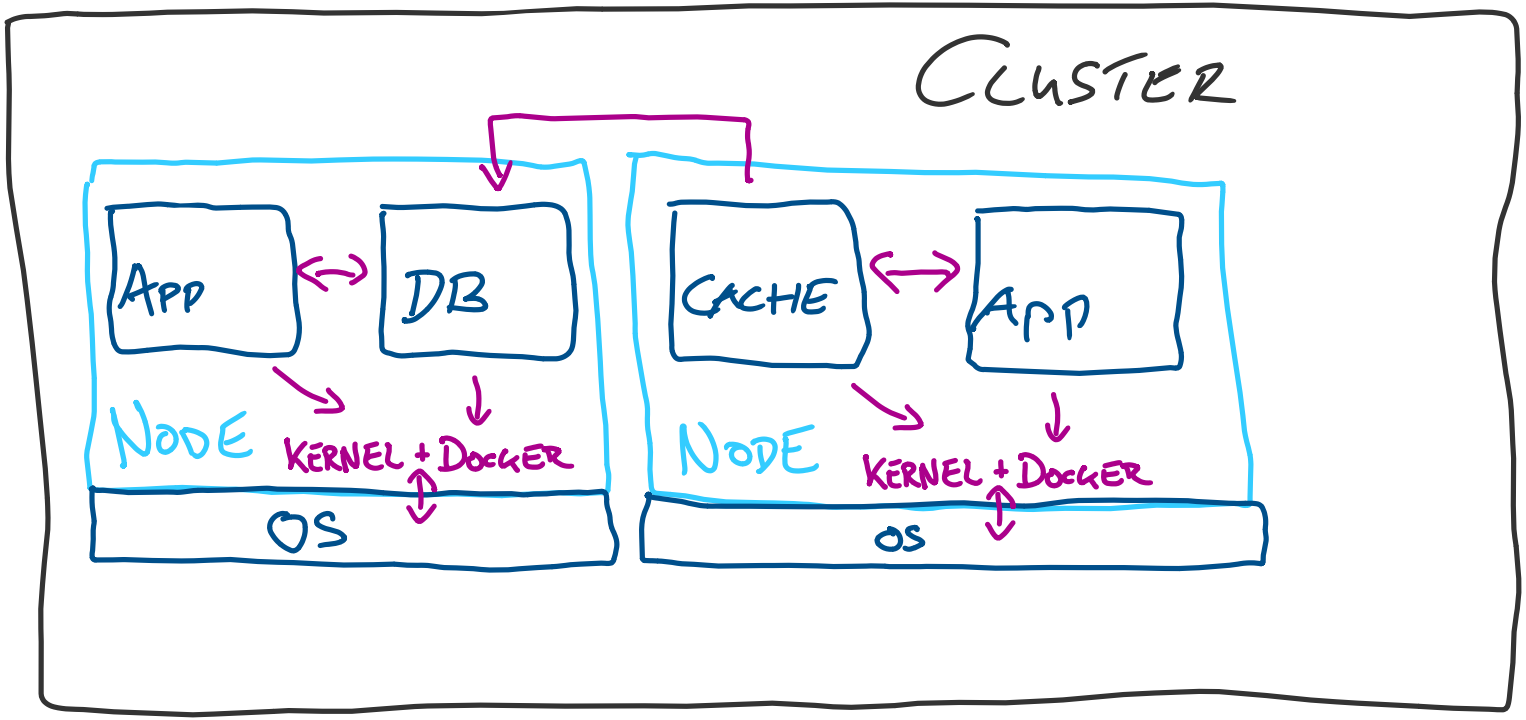
Container-optimiertes Linux?



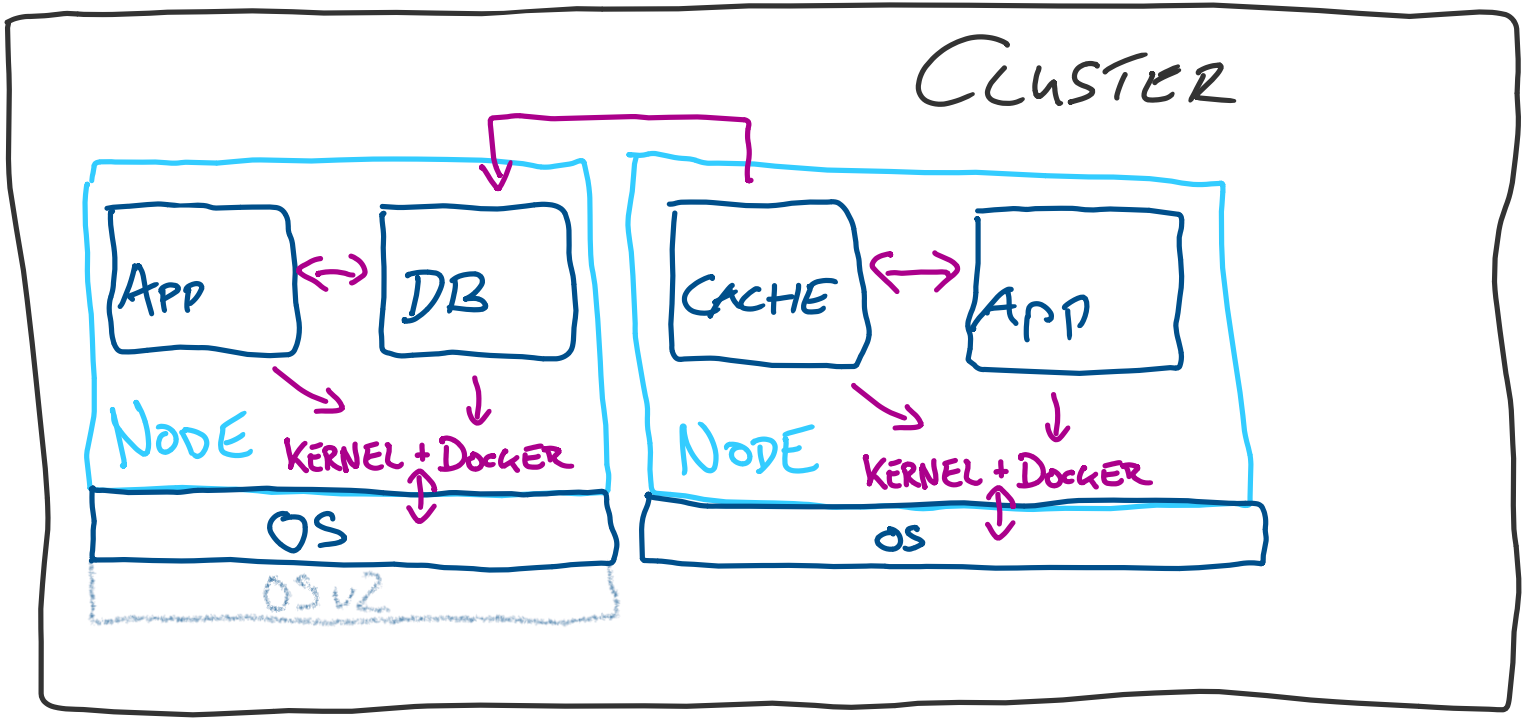
Container-optimiertes Linux?



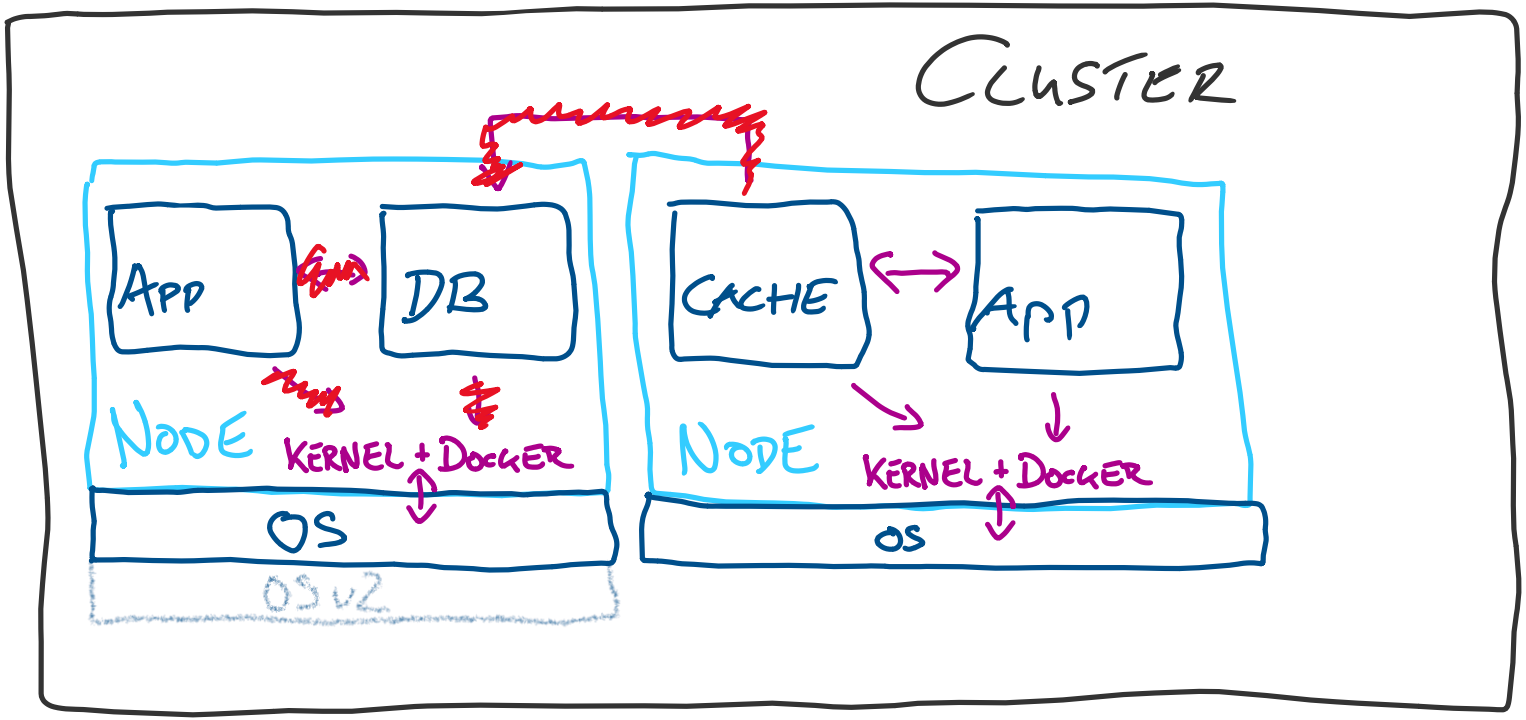
Container-optimiertes Linux?



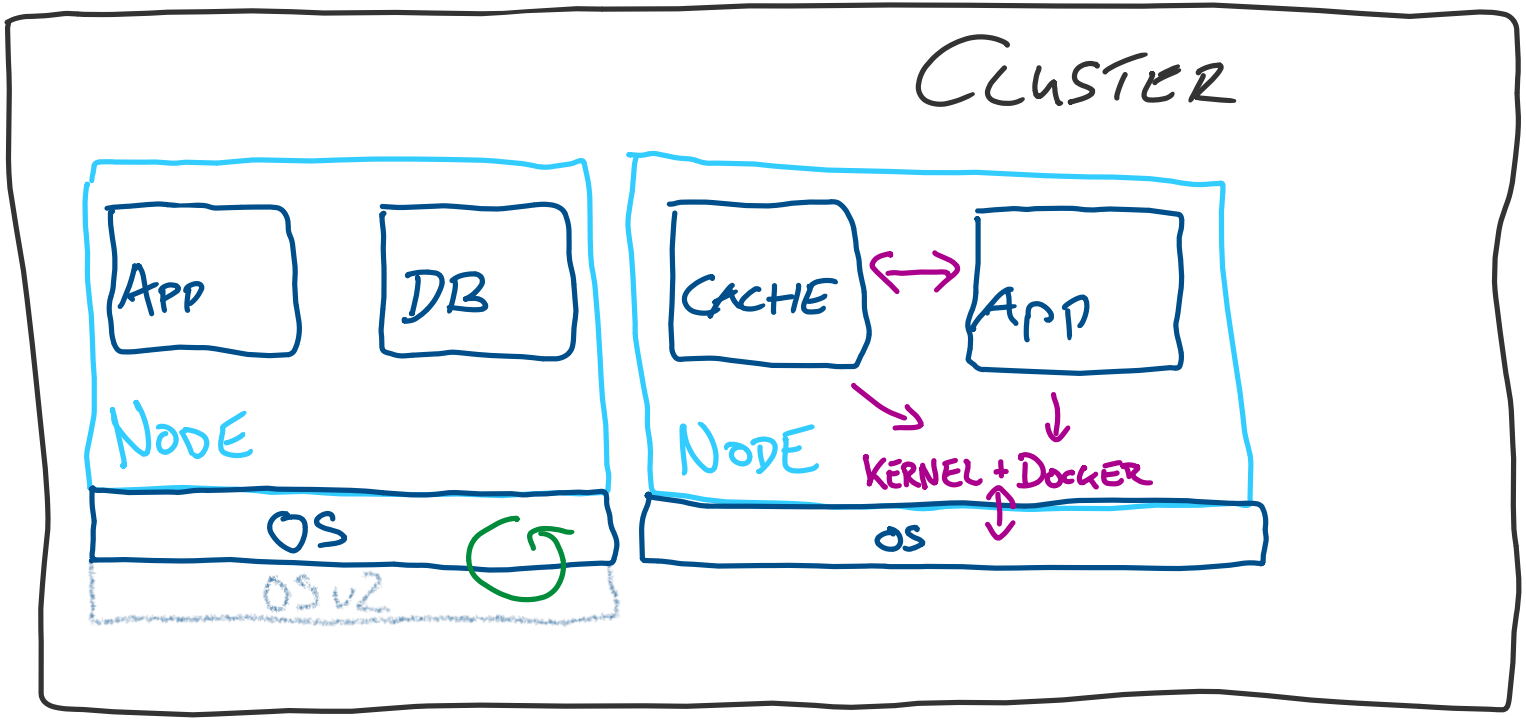
Container-optimiertes Linux?



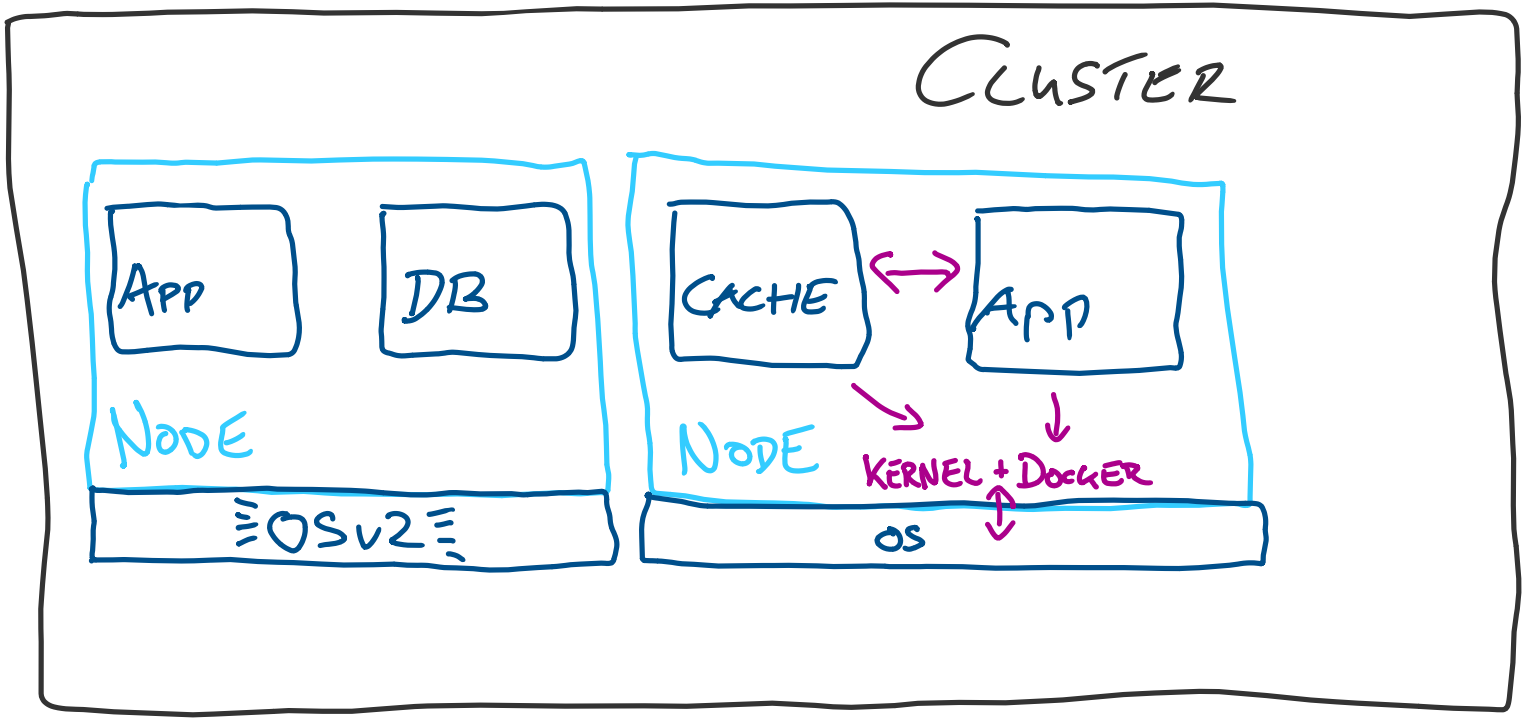
Container-optimiertes Linux?



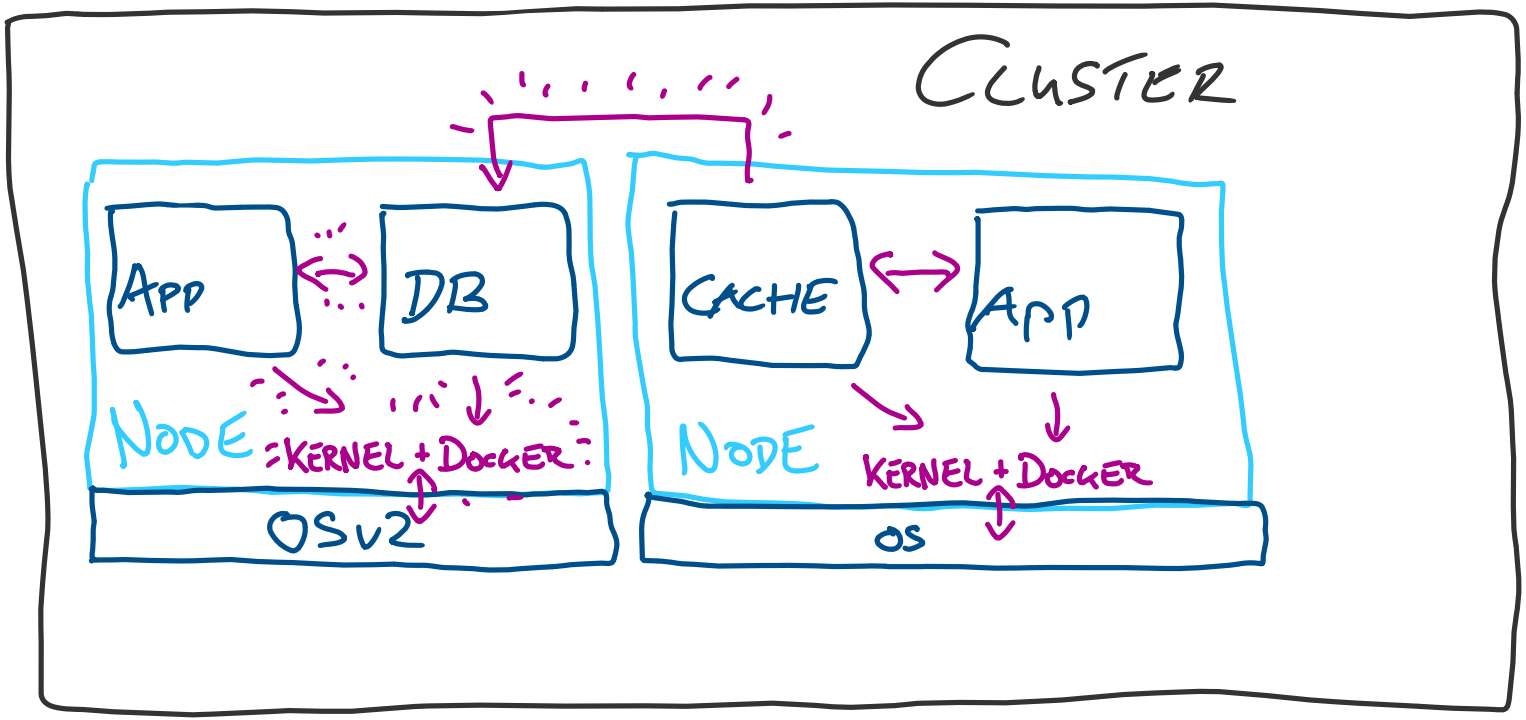
Container-optimiertes Linux?



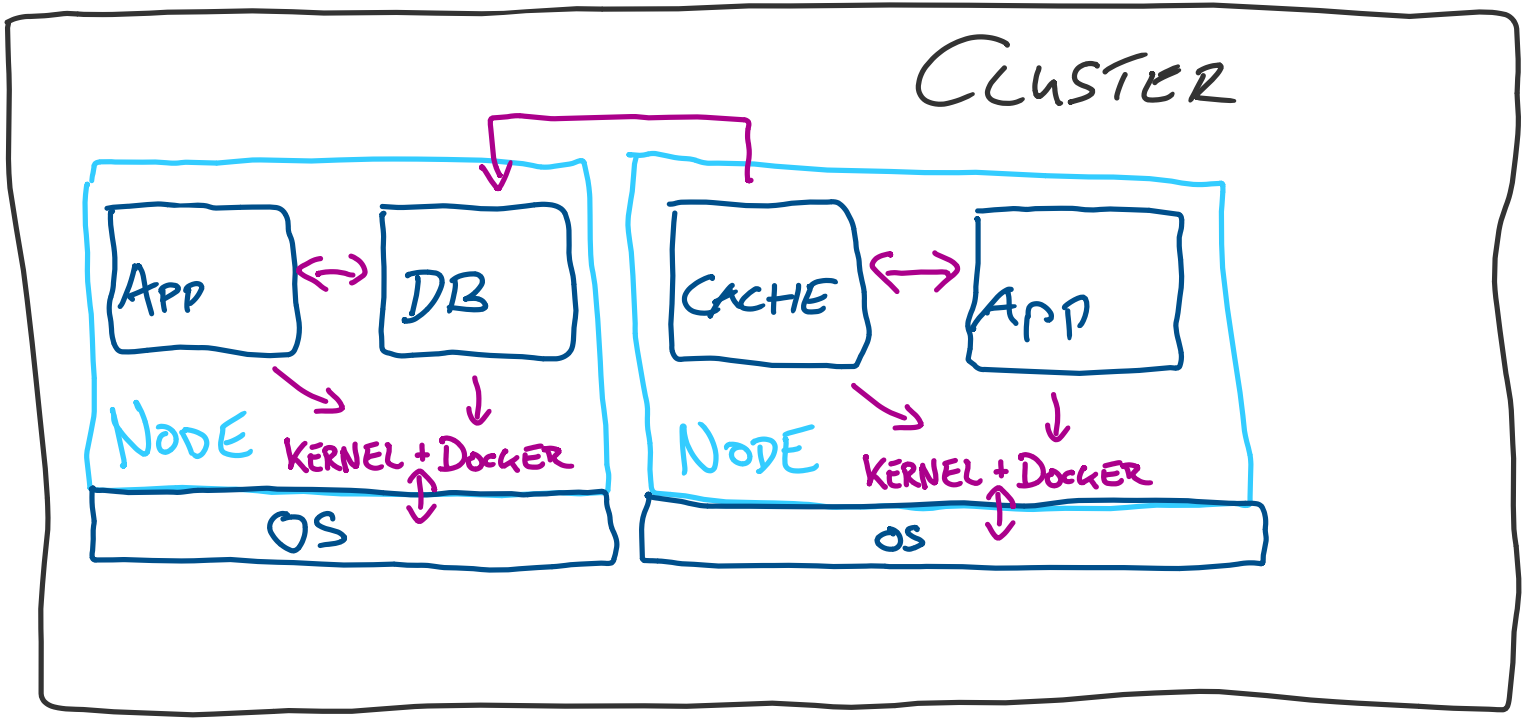
Container-optimiertes Linux?



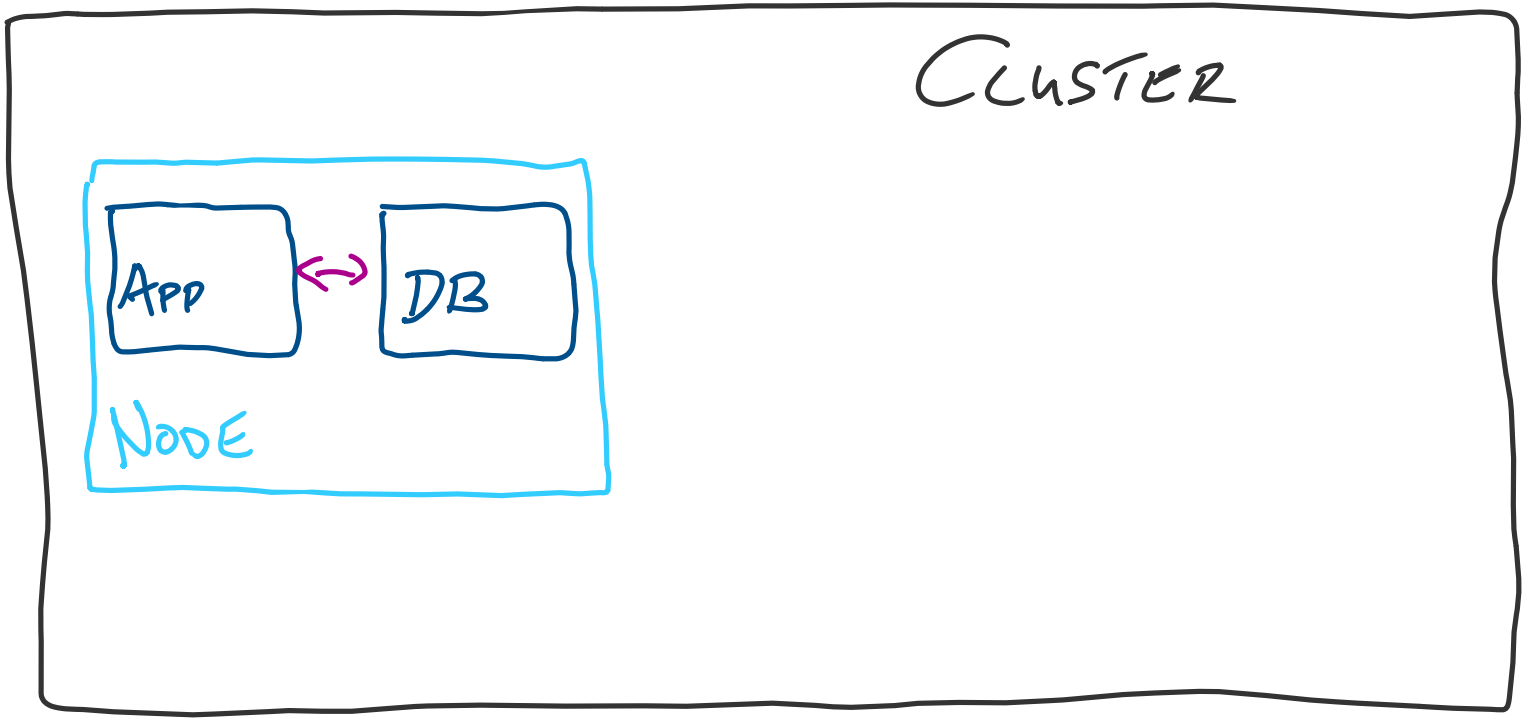
Container-optimiertes Linux?



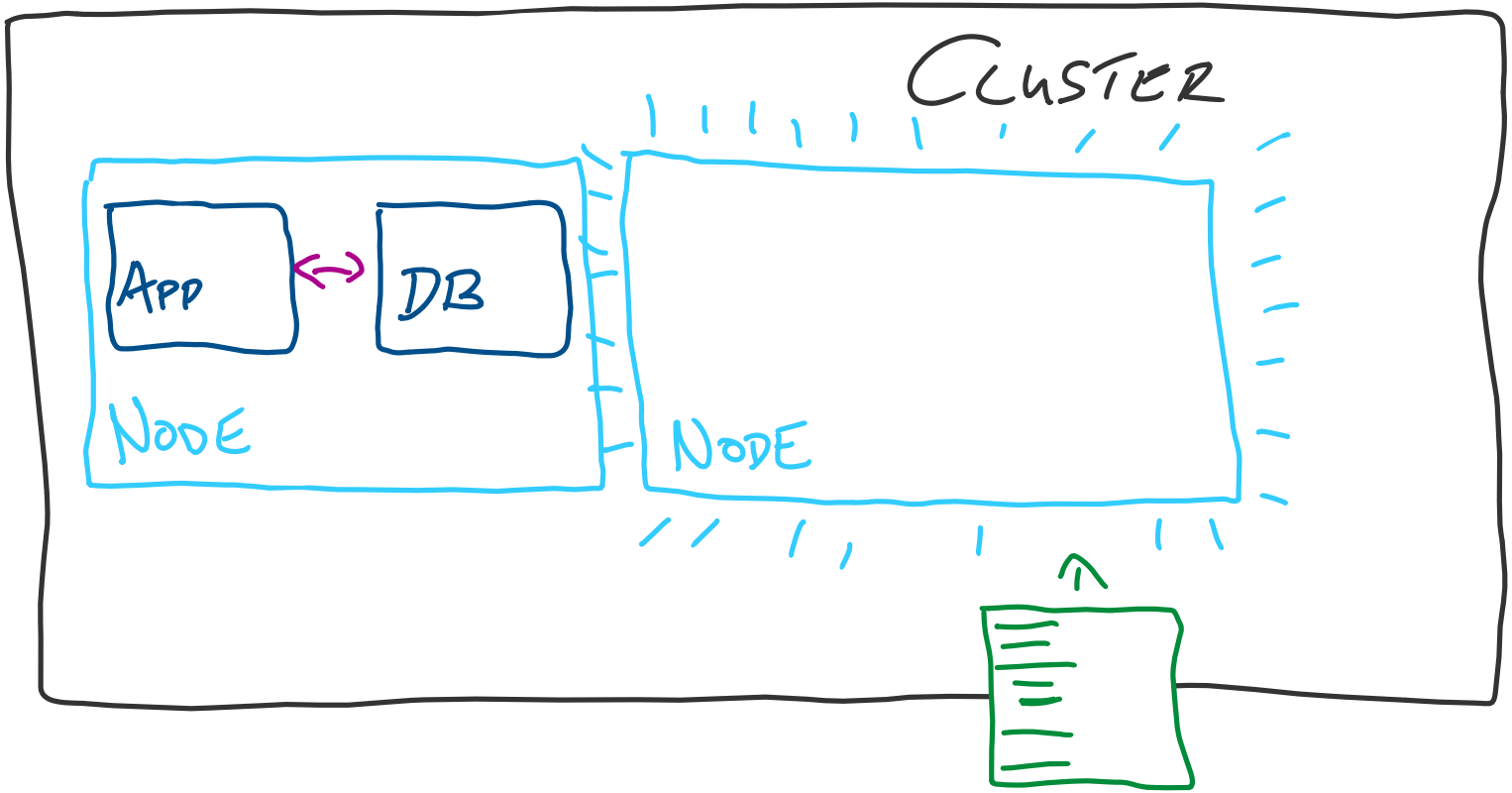
Container-optimiertes Linux?



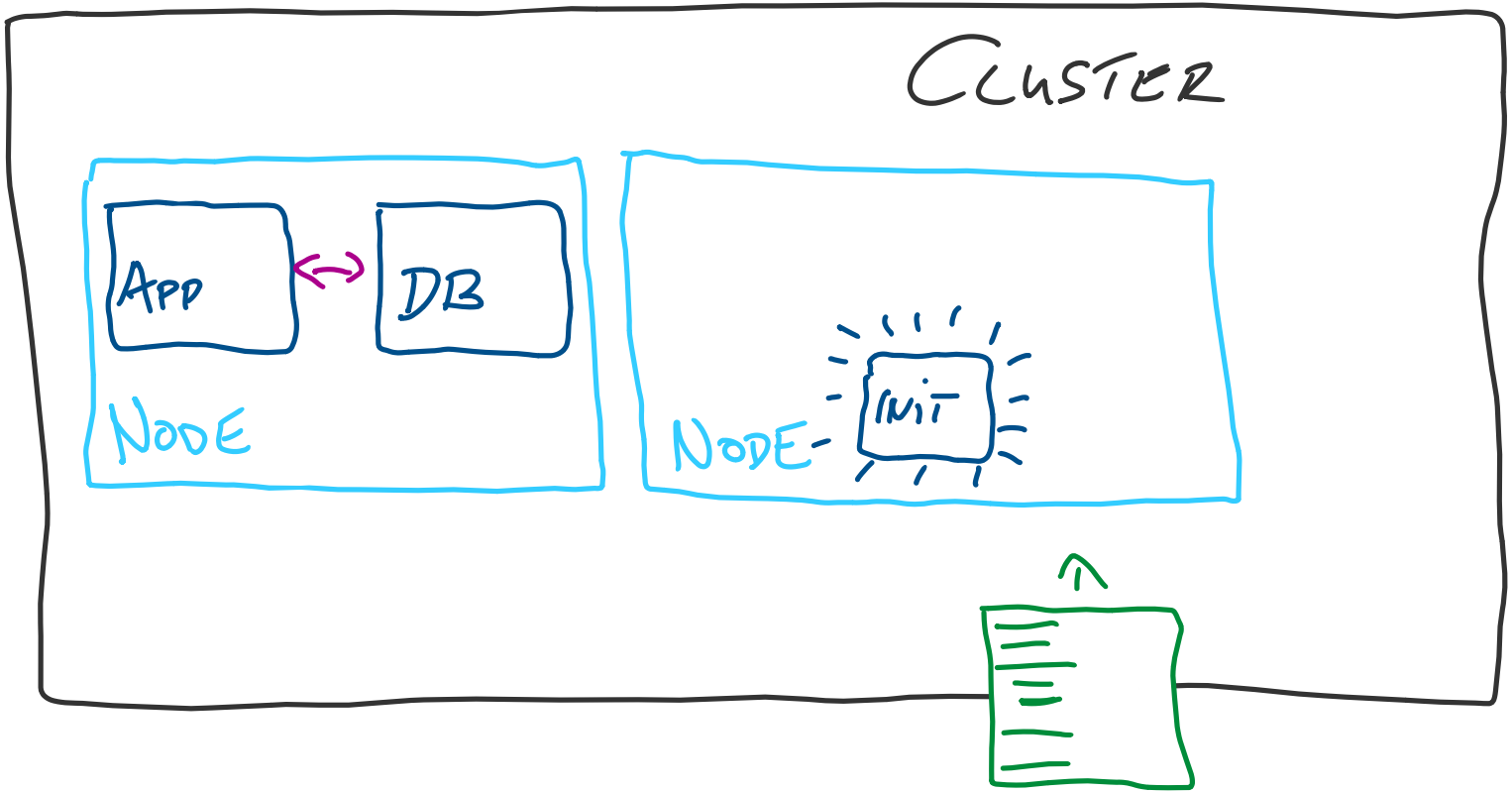
Container-optimiertes Linux?



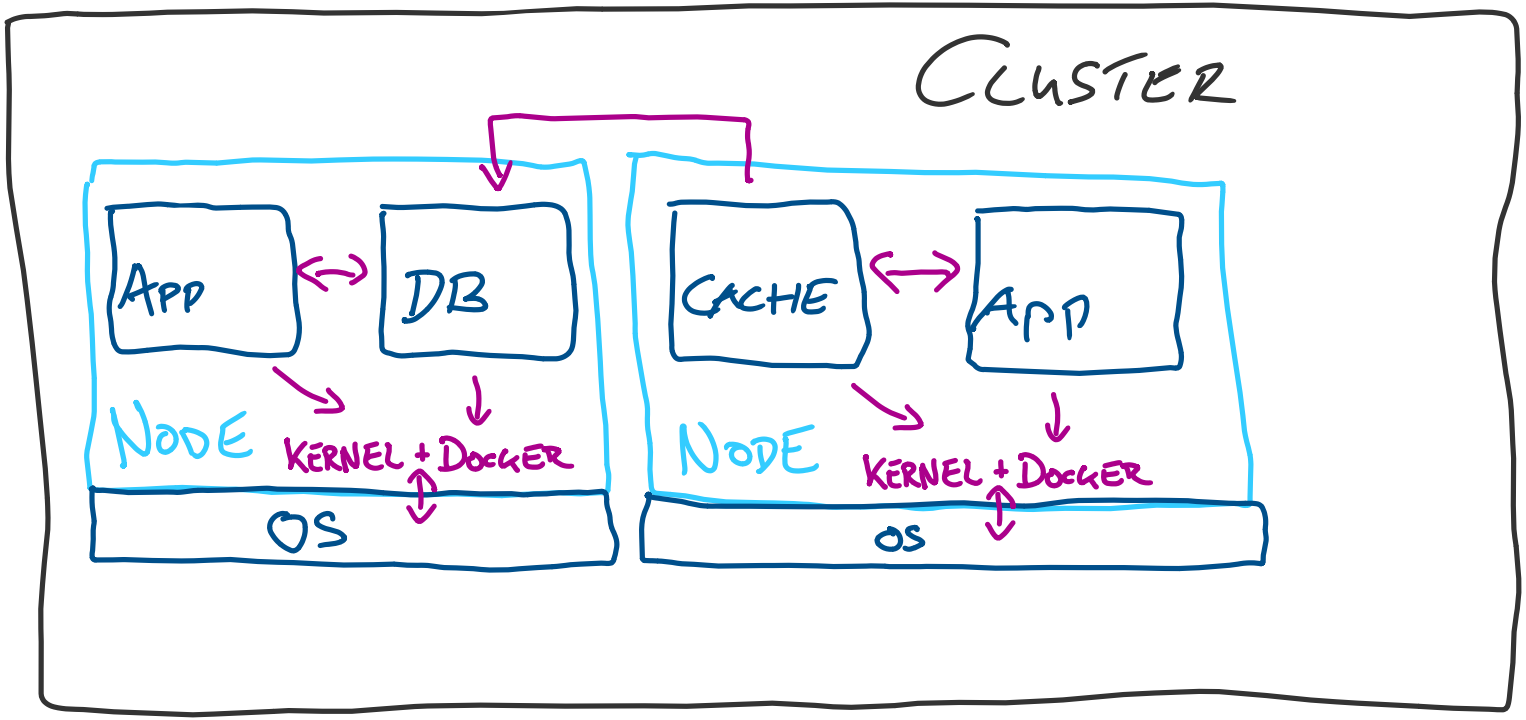
Container-optimiertes Linux?



Container-optimiertes Linux?



Container-optimiertes Linux?



Container-optimiertes Linux - Basis

Isolation der Apps untereinander und zum OS

Wohldefinierte Schnittstellen zwischen App und OS

Deklarative, automatisierbare Installation

Konfiguration bei Installation, ohne dynamische Änderungen

Automatische updates, kontrolliertes roll-(out|back)

Bereistellen, wechseln, aktivieren (oder zurückrollen)

==> Das OS als zustandslose Applikation.

Isolation

Keine Abhängigkeiten zwischen OS und Apps

Keine gemeinsamen Bibliotheken, nur Kernel und Docker.

Saubere API zwischen OS und Apps

Netzwerk, Storage etc. in Konfiguration dokumentiert.

OS-Updates (und roll-backs) ohne Seiteneffekte

Gut testbar über z.B. “Canary”-Muster.

Automatisierte Installation

Deklarative Konfiguration (per Ignition)

Aktiviert bei Installation.

Vollständig automatisierbares roll-out

Umfassende Konfigurationsoptionen

Einfache Integration in Orchestrierung

GO-bindings, Terraform provider, CAPI support, 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>Hallo, FroSCon!</h1>
          <img src="froscon.png" alt="FrOSCon logo" />
          </body></html>
    - path: /srv/www/html/froscon.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: froscon_logo_print_color.png

systemd:
  units:
    - name: froscon-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=FrOSCon 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 --name caddy \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop caddy
        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>Hallo, FroSCon!</h1>
          <img src="froscon.png" alt="FrOSCon logo" />
          </body></html>
    - path: /srv/www/html/froscon.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: froscon_logo_print_color.png

systemd:
  units:
    - name: froscon-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=FrOSCon 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 --name caddy \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop caddy
        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>Hallo, FroSCon!</h1>
          <img src="froscon.png" alt="FrOSCon logo" />
          </body></html>
    - path: /srv/www/html/froscon.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: froscon_logo_print_color.png
```


```
systemd:
  units:
    - name: froscon-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=FrOSCon 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 --name caddy \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop caddy
        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>Hallo, FroSCon!</h1>
          <img src="froscon.png" alt="FrOSCon logo" />
          </body></html>
    - path: /srv/www/html/froscon.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: froscon_logo_print_color.png
```


```
systemd:
  units:
    - name: froscon-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=FrOSCon 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 --name caddy \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop caddy
        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>Hallo, FroSCon!</h1>
          <img src="froscon.png" alt="FrOSCon logo" />
          </body></html>
    - path: /srv/www/html/froscon.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: froscon_logo_print_color.png
```


```
systemd:
  units:
    - name: froscon-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=FrOSCon 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 --name caddy \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop caddy
        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>Hallo, FroSCon!</h1>
          <img src="froscon.png" alt="FrOSCon logo" />
          </body></html>
    - path: /srv/www/html/froscon.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: froscon_logo_print_color.png
```


```
systemd:
  units:
    - name: froscon-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=FrOSCon 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 --name caddy \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop caddy
        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>Hallo, FroSCon!</h1>
          <img src="froscon.png" alt="FrOSCon logo" />
          </body></html>
    - path: /srv/www/html/froscon.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: froscon_logo_print_color.png
```


```
systemd:
  units:
    - name: froscon-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=FrOSCon 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 --name caddy \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop caddy
        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>Hallo, FroSCon!</h1>
          <img src="froscon.png" alt="FrOSCon logo" />
          </body></html>
    - path: /srv/www/html/froscon.png
      mode: 0644
      user:
        name: caddy
      group:
        name: caddy
      contents:
        local: froscon_logo_print_color.png
```


```
systemd:
  units:
    - name: froscon-demo-webserver.service
      enabled: true
      contents: |
        [Unit]
        Description=FrOSCon 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 --name caddy \
                  -v /srv/www/html:/usr/share/caddy \
                  docker.io/caddy caddy file-server \
                  --root /usr/share/caddy --access-log
        ExecStop=/usr/bin/docker stop caddy
        Restart=always
        RestartSec=5s
        [Install]
        WantedBy=multi-user.target
```

Image-basiertes Betriebssystem

Zustandslose Installation

Betriebssystem-Austausch ohne Seiteneffekte

Unveränderbare (read-only) OS-Partition

Binaries nicht änderbar (aus Versehen oder Absicht)

Verifizierbar / Attestierbar

OS ändert sich nicht und kann fortlaufend attestiert werden

Updates

Atomare Updates

Download + Bereitstellen im Hintergrund

Reboot wechselt in die neue Version

Automatisierte, konfigurierbare roll-backs

Wichtige Dienste können Rollback auslösen

Gesteuertes, kontrolliertes Ausrollen

Cluster-weite Reboot-Koordination

Updates – Stabilisierungsprozess

Alpha –> Beta -> Stable (->LTS)

Alpha für Entwickler, Beta für canaries, und Stable für Prod

Weitreichende Testszenarien für jedes Release

Komplexe Tests, z.B. Kubernetes workloads, CNI, etc.

Alle Releases gehen stets durch alle Tests

Beta-Canaries für Workload-spezifische Tests

Groβe Cluster / “at scale”

(Community) LTS

Wenn jegliche Wartung Schmerzen bereitet.

FOSS update server (Nebraska)

Updates selbst hosten und roll-out an Gruppen managen

Nutzerzentriert, Community-getrieben, Open Source

Schau zu, mach mit, und bring Dich ein!

Community-getrieben

Historisch stets FOSS, aber firmengetrieben

Direkt nach dem “Friendly Fork”

Umorientierung auf Community-Projekt

Beiträge, Koordination, Steuerung

Microsoft unterstützt als Sponsor

Öffentliche Verpflichtung bei Aquise

Microsofts Verpflichtung zur Unterstützung der 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.”

Microsofts Verpflichtung zur Unterstützung der 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."

Arbeit und Kommunikation öffentlich

Tagesgeschäft

Matrix, Slack – Maintainer-Interaktion, Release-Chat

Kurzzeit- und Langzeitplanung, Feedback

“Office hours” und “Release planning” meetings

Projekt-Boards

”Bug smashing” und “docs writing” - Tage

Mit Livestream / Videocall und Matrix chat!

Was ist grad so los? – Projekt-Boards

Implementierung

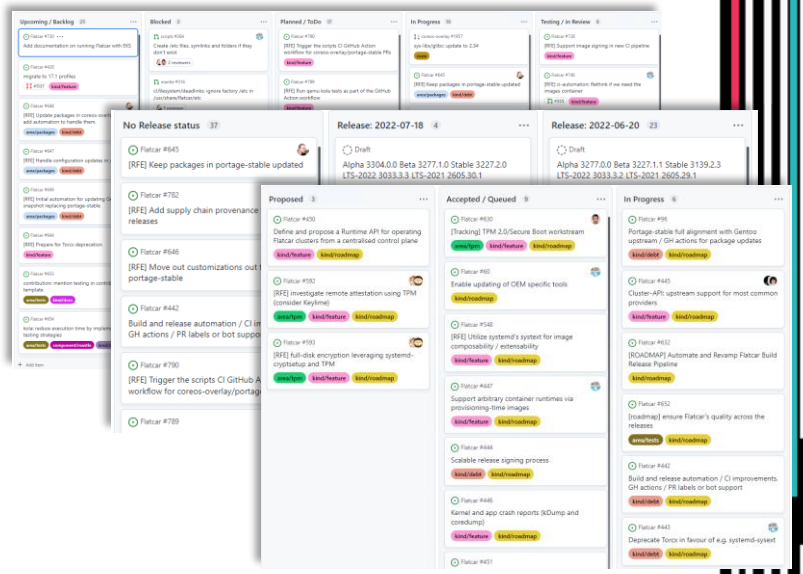
Tasks, Bugs, Features

Release

Wann ist Feature X verfügbar?

Roadmap

Epics und subtasks



Beiträge zu anderen Projekten

Gentoo

Package updates, Stabilisierungen, bug fixes

Ignition, Butane

Erweiterungen und bug fixes

Linux Kernel, Cilium, Falco

Bug fixes, Interoperationsverbesserungen

Mach mit!

Gib Feedback, melde Bugs, oder erstelle Features

Deine Meinung ist uns wichtig!

Bug smashing / doc writing - Tage

Mit Livestream und live chat

Package updates oder OS Image - Erweiterungen

Werde Flatcar-Maintainer!

Bleib in Verbindung

Office hours

Jeder zweite Dienstag im Monat um 17:30 Uhr (CE[S]T)

Matrix

Tägliche Kommunikation mit Maintainern und Entwicklern

Slack

siehe Matrix ;)

Demo time!

Wir nutzen die bereits gezeigte Konfiguration

(Nicht ganz. Wir schalten update reboots ab.)

Wir machen auch ein OS-Update.

Einfach, schnell und langweilig. Wie ein Lichtschalter.

Demo-Quellen sind hier verfügbar:

https://github.com/flatcar-linux/flatcar-demos/tree/main/froscon-2022-08-20

Thank you

Website – https://www.flatcar.org
office hours - https://github.com/flatcar-linux/Flatcar/#monthly-office-hours-and-release-planning
Mach mit – https://github.com/flatcar-linux/Flatcar
Melde Dich – https://app.element.io/#/room/#flatcar:matrix.org
– https://kubernetes.slack.com/archives/C03GQ8B5XNJ