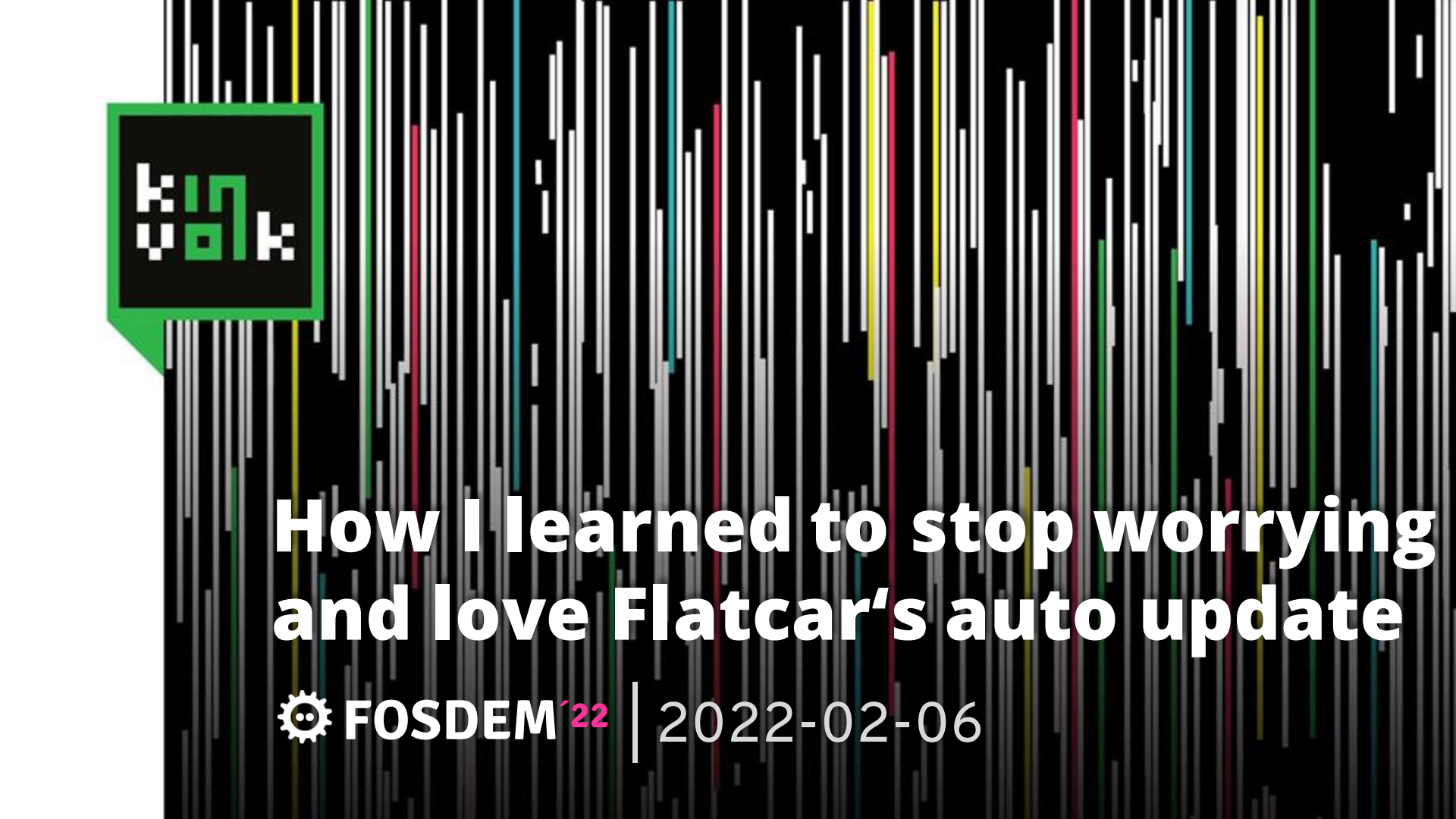


How I learned to stop worrying and love Flatcar's auto update

FOSDEM'22 | 2022-02-06

Hi, I'm Thilo

Thilo Fromm
Engineering manager, Microsoft

Github: t-lo
Twitter: ThiloFM
Email: thilofromm@microsoft.com

What's in the box?

This is an operations talk.
It is about updating the OS on your cluster. Automatedly.

It might be a tad boring.
Because OS updates should be.
But sometimes aren't.

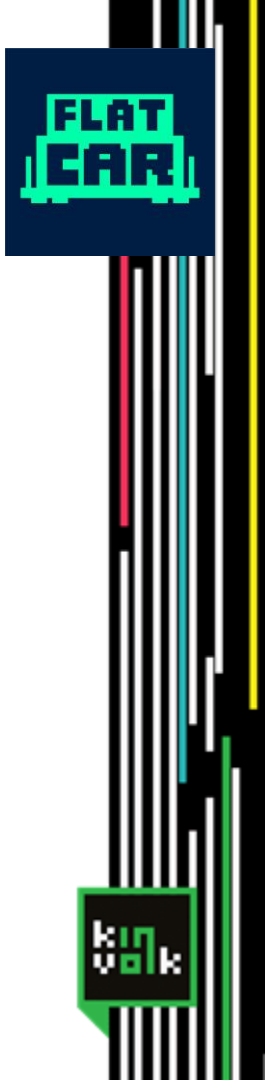
(The talk is also about Flatcar Container Linux.
Because we implement the good practices presented today)

Why even?

Stability and Performance

Pro-active security

Compliance



Nothing ever comes for free

Stability and Performance vs. new issues / bugs

Pro-active security vs. supply chain attacks

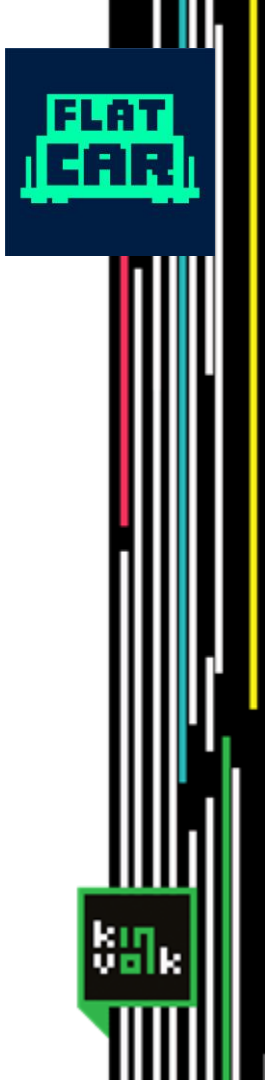
Compliance vs. maintenance overhead

But we can lower the costs

Keep changes manageable

Minimise blast radius of impacts

Ensure mistakes can be forgiven



Enter Flatcar Container Linux

Released as Image

- Updates / patches are also full images
- All releases undergo thorough testing

Stabilisation process makes Canaries easy to support

- New major releases go through stabilisation in stages ("channels"- Alpha -> Beta -> Stable)

Updates are atomic, roll-backs are built-in

Keep changes manageable

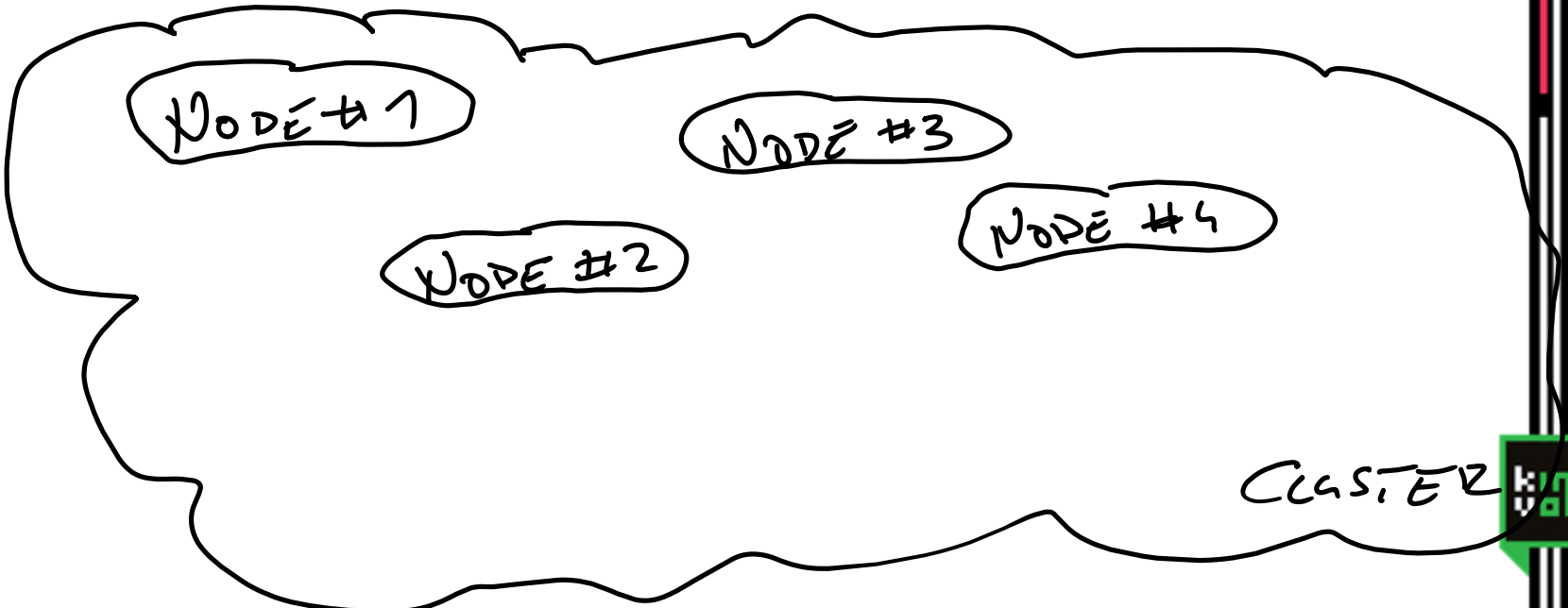
Flatcar releases / updates always come as an image

No package management, no version diversity, no diversity creep

Keep changes manageable

Flatcar releases / updates always come as an image

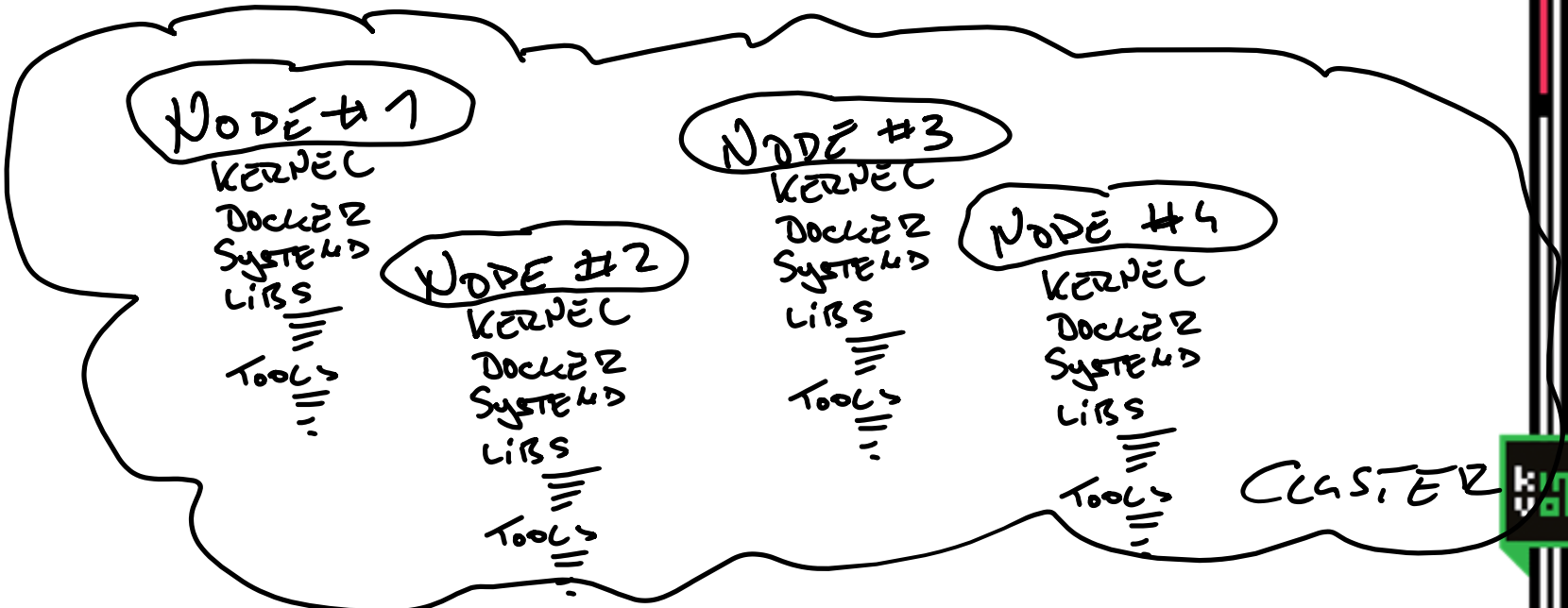
No package management, no version diversity, no diversity creep



Keep changes manageable

Flatcar releases / updates always come as an image

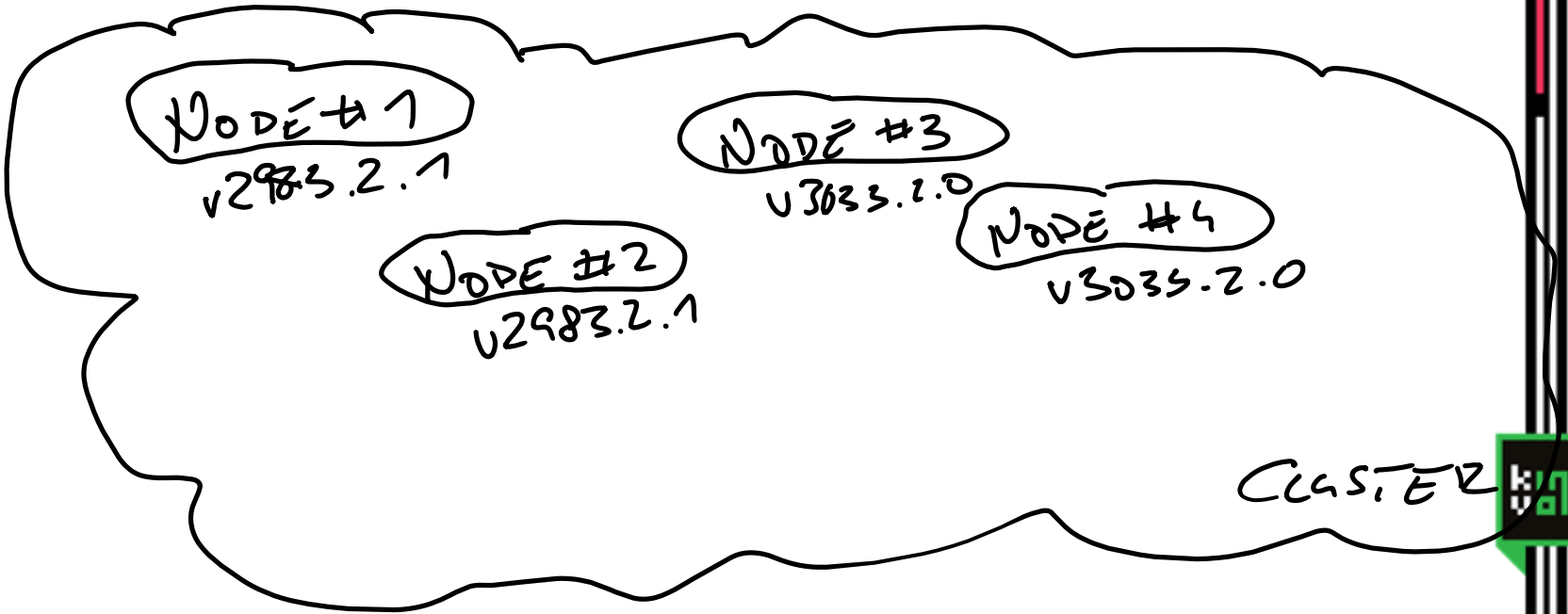
No package management, no version diversity, no diversity creep



Keep changes manageable

Flatcar releases / updates always come as an image

No package management, no version diversity, no diversity creep



Keep changes manageable

Flatcar releases / updates always come as an image

- No package management, no version diversity, no diversity creep

Changesets are tested

Updates can be vetted before roll-out

No difference between new nodes and updated nodes

Package-focused distros: Do your own releases (w/ distro binary packages).

- Create your own changesets / do custom gatekeeping
- Operate your own mirror / package server
- Run changesets through custom test harness before roll-out

Minimise blast radius

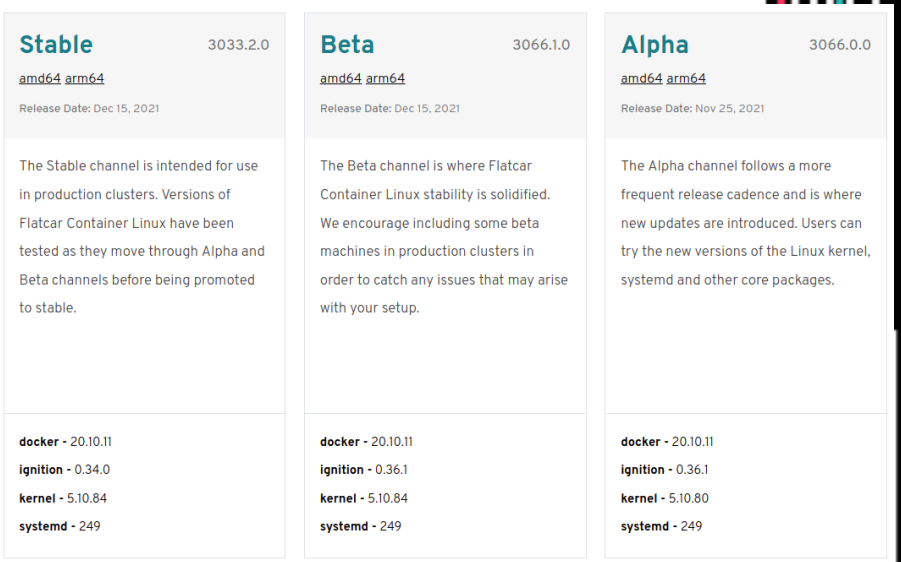
Test new changesets (“release testing”) for the main feature you use
- basic provisioning + configuration
- managing containers
- Kubernetes
- cluster networking

Use pre-prod and/or Canaries in prod clusters
- Pre-prod can be expensive
- Single (or small number of) nodes in prod
  - to validate your specific use case

Minimise blast radius

Flatcar release channels support validating your use case:

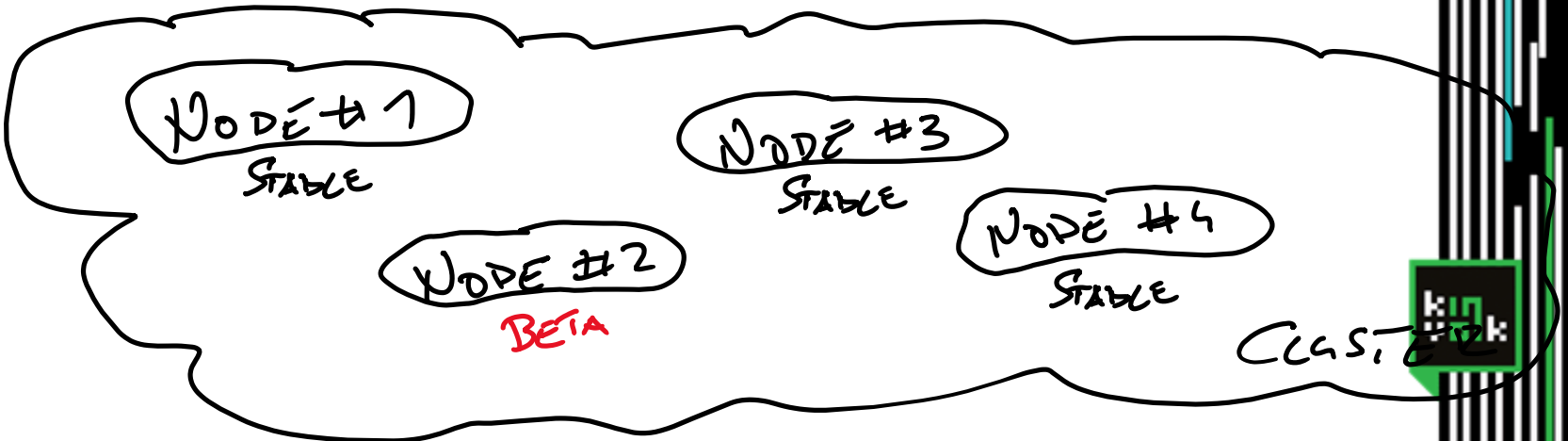
- ❑ *Any major change* in a changeset mandates a new major release.
- ❑ Major releases land in **Alpha** first.
  Alpha is for quick iteration.
  Each Alpha must pass full testing.
- ❑ **Beta** ships *meaningful* changesets.
  Beta is for user / use case validation.
- ❑ **Stable** ships *production-ready* changesets.



Stable	3033.2.0	Beta	3066.1.0	Alpha	3066.0.0
amd64 arm64		amd64 arm64		amd64 arm64	
Release Date: Dec 15, 2021		Release Date: Dec 15, 2021		Release Date: Nov 25, 2021	
The Stable channel is intended for use in production clusters. Versions of Flatcar Container Linux have been tested as they move through Alpha and Beta channels before being promoted to stable.		The Beta channel is where Flatcar Container Linux stability is solidified. We encourage including some beta machines in production clusters in order to catch any issues that may arise with your setup.		The Alpha channel follows a more frequent release cadence and is where new updates are introduced. Users can try the new versions of the Linux kernel, systemd and other core packages.	
docker - 20.10.11		**docker** - 20.10.11		**docker** - 20.10.11	
ignition - 0.34.0		**ignition** - 0.36.1		**ignition** - 0.36.1	
kernel - 5.10.84		**kernel** - 5.10.84		**kernel** - 5.10.80	
systemd - 249		**systemd** - 249		**systemd** - 249	

Minimise blast radius

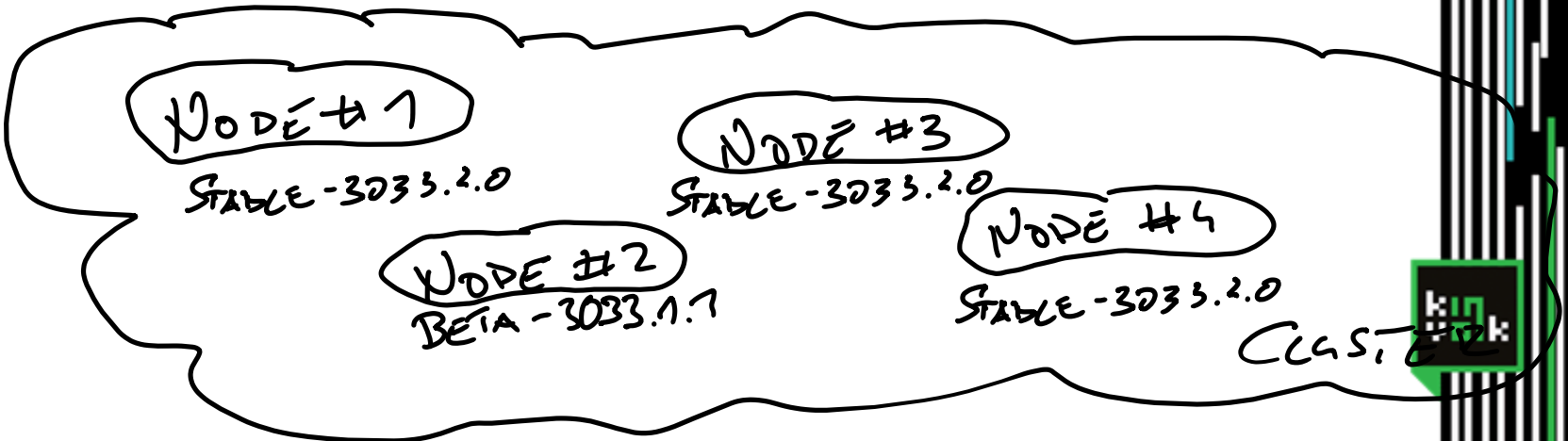
Use canaries in your production clusters.



Minimise blast radius

Use canaries in your production clusters.

New Alpha: 3066.0.0 – not actionable / take note

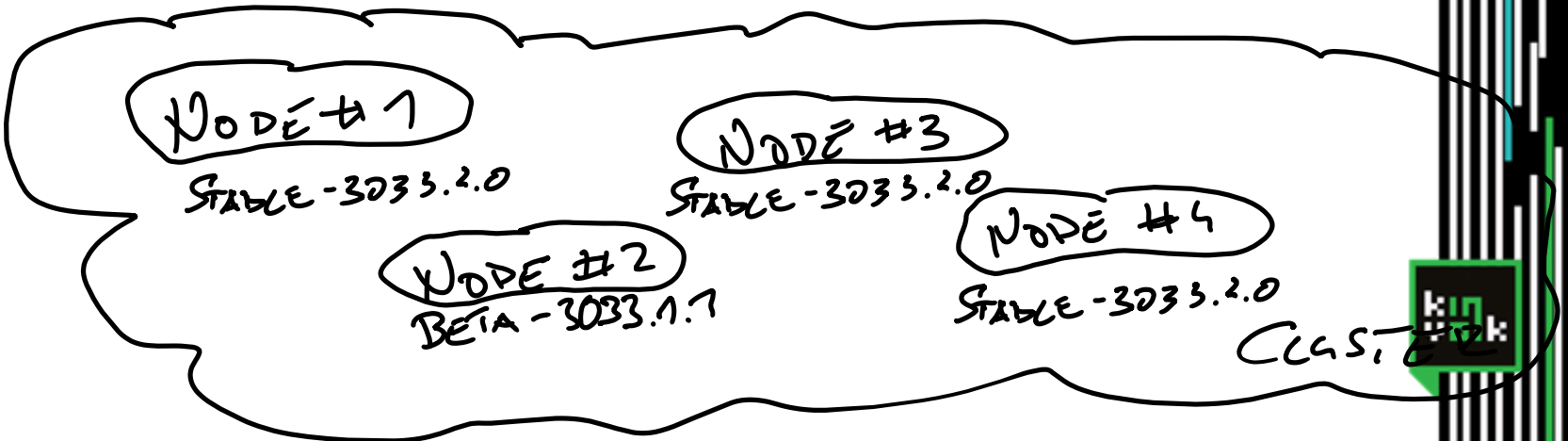


Minimise blast radius

Use canaries in your production clusters.

New Alpha: 3066.0.0 – not actionable / take note

New Beta : 3066.1.0

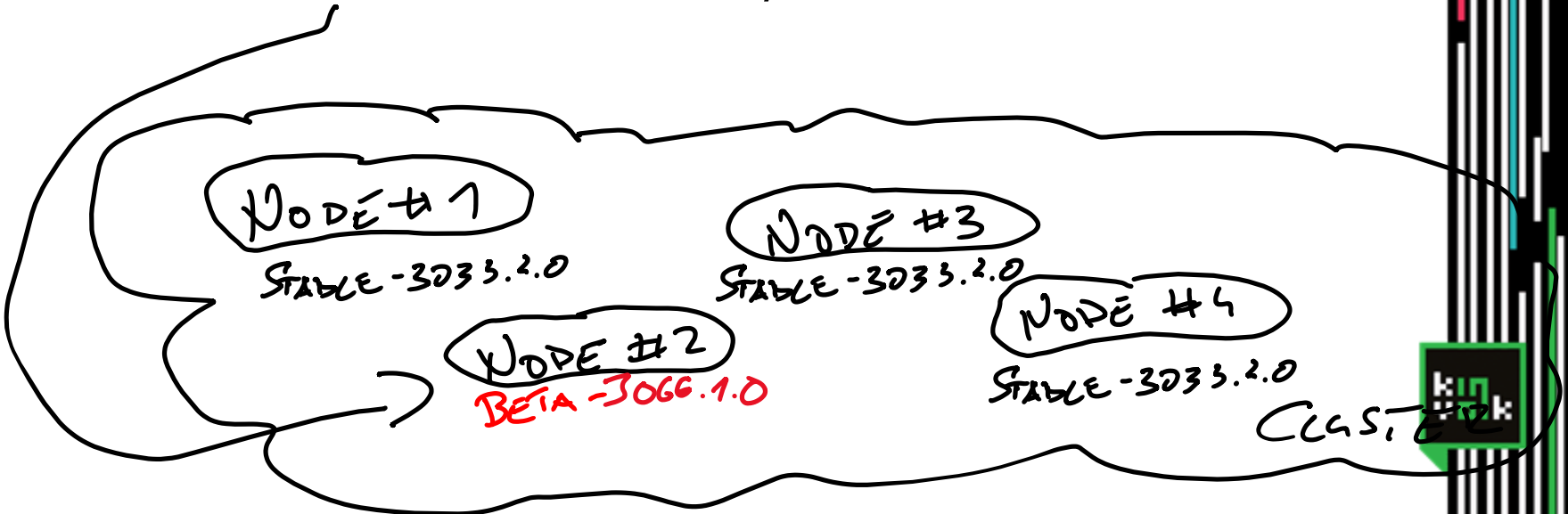


Minimise blast radius

Use canaries in your production clusters.

New Alpha: 3066.0.0 – not actionable / take note

New Beta : 3066.1.0 – lands on canary

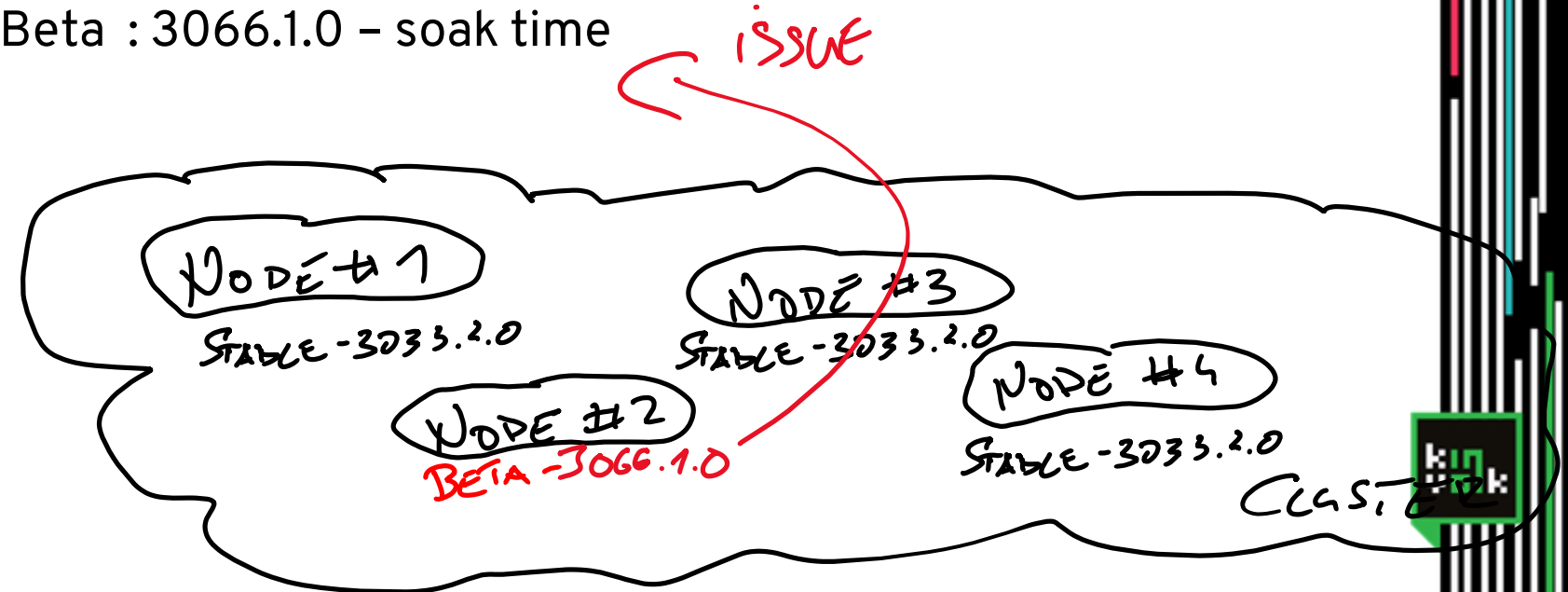


Minimise blast radius

Use canaries in your production clusters.

New Alpha: 3066.0.0 – not actionable / take note

New Beta : 3066.1.0 – soak time

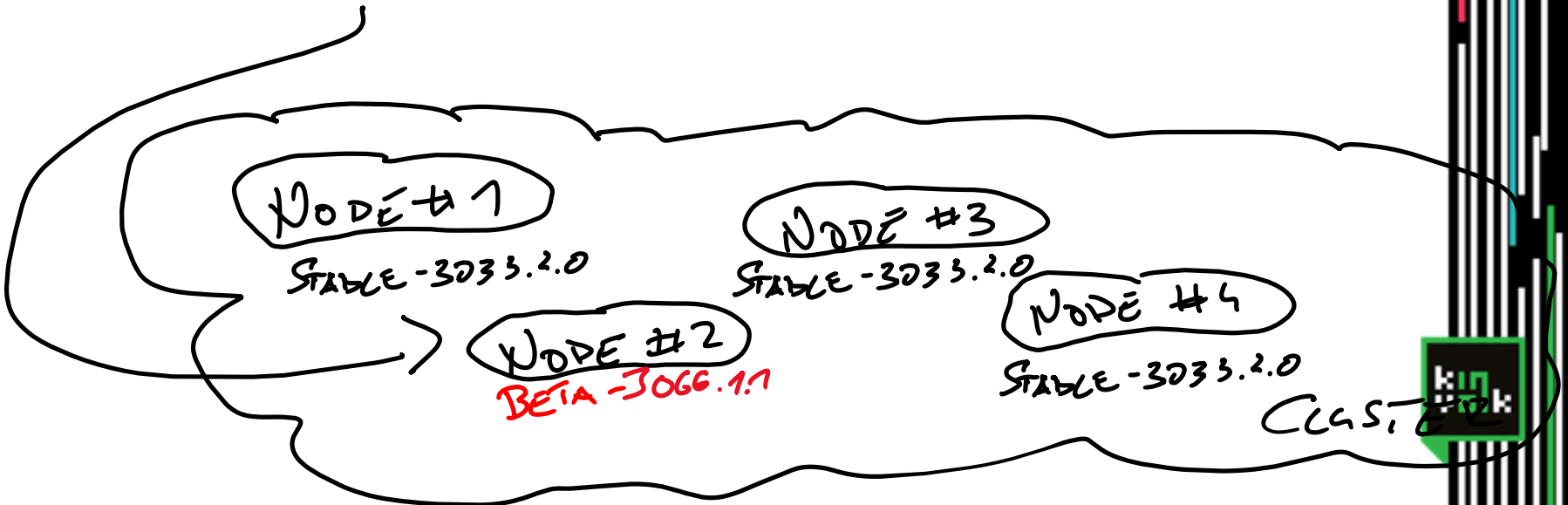


Minimise blast radius

Use canaries in your production clusters.

New Alpha: 3066.0.0 – not actionable / take note

New Beta : 3066.1.1 – Fix

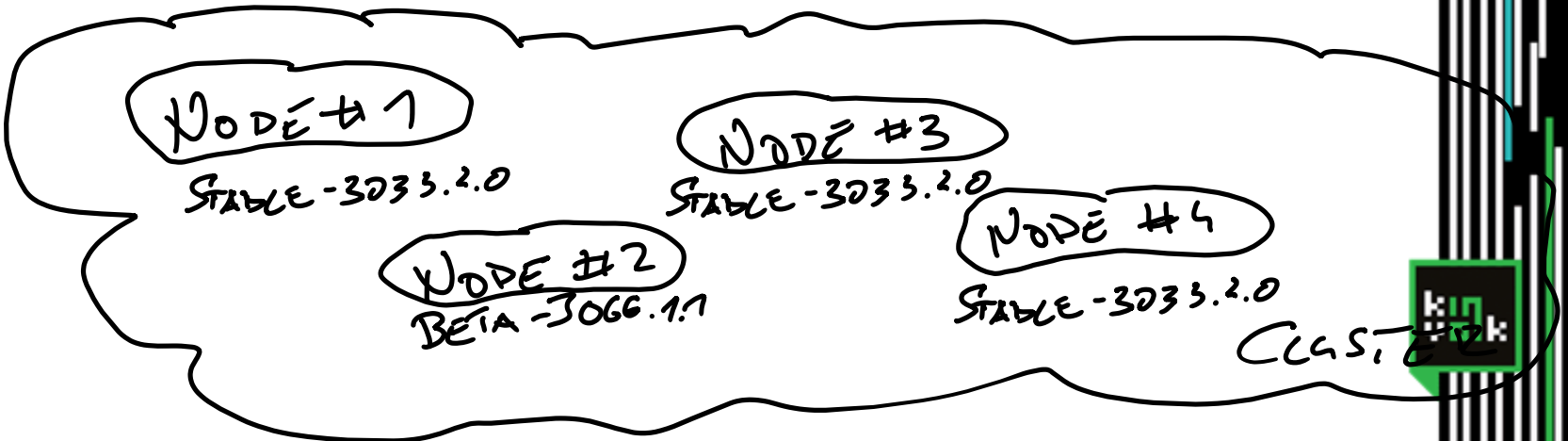


Minimise blast radius

Use canaries in your production clusters.

New Alpha: 3066.0.0 – not actionable / take note

New Beta : 3066.1.1 – Soaked



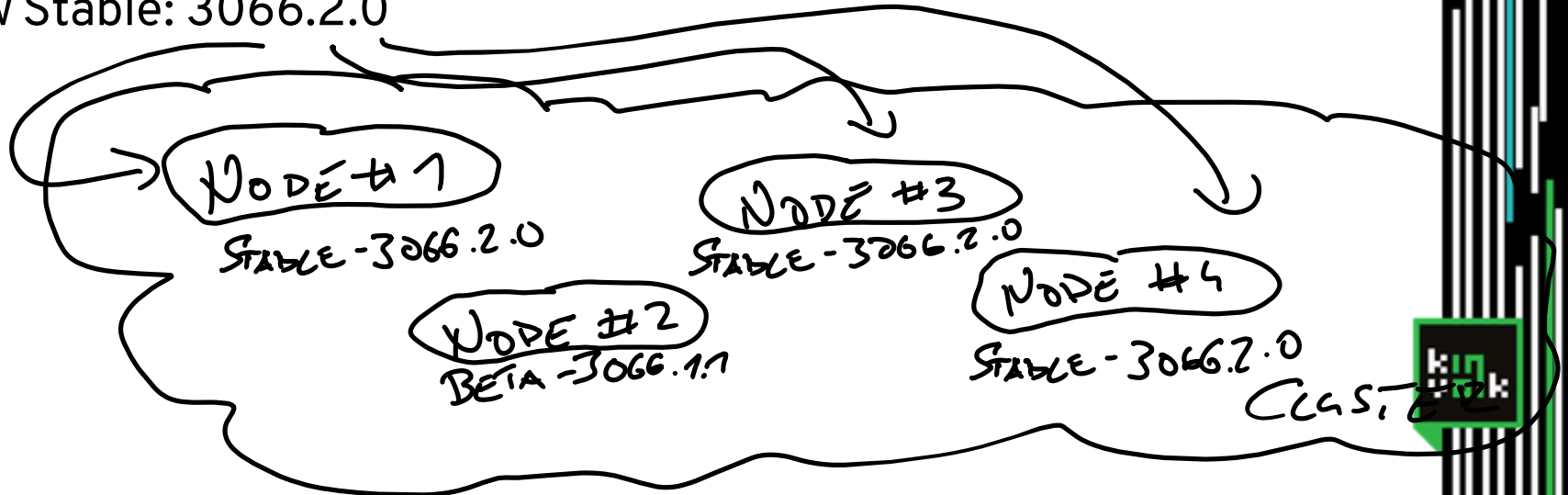
Minimise blast radius

Use canaries in your production clusters.

New Alpha: 3066.0.0 – not actionable / take note

New Beta : 3066.1.1 – Soaked

New Stable: 3066.2.0



Minimise blast radius

Release testing cannot cover all use cases (breadth, not depth)

Managing our own releases allows us to effectively canary changesets
allowing for in-depth validation

Issues are detected early and resolved before mass roll-out

Beta canaries are expected to have higher failure rate / regular roll-backs
(but it's "some pain" vs. "all the pain")

Make Mistakes forgivable

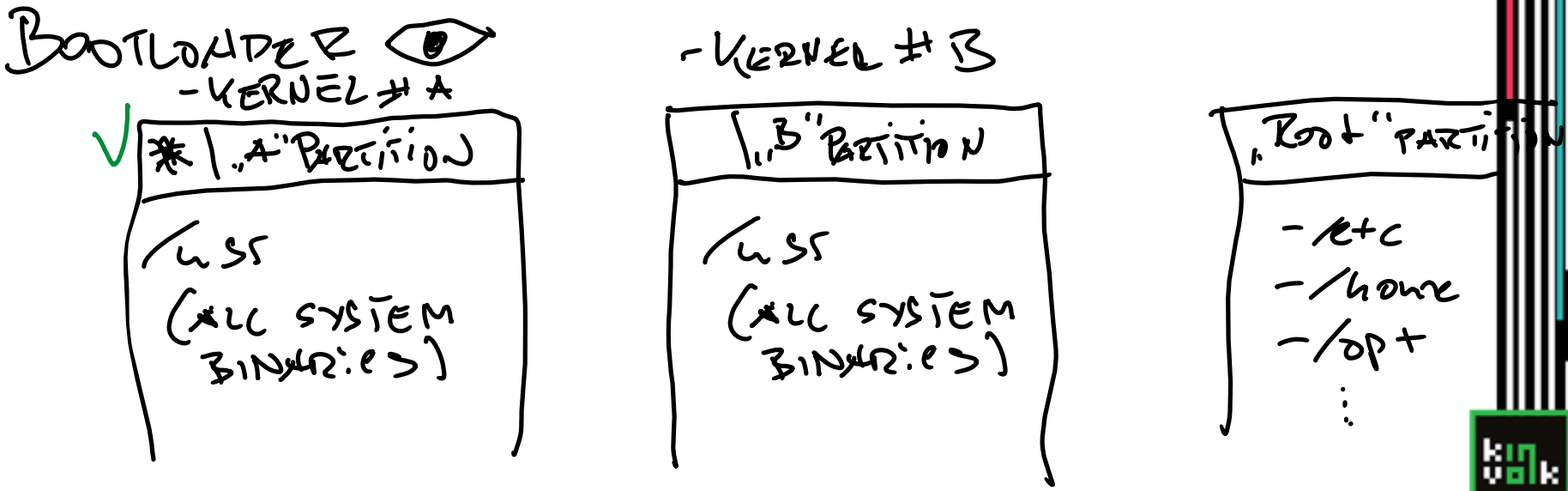
For regular updates we need a roll-back strategy

i.e. switch back to previous changeset

“No one wants backup, everybody wants restore”

Make Mistakes forgivable

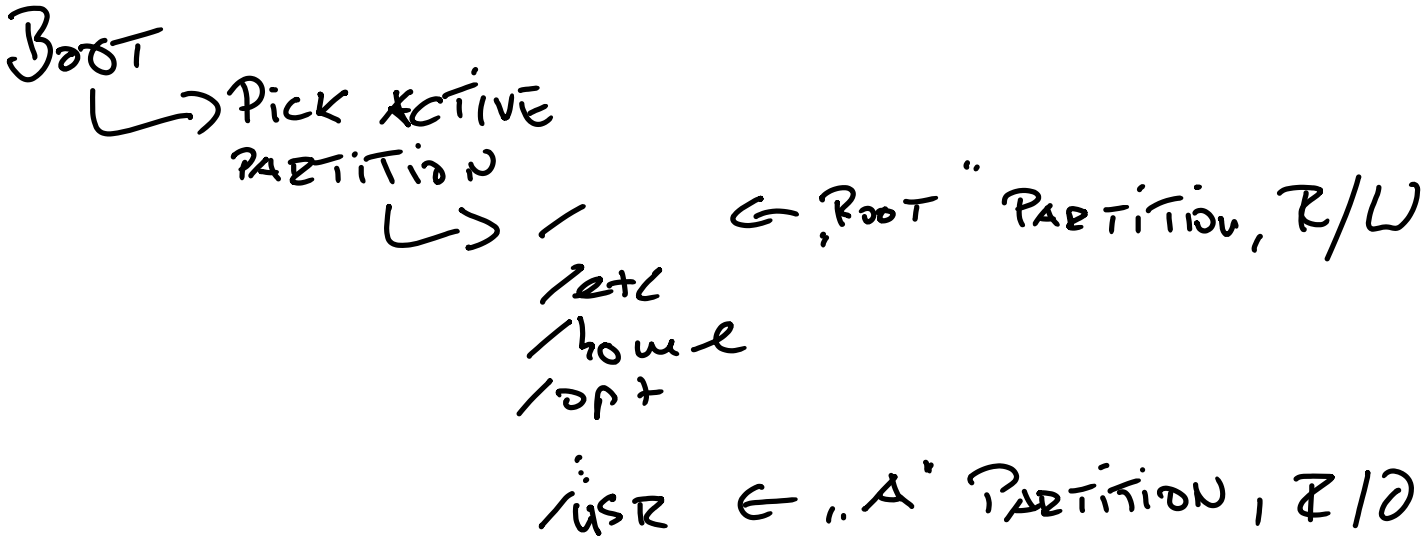
For roll-backs, Flatcar utilises an A/B partitioning scheme



Make Mistakes forgivable

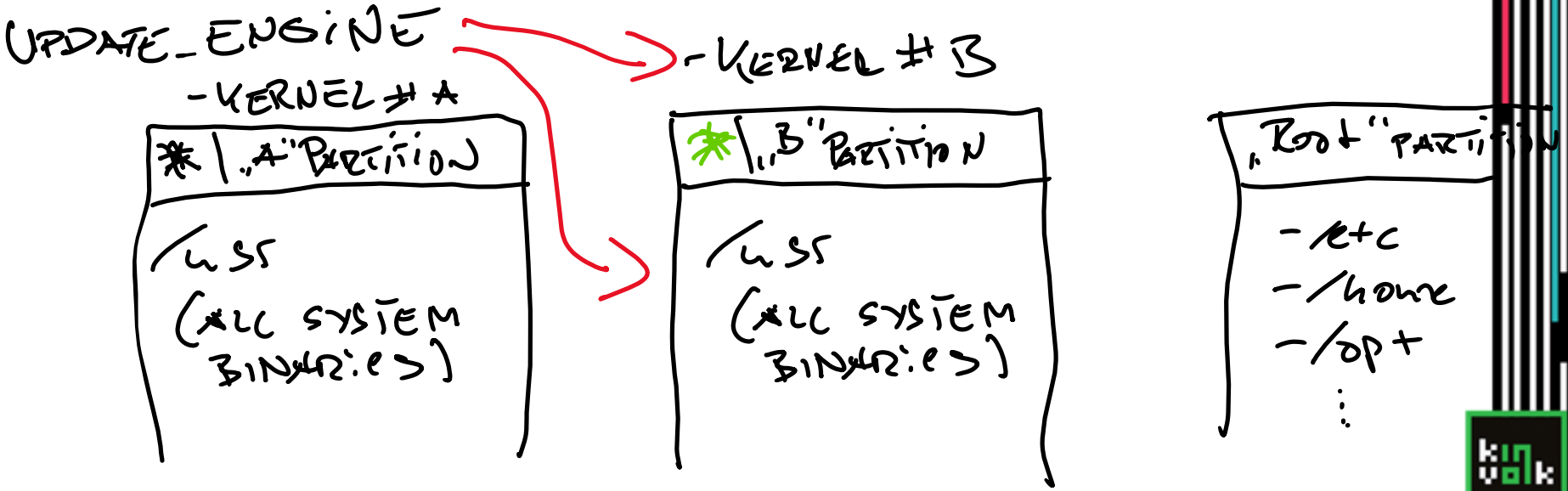
“Root” partition is r/w, for configs + user-supplied changes

“Usr” partition includes all binaries, is R/O.



Make Mistakes forgivable

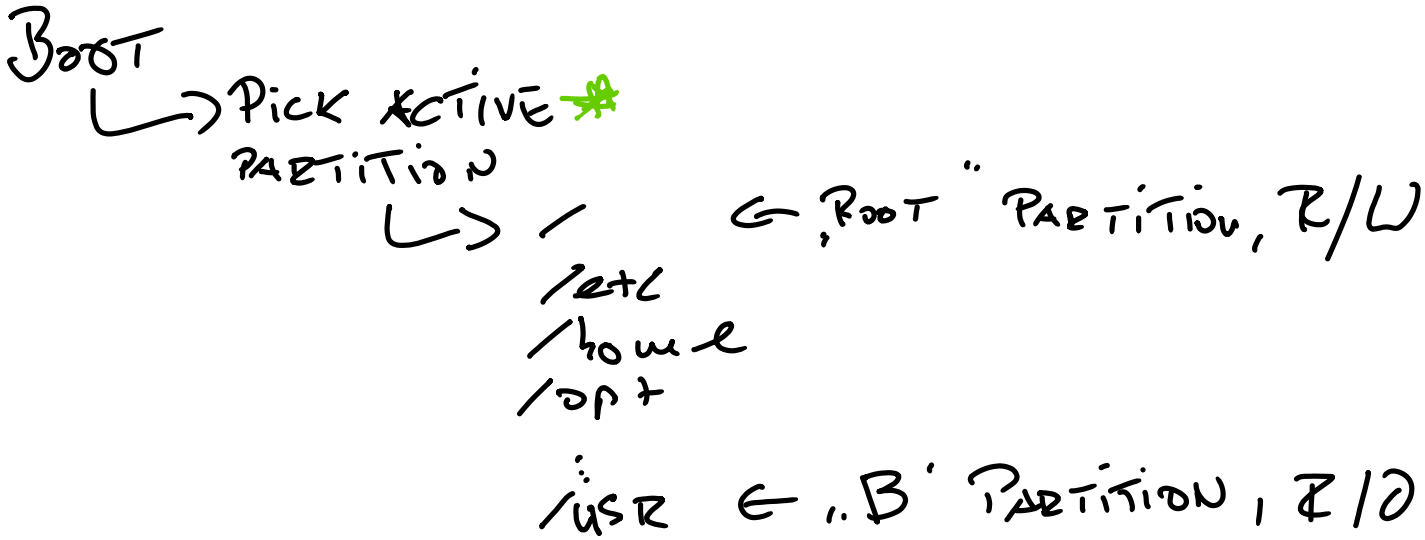
Updates are downloaded to “B” partition (in the background)



Make Mistakes forgivable

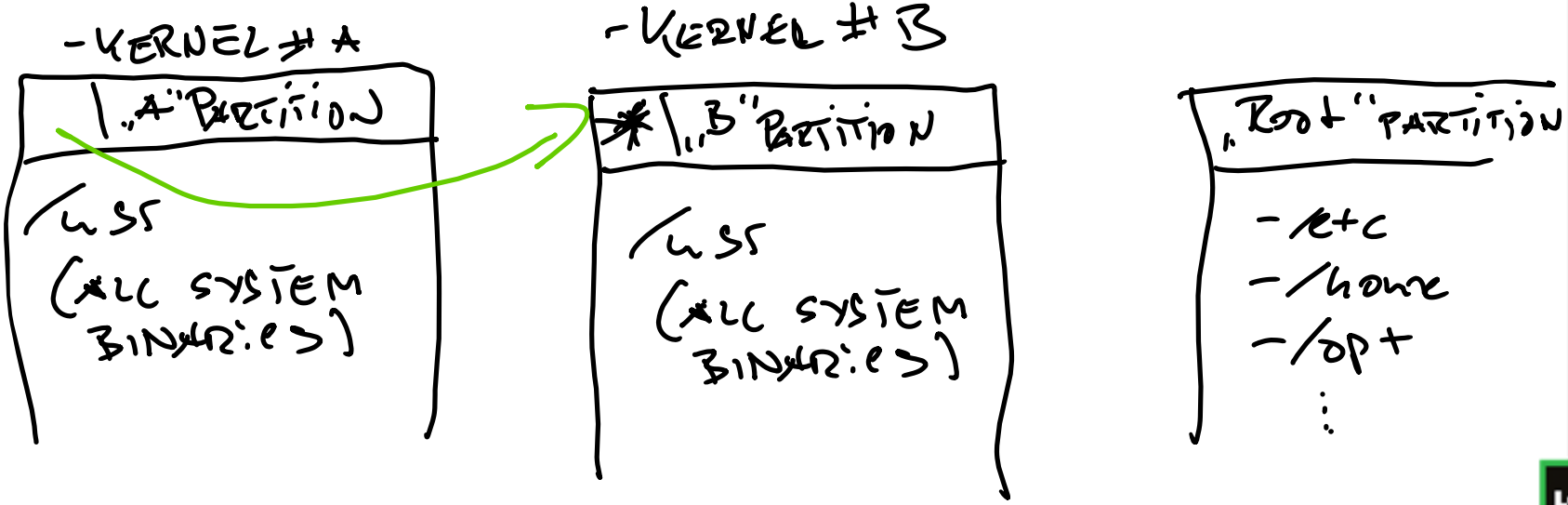
Applying the update needs a reboot.

The reboot will boot into the updated partition ***exactly once***.



Make Mistakes forgivable

Only after successful boot + some wait time, B is activated persistently



On error, the system just reboots (into “A” which is still active).

Make Mistakes forgivable

Atomic updates by use of a separate partition

- Updates downloaded, staged in background, during regular operation
- Minimal downtime (i.e. no “safe mode” to install packages)
- Roll-back via a simple reboot

Package-focused distros: mount the inactive partition

- on update, chroot + install updates to inactive partition
- copy / overwrite / migrate user configs
- reconfigure bootloader (savedefault / grub2-once)

Using changesets, we can roll forward and roll back our OS like an application

Scale out

We discussed

- changesets
- canaries
- and atomic updates / roll-backs

Effectively turning the OS into just another application.

But how do we automate all this?

Scale out

Glue:

Check for updated changesets, download and install
reboot , or signal reboot request

Infrastructure:

Image build infrastructure
Test infrastructure + test harness
Image / update server (for changesets)
and /or package cache

Scale out

Cluster-wide orchestration via reboot daemon

- Drain nodes
- Only reboot one node at a time

Implementations

- Kured – Kubernetes; daemonset, acts on a single file being present
- FLUO – Kubernetes; daemonset + operator for update_engine
- locksmith – etcd, for custom clusters w/o Kubernetes

Thank you 🙏

Read the docs - https://www.flatcar.org/docs/latest/

Chat with us – https://app.element.io/#/room/#flatcar:matrix.org
Contribute – https://github.com/flatcar-linux/Flatcar
Join our monthly calls - https://github.com/flatcar-linux/Flatcar/#monthly-community-meeting-and-release-planning