Regular Releases Are Wrong

Roll For Your Life!

Richard Brown

MicroOS Release Engineer

rbrown@opensuse.org

sysrich on Freenode.net

@sysrich

Who am I?

openSUSE contributor since it began
SUSE employee since 2013

Passionate advocate of rolling releases

Linux Distribution Engineer in Future Technology Team
focusing on two rolling distributions

openSUSE MicroOS – Single Purpose Self Administering OS
openSUSE Kubic – MicroOS with Kubernetes & Containers

Disclaimers

- This is an opinionated presentation
- I hold very strong opinions on this topic
- They are my views and not that of my employer or any Project/Group I am/have been affiliated with
- It’s OK if you do not agree with my view

Upstreams Change Fast

Glibc – New version every 6 months
Linux Kernel – New version every 3 months
Kubernetes – New version every 3 months
SaltStack – New version every 3-6 months
Uyuni – New version every 1-2 months
Ceph – New version every 1-2 months
Podman/skopeo/buildah – New version all the time
Cloud Foundry – New versions all the time

Short Upstream Support

Kernel Stable Releases – 4 months
Kernel LTS Releases – 6-7 years
Kubernetes – 1 year (before v1.19, was 9 months)
SaltStack – 1.5 years (feature freeze after 6 months)
Ceph – 2 years

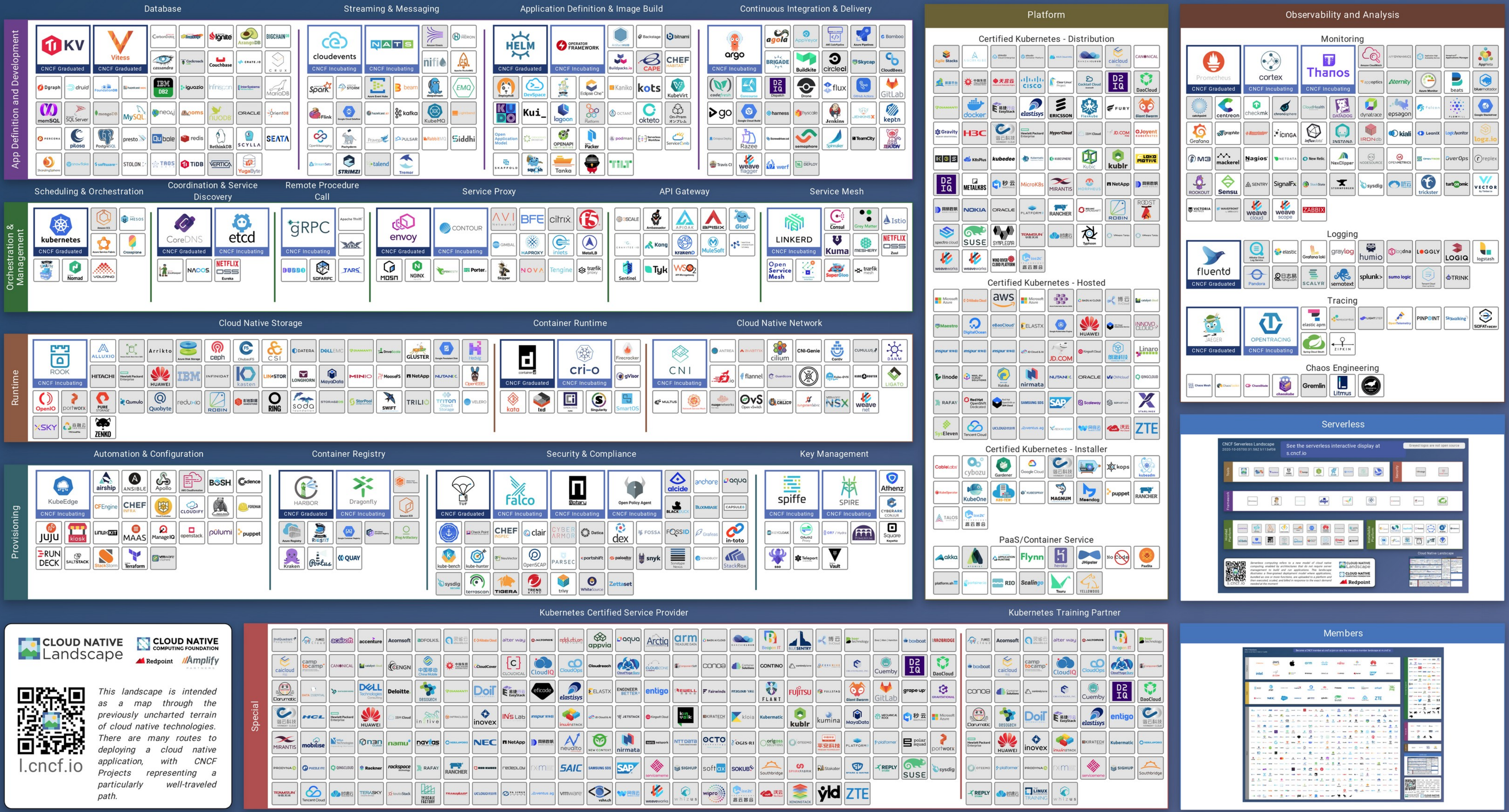
We Have More Upstreams

- Leap
- Tumbleweed
- Uyuni
- MicroOS
- Kubic

CNCF Cloud Native Landscape
2020-10-05T00:31:58Z b113ef08

Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io

Greyed logos are not open source



App Definition and Development
Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
Orchestration & Management
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Runtime
Cloud Native Storage
Container Runtime
Cloud Native Network
Provisioning
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Logging
Tracing
Chaos Engineering
Serverless
Members
Special
Kubernetes Certified Service Provider
Kubernetes Training Partner
CNCF Graduated
CNCF Incubating

CLOUD NATIVE Landscape

CLOUD NATIVE COMPUTING FOUNDATION

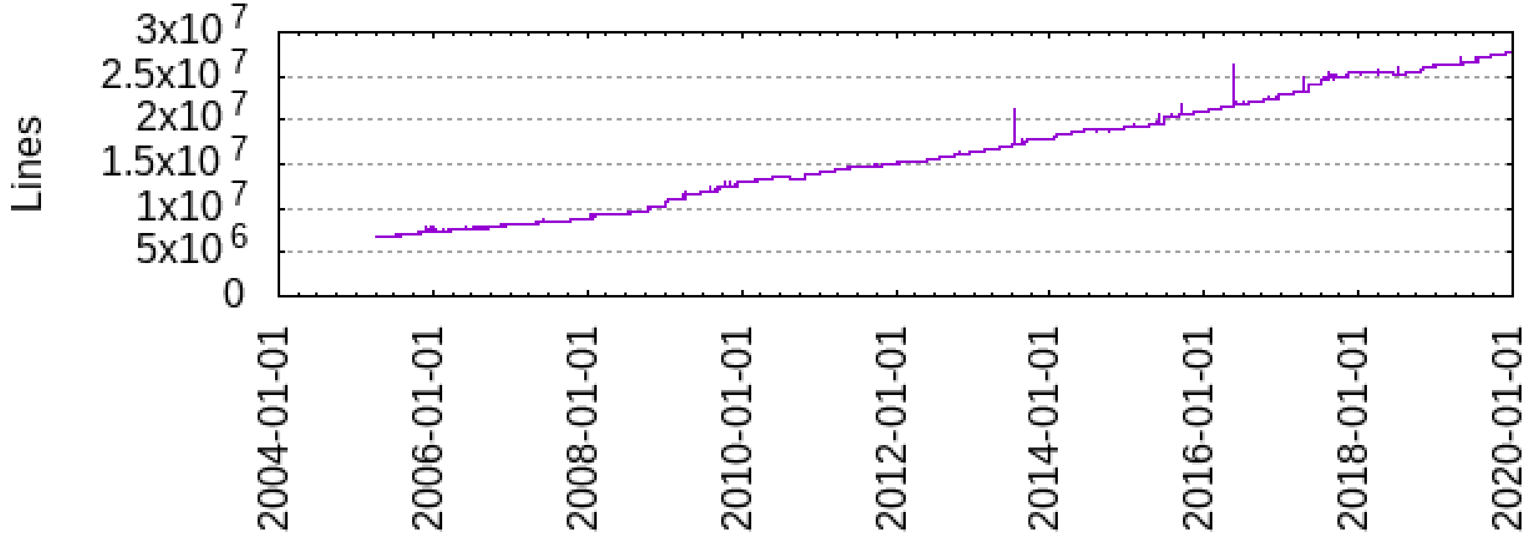
Redpoint

Amplify PARTNERS

l.cncf.io

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

We Have Larger Upstreams

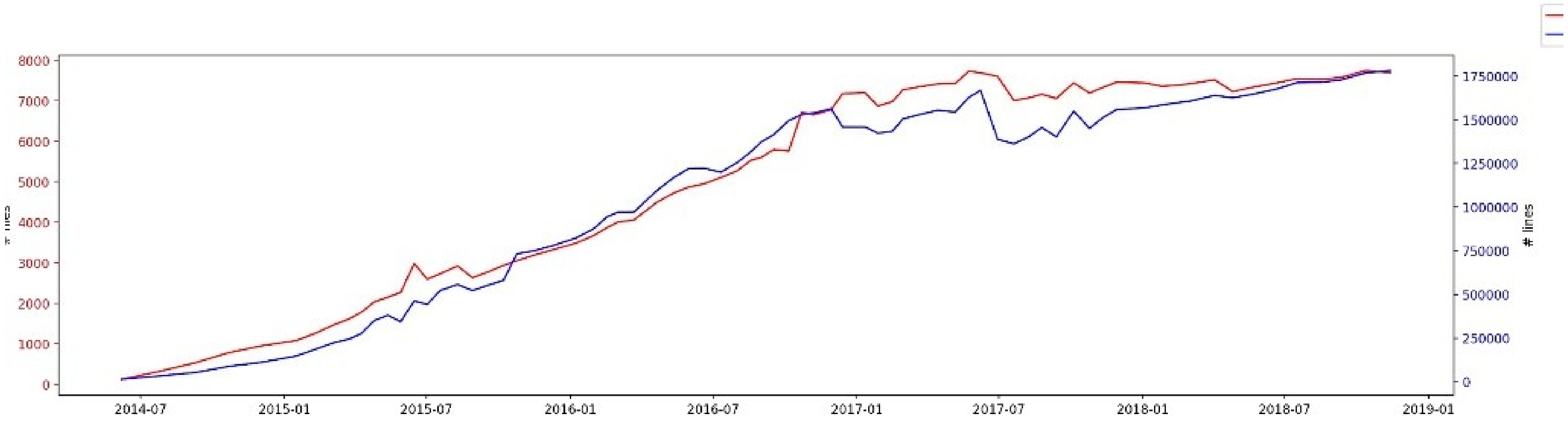


Source: phoronix.com “The Linux Kernel Enters 2020 At 27.8 Million Lines In Git But With Less Developers For 2019”

Kubernetes Growing Also

Number of files / LoC over time

source{d}



Stabilizing at > 1.75 Millions Line of Code, just under 8K files

Regular Releases Mean Well

- “How to condense thousands of moving packages from thousands of different people into something which other people can use?”

Change is Dangerous

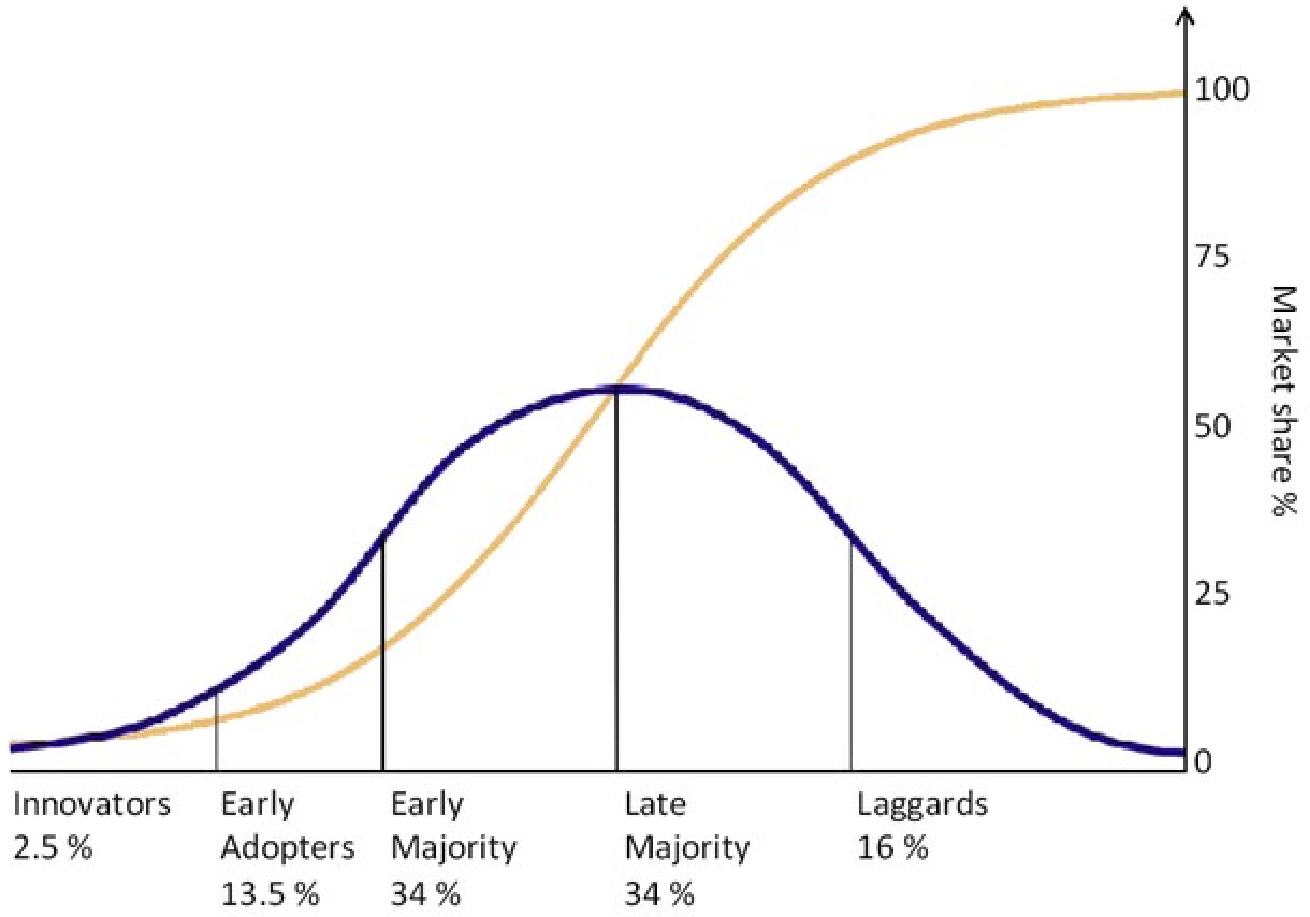
- You can’t break anything if you don’t change it
- But even regular releases need a LOT of change
- Minimal changes are safer, right?

Slow Is Not Safer

Slow Doesn’t Work

- SUSE Linux Enterprise 15 has shipped over 13780 changed packages so far, including over 2791 package version changes across a codebase of ~3500 packages.
- That is NOT including the 10000+ packages from openSUSE PackageHub.

Slow Isn't Where We Want To Be



Slow Isn’t More Sustainable

Every upstream is getting bigger

We are getting more upstreams

Every divergence/freeze from upstream produces more work for us

We aren’t growing fast enough

Slow Undermines Open Source

Linus’s Law states “given enough eyeballs, all bugs are shallow.”

Regular Releases throw most of the eyes away

Doesn’t that make more of our bugs deep?

Partially Slow is Totally Broken

Tumbleweed was originally started by Greg Kroah-Hartman in 2010

Rolling base atop regular openSUSE releases

Focus on Kernel, KDE, GNOME and some desktop Apps

Would overwrite/supersede packages from regular release

”Reset-to-zero” every regular release

Partially Slow is Totally Broken

“Partially Rolling” was painful for both users and engineers

Constant breakage over the growing chasm between the ‘stable’ base and rolling top

Ad-hoc tinkering/superseding of the ‘stable’ base stops it being stable

“Reset-to-zero” rebase to a new stable base every 8 months was brutally disruptive for all users

Build Together, Test Together

Modern Tumbleweed evolved out of efforts to stabilise openSUSE:Factory

Build all packages together, rebuild dependency tree as new/updated packages added (leveraging OBS)

Test all relevant use cases, focusing on the way users use them (leveraging openQA, LTP, and various release bots)

Sustainable engineered and usable for it’s target audience for 6 years running

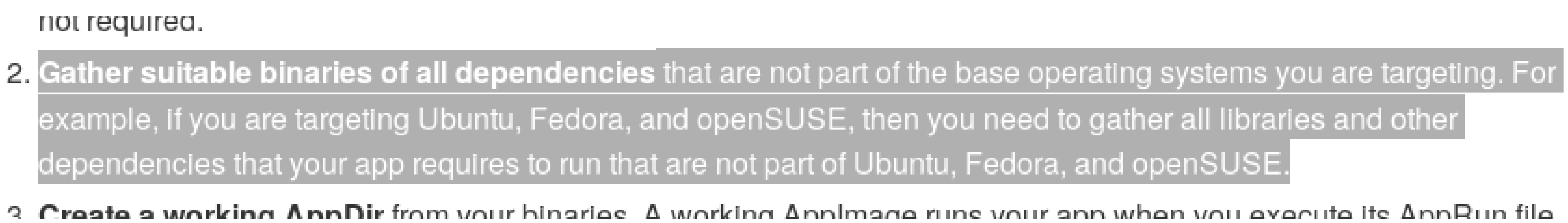
Containers Aren’t Magic

eg. AppImage

- Portable Software format, containing binaries and required libraries in an executable archive
- Promises “Linux apps that run anywhere”
- Used by various upstreams to distribute their own binaries

Containers Aren’t Magic

AppImages don’t run everywhere

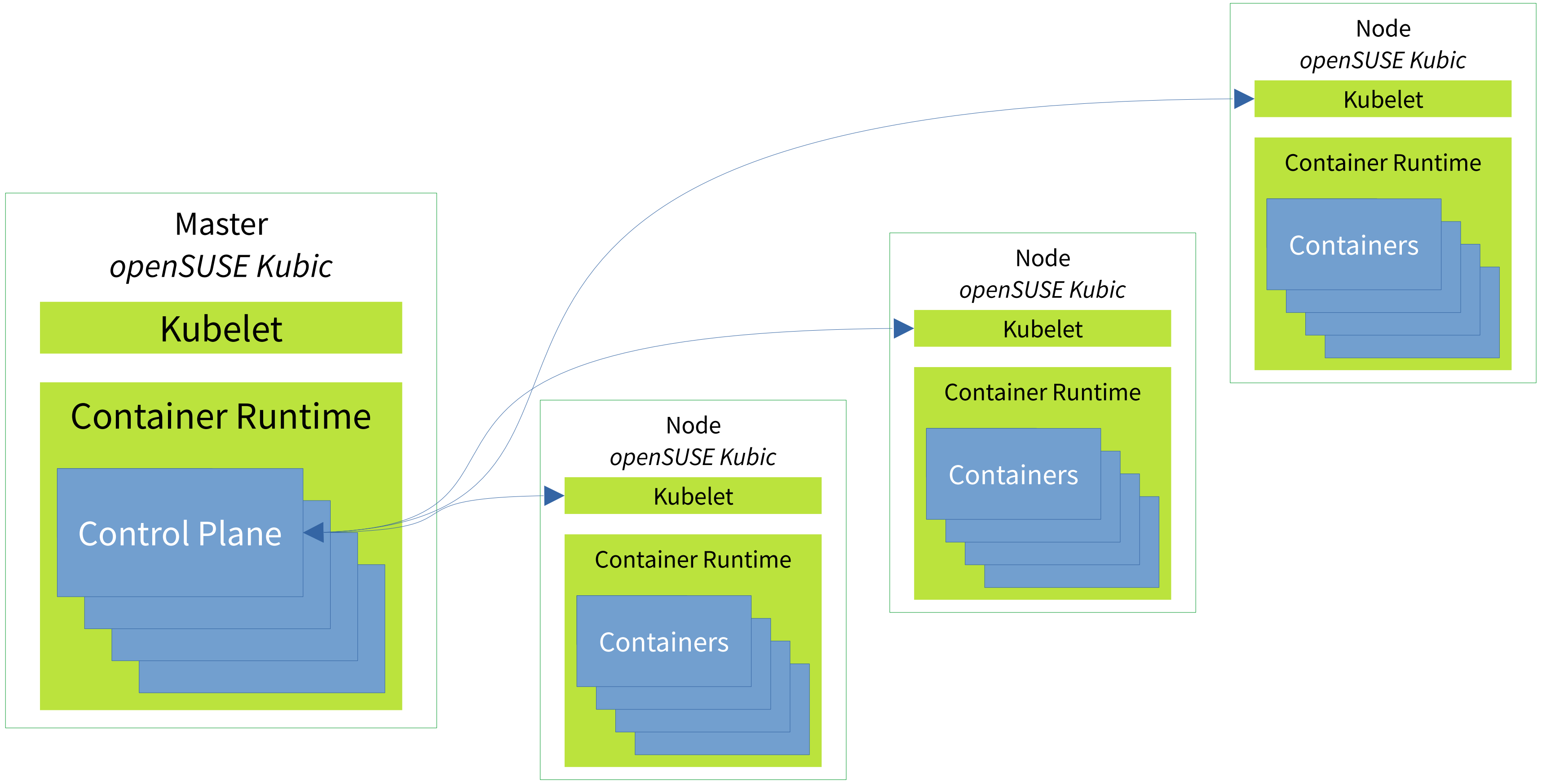


not required.

2. **Gather suitable binaries of all dependencies** that are not part of the base operating systems you are targeting. For example, if you are targeting Ubuntu, Fedora, and openSUSE, then you need to gather all libraries and other dependencies that your app requires to run that are not part of Ubuntu, Fedora, and openSUSE.

3. **Create a working AppDir** from your binaries. A working AppImage runs your app when you execute its AppRun file

Containers Still Depend On Their Hosts



Lesson Learned – Roll With Containers

Distribution neutral/system isolated containers is a myth

Building, testing & releasing containers in alignment with traditional RPM packages is essential

Containers can impose ‘unfair’ dependencies on the host OS that traditional packaging cannot model or resolve

Well orchestrated building, testing & releasing can enable for some ‘drift’ between containers and host

The Rolling Engineering Axiom

"In order to move ANYTHING quickly, you need to be able to move EVERYTHING quickly"

Rolling Has Real Benefits

Benefit as much as possible from upstreams, both for distro builders and our users

More easily contribute back to upstreams

Reduce double-work, retesting, and backports requiring backports requiring backports...

But Change Is Still Scary

Not everyone wants to move at the same speed as every other

Not every upstream is aligned with their related and dependent codebases

MPH
km/h

Unsafe at Any Speed?

Full speed ahead is not the only speed

Tumbleweed has proven processes for releasing at the pace of upstream/contributions

Maybe we need other rolling releases that strike a better balance between keeping up with upstreams and not imposing too much change on our users

Change Less – Use MicroOS

openSUSE MicroOS is a predictable & immutable. It cannot be altered during runtime.

MicroOS is reliable with automated updates and automated recover from faulty updates.

MicroOS is small with only what is needed to run it's "one job". Applications/Services are expected to be Containerised or Sandboxed.

My MicroOS Life

Nextcloud

jekyll

emulation station

MINECRAFT

openSUSE

Is Everyone Doing Everything Wrong?

- RPM packaging is awesome for building, but painful for users
- Why do we still make users deal with packages?
- What would a ‘Container-first’ Linux look like?

Containers Only - The Next Step?

- Developers & Sysadmins don’t want to care for packages
- They want their services and the languages/libraries they care about
- Containers give us a chance to redefine how we offer our stuff to them while reduce what we need to care about for the OS

COOL – Container Only Operating Layer

- Why not have something like MicroOS but with a large predefined library of containers ready for use?
- Runtimes – Language libs+toolchain bundled together (eg. Python, Golang, etc)
- Apps – Ready to go service containers (eg. Apache, MariaDB, DHCP, etc)

COOL Apps

Updating services would be as easy as

- podman pull cool/app/apache:latest

Building your own service based on our containers would be as simple as

- buildah from cool/app/apache:latest

COOL Runtimes

Any user could pull a runtime as easily as

- podman pull cool/runtime/python:3

Any developer could base their container on a runtime as easily as

- buildah pull cool/runtime/golang:latest

COOL Build Ideas

- COOL Apps & Runtimes could be built using OBS Subprojects
  - COOL:Runtimes:Python:$VER
  - COOL:Runtimes:Python:$VER:Apps
- All built together using OBS but able to diverge a little because we all know it will only be delivered via containers
- Containers then tested & released to registry.opensuse.org using a much simpler structure for users

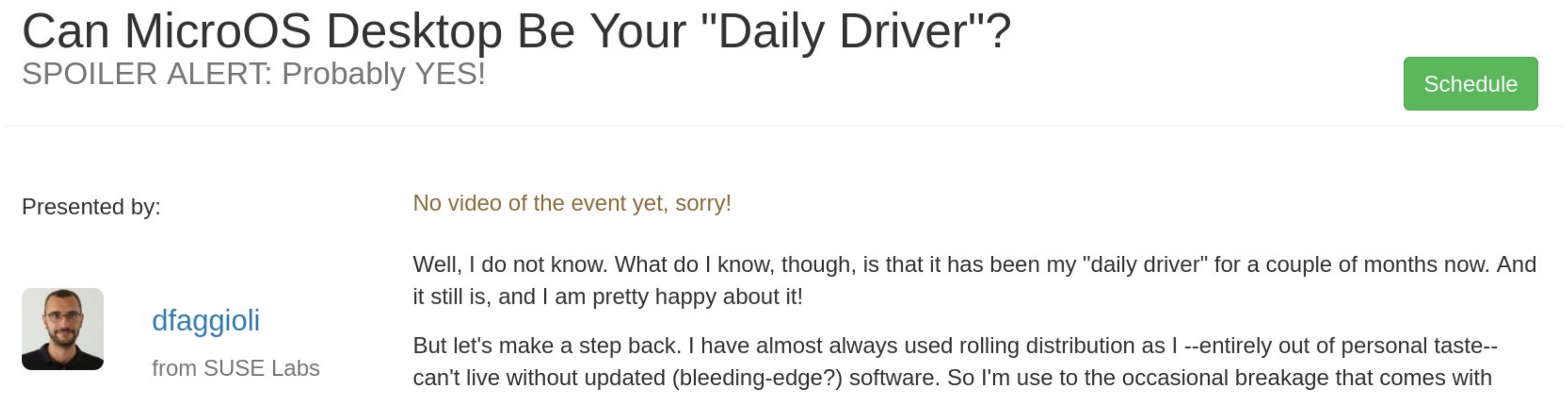
What about the Desktop?

MicroOS Desktop is MicroOS where the ”one job” is running as a Desktop.

MicroOS Desktop provides only a minimal base system with a Desktop Environment and Basic Configuration Tools ONLY.

All Applications, Browsers, etc are provided by FlatPaks from FlatHub.

Go To This Talk Too



Can MicroOS Desktop Be Your "Daily Driver"?

SPOILER ALERT: Probably YES!

Schedule

Presented by:

dfaggioli

from SUSE Labs

No video of the event yet, sorry!

Well, I do not know. What do I know, though, is that it has been my "daily driver" for a couple of months now. And it still is, and I am pretty happy about it!

But let's make a step back. I have almost always used rolling distribution as I --entirely out of personal taste-- can't live without updated (bleeding-edge?) software. So I'm use to the occasional breakage that comes with

Date: 2020 October 17 - 12:00

Duration: 30 min

Room: Room 1

openSUSE®