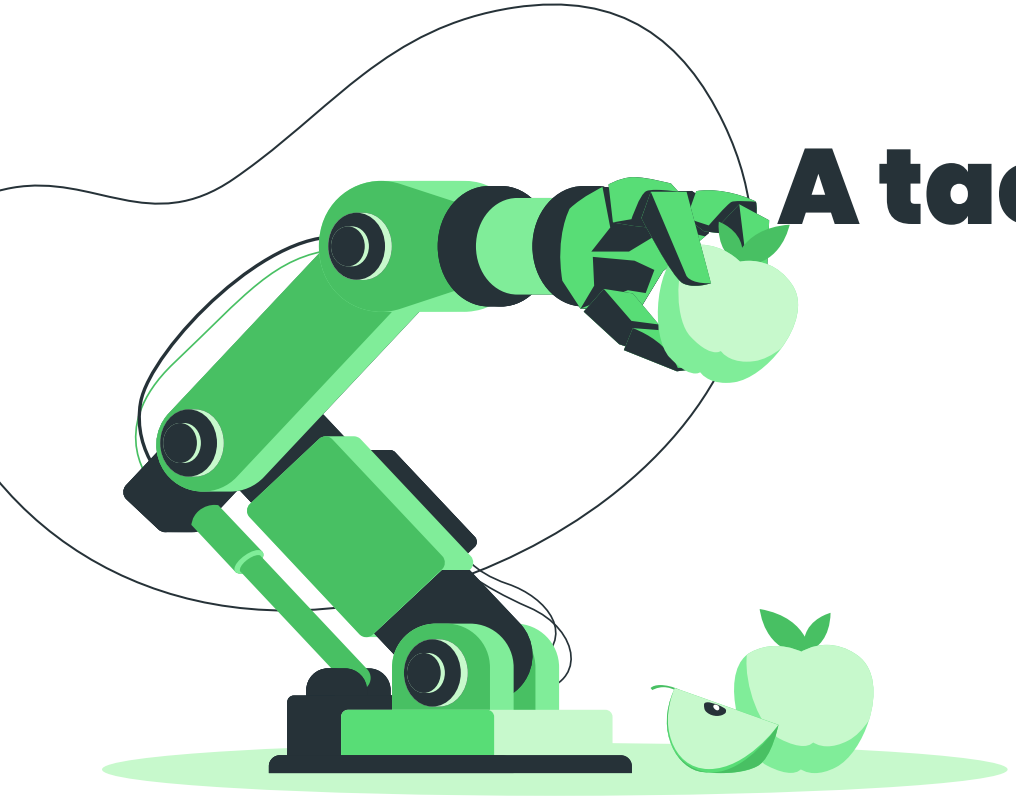
Do we really need Kubernetes? A tactical guide to Cloud Native



Max Körbächer - Founder &
Cloud Native Advisor

REPLY
LIQUID

Max Körbächer

Founder & Cloud Native Advisor @ Liquid Reply

Wir fokussieren uns auf Platform Engineering, Internal Developer Platforms & Cloud Native Engineering

- CNCF TAG Environmental Sustainability Initiator & Co-Chair
- CNCF Ambassador
- LF Europe Advisory Board
- Contributed 3y to the Kubernetes release team

THE LINUX FOUNDATION

TAG ENVIRONMENTAL SUSTAINABILITY

maxkoerbaecher

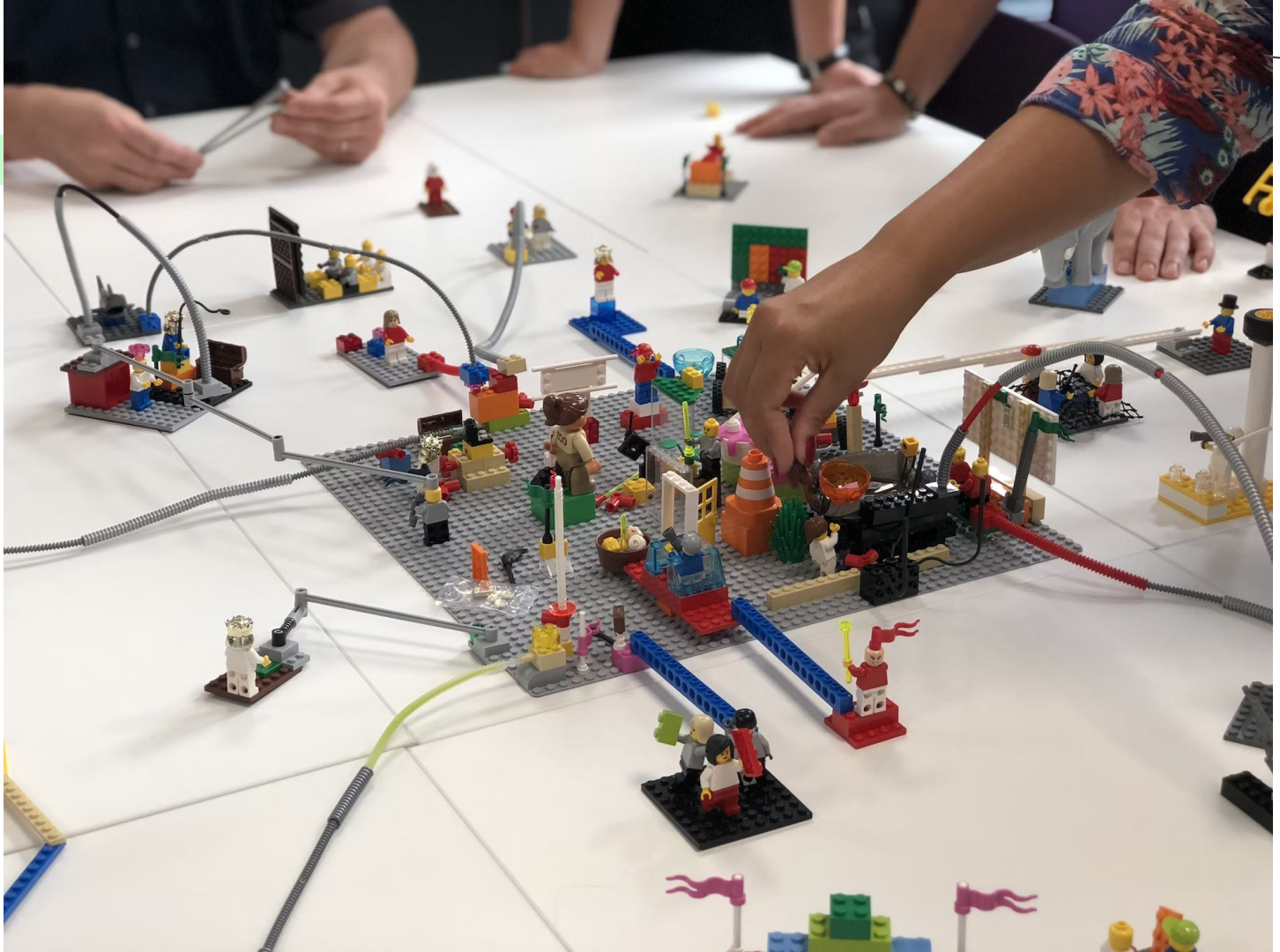
mkoerbi


Expectation
Reality
OpenTelemetry


WE USE
KUBERNETES NOW
BECAUSE IT SIMPLIFIES
OUR OPERATIONS, RIGHT?
IT SIMPLIFIES
OUR OPERATIONS, RIGHT?
imgflip.com

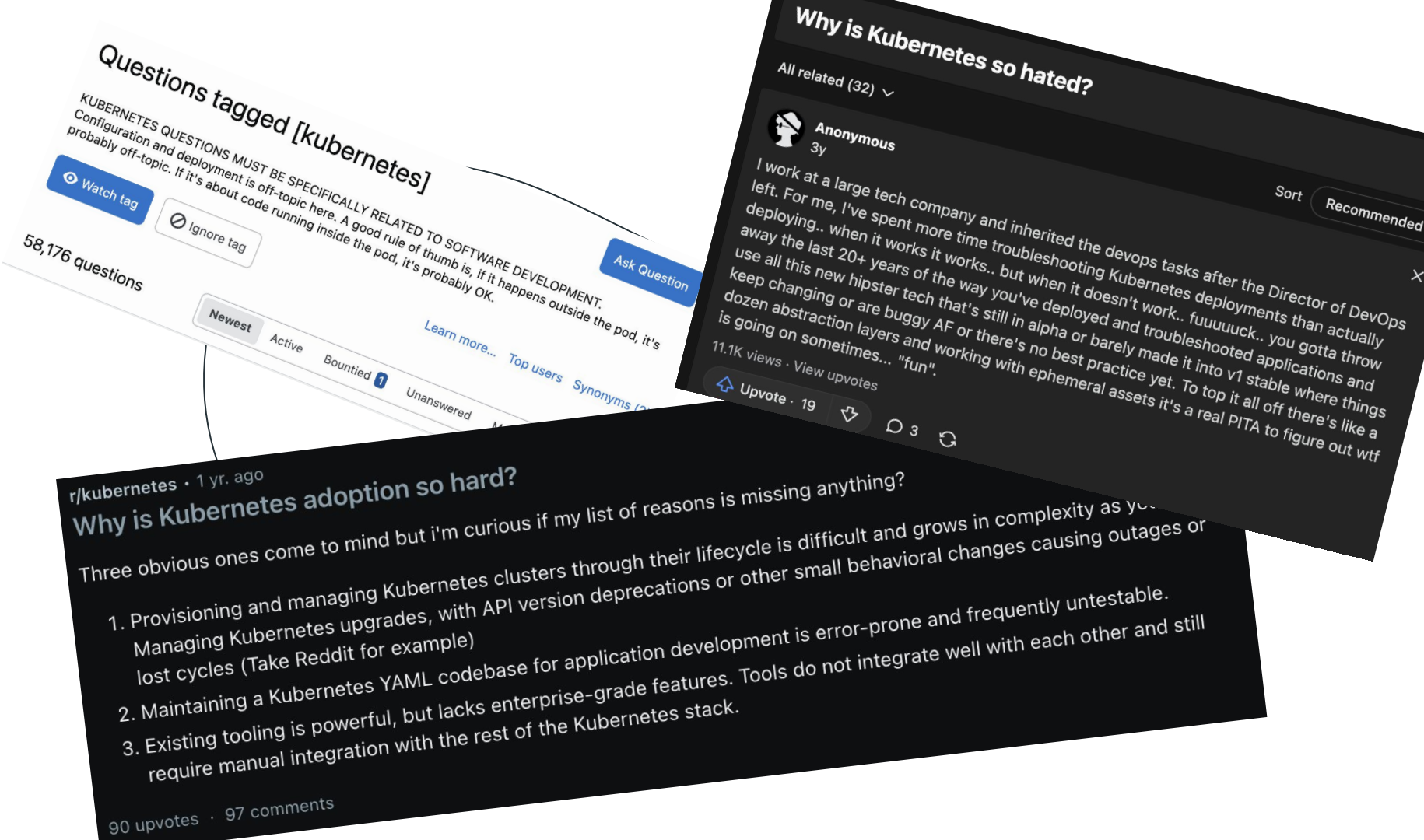

MAERSK


THIS IS NOT (ONLY)
KUBERNETES




MAERSK
KUBERNETES LINE
MAERSK MC-KINNEY MØLLER

Why does K8s Adoption Fail?

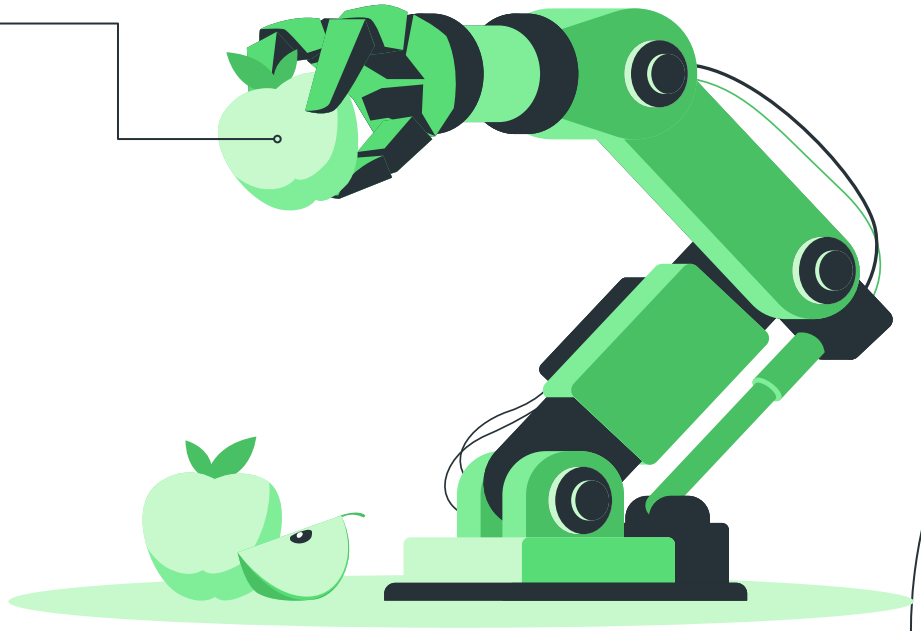

Questions tagged [kubernetes]
KUBERNETES QUESTIONS MUST BE SPECIFICALLY RELATED TO SOFTWARE DEVELOPMENT. Configuration and deployment is off-topic here. A good rule of thumb is, if it happens outside the pod, it's probably off-topic. If it's about code running inside the pod, it's probably OK.
Watch tag
Ignore tag
Ask Question
58,176 questions
Newest
Active
Bountied 1
Unanswered
Learn more...
Top users
Synonyms
Why is Kubernetes so hated?
All related (32)
Anonymous
3y
I work at a large tech company and inherited the devops tasks after the Director of DevOps left. For me, I've spent more time troubleshooting Kubernetes deployments than actually deploying.. when it works it works.. but when it doesn't work.. fuuuuuck.. you gotta throw away the last 20+ years of the way you've deployed and troubleshooted applications and use all this new hipster tech that's still in alpha or barely made it into v1 stable where things keep changing or are buggy AF or there's no best practice yet. To top it all off there's like a dozen abstraction layers and working with ephemeral assets it's a real PITA to figure out wtf is going on sometimes... "fun".
11.1K views · View upvotes
Upvote · 19
3
Sort
Recommended
r/kubernetes · 1 yr. ago
Why is Kubernetes adoption so hard?
Three obvious ones come to mind but i'm curious if my list of reasons is missing anything?
1. Provisioning and managing Kubernetes clusters through their lifecycle is difficult and grows in complexity as yo
Managing Kubernetes upgrades, with API version deprecations or other small behavioral changes causing outages or lost cycles (Take Reddit for example)
2. Maintaining a Kubernetes YAML codebase for application development is error-prone and frequently untestable.
3. Existing tooling is powerful, but lacks enterprise-grade features. Tools do not integrate well with each other and still require manual integration with the rest of the Kubernetes stack.
90 upvotes · 97 comments

Reasons

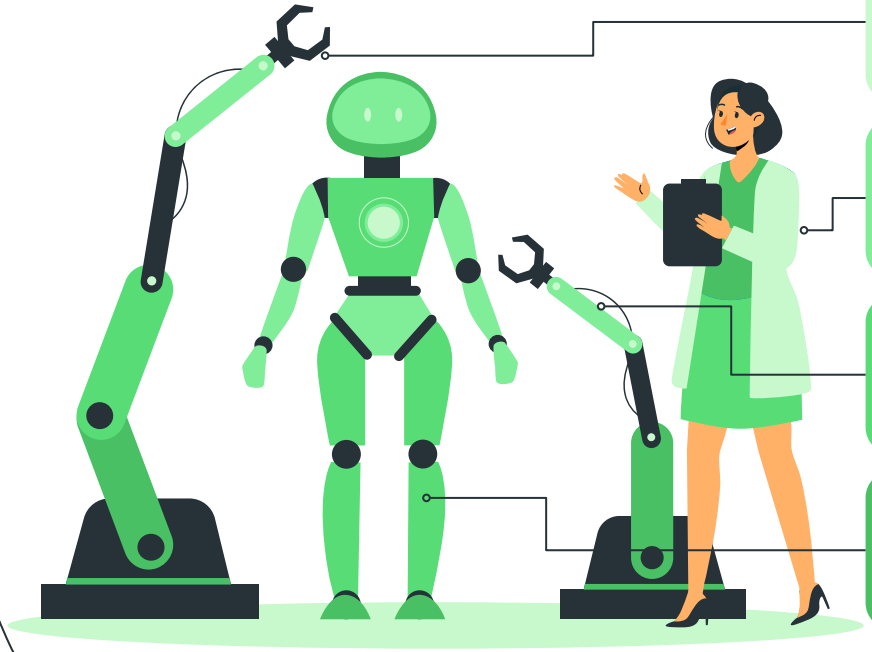
Wrong Roles

1

DevOps? Bullshit! Most Orgs are heavily wrong



People have the wrong roles & expectations



CloudFoundry
3.295 open questions on StackOverflow

OpenStack
2.597 open questions

AWS
159.009 open questions

Vsphere+Hypervisor+VMware
4.032 open questions

*Kubernetes is at 58.176 open questions

Misalignment

There is a huge misalignment between Dev and Ops

Developer

- Try to run K8s to run software
- Need to understand often everything for just a lil code
- Even don’t get down to the relevant concepts that supports them

Ops

- Actually like the way how infrastructure was done
- Are not used to be in a close exchange with Devs
- Also have to learn how all works

You don’t ask a watchmaker to build global watch factories

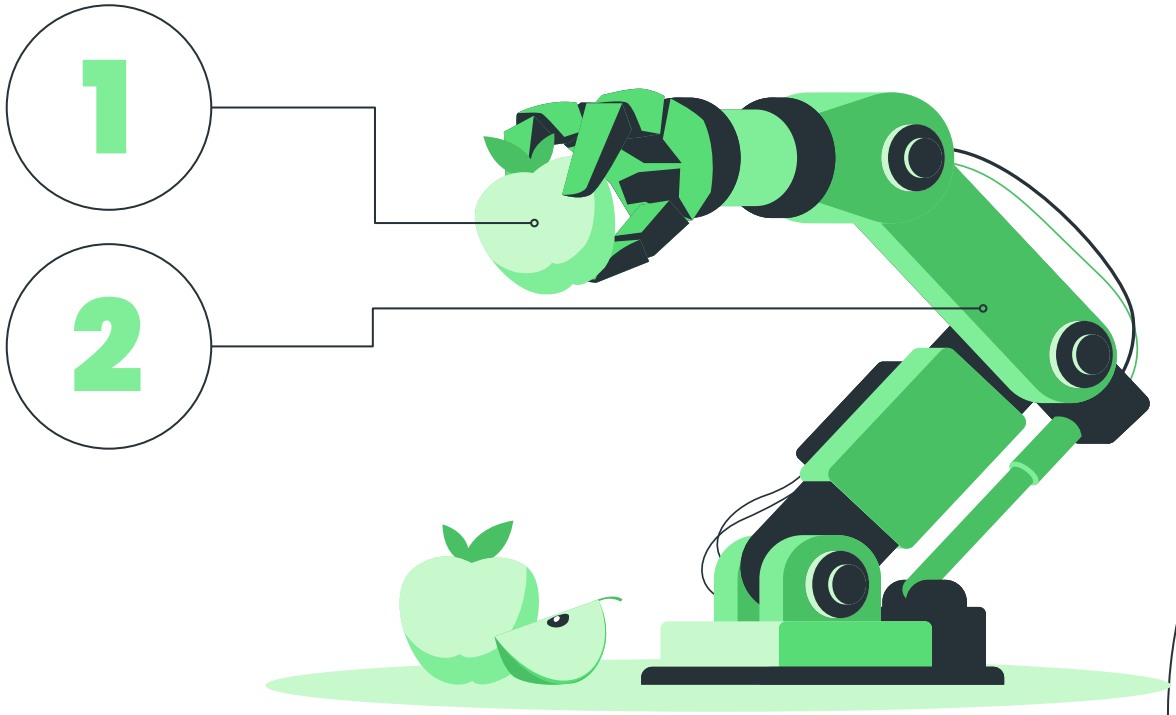
Reasons

Wrong Roles

DevOps? Bullshit! Most Orgs are heavily wrong

Poor Support

No time/budget for training and wrong training



Trainings ... or just time to learn

- No time
- No budget
- Fast, fast
- “We don’t want too complicated training”

46%

say lack of training is the biggest challenge facing organizations that have not started, or are just beginning, their cloud native journey

*CNCF Report 2023

- Doing the wrong thing
- Having no idea of Dev/Security/Observability/Ops with Kubernetes & Containers
- Raising frustration

Reasons

1

Wrong Roles

DevOps? Bullshit! Most Orgs are heavily wrong

2

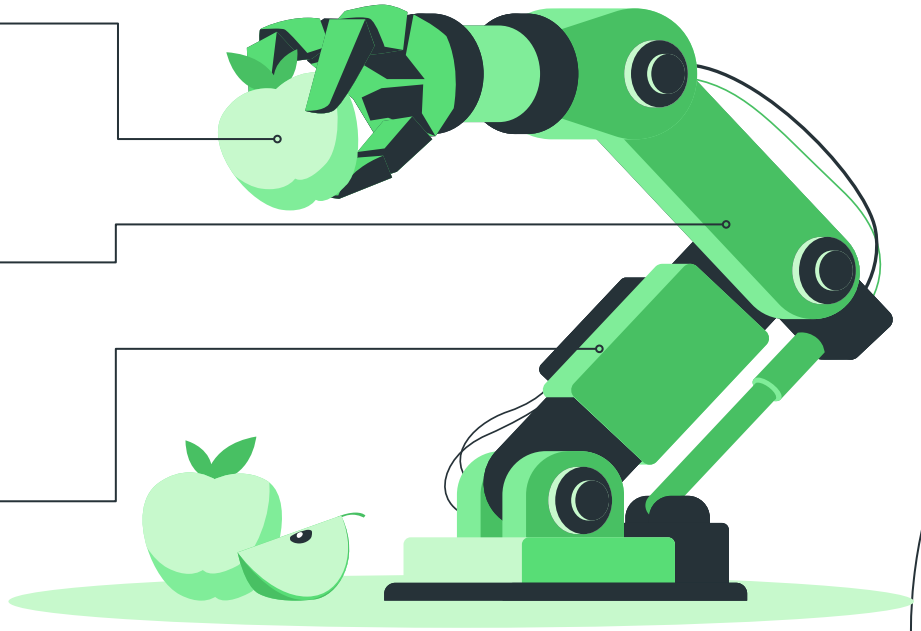
Poor Support

No time/budget for training and wrong training

3

Over Engineering

Misunderstanding of K8s



Over Engineering Kubernetes

Because of the common rumors “Kubernetes is complex”, “add up additional costs” or “is unmanageable”, teams tend to over engineering Kubernetes.

This leads to exactly the named problem causing this impression.

KISS Kubernetes!

You and your K8s

But what about?

Operational Overhead

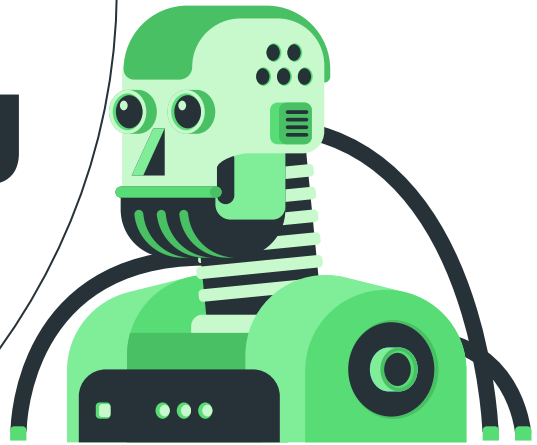
Security

...

Complexity

When K8s is wrong

No one ever says Kubernetes is the solution for everything!



When you better don’t run on K8s

COTS

Commercial software often lacks support, in many ways

Large Monoliths

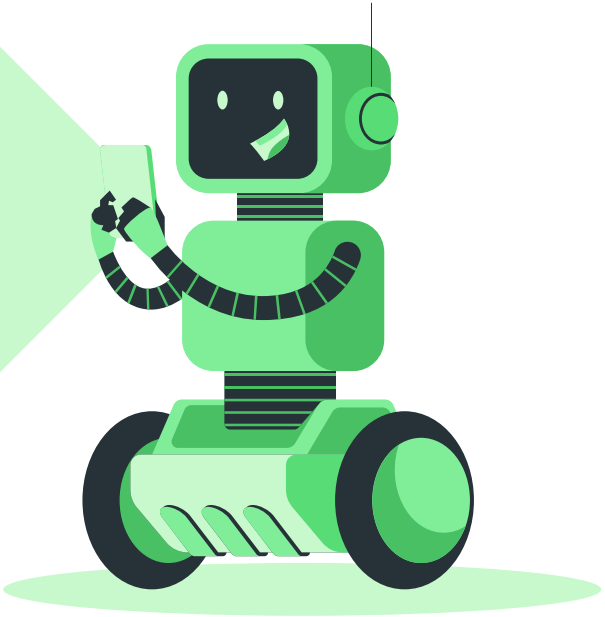
They could run on it, it’s just a bad idea.

Memes

Your team shares more memes on how bad K8s is than they actually work*

You run Container…

…elsewhere and you are happy. Why to change it?



*it’s a mindset thing, very difficult to change

More relevant: Stop thinking Small

Kubernetes

- Stop thinking K8s is the solution
- K8s alone never brings you any value

Platforms

- Think about platforms/IDPs with a clear target & focus
- Only a holistic approach bring you the value you are searching for

How do I find out when K8s is right and what do I need?

Topics to answer to find the right puzzle piece to your situation

1: Do you introduce it for one app or the whole org?

2: How does your common application architecture look like?

3: What are your expectations for scalability?

4: Does your team have experience or capacity to learn?

5: Who will run the environment?

Simple web service, a hand full of people

One App

Hundreds of devs & services?

1: Do you introduce it for one app or the whole org?

Whole Org

Single initiative, minor app

Better use a Cloud service

- VMs
- App Engines
- Container Services
- Serverless

2: How does your common application architecture look like?

- **Micro/ -Monolith/ Distributed** ✓
  - Is fault resistant, can run multiple times accessing the same data source ✓
  - Requires long preheat time for startup ✗
- **Large Monolith** ✗
  - Needs stability but can run simultaneously ✓

Monoliths, Microservices & co

Monoliths can run in Kubernetes but

VMs or dedicated resources usually suits beter to the requirements.

Is Serverless the answer?

It is strong when you just need to get started.

Maybe for microservices it is good to, but I often see a de-migration to a container based solution.

From 0 to 100 and back in a sec

Event, metrics, trigger based scaling

3: What are your expectations for scalability?

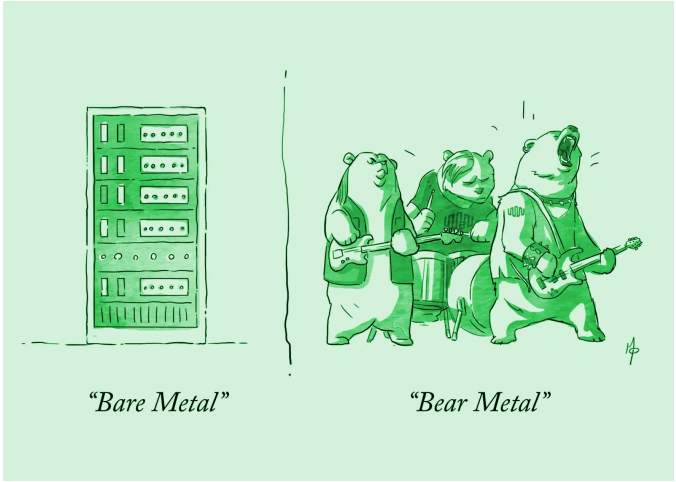
Can scale up and done in a second, also nodes etc.

Doesn't need more than a container

Scalability isn’t anymore a factor

The question is: How fast should it scale?

- Extremely fast: Serverless or Wasm based services
- Very fast: Container Services & Kubernetes
- Fast: VMs at some service provider
- Moderate: VMs at large cloud provider
- I really don’t care: Bare metal

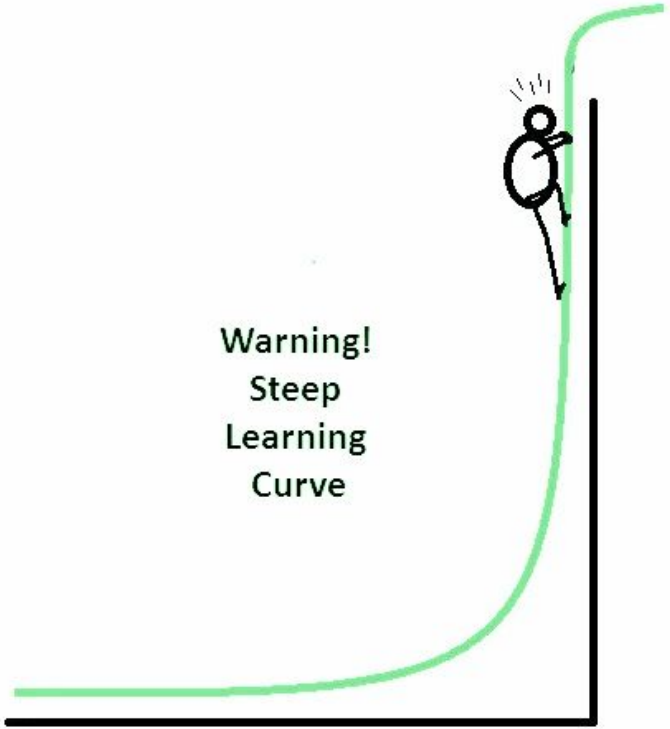


YES

4: Does your team have experience or capacity to learn?

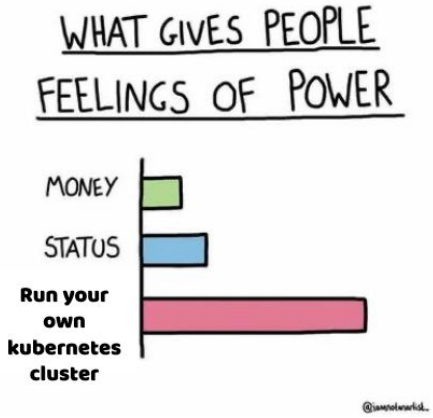
NO

Kubernetes Learning Curve is hard



The learning curve of K8s is brutal

- After you mastered the basics, there will come so much more
- The high flexibility and extensibility, the hundreds of FOSS solutions and the changes every 4 month can be a killer
- We aim today for a Platform not a K8s



5: Who will run the environment?

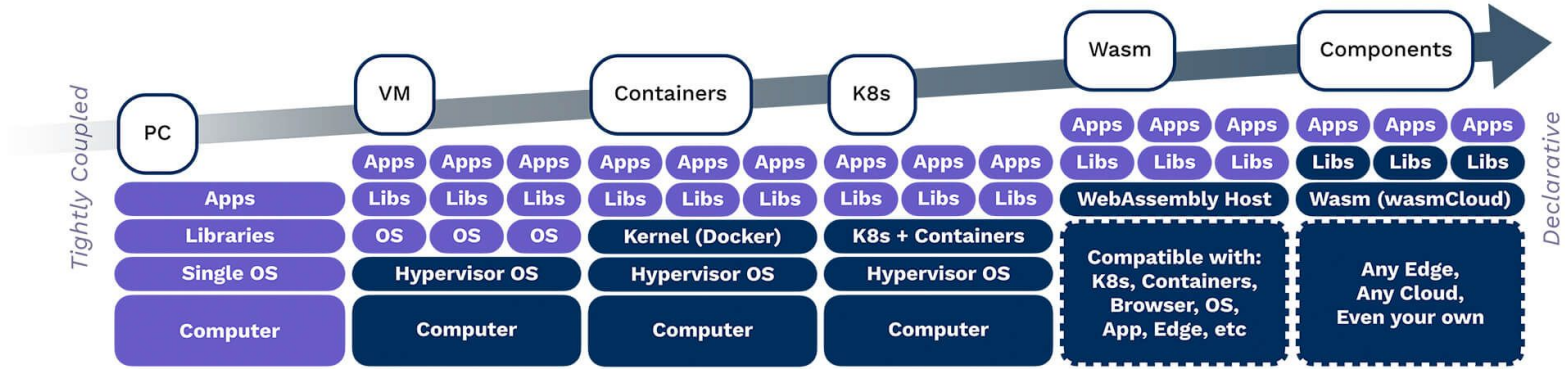
- **A dedicated team** ✓
  - The team is responsible for feature development ✓
- **Someone who wasn't fast enough running away** ✗
  - The team just get a finished platform to operate ✗

The Future

Cloud is a commodity

Yet, many organizations fail in its adaption.

Evolution of “runtimes”



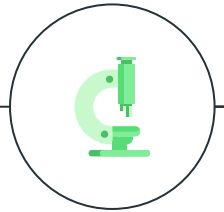
FORMAT	PC	CLOUD	CONTAINER	K8S	WASM	COSMONIC
EXECUTION	*Image (Datacenter)*	*VM (Public Cloud)*	*Container (Docker)*	*Containers (K8s / Cloud)*	*WASM (Everywhere)*	*Distributed WASM (Everywhere)*
Dev Responsibility	Full	OS, App, Lib	App, Lib	App, Lib	Wasm	Business Logic
Abstraction	-	CPU	Linux Kernel	K8s	Secure Sandbox	Sandbox + Capabilities
Compatibility	All	Most	Most	Most	Most	Most
Size	Large	Med	Small	Small	Tiny	Minuscule
Portability	-	Low	Med (CPU, Linux)	Med (CPU, Linux)	High	Highest
Security	System	OS	Process Boundary	Process Boundary	Capability	Actor
Location	On Prem & Co-location	Proprietary Cloud & Edge	Dev, Edge, Cloud, K8s	Dev, Edge, Cloud, K8s	Dev, App, Edge, Cloud, K8s, Browser, Devices	Dev, App, Edge, Cloud, K8s, Browser, Devices

Source: Cosmonic

K8s is not the answer to everything

Cloud Provider

A resource & capability provider



Kubernetes

A standardization platform to build platforms

A

B

C

Streamlined

Same behaviour, same capabilities everywhere

User Focused

Continuous extension to end user demand

Platform Foundation

Lay the the foundation for a simple software delivery and runtime

Thank you for your time

Max Körbächer

Founder & Cloud Native Advisor

REPLY LIQUID

CNCF Ambassador

LF Europe Advisory Board

Let's connect!