To k8s or not to k8s

Mateusz Dymiński

whoami

Mateusz Dymiński

- Software Developer at Nokia
- 8+ exp with Java
- 5+ exp with Go
- 4+ exp with K8s
- One of the organizer GoWroc - Golang Wroclaw Meetup
- Page: https://mateuszdyminski.com
- Github: github.com/mateuszdyminski
- Twitter: @m_dyminski
- LinkedIn: linkedin.com/in/mdyminski

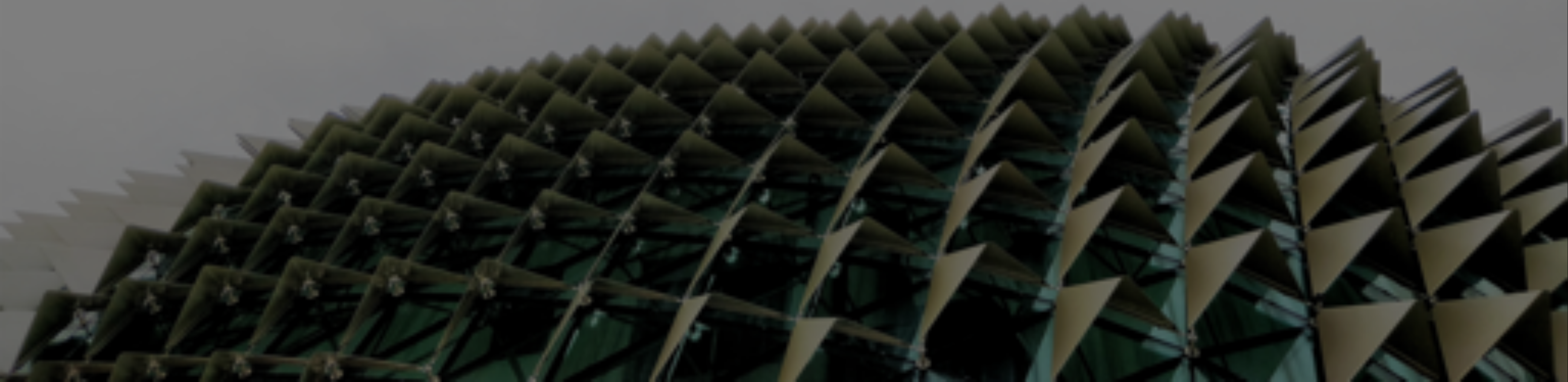
Agenda

- Why people decide to use Kubernetes
- Kubernetes (not that)killer features
- To k8s or not to k8s
  - Deployment
  - Architecture
- Summary

github.com/mateuszdyminski/k8s

Thesis

In 90% of cases you don’t need Kubernetes



I NEED TO KNOW WHY MOVING OUR APP TO THE CLOUD DIDN'T AUTOMATICALLY SOLVE ALL OUR PROBLEMS.
Dilbert.com @ScottAdamsSays
YOU WOULDN'T LET ME RE-ARCHITECT THE APP TO BE CLOUD-NATIVE.
JUST PUT IT IN CONTAINERS.
11-08-17 © 2017 Scott Adams, Inc./Dist. by Andrews McMeel
YOU CAN'T SOLVE A PROBLEM JUST BY SAYING TECHY THINGS.
KUBERNETES.

So why we(engineers) decide to take K8s?

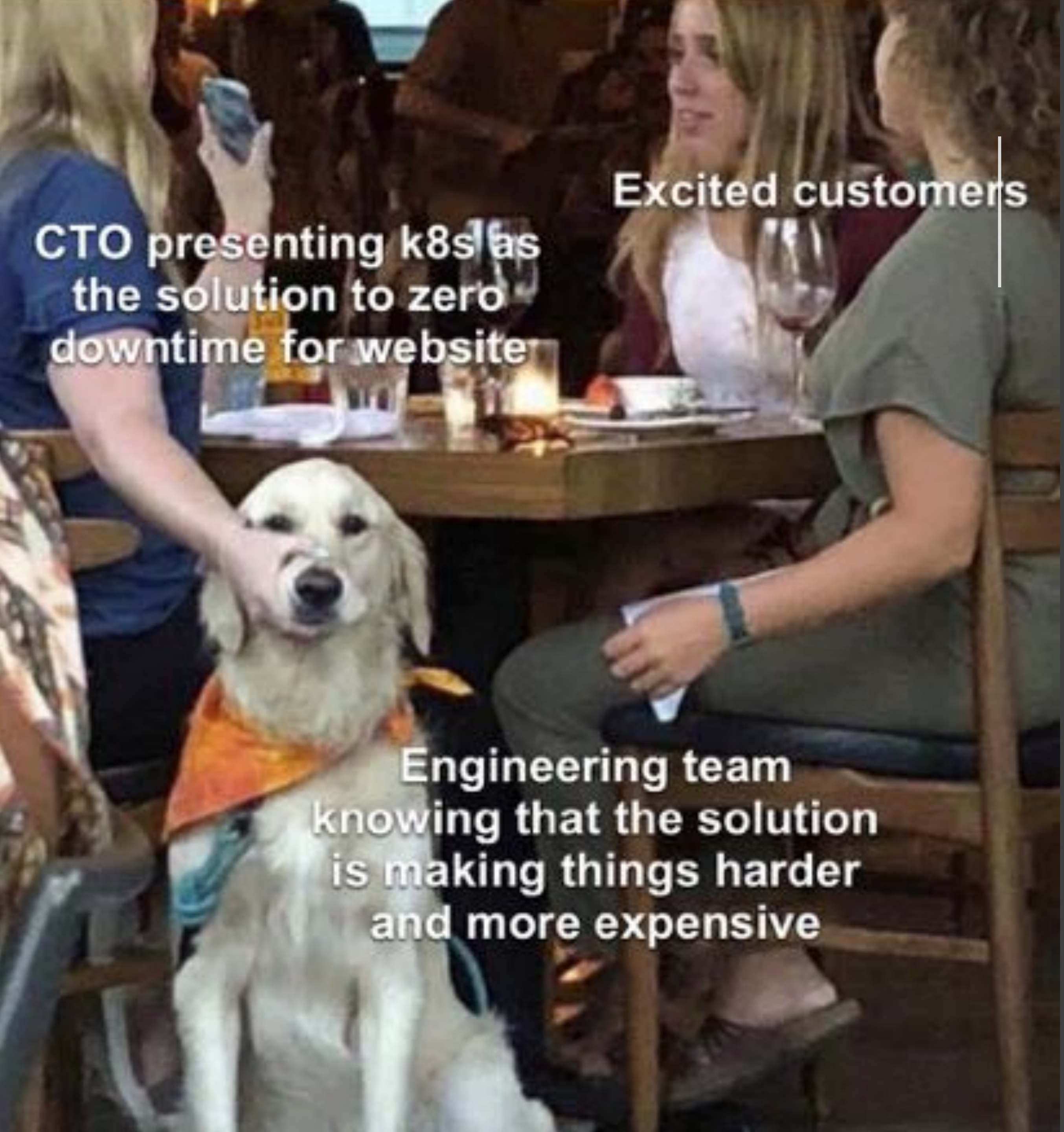
- To play with it
- Looks nice in CV
- Devs love new things and spend time on learning
- Everyone is using K8s
- Tons of examples on the web
- Fancy features: Scaling, Autoscaling, Zero Downtime Deployments, Fault tolerance
- It’s very rare to make analysis by engineers before making final decision
- Most of decisions in IT is hype-driven

So why managers decide to take K8s?

- To keep dev teams entertained
- It’s easier to hire people if team is working with new/fancy tech
- K8s features (scaling, autoscaling, healing, fault tolerance) are easy to sell to our company customers

That attitude leads to:

- Managers/devs sells features of Kubernetes like:
  - Scaling
  - Auto-Scaling
  - Zero downtime deployments
  - Fault tolerancy
- And they are claiming that if we use Kubernetes then the application will have all of these features
- But it’s not true!
- Kubernetes allows all of these things but it doesn’t mean that it’s done automatically
- We need to re-architecture our application to be cloud-native / k8s ready
- Usually senior devs look like:


Excited customers
CTO presenting k8s as the solution to zero downtime for website
Engineering team knowing that the solution is making things harder and more expensive

That!

Kubernetes Killer Features

Pods autoscaling

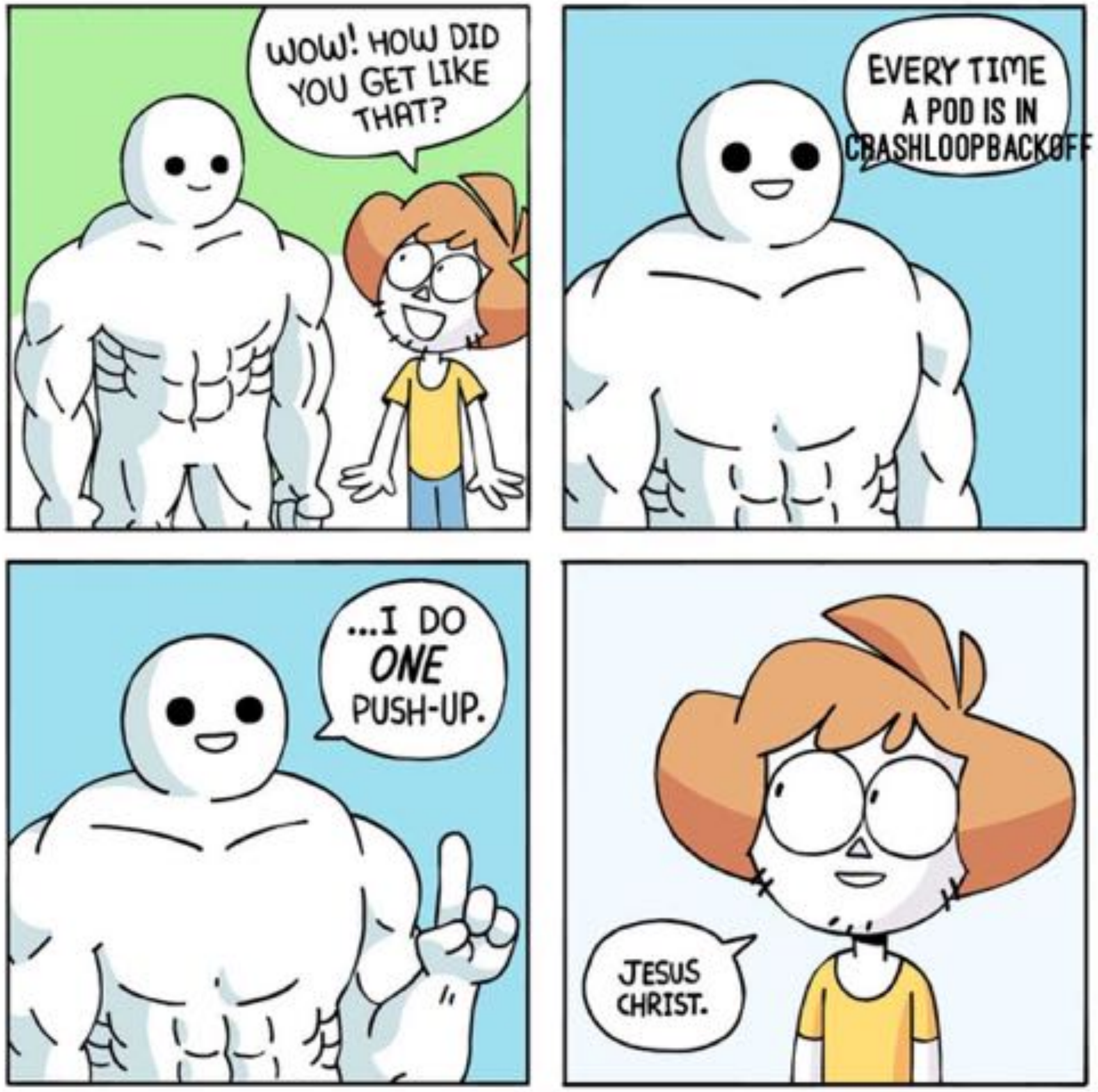
- You can based scaling on CPU, Mem, but it’s not very accurate
- Probably Prometheus integration would be nice
- Both – in K8s and inside of your application
- You need to be aware of cooldown/delay/scalingPolicies/StabilizationWindow – in many cases pod scale-up is too late or scale down too quickly
- It’s big challenge to tune your cluster for pods autoscaling

Zero Downtime Deployments

- You need to have proper signals catching in containers
- You need to implement graceful shutdown
  - You need to drain all ongoing requests
  - You need to stop accepting new ones
  - And add some sleep to be sure in 100% :)
- How about migration of DB?

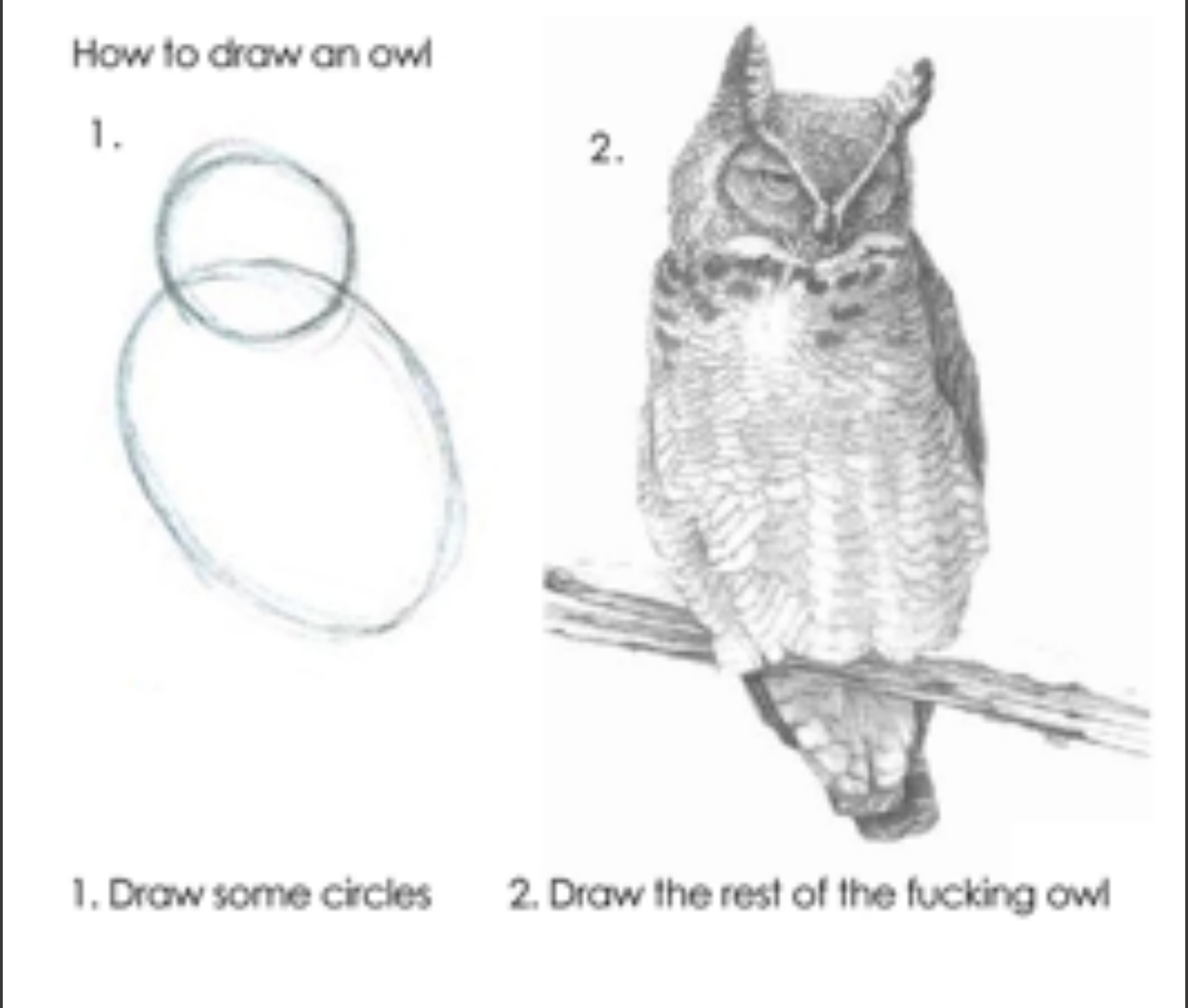
Fault tolerance

- At least 3 nodes just for masters(controlplane) – quite expensive
- In many cases clusters are heterogeneous – VMs are vary, some might be dedicated for fast storage, geographic etc
- If nodeSelectors are in place we might encounter crashloopbackoff
- We need to put pod limits everywhere
- We try to keep cluster busy with overbooking
- But - no resources = crashloopbackoff
- Make your cluster fault-tolerant is really hard job and reqiures weeks and months of observations


WOW! HOW DID YOU GET LIKE THAT?
EVERY TIME A POD IS IN CRASHLOOPBACKOFF
...I DO ONE PUSH-UP.
JESUS CHRIST.
SHEN COMIX

Scaling

- Adding automatically new VM is straightforward
- But when workloads are move to new VM and the traffic goes away
- The VM is still there :)


How to draw an owl
1.
2.
1. Draw some circles
2. Draw the rest of the fucking owl

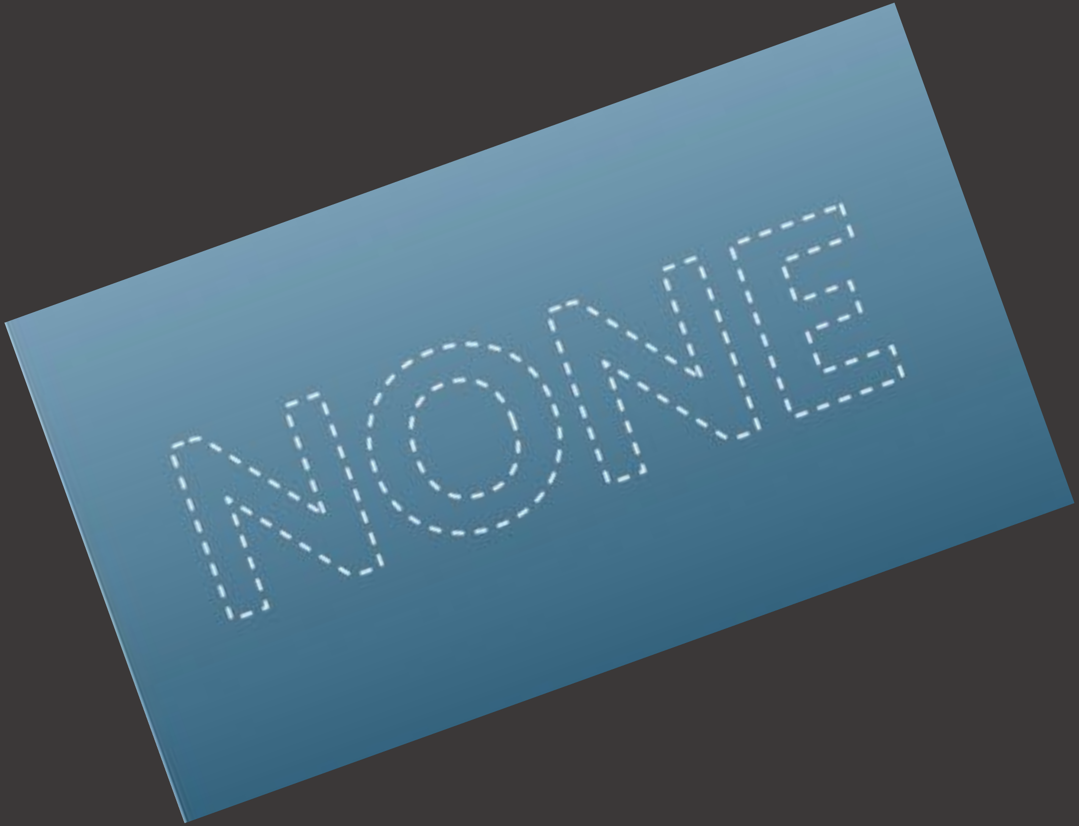
To K8s or not to K8s

Deployment of Kubernetes

Deployment of Kubernetes

- The very first question to ask yourself is:
- Where we would like to use Kubernetes?
- On-premise vs Cloud

On-premise – pros



- Dedicated hardware*

On-premise – cons

- Hard Ops
- Upgrades of cluster!
- Storage?
- You need great ops engineers
- Security
- Microsoft, Amazon, Google spend years to make them stable and usable
- Do you really think it’s that easy?

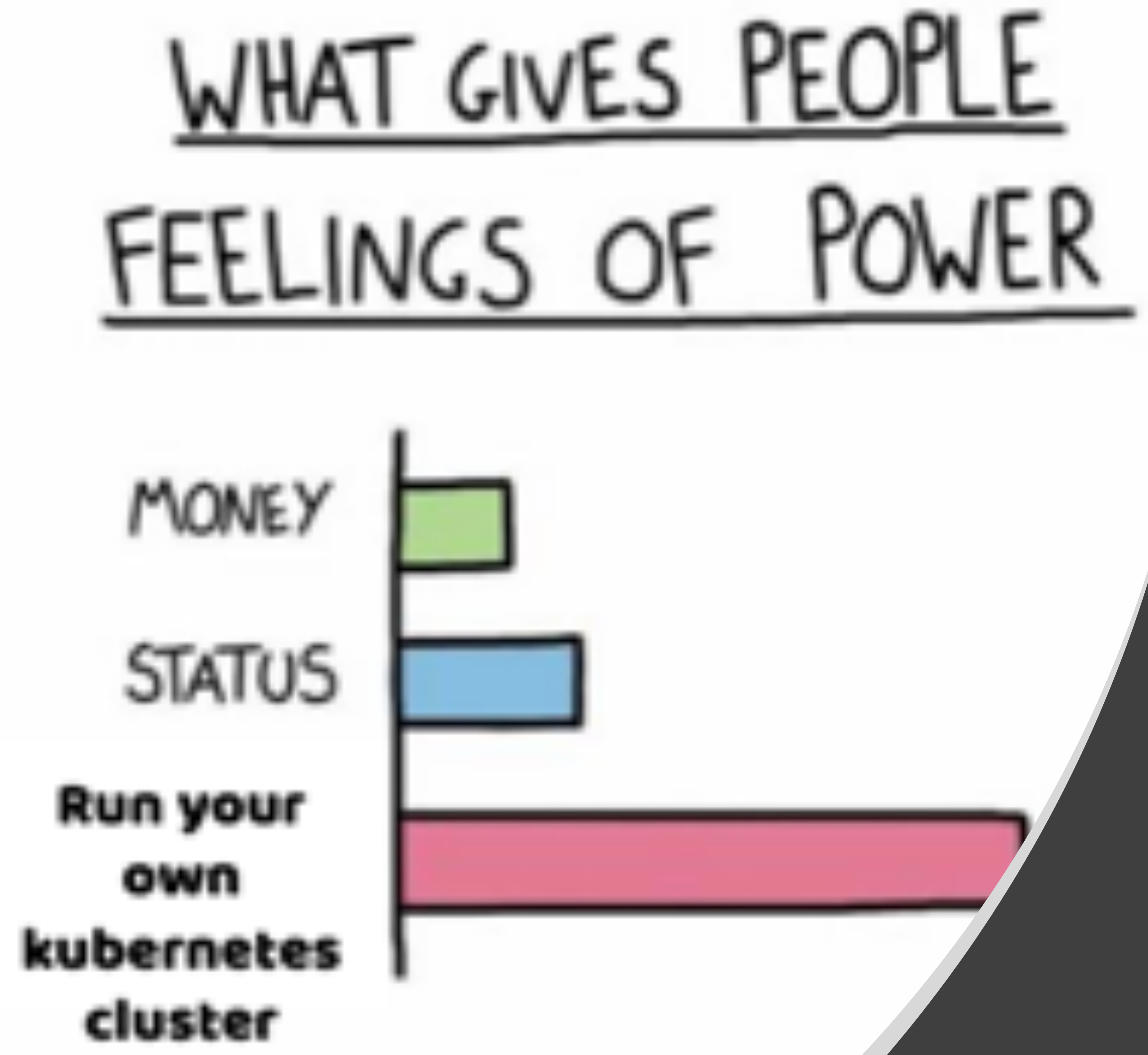
On-premise – when?

- You absolutely must have your compute local (for latency or security or legal reasons)
- Your workloads need to be on the edge (for latency or security)
- You need truly HUGE amounts of compute and/or data storage and managing your own datacenters is more cost effective
- You have some fancy hardware needs that you can’t get (cheaper) from a cloud provider

OH YOU HAVE A K8S BARE METAL CLUSTER?!
PLZ TELL ME MORE
imgflip.com

Going back to bed after successfully WASTING a day installing an on-prem cluster

K8s cluster
Your SRE team


WHAT GIVES PEOPLE FEELINGS OF POWER
MONEY
STATUS
Run your own kubernetes cluster

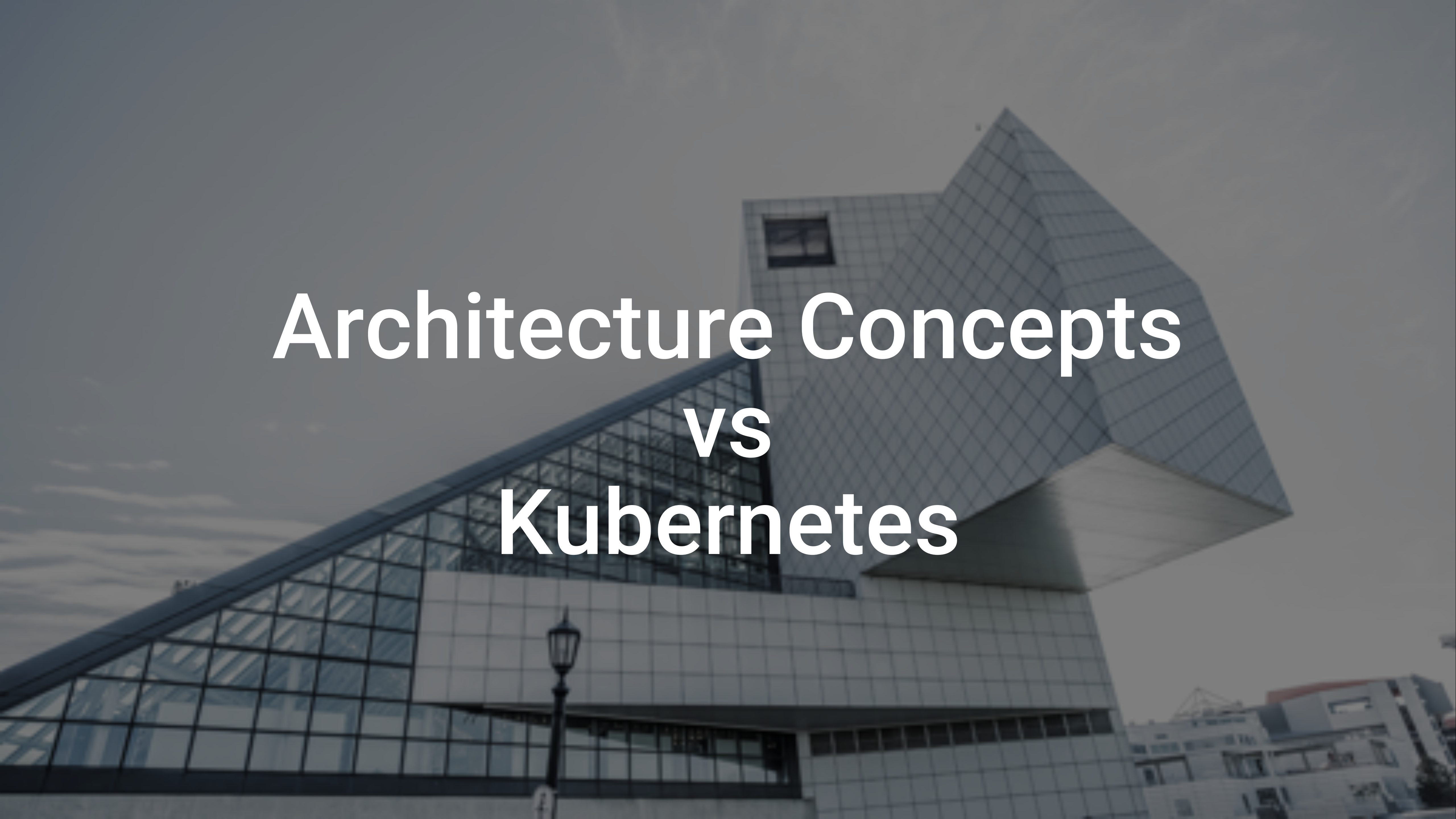
Kubernetes On Cloud

Own Kubernetes Cluster On Public Cloud

- No sense to do that
- All cons from on premise are still in place

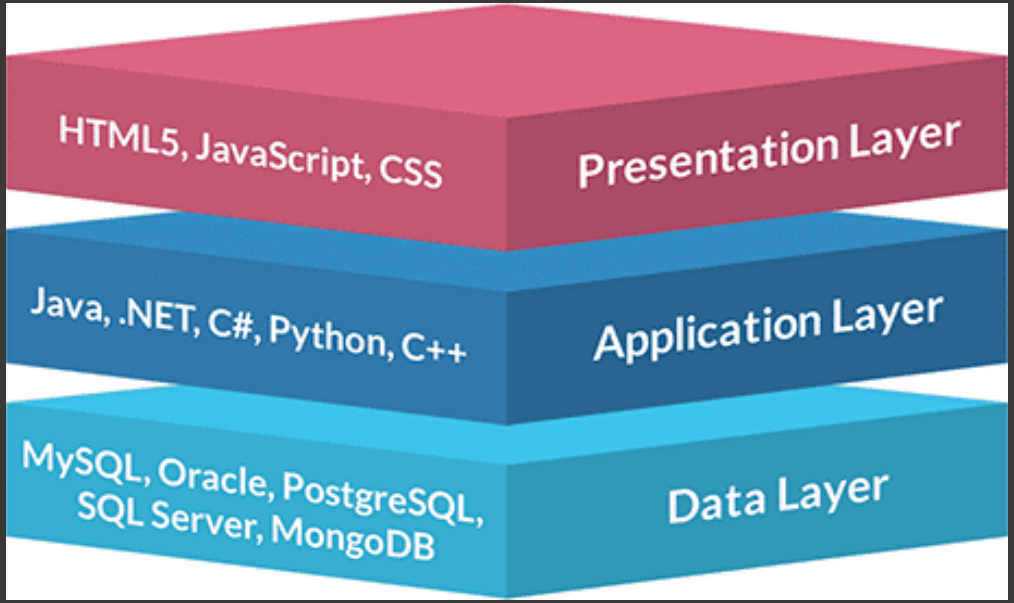
Managed Kubernetes Cluster On Public Cloud

- All hard operational things go away
- Finally you can focus on building your application
- Quite expensive
- But it’s worth to pay for „good night sleep”
- If you don’t have great DevOps engineers – it’s the only way to have Kubernetes custer in good shape
- Microsoft, Amazon, Google spend years to make K8s stable



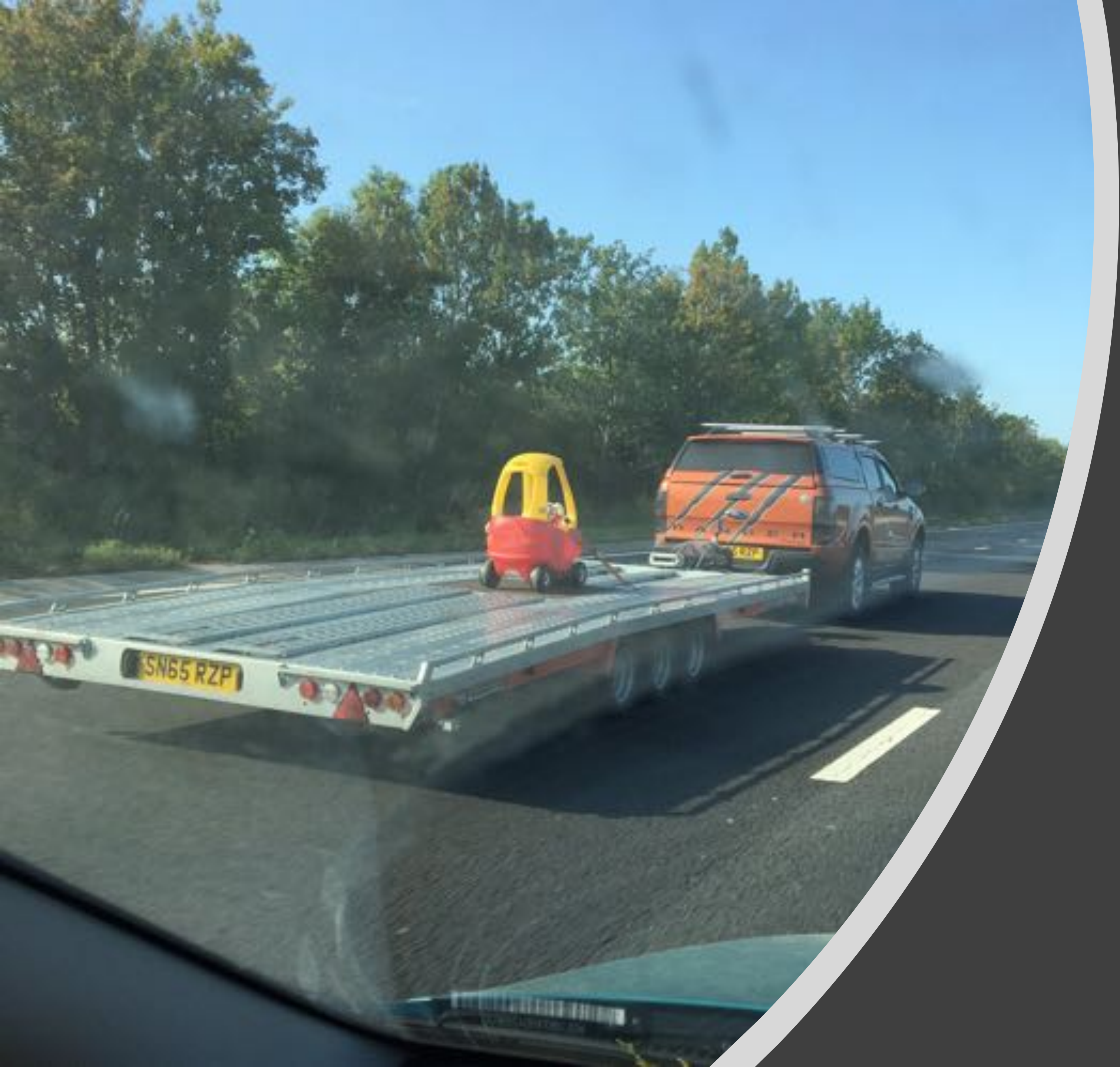
Architecture Concepts
vs
Kubernetes

{2,3}-tier architecture


HTML5, JavaScript, CSS
Presentation Layer
Java, .NET, C#, Python, C++
Application Layer
MySQL, Oracle, PostgreSQL, SQL Server, MongoDB
Data Layer

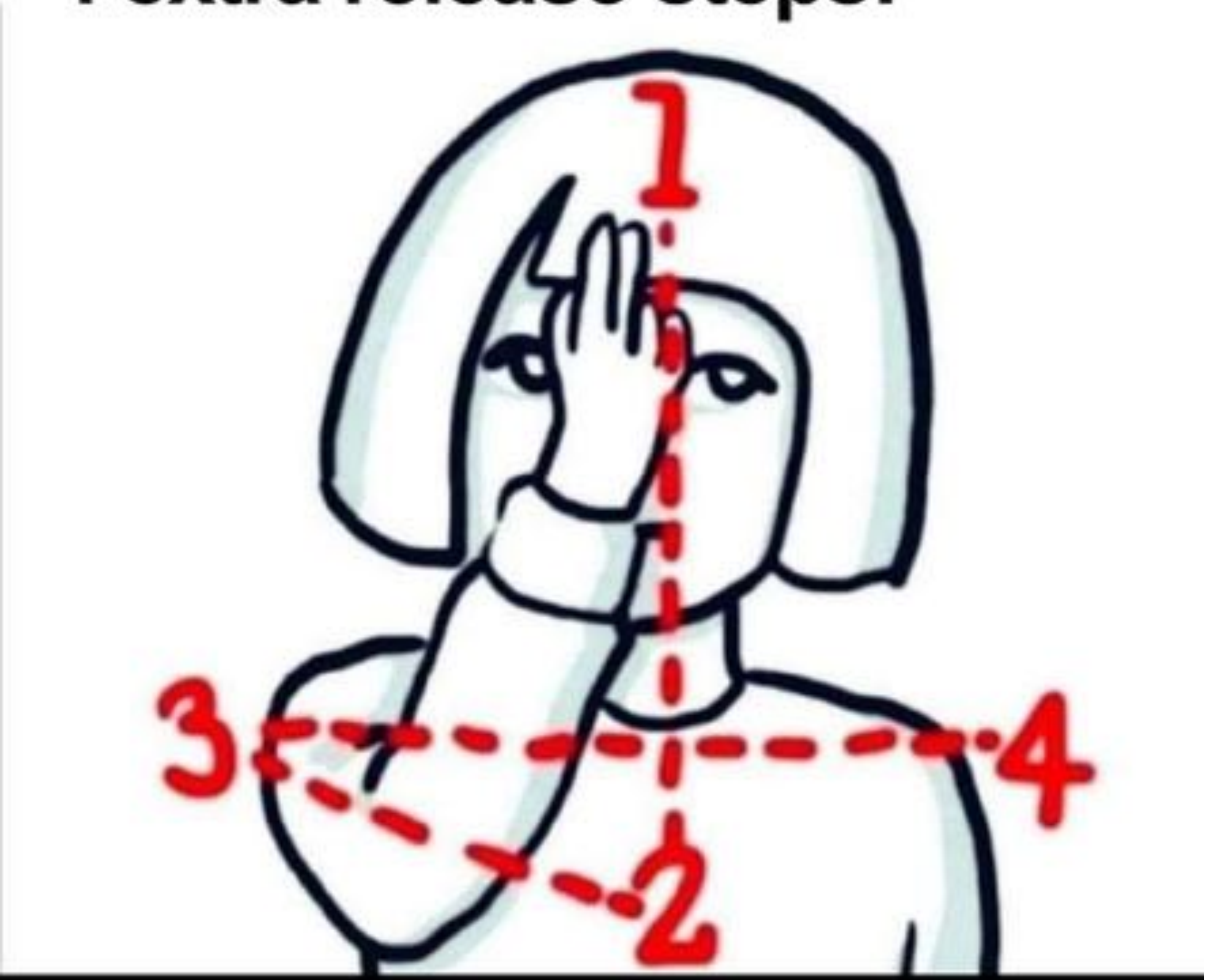
- Classic 3-tier App: DB – Http Server – Http Client/Mobile client
- Classic 2-tier App: DB – Http Client/Mobile client
- Of course K8s is not needed for you!

Source: https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/



You will end up with:

If you deploy databases in Kubernetes, we recommend following these 4 extra release steps:



And remember about:

Docker-compose architecture

- Usually created as PoC
- Then - someone made the decision to productify it
- Applicable for green fields projects too
- Depends on needs:
  - Are you planning microservices architecture?
  - Do you really need the whole K8s complexity?
  - Does your team has experience with K8s?
  - Try to think about feature of k8s which you really need and find the alternative
  - Make that decision very carefully because reverting it might take months
  - Make ADR – Architecture Decision Record before such important decision

  - more: https://github.com/joelparkerhenderson/architecture_decision_record

Distributed monolith (aka wannabe microservices) architecture

- You have screwed-up microservices so there is high chance you will screw up migration
- Another new things to learn
- Current problems won’t disappear with K8s
- Overall complexity goes much higher
- K8s won’t help you
- Fix the architecture first, move to K8s then

Microservices architecture

- K8s really shines here!
- If your are doing microservices well – K8s is for you

- Consider alternative: Hashicorp Nomad + Consul

Monolith

- IBM WebSphere with multiple Java EE applications?
- Single old-time Java App (version 5,6,7)?
- Single pod(s) or few pods on K8s?
- Shared resources (CPU, Mem, Network) will kill you(or your app)
- Running SQL DB on K8s?

- Definatelly K8s is not for you!

MANAGED KUBERNETES
LEGACY APPLICATION

Monolith which needs to be modernise

- Rearchitecture your app first!
- Modularise components and put them in containers
- Run K8s cluster next to old monolith and move components one by one into K8s cluster

- K8s makes sense here

Summary

Takeaways

- K8s is a great tool, but most of you don’t need it
- Having your own K8s cluster is extremely hard to operate in long-term
- K8s on premise - only when you really don’t have any other choice
- And have great engineers
- K8s won’t fulfill your non-function requirements
- Most of its features are available with other tools
- Think 3 times before choose K8s
- Or 10 times!

Kubernetes: The Complete Guide
Kubernetes:
Bits You Actually Need
MVC Framework
in Action

Q&A