
KubeCon
CloudNativeCon
Europe 2023
TiKV

Use Knative When You Can

Kubernetes When You Must

KubeCon | CloudNativeCon

Europe 2023

A fresh look into Knative

@davidhadas and @maximilien

https://www.linkedin.com/in/davidhadas

https://www.linkedin.com/in/drmaximilien

Who are we…

KubeCon CloudNativeCon Europe 2023

@davidhadas

David Hadas @IBM Research

Day job:

- Cloud security
- ML

Knative Security-Guard
A 2.5 years research to protect K8s services

15 years **IBM Research**
15 years in the IL Startup scene

@maximilien

Michael Maximilien (aka Max or Dr. Max)

A distinguished engineer with IBM

Responsibility:

- Open serverless
- Open quantum

25 years of contributions to open source:
Java, Cloud Foundry, Knative, Qiskit.

An avid cyclist and award winning photographer.

Cloud Computing

The Big Picture

At the center of the modern interconnected world.

Used by most modern applications for

Aggregating & processing data

Constructing information that edge devices need

The downside:

Demand is expected to grow annually by 15%.

Projected to reach 50% of IT spending in key market segments

Already consumes 1-1.5% of global energy and its growth represents an actual threat to the environment.

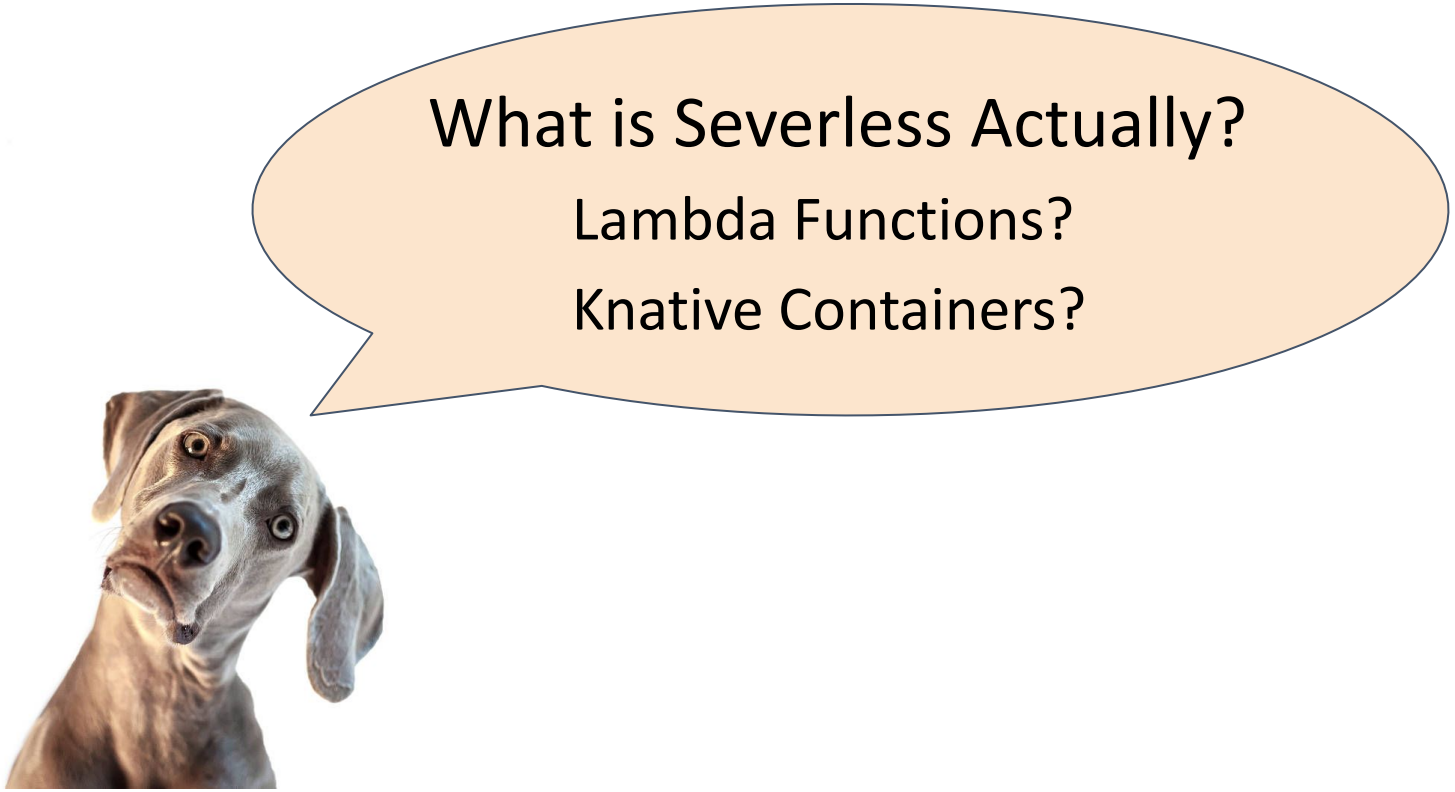
Cloud Computing
Takeaways from the Big Picture

- In view of the growing demand,
  We need to improve **energy-efficiency** and **cost-efficiency:**
  - Better control over the amount of resources used per service
    - Optimize costs
    - Optimize power consumption

  - Serverless offers these twin controls — out of the box

- Goal:
  Use the benefits of serverless systems for the deployment of Microservices

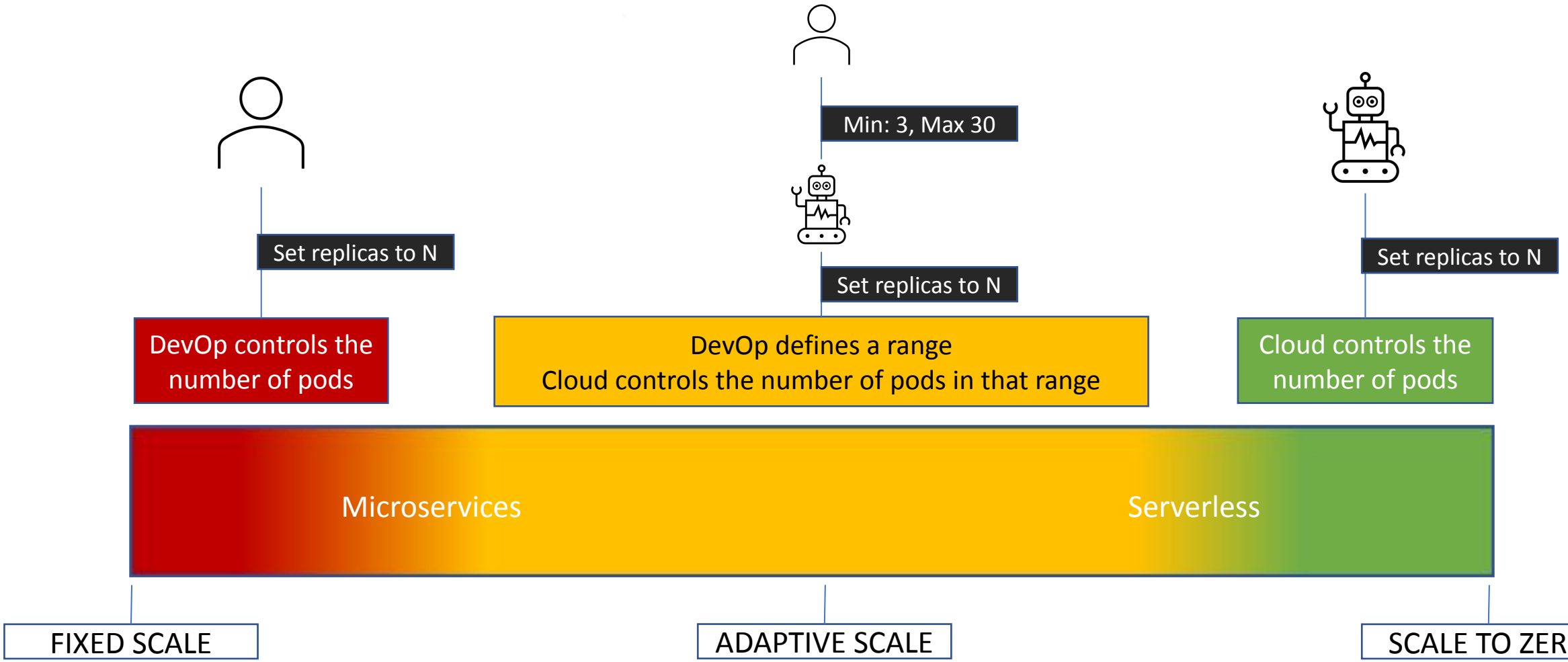
Using Microservices vs. Serverless



An adaptive and efficient way to consume the same compute resources (containers)

- The control over the amount of resources used is offloaded to the cloud
- Pay as you go (truly)

Microservice and Serverless Containers as a continuum of the same architectural style

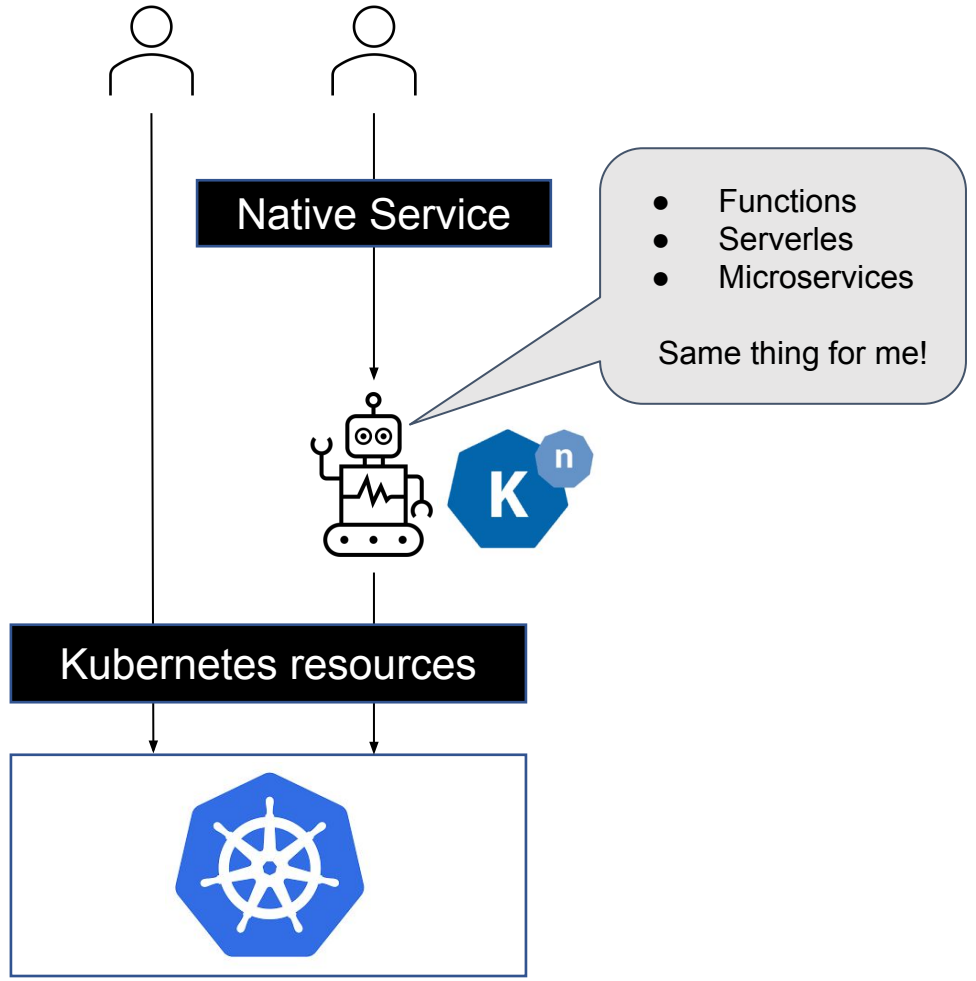


Knative

Orchestration Platform for Native K8s Services

A "Native Service" is any of the following:

- A Microservice (a typical twelve-factor app!)
- A Serverless Container
- A Serverless Function



- Kubernetes centers around **container** orchestration
- Knative centers around the orchestration of **Native Services** in Kubernetes.

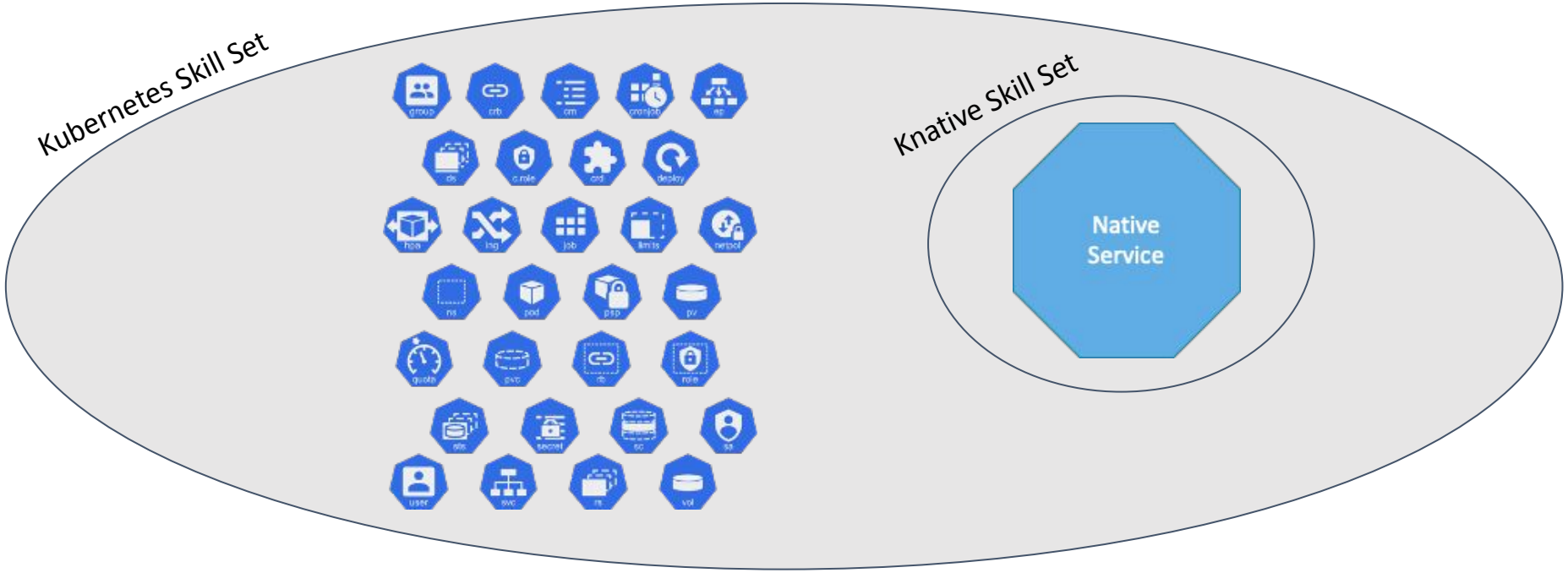
Lower Bar Skill Set

What makes Knative Simpler to Use and to Maintain?

Knative eliminates many Kubernetes complexities.

- Automates the entire microservice deployment process.
- Users need only provide a high-level service definition.
- Knative does not require full understanding of Kubernetes.
- Built in auto scaling and blue-green deployments — revision control.
- Easily tie and react to eventing sources and sinks.

Knative offers higher return on the time invested



Knative is an Opinionated Kubernetes

- Knative comes with
  - Present of mainstream best practices
  - Build-in Security

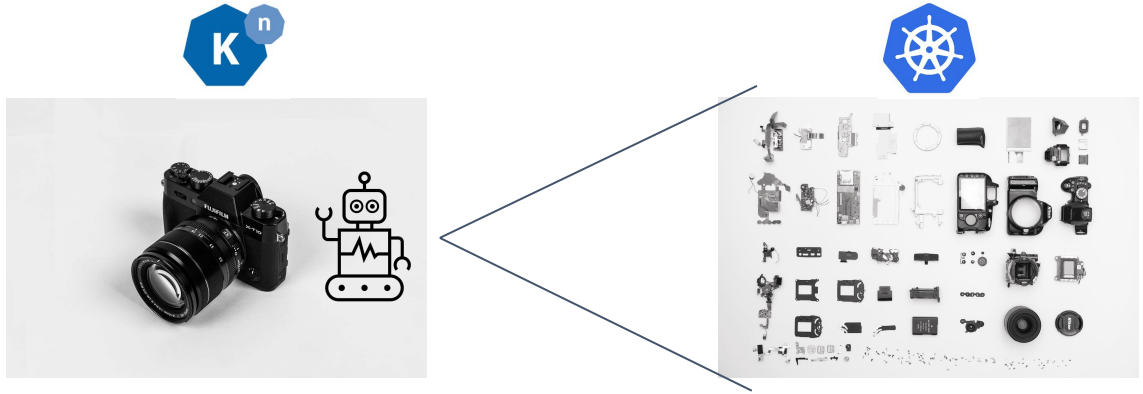
- Less effort to both deploy and then maintain.
  - Simpler and straightforward CLI tool (See KN)
  - Use a single yaml per each Native Service

    (Skip implementation details)

Knative comes pre-assembled

It offers less knobs to tweak and to control

It is less complicated



Microservice deployment

```
kn service create testsrv --image davidhadas/test-service --env SLEEP=3s --scale-min 1
```

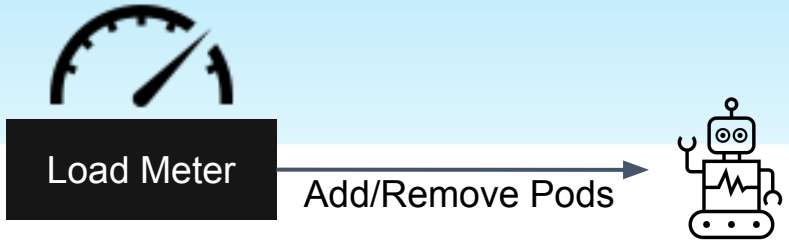
Serverless deployment

```
kn service create testsrv --image davidhadas/test-service --env SLEEP=3s
```

Function deployment

```
cd </Path/To/My/Proj/Dir>
kn func deploy
```

Auto-Scaling



Out-of-the-box automated horizontal scaling for **Native K8s Services**.

- Adjusts the number of pod replicas to the actual load as needed
- Various controls: min-scale, max-scale
- Scales based on request load — tested at scale

Pre-provisioning

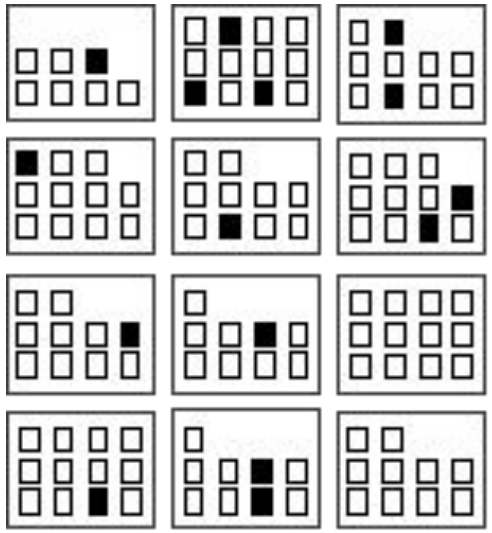
- Users required to ensure sufficient resources are provisioned to absorb any peak time loads.
- Knative eliminates the need for pre-provisioning

Consequently, Knative will use fewer pods on average

- Run more services to be served by any given Kubernetes cluster.

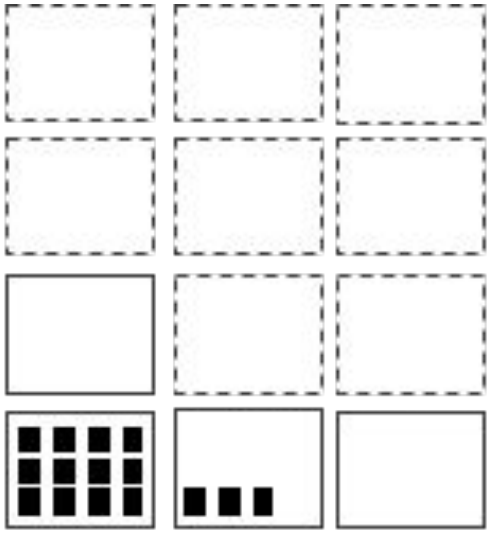
Auto-Scaling

Why is it so important? << Potential



Idle resource

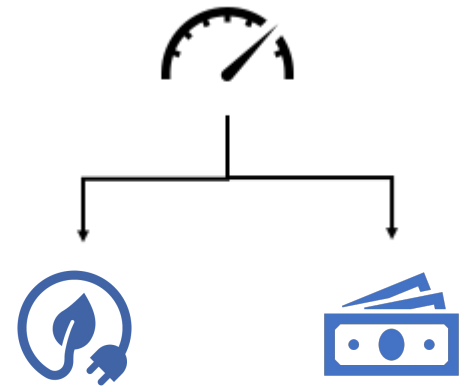
Used resource



Potential energy savings with Knative

Suspended VM

Active VM



Thanks to Auto-Scaling
We have no (or less) idle pods

Option to suspend VMs

Reduce energy footprint - Reduce costs

Revision Control

Microservice maintenance == Deploying new revisions in production

- How to ensure the new revision does not take the service down?
- How to rollback?
- How to lower the risk?
- How to test different versions, e.g., for user happiness

Kubernetes achieve rolling updates pod-by-pod with slight performance degradation - Best practice is to try a new revision with some small percentage of user traffic, and grow the ration served by the new revision over time, so-called blue-green deployment.

Knative automates offers a more controlled process to safeguard against the hazards of deploying new revisions.

	0%	1%	5%	25%	100%
	100%	99%	95%	75%	0%
	Day 0	Revision Launch	Grow Confidence	Looking Good	Can Scale

Suitable for blue/green deployments and canary deployments.

Security
What makes Knative more secure?

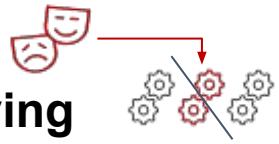
- Knative takes care of **TLS and certificates**
  - All hops protection is under work

- Rolling out **Patches**
  - Easier and safer to patch

- **Monitoring** and Controlling the the Services
  - Integrated Security Behavior Monitoring & Control
  - See in next slides…

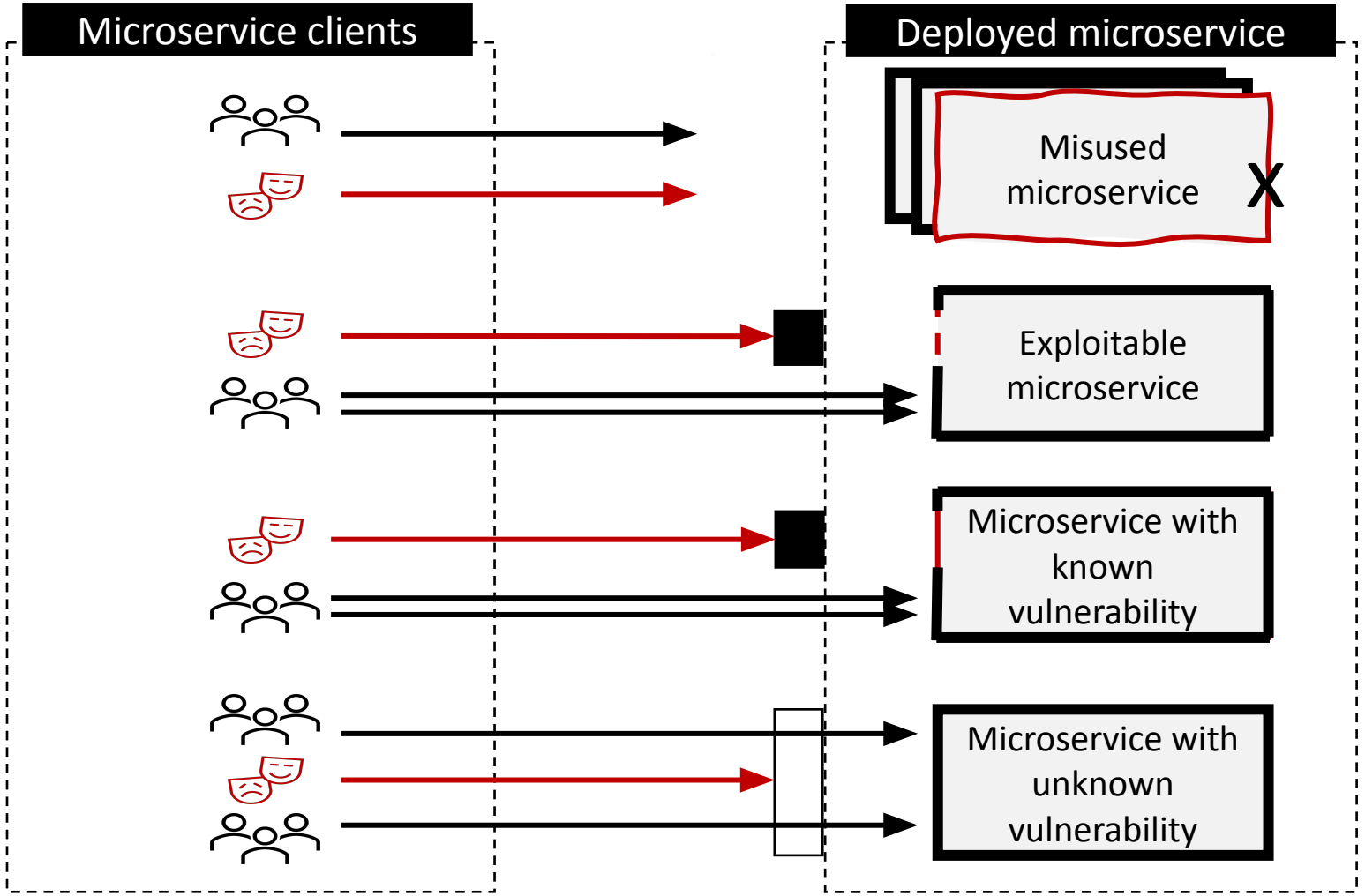
- Avoid **Misconfiguration** 
  - Drastically less room for Misconfiguration upon deployment
  - Blocking insecure unallowed configuration
  - Warning about insecure allowed configuration

- Avoids **Configuration Drift** 

- Continual **Repaving** 
  - Making it hard for an attacker to persist

Security: Kubernetes Blog Post

"Consider All Microservices Vulnerable"



Misused

Detect/Remove a misused microservice pod, allow other pods to continue the service

Exploitable

Detect/Block a known exploit and allow service to continue

Vulnerable

Detect/Block patterns that may be used to exploit a known vulnerability

Normal

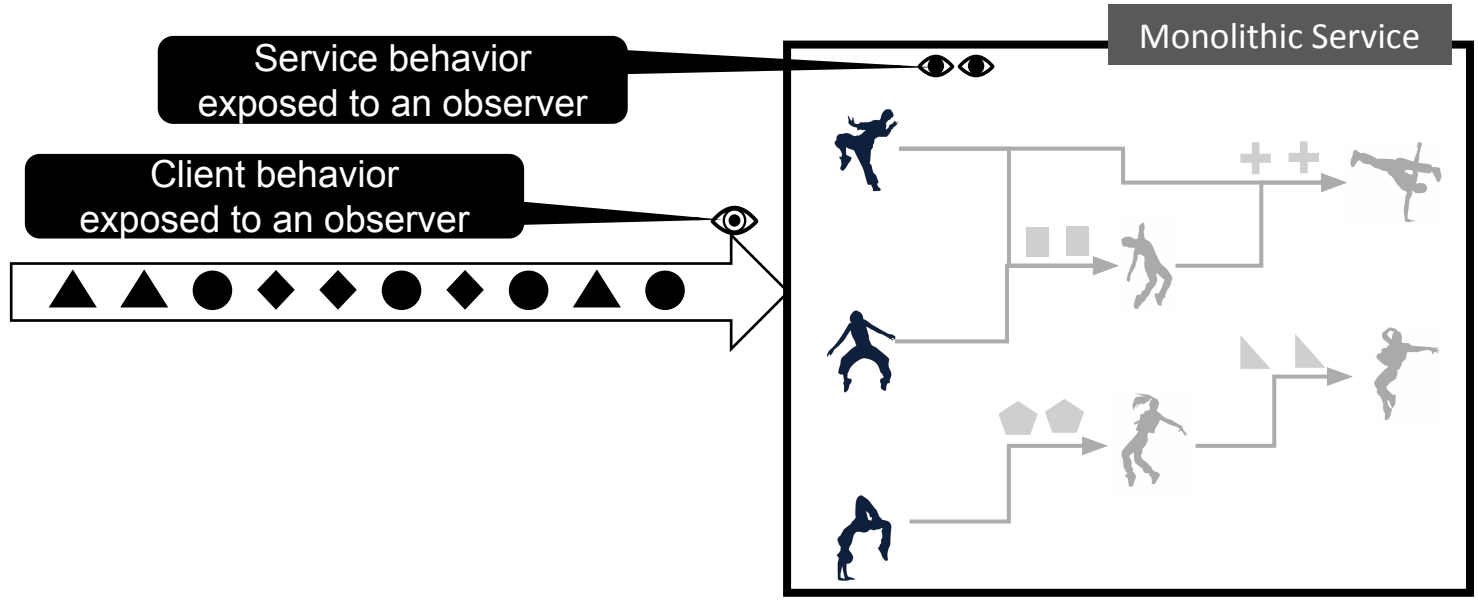
Detect/Block unknown patterns which may be part of a zero-day exploit

Security - Guard

Guard uses **Security Behavior Analytics**

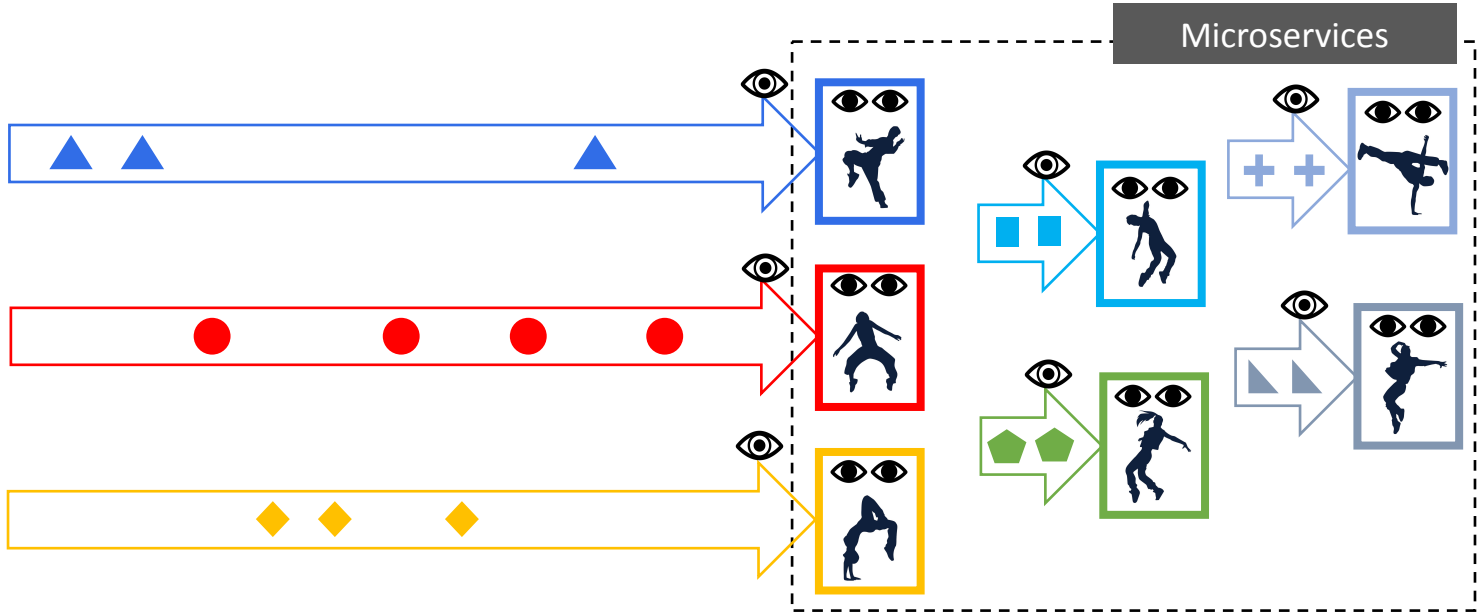
to protect vulnerable **Native K8s Services**

from being exploited.



Security Behavior Analytics is

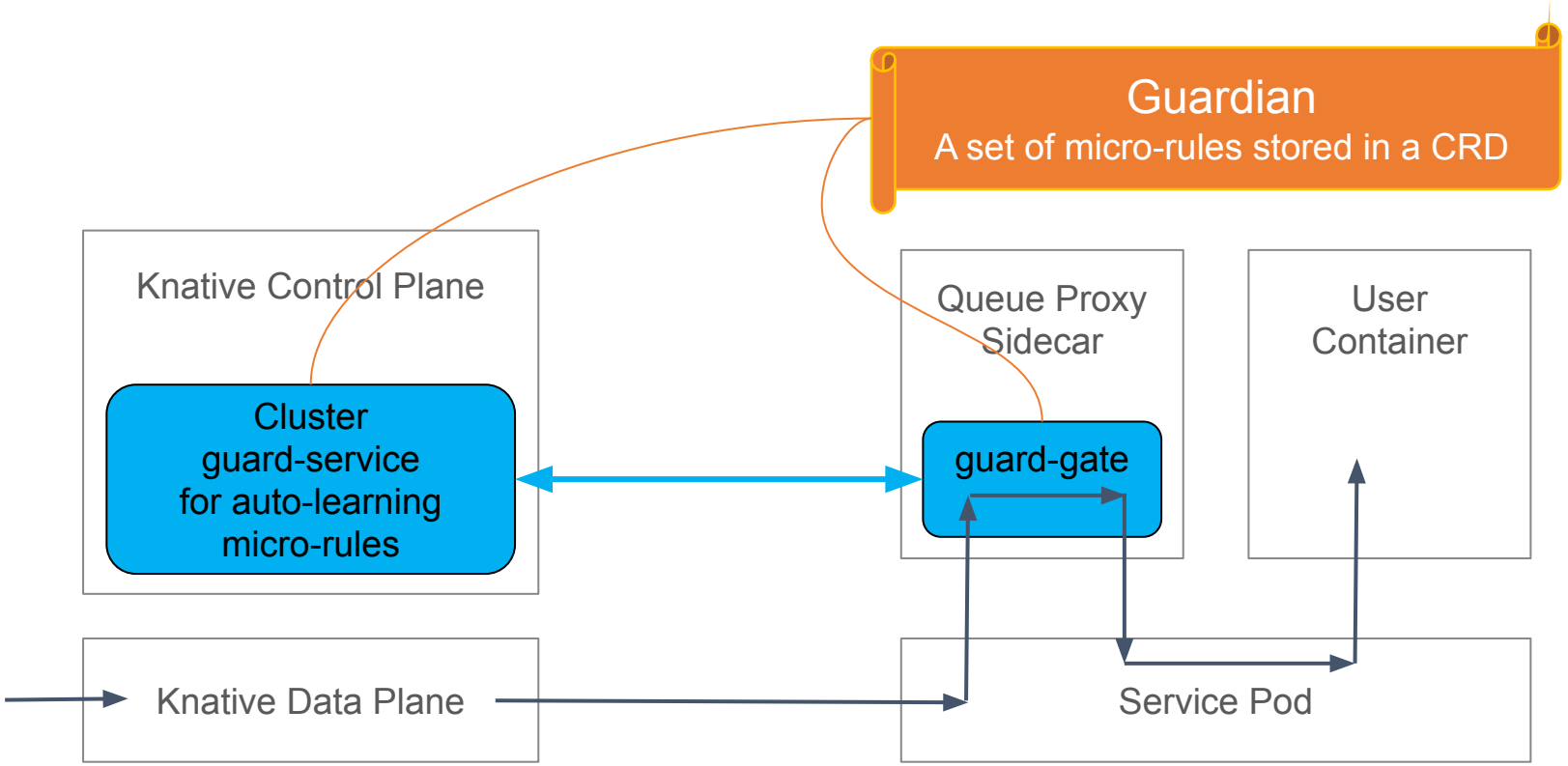
well adapted to Microservices



Security - Guard Architecture

Knative's opinionated and well structured principles, enable integrating security into the platform.

The integrated **Security Behavior Analytics** protect user containers from exploitation and identify compromised user containers once they become active.



Demo

Watch on YouTube:
https://youtu.be/IjQ596Eafkw

KubeCon CloudNativeCon Europe 2023

1. Deploy a service

```
kn service create testsrv --image=... --env SLEEP=2s --scale-min 1
```

2. Use wrk to load the service and see that it auto scales up and down

```
wrk -t10 -c4000 -d2000s http://testsrv.default.127.0.0.1.sslip.io
```

3. Revisions:
   a. Keep WRK running
   b. Create a revision, give it 5%
   c. Move to 50%
   d. Move to 100%
   e. Revert to prev

```
kn service update testsrv --env SLEEP=4s --traffic @latest=5 --traffic testsrv-00001=95
kn service update testsrv --env SLEEP=4s --traffic testsrv-00002=50 --traffic testsrv-00001=50
kn service update testsrv --env SLEEP=4s --traffic testsrv-00002=100
kn service update testsrv --env SLEEP=4s --traffic testsrv-00001=100
```

4. Security

```
curl http://testsrv.default.127.0.0.1.sslip.io -A "hd(j)kahh"
```

   a. Show queue logs, use curl to create a different request, and show the alert

```
"SECURITY ALERT! Session ->[HttpRequest:[Headers:[KeyVal:[
                KnownKey User-Agent:[Special Chars Used:[Unexpected Flags RoundBracket  (0x80) in Value,],],],],],]"
```

Use Knative When You Can

Kubernetes When You Must

A fresh look into Knative

@davidhadas and @maximilien

Limitations of Knative

Knative is an Opinionated Kubernetes

- Compels all deployed services to **abide by certain design patterns**
  - Twelve-factor app
  - Serverless
- These patterns however are extremely common in Kubernetes microservices

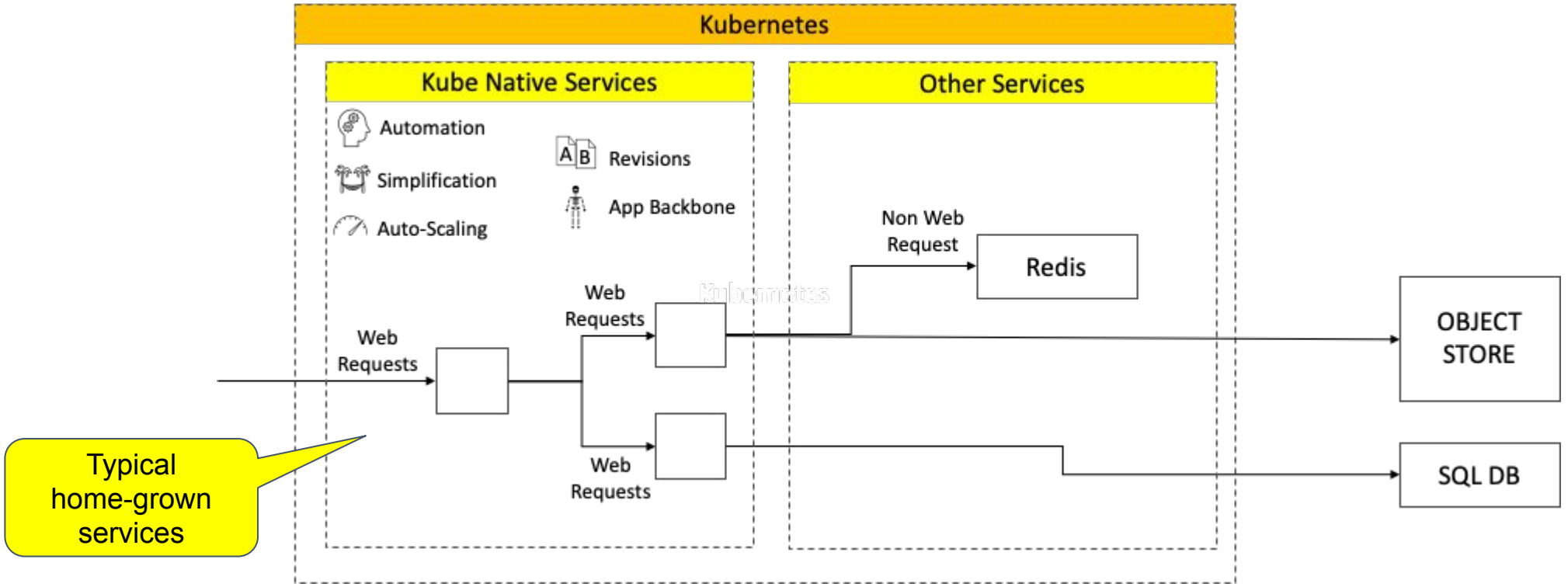
Main Limitations

- **Web services only -** Serves HTTP/1.1, HTTP/2 (+gRPC) - not serving non web traffic
- **Single ingress port per service -** Either a single user container or multiple containers, where only one exposed
- **Other limitations -** Not meant to offer as many configuration options as Kubernetes. Yet, a typical Twelve-Factor app microservice is covered.
- If you decide to scale to zero….. **Cold start penalty**

Mix and Match

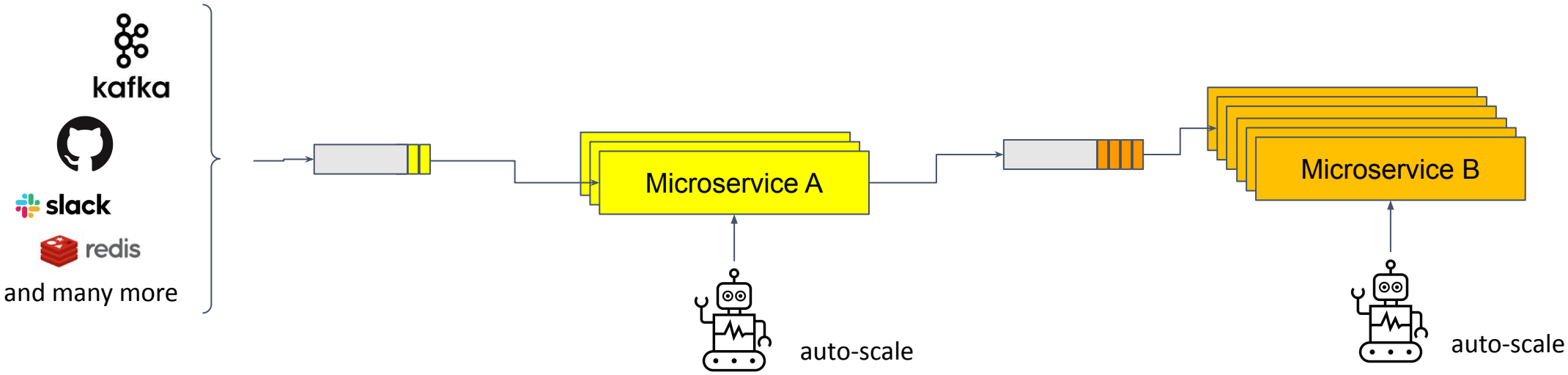
Use Knative When You Can
Kubernetes When You Must

Since Knative extends Kubernetes, it allows a mix and match between Native K8s services (typically most of your home-grown services) and other services.



Application Backbone - Knative Eventing

- Serve as the application backbone - queues and distributes the application events
  - Events can be from any sources (internal to cluster or external)
  - Connecting multiple Native K8s Services
  - Rate Decoupling between Native K8s Services
  - Each application layer can scale independently.
  - Absorb application load as needed until the application layer is auto-scaled.
- Handle Eventing sources (any) - See: https://knative.dev/docs/eventing/sources/



Summary

- Simpler to use
- Simpler to maintain
- Consume functions or images
- More secure
- Offers Revision Control
- No need to pre-provision
- Packed with an event system

Developers
Build and Deploy Apps

API

Knative

Kubernetes

Scales nicely!

Users and Systems (IoT et al)
Use the services deployed by the developers

- More secure
- Higher utilization (lower costs)
- Less energy - green!

Operators
Deploy and manage
Knative and Kubernetes

Use Knative When You Can

Kubernetes When You Must

KubeCon | CloudNativeCon

Europe 2023

A fresh look into Knative

@davidhadas and @maximilien

https://www.linkedin.com/in/davidhadas

https://www.linkedin.com/in/drmaximilien

Please scan the QR Code above
to leave feedback on this session