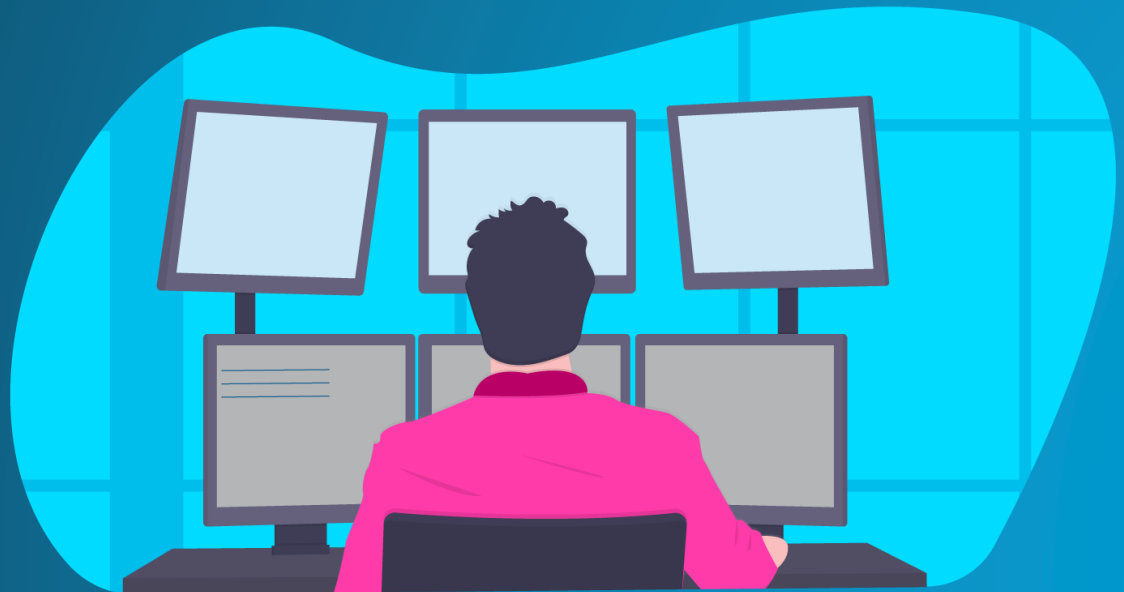
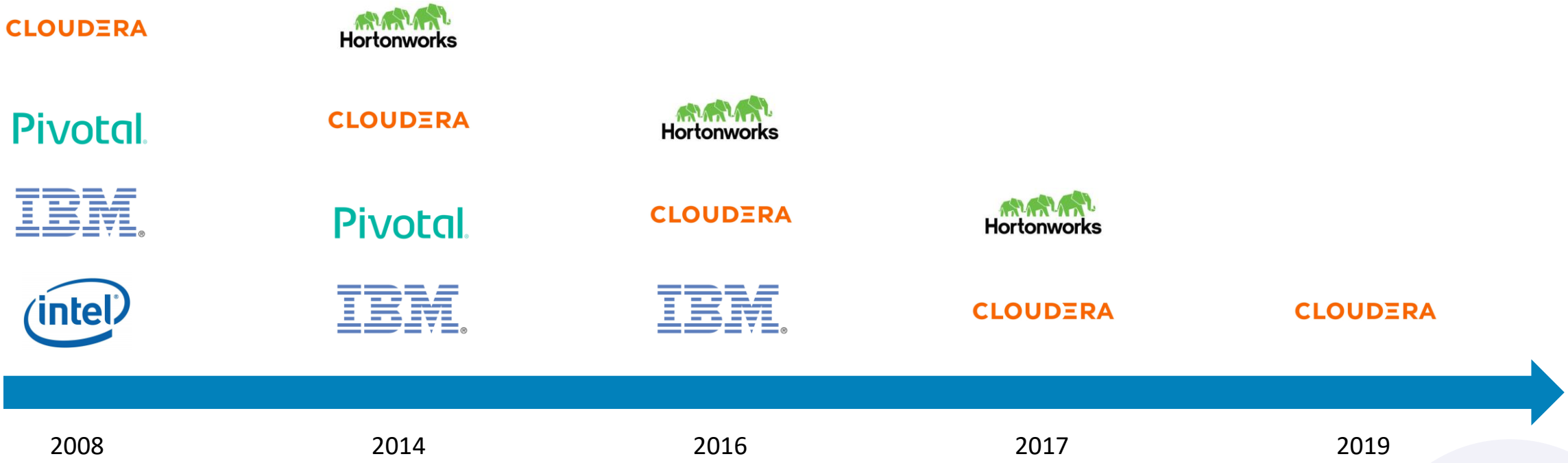
Stackable @ >BLN BZZ/ WRDS

Building a New Big Data Distribution Based on Kubernetes – With a Twist!

15th June 2021 | Track **Scale** | Format **Talk Intermediate**

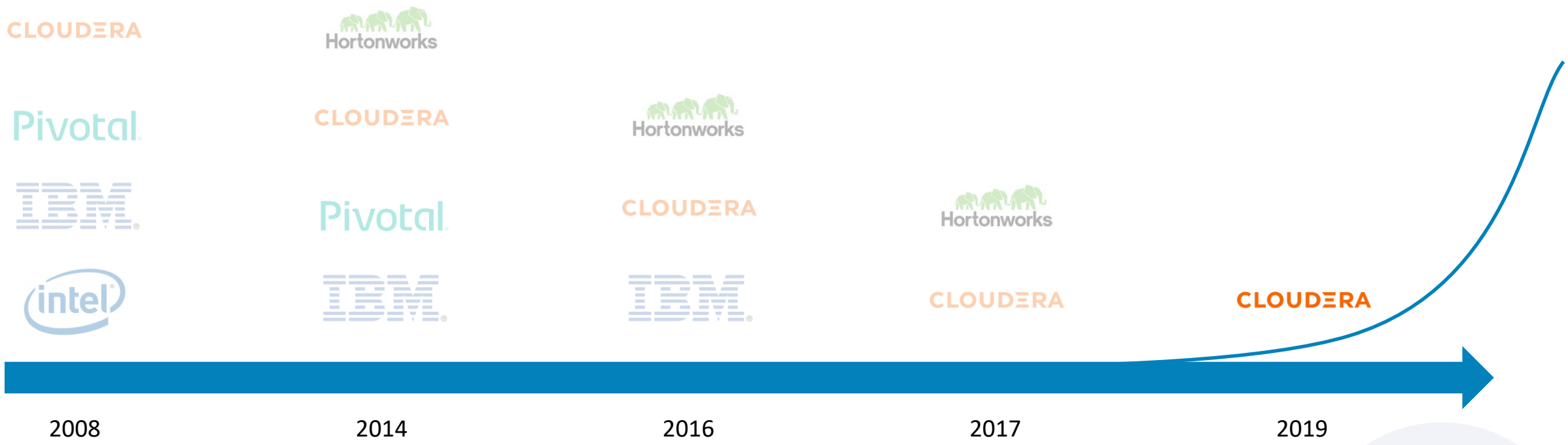


Hadoop Distributions over Time



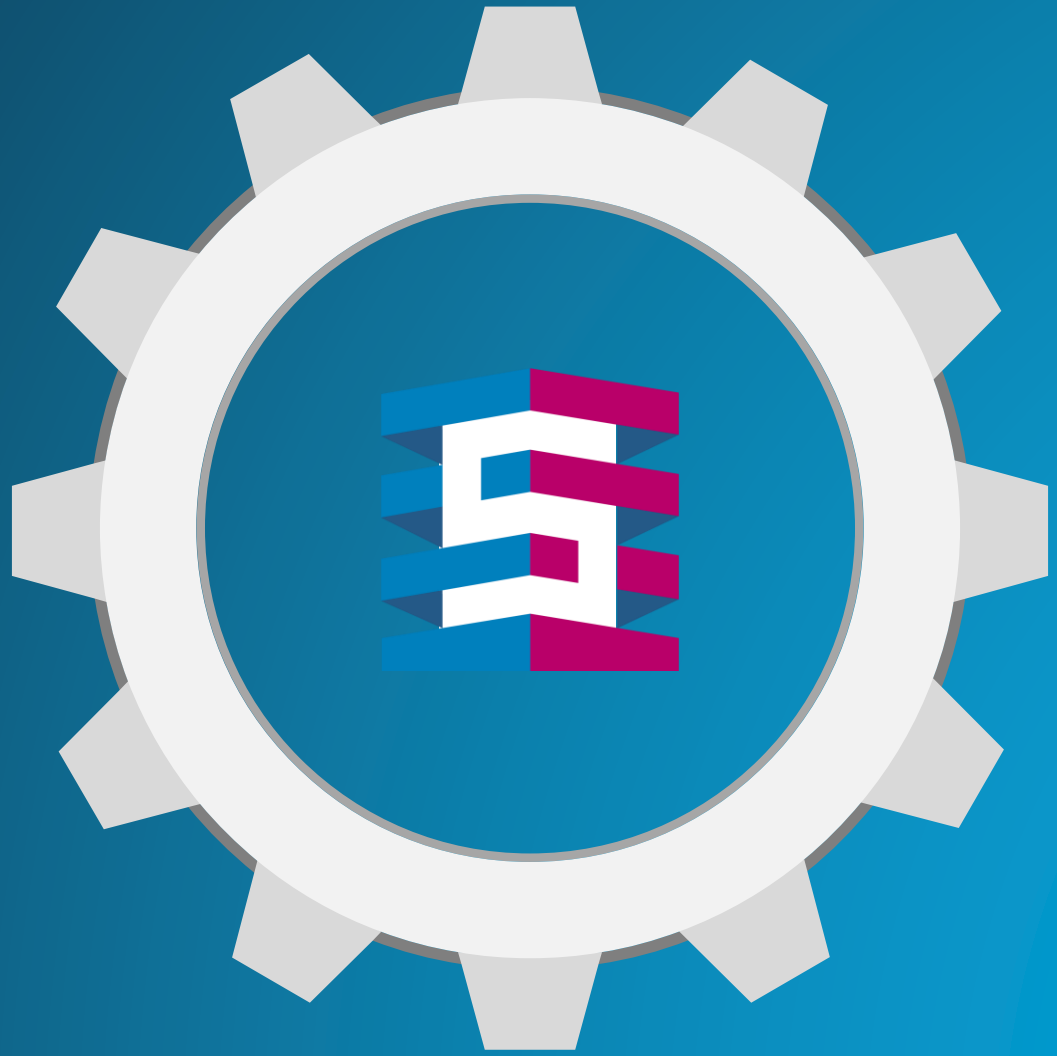
Stackable

How Did This Impact the per Node Price?



Stackable

The History of Stackable



Stackable

Kubernetes on One (Well, Two) Slide(s)

An orchestration framework to manage services

Declarative by nature - you define what reality should look like and Kubernetes goes and makes it happen

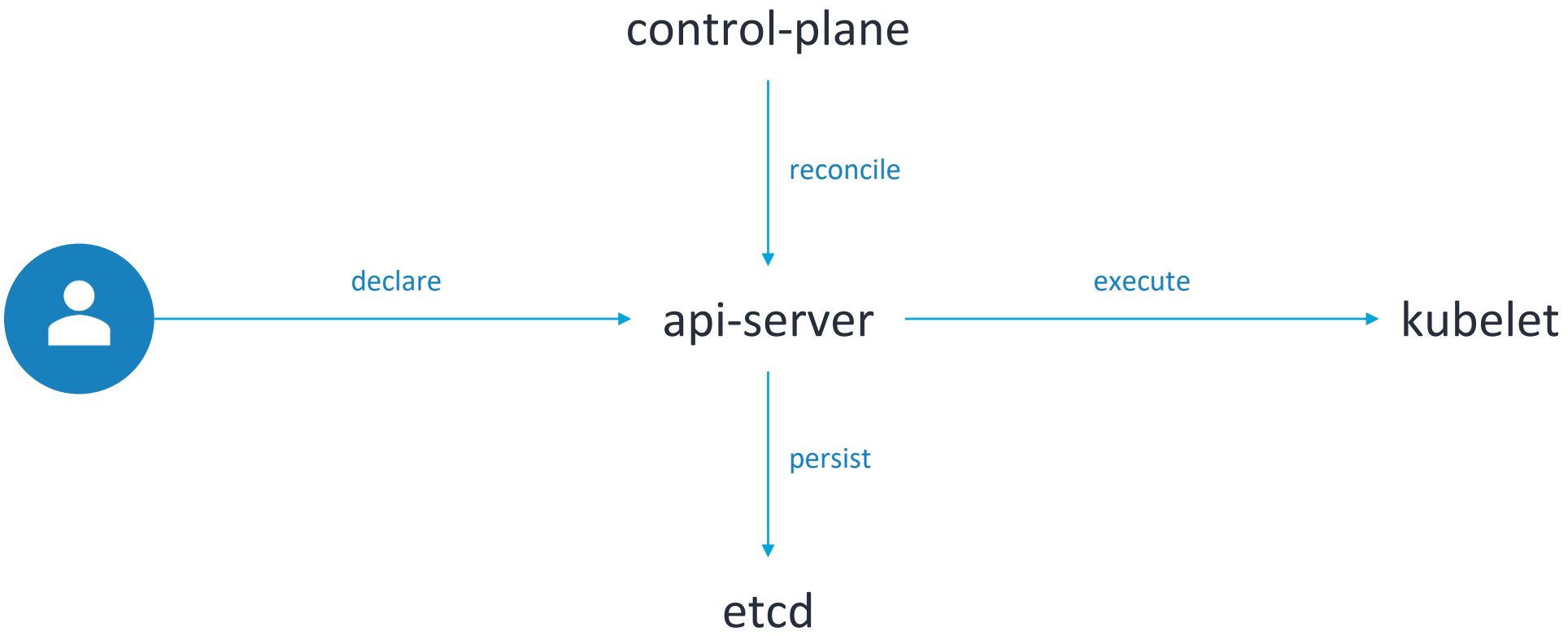
(This has some shortcomings, more on that later)

Kubernetes has built-in data types

Extensible via Custom Resource Definitions

Stackable

Kubernetes from Outer Space



Stackable

Stackable & Kubernetes

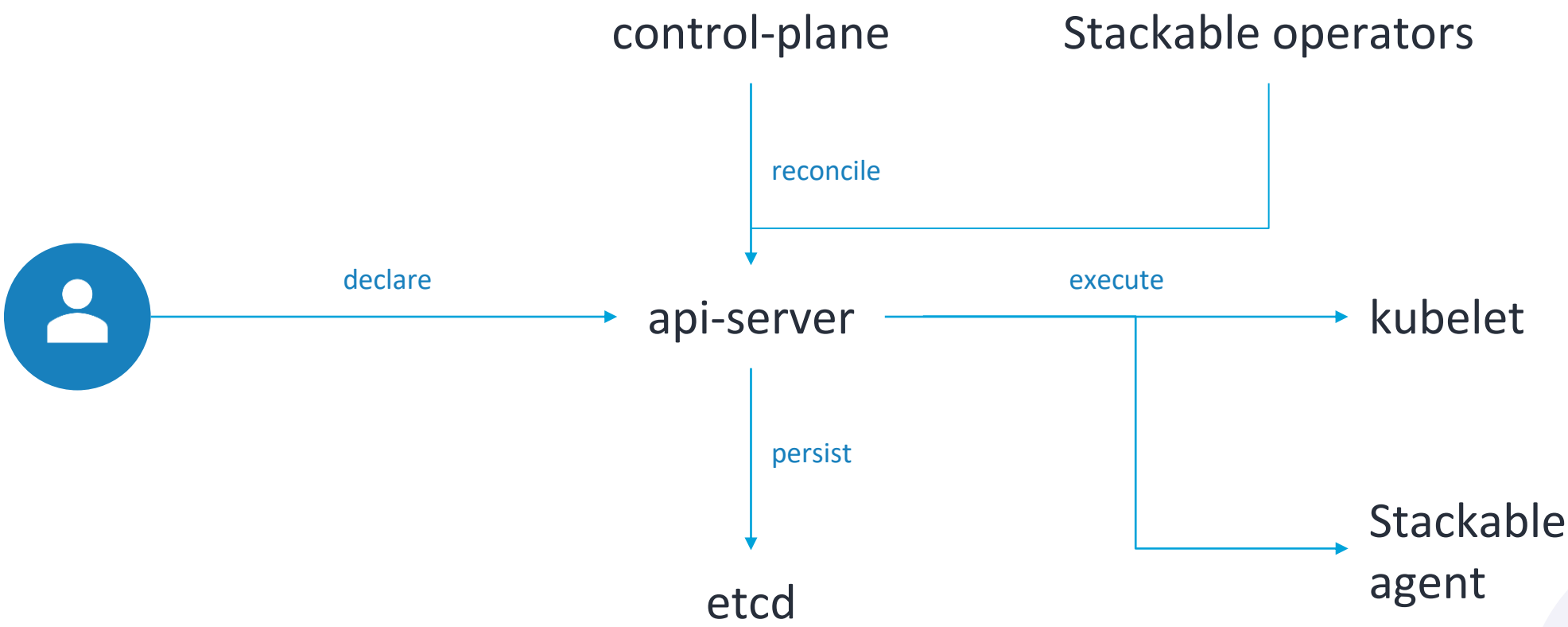
Stackable

Stackable & Kubernetes

- We wanted to build a distribution on top of Kubernetes.
- Customers told us “no”.
- So, we came up with a way to use the best of both worlds.
- Use the Kubernetes control plane and concepts (e.g. Operators,GitOps, …) but allow reusing existing admin knowledge (e.g. systemd, OS Packages, …)
- This allows hybrid scenarios:
  - Part of the stack on “bare metal”
  - Part in native Kubernetes in containers

Stackable

Stackable & Kubernetes

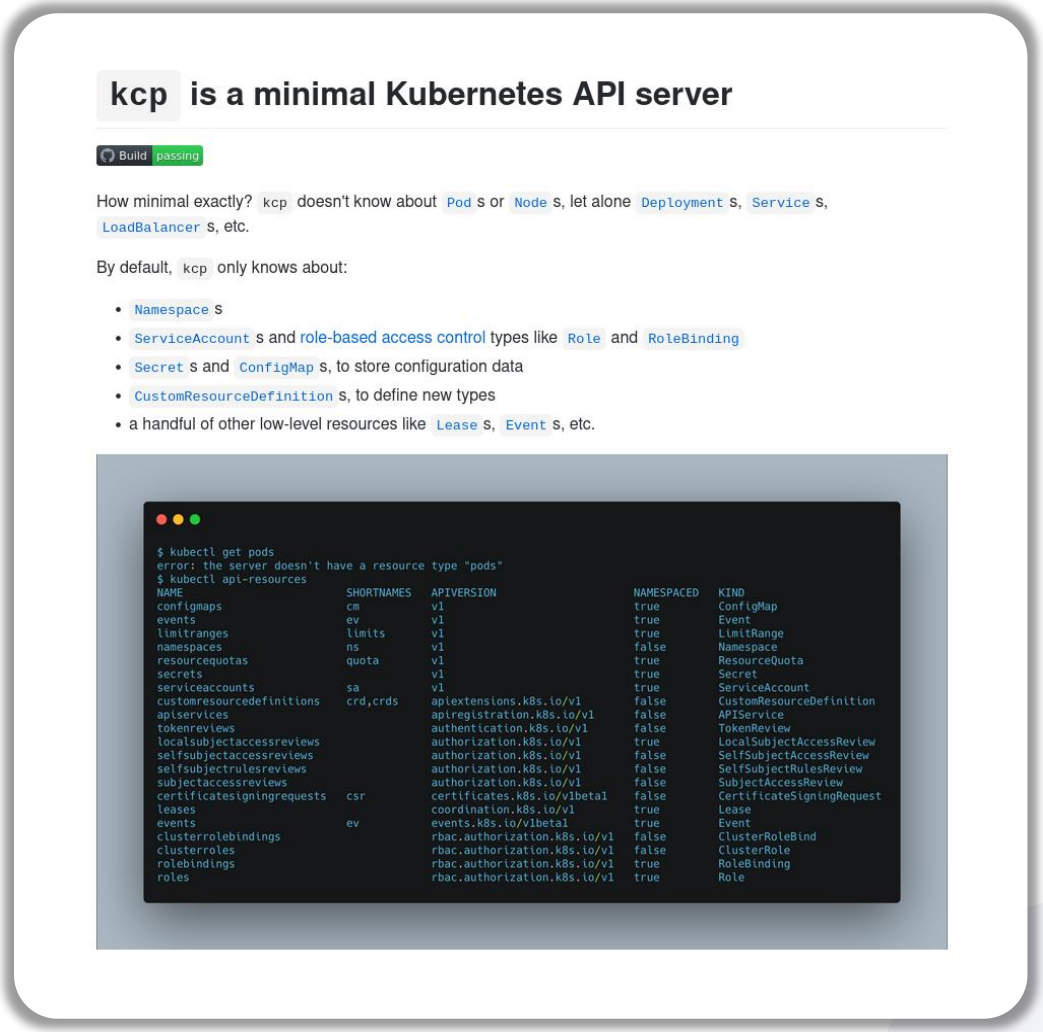


Stackable

Stackable & Kubernetes

We only use a few core concepts at the moment:

- Pod
- Node
- ConfigMap
- Secret
- …add more as needed



Stackable

Operators

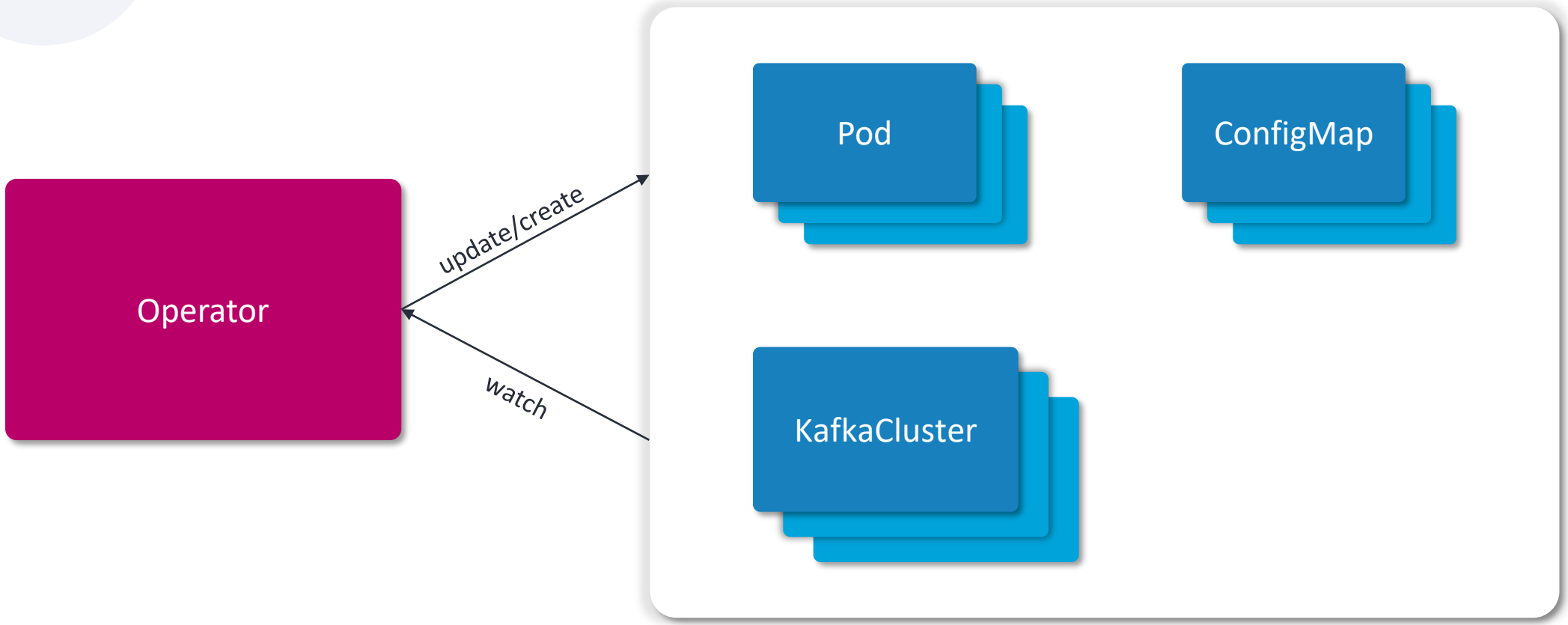
Stackable

Operator

```
let operator = extract_brain(human);
operator.run(my_cluster);
```

Stackable

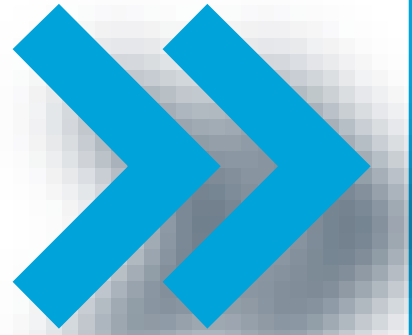
Operator



Stackable

Operator Best Practices

- Reconcile loop
- Common labels
- Common nomenclature
- Common status & events
- Common configuration/CRDs
- Common monitoring, tracing, metrics
- Hacks: Imperative commands



Our distribution should feel consistent hence we decided to write our own framework.

Stackable

Reconcile Loops – Two Alternative Styles

SMALL STEPS

```
pub fn reconcile(resource) -> ReconcileResult {

    let changed = reconcile_pod1(...)?;

    if changed {

        return ReconcileResult::Requeue;

    }


    let changed = reconcile_pod2(...)?;

    if changed {

        return ReconcileResult::Requeue;

    }

    ...

}
```

HUGE STEPS

```
pub fn reconcile(resource) -> ReconcileResult {

    reconcile_pod1().await?;

    reconcile_pod2().await?;

    ...

}
```

Stackable

Stackable
Operator Framework

Stackable

Stackable Operator Framework

Lots of convenience functions

- Retrieve existing Pods
- Retrieve/Set conditions
- Remove pods
- …

Often used Higher Level Abstractions

- Remove unneeded pods
- Find nodes which should have pods
- …

This requires a few conventions about what our objects look like!

Stackable

So, How Does It Work in Real Life?

```
self.init_status()
  .then(self.context.delete_illegal_pods(ContinuationStrategy::AllRequeue))
   .then(self.context.wait_for_terminating_pods())
   .then(self.context.wait_for_running_and_ready_pods())
   .then(self.context.delete_excess_pods(ContinuationStrategy::OneRequeue))
   .then(self.create_missing_pods())
```

Stackable

Why Rust?

Most Kubernetes code & 3rd-party operators are written in Go.
Why did you choose Rust?

 We didn't know either language

 We tried both

 We looked at the library ecosystem

 We liked Rust. End of story.

 If you want details:

- Error Handling
- Enums
- Generics
- No GC
- Security

 Go's structural typing is sweet

Stackable

Rust Kubernetes Ecosystem - k8s-openapi

This crate is a Rust Kubernetes API client. It contains bindings for the resources and operations in the Kubernetes client API, auto-generated from the OpenAPI spec.

This crate is *not* generated using Swagger or the OpenAPI Generator directly, as clients generated by the common client generator are. This gives this crate a few important advantages.

More Details on
https://github.com/Arnavion/k8s-openapi

Stackable

Rust Kubernetes Ecosystem - kube-rs

kube-rs

Rust client for Kubernetes in the style of a more generic client-go, a runtime abstraction inspired by controller-runtime, and a derive macro for CRDs inspired by kubebuilder.

More Details on
https://github.com/clux/kube-rs

GitHub

Stackable

Rust Kubernetes Ecosystem - Krustlet

Krustlet: Kubernetes Kubelet in Rust for running WASM

Krustlet 1.0 coming soon!

Krustlet acts as a Kubelet by listening on the event stream for new pods that the scheduler assigns to it based on specific Kubernetes tolerations.

The default implementation of Krustlet listens for the architecture `wasm32-wasi` and schedules those workloads to run in a `wasmtime`-based runtime instead of a container runtime.

More Details on
https://github.com/deislabs/krustlet

GitHub

Stackable

Demo Time!

Stackable

Lars Francke
lars.francke@stackable.de
+49 4103 926 3100
www.stackable.de
Contact me on

>BLN
BZZ/
WRDS

Thank You!

Stackable

Sönke Liebau
soenke.liebau@stackable.de
+49 4103 926 3100
www.stackable.de
Contact me on