Themelios: a model-checked reimplementation of Kubernetes

Andrew Jeffery, Richard Mortier
University of Cambridge

Motivation

Is Kubernetes suited to its deployments at the edge?
What architectural changes can we make?
Before making changes we need to understand what *correctness* is for Kubernetes.

Correctness

Extract → Implement → Check

Extracting properties

- No formalisation of Kubernetes, or orchestration
- Only prose documentation, or tests
- Tests don't typically cover general cases

Mapping them to a new model

- Implementing a Kubernetes model for flexibility
- Suitable for model-checking, using actor model
- Extracted properties can then be expressed

Checking the extracted properties

- Run against Kubernetes integration tests
- Can reproduce issues such as 'Stale Reads'
- Checks more traces than Kubernetes tests

Design choices

Why not an abstract model?

+ Separates specification and implementation
+ Smaller scope to check
- Hard to ensure it matches the implementation
- Different languages and expertise required

Why not a synthesizer?

- Lower performance, difficult to optimise
- Human-unfriendly code
- Trust in the toolchain
- Hard to integrate with existing codebases

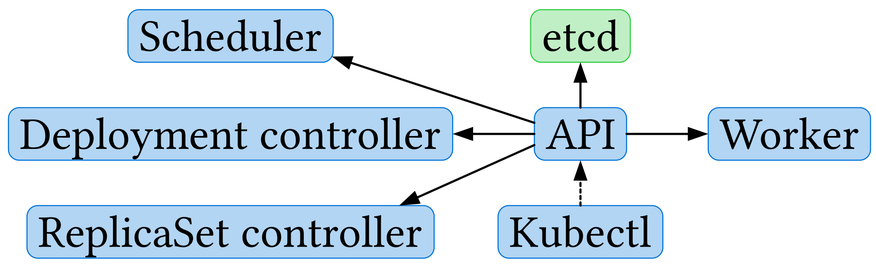
Why a reimplementation?

- More {memory-, type-, race-}safety
- Second source of understanding
- Actor model design, isolating controller logic

What is a controller?

```rust
fn step(&mut self, // local state
        gs: &GlobalState) // etcd state
        -> Option<Action>; // what to do
```

Kubernetes architecture

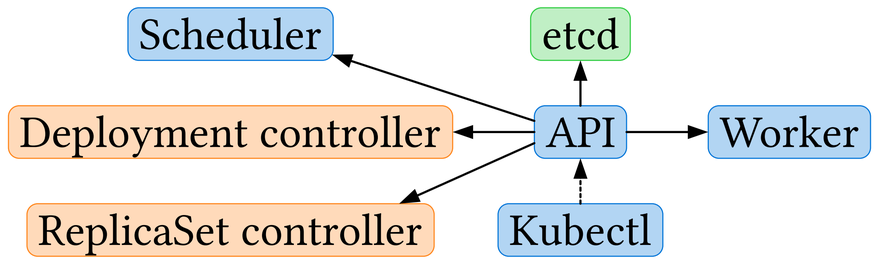


The Kubernetes architecture, Kubernetes in blue.

Model execution

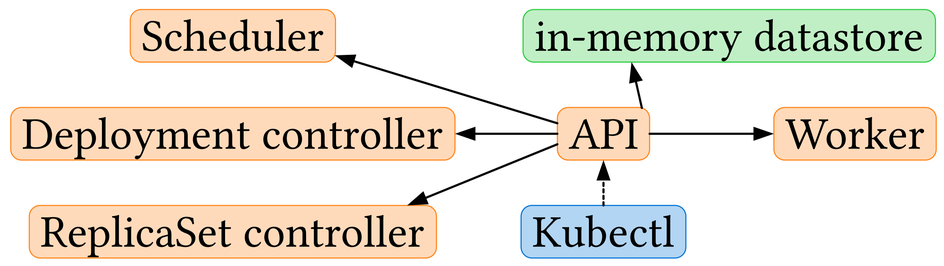
The reimplementation is in Rust and can be directly run as Kubernetes cluster controllers.

Integrate with an existing cluster



Themelios (orange) replacing the Kubernetes default controller-manager. Model-checked controllers connect directly to a live Kubernetes cluster.

Deploy a new cluster (locally)

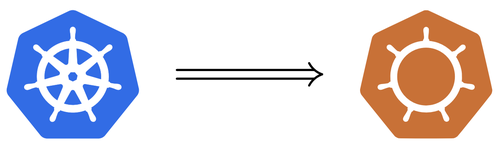


Kubectl interacting with the Themelios model-checked controllers (orange) deployed in a local setup showing viability as a standalone cluster.

Aiding in systems design

Having a more formalised model of correctness, a strong foundation, we can experiment with consistency requirements and architectures.
What guarantees are there without linearizability?
Could we use hybrid consistency?
What other shapes can we fit Kubernetes to?

Maybe we can make Kubernetes look a bit different:



Acknowledgement: This project has received funding from the EUROPEAN HEALTH AND DIGITAL EXECUTIVE AGENCY (HADEA) program under Grant Agreement No 101092950 (EDGELESS project).