



# A K8-Based Mechanism for Remote Monitoring and Control of IoT Devices

---

Sina Shabani Kumeleh

SUMMER OF 2023

# Motivation

---

# Motivation

---

- Potential capabilities inside Edge Computing
- Lack of a centralized and unified solution
- Lack of a well documented and open source solution
- Current solutions (if any) are not scalable
- And very complex

# Related Works

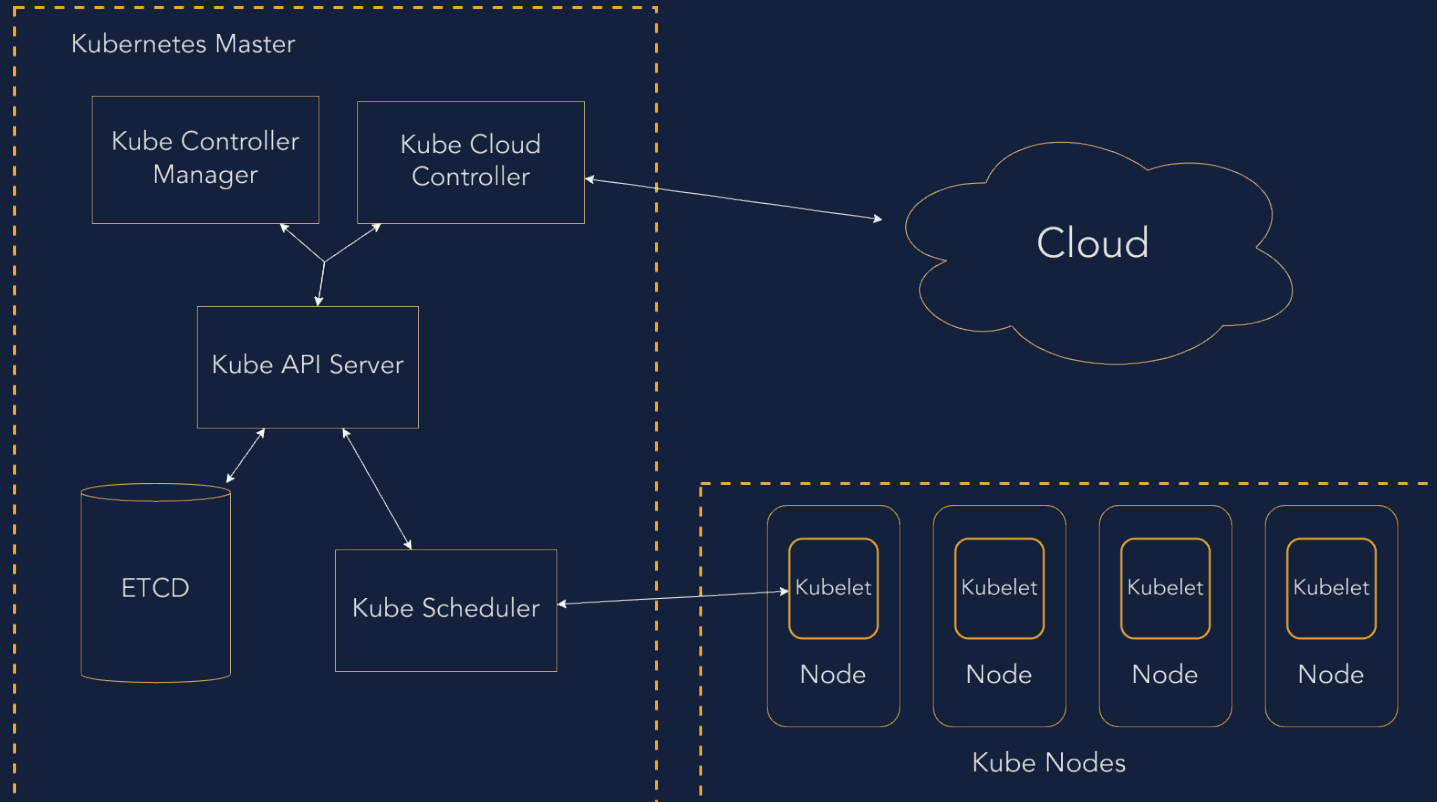
---

# Related Works - Container Technologies

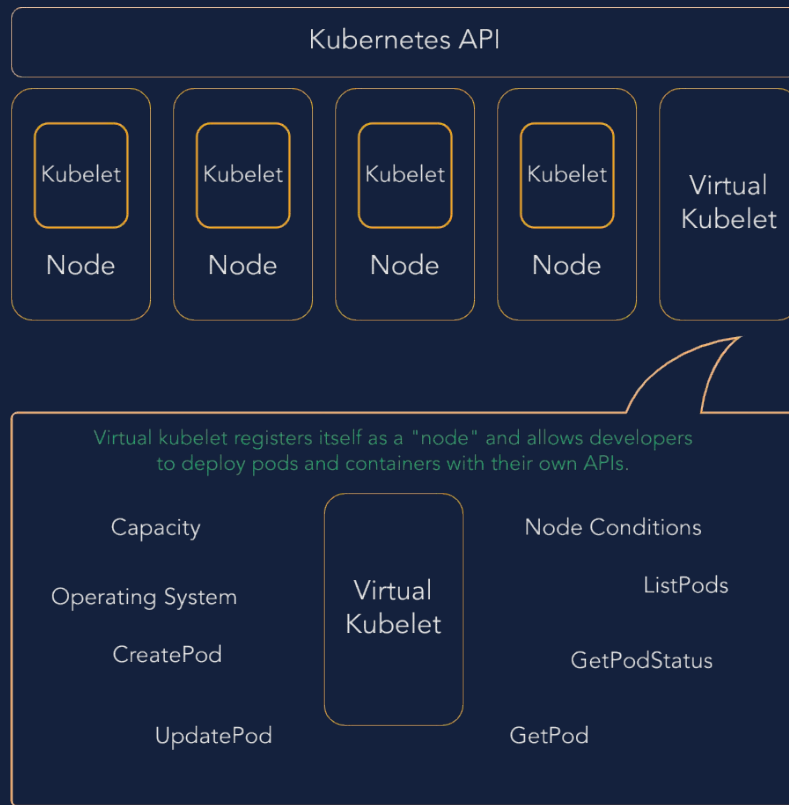
---

- Kubernetes
  - Highly scalable container platform
  - Battle-Tested
  - Fault Tolerant
  - Massive community
  - Basically the cloud we know today
- Virtual Kubelet
  - A Kubernetes node proxy
  - Registers itself as a node with a custom backend
  - Open source
  - Growing community

# Kubernetes



# Virtual Kubelet



# Design

---

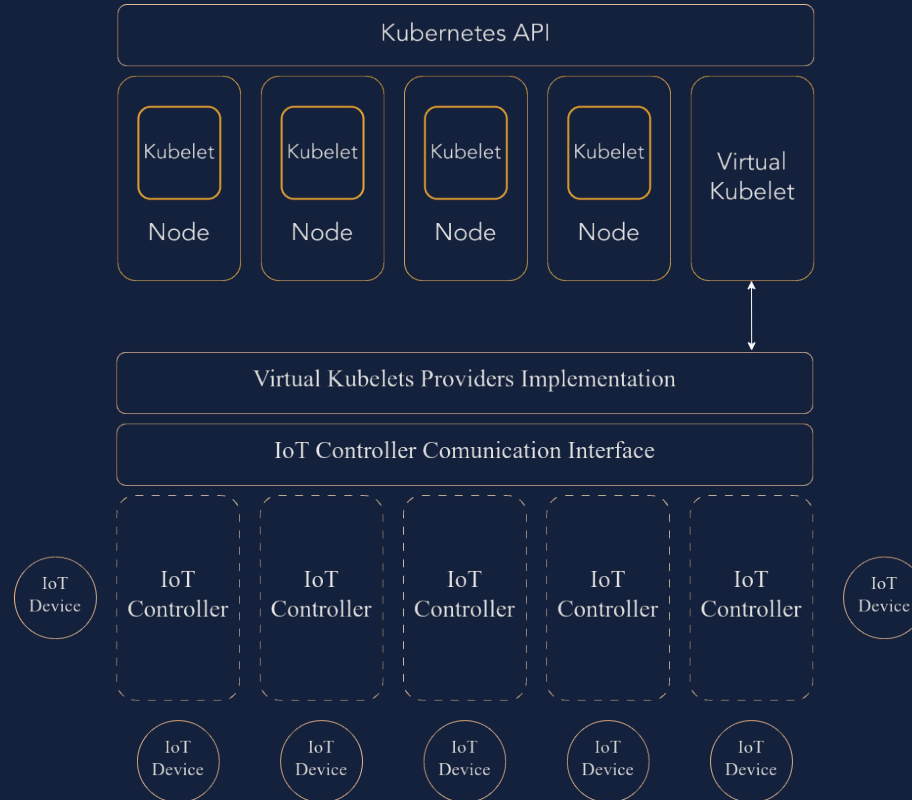


# Design

---

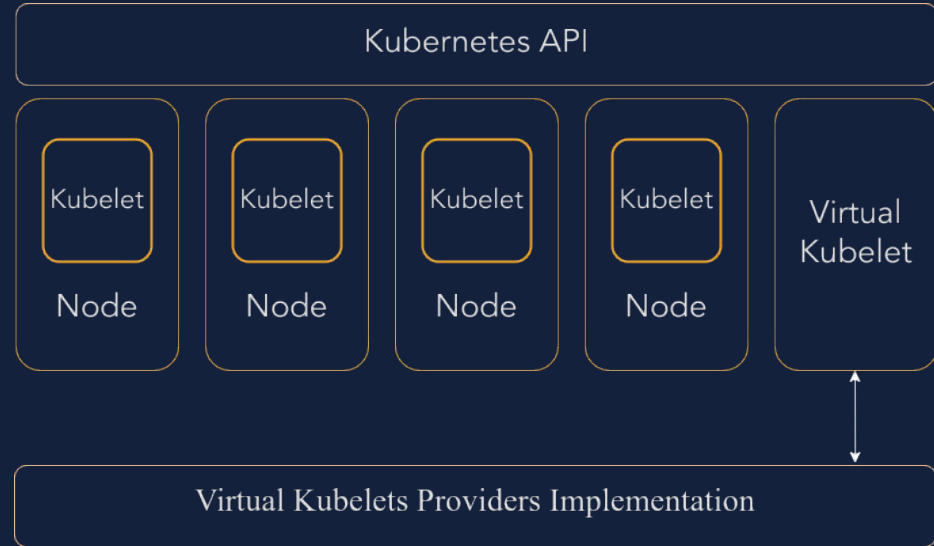
- Scalable
  - Kubernetes
  - Fan-Out design
- Simple
  - Used simple and popular transport protocols
    - HTTP
  - Used simple and popular data serialization
    - JSON
- Written in Golang
  - Massive concurrency support
  - Simple to learn
  - Massive community

# Design



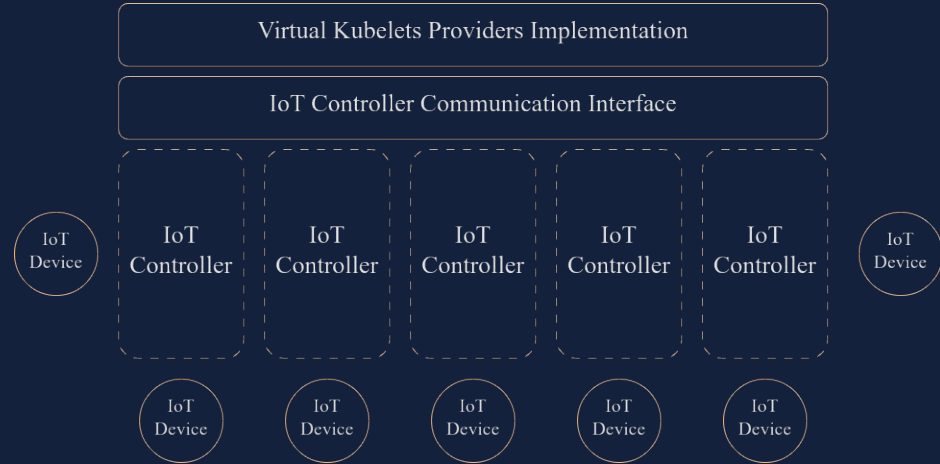
# Design - Provider

By implementing virtual kubelet's provider interface, we can have kubernetes node with a custom backend that receives kubernetes client API calls.



# Design - Communication Interface

- Provider communicates with Communication Interface through callbacks.
- Communication Interface then fetches Device states by sending HTTP requests to IoT Controllers.
- Communication Interface uses a Fan-Out technique.



# Design - Simulation Software

---


- Simple CRUD software
- Attempts to simulate a smart lock system
- Can have multiple controllers
- And multiple devices per controllers
- Written in Golang


# Design - Graphical User Interface

- Was not the the goals of this project
- However provided to show the concept better
- Simple GUI with two main part
  - One that connects to Kubernetes and shows Pods and Nodes
  - One that connects to the simulation software and shows IoT Controllers and Devices
- Written with Vuejs

# Design - Graphical User Interface




 Controllers










 Nodes





IoT Dashboard > Controllers > Controller1

Devices

ADD

Search 

Name	Readiness 	Actions
lock-main	NOT-READY	 
lock-storage	READY	 
lock-room1	READY	 
lock-room2	READY	 

Items per page: 10 1-4 of 4    

# Performance Benchmark

---



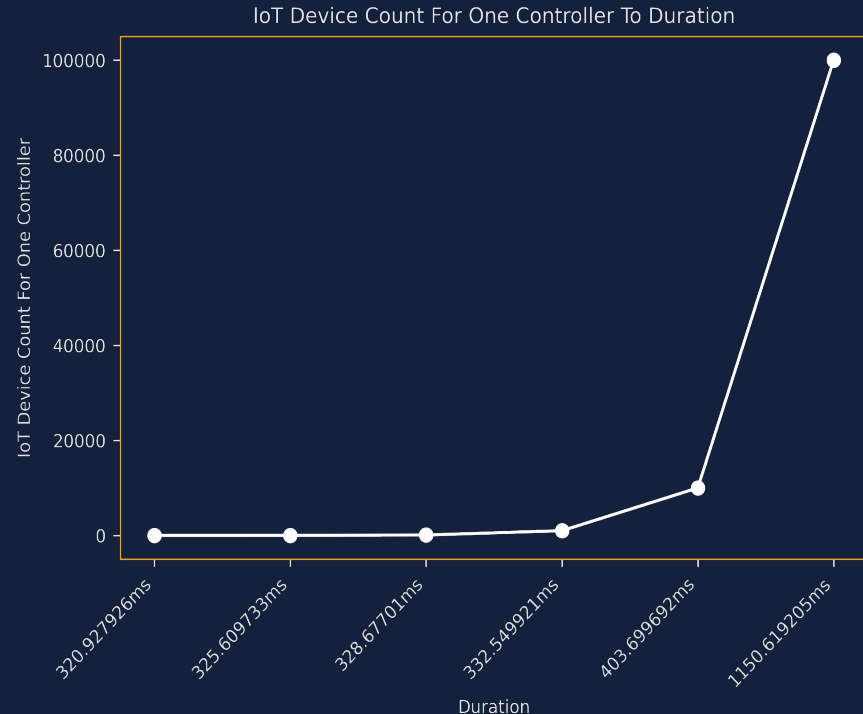
# Performance Benchmark

---

- Benchmarks are generated with custom software
- Benchmark software is written in Golang as for the rest of the project
- A shell script then takes this software and generates desired results
- The results are then fed to a python script that draws the charts using matplotlib
- The entire project ran on a single Raspberry Pi 4 Model B Rev 1.4 with 4 cores and 8GiBs of DDR3 memory.

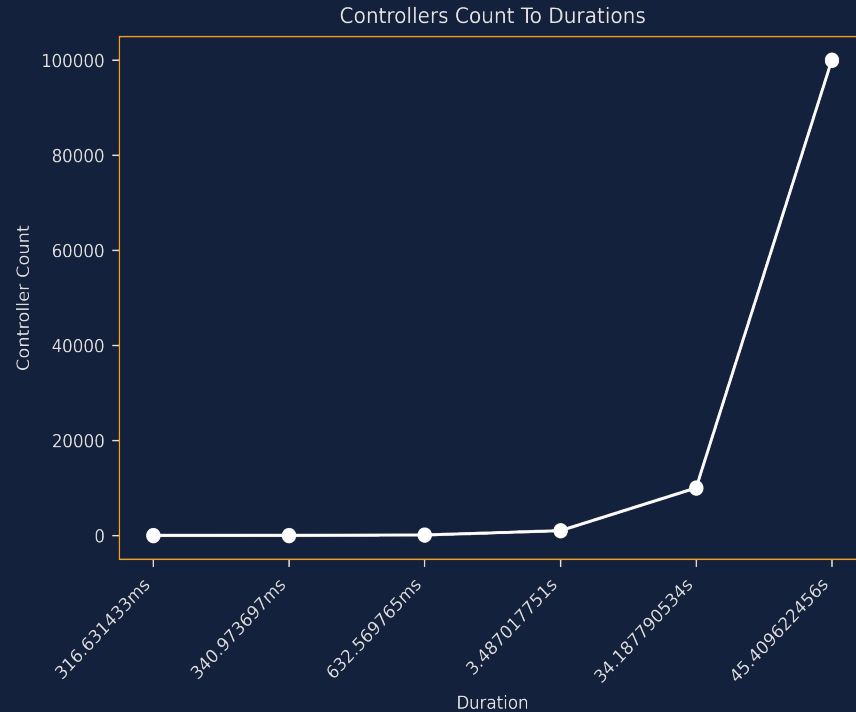
# Performance Benchmark

- Only a single provider
- As you can see with a single controller we were able to control 100K IoT Devices with reasonable time.
- It took 1.15 seconds to fetch the state of 100K devices from a single controller
- This gives us a highly scalable and performant way of controlling IoT Devices.



# Performance Benchmark

- Still a single provider
- This time we tested a single device per controllers and then changed controller's count.
- It takes Significantly more to fetch IoT Device states.
- Still reasonable.
- We have 10x HTTP traffic
- Easily scalable through adding more provider



# Future Work

---

# Future work

---

- Security
  - Authentication
  - TLS
  - WireGuard
- More performant transport protocols
  - Websockets
  - gRPC
- More performant data serialization
  - Protocol buffers
- Framework
  - Better developer experience
  - Better monetization

Any Questions?

Thanks.

