Akri

Presented by Kate Goldenring and Jiri Appl

Who We Are

Jiri Appl

Principal Software Engineer Microsoft

@jiriappl @jiria

Kate Goldenring

Software Engineer Microsoft

@kate-goldenring

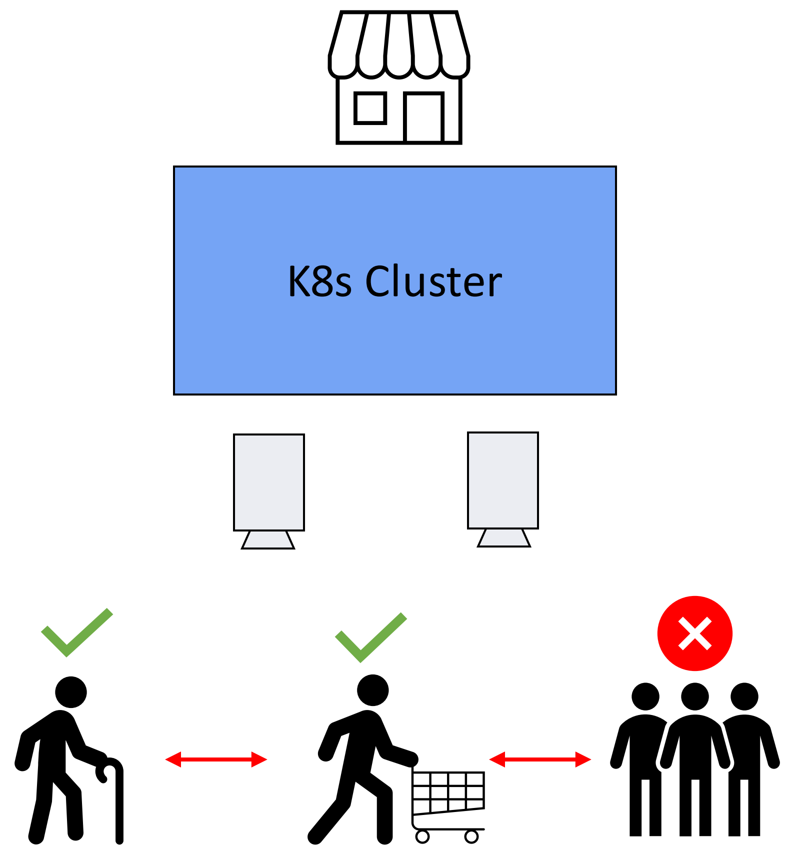
What can run Kubernetes on the edge?

1. Devices that have enough compute power to run Kubernetes effectively
2. Devices that are expected to be used as general compute

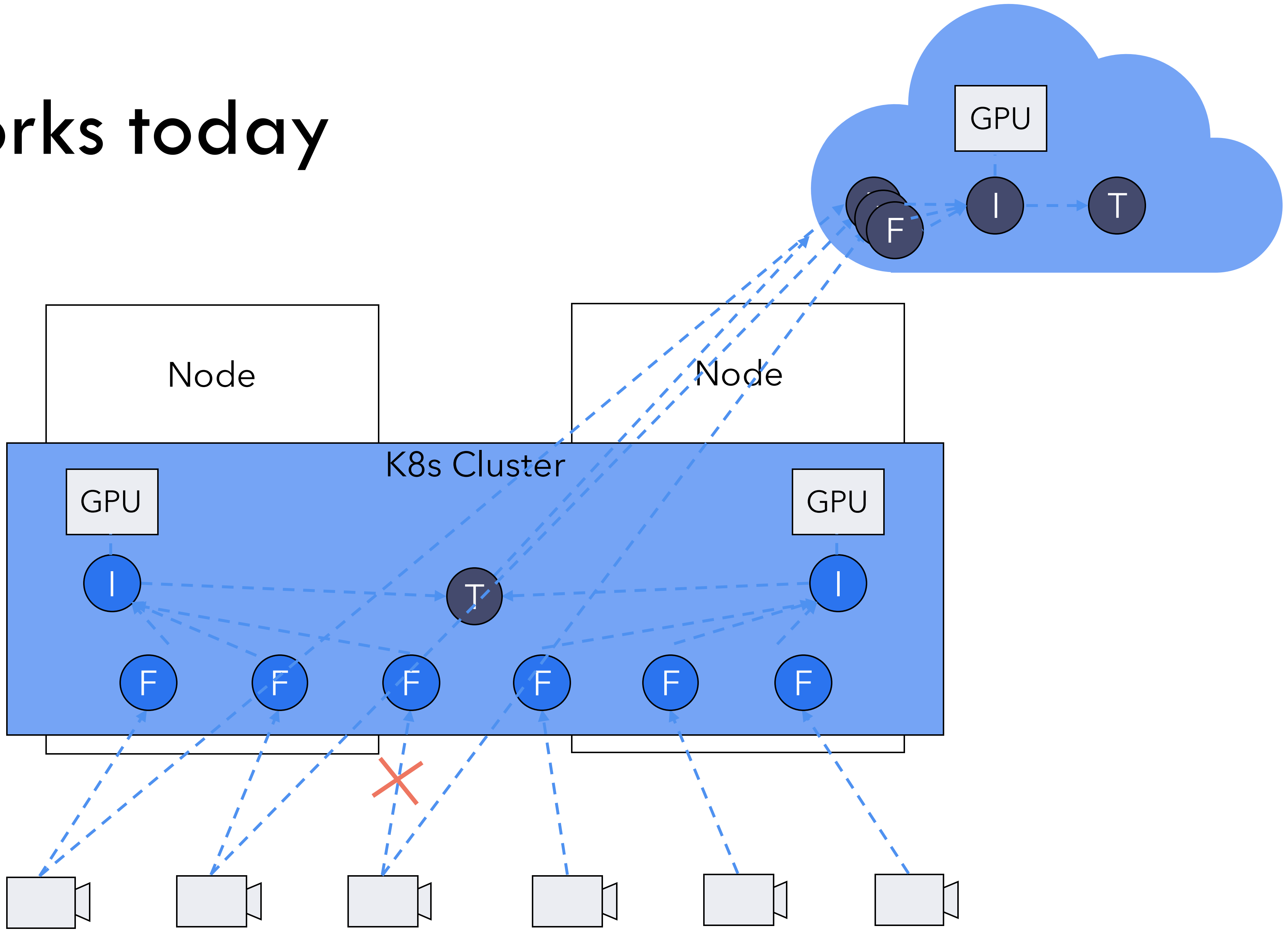
Heavy Edge	Light Edge	Tiny Edge
Server Class	Gateway Class Industrial PCs	MCU Class Smart Camera
✓	✓	?

Scenario Example

- Smart Retail
  - Existing Kubernetes Cluster, GPU capable nodes
  - IP based Cameras on the network (ONVIF)
- Kubernetes Application for Social Distancing
  - How can I direct traffic throughout my retail store to ensure social distancing?



How this works today



F	=	Frame Server Pod
I	=	Inferencing Pod
T	=	Tracking Pod

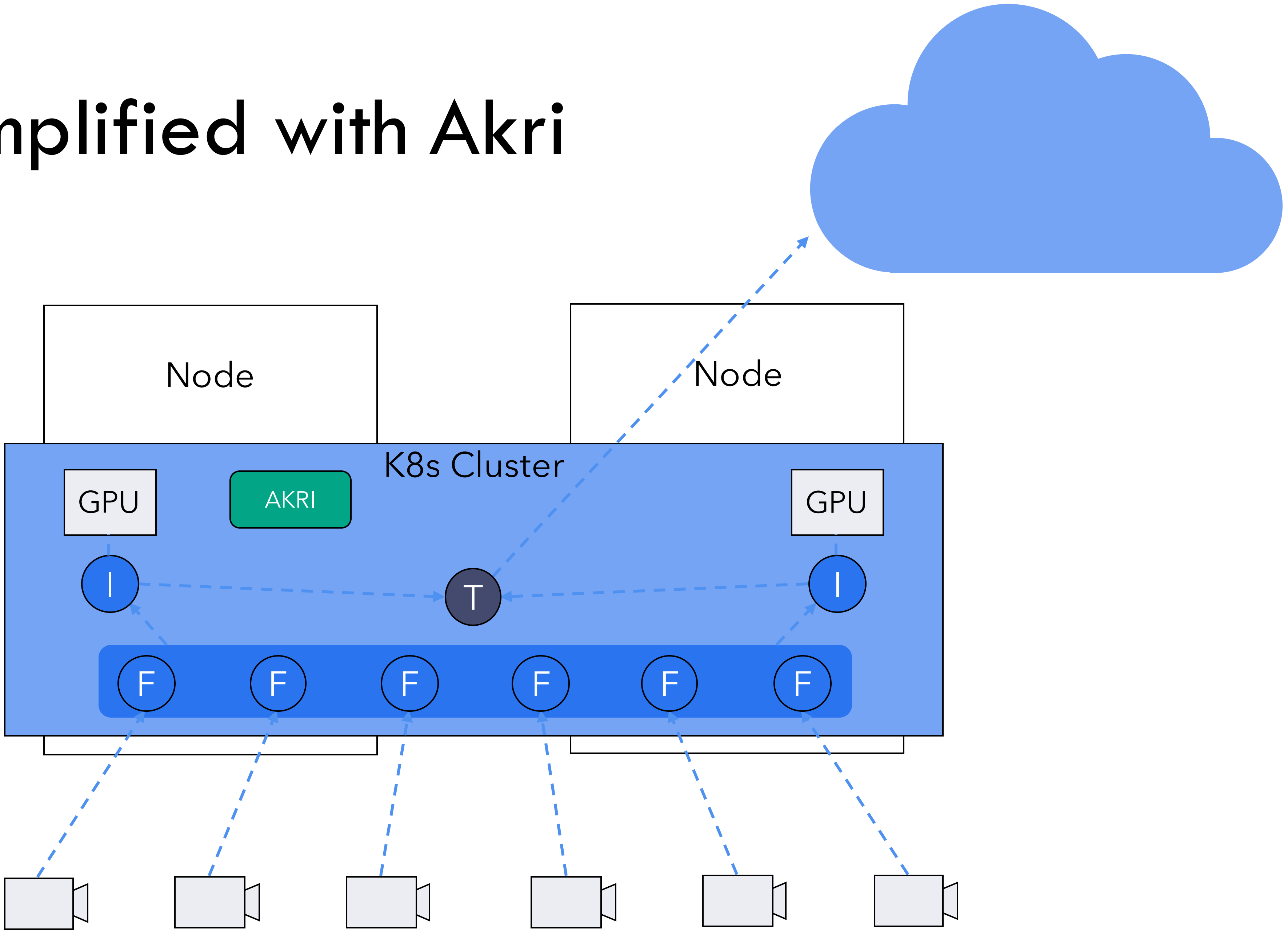
Akri – A Kubernetes Resource Interface (for the Edge)

- Open source project that attempts to define a Kubernetes based approach on representing leaf devices (such as IP cameras and USB devices) found on the edge as native Kubernetes resources
- It also supports the exposure of simple embedded hardware resources
- Schedules workloads based on the availability of leaf devices
- Utilizes Kubernetes device plugin framework
- Edge optimized for low footprint clusters- Implemented in Rust

Scenario simplified with Akri

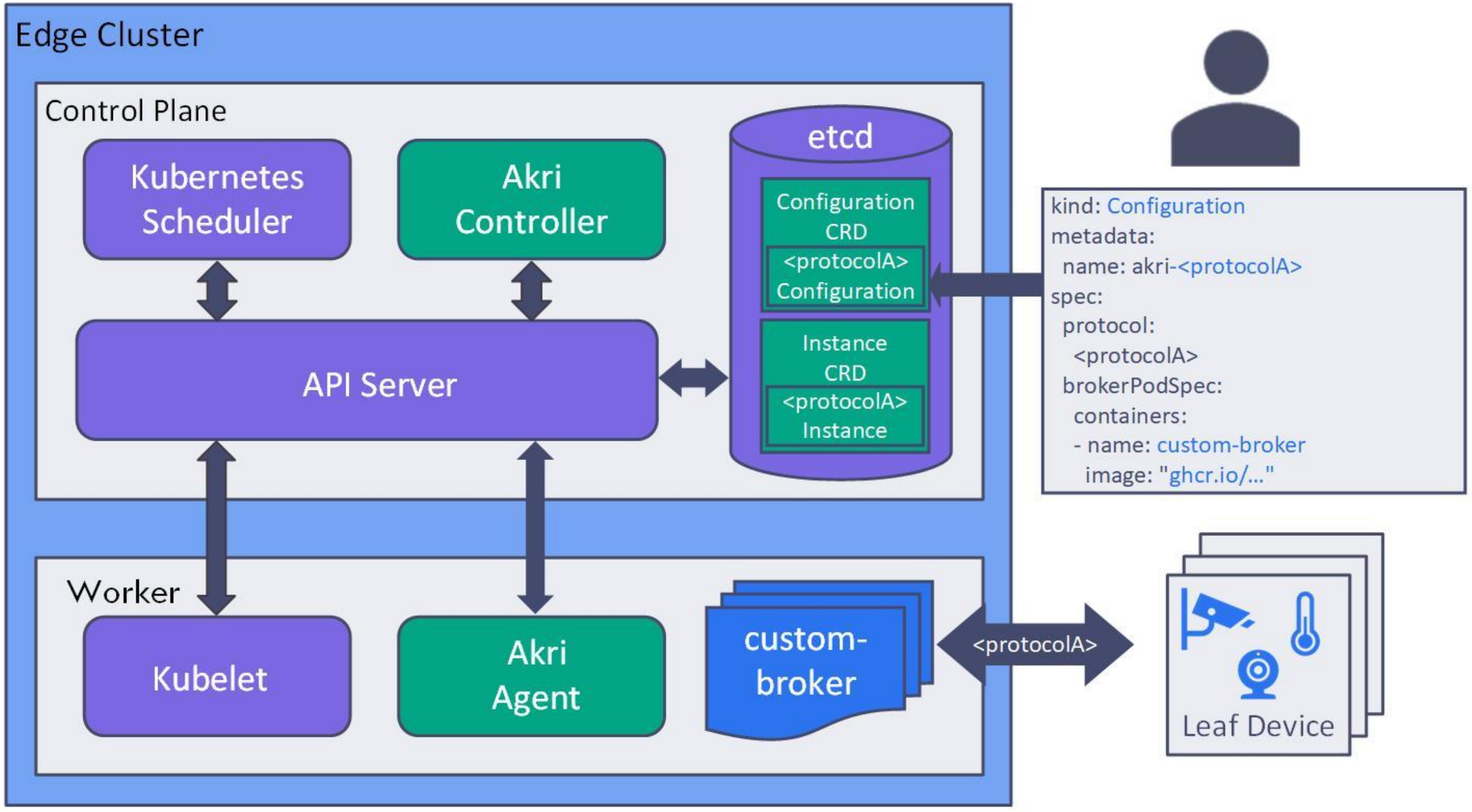
```
kind: Configuration
metadata:
  name: akri-udev-gpu
spec:
  protocol:
    udev:
  brokerPodSpec:
    containers:
    - name: akri-udev-gpu-broker
      image: "ghcr.io/.../ml-inference"
```

F	=	Frame Server Pod
I	=	Inferencing Pod
T	=	Tracking Pod



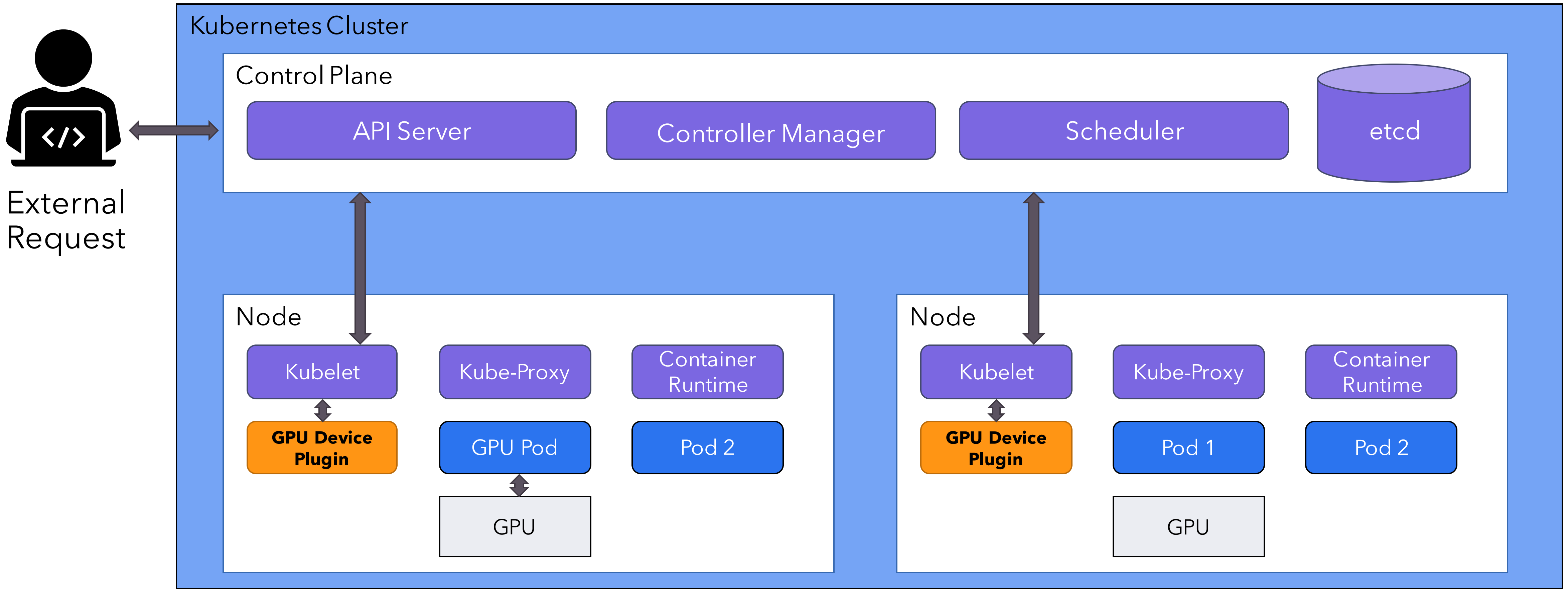
Technical Overview

Akri Architecture

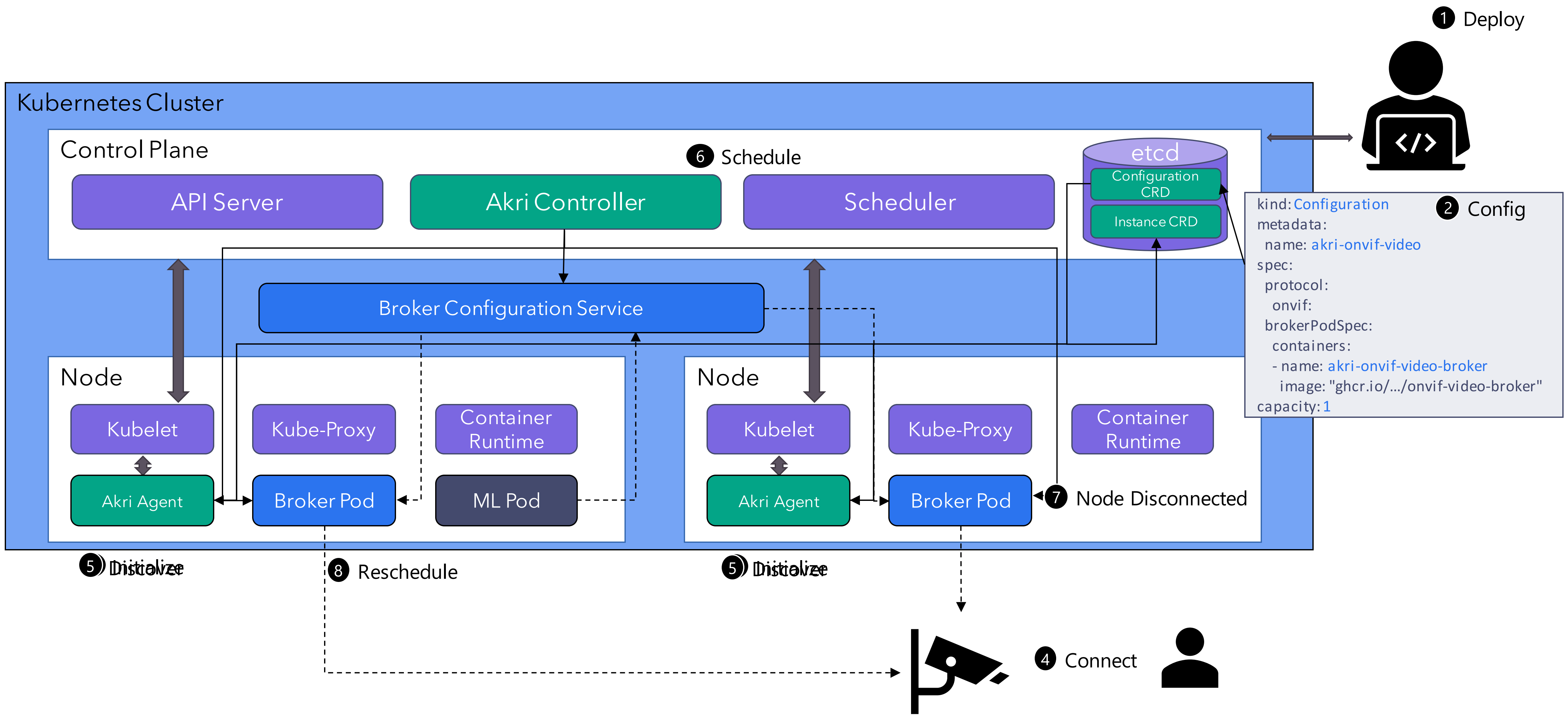


```
kind: Configuration
metadata:
  name: akri-<protocolA>
spec:
  protocol:
    <protocolA>
  brokerPodSpec:
    containers:
    - name: custom-broker
      image: "ghcr.io/..."
```

Kubernetes Device Plugin Refresher



Akri Flow



```
kind: Configuration
metadata:
  name: akri-onvif-video
spec:
  protocol:
    onvif:
  brokerPodSpec:
    containers:
    - name: akri-onvif-video-broker
      image: "ghcr.io/.../onvif-video-broker"
capacity: 1
```

Akri Functionality

Resource Interface
Abstraction for declaring what devices should be discovered

Device Discovery
Discovery of leaf devices and graceful handling of intermittent connectivity and removals

Broker Deployment
Broker deployment strategy similar to DaemonSet

High Device Availability
HA of leaf devices via "sharing" / capacity

Automated Services
Targeting specific devices or device type

Support for Multiple Protocols
Currently supported:
ONVIF and udev

Extensible
BYO protocol handlers and broker pods

Standards Based
Application specific protocol translation gateways

Akri’s Four Components

Working together so “You name it, Akri finds it, You use it”

Configuration CRD – where “you name it”

Akri Agent – how “Akri finds it”

Instance CRD – a representation of “it”

Akri Controller - helps “you use it”

Demo

Roadmap

- Additional discovery protocols and improving extensibility
- Additional deployment strategies
- Improving composability

Most importantly – We want to hear from you!

Together, we can electrify the edge for Kubernetes users

1

Try it out

End to end demo discovering mock cameras

aka.ms/akri/e2e

2

Learn more

Akri docs

aka.ms/akri/docs

3

Tell us what you think

GitHub Issues

Akri channel

aka.ms/akri/channel

4

Help us define the future for Akri

Proposals

aka.ms/akri/proposals

5

Help us discover more

Contribute

aka.ms/akri/contributing

github.com/deislabs/akri