Introduction to key technologies II: KubeEdge

Vittorio Cozzolino
E-mail: vittorio.cozzolino@huawei.com
Senior Software Engineer
Munich/Dublin Research Center

2023.11.06

DECICE Workshop
Agenda

Part 2

DECICE Workshop – Part 2

KubeEdge

KubeEdge is built upon Kubernetes and extends native containerized application orchestration and device management to hosts at the Edge. It consists of **cloud part** and **edge part**, provides core infrastructure support for networking, application deployment and metadata synchronization between cloud and edge. It also supports **MQTT** which enables edge devices to access through edge nodes.

Advantages

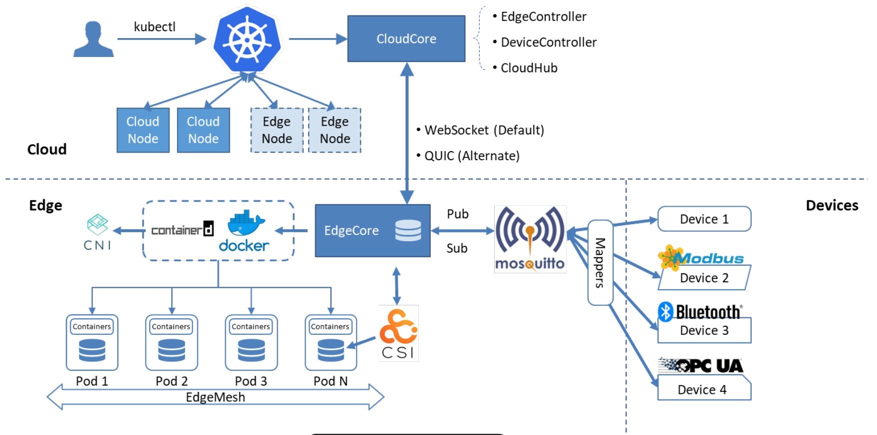
• **Kubernetes-native support**: Managing edge applications and edge devices in the cloud with fully compatible Kubernetes APIs.

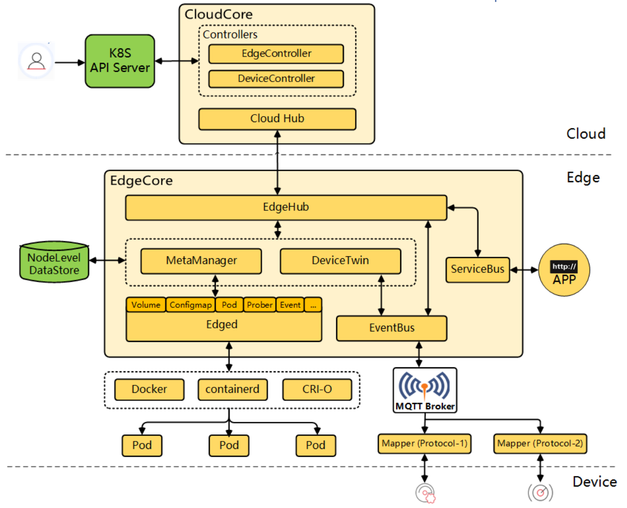
• **Cloud-Edge Reliable Collaboration**: Ensure reliable messages delivery without loss over unstable cloud-edge network.

• **Edge Autonomy**: Ensure edge nodes run autonomously and the applications in edge run normally, when the cloud-edge network is unstable or edge is offline and restarted.

• **Edge Devices Management**: Managing edge devices through Kubernetes native APIs implemented by CRD.

• **Extremely Lightweight Edge Agent**: Extremely lightweight Edge Agent(EdgeCore) to run on resource constrained edge.





DECICE Workshop – Part 2

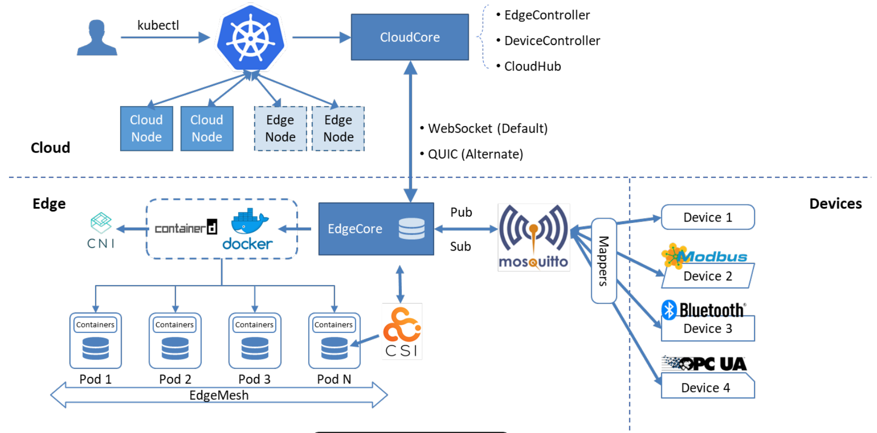
KubeEdge

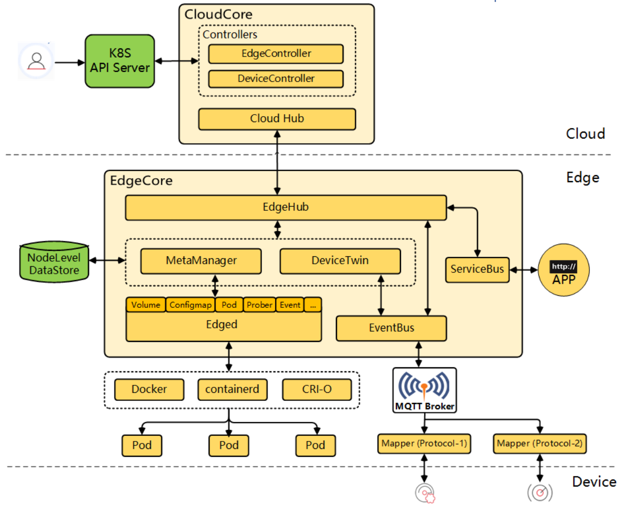
In the Cloud

- **CloudHub**: a web socket server responsible for watching changes at the cloud side, caching and sending messages to EdgeHub.
- **EdgeController**: an extended kubernetes controller which manages edge nodes and pods metadata so that the data can be targeted to a specific edge node.
- **DeviceController**: an extended kubernetes controller which manages devices so that the device metadata/status data can be synced between edge and cloud.

On the Edge

- **EdgeHub**: a web socket client responsible for interacting with Cloud Service for the edge computing (like Edge Controller as in the KubeEdge Architecture). This includes syncing cloud-side resource updates to the edge, and reporting edge-side host and device status changes to the cloud.
- **Edged**: an agent that runs on edge nodes and manages containerized applications.
- **EventBus**: a MQTT client to interact with MQTT servers (mosquitto), offering publish and subscribe capabilities to other components.
- **ServiceBus**: an HTTP client to interact with HTTP servers (REST), offering HTTP client capabilities to components of cloud to reach HTTP servers running at edge.
- **DeviceTwin**: responsible for storing device status and syncing device status to the cloud. It also provides query interfaces for applications.
- **MetaManager**: the message processor between edged and edgehub. It is also responsible for storing/retrieving metadata to/from a lightweight database (SQLite).

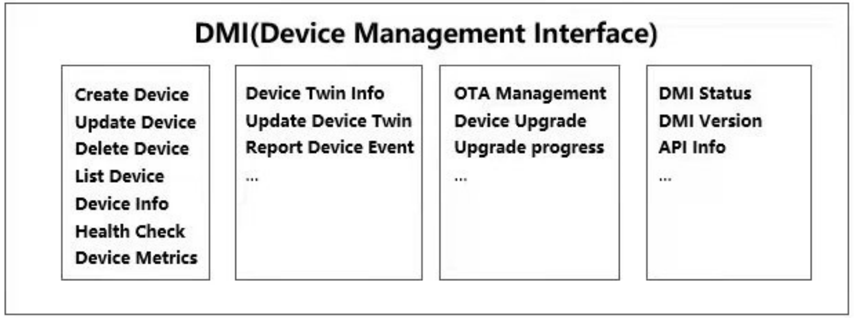




DECICE Workshop – Part 2 – Extra

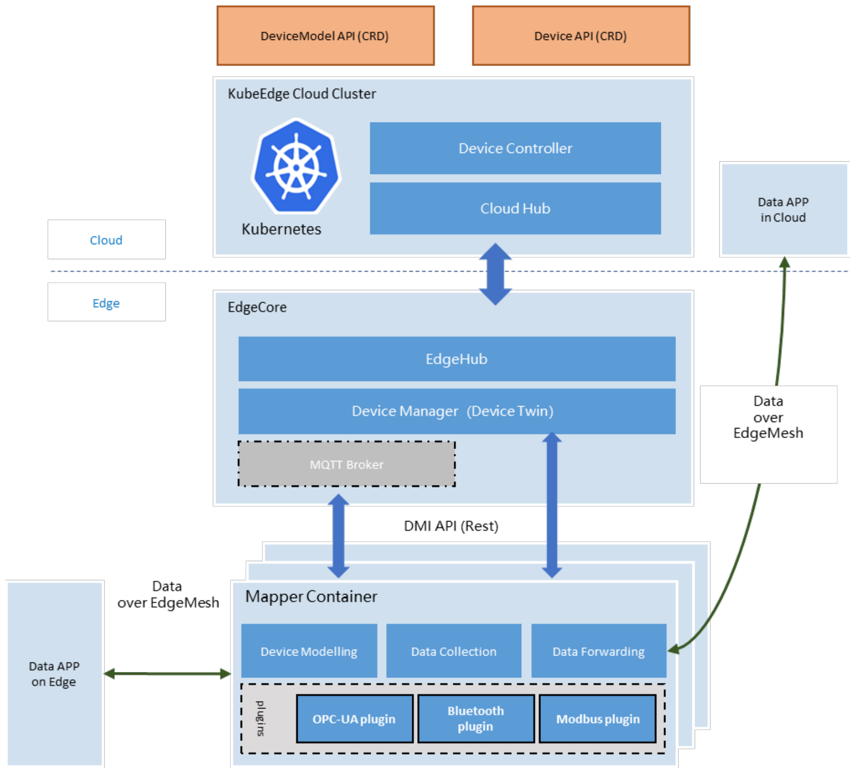
KubeEdge – Device Management Interface (DMI)

- Decoupled control plane and data plane for IoT devices
- Device (data) as a Service
- Help developers to focus on their own application development
- Reduced channel congestion between cloud and edge
- A more flexible and unified way to manage IoT Devices



DMI(Device Management Interface)

Create Device	Device Twin Info	OTA Management	DMI Status
Update Device	Update Device Twin	Device Upgrade	DMI Version
Delete Device	Report Device Event	Upgrade progress	API Info
List Device	…	…	…
Device Info			
Health Check			
Device Metrics			



DECICE Workshop – Part 2 – Extra
KubeEdge – EdgeMesh

EdgeMesh satisfies the new requirements in edge scenarios (e.g., limited edge resources, unstable edge cloud network, complex network structure, etc.), that is, high availability, high reliability, and extreme lightweight:

High availability

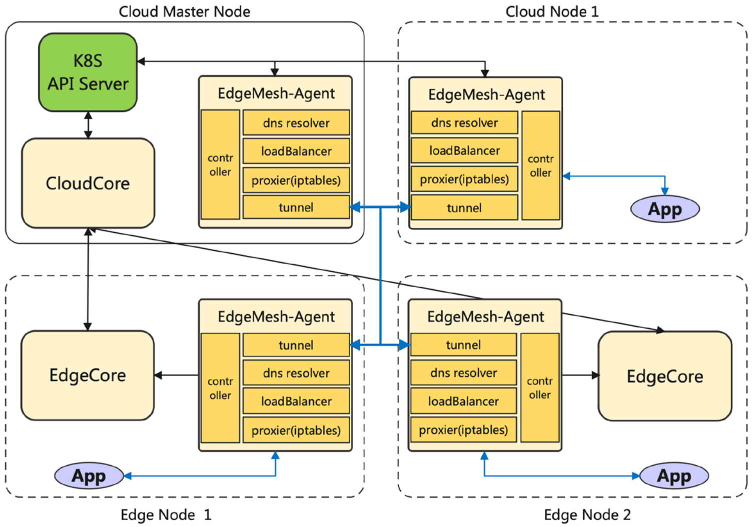
- Use the capabilities provided by LibP2P to connect the network between edge nodes
- Divide the communication between edge nodes into intra-LAN and cross-LAN

High reliability (offline scenario)

- Metadata is distributed through the KubeEdge edgehub/cloudhub tunnel, no need to access the cloud apiserver
    - EdgeMesh integrates a lightweight node-level DNS server, service discovery no longer accesses the cloud CoreDNS

Extreme lightweight

- Each node has one and only one Agent, which saves edge resources



Core Components

• **Proxier**: Responsible for configuring the kernel's iptables rules, and intercepting requests to the EdgeMesh process.
• **DNS**: Built-in DNS resolver, which resolves the DNS request in the node into a service cluster IP.
• **LoadBalancer**: Load balancer, which forwards requests to corresponding backend instances through rich load balancing strategies.
• **Controller**: Obtains metadata (e.g., Service, Endpoints, Pod, etc.) by accessing the apiserver of Kubernetes or KubeEdge.
• **Tunnel**: Based on LibP2P implementation, using automatic relay, MDNS and hole punching to provide the ability to communicate across subnets.

User value

• Enable users to have the ability to access edge-to-edge/edge-to-cloud/cloud-to-edge applications across different LANs

• Compared to the mechanism of CoreDNS + Kube-Proxy + CNI service discovery, users only need to simply deploy an Agent to finish their goals