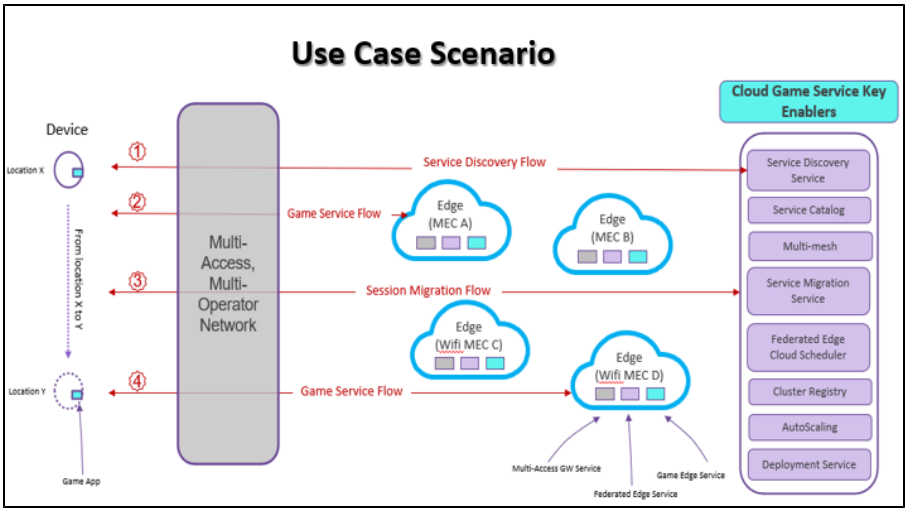
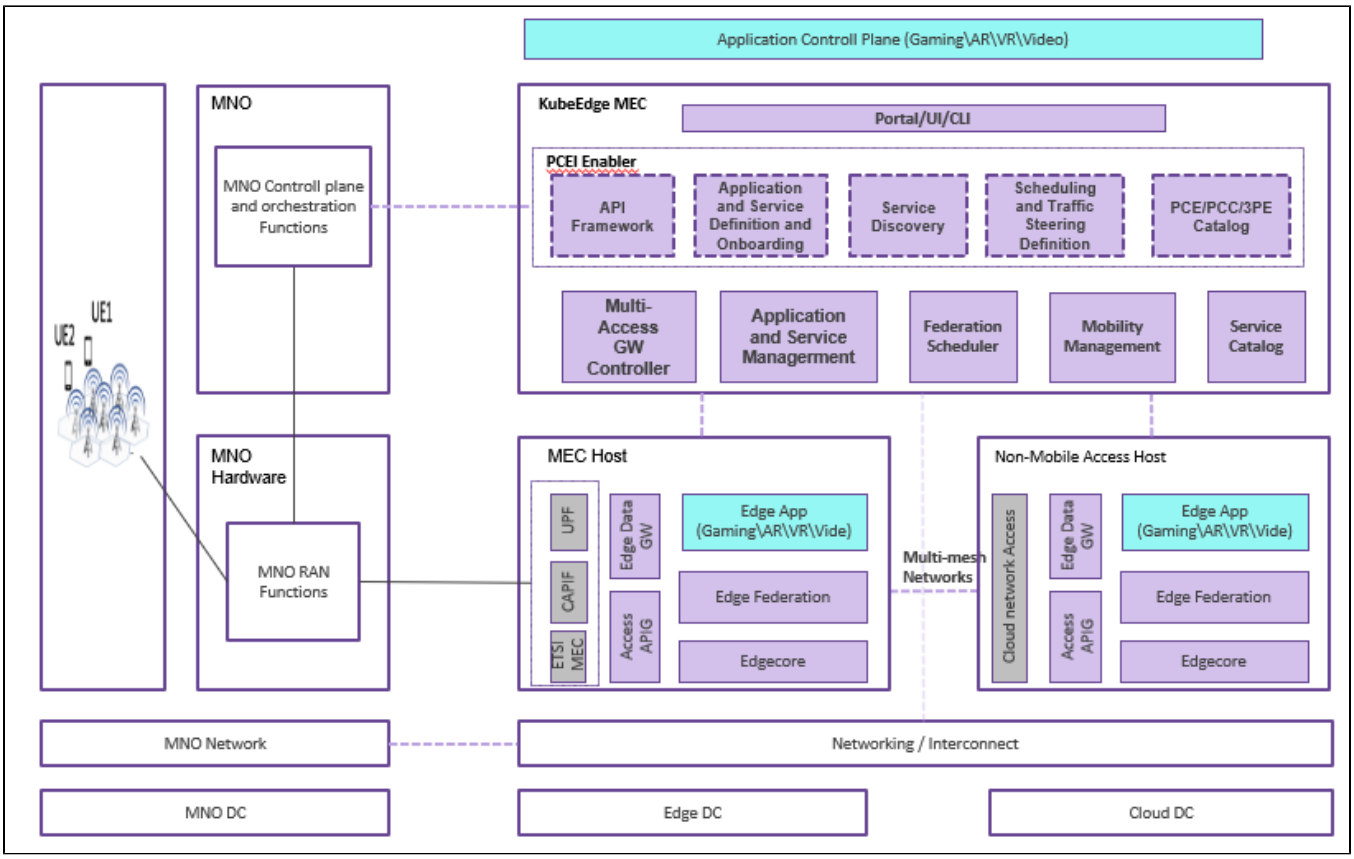
R5 Architecture Document

Introduction



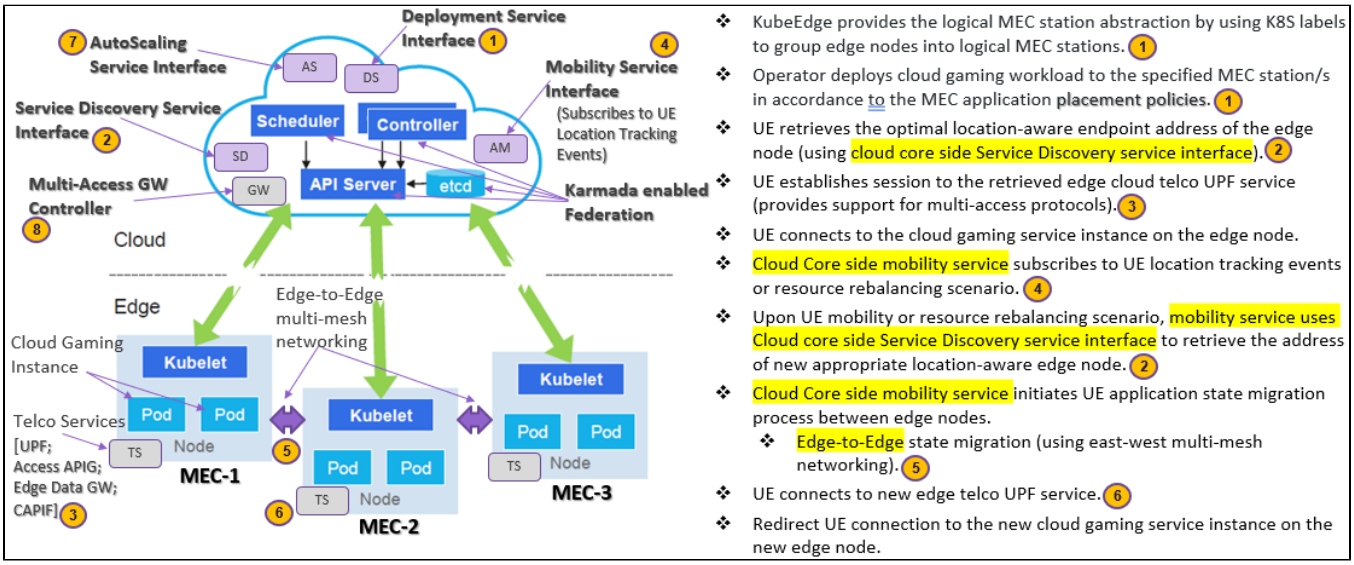
The blueprint highlights an end-to-end solution for mobile game deployed across multiple heterogeneous edge nodes using various network access protocols such as mobile and WiFi and others. This blueprint demonstrates how an application leverages a distributed and multi access network edge environment in order to get all the benefits of edge computing.

Federated MEC Cloud Platform – Functional Diagram



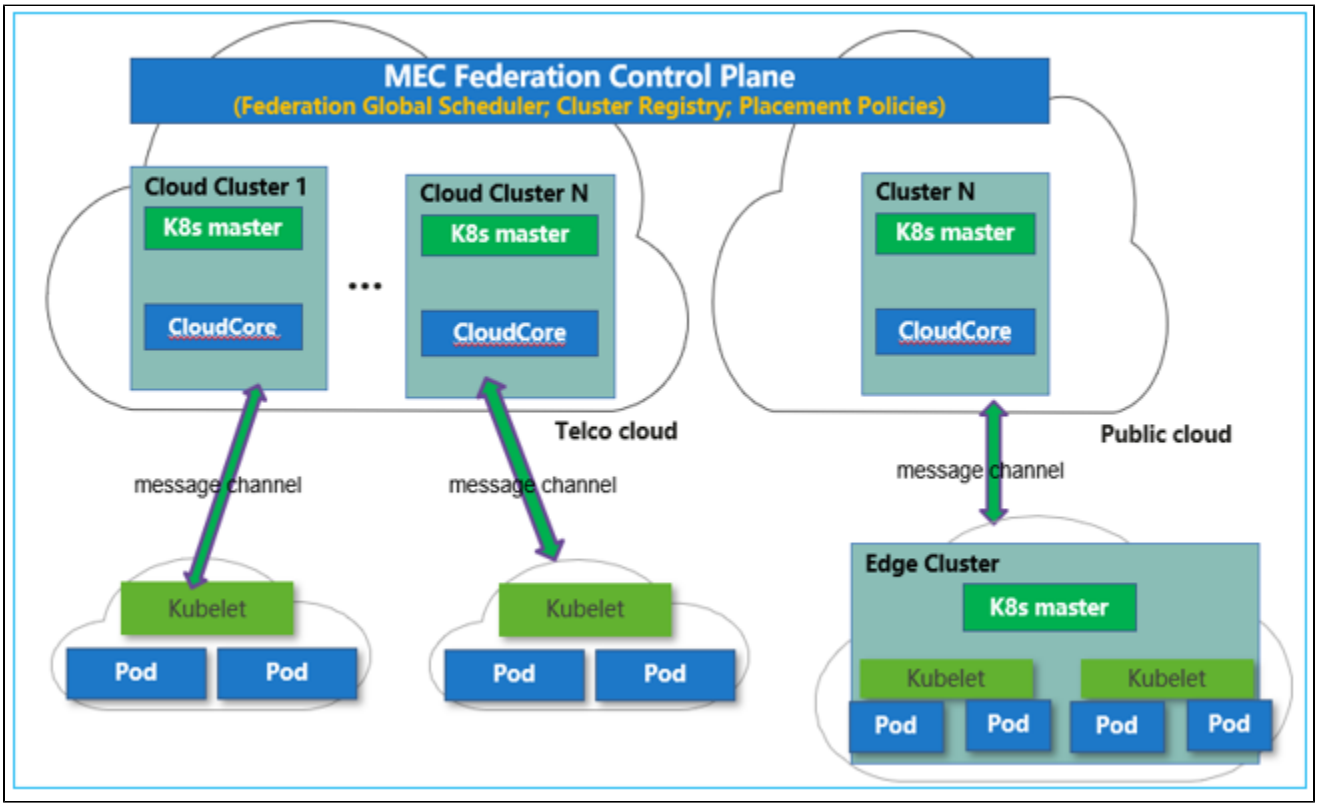
- The key component is this federated multi-access edge platform. The platform sits between applications and underlying heterogeneous edge infrastructure and also abstracts the multi-access interface and exposes application developer friendly APIs. This blueprint leverages upstream project KubeEdge as baseline platform – this includes the enhanced federation function (Karmada).
- Telco/GSMA side complexities (5GC/NEF etc.) need to be thought through and designed appropriately in order to realize extremely low latencies (10 ms) requirements desired by typical MEC use cases.
- For the multi access, we may initially use a simulated mobile access environment to mimic a real time device access protocol conditions as part of the initial release/s.

Cloud Gaming Detail Flow



- KubeEdge provides the logical MEC station abstraction by using K8S labels to group edge nodes into logical MEC stations. 1
- Operator deploys cloud gaming workload to the specified MEC station/s in accordance to the MEC application placement policies. 1
- UE retrieves the optimal location-aware endpoint address of the edge node (using cloud core side Service Discovery service interface). 2
- UE establishes session to the retrieved edge cloud telco UPF service (provides support for multi-access protocols). 3
- UE connects to the cloud gaming service instance on the edge node.
- Cloud Core side mobility service subscribes to UE location tracking events or resource rebalancing scenario. 4
- Upon UE mobility or resource rebalancing scenario, mobility service uses Cloud core side Service Discovery service interface to retrieve the address of new appropriate location-aware edge node. 2
- Cloud Core side mobility service initiates UE application state migration process between edge nodes.
  - Edge-to-Edge state migration (using east-west multi-mesh networking). 5
- UE connects to new edge telco UPF service. 6
- Redirect UE connection to the new cloud gaming service instance on the new edge node.

Federated KubeEdge (Karmada) - Hard Multitenancy using multiple autonomous K8S clusters

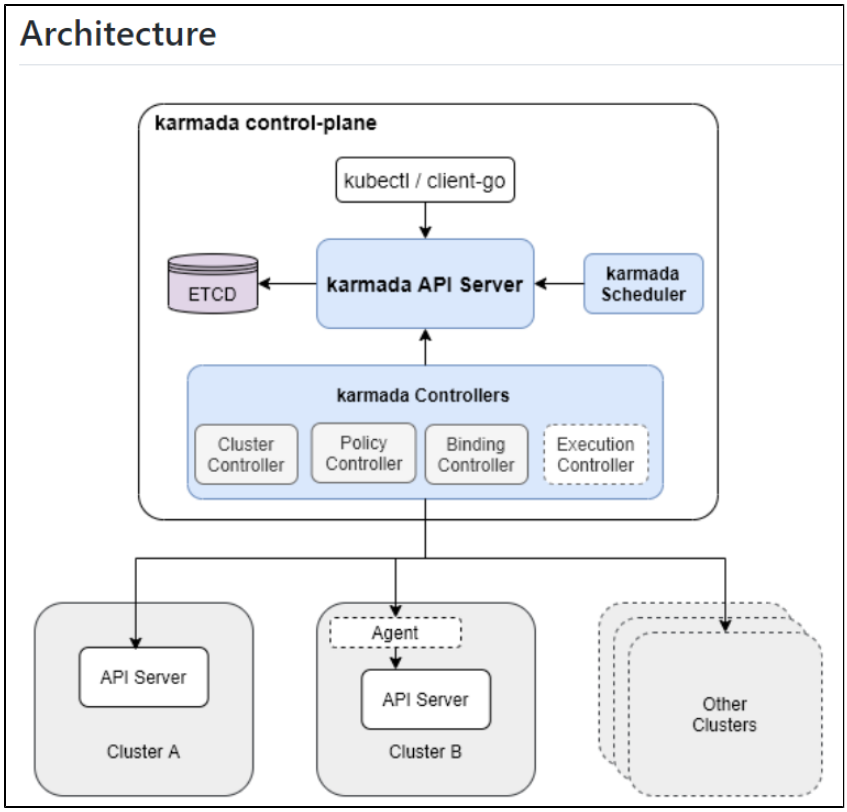


Key Enabling Architectural Components

- **Federation Scheduler (Included in Release-5)**: As a “Global Scheduler”, responsible for application QoS oriented global scheduling in accordance to the placement policies. Essentially, it refers to a decision-making capability that can decide how workloads should be spread across different clusters similar to how a human operator would. It maintains the resource utilization information for all the MEC edge cloud sites. Cloud federation functionality in our blueprint is enabled using open source Karmada project.

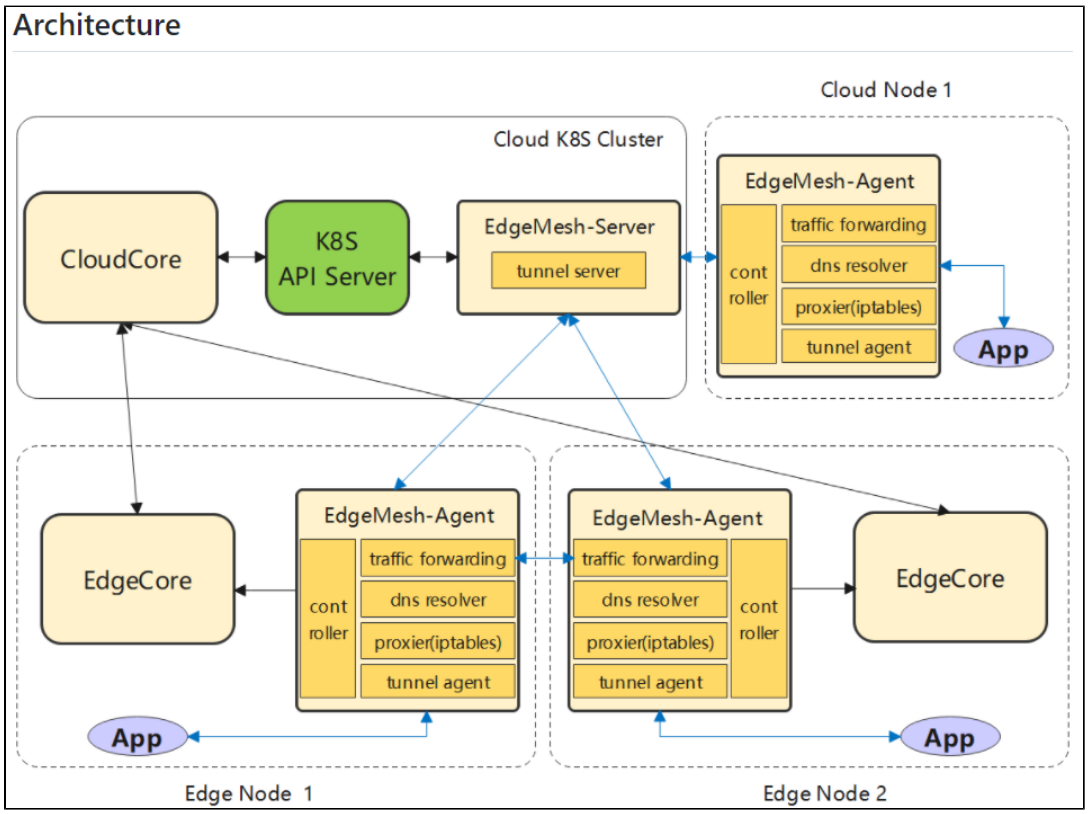
Karmada (Kubernetes Armada) is a Kubernetes management system that enables cloud-native applications to run across multiple Kubernetes clusters and clouds with no changes to the underlying applications. By using Kubernetes-native APIs and providing advanced scheduling capabilities, Karmada truly enables multi-cloud Kubernetes environment. It aims to provide turnkey automation for multi-cluster application management in multi-cloud and hybrid cloud scenarios with key features such as centralized multi-cloud management, high availability, failure recovery, and traffic scheduling.

The following is an architecture diagram for Karmada. More details related to Karmada project can be found here.



- **EdgeMesh (Included in Release-5)**: Provides support for service mesh capabilities for the edge clouds in support of microservice communication cross cloud and edges. EdgeMesh provides a simple network solution for the inter-communications between services at edge scenarios (east-west communication). The network topology for edge cloud computing scenario is quite complex.

Various Edge nodes are often not interconnected and the direct inter-communication of traffic between applications on these edge nodes is highly desirable requirement for businesses. EdgeMesh addresses these challenges by shielding the complex network topology at the edge applications scenario. The following is a architecture diagram for Karmada. More details related to EdgeMesh project can be found here.



- **Service Discovery**: Retrieves the endpoint address of the edge cloud service instance depending on the UE location, network conditions, signal strength, delay, App QoS requirements etc.
- **Mobility Management**: Cloud Core side mobility service subscribes to UE location tracking events or resource rebalancing scenario. Upon UE mobility or resource rebalancing scenario, mobility service uses Cloud core side Service Discovery service interface to retrieve the address of new appropriate location-aware edge node. Cloud Core side mobility service subsequently initiates UE application state migration process between edge nodes. Simple CRIU container migration strategy may not be enough, much more complex than VM migration.
- **Multi-Access Gateway:** Multi access gateway controller manages Edge Data Gateway and Access APIG of edge nodes. Edge data gateway connects with edge gateway (UPF) of 5g network system, and routes traffic to containers on edge nodes. Access APIG connects with the management plane of 5g network system (such as CAPIF), and pulls QoS, RNIS, location and other capabilities into the edge platform.
- **AutoScaling**: Autoscaler provides capability to automatically scale the number of Pods (workloads) based on observed CPU utilization (or on some other application-provided metrics). Autoscaler also provides vertical Pod autoscaling capability by adjusting a container's ”CPU limits” and ” memory limits” in accordance to the autoscaling policies.
- **Service Catalog**: It provides a way to list, provision, and bind with services without needing detailed knowledge about how those services are created or managed.