Hummingbird –Small node management of StarlingX

Mingyuan Qi, Cloud Software Engineer, Intel IoT Group

1 Background

StarlingX aims to provide a platform for edge and IoT use cases with tight resource constraints. However, the minimum hardware requirement of a simplex deployment is Intel® Xeon® D-15xx family (8 cores) with 64G Ram. This requirement is too high to fit into the IoT edge nodes with limited hardware resources.

In a typical edge use case, StarlingX usually plays a role as the edge servers located in the server rooms Figure 1 is the node diagram of a typical edge use case.

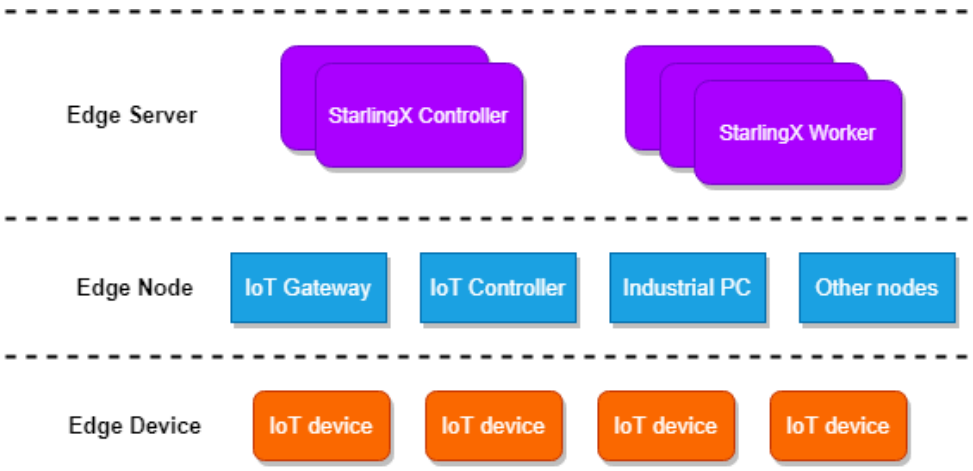

Figure 1 – Node diagram of a typical edge use case

The edge nodes are not managed by StarlingX in this diagram, and the edge applications running on the edge nodes are not orchestrated by StarlingX either. There are 3 setbacks for the edge nodes' manageability of StarlingX:

1. The edge nodes have limited hardware resources, e.g. Intel® Core™ or Intel Atom® and less than 32GB memory.
2. The edge nodes are physically close to the edge devices and are connected to the servers across IT network. This means the network connection between StarlingX controllers and the edge nodes are neither guaranteed "always online", nor within the same subnet.
3. The operating systems of the edge nodes are provided by 3rd party vendor and/or managed under IT administration.

In order to enable the edge node management for StarlingX, a project, code name: Hummingbird, is introduced to resolve the underlying setbacks throughout the whole software stack of StarlingX.

2 Hummingbird

Hummingbird is the code name of the project which brings the ability of edge node management to StarlingX. The edge nodes are typically small footprint nodes like IoT gateway, IoT controller, industrial PC, etc. In Hummingbird's node management concept, the edge nodes are loosely managed by StarlingX controllers.

Loose management represents a management method with less intervention during the nodes' software lifecycle, from bootstrap to application orchestration. In the bootstrap process, the edge nodes are either bootstrapped by the preinstalled out-of-box system, so called BYOD (Bring your own device), or installed with a custom system profile by the installation tools. Both approaches to bootstrap an edge node are out of StarlingX controller's control.

During the daily operation, the manageability of the nodes is not always available because of the uncertain stability of IT network. In terms of the loose management, it is acceptable and tolerated if the nodes could self-maintain the status of the applications with the other nodes in a same federation. A federation is a group of nodes working on the same process. StarlingX monitors nodes' heartbeat to ensure their high availability, the heartbeat lost will lead to various alarms and the nodes will be recovered automatically.

Table 1 shows the comparison of "tight management" of current StarlingX implementation and "loose management" proposed by Hummingbird.

Perspective from nodes	Tight Management (StarlingX default)	Loose Management (Hummingbird)
Bootstrap/ Installation	By controller via mgmt. network	BYOD*/ By independent installation tool
Provision	By controller via mgmt. network	By controller via mgmt. network
Monitor	High frequent heartbeat	Low frequent heartbeat
Networking	L2 network	L2 or L3 network
OS	Centos	Any container available Linux
Services	Flock services** + Kubernetes	Partial containerized flock services on Kubernetes

* BYOD: Bring Your Own Device

** Flock services: StarlingX specific services for system management

Table 1 - Tight management VS. loose management

In this chapter, the following items will be described in detail:

- Personality: EdgeWorker
- Networking of the EdgeWorker nodes
- Provisioning EdgeWorker
- EdgeWorker manageability
- EdgeWorker group storage
- Edge application orchestration

2.1 Personality: EdgeWorker

A new personality “EdgeWorker” will be added to identify the edge nodes which is different from the StarlingX “Worker” nodes. Figure 2 shows the EdgeWorker role in a typical edge use case.

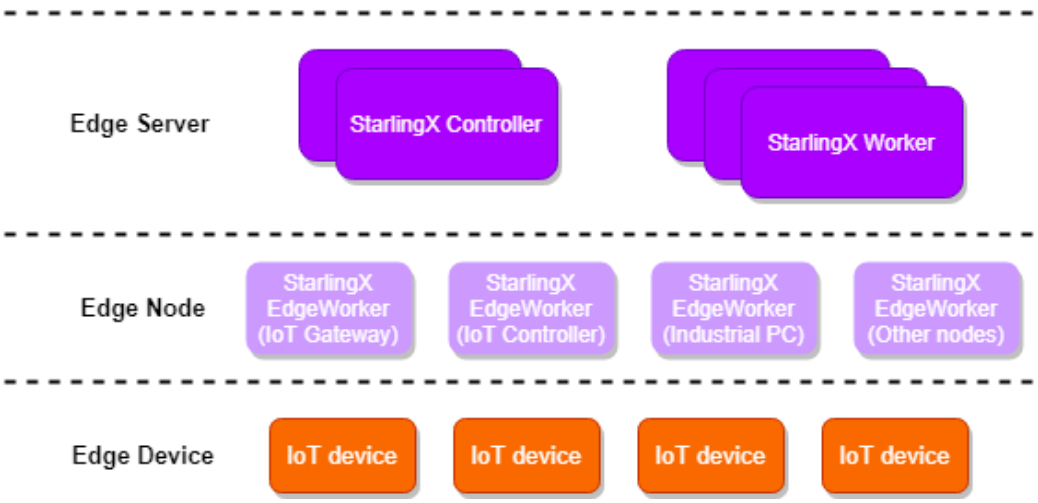

Figure 2 – EdgeWorker role

The EdgeWorker plays the role of communicating with the edge devices closely, it targets to the edge node layer. The EdgeWorker is responsible for controlling, monitoring the IoT devices as well as fetching data from them via specific protocol.

The EdgeWorker is loosely managed by StarlingX controllers which allows the EdgeWorker to be deployed in an area far from the servers and to be connected via a layer 3 network.

2.2 Networking of the EdgeWorker nodes

Since the EdgeWorker nodes locate in the field close to IoT devices, the types of network topology may vary. In this section, three commonly used types of networking topology will be described:

1. EdgeWorker nodes connect to controllers directly within a L2 subnet.
2. EdgeWorker nodes connect to controllers via a L3 network.
3. EdgeWorker nodes connect to controllers via a tunnel.

2.2.1 Direct connection

The EdgeWorker nodes connect to the management network (Hereinafter referred to as “MN”) of a StarlingX cluster within the same subnet. It means the connected controllers are in the field as the same location as EdgeWorker nodes, no matter it’s a sub-cloud of a StarlingX distributed cloud or a single cluster.

Figure 3 shows the network topology of the EdgeWorker nodes directly connecting to a StarlingX cluster’s MN. One interface of an EdgeWorker connects to MN while the other interfaces connect to IoT network or OT (Operational Technology) network. The EdgeWorker may or may not have an OAM interface due to the administration policy of the organization.

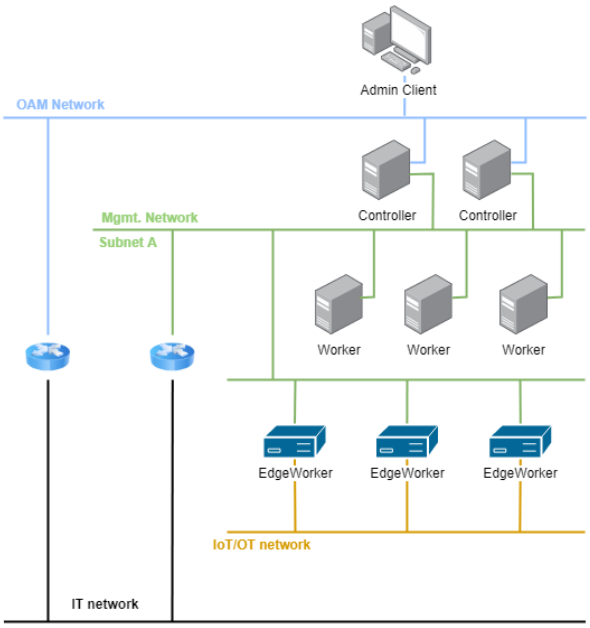

Figure 3 – EdgeWorker direct connection

With this deployment, the Worker nodes are responsible for offloading the workloads from the EdgeWorker nodes, while the EdgeWorker nodes are focus on real-time control, data transition, status monitoring via various protocols on IoT/OT network.

2.2.2 L3 network

If a StarlingX cluster is deployed for a general purpose: consolidate workloads to servers, the cluster is probably deployed in a server room. Thus, the network connection is more complicated in this situation where the EdgeWorker nodes are connected to controllers via a layer 3 network. Figure 4 indicates the network topology of a L3 network connection of EdgeWorker and controllers.

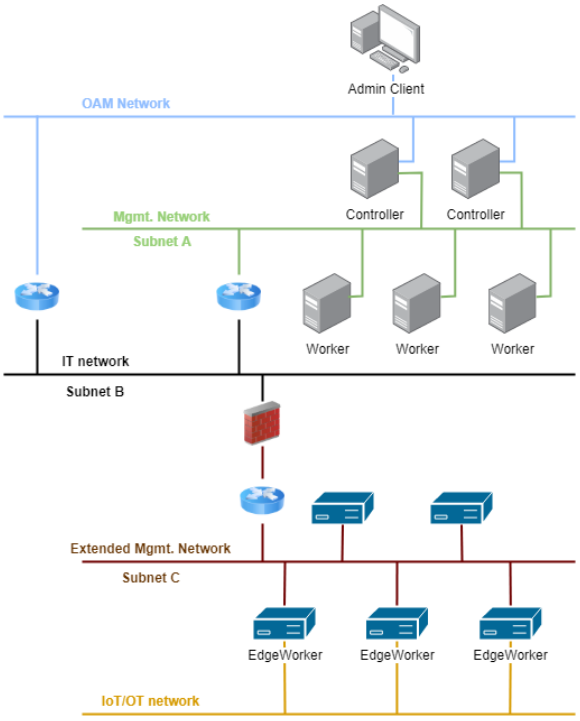

Figure 4 – L3 network topology

It's a common network topology of industrial implementation within a factory. The OAM network and MN within the server room connects to IT network, meanwhile, the network connecting to the EdgeWorker nodes (Hereinafter referred to as "EMN", "Extended Management Network") is isolated behind a firewall. The MN and EMN are in different subnet and are connected via L3 routing.

2.2.3 Tunneling

The L3 network requires the authorization and configuration of the routers connecting to the MN and the EMN. Sometimes it is not controllable and even not possible because of IT network regulation. The alternative and more secure way is to establish a tunnel between the MN and the EMN. Tunneling technologies like VXLAN (Virtual Extensible Local Area Network), VPN (Virtual Private Network) could well serve this target.

By establishing a tunnel, the MN and the EMN are connected within the same L2 subnet (figure 5).

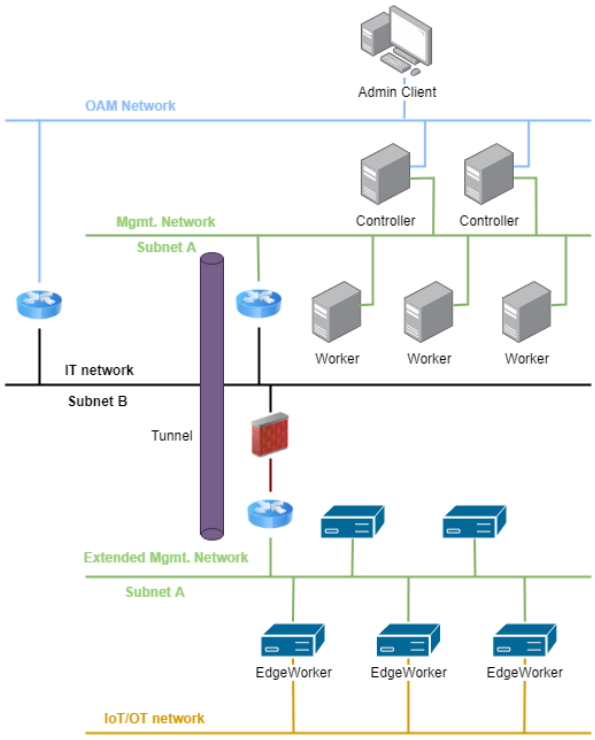

Figure 5 – Tunneling between the MN and the EMN

The EdgeWorker management via the L2 network and the tunneling will be considered in Hummingbird's scope.

2.3 Provisioning EdgeWorker nodes

The EdgeWorker nodes could be an OS pre-installed node or a node installed by an independent installation tool.

The basic idea of Hummingbird's EdgeWorker management is to provision a "kubelet" daemon and corresponding container runtime for EdgeWorker nodes to take them as Kubernetes nodes. With the help of network connectivity described in section 2.2, the EdgeWorker node could be provisioned via MN-EMN.

The provisioning process includes the node configuration, the package installation, the Kubernetes node bootstrap and the system application enabling. All the steps are implemented via ansible playbook.

1. In the node configuration step, the setup environment of the EdgeWorker node is prepared for later services enabling.
2. In the package installation step, the packages such as kubelet, containerd and their dependency packages are installed based on the EdgeWorker nodes' OS distribution. Due to the different ways of EdgeWorker nodes' onboarding, the packages could either be installed in this step or pre-installed in OS bootstrapping step described in section 2.3.
3. The EdgeWorker nodes are joined to the Kubernetes cluster in the Kubernetes node bootstrap step. In this step, the essential container images will be pulled to the host prior to the pod's initiation.

4. After the EdgeWorker nodes are joined to the Kubernetes cluster and ready, the optional system applications are orchestrated to the EdgeWorker nodes for enabling more platform functionality.

Figure 6 is a provisioned EdgeWorker with kubelet and container runtime running.

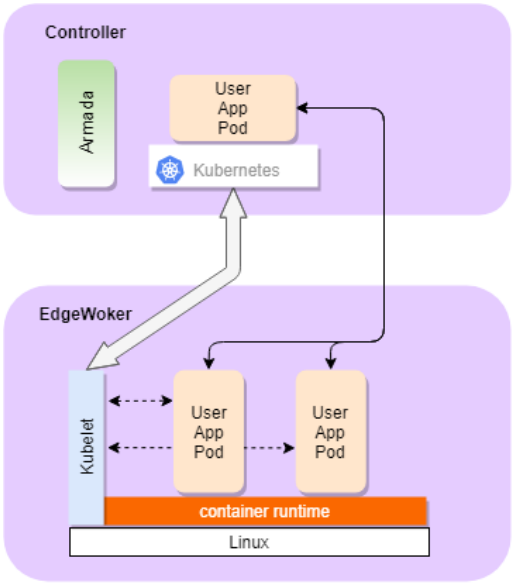

Figure 6 – A provisioned EdgeWorker node with kubelet and container runtime running

"stx-openstack" is an example of how StarlingX brings the computing/networking virtualization functionality to the EdgeWorker nodes. Figure 7 shows the system application "stx-openstack" is being orchestrated to an EdgeWorker node and the VM is launched by the containerized nova service.

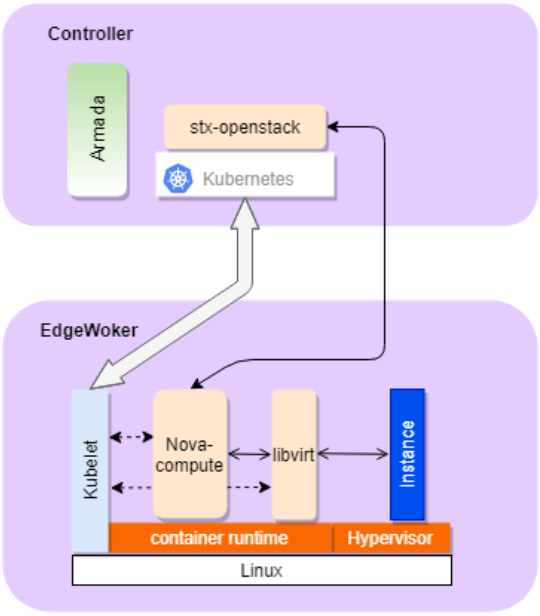

Figure 7 – stx-openstack is being orchestrated to an EdgeWorker node

2.4 EdgeWorker manageability

After the EdgeWorker node added to the Kubernetes cluster, StarlingX has extended the ability of orchestrating applications to edge node layer. However, the controller is not able to manage the whole lifecycle of the EdgeWorker node at the infrastructure level. The target is what the flock services aim at.

Flock services are the StarlingX specific services for system management, including Configuration Management, Host Management, Fault Management, etc. Due to the differentiation of the EdgeWorker OS bootstrap method, the flock services on EdgeWorker nodes are running within containers, unlike the way for Worker nodes that those services directly running on the host.

By containerization, the flock service agents can be deployed flexibly due to the functionality of the EdgeWorker nodes. Figure 8 shows the inception of the containerized flock services running on EdgeWorker nodes.

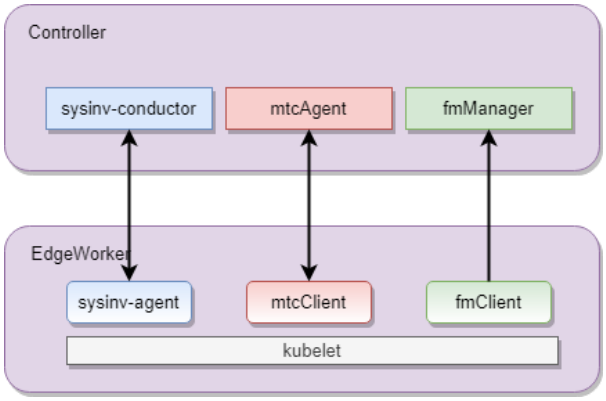

Figure 8 – Containerized flock services running on the EdgeWorker node

2.5 EdgeWorker group storage

The EdgeWorker nodes are managed by controllers via IT network. Due to the unpredictable IT network availability, the Ceph storage cluster from controller nodes could not promise “always-online” for EdgeWorker nodes because of the Ceph monitors being communicated across IT network.

Moreover, in current architecture, the persistent data generated from EdgeWorker nodes will be frequently transmitted to OSDs on the controllers via IT network. This will lead to the storage performance problem and cause the security issues as well.

Considering the availability, performance and security requirements, a group storage architecture is introduced for resolving the underlying problems. With Rook-Ceph enabled, creating multiple storage clusters is supported. In this architecture, each EdgeWorker group has its own Ceph cluster for group data storage and sharing while the data is transmitted via EMN within a group.

Figure 9 shows the architecture of EdgeWorker group storage

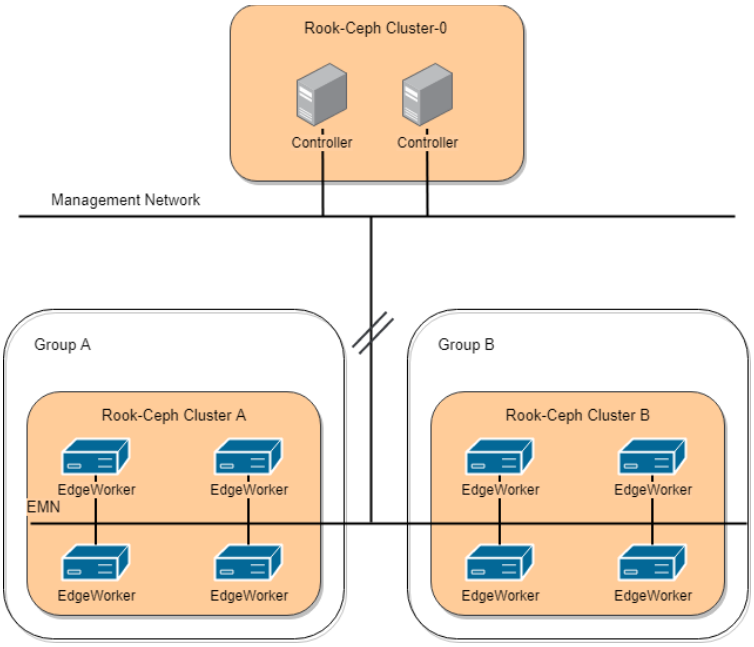

Figure 9 – Group storage for EdgeWorker nodes

2.6 Edge application orchestration

The user applications are orchestrated by Armada and Kubernetes from StarlingX controller nodes. With the EdgeWorker node joined, the orchestration of the user applications is required for more precise, more flexible and more resilient, since the availability of the applications running on the EdgeWorker nodes have great impact on business.

For example, the redundancy design is commonly used in industrial use case for high availability. Several EdgeWorker nodes are combined to a federation group for designated activities. The application along with its configuration for each activity is ready to run on at least 2 EdgeWorker nodes for redundancy (See figure 10). The orchestrator is required to switchover the application to the backup node if the primary node is either voluntary or involuntary disrupted within a very short period.

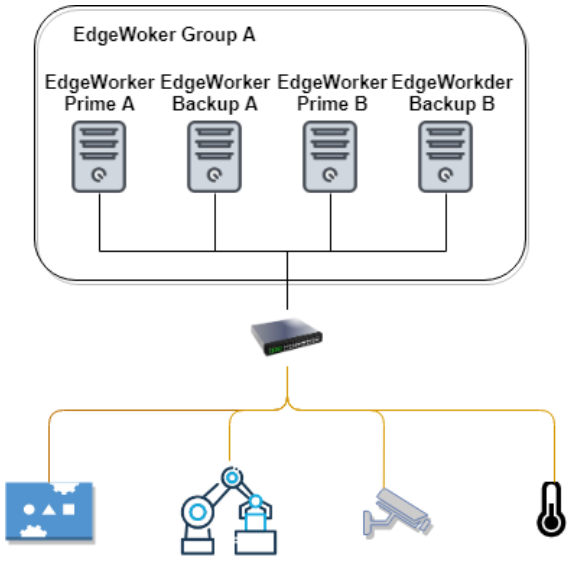

Figure 10 – the EdgeWorker redundancy in industrial use case

Kubernetes provides the platform level ability for orchestrating applications, while Hummingbird is targeting to provide administrative interface for edge application orchestration, including:

- Declarative labeling of the EdgeWorker nodes and EdgeWorker groups
- Provision interface for respective storage backend among EdgeWorker groups

3 Hummingbird and Distributed Cloud

Hummingbird aims to manage small nodes which brings the manageability of EdgeWorker to central cloud. With the flock services running on EdgeWorker, the way that the EdgeWorker nodes being operated from the central cloud is the same as the Worker nodes being operated. Figure 11 illustrated an architecture of EdgeWorker nodes in distributed cloud.

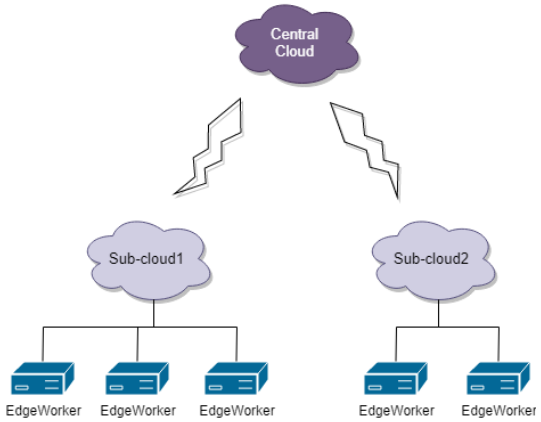

Figure 11 – EdgeWorker management from central cloud

4 Implementation Plan

The project implementation is proposed in several phases.

For stx.5.0 release

- Add new personality “EdgeWorker”
- Add provisioning ansible playbook for EdgeWorker node
- Containerize sysinv-agent in partial functionality
- Enable group storage by rook-ceph with labels

For stx.6.0 and later release

- Tunneling network solution over L3 network
- Group storage autonomous enhancement
- Edge app orchestration specific feature
- Containerize sysinv-agent, mtcClient

5 Conclusion

Hummingbird is a project for EdgeWorker nodes management in StarlingX, covering from bootstrapping, provisioning, node management to app orchestration stages. Hummingbird enables StarlingX to manage small nodes for various use cases such as IOT, industrial, retail, etc. It resolves the small node manageability issues which are introduced by complicated environment of network, storage, placement and by the node hardware limitation. After Hummingbird’s implementation, StarlingX will be a key competitor project of edge node management and edge app orchestration.