Bring Kubernetes to the Arm64 Edge Node by K3s
Agenda

● Why K3s
● What is K3s
● What we do to enhance the Kubernetes – K3s Architecture
Running Kubernetes on the edge node?

- Most Kubernetes distributions don’t support Armhf
- Kubernetes could easily consume up to 4gb of RAM
- Kubernetes wasn’t built for embedded or offline management
Agenda

- Why K3s
- What is K3s
- What we do to enhance the Kubernetes - K3s Architecture
Introduction K3s

- Lightweight certified Kubernetes distro
- 40MB binary, 512MB memory consumption
- Single process integrated Kubernetes master, Kubelet, and containerd
- SQLite in addition to etcd
- Simultaneously released for x86_64, ARM64, and ARMv7
Things Removes and Adds

Removes
- Legacy and non-default features
- Alpha features
- In-tree cloud providers
- In-tree storage drivers
- Docker (optional)

Adds
- Simplified installation
- SQLite3 support in addition to etcd
- TLS management
- Automatic Manifest and Helm Chart management
- containerd, CoreDNS, Flannel
K3s Architecture

[Diagram showing the K3s Architecture with components like SQLite, API Server, Tunnel Proxy, Scheduler, Controller Manager, k3s Server, k3s Agent, Tunnel Proxy, kube-proxy, Flannel, Kubelet, containerd, Pod, Pod, Pod, Pod]
Get started with k3s in two easy steps

Quick Start

1. Download k3s - latest release, x86_64, ARMv7, and ARM64 are supported

2. Run server

```bash
sudo k3s server &
# Kubeconfig is written to /etc/rancher/k3s/k3s.yaml
sudo k3s kubectl get node

# On a different node run the below. NODE_TOKEN comes from /var/lib/rancher/k3s/server/node-token
# on your server
sudo k3s agent -server https://myserver:6443 --token ${{NODE_TOKEN}}
```
Agenda

- Why K3s
- What is K3s
- What we do to enhance the Kubernetes – K3s Architecture
Here is Rancher’s Solution, but not OSS
Things to Enhance the Cloud-Edge Architecture

Kubernetes Cluster
- Multiple K3s Clusters
- Operation

K3s Operator
- Provisioner
- Monitor
- APP deployment

K3sup

Data Center

Edge Nodes

K3s Cluster 1
- master
- worker1
- worker2
- worker3

K3s Cluster 2
- master
- worker1
- worker2
- worker3
Custom Resource Definition (CRD):
- An object that extends the Kubernetes API or allows user to introduce their own API in K8s.
- A CRD file defines your own object kinds and lets the Kube API Server handle the entire lifecycle.

Operator:
- Watch the CRD from API server, and do the right things accordingly.
- “Cloud Native” method to get through the resources with K8s cluster.
Operator Control Process

- Observe
- Analyze
- Act

Comparison
- Desired State:
  - Spec: Bacon: 3
- Current State:
  - Status: Bacon: 1

Update CRD Object
K3sup: Operation tools for K3s
K3s Operator

Custom Resource Definition (CRD):
K3sNode1:
  ip: 192.168.100.1
  sshPort: 22
  type: Server
  skipInstall: No
  passwd: **

Watch CRD Change

K3s Node 1

K3s Operator

K3sup

Setup K3s on Edge

Update CRD Change
Demo

Come to join us at Demo Friday!
Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contactus@linaro.org