Simulating Edge Environments with QEMU and KubeEdge

Accelerating Edge Validation Before Hardware Availability

KubeCon + CloudNativeCon India 2026 · Mumbai

Sachin Jha
LFX Mentee – KubeEdge (CNCF)
Engineering Intern – Zycus

CNCF KubeEdge QEMU

About Me

Sachin Jha

- LFX Mentee – KubeEdge (CNCF)
- Engineering Intern – Zycus
- KubeCon + CloudNativeCon India 2026 Speaker
- Cloud-Native & Edge Computing Enthusiast
- Open Source Contributor

LFX Mentorship Program

KubeEdge · CNCF · 2024–2025

Focus Area

Edge Simulation · ARM64 · KubeEdge

Conference

KubeCon + CloudNativeCon India 2026

Community

CNCF · Open Source Contributor

The Rise of Edge Computing

Compute is moving to where data is generated

Automotive

Manufacturing

Retail

Healthcare

Smart Cities

Robotics

AI Cameras

Industrial IoT

Edge deployments are growing rapidly · Validating early reduces risk · Hardware constraints are real

Why Validation Matters

Architecture Mismatch

x86 code runs fine locally; fails on ARM64 edge device

Resource Constraints

Memory & CPU limits differ drastically from dev machines

Failed Deployments

Production failures discovered after shipping hardware

Network Issues

Edge connectivity assumptions not validated pre-deploy

Hardware Dependencies

GPIO/NPU code paths untested; breakage at the edge

AI Workload Failures

Inference pipelines built without target hardware validation

A Universal Problem in Edge Development

How do you validate edge software before the hardware exists?

01

Hardware Is Scarce

Edge boards are expensive, delayed, or geographically inaccessible at project start

02

Teams Can't Wait

Software development, CI/CD pipelines, and deployment workflows must move forward

03

Testing in Production Is Costly

Discovering failures on real deployed hardware is measured in hours of downtime and millions of dollars

04

The Answer: Simulation

Simulate the target environment faithfully enough to validate what actually matters before hardware arrives

$356,000,000

lost in a single day

Company	Date	Root Cause
Toyota	**Aug 29, 2023**	**System failure**
Japan — 14 plants	One production day	during maintenance update

All 28 assembly lines across 14 plants halted — because a server update was not validated before it touched production systems.

Source: Carscoops, August 2023

The Cost of Skipping Validation

Toyota is not an isolated case — this pattern repeats across industries

$356M

in one day

Automotive

Toyota, 2023

Server malfunction during a maintenance update halted all 28 assembly lines across 14 Japan plants. Production order servers failed; no simulation or staging environment caught the issue before it hit production.

23 hrs

avg downtime per incident

Industrial IoT

Manufacturing sector

IoT edge device firmware updates deployed without staging environments regularly cause sensor blind spots and line stoppages. Industry analysts estimate unplanned downtime costs discrete manufacturers $260B annually.

100%

transaction loss during outage

Retail Edge

POS & inventory systems

Retail edge deployments with untested software updates have caused point-of-sale failures during peak periods. Without simulation, architecture differences between dev and edge targets surface only in production.

Simulation is not a nice-to-have. At scale, it is the difference between controlled testing and a nine-figure outage.

Exploring Alternative Solutions

Engineering decision-making — evaluating all options systematically

Approach	Cost	Realism	Flexibility	Independence
Cloud ARM Instances	Medium	High	Medium	High
Containers (x86)	Low	Low	High	High
Cross Compilation	Free	Low	Medium	High
Remote HW Access	Free*	**Very High**	Low	Low
QEMU ARM64 ✓ Chosen	**Low**	**High**	**Very High**	**Very High**

** Remote hardware had availability, latency, and debugging constraints*

Why QEMU Inside x86 — Not a Real ARM64 Instance

AWS t4g (Real ARM64)

Not simulation

- ✗ Real ARM64 chip — no emulation happening
- ✗ Anyone can do this — not a novel contribution
- ✗ Talk title "simulation" doesn't hold up
- ✗ More expensive than x86 t3 instances

VS

QEMU ARM64 Inside x86

Actual simulation ✓

- ✓ x86 CPU pretending to be ARM64 — genuine emulation
- ✓ Reproducible on any cloud, any provider
- ✓ Scale to multiple nodes on one instance
- ✓ Same setup validates N different ARM64 boards

RK3588: What KubeEdge Actually Sees

01 ARM64 Instruction Set (aarch64)

The CPU architecture — what the OS, kernel, and binaries are compiled for

02 Linux-capable with systemd

EdgeCore runs as a systemd service — needs a real Linux userspace

03 containerd Container Runtime

KubeEdge uses containerd to manage edge workloads on the node

04 Network Connectivity to CloudCore

WebSocket on port 10000 — the only network requirement KubeEdge has

Key Insight: QEMU provides all four of these — making it a valid simulation environment for KubeEdge validation

PART 2

The Simulation Architecture

Two x86 EC2s · QEMU inside EC2 · ARM64 inside QEMU

Two EC2 Instances — One Setup

EC2-1 · t3.medium · x86

k3s (Kubernetes)

CloudCore pod

Listens on port 10000

Kubernetes Control Plane

port 10000

EC2-2 · t3.xlarge · x86

QEMU ARM64 Emulation

- Ubuntu 22.04 ARM64
- ▶ **uname -m → aarch64**
- EdgeCore (arm64 binary)
- containerd · systemd

Simulation Is More Than Emulation

We validate deployment workflows, not just application binaries

Edge Applications & Workloads	*ARM64 container images: nginx, node-exporter, custom apps*
Containers (containerd)	*Container runtime managing ARM64 image lifecycle*
KubeEdge EdgeCore	*KubeEdge edge daemon running as systemd service (aarch64 binary)*
ARM64 Linux (Ubuntu 22.04)	*Full Linux userspace — systemd, networking, storage*
QEMU System Emulation	*x86 host machine pretending to be ARM64 hardware*
Virtual Hardware / EC2 x86	*AWS t3.xlarge — the actual underlying compute*

KubeEdge sees a fully functional ARM64 node — architecture-aware scheduling, pod deployment, and node metrics all work

PART 3 · LIVE DEMO

Simulation in Action

QEMU ARM64 → EdgeCore → CloudCore → kubectl

Step 1 · Verify the ARM64 Environment

```
# SSH into QEMU VM on EC2-2
$ ssh -p 2222 ubuntu@localhost

# Verify architecture
$ uname -m
aarch64

# Verify EdgeCore binary is ARM64
$ file /usr/local/bin/edgecore
ELF 64-bit LSB executable, ARM aarch64

$ systemctl status edgecore
● edgecore.service – active (running)
```

Step 2 · Edge Node Appears in the Cluster

```
# On EC2-1 (CloudCore side)

$ kubectl get nodes -o wide


NAME                  STATUS   ROLES    AGE   ARCH

ec2-cloudcore         Ready    master   10m   amd64

rk3588-sim-node       Ready    <none>    1m   arm64   ← your simulated node


$ kubectl describe node rk3588-sim-node | grep -i arch

kubernetes.io/arch=arm64

kubernetes.io/os=linux
```

Step 3 · Deploy ARM64-Only Workload

```
# Deploy pod constrained to ARM64 only
$ kubectl apply -f arm64-nginx.yaml
  nodeSelector:
    kubernetes.io/arch: arm64
  image: arm64v8/nginx:alpine

$ kubectl get pod arm64-test -w
NAME          STATUS              NODE
arm64-test    Pending             <none>
arm64-test    ContainerCreating   rk3588-sim-node
arm64-test    Running             rk3588-sim-node    ← success
```

PART 4

What This Validates and What It Doesn't

Honest scope · No overclaiming

Honest Simulation Scope

✓ What We Validated (KubeEdge concerns)

- ARM64 instruction set — EdgeCore runs as aarch64
- containerd on ARM64 — images pull and run correctly
- EdgeCore ↔ CloudCore communication (port 10000)
- Pod scheduling with ARM64 node selectors
- Node metrics via Prometheus node-exporter
- kubectl logs / kubectl exec via tunnel

– Requires Real Hardware

- NPU workloads (RK3588 neural processing unit)
- Mali GPU compute tasks
- GPIO / hardware peripherals
- Thermal and power behavior
- big.LITTLE CPU scheduler (A76 + A55 cores)

The Bigger Picture — Scale and CI/CD

Multi-node simulation on one t3.xlarge

rk3588-node-1	QEMU ARM64 VM · port 2222
rk3588-node-2	QEMU ARM64 VM · port 2223
rk3588-node-3	QEMU ARM64 VM · port 2224

All three appear as arm64 nodes in kubectl get nodes

N×

Simulate multiple edge nodes
on a single instance

CI/CD Pipeline Integration

GitHub PR → **Spin up QEMU** → **Deploy workload** → **Validate** → **Merge**

ARM64 regression testing on every PR · no hardware required

Key Learnings

01

Hardware Remains Important

Simulation accelerates development but does not replace real-hardware validation for NPU, GPIO, and power-sensitive workloads.

02

Simulation Accelerates Development

Developers can iterate on KubeEdge deployments, node configurations, and container workflows days before hardware ships.

03

Open-Source Ecosystems Solve Real Problems

QEMU, KubeEdge, containerd, and k3s — all CNCF or open-source tools — combined to solve a real engineering challenge.

04

Early Validation Reduces Risk

Catching architecture mismatches, resource constraints, and network issues before production deployment saves significant cost.

Future Scope

Where this work goes next

Edge AI Inference

Validate AI pipelines on ARM64 before NPU hardware is available

Multi-Architecture Testing

Extend simulation to RISC-V, ARM32 and other edge targets

CI/CD Integration

Automated ARM64 regression testing on every pull request

Digital Twins

Full environment digital twins for complex multi-node topologies

RISC-V Support

QEMU supports RISC-V — same approach applies to emerging ISA

Cloud-Native Edge Platforms

Deeper integration with KubeEdge, Akri, and OpenYurt

Validate before you buy.

Simulation is not a replacement for real hardware.

It is a practical engineering strategy that enables developers and organizations to validate, iterate, and innovate before hardware constraints slow progress.

- Two EC2 instances. QEMU ARM64. Full KubeEdge edge node simulation.
- Anyone can reproduce this in under an hour.
- Work rooted in the KubeEdge LFX Mentorship — real problem, real solution.

KubeEdge · CNCF · QEMU · Cloud-Native Edge · ARM64 Simulation

Thank You

Sachin Jha

LFX Mentee – KubeEdge (CNCF) · Engineering Intern – Zycus

jhasachin0115@gmail.com

GitHub

sachin21212121

LinkedIn

Sachin Jha

All scripts + setup guide

Thank you to the KubeEdge community and CNCF mentorship program · #KubeCon #CloudNativeCon