Edge AI and Hardware co-Design

Marco Gonzalez
Sr. Software Engineer @Red Hat

Table of contents

DISCLAIMER

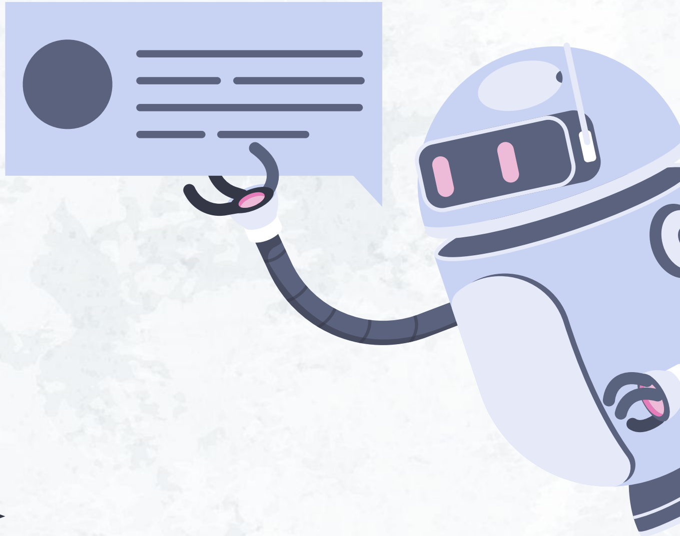
This presentation contains personal opinions and insights. The views expressed are my own and do not necessarily reflect the positions, strategies, or opinions of my employer.

01 →

What is Edge AI?

(Edge AI) = AI meets the Edge

→

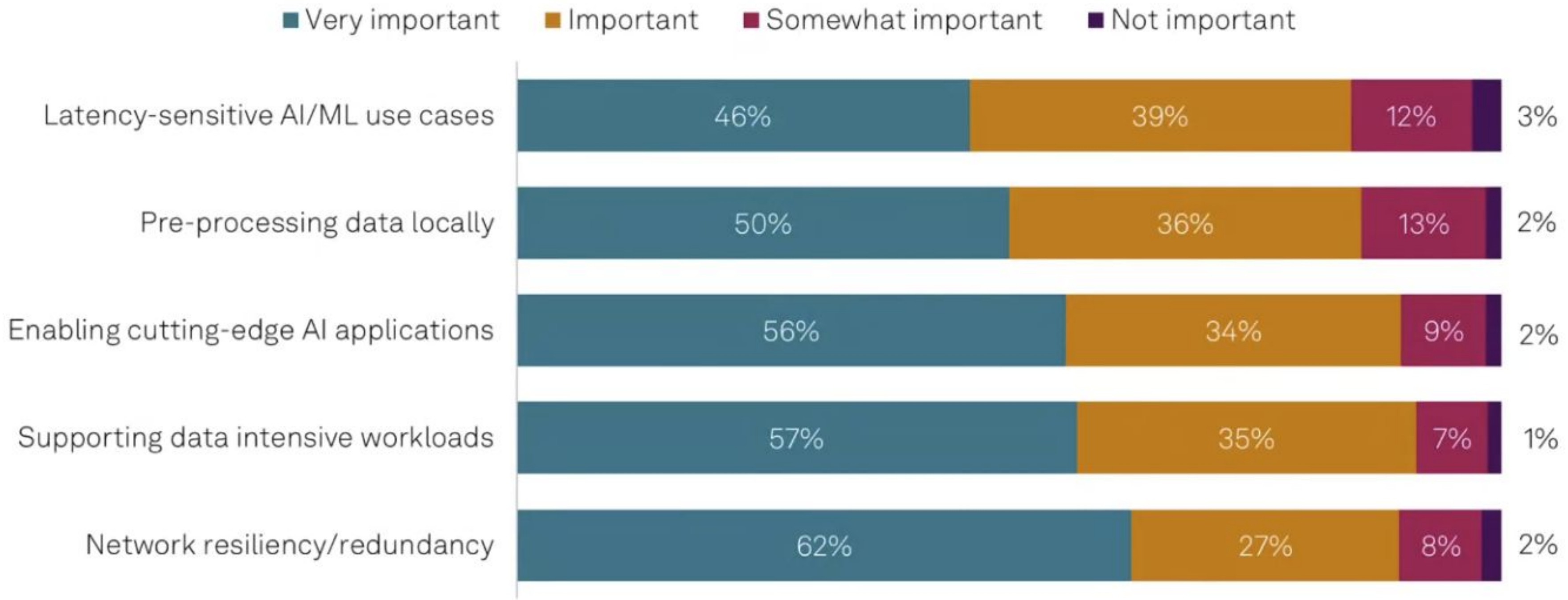


Edge AI

The defining characteristic of Edge AI, is that the **inference stage** (where trained models process inputs to generate predictions or decisions) is increasingly moved to the edge.

This approach leverages cloud resources for complex training while enabling **rapid, localized inference at the edge.**

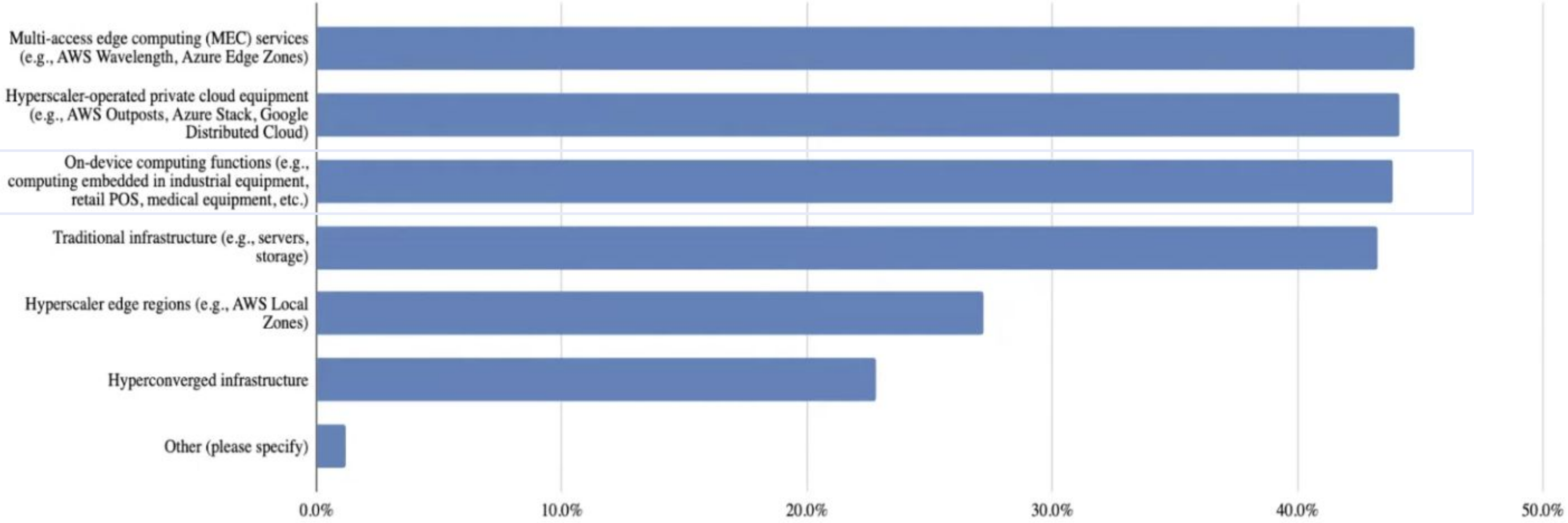
How important is edge as a part of an AI strategy?



Q. How important is the role of edge computing in your AI/ML strategy for each of the following?
Base: All respondents (n=693).
Source: Voice of the Enterprise: AI and Machine Learning, Infrastructure 2024.

S&P Global
Market Intelligence

Which of the following types of equipment or services are deployed in, or in connection with your organization's edge computing environment? (Please select all that apply.)



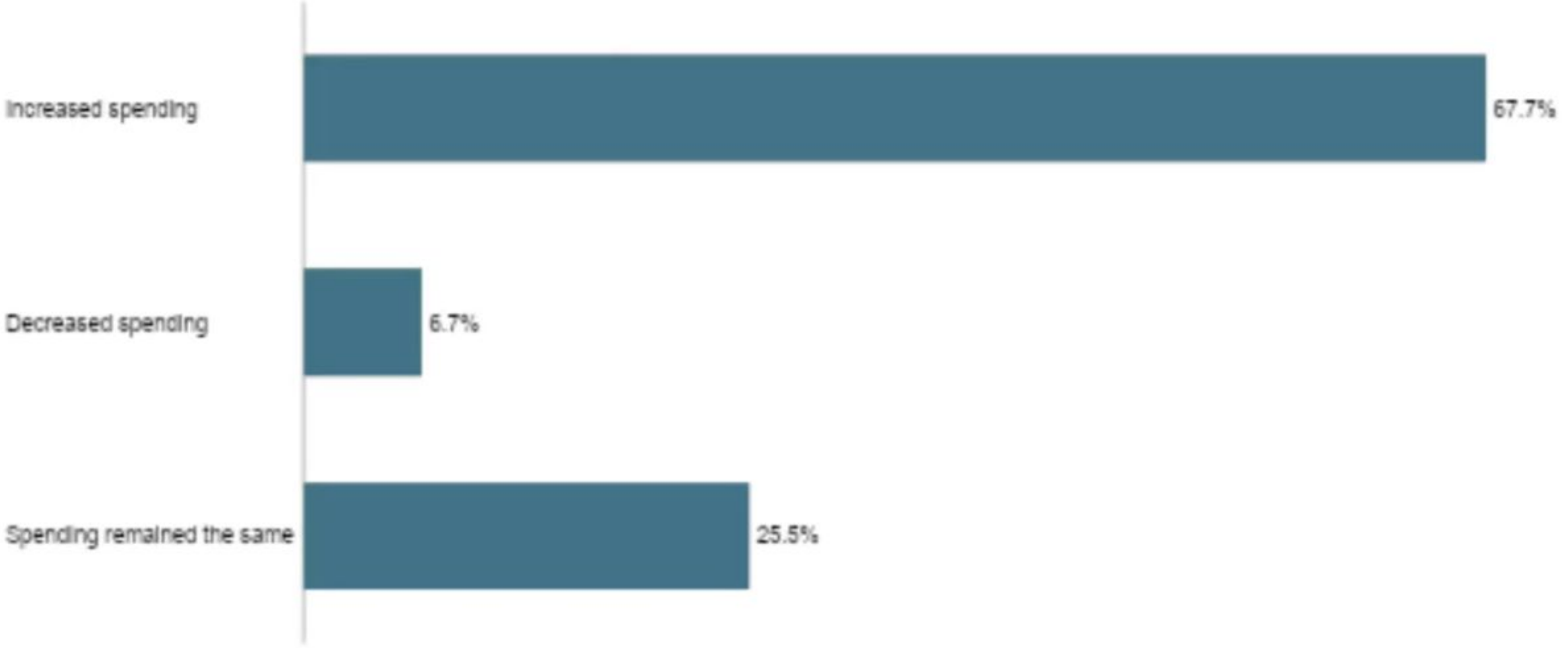
Q. Which of the following types of equipment or services are deployed in, or in connection with your organization's edge computing environment? (Please select all that apply.)

Base: Respondents whose organizations currently have data or plan to deploy computing in edge environments (n=342)

Source: 451 Research's Voice of the Enterprise: Edge Infrastructure & Services, Use Cases & Barriers 2025

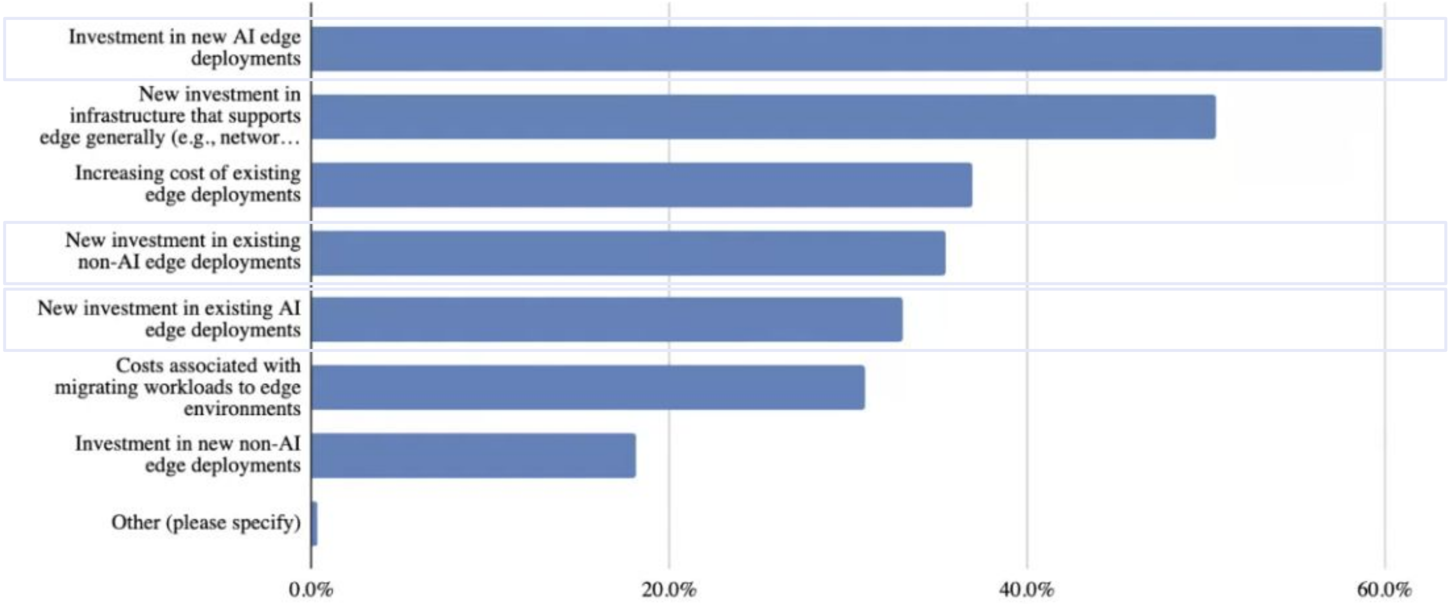
S&P Global
Market Intelligence

Which of the following best describes the change in your organization's spending on edge infrastructure and services over the last 12 months?



Q. Which of the following best describes the change in your organization's spending on edge infrastructure and services over the last 12 months?
Base: Respondents whose organizations currently have data or plan to deploy computing in edge environments (n=341)
Source: 451 Research's Voice of the Enterprise: Edge Infrastructure & Services, Use Cases & Barriers 2025

Which of the following factors, if any, are notable drivers of your organization's spending on edge infrastructure and services? (Please select all that apply.)



Q. Which of the following factors, if any, are notable drivers of your organization's spending on edge infrastructure and services? (Please select all that apply.)

Base: Respondents whose organizations currently have data or plan to deploy computing in edge environments (n=336)

Source: 451 Research's Voice of the Enterprise: Edge Infrastructure & Services, Use Cases & Barriers 2025

S&P Global
Market Intelligence

Findings from Inferencing@Edge Study

Use case categories:

- Predictive Maintenance, Surveillance, "Physical AI", Assisted Maintenance / Diagnostics

Noteworthy feedback from interviews:

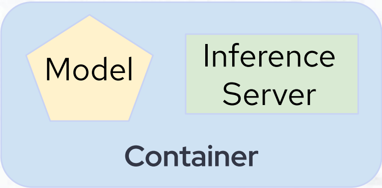
- Money is on inferencing & Pred AI (near/mid-term)
- Predictability is key: detect & correct prediction errors, solve hallucinations, escalations to human
- Purpose-built HW (Jetson, AMD Embedded+, NXP, ...) or reusing existing underutilized HW

Identified gaps:

- From PoC to Prod?
  - skills, large→small, generic→specific
- How to close the MLOps loop?
  - model&tool lifecycle, model monitoring
- Missing/slow hardware enablement!
- Scale-down inferencing stack?

Model in containers: Embedded VS External file VS Container

Model embedded



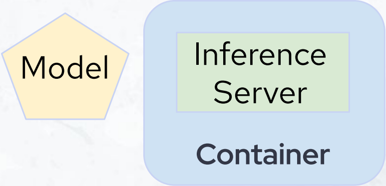
Pros:

- Self-contained
- No dependency on storage
- Simple

Cons:

- Harder to update
- Larger container size
- Less flexible

Model in external file



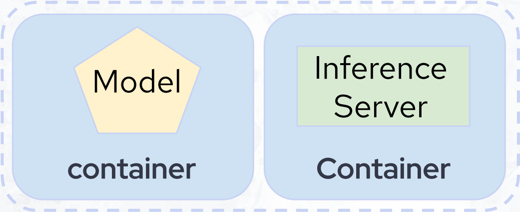
Pros:

- Decoupled model and inference
- Smaller container image

Cons:

- Complex model updates
- Dependency on storage

Model in container



Pros:

- Decoupled model and inference
- Scalable and kubernetes friendly

Cons:

- More complex deploy orchestration required with RHEL
- Dependency on storage

Edge AI - Primary Strategies

Strategy	Mechanism	Benefits	Drawbacks
Model Embedded in Container	• Packages the model and inference server together in a single, self-contained container	• Creates a self-contained unit with no external dependencies for storage or model loading. • Simplifies deployment, making it ideal for edge environments with limited infrastructure.	• Significantly increases container size due to the embedded model. • Can cause deployment challenges in bandwidth-constrained environments. • Model updates require rebuilding and redeploying the entire container.
Model as External File	• Decouples the model from the inference server container. • The inference server is configured to locate and load the model from an external file.	• Allows for independent updates of either the model or the inference server. • The inference server container is significantly smaller, reducing deployment times and resource utilization.	• Introduces a dependency on external storage systems. • Can complicate deployment in isolated or edge environments. • Model updates require careful coordination to ensure compatibility and proper loading.
Model in Separate Container	• Packages the model and inference server in separate containers. • Deploys the containers together as a cohesive unit. • Often uses init-containers to extract the model before the inference server starts.	• Creates a highly scalable, Kubernetes-friendly architecture. • Aligns well with modern orchestration practices (e.g., KServe's ModelMesh). • Allows inference components to be scaled independently based on workload.	• Requires more sophisticated deployment orchestration. • Can be complex to implement in non-Kubernetes environments like Podman or with RHEL.

Table 1: Edge AI - Deployment strategies

AI Model deployment at the Edge

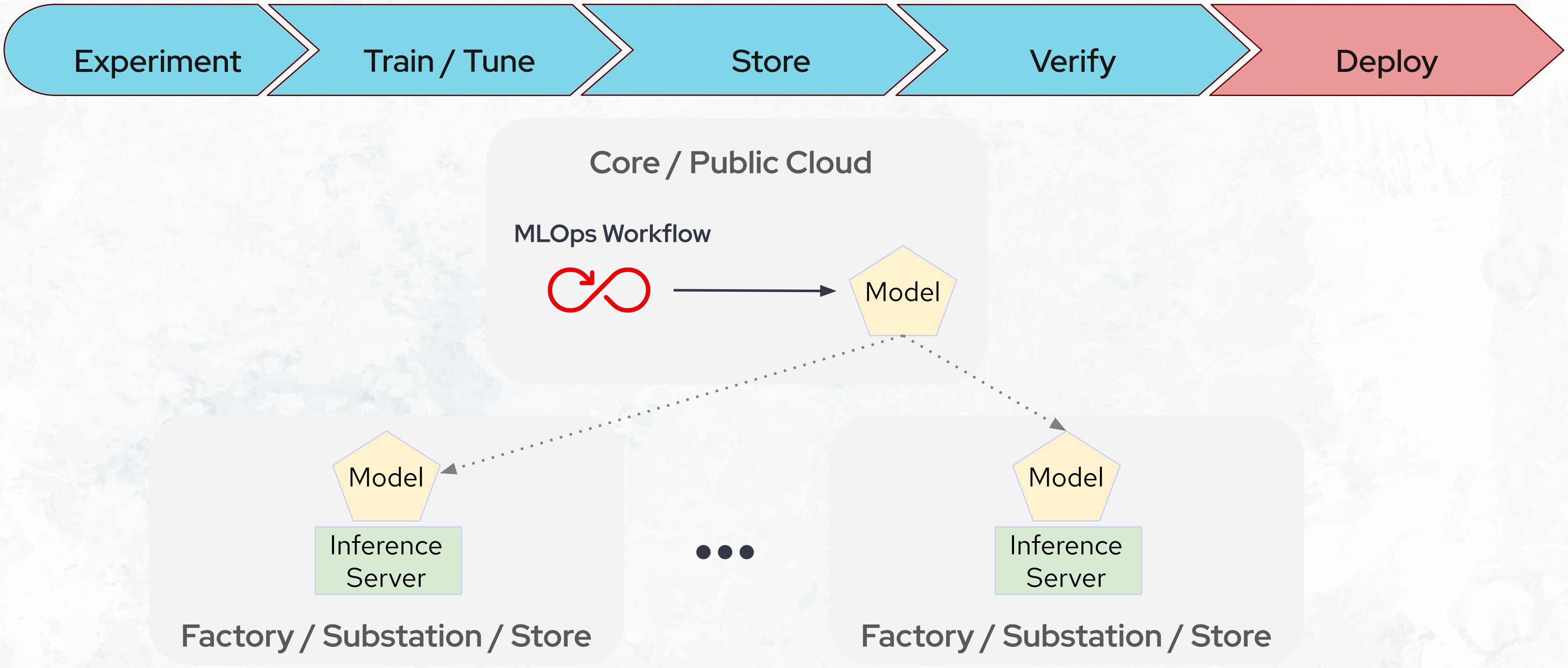


Figure 1: AI Model Deployment - General Overview

Edge AI - High Level Architecture

Core / Public Cloud / Near Edge

- Model Registry
- MLOps Workflow
- Container Build Pipeline
- Container Registry

Edge Layer (Factory / Substation / Store)

- Device Edge (Standalone Inference on Edge Devices)
- MicroShift on Device Edge (Lightweight Kubernetes Inference)
- Red Hat OpenShift (K8S Cluster)

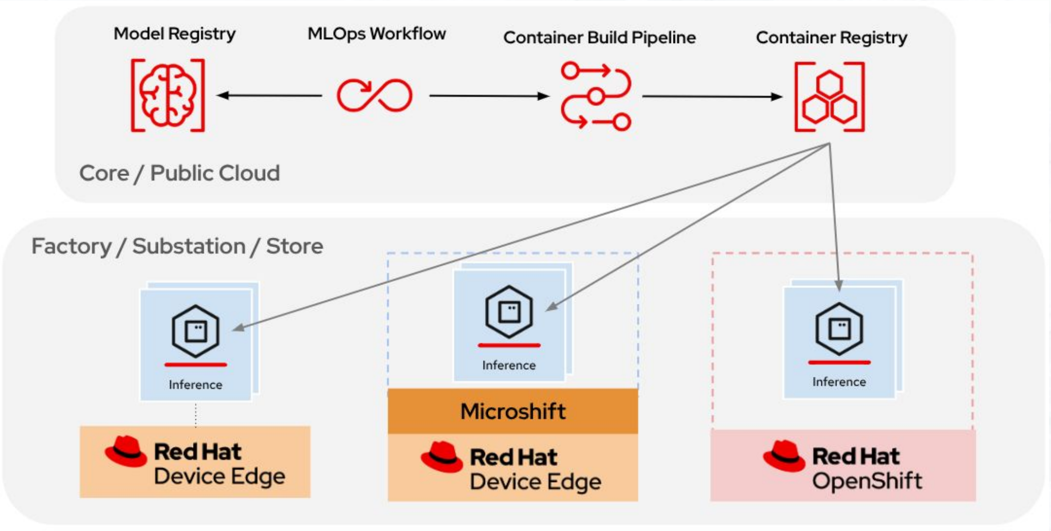


Figure 2: Edge AI High Level Architecture- Red Hat

Edge AI Inference

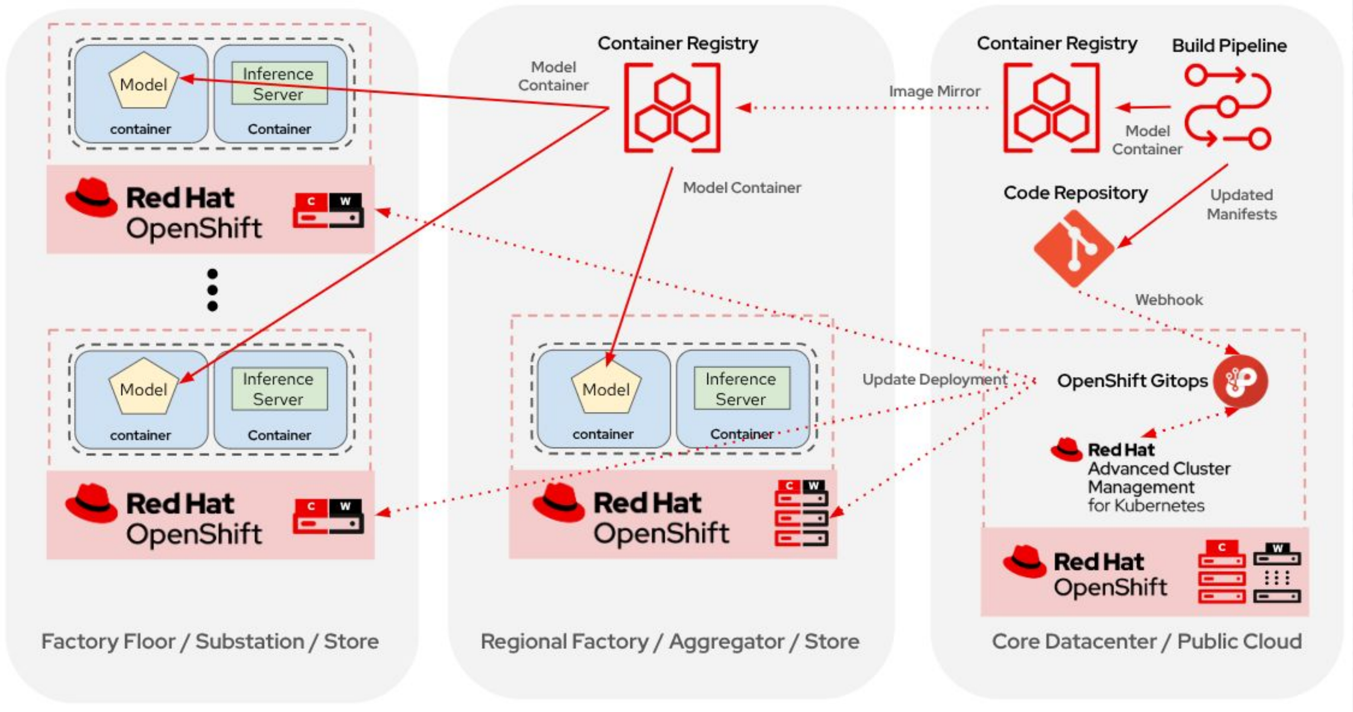


Figure 3 : High scale with mid-to-large Hardware footprint - Red Hat

02 →

AI Edge HW Co-Design

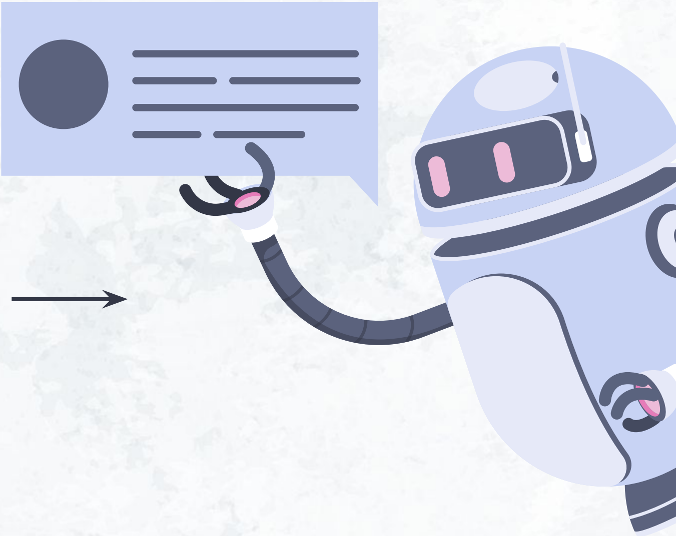
AI Edge HW Co-Design

Refers to the **concurrent and symbiotic design of both the hardware and the software (AI model/algorithm)** for a specific application.

Key Pillars of Edge AI Co-Design

- **Hardware:** Custom Accelerator Architecture (NPU, ASIC, FPGA).
- **Software:** Model Structure and Training
- **Compiler/Toolchain:** Developing hardware-aware compilers (e.g., Intel OpenVINO, Google Edge TPU Compiler).

(Co-Design)
Accelerators →



Accelerator Type	Main Vendors	Core Strengths	Key Weaknesses	Ideal Use Cases
GPU (Graphics Processing Unit)	NVIDIA (Jetson), AMD	High parallel processing, general-purpose flexibility, great for complex workloads	Higher power consumption, thermal management challenges, weight	Robotics, autonomous driving, complex video analytics
NPU (Neural Processing Unit)	Intel (Core Ultra), Arm (Ethos)	Specialized for neural network inference, high performance-per-watt efficiency	Less flexibility for general AI workloads	Mobile devices, IoT, real-time sensing
TPU (Tensor Processing Unit)	Google (Coral)	Optimized for matrix operations, fast inference at ultra-low power	Limited to lightweight, quantized models, lack of hardware flexibility	Battery-powered devices, embedded vision, real-time voice recognition

Table 2 : Accelerator types

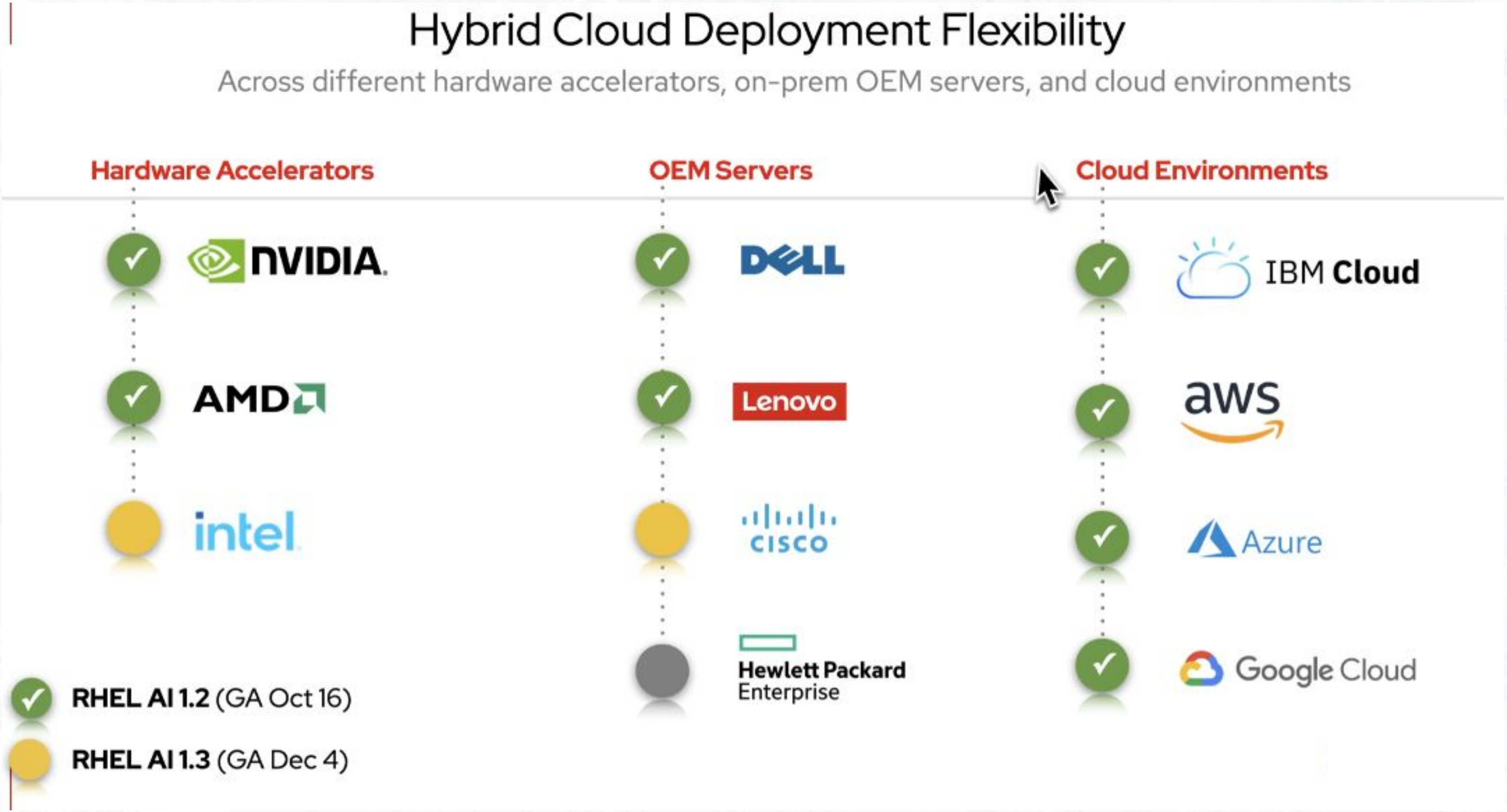
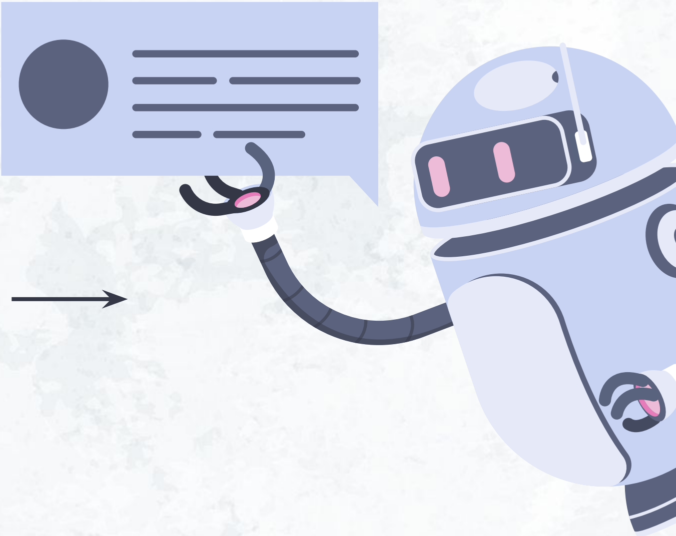


Figure 4 : High scale with mid-to-large Hardware footprint - Red Hat

(Co-Design)

Software



Red Hat Device Edge nodes distribution types

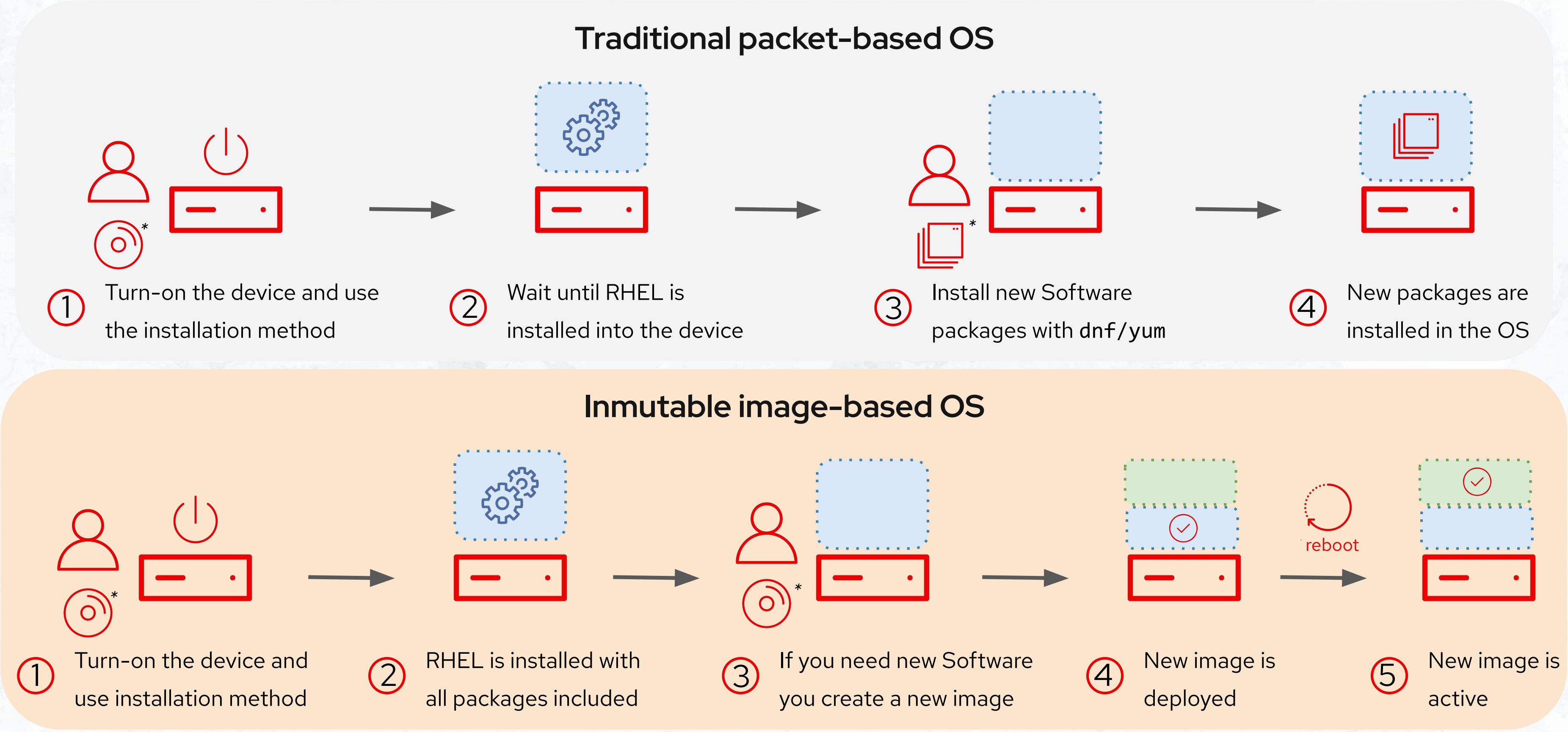


Figure 5 : Edge Nodes distribution types - Red Hat

The base for everything: **The Operating System**

Traditional
packet-based OS

Great Flexibility

VS

Inmutable
image-based OS

Great Consistency

(and simplicity)

...and new features *

Figure 6 : Inmutable Image-based OS - Red Hat

* *Automatic system rollbacks, Over-the-air differential atomic upgrades, Immutable File System, ...*

Image-based RHEL today and in the future

OSTree

Before

OSTree

+

bootc

Now *

OSTree: content-addressable file-system that stores the operating system as a series of immutable, versioned, de-duplicated file tree

** RHEL 9.6.+*

What is bootc?

Bootc is at the core and center * of bootable containers.

** It is a CLI tool that ships with a number of systemd services to manage a bootable container. It is responsible for downloading and queuing updates, and can be used by other higher-level tools to manage the system and inspect the system status.*

Bootable Containers

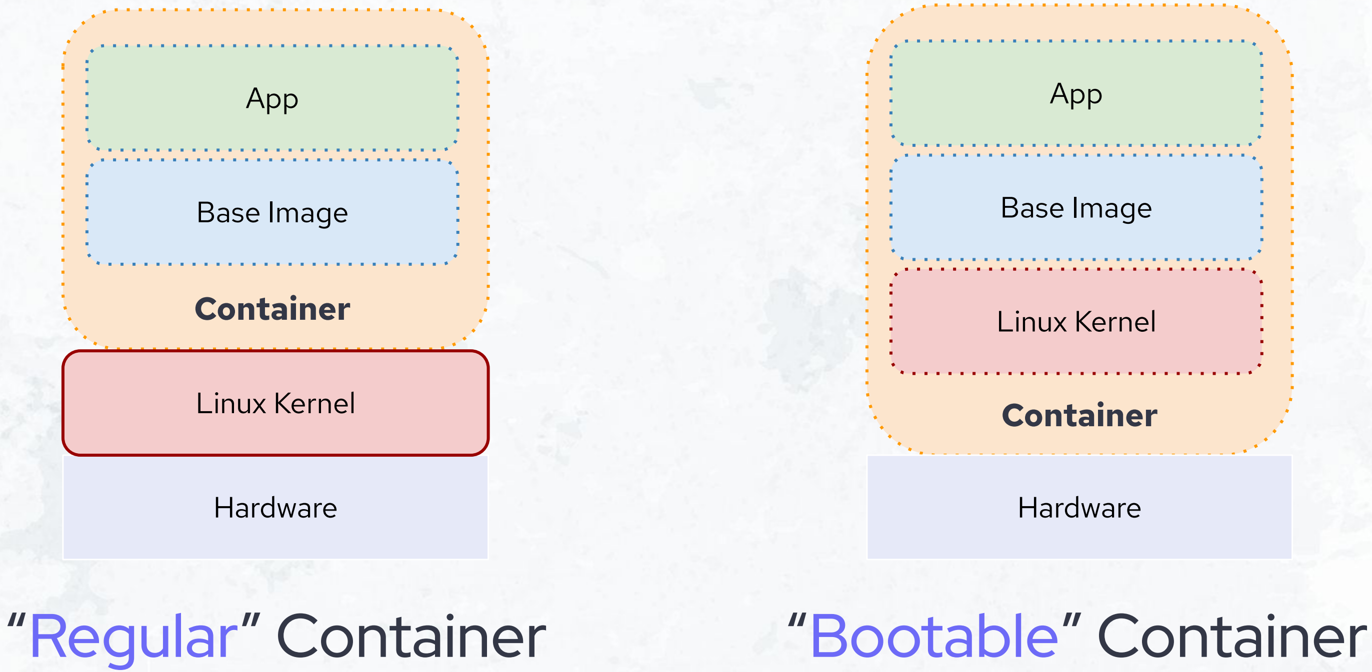


Figure 7 : Bootable Container solution - Red Hat

https://containers.github.io/bootc
https://docs.fedoraproject.org/en-US/bootc/getting-started/

Bootable Container Operating System deployment

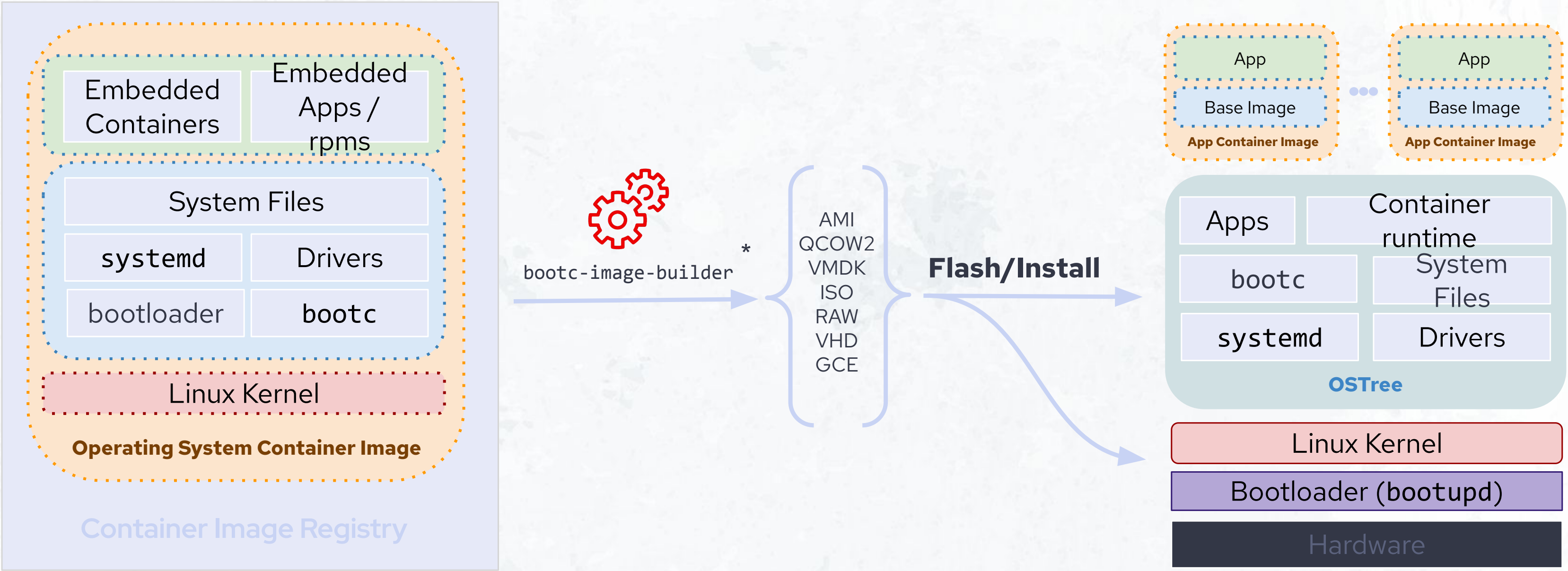


Figure 8: Bootable Container - System Deployment

* https://github.com/osbuild/bootc-image-builder

Bootable Container Operating System upgrade

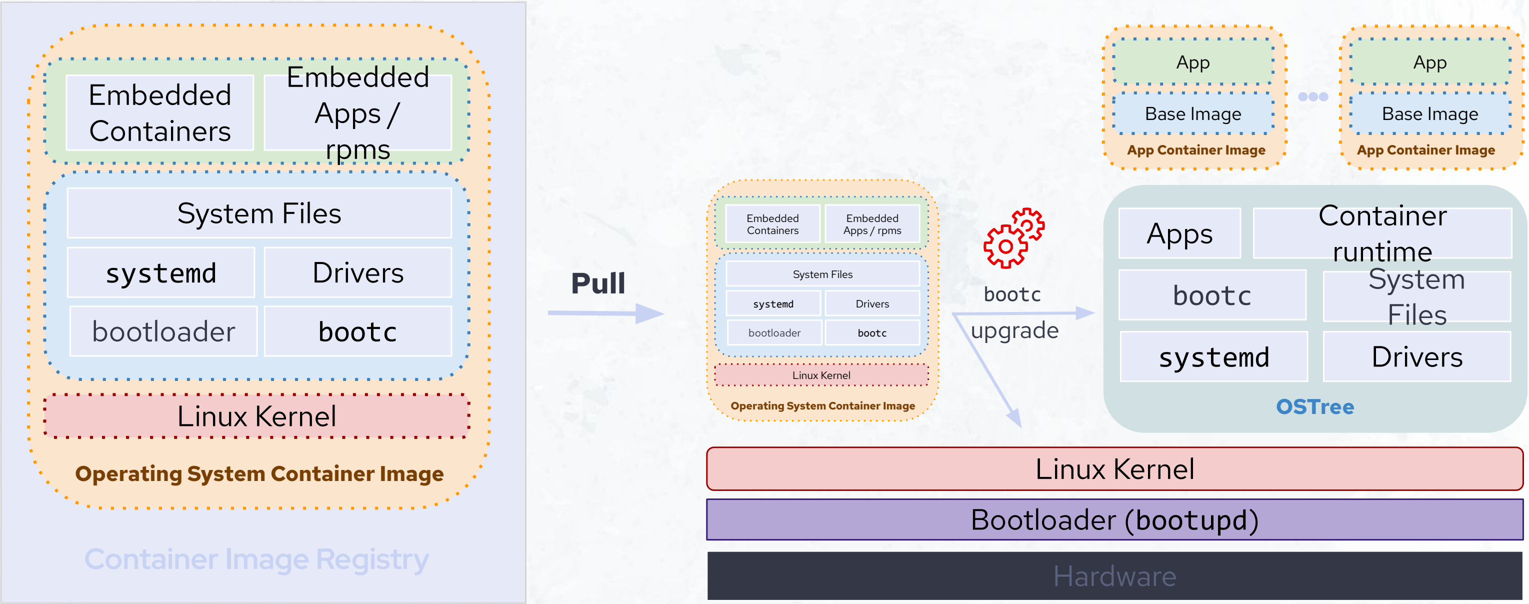


Figure 9: Bootable Container - System Upgrade

What are the benefits of Bootable Containers?

1. **A Unified Approach for DevOps**
2. **Simplified Security**
3. **Speed and Ecosystem Integration**

https://docs.fedoraproject.org/en-US/bootc/getting-started/

03 →

GPU-Enabled Platforms

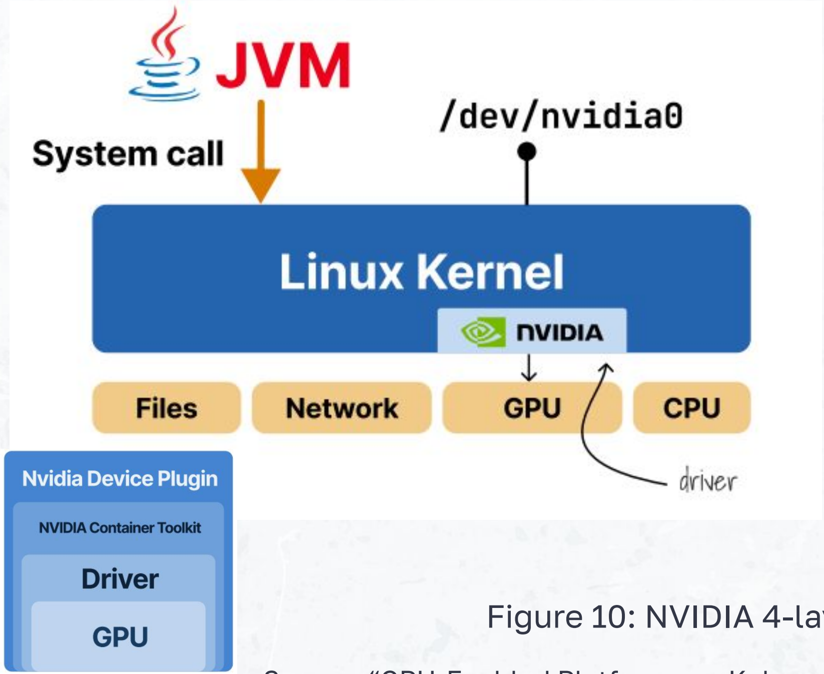
The GPU problem in Kubernetes

1. **The Kubernetes scheduler cannot effectively assign workloads to GPU-equipped nodes** because it only recognizes CPU and memory.
2. Once a pod is placed on a GPU node, its **container environment is too isolated to interact with the underlying GPU devices.**

1. Make GPU device files visible inside the container
2. Load the right libraries to talk to the GPU
3. Set up the CUDA context

Source: “GPU-Enabled Platforms on Kubernetes” by Daniele Polencic & Saiyam Pathak

NVIDIA Container Toolkit workflow



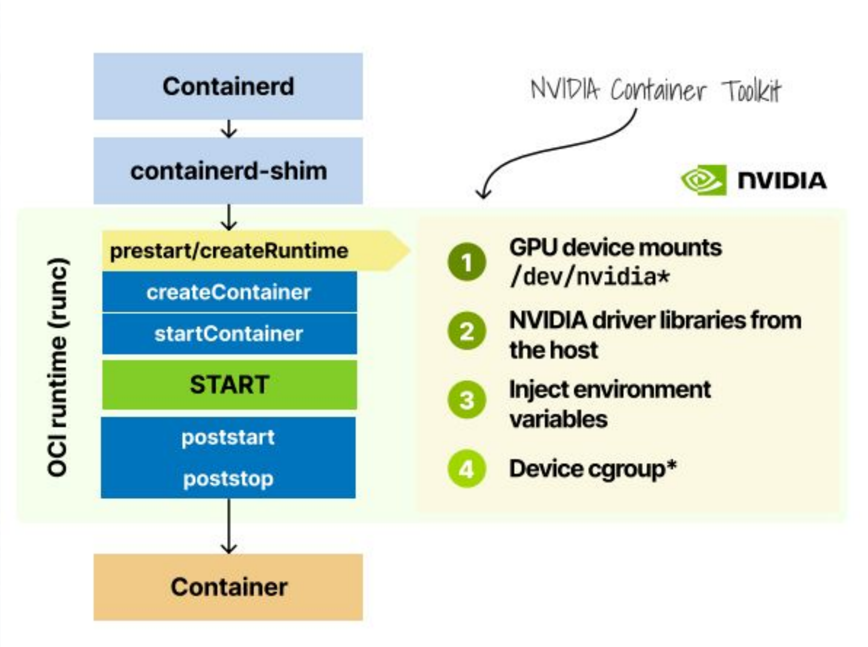


Figure 10: NVIDIA 4-layer GPU Integration stack

Source: “GPU-Enabled Platforms on Kubernetes” by Daniele Polencic & Saiyam Pathak

NVIDIA Device Plugin

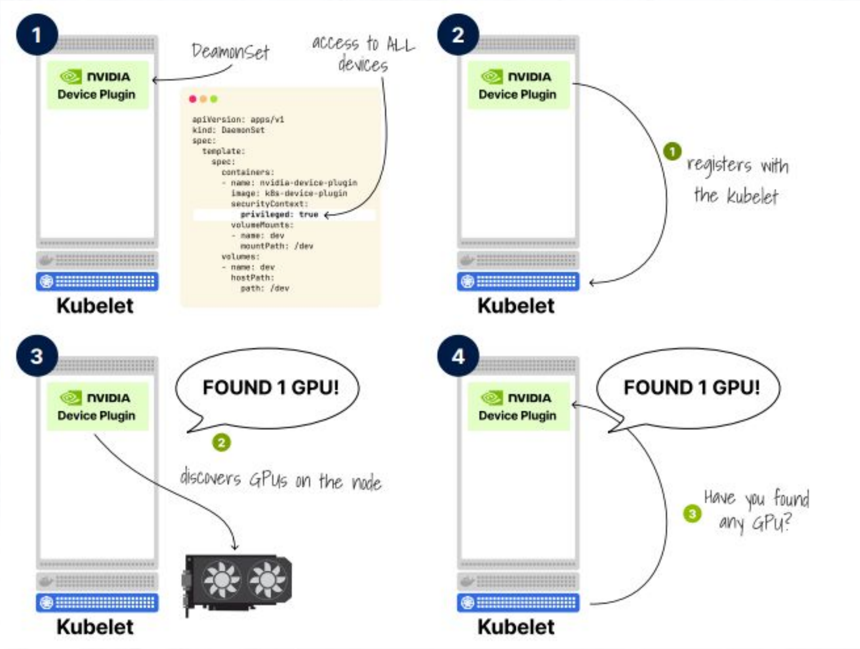
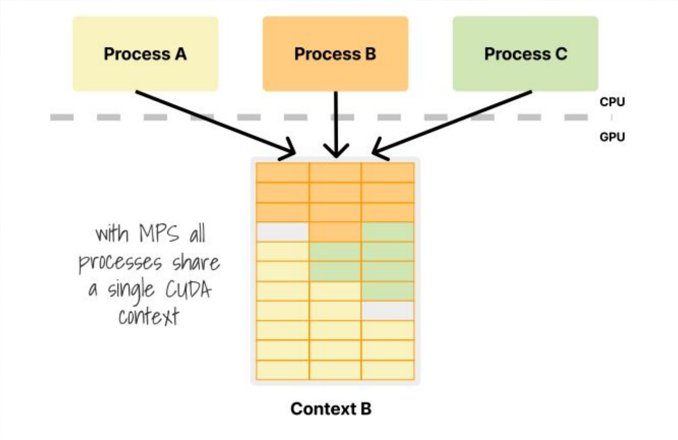


Figure 11: NVIDIA Device Plugin’s importance

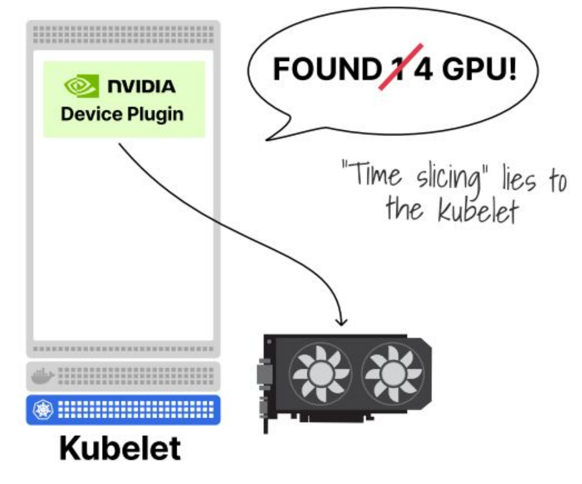
Source: “GPU-Enabled Platforms on Kubernetes” by Daniele Polencic & Saiyam Pathak

1. A DaemonSet deploys the NVIDIA Device Plugin to all nodes with GPU resources.
2. The registration process establishes communication between the device plugin and kubelet to manage GPU resources.
3. The device plugin discovers GPUs on the node but it doesn't report itt o the kubelet.
4. The kubelet queries for GPUs from the Device Plugin

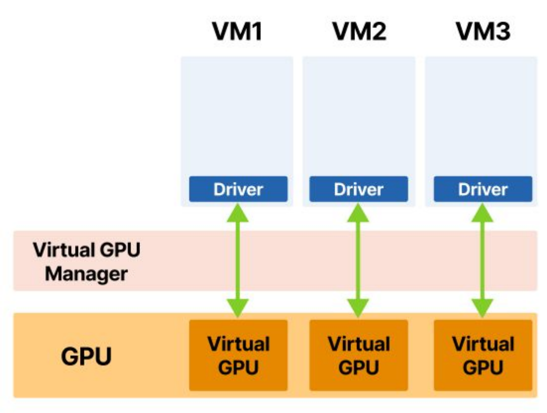
GPU Multi-Tenancy



Multi-Process Service (MPS) is the easiest way to share a GPU.



Multi-Process Service (MPS) is the easiest way to share a GPU.



vGPU creates virtual machines that each believe have their own dedicated GPU

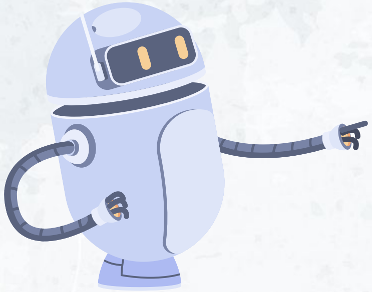
Source: “GPU-Enabled Platforms on Kubernetes” by Daniele Polencic & Saiyam Pathak

04 →

Key Takeaways

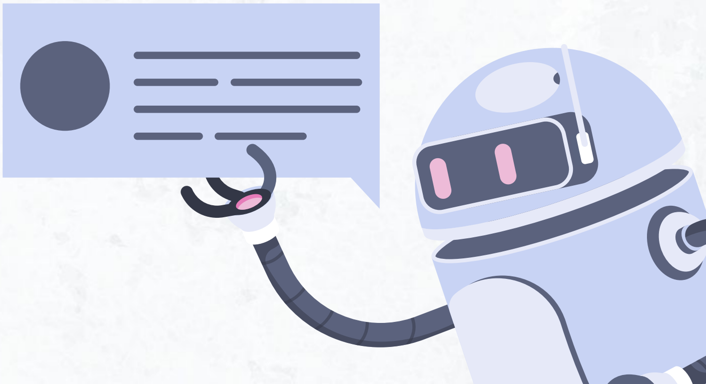
Edge AI Definition:

- Edge AI moves model inference to the edge for rapid, localized processing, while training remains in the cloud.
- Deploying production-ready LLMs at the Edge requires balancing model quality, responsiveness, and cost (HW Budget constraints).



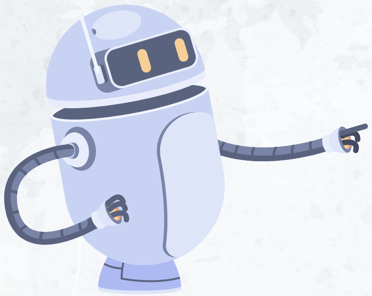
AI Edge Hardware Co-Design:

- AI Edge Hardware Co-Design integrates hardware and software, including custom accelerators and specialized compilers.



Challenges of GPU on Kubernetes

- Kubernetes natively struggles with GPU scheduling and container access to GPU devices, requiring specific solutions for enablement.
- Each GPU sharing method involves trade-offs: time-slicing exchanges isolation for simplicity, MPS exchanges safety for performance, MIG exchanges flexibility for security, and vGPU exchanges performance for isolation.



Edge AI and Hardware co-Design

Thanks!

Any questions? →

https://linktr.ee/mgonzalezo

Q&A

Why not attending Open-Source Summit Korea 2025?

Featured Session:
Exploring Kepler's Next Chapter: Achieving Cloud Native Sustainability With MCP Integration
(Wednesday, November 5, 16:35 - 17:05 KST) in the Chrysanthemum (5F) room.

https://events.linuxfoundation.org/open-source-summit-korea/

More details:
https://sched.co/2913z