linaro Connect 2025

RamaLama

Making AI Workloads on ARM Boring

Sergio Lopez Pascual & Eric Curtin

RedHat

An **easy**, **secure**, and **open** way to run and serve LLMs locally.

Have you...

Experimented with an LLM?

Run an LLM locally?

Built an AI-powered application?

Introducing RamaLama

Easy to use

Run or serve a chatbot with a single command

Secure

Experiment with LLMs while keeping your data private

Open

Built using open standards and AI building blocks

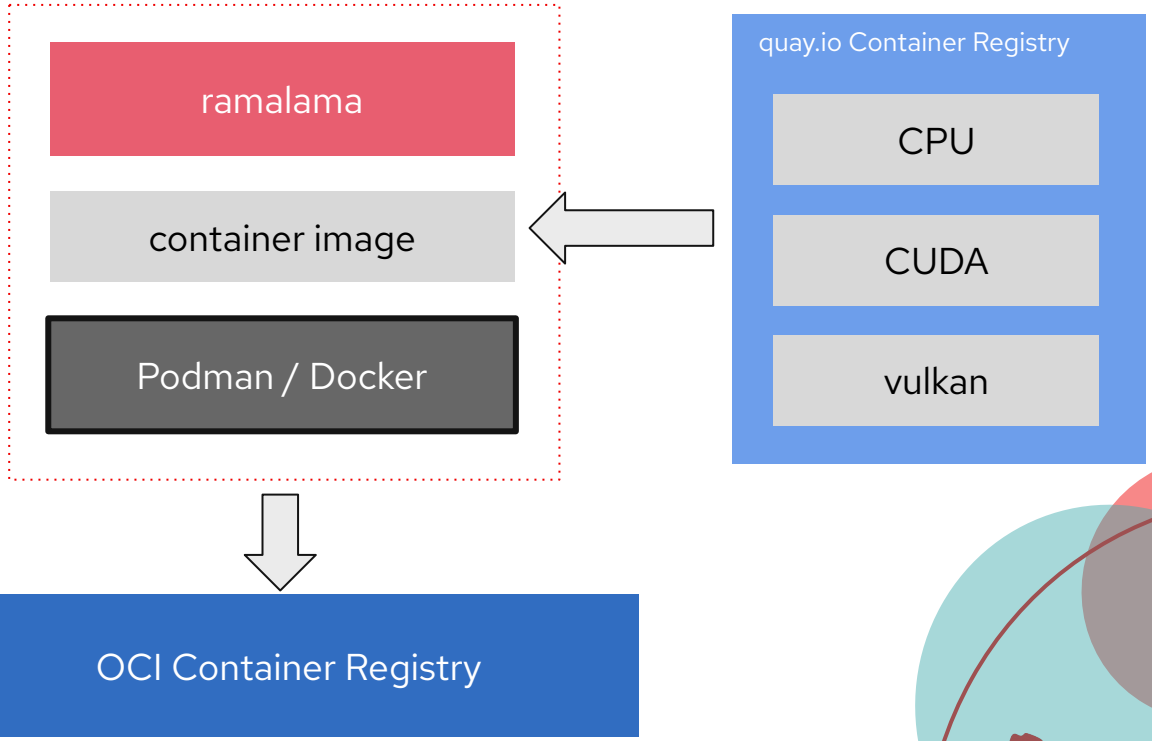
```
> ramalama run gemma3
```

Better with containers: easy, secure and open

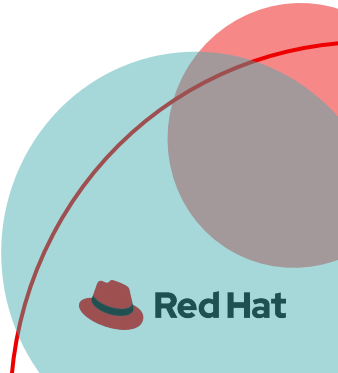
Built on Podman

Pulls in the relevant container image

Deploy to an OCI container registry



Running the model



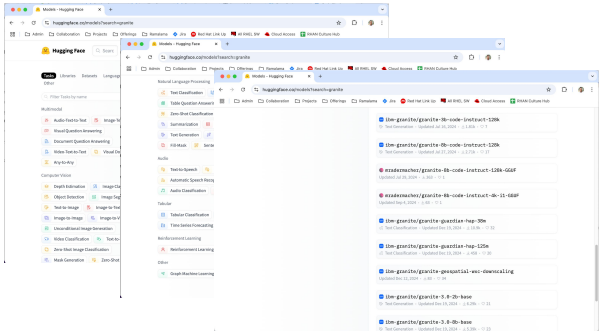
Complexity made easy

Which should I choose?

quantization
parameter size
alignment technique
. . .

It's hard to get hardware acceleration working!

Shortnames



```
> ramalama run gemma3
```

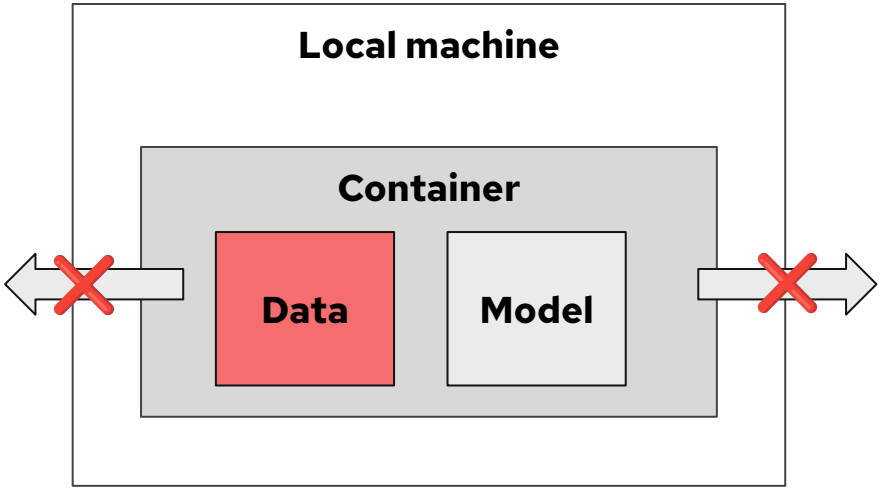
Performance made easy

Auto-detects system configuration and switches to the best performing accelerator stack

- **Out-of-the-box easy:** CPU-only
- **Minimal setup:** open source accelerator stack (Vulkan)
- **Full performance:** vendor-specific accelerator stack

Red Hat

Security: local and containerized



Keeping your data secure



- **Local:** where is your data going?
- **Containerized:** provides tighter **control** and **isolation**

Red Hat

Open: maximizing choice

Containerization enables **modularity**, easily swapping out for new blocks

Model Registry Choice

Hugging Face

OPEN CONTAINER INITIATIVE

?

Inference Runtime Choice

LLaMA C++

wHISper C++

?

Red Hat

Serve with standard APIs

```
> ramalama serve gemma3
```

OpenAI REST APIs

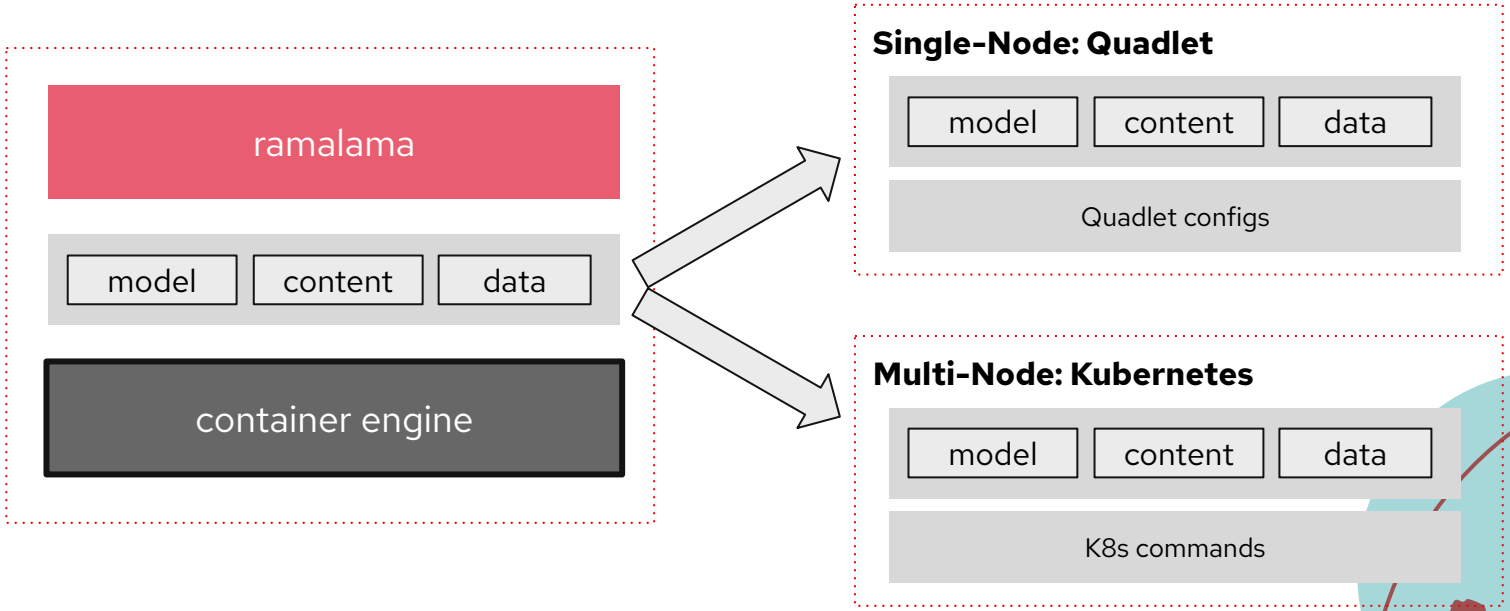
Industry standard endpoint APIs

Llama Stack

Connecting the front and back-ends together

Getting to production

Deploy to **Kubernetes** without being a Kubernetes expert



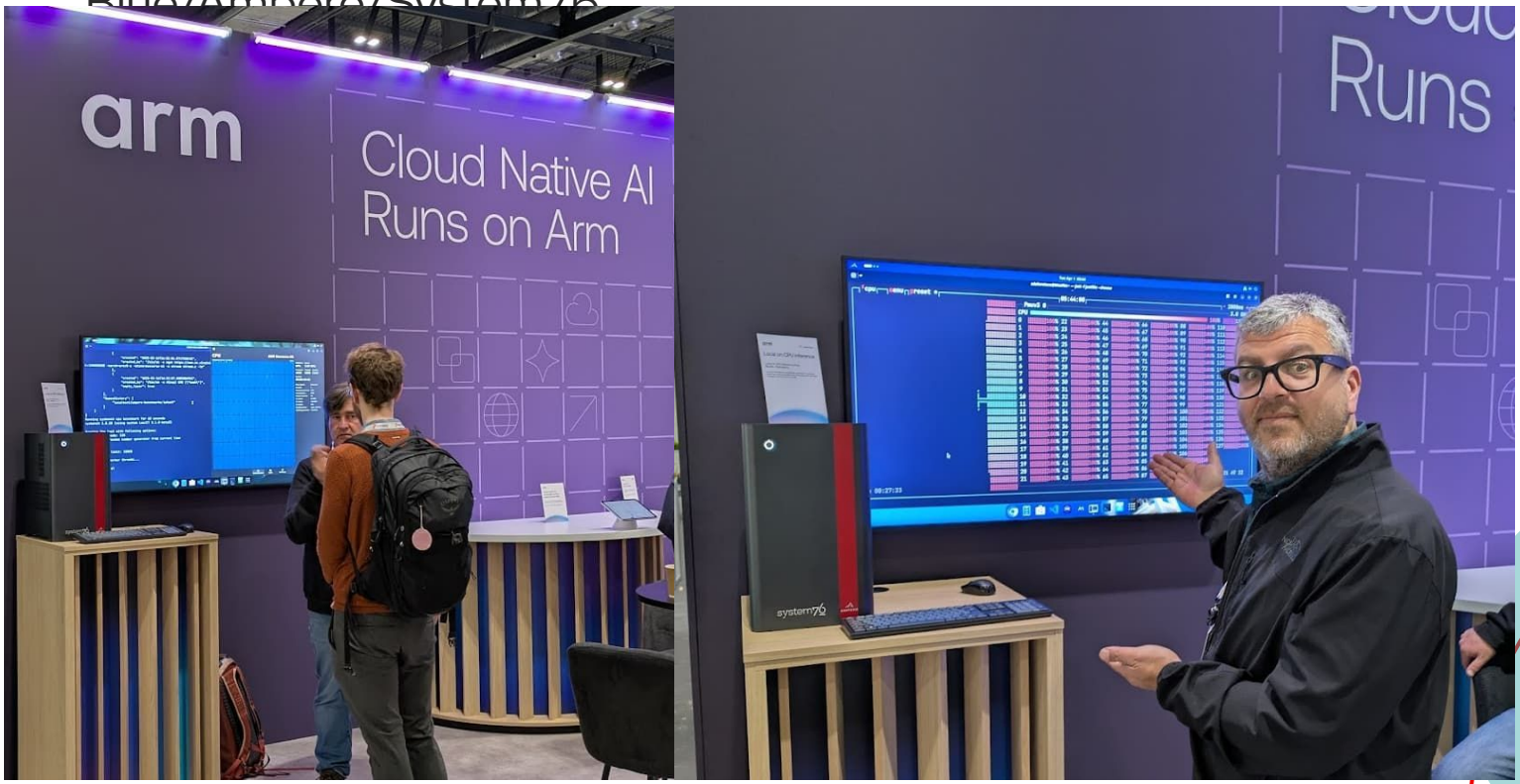
Red Hat

AI the open source way

Red Hat

Cloud Native AI Runs on Arm @ Kubecon

A collaborative effort between RamaLama/Universal Blue/Ampere/System76



Red Hat

Arm Support

NVIDIA

AMPERE

Vulkan

HUAWEI

CANN

Red Hat

Why Arm for AI?

- Power efficiency (AI workloads are power hogs)
- Multiple components for inference: GPU, NPU, CPU, etc.
- Scalable architecture from iot to edge to cloud (in terms of horizontal and vertical scalability)
- Edge AI brings lower-latency AI responses
- High memory bandwidth
- Unified memory

Why RamaLama for Arm?

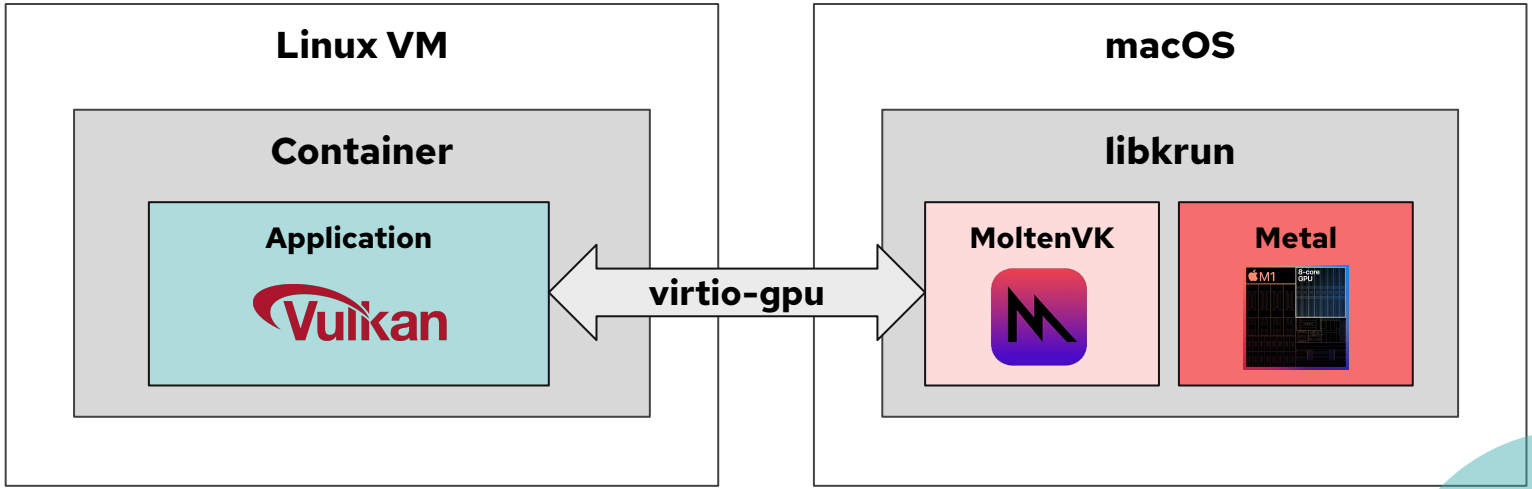
- Other AI tools tend to focus on Nvidia for x86_64, maybe macOS on ARM for Desktop usage, but often not Linux on ARM
- Avoid lock-in to a specific piece of hardware
- Avoid lock-in to a specific model registry (ollama, hf, dockerhub, etc.)
- Cloud native approaches, can use existing container infrastructure

Vulkan as common language for GPUs

- Supporting multiple frameworks is cumbersome.
  - Multiple container images.
  - Very large footprint.
- GLSL is expressive enough for implementing good performing shaders/kernels.
- Open Source stack (Mesa).

macOS: enabling GPU in containers

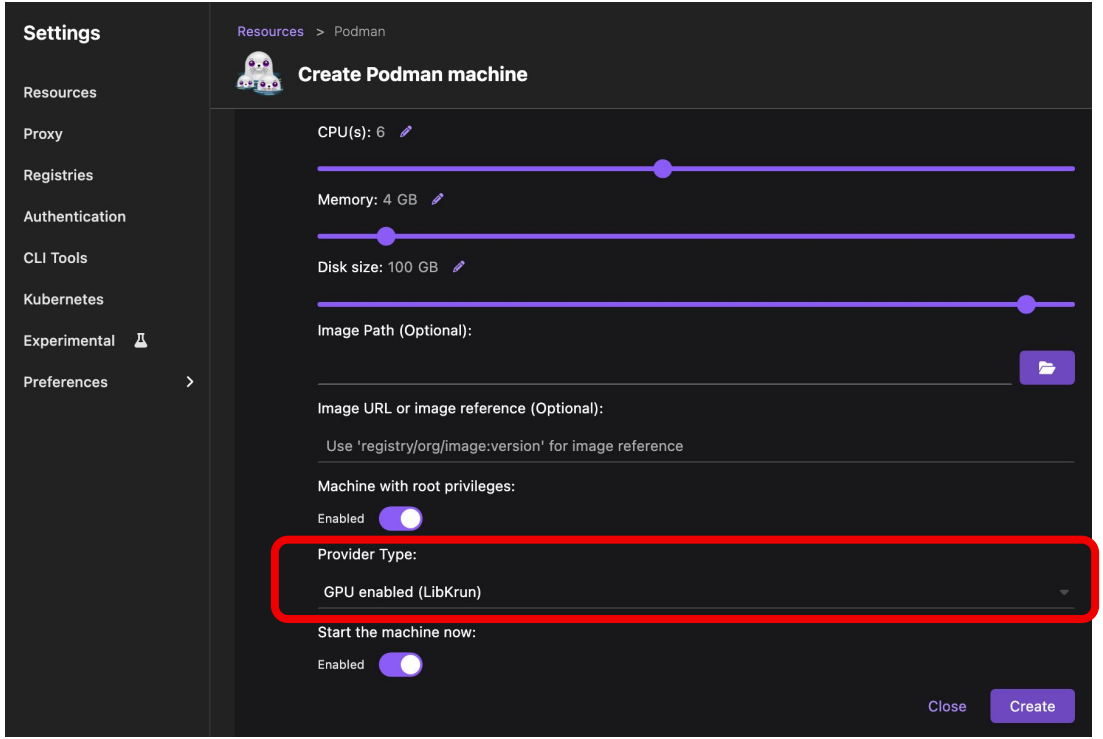
- Containers on macOS run in a VM



Red Hat

macOS: enabling GPU in containers

- Create a machine with libkrun as provider



Red Hat

macOS: running ramalama

- On macOS, execute “ramalama run gemma3”
- Inference engine (llama.cpp, whisper.cpp, etc.) will run on Linux within podman-machine

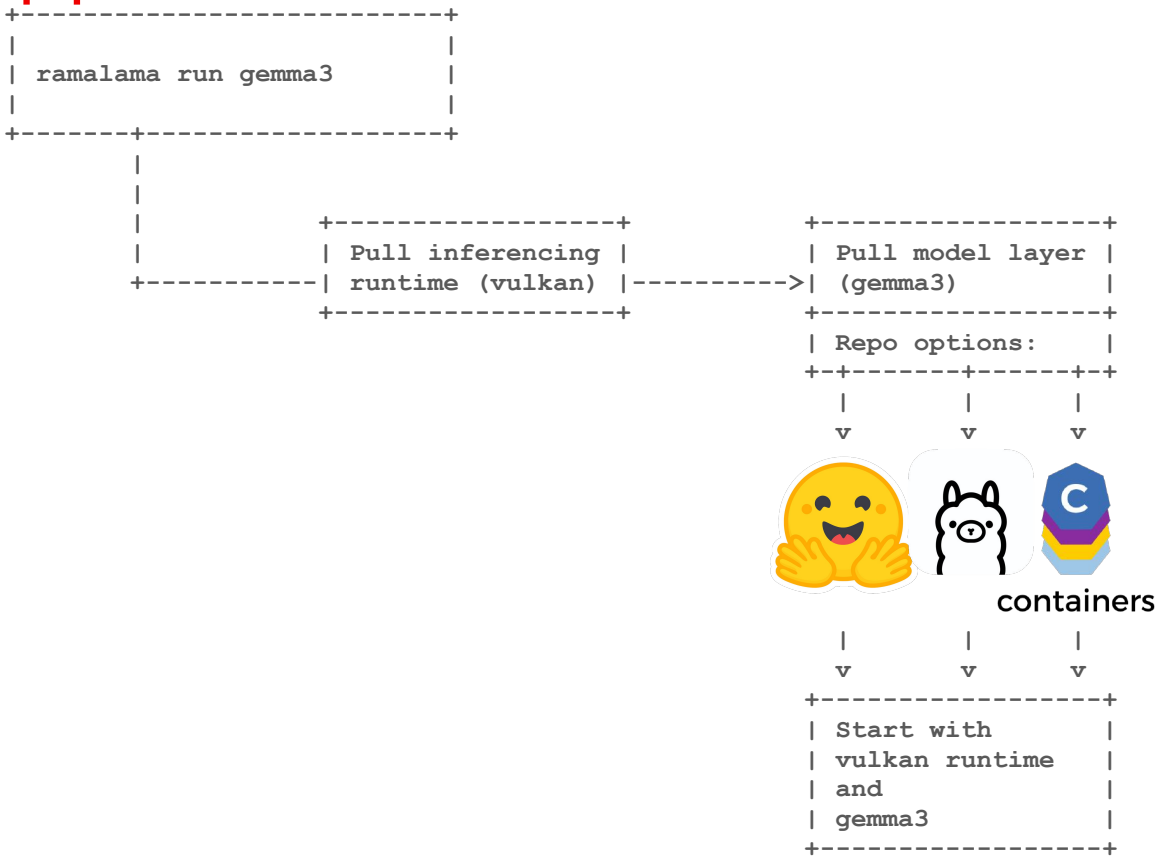
Linux llama.cpp binaries running on macOS?

- Yes and with accelerated GPU access

Linux llama.cpp binaries running on macOS?

- llama-server binary is build with vulkan support
- llama-server uses mesa implementation for vulkan
- This communicates with macOS via virtio-gpu
- Which speaks to MoltenVK
- MoltenVK acts as the translator, translates Vulkan to Metal so we can access the GPU

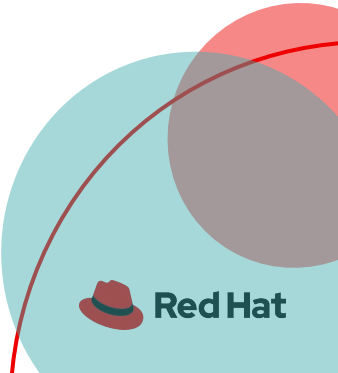
The Apple Solution



Red Hat

Demo

- Lets demo “ramalama serve gemma3” and Open WebUI



Join the conversation on Matrix!

Join Us!

github.com/containers/ramalama

https://ramalama.ai

Red Hat