From RustVMM to Kata-Containers: Securing Container Workloads with RISC-V H-ext Based Virtualization Software

Ruoqing He[1], Sheng Qu[1] and Yanjun Wu[1]

[1]Institute of Software, Chinese Academy of Sciences

Abstract

Our objective is to preemptively construct a complete Rust virtualization software stack for future RISC-V chips compliant with the RVA23 specification and server platform standards. Centered around the rust-vmm framework, we enable lightweight hypervisors like Dragonball, StratoVirt, Cloud-Hypervisor, and Firecracker—all designed to provide high performance, strong isolation, and virtualization-based security. Integrating these hypervisors with Kata Containers, we explore virtualization-based isolation of containerized workloads on RISC-V. By simulating hardware environments and leveraging forward-compatible software designs, we aim to be fully prepared for the introduction of real RISC-V hardware that meets RVA23 standard and RISC-V Server Platform specification. This ensures architectural parity in RISC-V's evolution with other architectures, provides plug-and-play software validation benchmarks for future hardware, and accelerates the maturation of the RISC-V server ecosystem from technical prototypes to production readiness through code-driven standardization collaboration

Introduction

Onboarding software like Kata-Containers [1] is critical for laying the foundation of a Confidential Computing software stack that will fully support the RISC-V architecture, equipped with H Extensions [2], AIA [3], and IOMMU [4] capabilities. As the computing industry moves toward more secure infrastructures, RISC-V presents a unique opportunity with its open-source instruction set architecture (ISA) and its potential to be tailored for security-sensitive applications. Our lightweight virtualization framework, built upon RustVMM [5], deeply integrates three critical hardware security features - RISC-V H (Hypervisor) Extension, Advanced Interrupt Architecture (AIA), and IOMMU. This integrated architecture delivers lightweight RISC-V virtualization solutions including Dragonball [6], StratoVirt [7], Cloud-Hypervisor [8], and Firecracker [9].These hypervisors, coupled with container runtimes like Kata Containers, provide the required isolation and performance needed in confidential computing environments, where the stakes for data privacy and security are higher than ever. By proactively developing this stack ahead of hardware availability, we ensure that once RISC-V chips with hardware extensions for virtualization and secure memory management hit the market, the ecosystem is "plug-and-play" ready.

Furthermore, by aiming to meet the RISC-V Server Platform Specification, our stack positions itself as an integral piece in the adoption and standardization of Confidential Computing across RISC-V-based server environments. We are creating an ecosystem that integrates secure VM isolation, efficient interrupt and I/O management, and flexible orchestration, all within a cloud-native framework. Our approach, reliant on Rust's inherent safety guarantees, provides a secure-by-design foundation that mitigates common vulnerabilities seen in traditional programming languages.

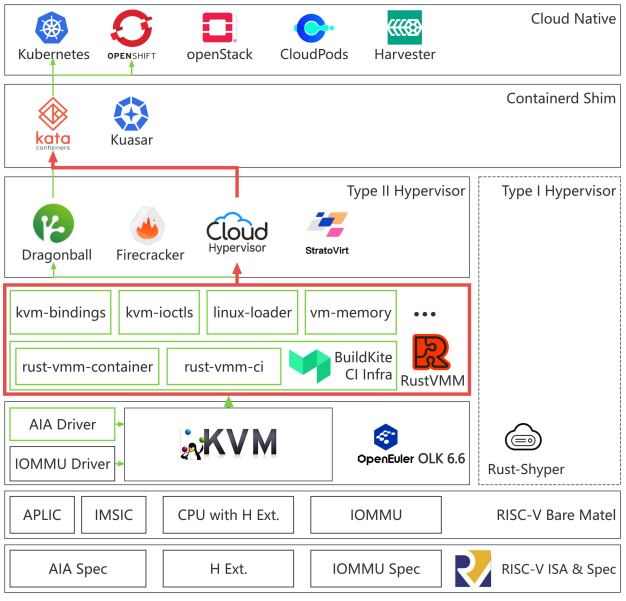

Figure 1. Secure Container or RISC-V Roadmap

This roadmap outlines our plan to enable secure container support on RISC-V. First, we will extend RustVMM's

compatibility to RISC-V architecture, including but not limited to adapting critical architecture-related crates. With RustVMM in place, we have chosen Cloud-Hypervisor as the primary virtual machine monitor (VMM) to be supported in our first stage. Cloud-Hypervisor delivers virtualization capabilities for software relies on that. Subsequently, we will integrate RustVMM and Cloud-Hypervisor into Kata-containers to provide secure container implementation on RISC-V platforms. We aim to establish a fully integrated, reliable, and thriving RISC-V virtualization software ecosystem that enables RISC-V to compete with x86 and ARM architectures in both virtualization and cloud-native software ecosystems.

This forward-looking initiative not only maintains synchronization with emerging hardware advancements but also proactively shapes the software ecosystem required for secure, scalable, and highly reliable RISC-V server platforms. It empowers global cloud service providers, enterprises, and developers to fully unlock RISC-V architecture's potential, transforming its open customization capabilities and security-enhanced features into core competitive advantages for building next-generation confidential computing infrastructure and adaptive cloud architectures.

Methodologies

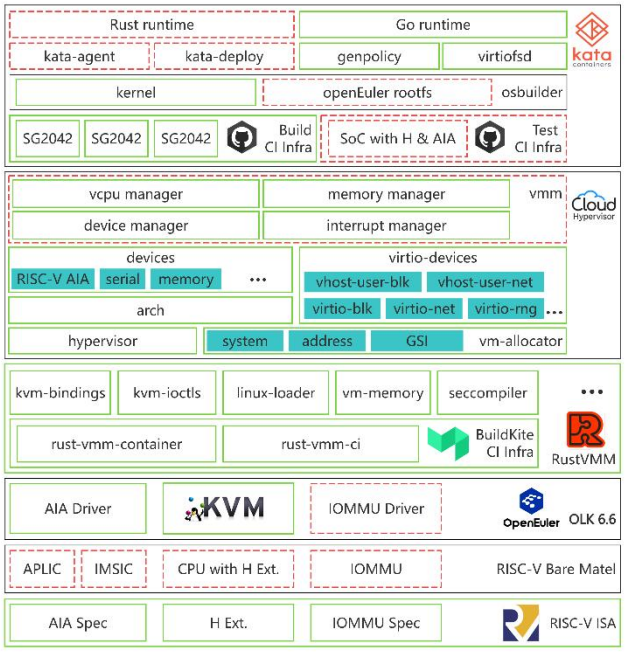

Figure 2. Upstream status

As shown in upstream status, we follow the principle of "upstream first". We have submitted our work (marked with green boxes in openEuler, RustVMM, Cloud-Hypervisor and Kata Containers) to communities concerned, and most of them are accepted. The rest is still under heavy development and should be ready soon.

Kata Containers

The diagram visualize the differences between Kata Containers and traditional containers in implementing secure containers. As shown, Kata Containers leverage a hypervisor-based architecture to establish secure container infrastructure. This design significantly strengthens container security boundaries through hardware-enforced isolation while maintaining seamless compatibility with traditional container ecosystems. Such an architecture enables the team to build cloud-native standards-compliant secure virtualization layers for RISC-V based on Kata Containers. This approach not only enhances RISC-V's competitiveness in critical workload scenarios but also provides an extensible technical foundation for unified management across heterogeneous architectures.

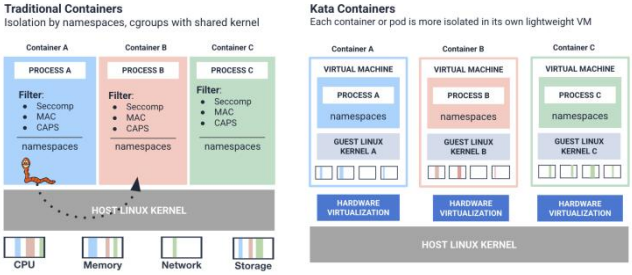

Figure 2. Difference between traditional & Kata containers

RustVMM

Due to the absence of RISC-V SoCs with both AIA and IOMMU ready, we are facing great challenges while trying to upstream our work, since there is no real hardware we could use for integrating into communities' CI instances. We managed to address this problem by using QEMU to provide full-emulated RISC-V virt board with AIA and IOMMU in place to illustrate our works are theoretically correct, and get RISC-V code to merge and evolve with other architectures.

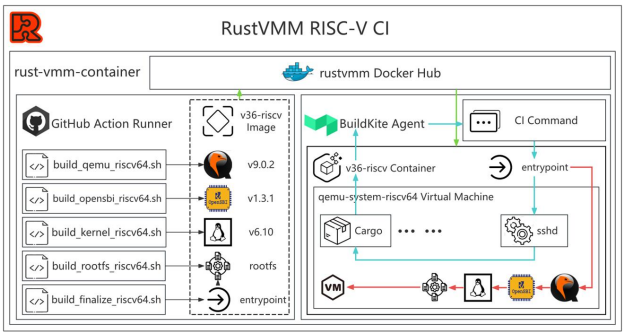

Figure 3. RISC-V CI for RustVMM community

Currently, we have successfully introduced RISC-V architecture support for RustVMM components and facilitated the upstream merging of relevant code to achieve full compatibility with the RISC-V architecture. The RISC-V CI (Continuous Integration) for RustVMM was officially launched on September 2, 2024. Since September 23,

critical core repositories such as kvm-bindings and kvm-ioctls have successively released versions with RISC-V support. This work establishes a critical technical foundation for virtualization development on RISC-V, and makes RISC-V the third officially supported architecture.

Cloud- Hypervisor

We took a similar approach in Cloud-Hypervisor comunity to bring up CI for RISC-V. With these CIs in place, we launched RISC-V support for architecture-dependent crates in RustVMM community, which made RISC-V the third architecture officially supported. The completion of RustVMM RISC-V enables more than 1,900 projects (hosted on GitHub) to be able to support RISC-V.

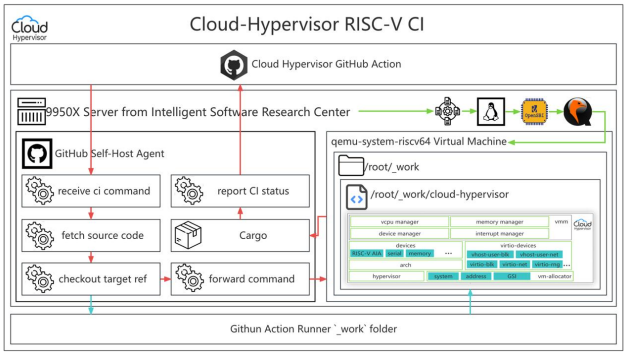

Figure 4. RISC-V CI for Cloud-Hypervisor community

Cloud-Hypervisor here as a outstanding hypervisor which is capable of working with Kata-Containers is supported on RISC-V architecture. The overall structure of Cloud-Hypervisor is refactored along the process of supporting RISC-V. After its v45 release, Cloud-Hypervisor is RISC-V enabled and Kata-Containers RISC-V integrated.

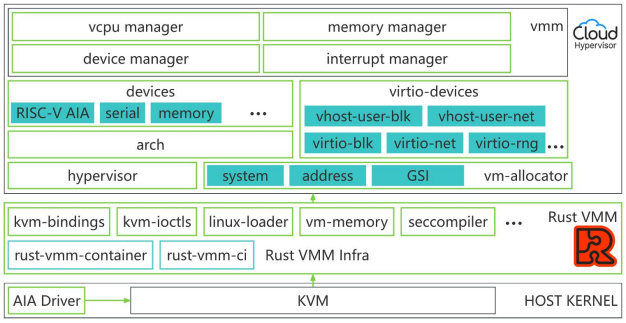

Figure5. Architecture of Cloud-Hypervisor Project

The above secure container software stack owe its secutrity primarily to RISC-V H extension. Even if SoCs with H extension and other features specified in RVA23 profile have not yet come out, we will continue to expand the virtualization software ecosystem of RISC-V, and explore the ways it could be used in secure containers. These works will be firstly tested, verified and distributed on openEuler [10], and extended to other distibutions.

Discussion

The key value of this work lies in paving the way for future RISC-V hardware through a software-first strategy during the early stages of the RISC-V hardware ecosystem's immaturity. Despite the current absence of RISC-V hardware simultaneously supporting the H-extension, Advanced Interrupt Architecture (AIA), and IOMMU, we have established comprehensive RISC-V CI pipelines within the RustVMM and Cloud-Hypervisor communities using a fully emulated QEMU environment. This ensures architectural-level code evolves in sync with x86/ARM counterparts. These achievements not only validate the feasibility of the RISC-V virtualization software stack but also provide plug-and-play software validation benchmarks for hardware vendors through code-driven standardization collaboration, significantly accelerating the RISC-V server ecosystem's transition from prototypes to production readiness.

Regarding community contributions, we have spearheaded RISC-V support in three core open-source communities: RustVMM, Cloud-Hypervisor, and Kata Containers. Specifically:

1. RustVMM: Established RISC-V as the third officially supported architecture after x86 and ARM, completed adaptations for critical libraries such as kvm-bindings and kvm-ioctls, and extended its CI coverage to over 1,900 downstream projects.
2. Cloud-Hypervisor: Implemented architectural refactoring to enable RISC-V support, making it the first virtualization solution integrated with Kata Containers on RISC-V.
3. Kata Containers: Built cloud-native standards-compliant secure container infrastructure for RISC-V by leveraging hypervisor-enforced isolation.

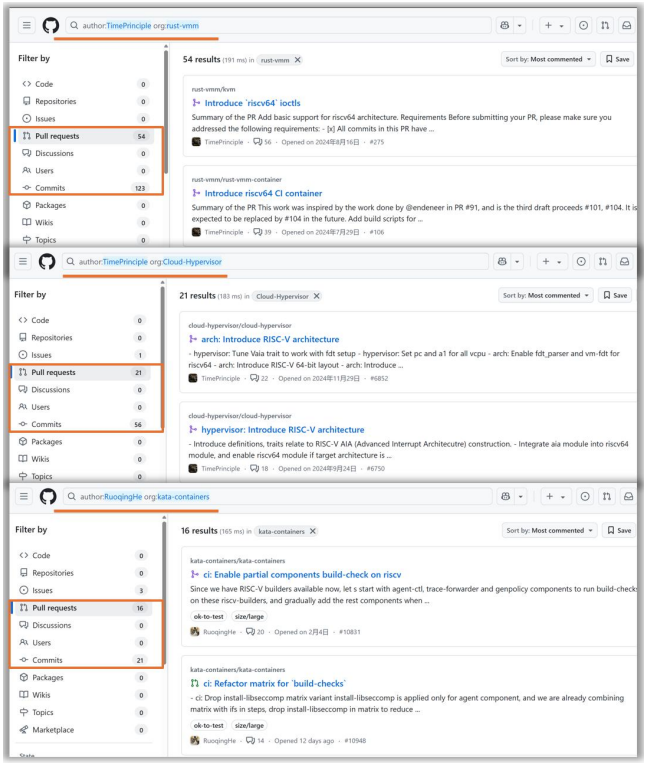

Figure6. Details of the Code Contributions Across Three Communities

We have dedicated to this course for a year, that nearly all RISC-V-related code upstreaming, CI/CD implementations, and architectural designs in these three communities originate from our team. This deep engagement has not only accelerated the maturation of the RISC-V virtualization ecosystem but also solidified our technical leadership within the open-source community. These efforts establish an indispensable software foundation for RISC-V to compete in confidential computing and cloud-native domains.

References

[1] Kata-Containers: https://katacontainers.io/.

[2] RISC-V H Extension: https://github.com/riscv/riscv-isa-manual.

[3] RISC-V AIA: https://github.com/riscv/riscv-aia.

[4] RISC-V IOMMU: https://github.com/riscv-non-isa/riscv-iommu.

[5] Rust-VMM: https://github.com/rust-vmm.

[6] Dragonball: https://github.com/kata-containers/kata-containers/tree/main/src/dragonball.

[7] StratoVirt: https://gitee.com/openeuler/stratovirt.

[8] Cloud-Hypervisor: https://www.cloudhypervisor.org/.

[9] Firecracker: https://firecracker-microvm.github.io/.

[10] openEuler: https://www.openeuler.org/en/.