Kata Runtime-rs with CoCo Support

Author: Alex Lyn(LiYa'nan) Ant Group & Kata Containers

Runtime-rs Architecture Evolution, Features, and Advantages

1. Why we chose to refactor Kata Runtime with Rust

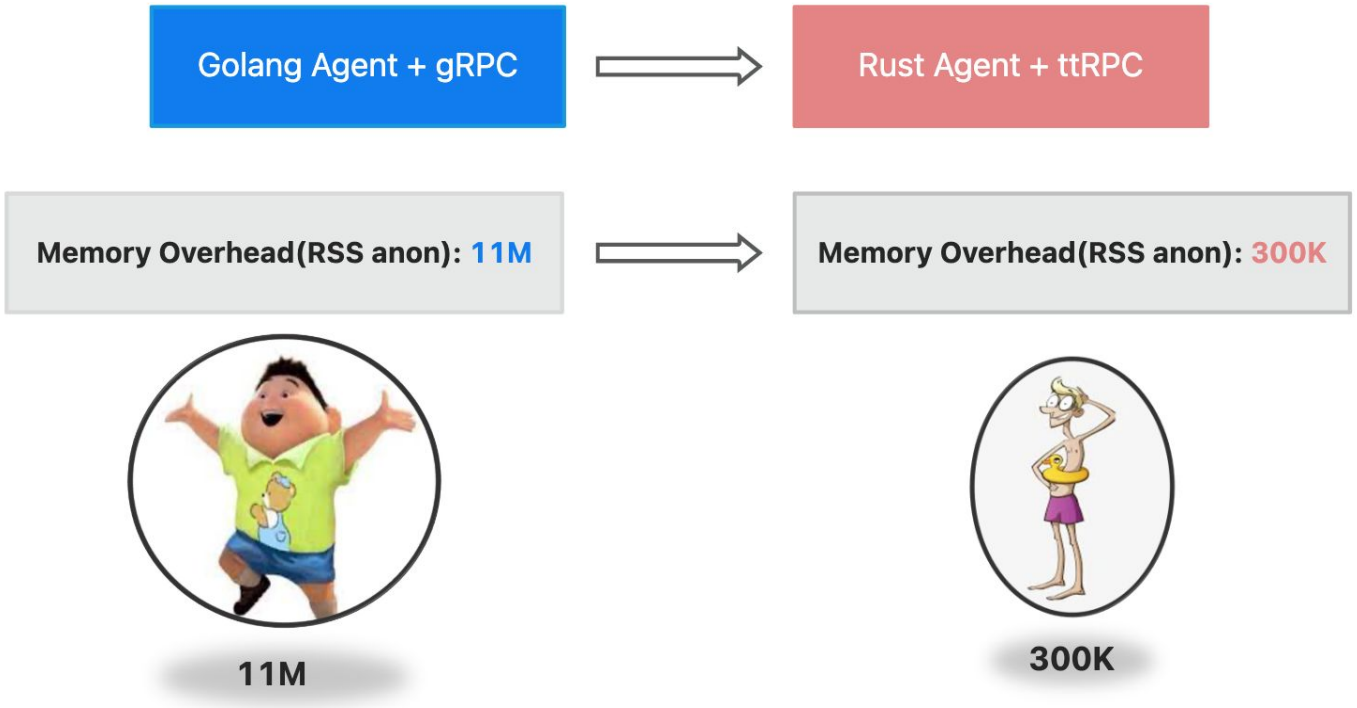
"Did we choose Rust because it's fashionable? No! We chose it to solve prob lems. We notably saw a significant reduction in memory usage with **kata agent(rust)**.

Years ago, to address the excessive memory consumption of the **golang-based kata-agent**, which limited high-density Kata deployments within Ant Group, we refactored the agent using rust. This significantly reduced **RSS anonymous memory usage** and substantially increased per-node deployment density.

This is a significant and beneficial attempt to refactor Go code with Rust, bringing us significant benefits.

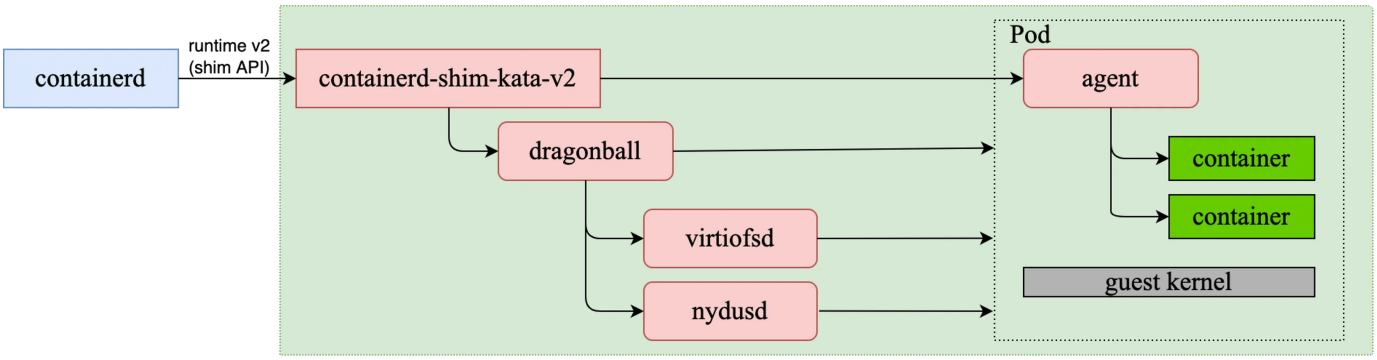
"The main benefit users will see is a 10-fold improvement in size, from 11MB to 300KB. "

> It comes from the kata blog 21/10/2020



2. Kata runtime-rs Architecture

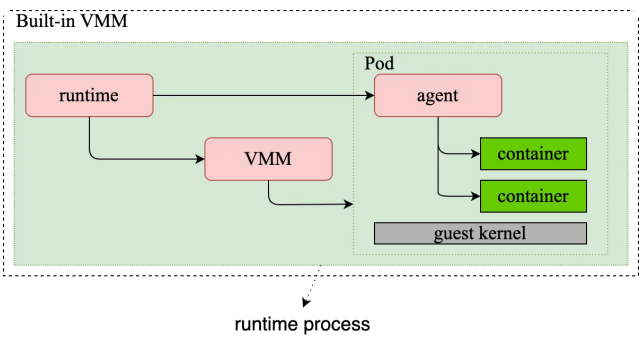
Kata `runtime-rs` is a **single-process architecture version** open-sourced in 2022 by **Ant Group/Alibaba Cloud**, based on our internal implementation of `RunD` . It aims to significantly optimize performance and reduce resource overhead.



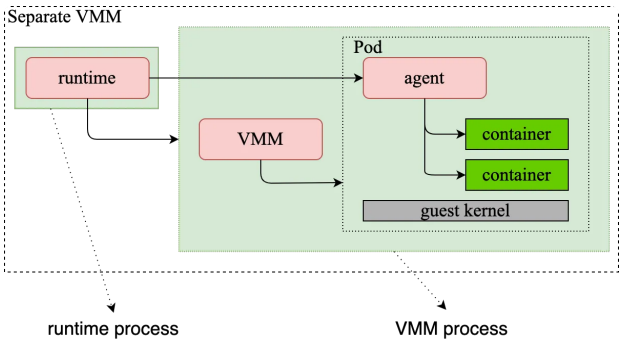
Its key features include:

- **Simplified Control Plane**
- **Built-in Image Management**
- **Improved Performance and Reduced Overhead**
- **Support Builtin and Optional VMMs**

Builtin Dragonball



Qemu/Cloud-hypervisor/Firecracker



> Detailed Architecture about rust runtime, Please see Kata runtime-rs Architecture

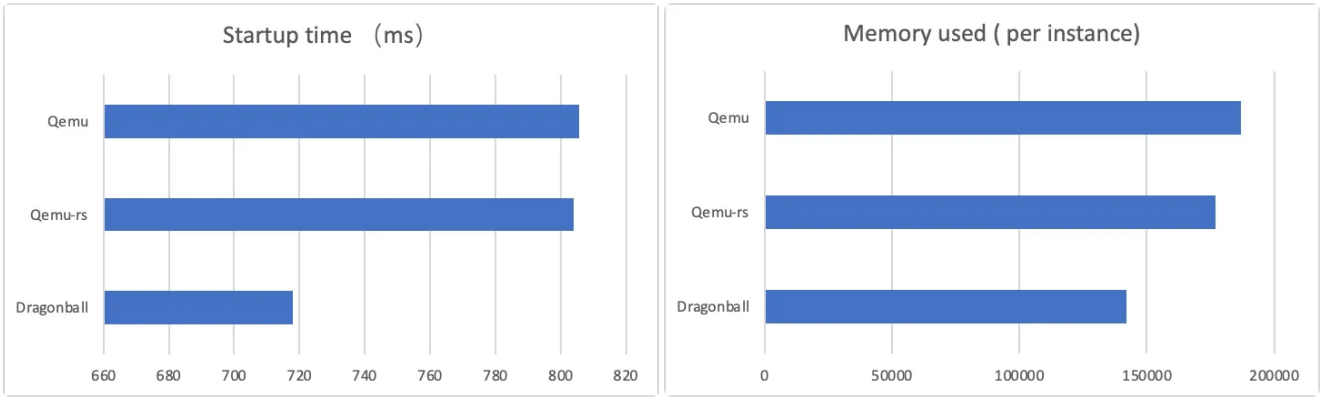
3. Performance/Resource Overhead Comparison

3.1 Startup Time and Memory Footprint

This test aims to roughly measure the overhead of running Kata Containers, focusing on individual container startup time and the aggregate memory consumption when running multiple Kata Pods.

Metric	Dragonball	Qemu-rs	Qemu
Startup time	718.061872 ms	803.950059 ms	805.577514 ms
Memory used (per instance)	141944.64(k)	176848.56(k)	186853.6(k)

> Qemu-rs is for runtime-rs and qemu, Qemu is for runtime-go and qemu



Analysis of the Result:

(1) Startup Time: (Dragonball provides a faster VM boot experience.**)**

- **Dragonball** is the clear winner with a startup time of **718.06 ms**.
- **Dragonball is approximately 10.9% faster than Qemu**.

(2) Memory Used (Dragonball is the most memory-efficient**):**

- **Dragonball** uses approximately **19.8% less memory per instance than Qemu-rs**.
- **Dragonball** uses approximately **24.0% less memory per instance than Qemu**.

(3) Conclusion:

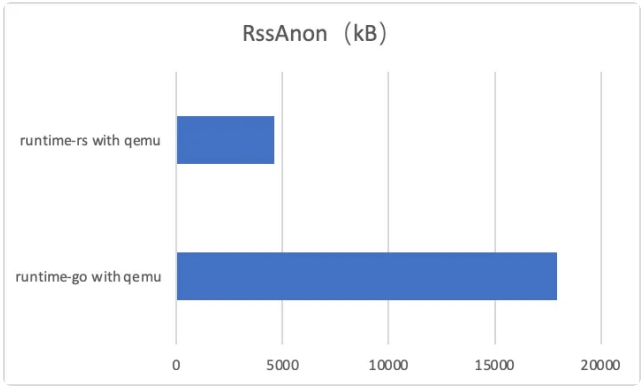
Dragonball demonstrates better performance in both startup time and memory efficiency, **indicating benefits from its builtin VMM implementation.**

3.2 RssAnon used by the shim-kata-v2 process

To offer a more precise calculation of resource consumption(**Memory Consumption**), we measured the `RssAnon` (**Resident Set Size – Anonymous**) memory occupied by `containerd-shim-kata-v2` process.

It's often a good indicator of the actual memory footprint of a process's core operations and its runtime environment.

containerd-shim-kata-v2	RssAnon
runtime-go with qemu	17928 kB
runtime-rs with qemu	4596 kB



(1) Comparison Data:

- **`containerd-shim-kata-v2`** **(runtime-go with qemu):** 17928 kB `RssAnon`
- **`containerd-shim-kata-v2`** **(runtime-rs with qemu):** 4596 kB `RssAnon`

(2) From the data: The `runtime-rs` uses dramatically less `RssAnon` memory compared to the `runtime-go`. **Significant Reduction as below:**

- **Memory Savings:** 17928 kB − 4596 kB = 13332 kB
- **Percentage Reduction:** (13332 kB / 17928 kB) × 100% ≈ 74.37%

Specifically, the `runtime-rs` uses approximately **74.4% less anonymous resident memory**.

(3) Conclusion:

- The `containerd-shim-kata-v2` process, when utilizing the `runtime-rs`, demonstrates a **drastic reduction** in its `RssAnon` memory footprint compared to the `runtime-go`.

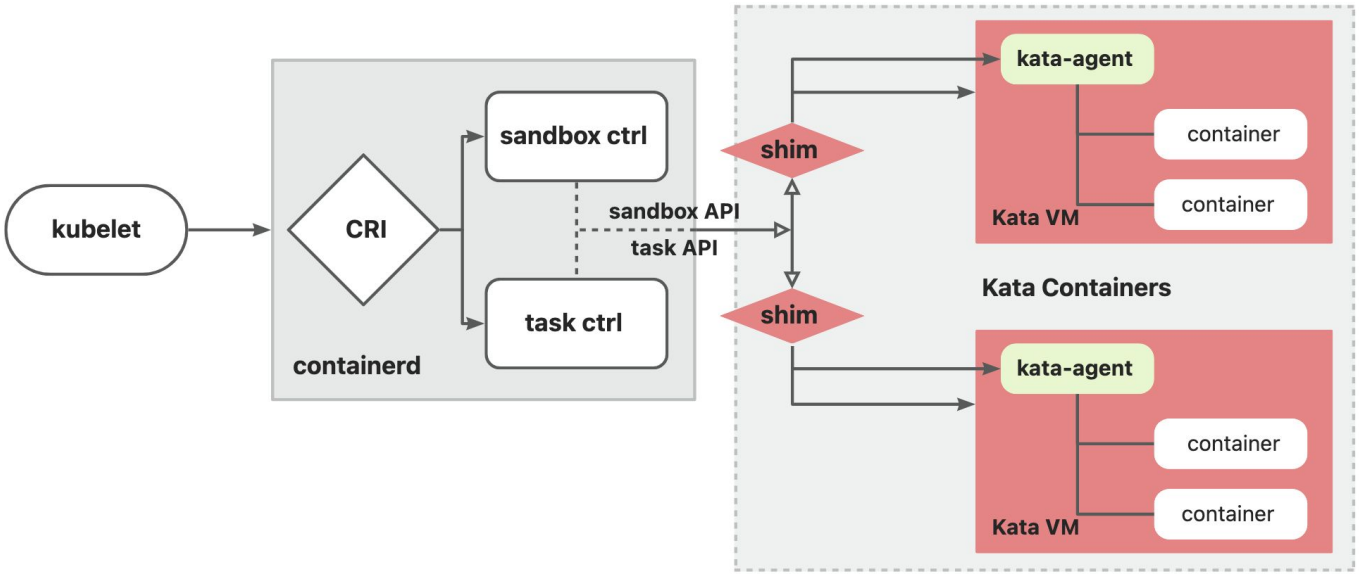
- This result also strongly supports the decision to refactor components like the Kata Agent (or underlying runtime shims) from Go to Rust, as previously discussed.

The main benefit users will see is a 3.9-fold improvement in size, from 17.5MB to 4.49MB.

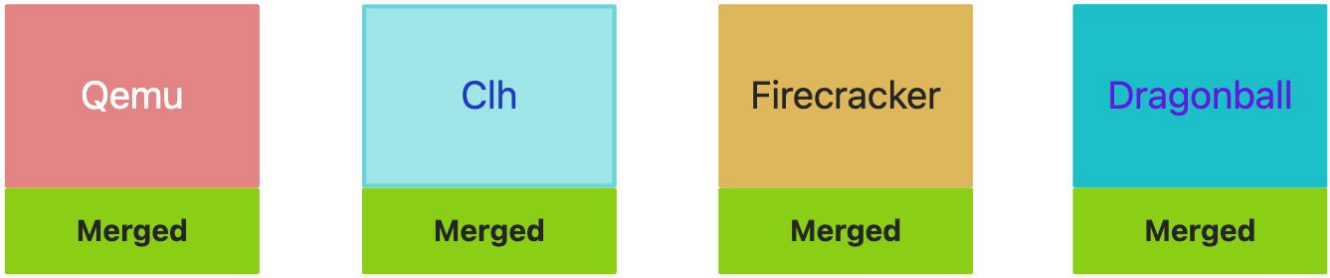
> 17928 kB/4596 kB = X ≈ 3.90

4. Features in runtime-rs

4.1. Support for containerd 2.x and SandboxAPI



4.2. Support for multiple VMMs



4.2.1. Feature List or Gaps between Runtime-go and Runtime-rs

Reference：https://github.com/kata-containers/kata-containers/issues/8702

- vhost-vsock/hybrid-vsock

- virtio-blk/virtio-pmem/virtio-scsi/vhost-user-blk
- virtio-net/vhost-net/vhost-user-net/directly attachable network(DAN)
- network modeil: Tc filter/none
- net endpoint: veth/physical/vlan/tap/Tuntap/IPVlan/MacVlan/
- virtio-fs and inotify/watchable/vhost-user-fs
- sandbox bind mounts
- virtio-fs rootfs/nydus rootfs/block rootfs
- virtio-mmio device
- resize cpus/memory, cpu hotplug/unplug, CPU Topology Passthrough
- virtio-mem/virtio-balloon
- Tracing/Event/Kata-Monitor/Metrics
- VFIO device hotplug
- device manager/PCIe Topology
- Propagate k8s configs when sharefs is disabled
- measured Rootfs
- docker/nerdctl/containerd/crio
- initdata Spec (WIP)
- guest image pull (WIP)

4.2.2. TEE support

- AMD SEV/SEV-SNP（**Pavel Mores / RedHat**）
- Intel TDX (WIP) （**Alex Lyn / AntGroup**）

4.2.3. Supported Archs

- x86
- s390x
- ppc64le
- aarch64

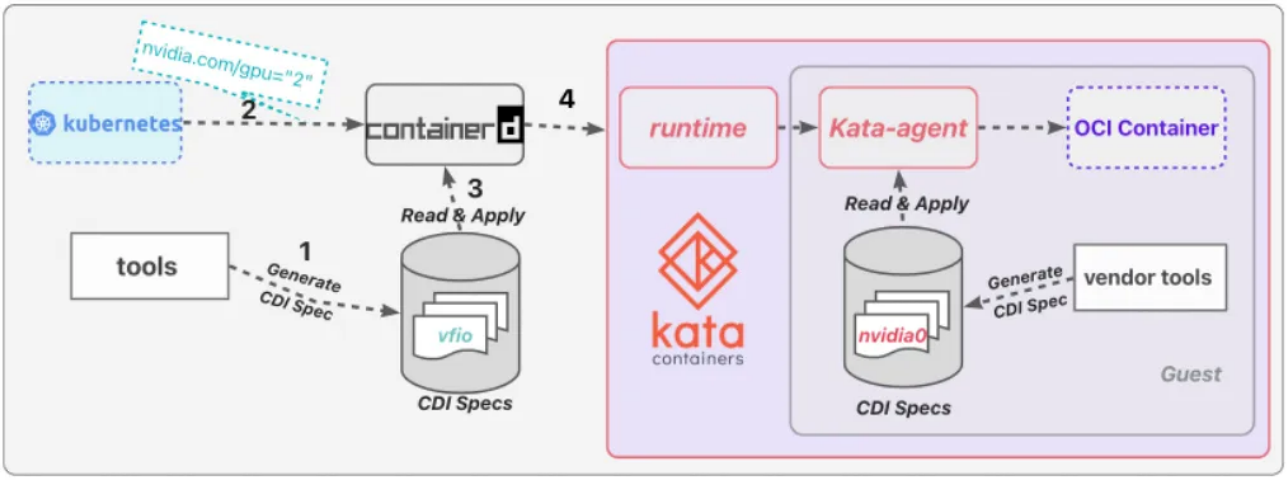
4.3. Support Remote Hypervisor

Support for `Peer Pod` solutions, to address issues where cluster worker nodes do not support nested virtualization or confidential computing.

4.4. Introduce CDI (Zvonko Kaizer/Alex Lyn)

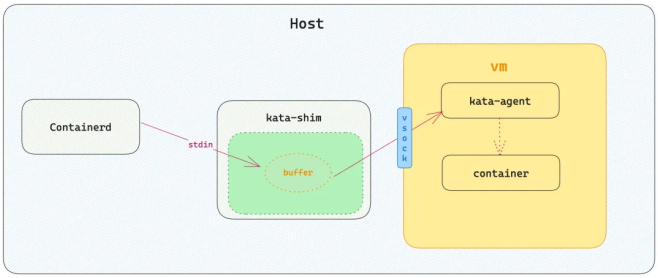
Two-tier CDI System

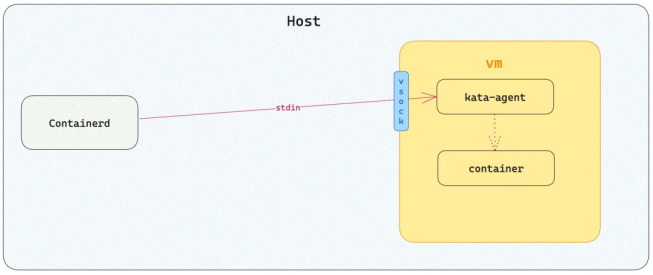
- **Outer CDI (Containerd 1.7+)** Responsible for mapping hardware devices into the OCI Spec as VFIO devices to achieve device passthrough to the Guest.
- **Inner CDI (Kata-Agent)** Responsible for mapping devices and their related Mounts, Envs, and Hooks files to the OCI Spec, ensuring the devices are correctly exposed to the container.

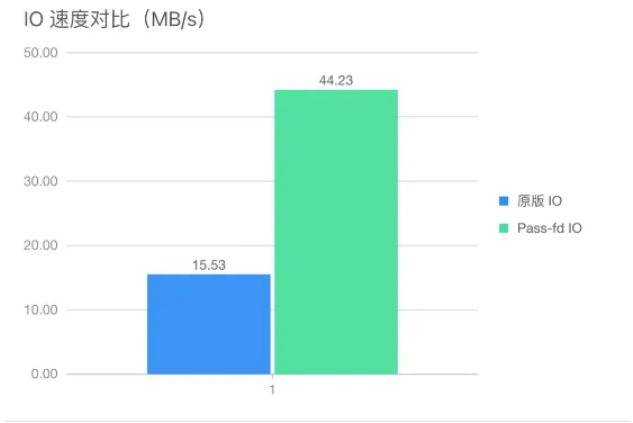


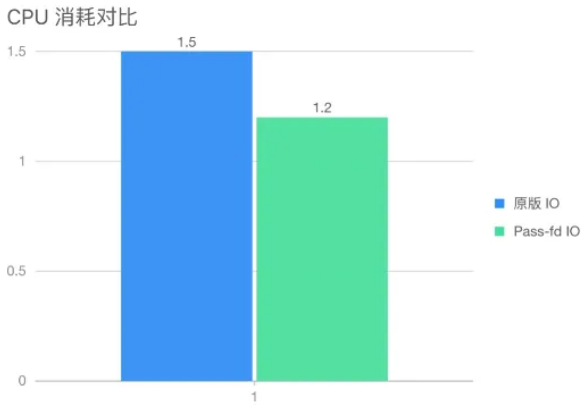
4.5. Optimized Kata I/O performance using passthrough-fd (FupanLi)

Passthrough-fd is an I/O optimization developed from AntGroup internal practices, significantly enhancing Kata IO performance.









Performance Comparison

- **I/O Speed Comparison:** Passthrough-fd I/O is 185% faster than Legacy I/O.
- **CPU Utilization Comparison:** Passthrough-fd I/O saves 20% CPU compared to Legacy I/O.

Kata Runtime-rs in Internal Practice

1. General Scenario Practices based on runtime-rs

- **Mixed Deployment Isolation:**
  - Utilizing its excellent **performance and fault isolation** capabilities to deploy runC containers and Kata containers with **different SLOs** (Service Level Objectives) simultaneously on the same node.
- **Ant Group Container Security product:**
  - **Ant Group Cloud Workload Protection Platform (AntCWPP) ,** implemented based on **Kata** and **eBPF** technologies. Using Kata security containers and Linux kernel eBPF technology to build a rich-featured, stable, efficient, and secure CWPP system integrated into internal infrastructure.

2. Internal Practices of CoCo based on kata runtime-rs

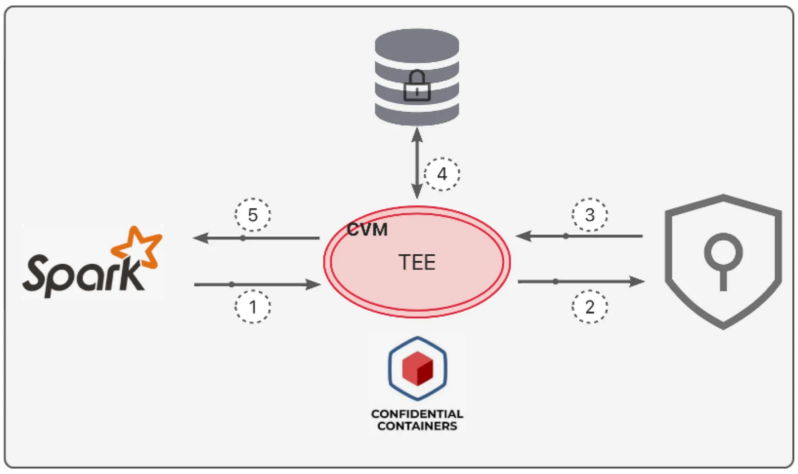
2.1 Typical Scenario:

(1) The Internal business need is to securely run Spark jobs within a Kubernetes cluster for distributed data processing, particularly for machine learning or identity matching.

(2) The crucial demand is to ensure that **sensitive datasets remain continuously encrypted from their original trusted source to the cloud environment, and are only processed in plaintext *inside* a TEE ENV.**

(3) This guarantees the confidentiality and integrity of highly sensitive data throughout its lifecycle in the cloud.

The process leverages CoCo' remote attestation and secure key release capabilities to achieve this secure data processing workflow:



(1) Schedule Spark jobs onto TEE nodes.

(2) Request a key before precede any decryption attempt.

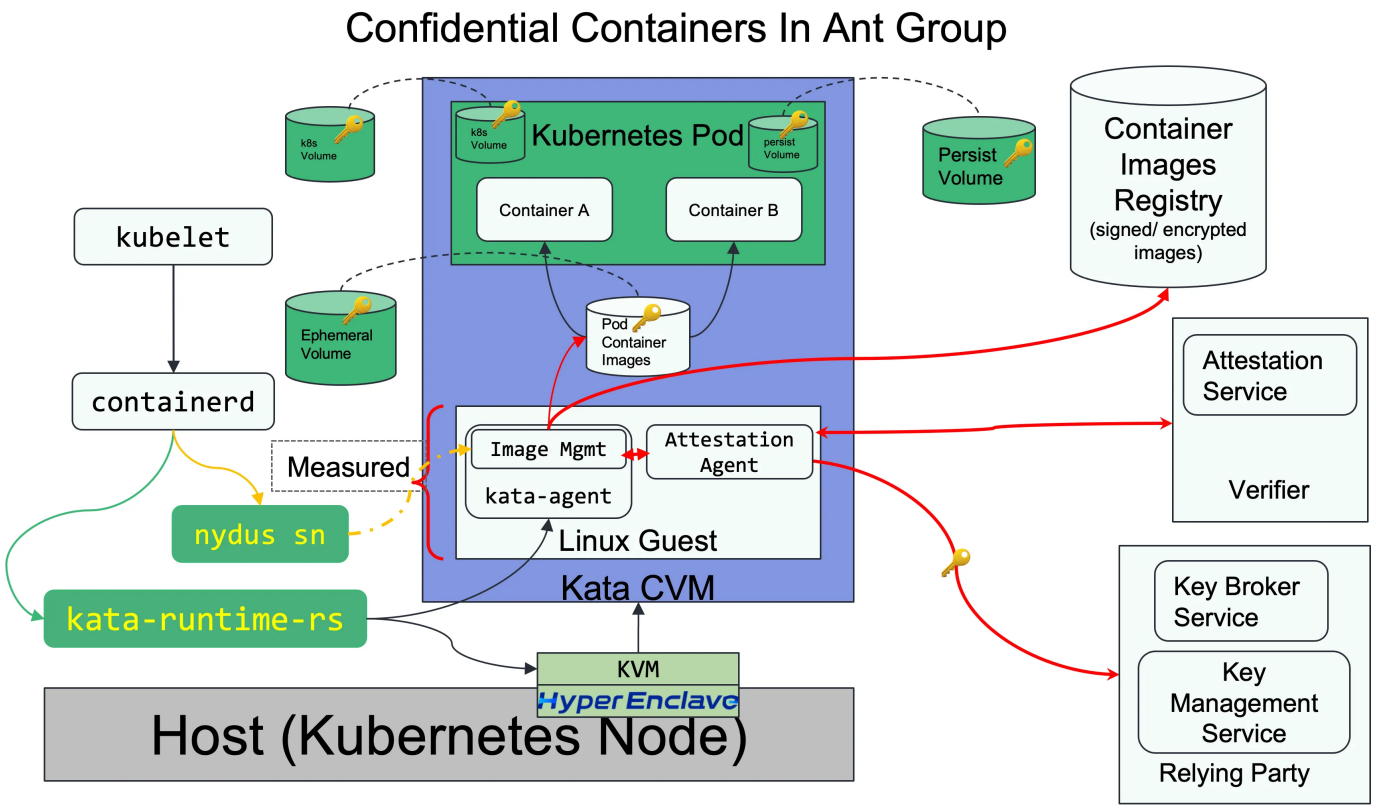
(3) Do release secure Key to TEE.

(4) Decrypt data In-TEE & Process.

(5) Encrypt results output and return them.

2.2 Confidential Big Data Analytics based on CoCo

While our internal CoCo implementation largely aligns with the community's architecture, including boot measurement, remote snapshotter (Nydus), guest image management, and Attestation-Agent/AS/KMS, we also undertook additional development efforts. These efforts primarily aim to bridge the current gaps between `runtime-rs` 's CoCo feature implementation and kata `runtime-go` .



(1) **Kata CVM: kata runtime-rs / dragonball for CoCo**

(2) **Runtime-rs supports guest pull: kata virtual volume supported**

(3) **Disable virtiofs with `shared_fs = none`**

(4) **Propagate k8s configs:** service account, secrets, configmap when sharefs is disabled.

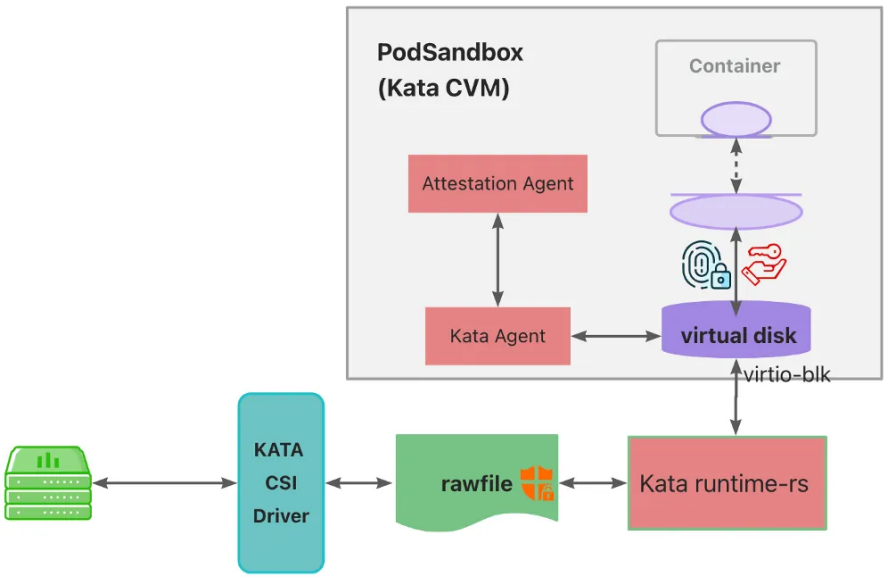
(5) **InitData: pass private registry auth files and etc-hosts via initdata**

(6) **Ephemeral Storage / Persistent Storage / emptyDir**

To support **Ephemeral Storage with Mount device not volume,** we extended support for target devices

- The csi-kata-directvolume is responsible for dynamically allocating block device volumes:
- Kata `runtime-rs` uses `virtio-blk` to hot-plug devices into the VM.

- Kata-agent initializes the storage device, obtains the key from the Attestation Agent (AA), and decrypts the data.



(7) **GPU support with CDI**

In our scenarios, we also leverage GPU acceleration to improve data processing efficiency, which involves GPU device passthrough. This part is already integrated with our existing infrastructure, so we directly reuse the CDI solution to achieve device passthrough capabilities. And we also need a device plugin responsible for device allocation. Please see the kata-xpu-device-plugin

Runtime-rs Support for Confidential Containers (CoCo) – Work in Progress

1. VMM

- qemu-rs
  - SEV-SNP (**Pavel Mores/Redhat**)
  - Intel TDX (**Alex Lyn/AntGroup**)
- cloud-hypervisor Actively supporting Intel TDX
- dragonball (**WIP**)

2. TEE Platform

- TDX
  - https://github.com/kata-containers/kata-containers/pull/11179 [CR]
- SEV-SNP
  - https://github.com/kata-containers/kata-containers/pull/10819 [Merged]
  - https://github.com/kata-containers/kata-containers/pull/10941 [Merged]

3. Guest Pull

- shared_fs = "none"
  - https://github.com/kata-containers/kata-containers/pull/10697 [CR]
- guest image pulling
  - https://github.com/kata-containers/kata-containers/pull/10698 [CR]

4. Propagate k8s configs

- secrets, configmap, serviceaccount, downward-api
  - https://github.com/kata-containers/kata-containers/pull/11240 [Merged]

5. InitData inject configurations

- https://github.com/kata-containers/kata-containers/pull/11181 [CR]

6. Remote Hypervisor

- https://github.com/kata-containers/kata-containers/pull/10225 [Merged]

Test Methodology

1. Startup Time and Memory Footprint

- kata release 3.17

https://github.com/kata-containers/kata-containers/releases/download/3.17.0/kata-static-3.17.0-amd64.tar.xz

- Kata Pod Specifications

```python
# Default number of vCPUs per SB/VM:
# unspecified or 0                --> will be set to 1
# < 0                             --> will be set to the actual number of physical cores
# > 0 <= number of physical cores --> will be set to the specified number
# > number of physical cores      --> will be set to the actual number of physical cores
default_vcpus = 1

# Default memory size in MiB for SB/VM.
# If unspecified then it will be set 2048 MiB.
default_memory = 2048
```

- Test Methodology：
  - Measure the total time elapsed from the execution of the `ctr run` command for a Kata container until the test command (`uname -r`) inside the container completes and returns its output.
  - Observe the memory change of the entire node when 50 pods are run using `free -k` . Focus on the reduction in `memory free` to roughly estimate the memory occupied by running 50 pods on the node.

```python
sudo ctr run --runtime io.containerd.kata.v2 --rm -t alpine:latest "$CONTAINER_NAME" /bin/sh -c "uname -r"
```

- Container image："m.daocloud.io/docker.io/library/alpine:latest"

2. RssAnon used by the shim-kata-v2 process

```Python
# accordingly set configuration.toml for dragonball,qemu-rs and qemu
[root@pek1000 /home/alex.lyn] # nerdctl run --runtime io.containerd.kata.v2 -ti --rm --net=none --name=dragonball-test m.daocloud.io/docker.io/library/fedora:41
[root@pek1000 /home/alex.lyn] # nerdctl run --runtime io.containerd.kata.v2 -ti --rm --net=none --name=qemu-test m.daocloud.io/docker.io/library/fedora:41
[root@pek1000 /home/alex.lyn] # nerdctl run --runtime io.containerd.kata.v2 -ti --rm --net=none --name=qemu-rs-test m.daocloud.io/docker.io/library/fedora:41
[root@pek1000 /home/alex.lyn] #nerdctl ps
CONTAINER ID    IMAGE                                             COMMAND
       CREATED              STATUS    PORTS    NAMES
3b989592cb45    m.daocloud.io/docker.io/library/fedora:41    "/bin/bash"    About an hour ago    Up                dragonball-test
3e5f15741ceb    m.daocloud.io/docker.io/library/fedora:41    "/bin/bash"    About an hour ago    Up                qemu-test
2d6de0a82cdc    m.daocloud.io/docker.io/library/fedora:41    "/bin/bash"    About an hour ago    Up                qemu-rs-test
[root@pek1000 /home/alex.lyn] #pgrep -f 2d6de0a82cdc
30699
..
[root@pek1000 /home/alex.lyn] #PIDX=30699
[root@pek1000 /home/alex.lyn] #cat /proc/${PIDX}/status |grep RssAnon
RssAnon:        4596 kB
[root@pek1000 /home/alex.lyn] #pgrep -f 3e5f15741ceb
64905
..
[root@pek1000 /home/alex.lyn] #PIDX=64905
[root@pek1000 /home/alex.lyn] #cat /proc/${PIDX}/status |grep RssAnon
RssAnon:       17928 kB
```