Red Hat

CONFIDENTIAL CONTAINERS

Five big problems with confidential containers

and why we need KVM developers to help

Christophe de Dinechin
Senior Principal Software Engineer
dinechin@redhat.com

Agenda

Key topics we are going to cover today

- Quick primer on confidential containers

Agenda

Key topics we are going to cover today

- Quick primer on confidential containers
- Attestation: ensuring you know what's running

Agenda

Key topics we are going to cover today

- Quick primer on confidential containers
- Attestation: ensuring you know what's running
- Performance overhead: A hardware vendor's dream

Agenda

Key topics we are going to cover today

- Quick primer on confidential containers
- Attestation: ensuring you know what's running
- Performance overhead: A hardware vendor's dream
- Image download: Lather, Rinse, Repeat!

Agenda

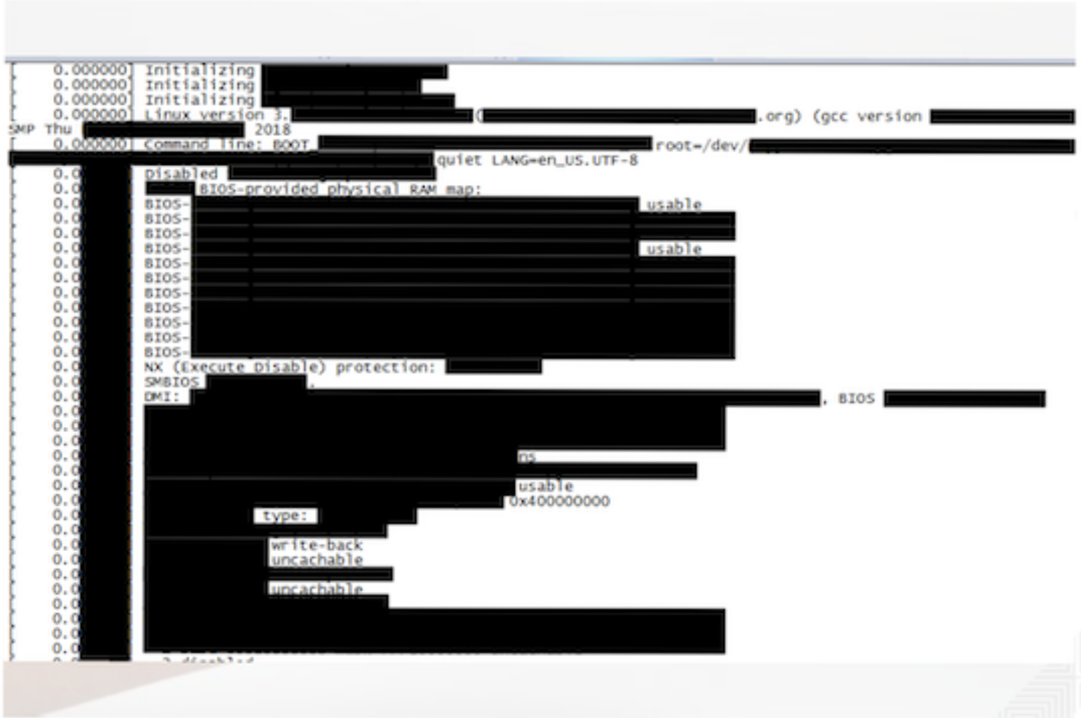
Key topics we are going to cover today

- Quick primer on confidential containers
- Attestation: ensuring you know what's running
- Performance overhead: A hardware vendor's dream
- Image download: Lather, Rinse, Repeat!
- Access control: Rethinking Kubernetes credentials

Agenda

Key topics we are going to cover today

- Quick primer on confidential containers
- Attestation: ensuring you know what's running
- Performance overhead: A hardware vendor's dream
- Image download: Lather, Rinse, Repeat!
- Access control: Rethinking Kubernetes credentials
- Debuggability: even printf-debugging is out



Confidential Containers

A quick primer

Confidential Containers

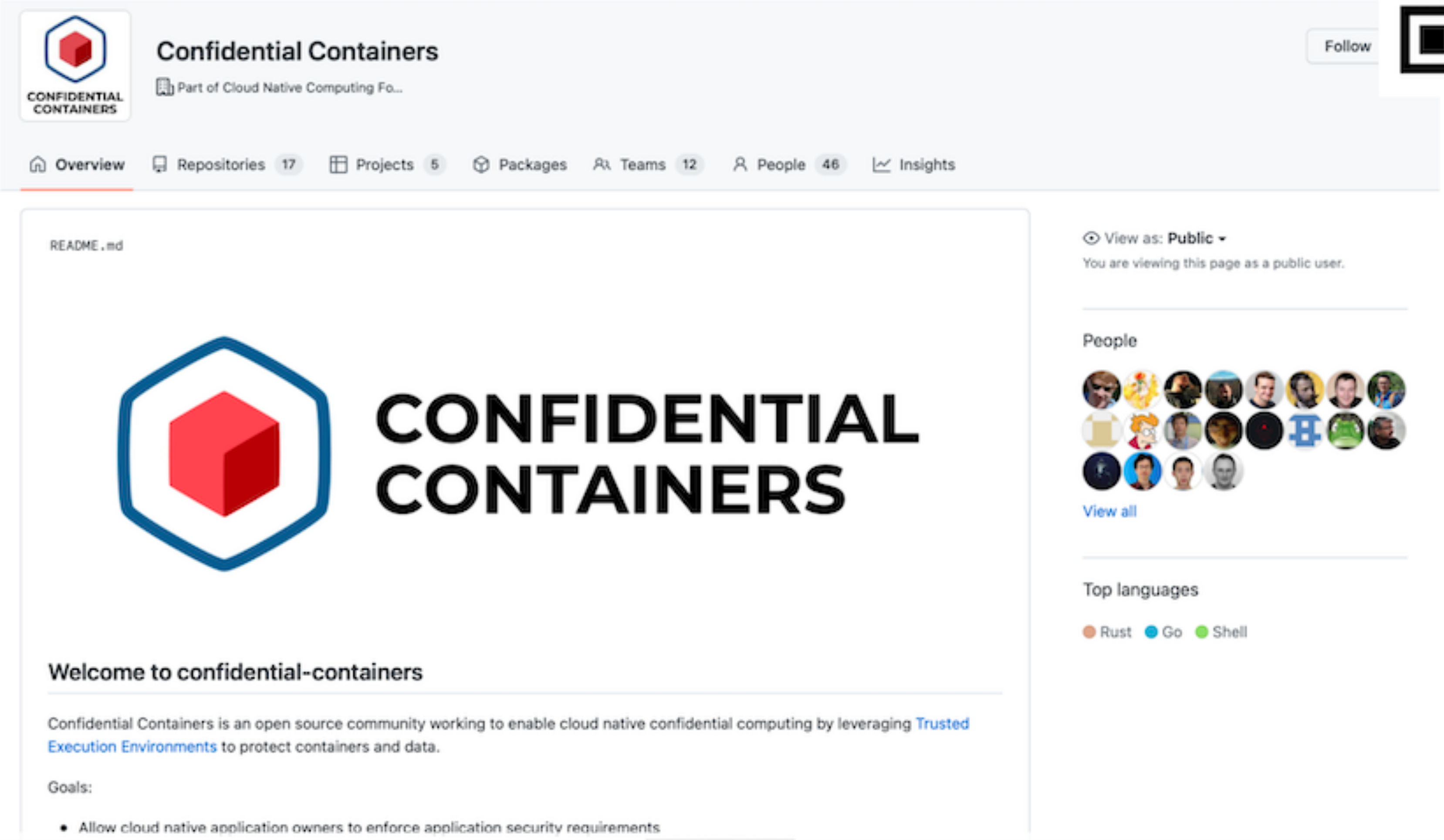
A quick primer

I love cunning containers as much as anyone, but I've found that if I get rid of everything I don't need, I often don't need a container at all.

Gretchen Rubin

A very active project

https://github.com/confidential-containers



Software now runs on hardware you do not own, like a cloud provider

Confidential Computing

Why should infrastructure see your data?

Hardware resources are owned by the host

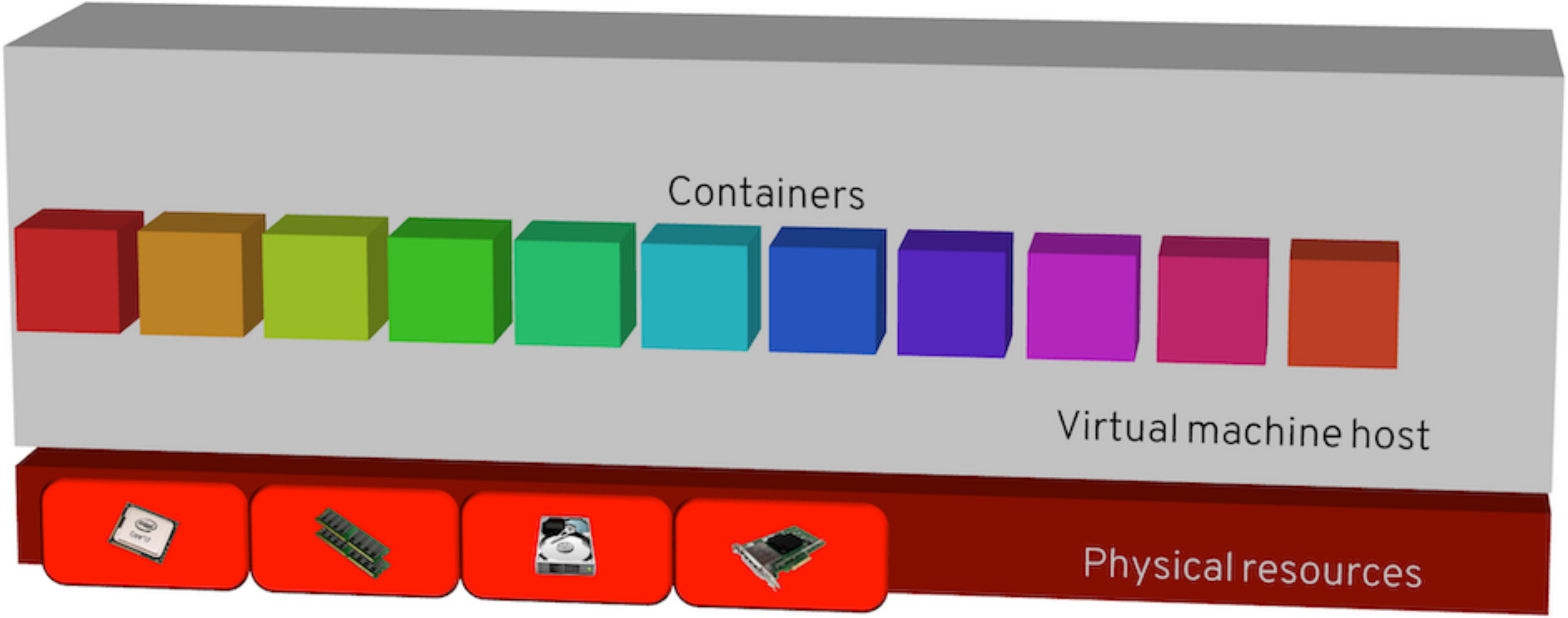
Confidential Computing

Why should infrastructure see your data?

Containers carve out resources from the host

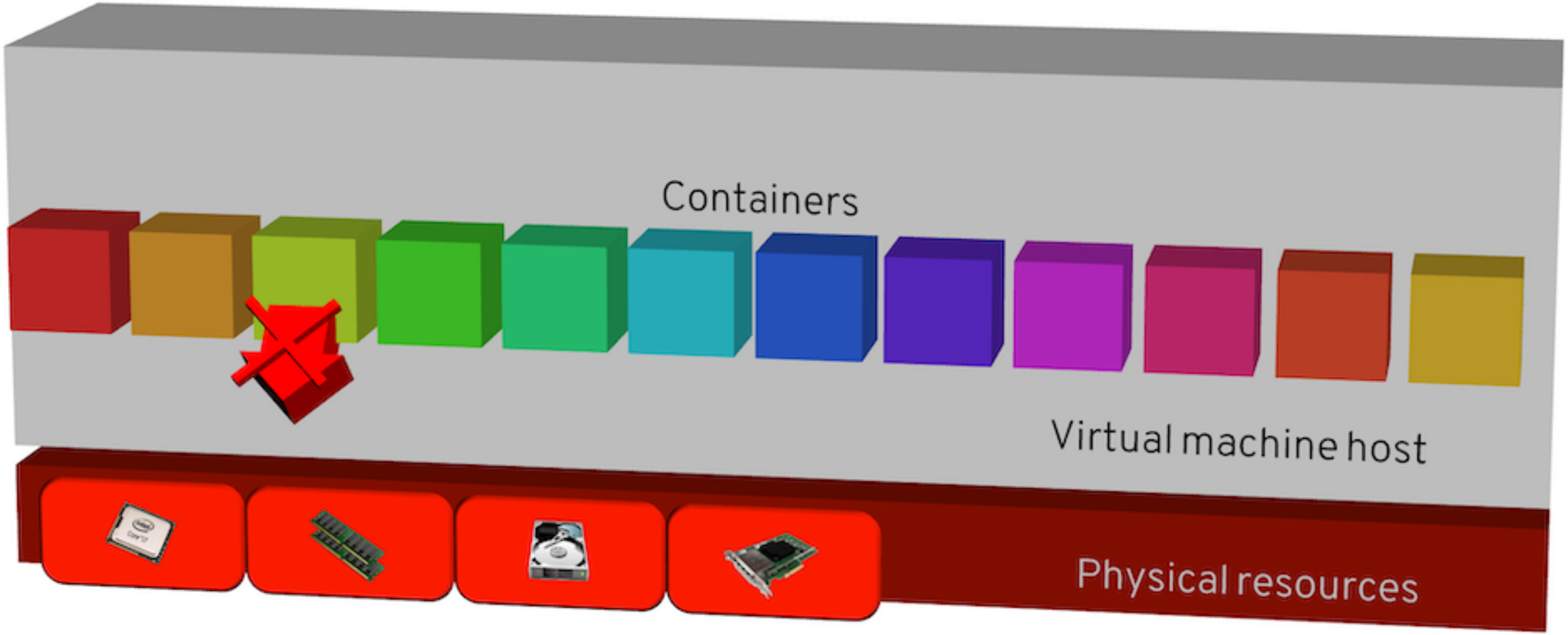
Confidential Computing

Why should infrastructure see your data?



Confidential Computing

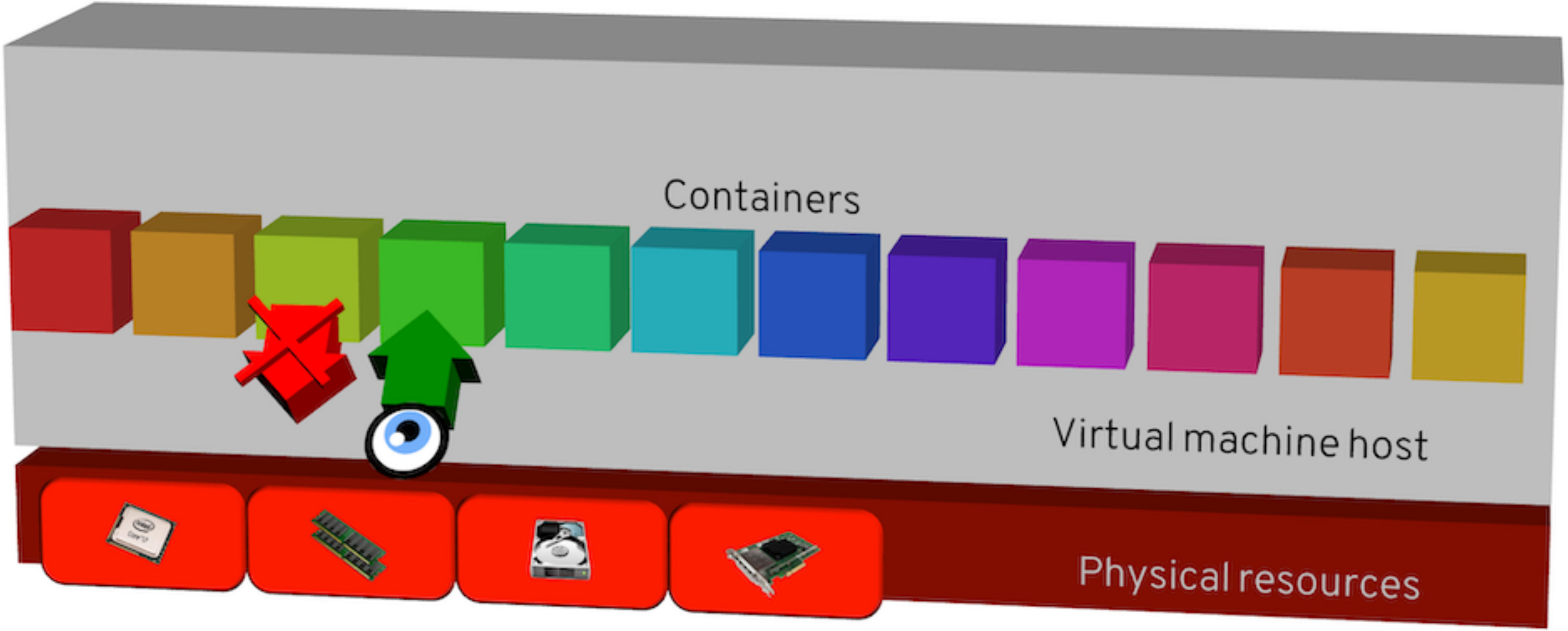
Why should infrastructure see your data?



The host can freely peek insdie the container, for example read its memory

Confidential Computing

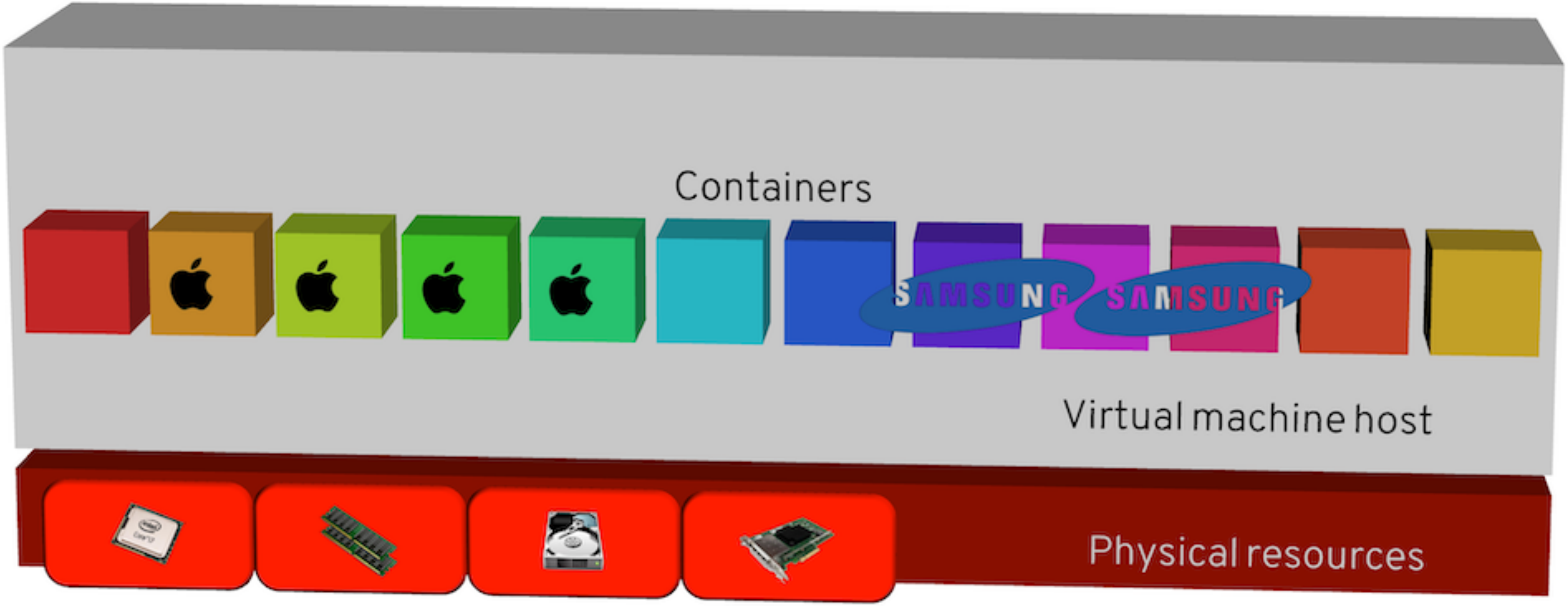
Why should infrastructure see your data?



For that reason, multiple tenants do not want to share hardware when processing sensitive data

Confidential Computing

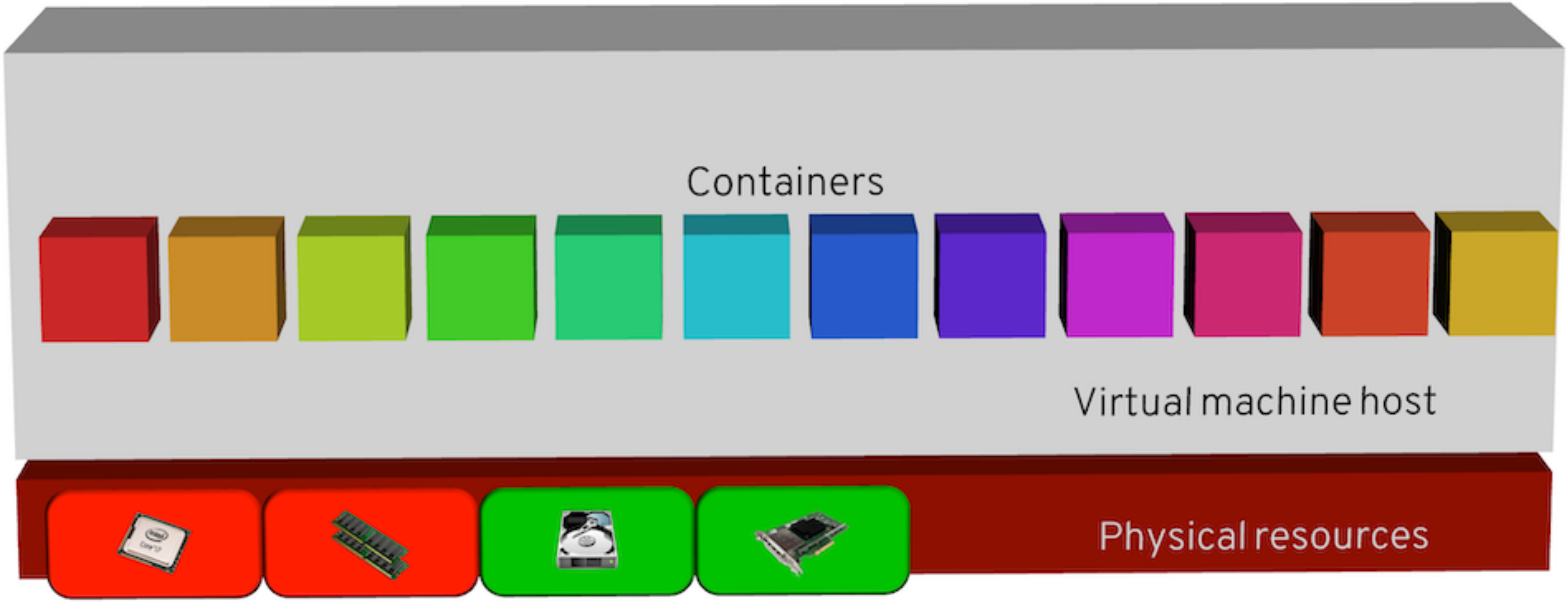
Why should infrastructure see your data?



Data on disk or in network is already encrypted today, so the VM host cannot read it nor tamper with it

Confidential Computing

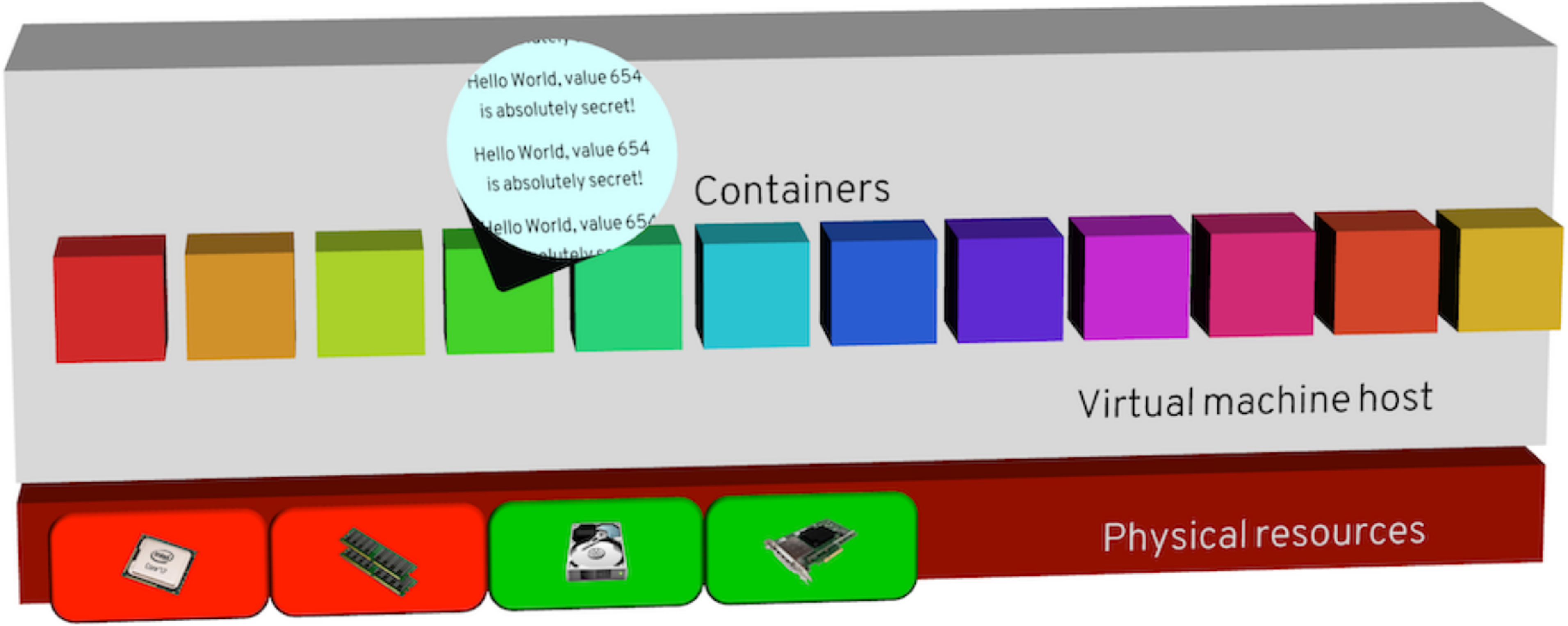
Why should infrastructure see your data?



In non-CC architectures, data in memory is not encrypted, so it can be accessed by the host

Confidential Computing

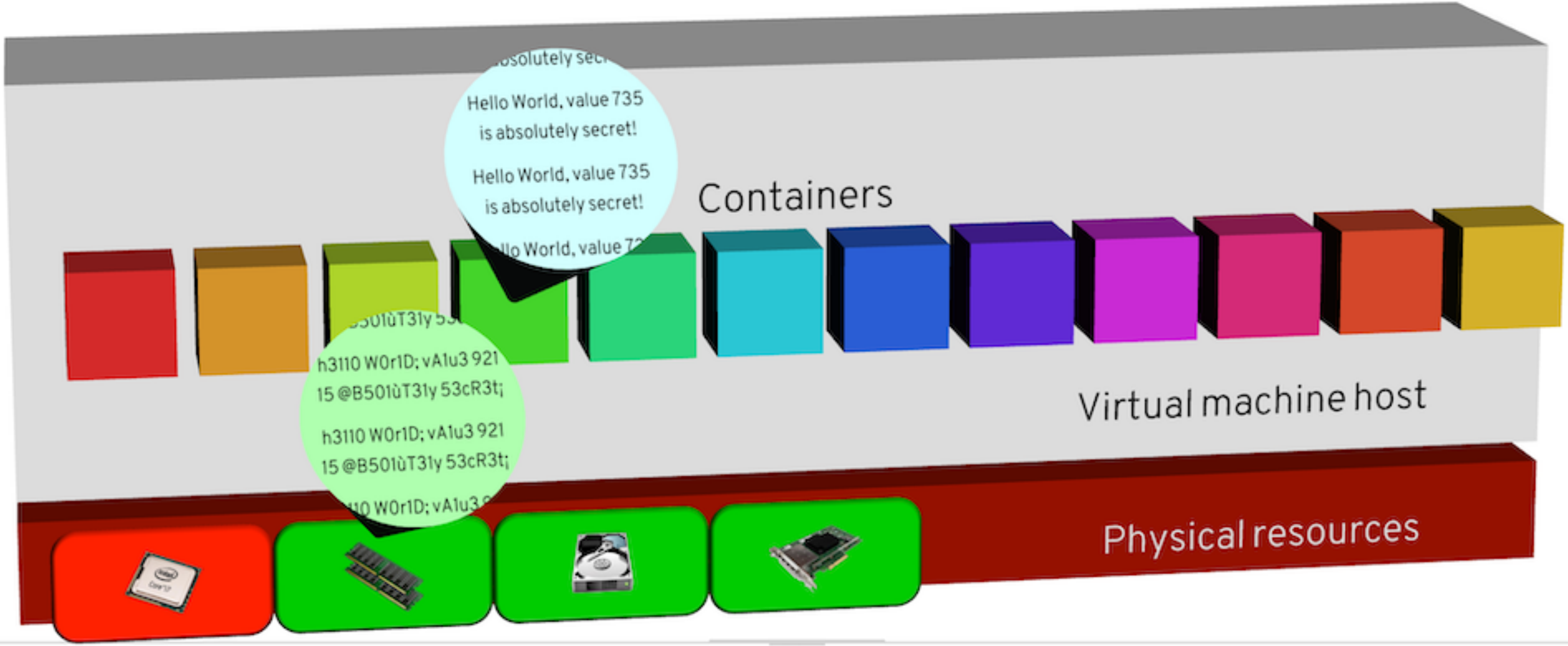
Why should infrastructure see your data?



Encrypted memory stores live data as cypher text, so that it becomes garbled when read by the host

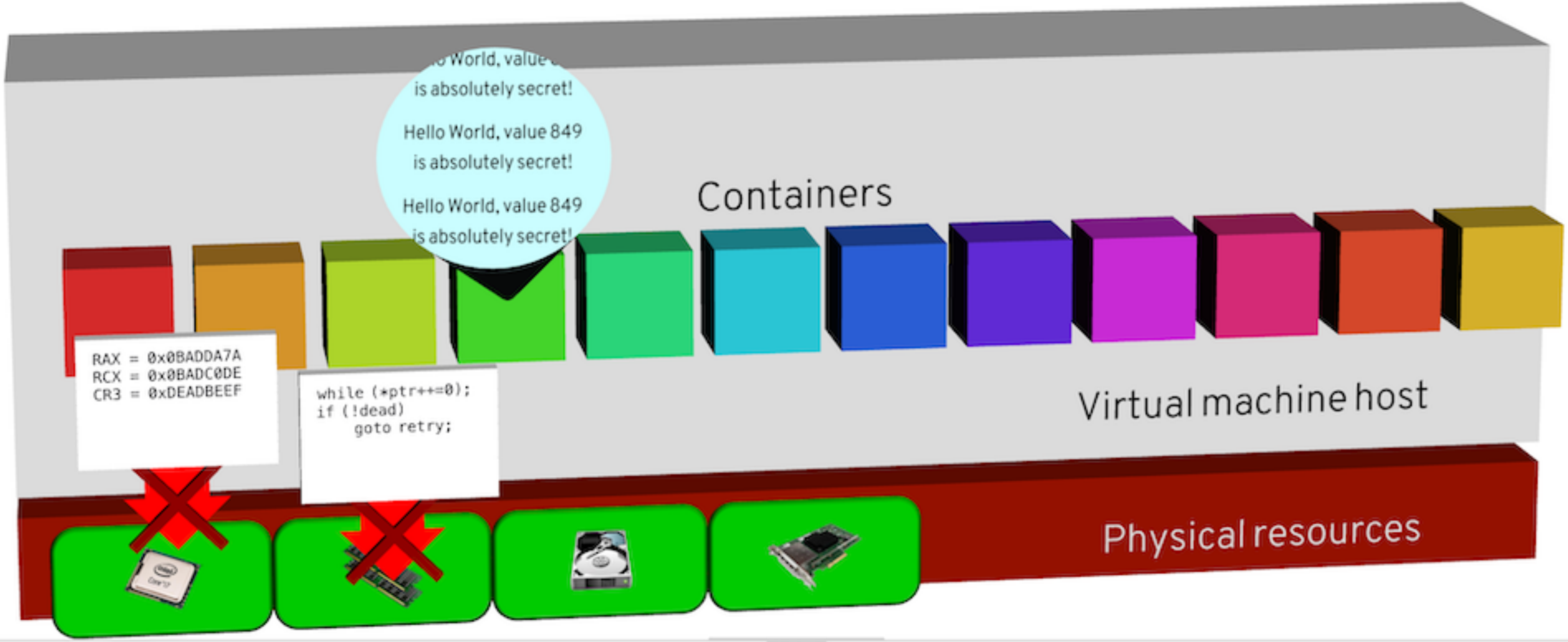
Confidential Computing

Why should infrastructure see your data?



Confidential Computing

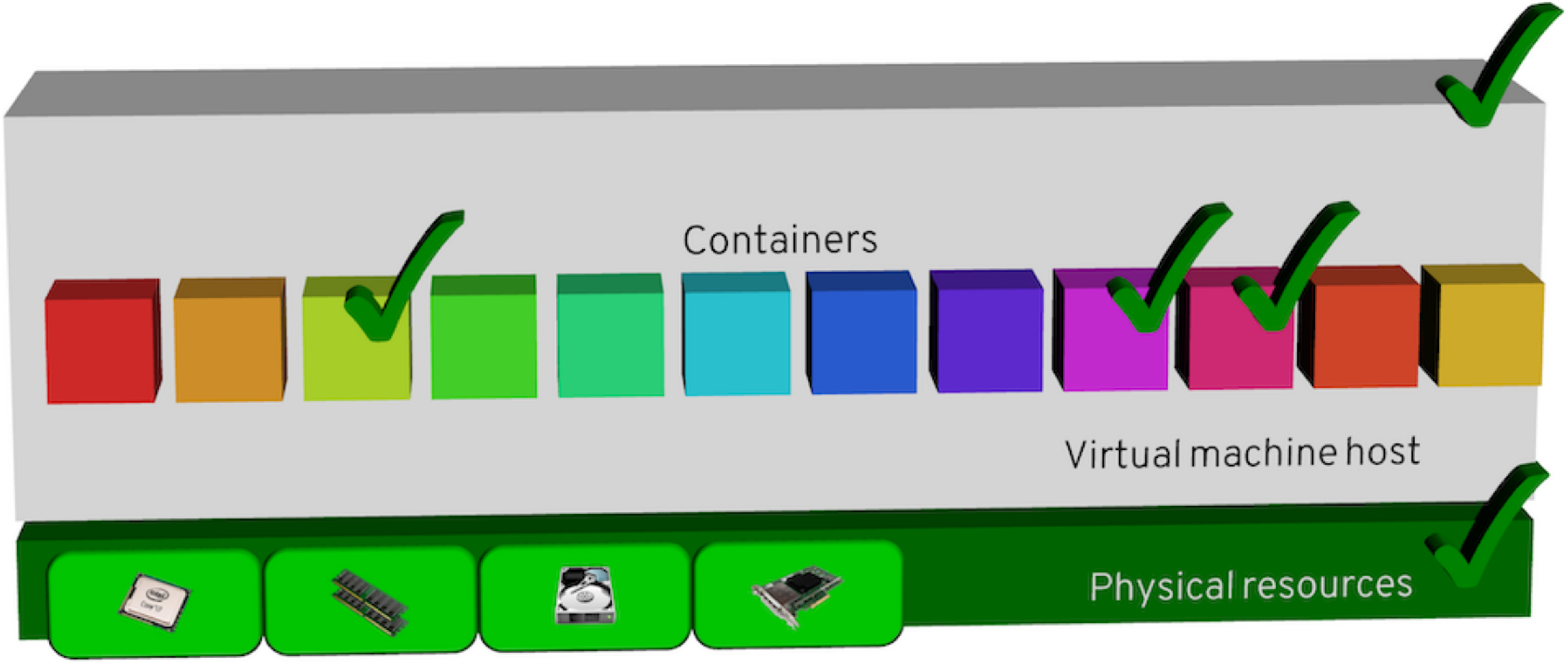
Why should infrastructure see your data?



Attestation proves where you are running and what you are running

Confidential Computing

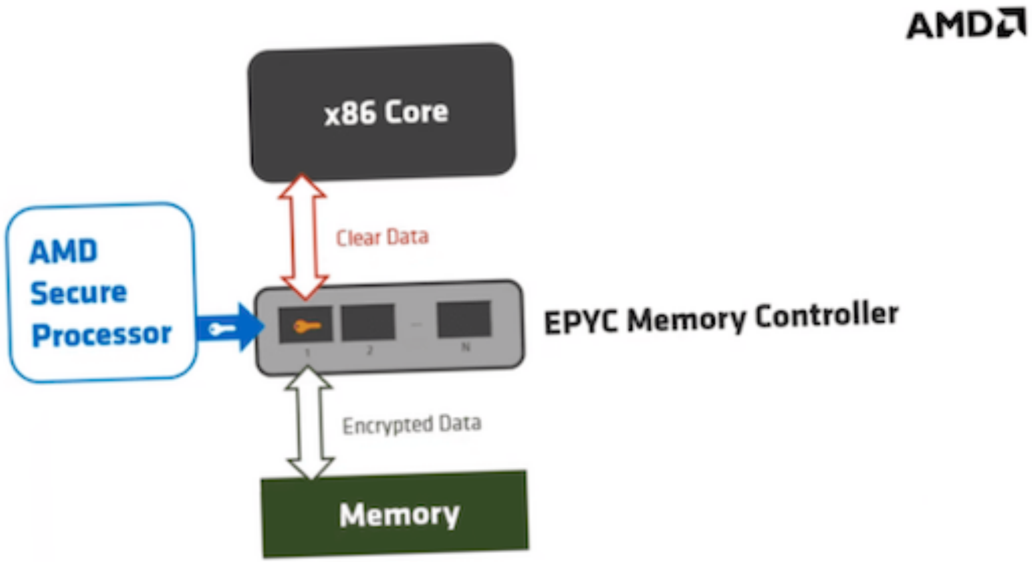
Why should infrastructure see your data?



Vendor landscape

Different vendors with different approaches?

- AMD: Secure Encrypted Virtualization (SEV)



Vendor landscape

Different vendors with different approaches?

- AMD: Secure Encrypted Virtualization (SEV)
  - SEV-ES adds Encrypted State (e.g. CPU register file)

Vendor landscape

Different vendors with different approaches?

- AMD: Secure Encrypted Virtualization (SEV)
  - SEV-ES adds Encrypted State (e.g. CPU register file)
  - SEV-SNP adds Secure Nested Pages (integrity protection)

✓ = Mitigated ★ = Optionally Mitigated ⊘ = Not Mitigated

Potential Threats	SEV	SEV-ES	SEV-SNP
Confidentiality			
VM Memory *Example attack: Hypervisor reads private VM memory*	✓	✓	✓
VM Register State *Example attack: Read VM register state after VMEXIT*	⊘	✓	✓
DMA Protection *Example attack: Device attempts to read VM memory*	✓	✓	✓
Integrity			
Replay Protection *Example attack: Replace VM memory with an old copy*	⊘	⊘	✓
Data Corruption *Example attack: Replace VM memory with junk data*	⊘	⊘	✓
Memory Aliasing *Example attack: Map two guest pages to same DRAM page*	⊘	⊘	✓
Memory Re-Mapping *Example attack: Switch DRAM page mapped to a guest page*	⊘	⊘	✓
Availability			
Denial of Service on Hypervisor *Example attack: Malicious guest refuses to yield/exit*	✓	✓	✓
Denial of Service on Guest *Example attack: Malicious hypervisor refuses to run guest*	⊘	⊘	⊘
Physical Access Attacks			
Offline DRAM analysis *Example attack: Cold boot*	✓	✓	✓
Active DRAM corruption *Example attack: Manipulate DDR bus while VM is running*	⊘	⊘	⊘
Misc.			
TCB Rollback *Example attack: Revert AMD-SP firmware to old version*	⊘	⊘	✓
Malicious Interrupt/Exception Injection *Example attack: Inject interrupt while RFLAGS.IF=0*	⊘	⊘	★
Indirect Branch Predictor Poisoning *Example attack: Poison BTB from hypervisor*	⊘	⊘	★
Secure Hardware Debug Registers *Example attack: Change breakpoints during debug*	⊘	⊘	★
Trusted CPUID Information *Example attack: Hypervisors lies about platform capabilities*	⊘	⊘	★
Architectural Side Channels *Example attack: PRIME+PROBE to track VM accesses*	⊘	⊘	⊘
Page-level Side Channels *Example attack: Track VM access patterns through page tables*	⊘	⊘	⊘
Performance Counter Tracking *Example attack: Fingerprint VM apps by performance data*	⊘	⊘	⊘

Vendor landscape

Different vendors with different approaches?

- AMD: Secure Encrypted Virtualization (SEV)
  - SEV-ES adds Encrypted State (e.g. CPU register file)
  - SEV-SNP adds Secure Nested Pages (integrity protection)
- Intel: Trusted Domain Extensions (TDX)

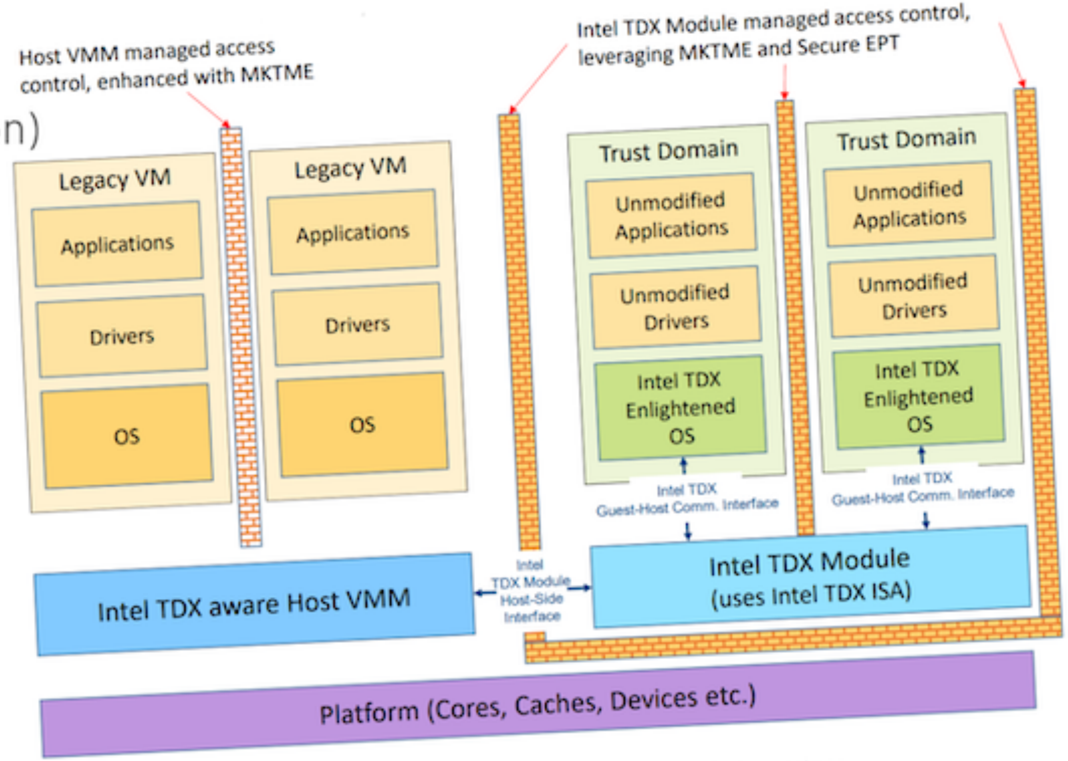
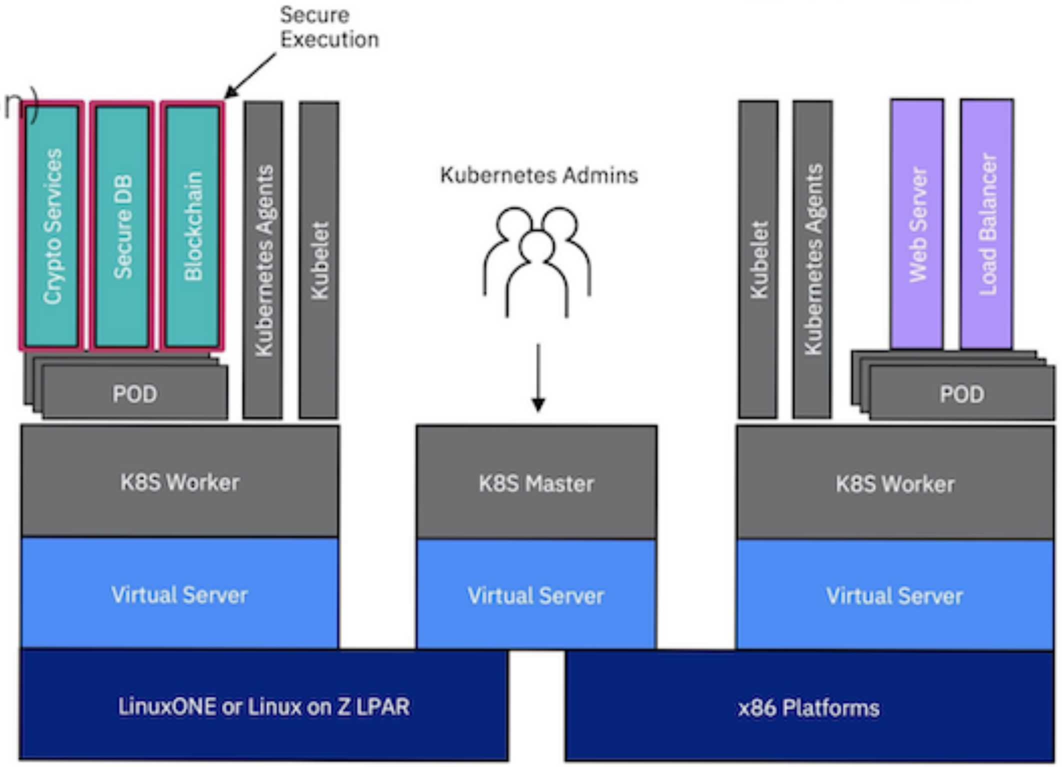


Figure 2.1: Components of Intel Trust Domain Extensions

Vendor landscape

Different vendors with different approaches?

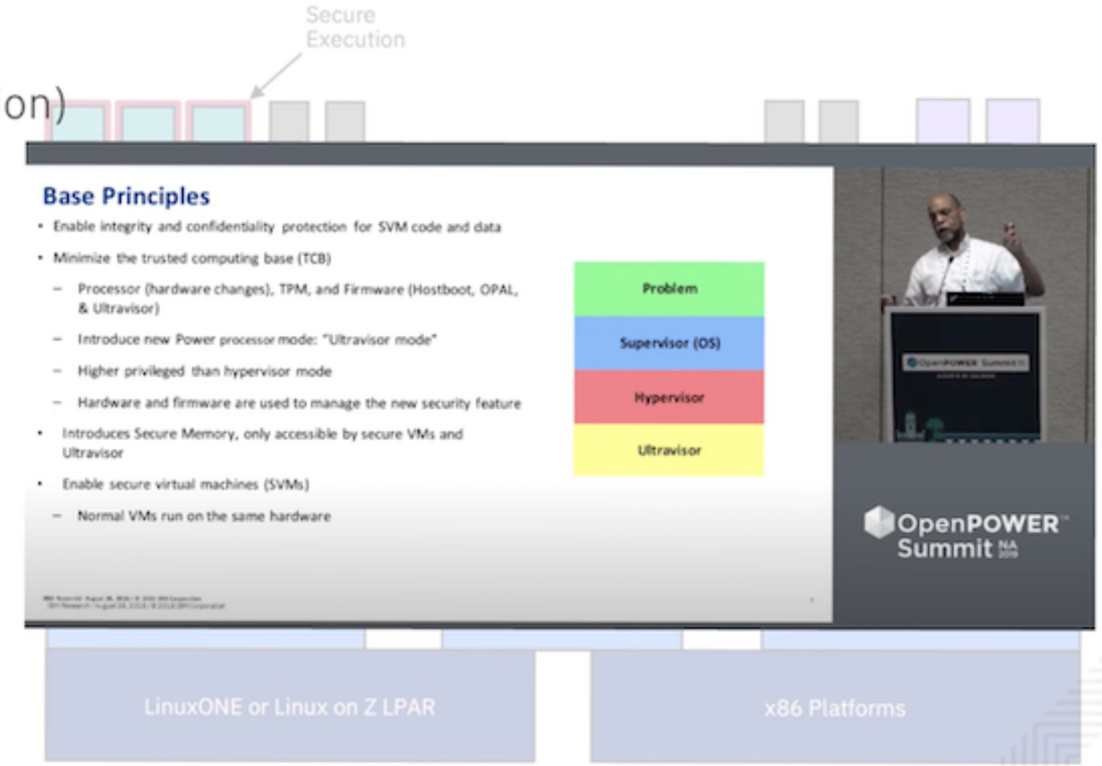
- AMD: Secure Encrypted Virtualization (SEV)
  - SEV-ES adds Encrypted State (e.g. CPU register file)
  - SEV-SNP adds Secure Nested Pages (integrity protection)
- Intel: Trusted Domain Extensions (TDX)
- IBM S390: Secure Execution (SE)



Vendor landscape

Different vendors with different approaches?

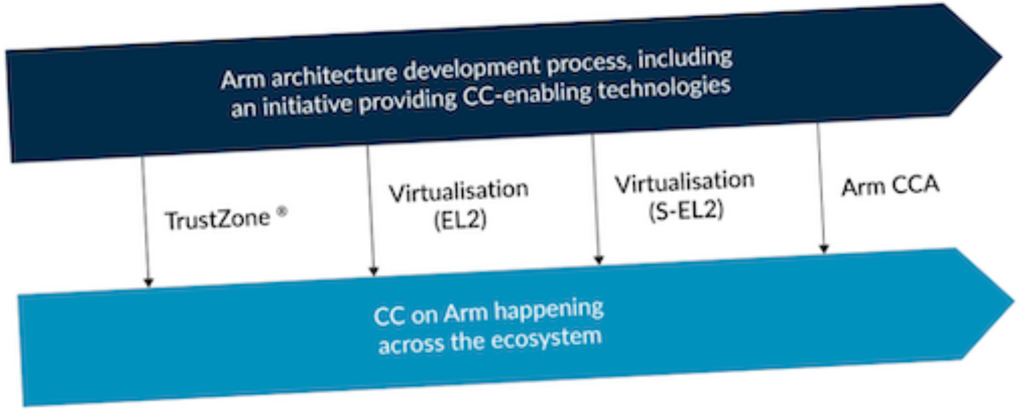
- AMD: Secure Encrypted Virtualization (SEV)
  - SEV-ES adds Encrypted State (e.g. CPU register file)
  - SEV-SNP adds Secure Nested Pages (integrity protection)
- Intel: Trusted Domain Extensions (TDX)
- IBM S390: Secure Execution (SE)
- Power: Protected Execution Facility (PEF)



Vendor landscape

Different vendors with different approaches?

- AMD: Secure Encrypted Virtualization (SEV)
  - SEV-ES adds Encrypted State (e.g. CPU register file)
  - SEV-SNP adds Secure Nested Pages (integrity protection)
- Intel: Trusted Domain Extensions (TDX)
- IBM S390: Secure Execution (SE)
- Power: Protected Execution Facility (PEF)
- Arm: Confidential Computing Architecture (CCA)



Vendor landscape

Different vendors with different approaches?

- AMD: Secure Encrypted Virtualization (SEV)
  - SEV-ES adds Encrypted State (e.g. CPU register file)
  - SEV-SNP adds Secure Nested Pages (integrity protection)
- Intel: Trusted Domain Extensions (TDX)
- IBM S390: Secure Execution (SE)
- Power: Protected Execution Facility (PEF)
- Arm: Confidential Computing Architecture (CCA)
- All these technologies are based on virtualization

Vendor landscape

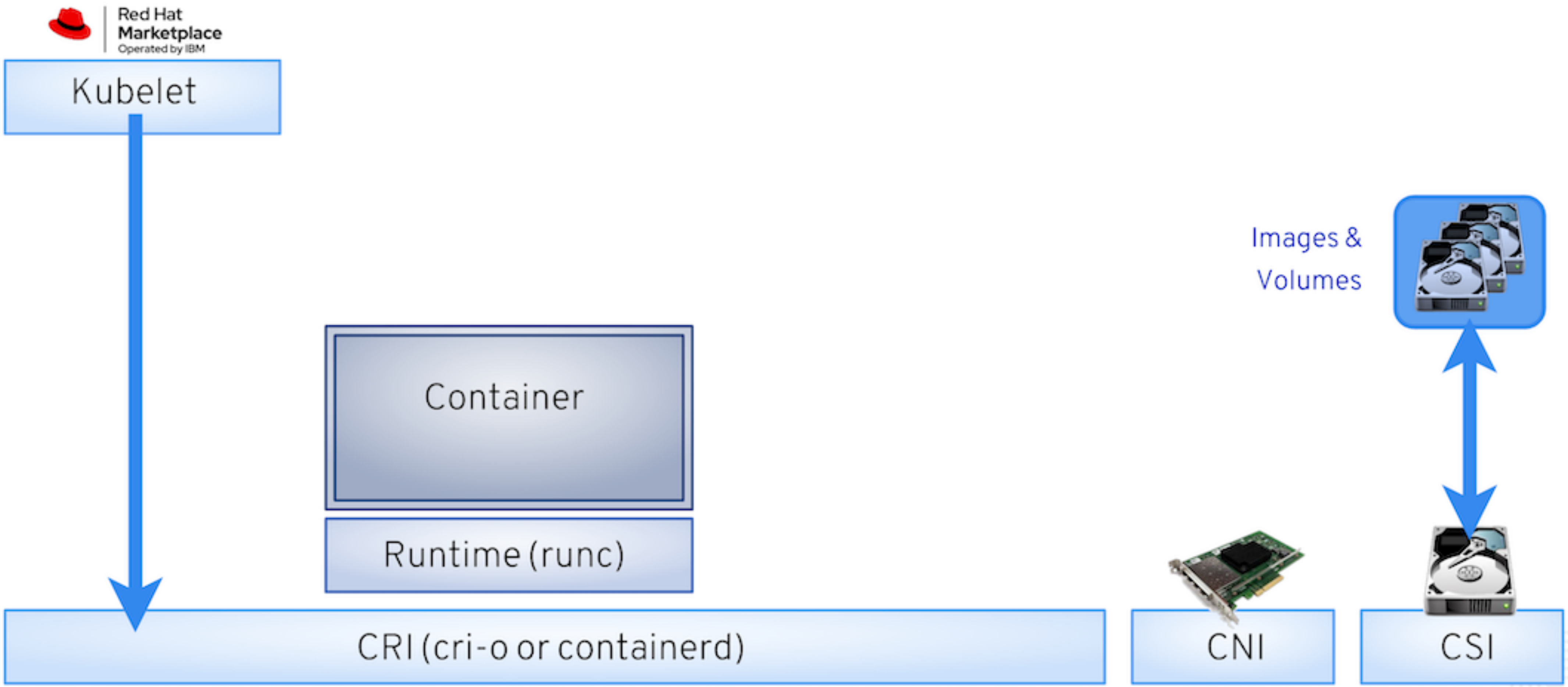
Different vendors with different approaches?

- AMD: Secure Encrypted Virtualization (SEV)
  - SEV-ES adds Encrypted State (e.g. CPU register file)
  - SEV-SNP adds Secure Nested Pages (integrity protection)
- Intel: Trusted Domain Extensions (TDX)
- IBM S390: Secure Execution (SE)
- Power: Protected Execution Facility (PEF)
- Arm: Confidential Computing Architecture (CCA)
- All these technologies are based on virtualization
- They all work differently

Kata Containers Overview

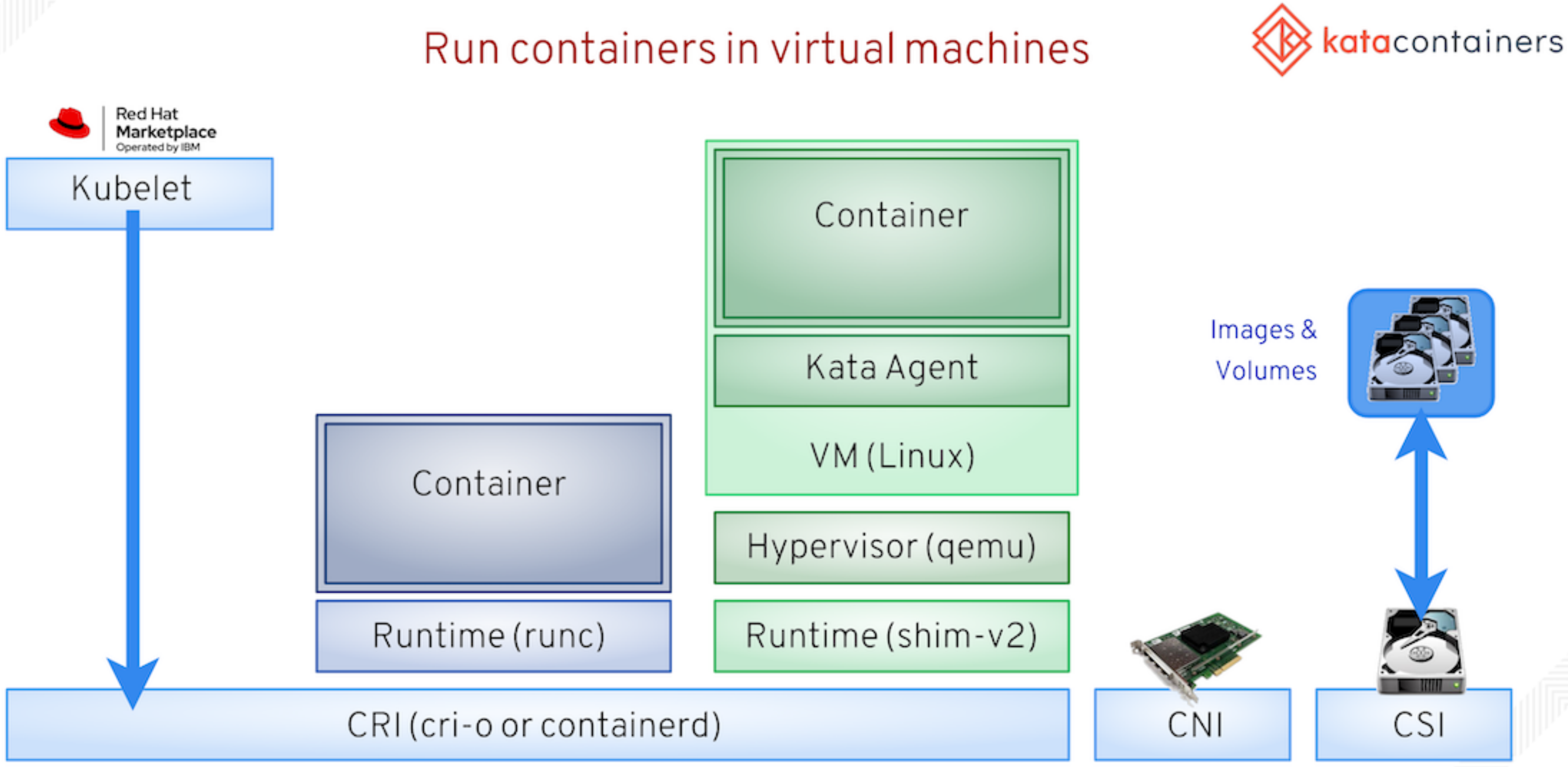
Run containers in virtual machines

katacontainers



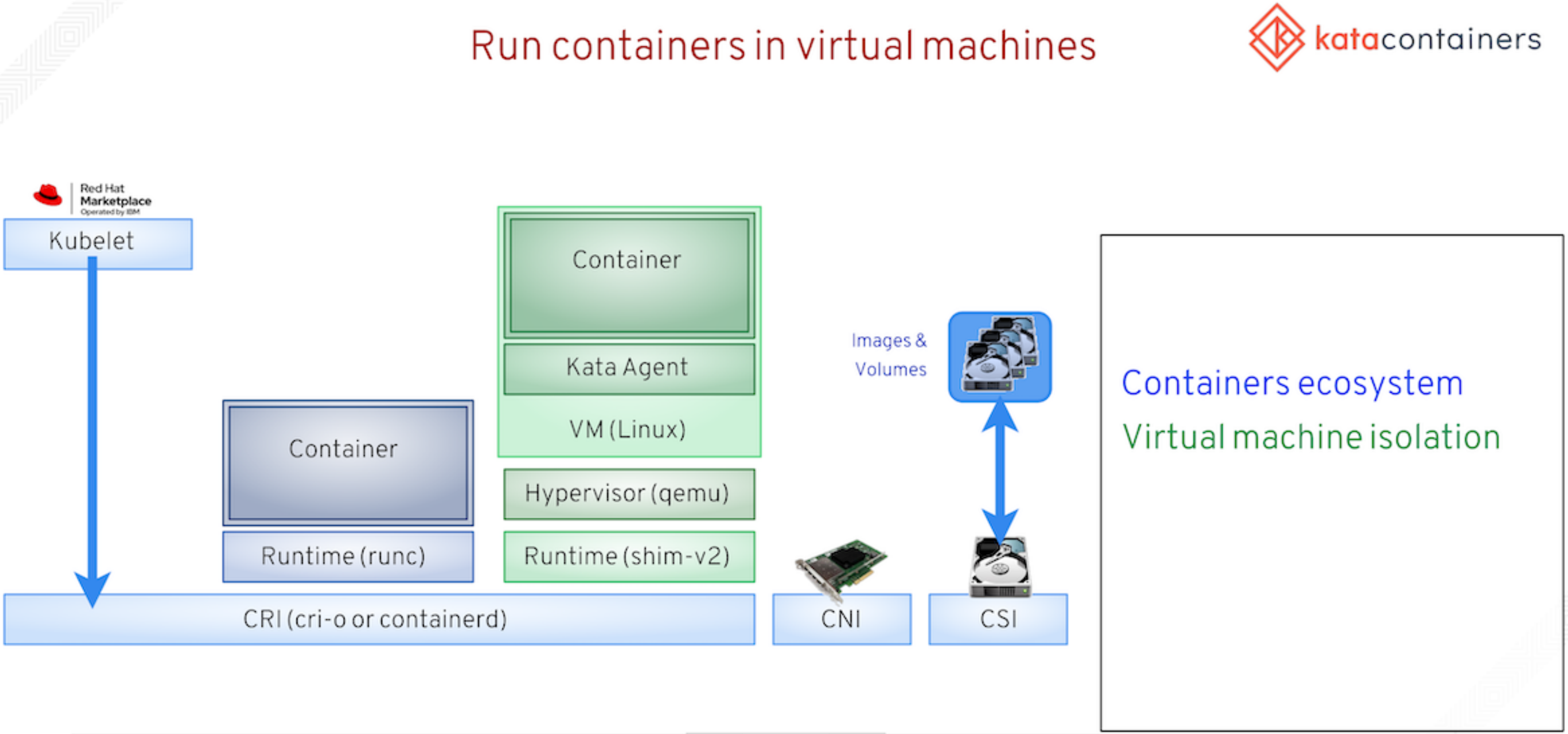
Kata Containers Overview

Run containers in virtual machines



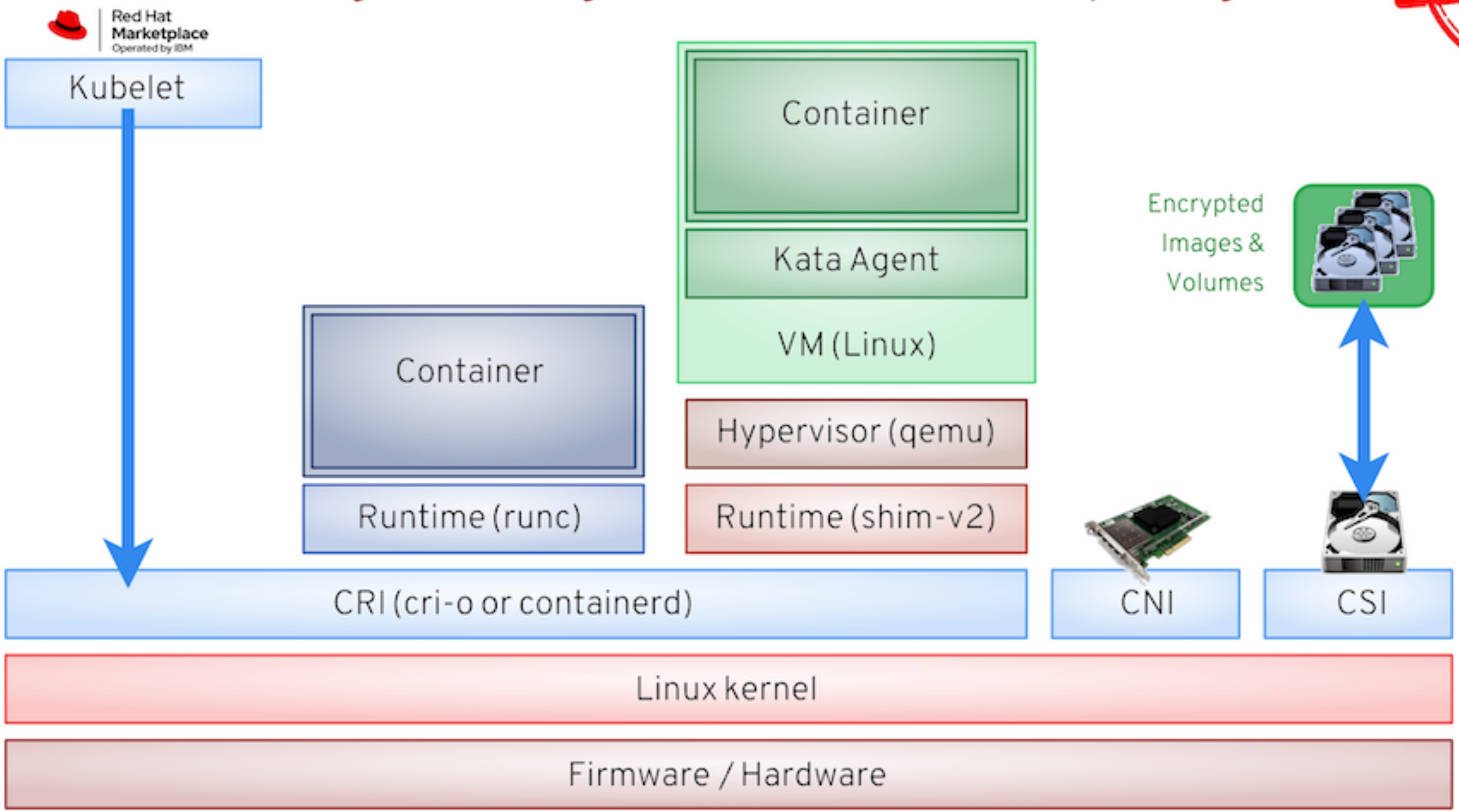
Kata Containers Overview

Run containers in virtual machines



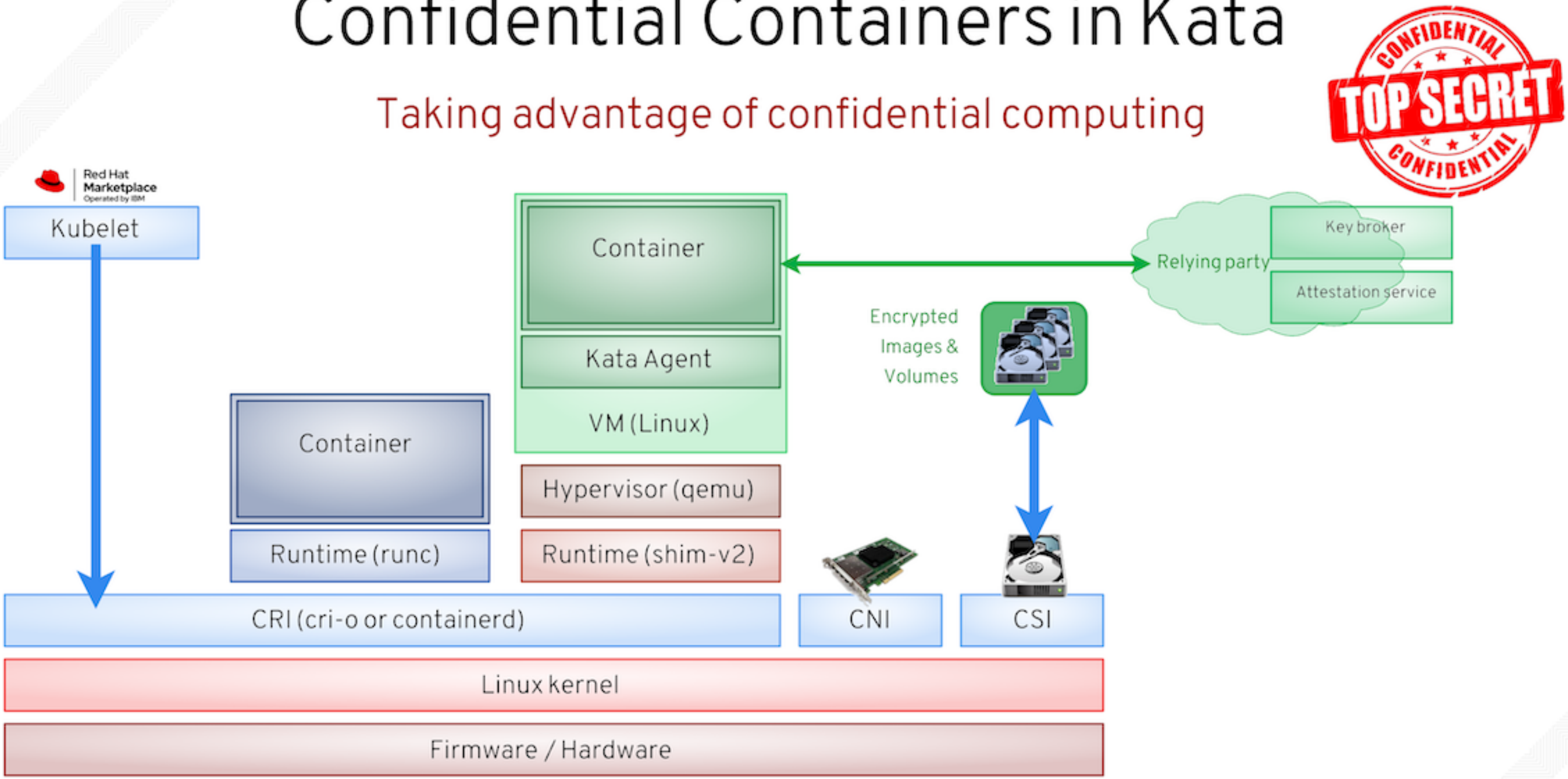
Confidential Containers in Kata

Taking advantage of confidential computing



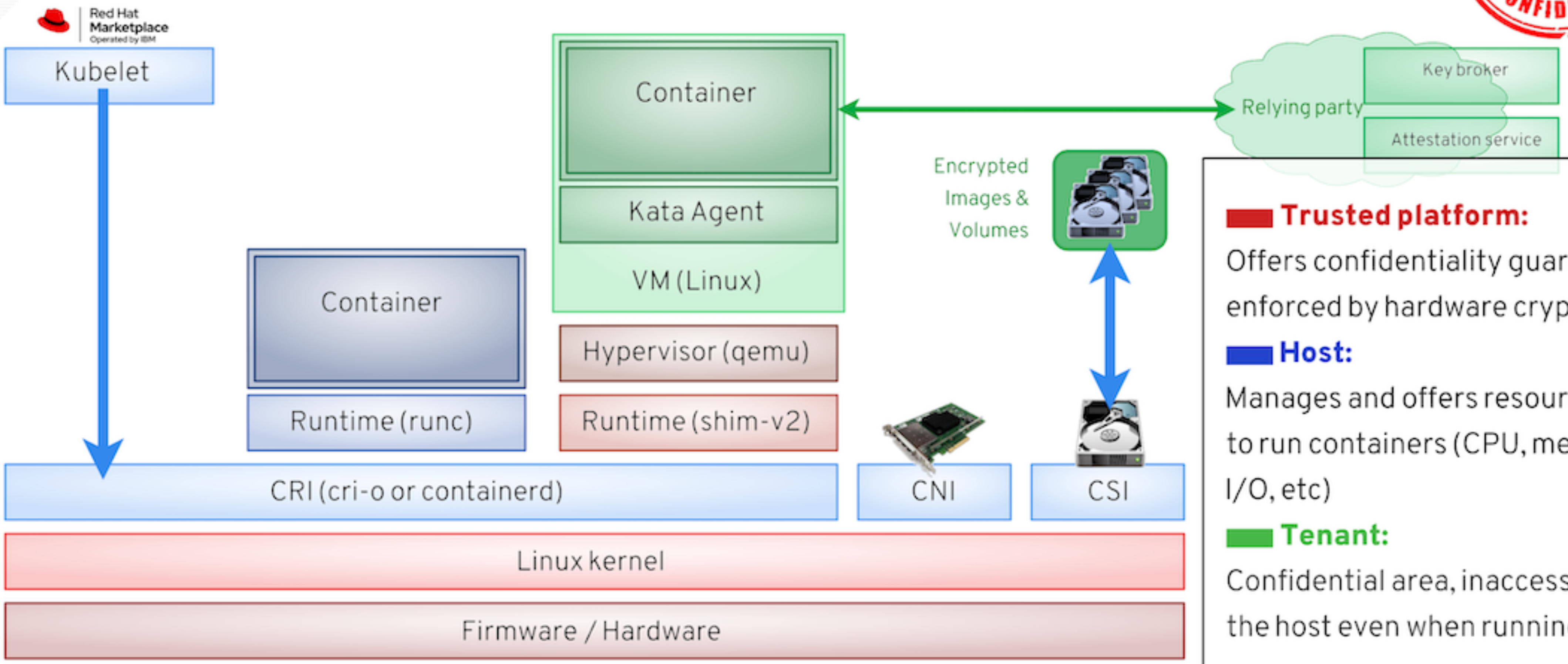
Confidential Containers in Kata

Taking advantage of confidential computing



Confidential Containers in Kata

Taking advantage of confidential computing



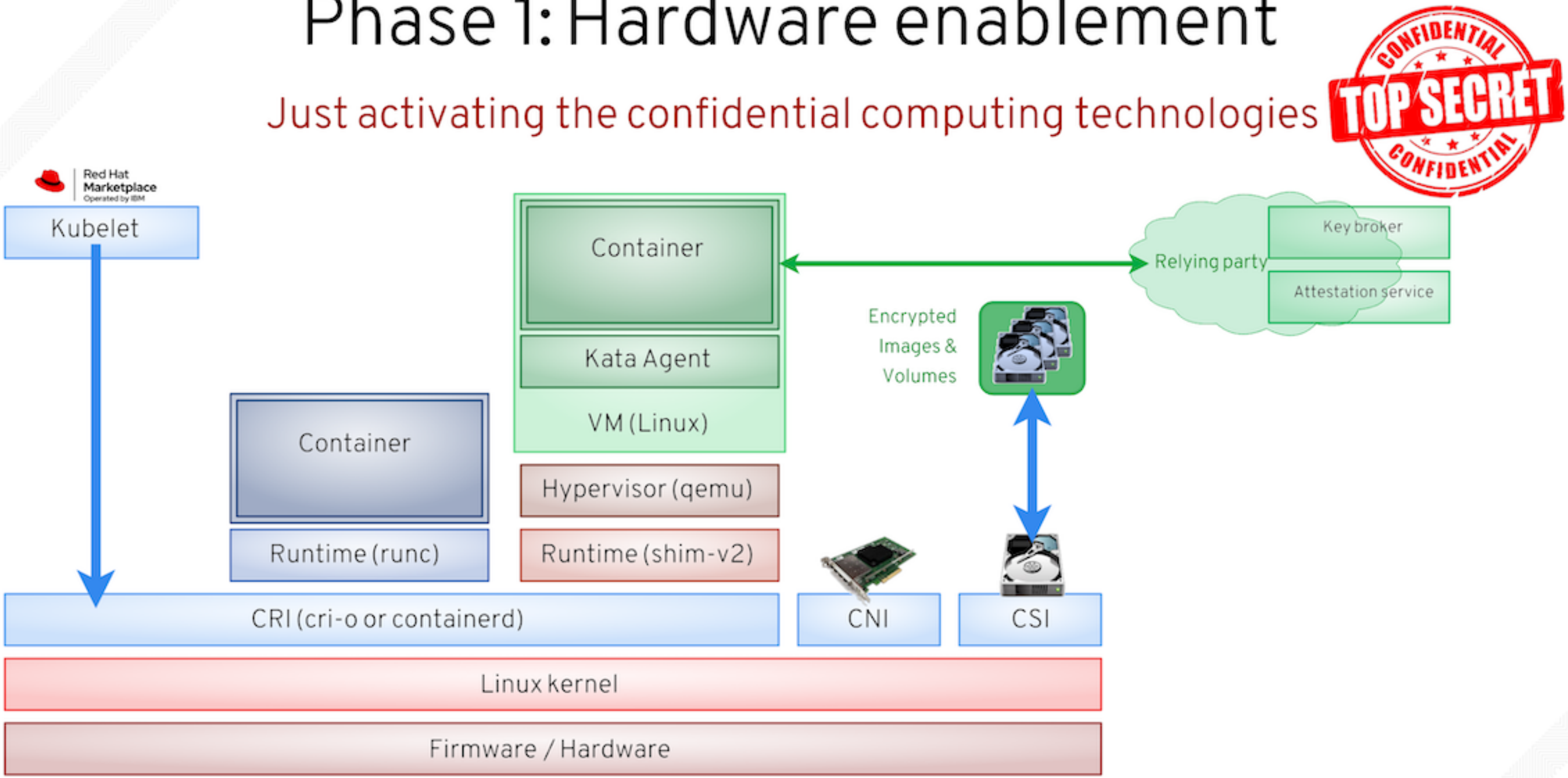
Trusted platform:
Offers confidentiality guarantees enforced by hardware cryptography

Host:
Manages and offers resources used to run containers (CPU, memory, I/O, etc)

Tenant:
Confidential area, inaccessible to the host even when running on it

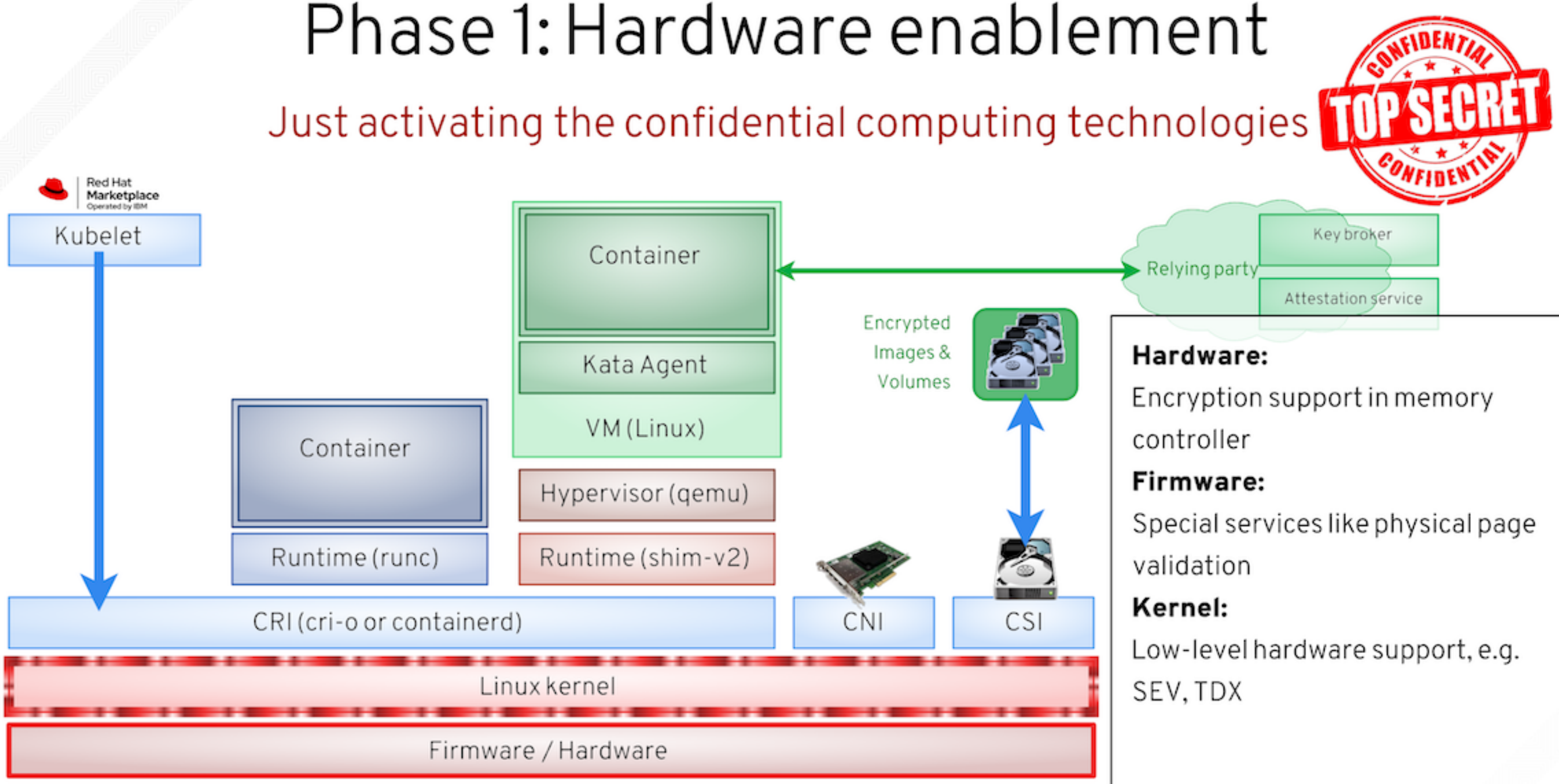
Phase 1: Hardware enablement

Just activating the confidential computing technologies



Phase 1: Hardware enablement

Just activating the confidential computing technologies



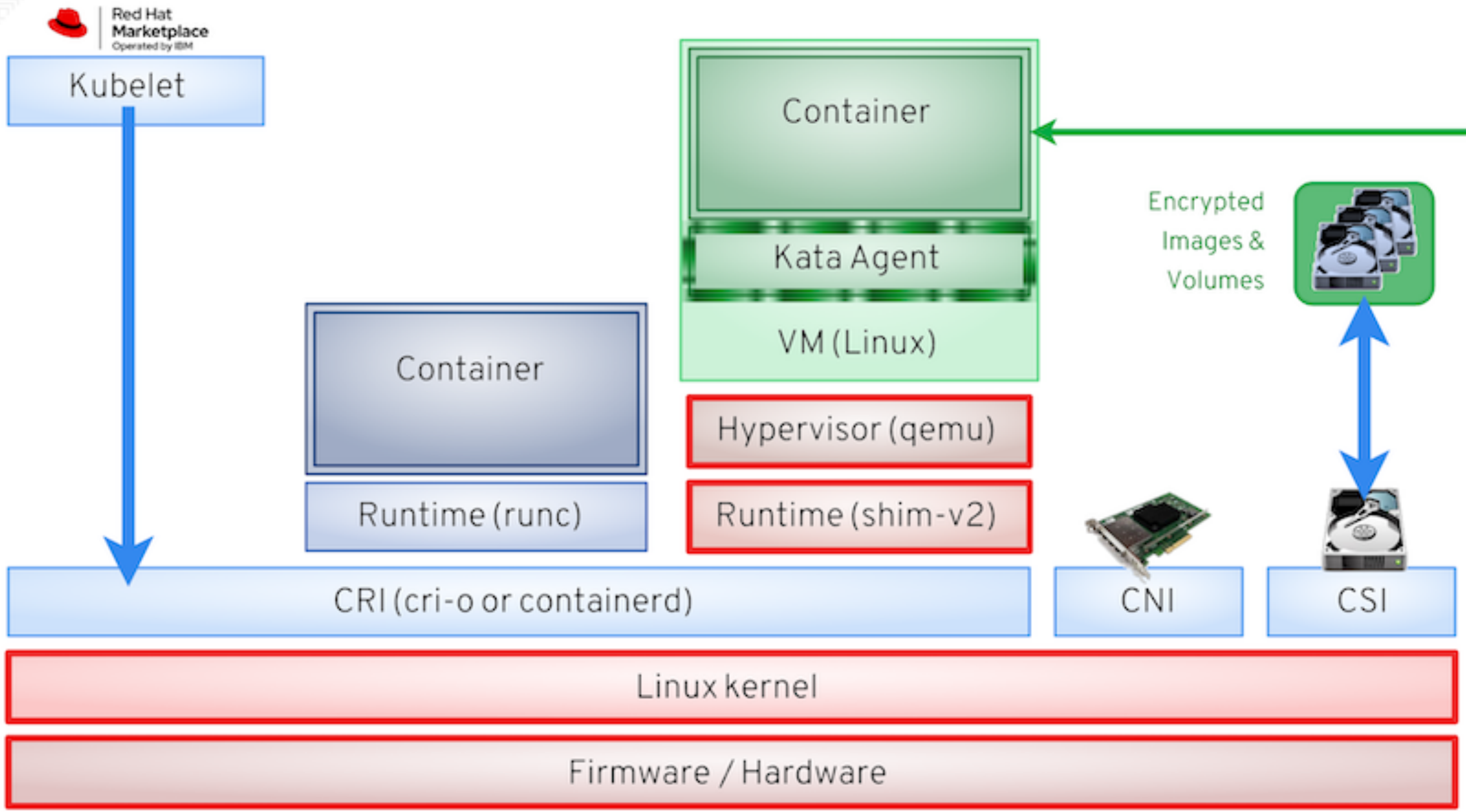
Hardware:
Encryption support in memory controller

Firmware:
Special services like physical page validation

Kernel:
Low-level hardware support, e.g. SEV, TDX

Phase 1: Hardware enablement

Just activating the confidential computing technologies



Hardware:
Encryption support in memory controller

Firmware:
Special services like physical page validation

Kernel:
Low-level hardware support, e.g. SEV, TDX

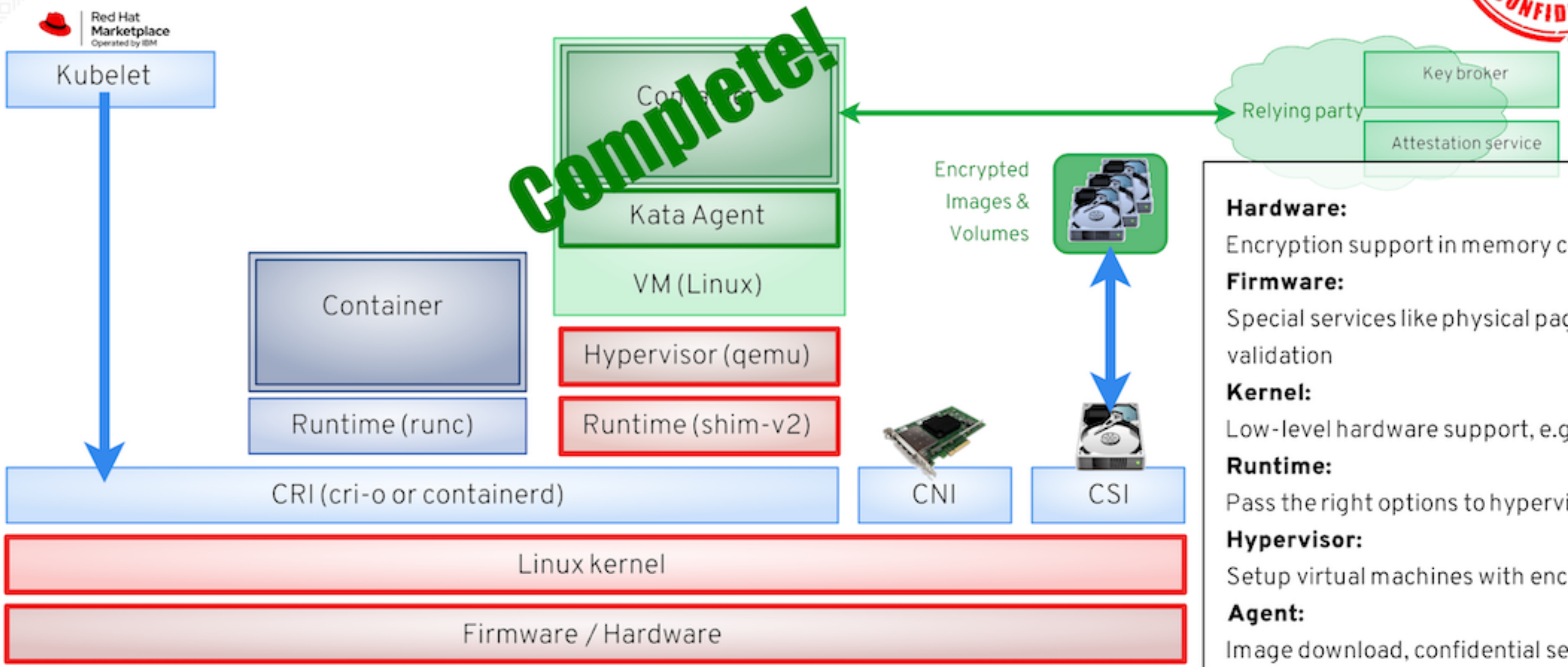
Runtime:
Pass the right options to hypervisor

Hypervisor:
Setup virtual machines with encryption, etc

Agent:
Image download, confidential services

Phase 1: Hardware enablement

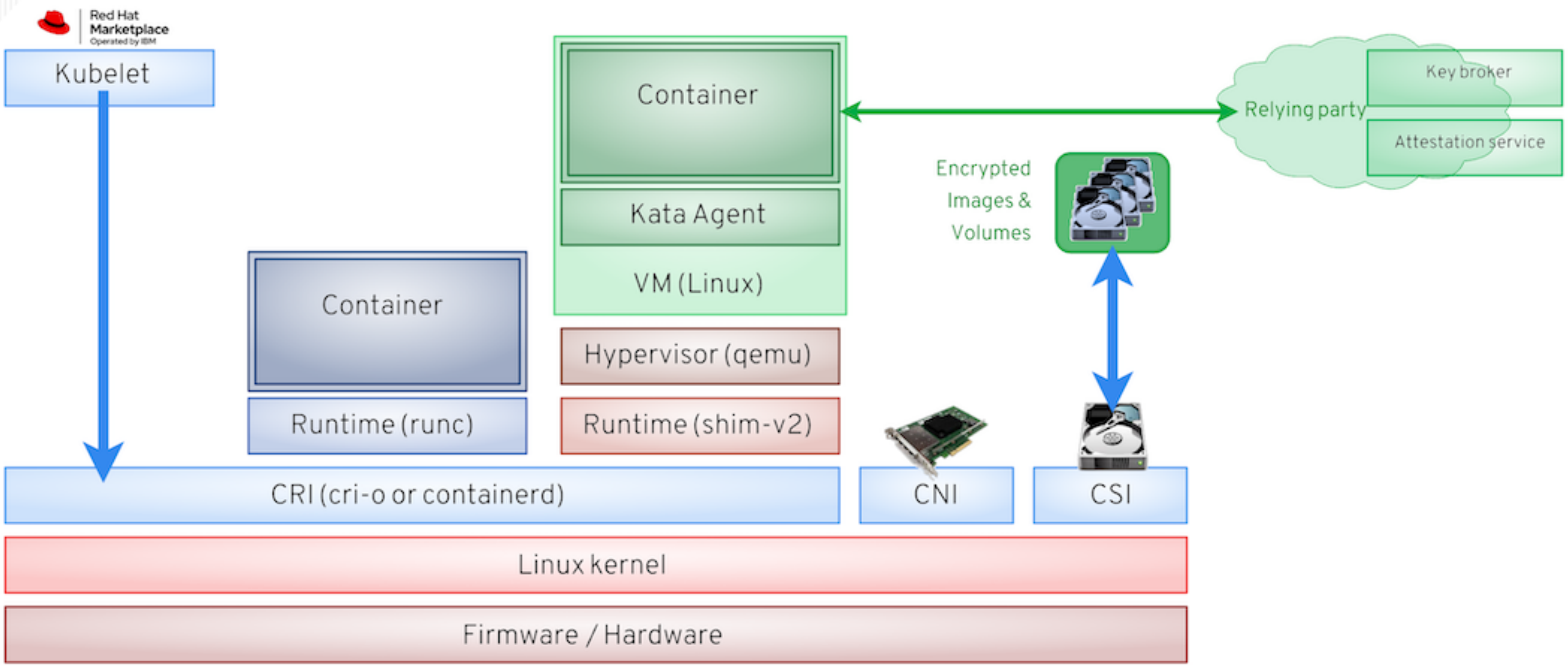
Just activating the confidential computing technologies



Hardware:
Encryption support in memory controller
Firmware:
Special services like physical page validation
Kernel:
Low-level hardware support, e.g. SEV, TDX
Runtime:
Pass the right options to hypervisor
Hypervisor:
Setup virtual machines with encryption, etc
Agent:
Image download, confidential services

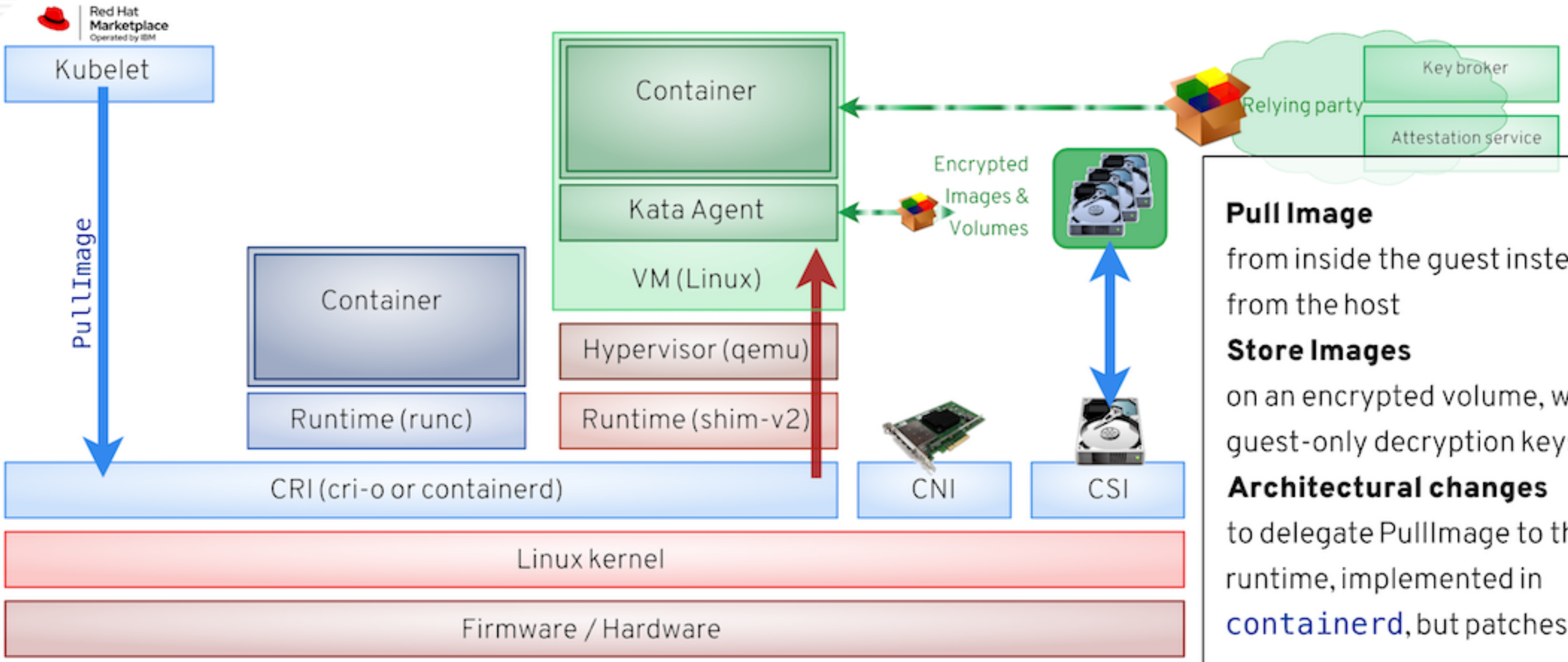
Phase 2: Securing image pull

Download images from within the guest



Phase 2: Securing image pull

Download images from within the guest



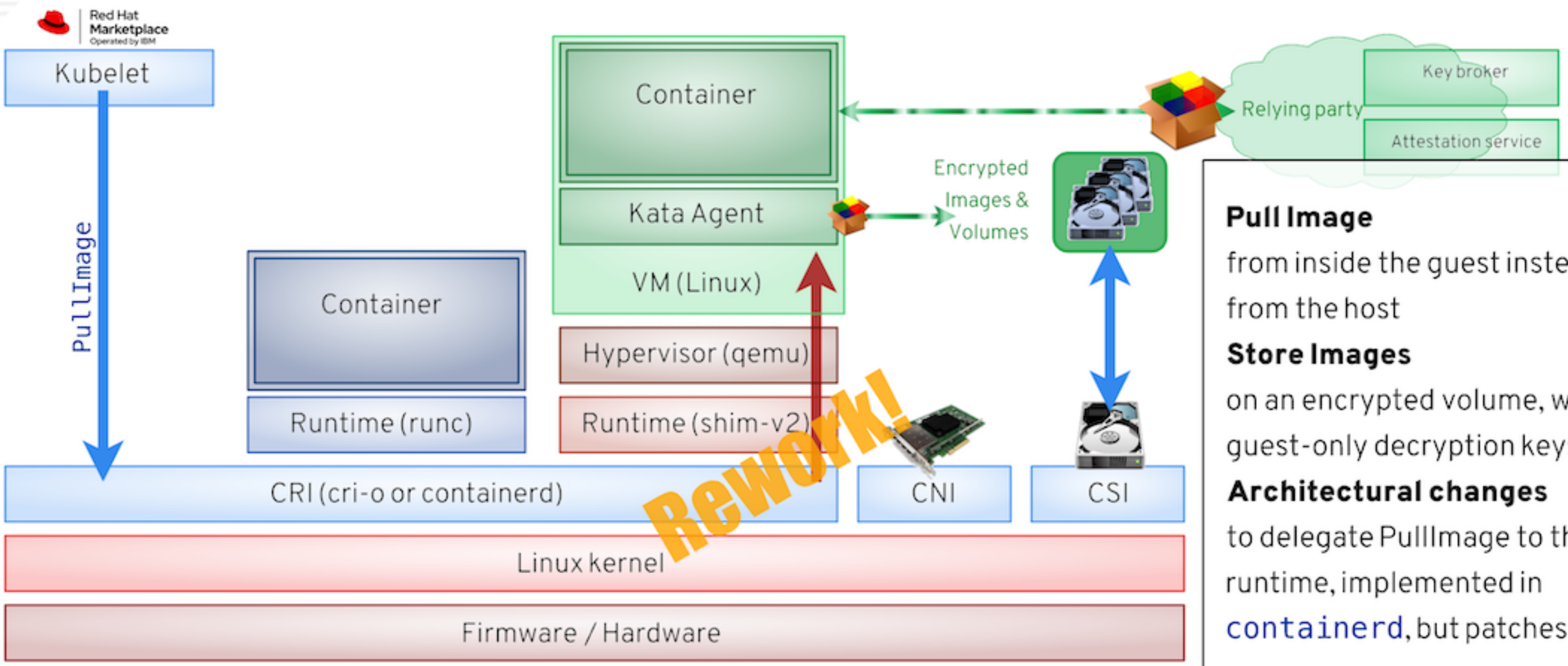
Pull Image
from inside the guest instead of from the host

Store Images
on an encrypted volume, with guest-only decryption keys

Architectural changes
to delegate PullImage to the runtime, implemented in `containerd`, but patches rejected

Phase 2: Securing image pull

Download images from within the guest



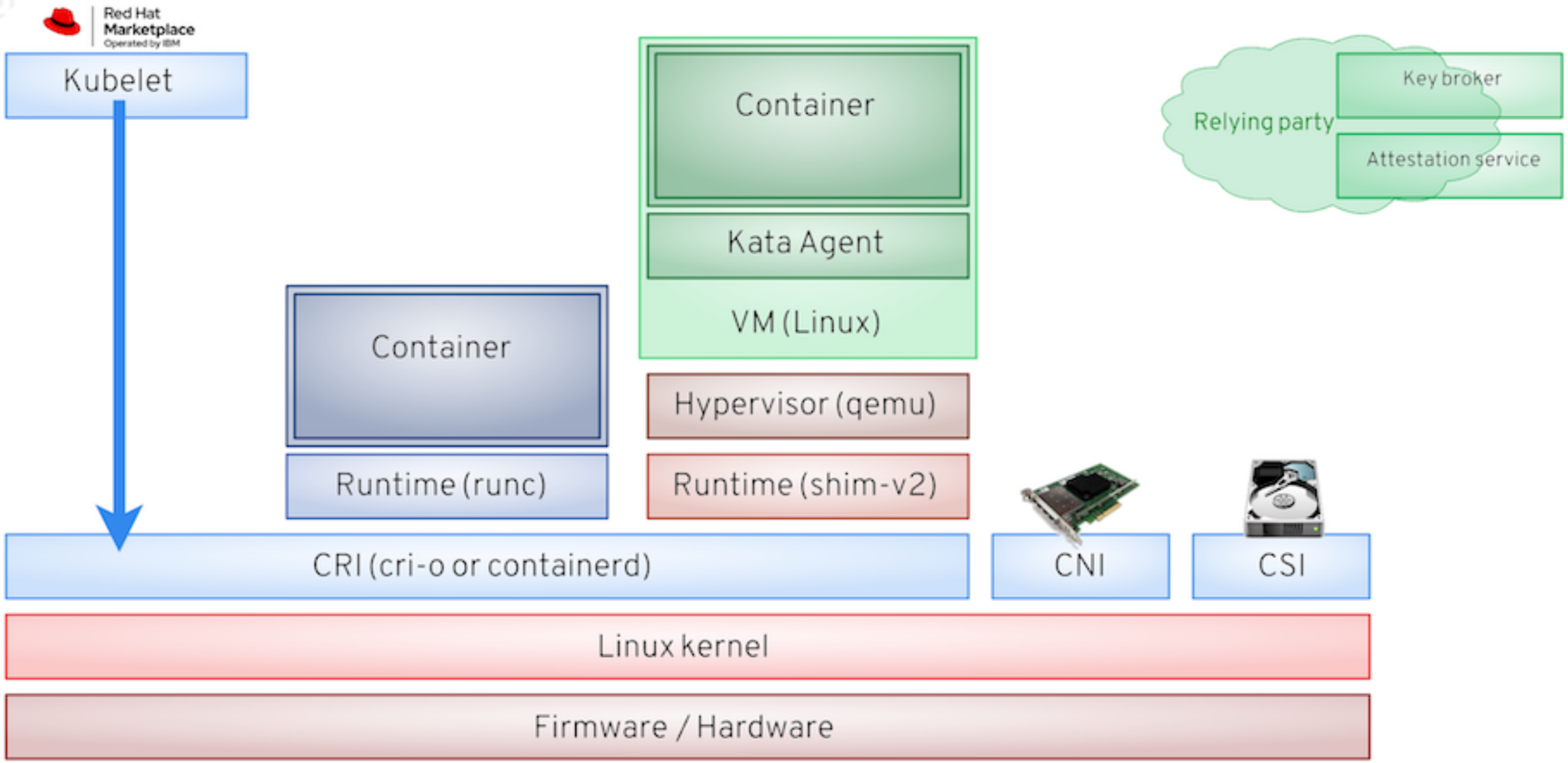
Pull Image
from inside the guest instead of from the host

Store Images
on an encrypted volume, with guest-only decryption keys

Architectural changes
to delegate PullImage to the runtime, implemented in `containerd`, but patches rejected

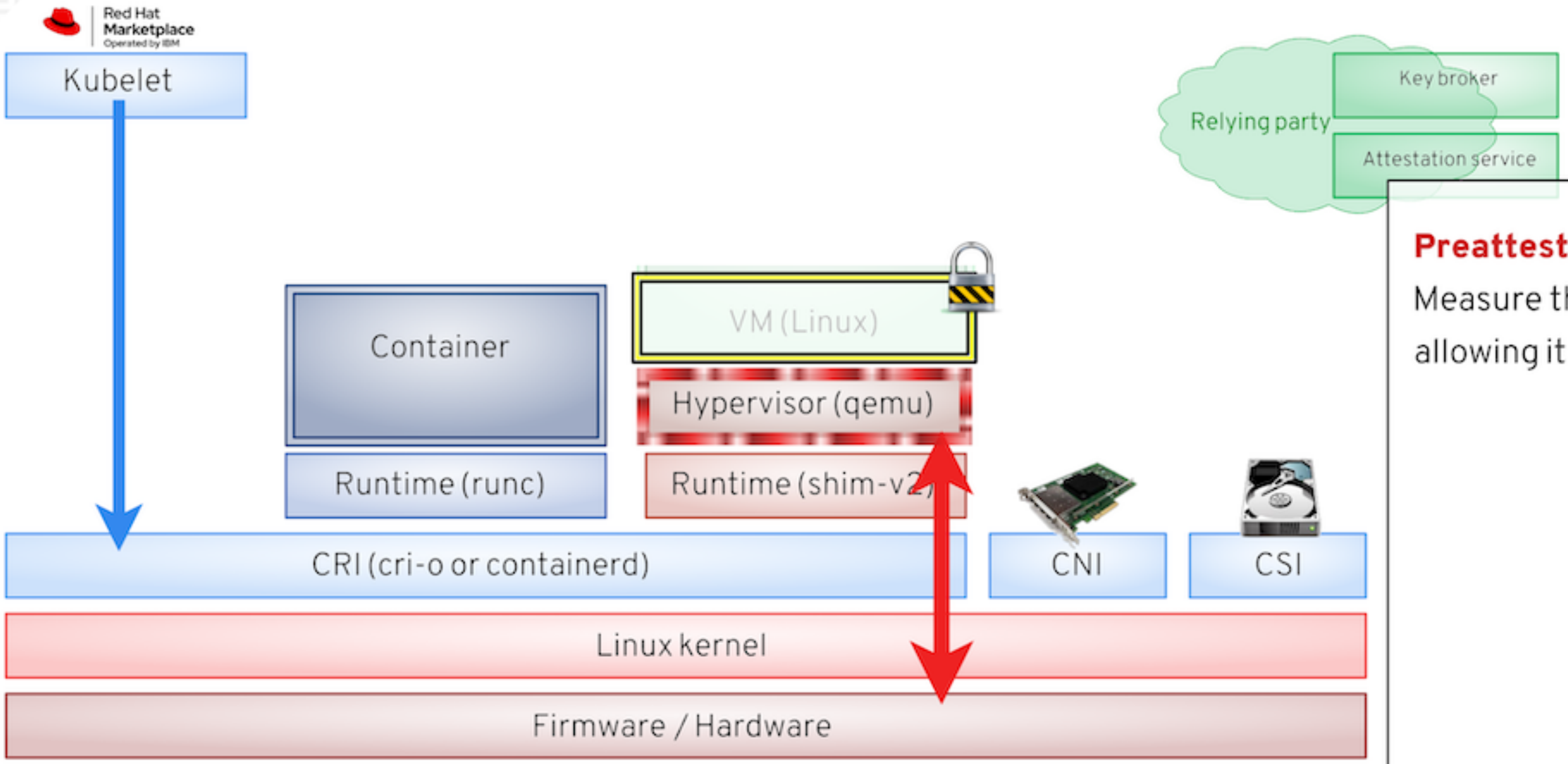
Attestation

Measuring what we run using cryptography



Attestation

Measuring what we run using cryptography

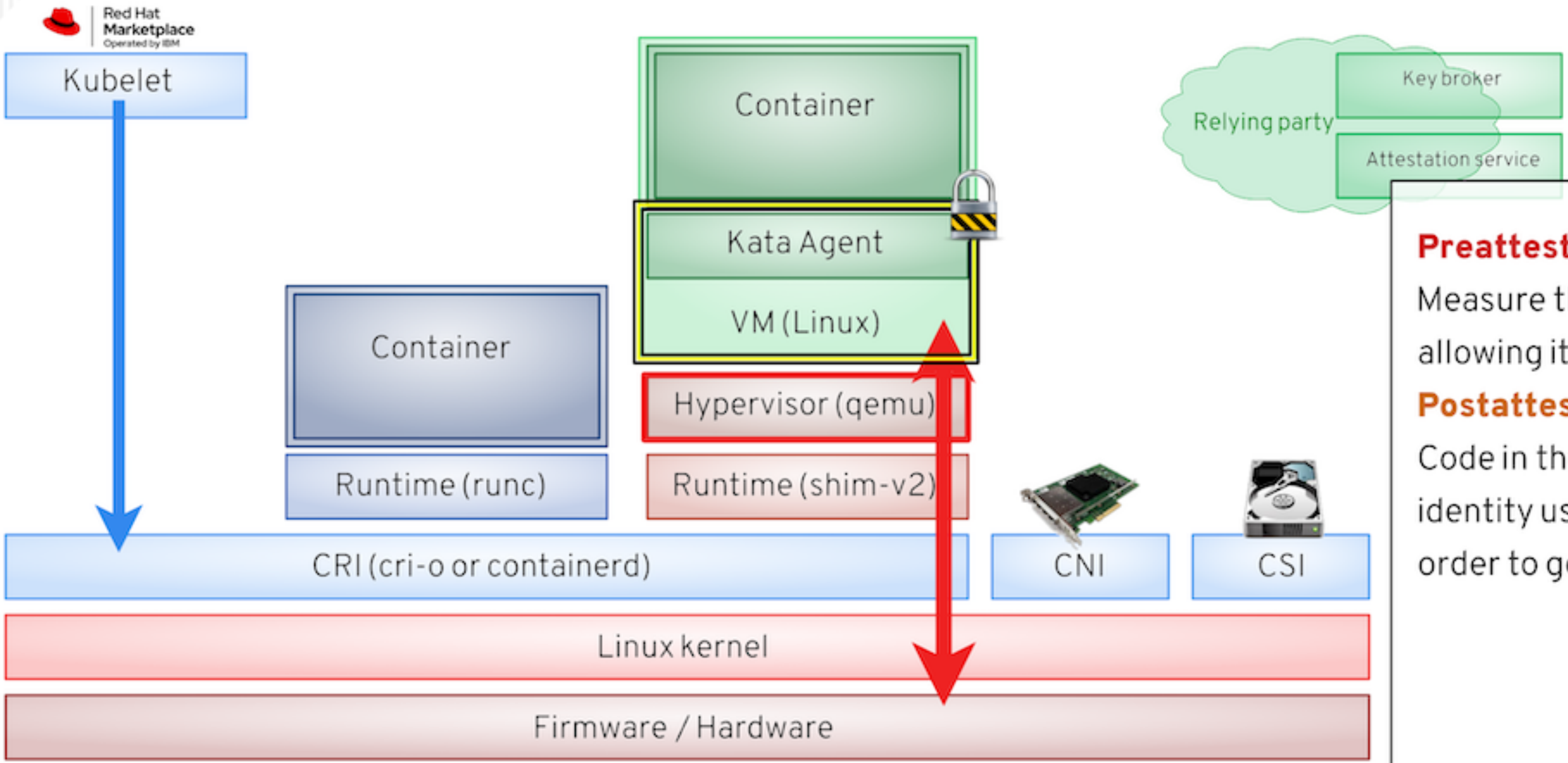


Preattestation:

Measure the payload before allowing it to start (original SEV)

Attestation

Measuring what we run using cryptography

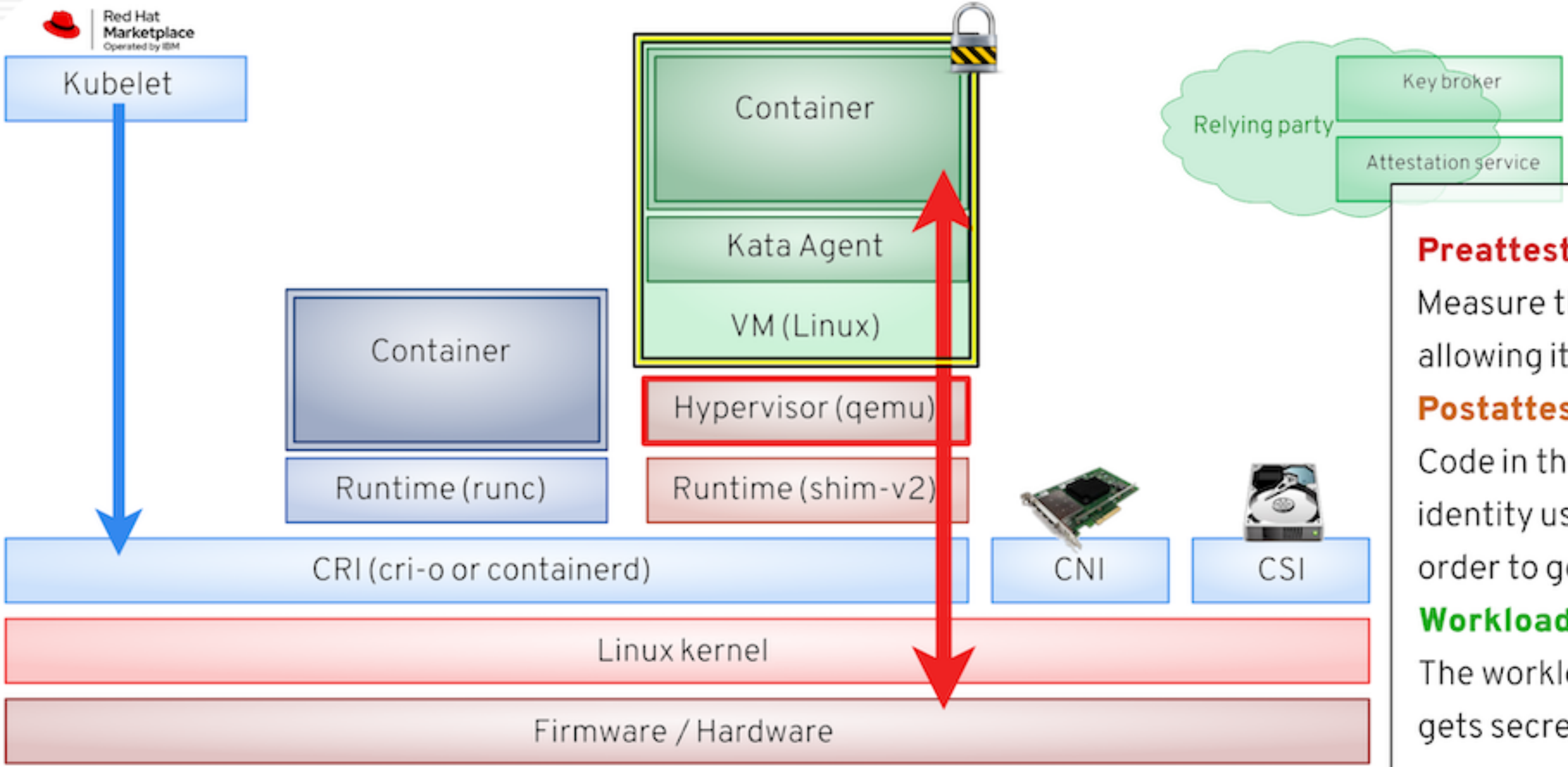


Preattestation:
Measure the payload before allowing it to start (original SEV)

Postattestation:
Code in the payload can confirm its identity using the measurement in order to get secrets

Attestation

Measuring what we run using cryptography



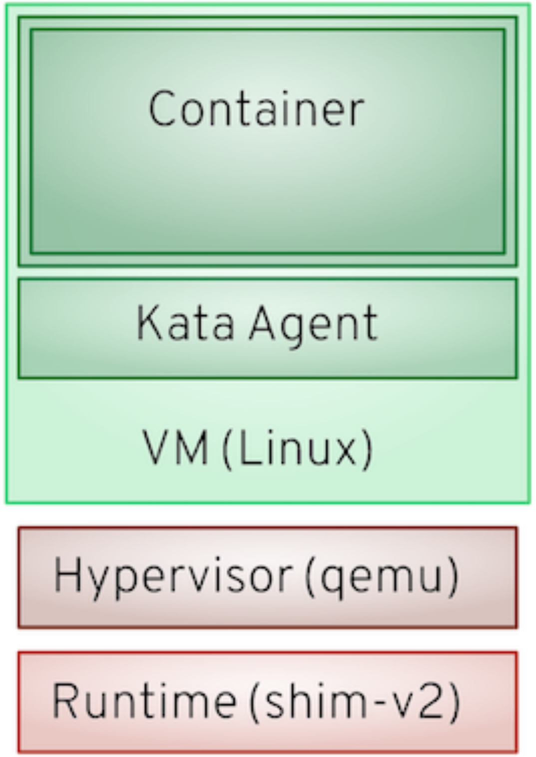
Preattestation:
Measure the payload before allowing it to start (original SEV)

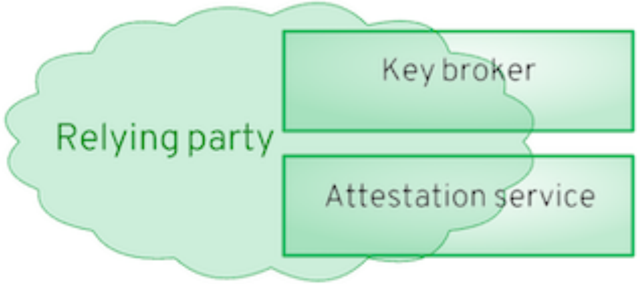
Postattestation:
Code in the payload can confirm its identity using the measurement in order to get secrets

Workload attestation
The workload itself is attested, e.g. gets secrets from attestation

How does attestation work?

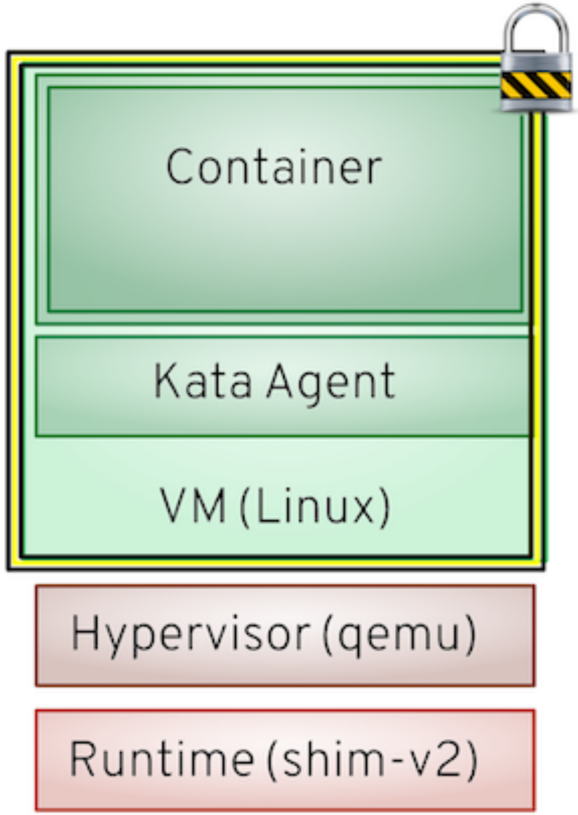
Challenge / response to deliver secrets

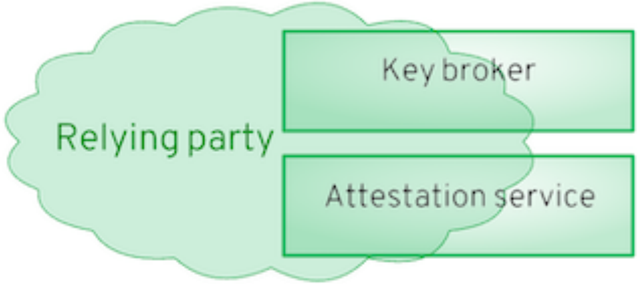




How does attestation work?

Challenge / response to deliver secrets

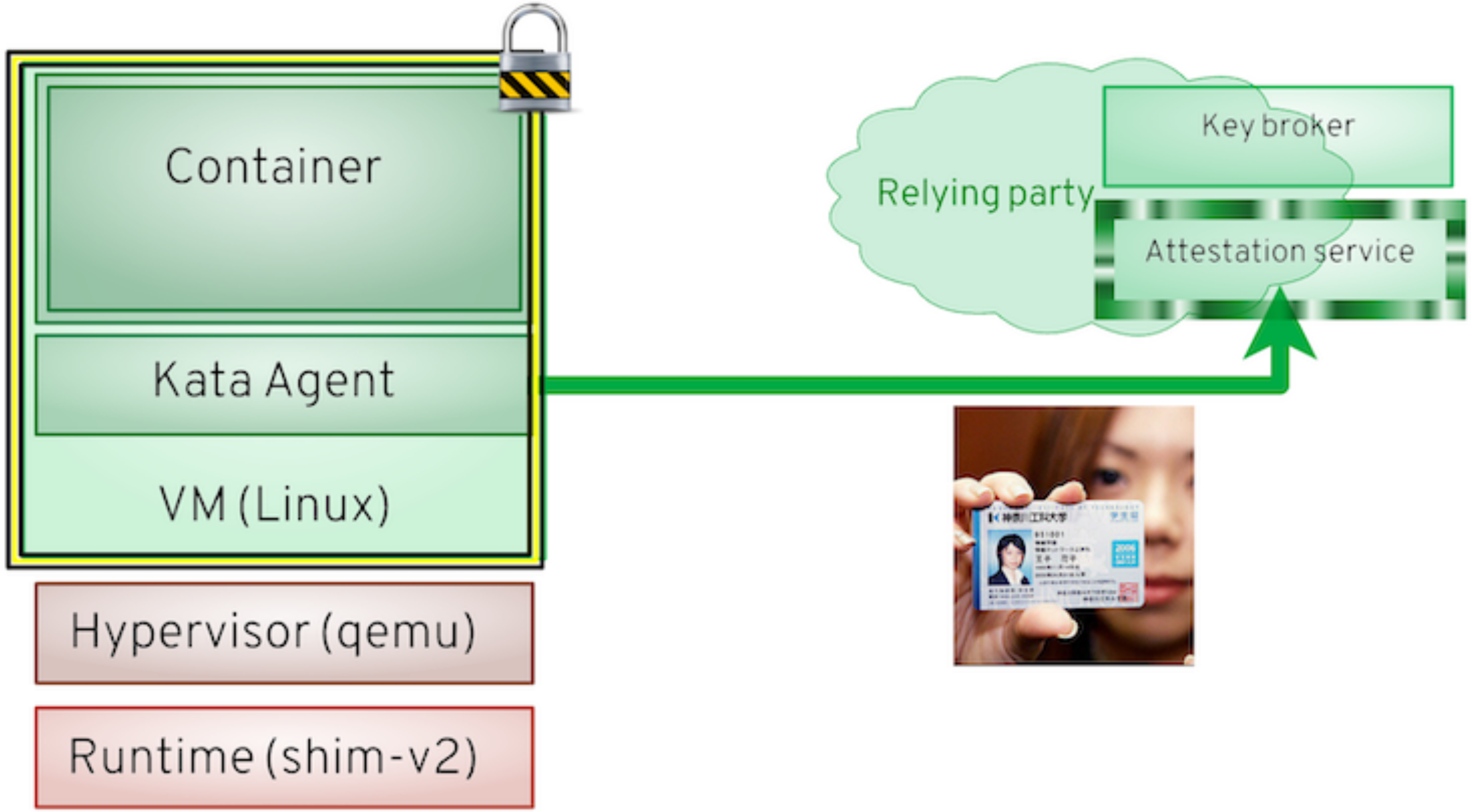




Cryptographic measurement:
Measurement of relevant memory performed by hardware / firmware

How does attestation work?

Challenge / response to deliver secrets



Cryptographic measurement:

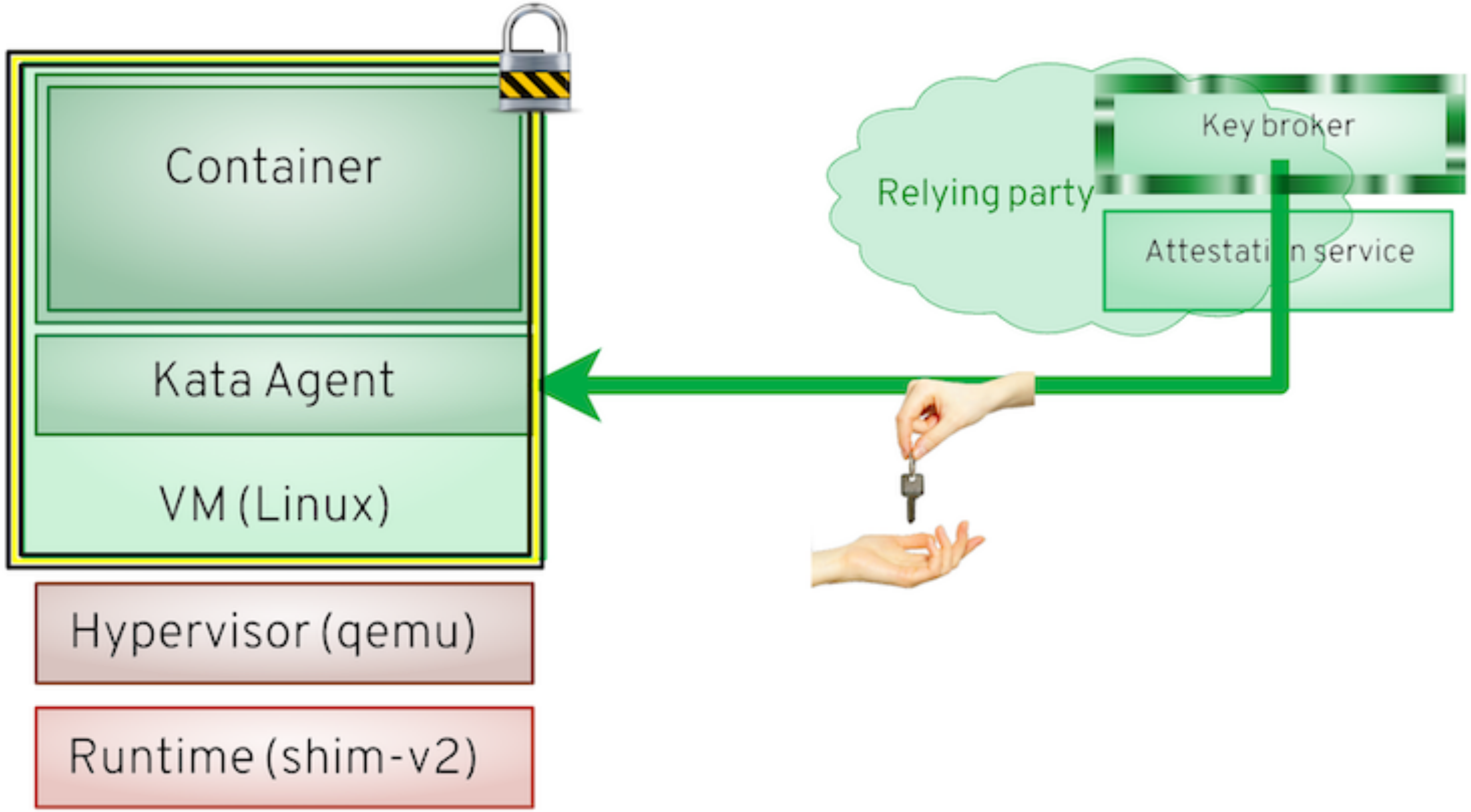
Measurement of relevant memory performed by hardware / firmware

Cryptographic challenge:

Send proof of identity (salted)

How does attestation work?

Challenge / response to deliver secrets



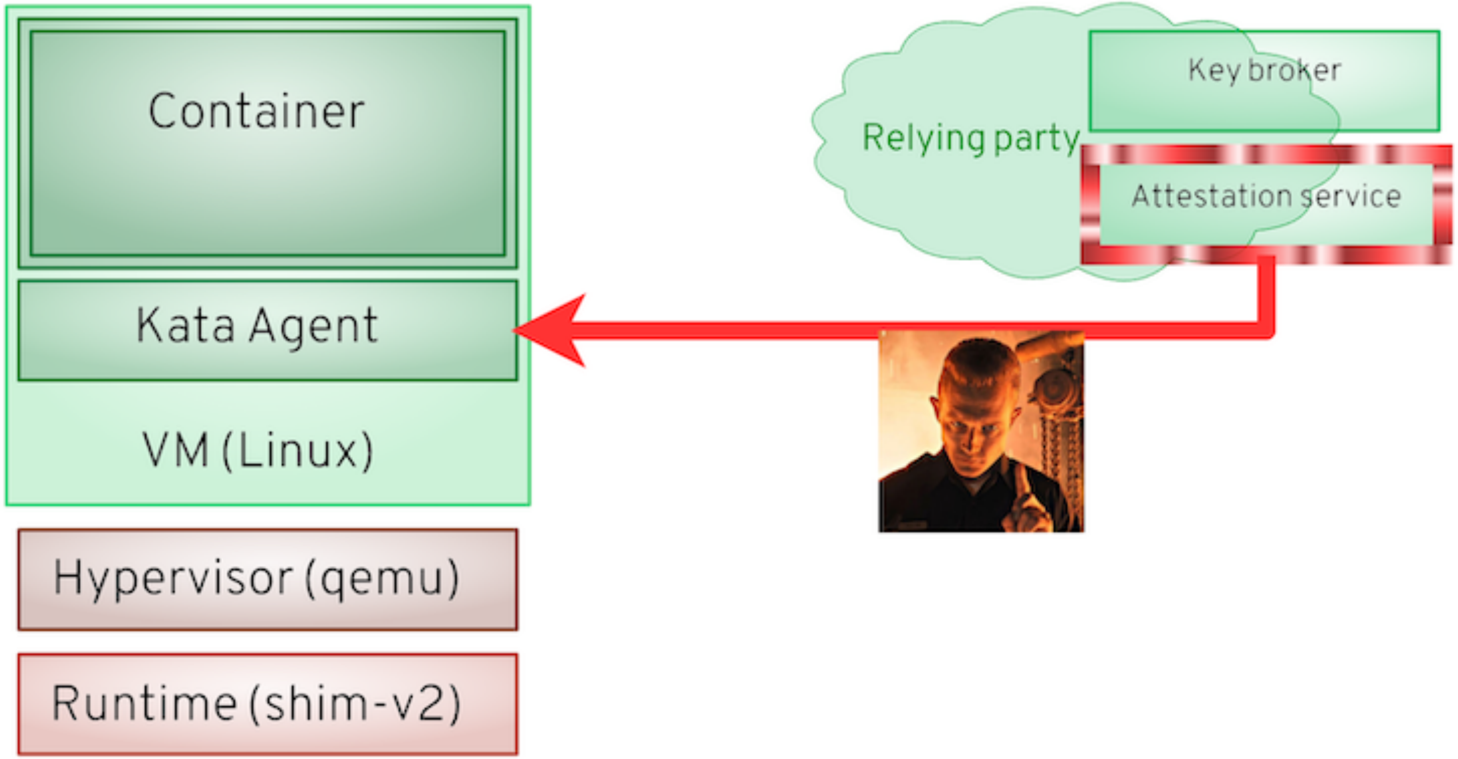
Cryptographic measurement:
Measurement of relevant memory performed by hardware / firmware

Cryptographic challenge:
Send proof of identity (salted)

Secret delivery
Ensure the workload cannot do harm if not attested

How does attestation work?

Challenge / response to deliver secrets



Cryptographic measurement:
Measurement of relevant memory performed by hardware / firmware

Cryptographic challenge:
Send proof of identity (salted)

Secret delivery
Ensure the workload cannot do harm if not attested

Remote attestation:
Can invalidate workloads e.g. if compromized

So, what is the problem?

It looks like you got everything under control

Let's start with attestation

The problem with standards...

KVM FORUM

Let's start with attestation

The problem with standards...

The good thing about standards is that there are so many to choose from.

Andrew Tannenbaum

Red Hat

Attestation is a very general thing

So there is More One Way To Do It™

- Remote Attestation Procedures (RATS)

Workgroup:	RATS Working Group
Internet-Draft:	draft-ietf-rats-architecture-05
Published:	10 July 2020
Intended Status:	Informational
Expires:	11 January 2021

Authors:
H. Birkholz *Fraunhofer SIT* D. Thaler *Microsoft* M. Richardson *Sandelman Software Works* N. Smith *Intel* W. Pan *Huawei Technologies*

Remote Attestation Procedures Architecture

Abstract

In network protocol exchanges, it is often the case that one entity (a Relying Party) requires evidence about a remote peer to assess the peer's trustworthiness, and a way to appraise such evidence. The evidence is typically a set of claims about its software and hardware platform. This document describes an architecture for such remote attestation procedures (RATS).

Note to Readers

Discussion of this document takes place on the RATS Working Group mailing list (rats@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/rats/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-rats-wg/architecture.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisior

Internet-Drafts are working documents of the Internet Engineering Ta may also distribute working documents as Internet-Drafts. The list of https://datatracker.ietf.org/drafts/current/.

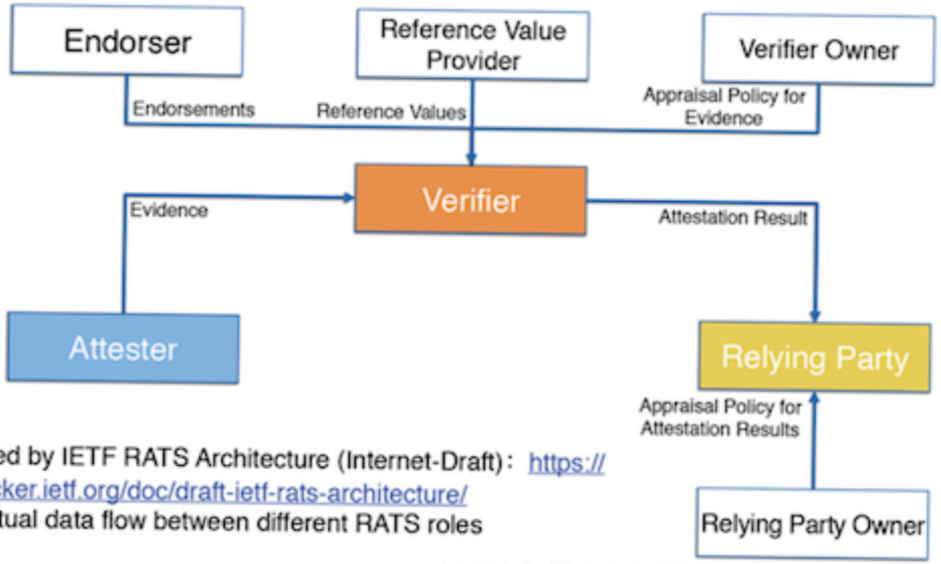
Internet-Drafts are draft documents valid for a maximum of six montl obsoleted by other documents at any time. It is inappropriate to use I to cite them other than as "work in progress."

This Internet-Draft will expire on 11 January 2021.

Attestation is a very general thing

So there is More One Way To Do It™

- Remote Attestation Procedures (RATS)
  - A relatively complex topic in itself

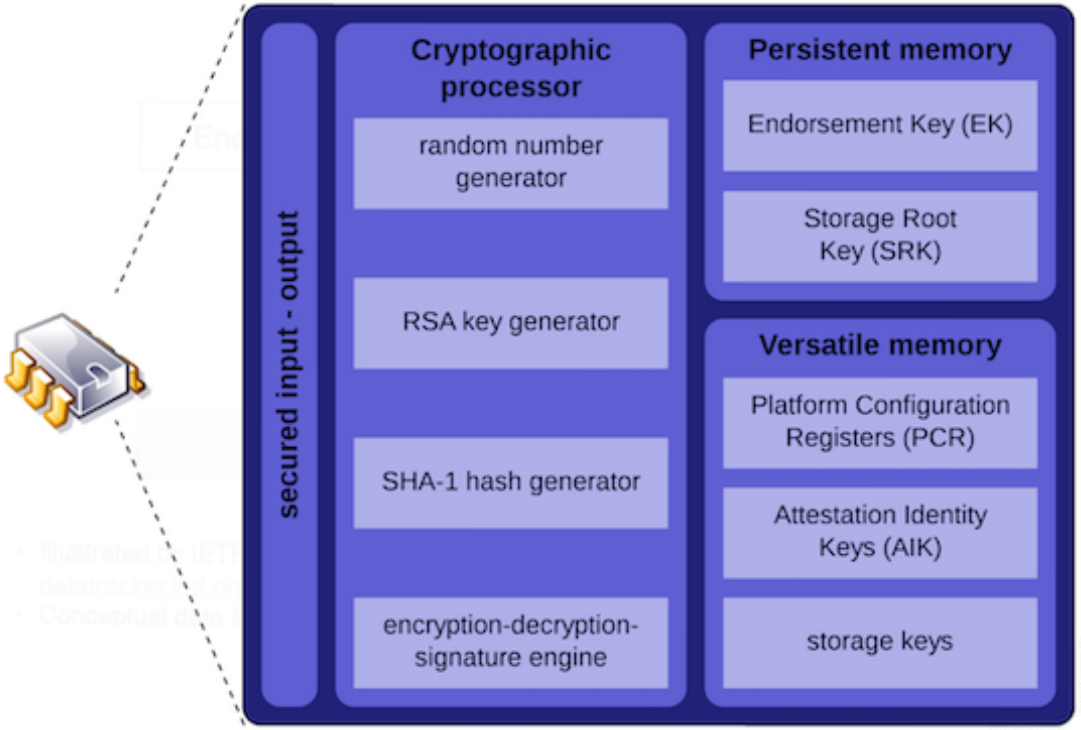


- Illustrated by IETF RATS Architecture (Internet-Draft): https://datatracker.ietf.org/doc/draft-ietf-rats-architecture/
- Conceptual data flow between different RATS roles

Attestation is a very general thing

So there is More One Way To Do It™

- Remote Attestation Procedures (RATS)
  - A relatively complex topic in itself
- Secure Boot and Trusted Platform Module



Attestation is a very general thing

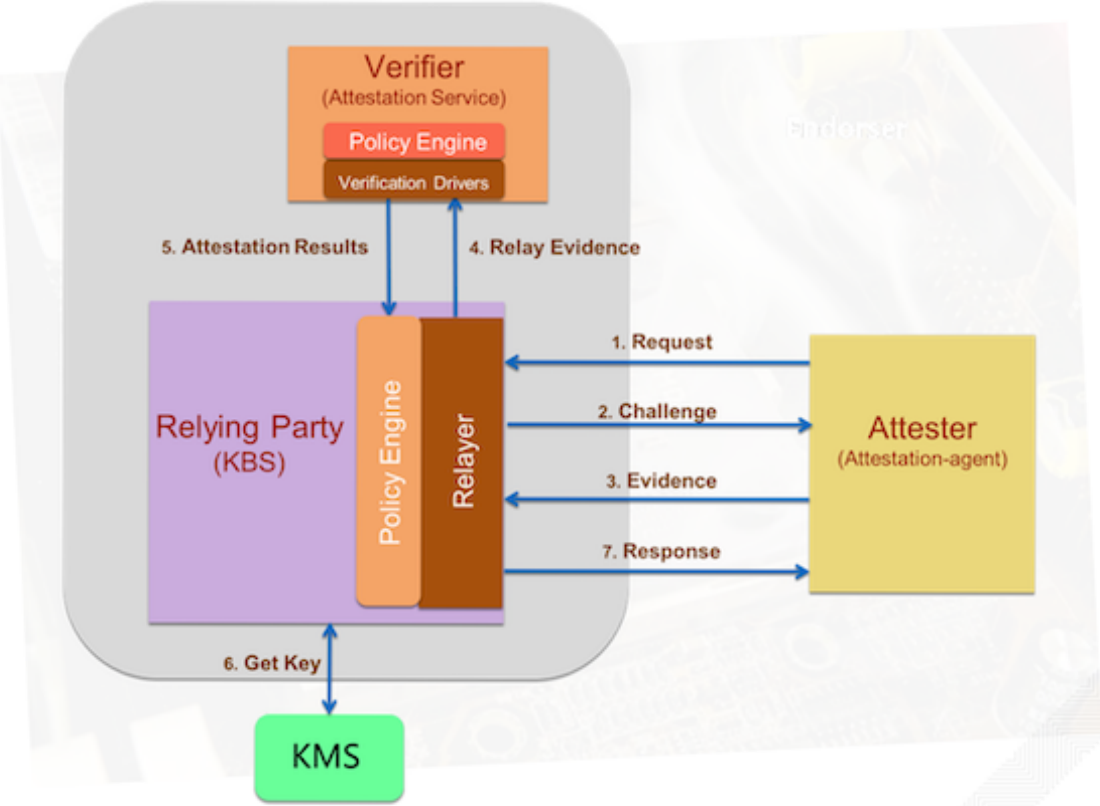
So there is More One Way To Do It™

- Remote Attestation Procedures (RATS)
  - A relatively complex topic in itself
- Secure Boot and Trusted Platform Module
  - How about manufacturing transient virtual TPMs?

Attestation is a very general thing

So there is More One Way To Do It™

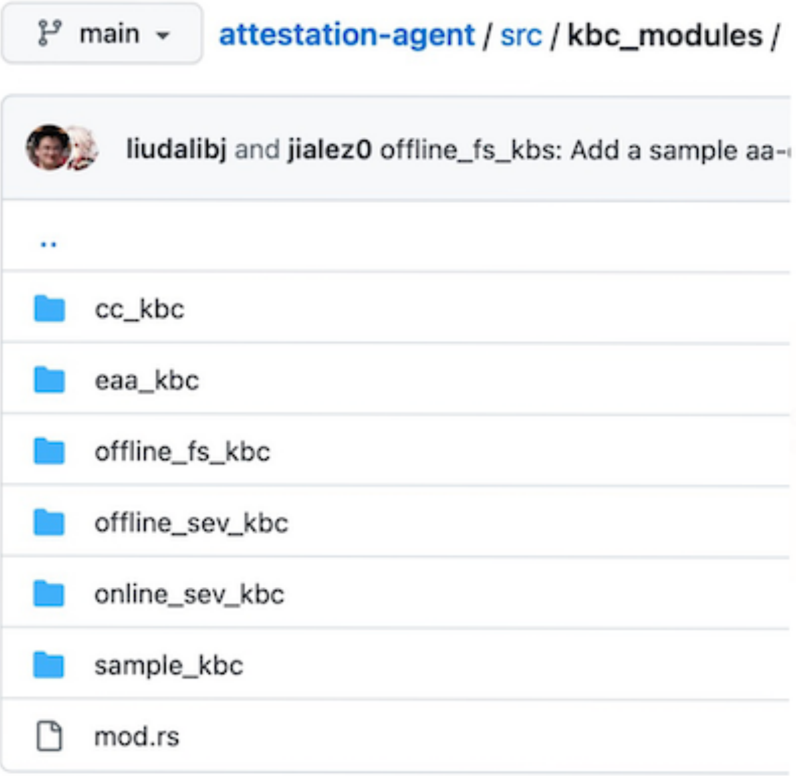
- Remote Attestation Procedures (RATS)
  - A relatively complex topic in itself
- Secure Boot and Trusted Platform Module
  - How about manufacturing transient virtual TPMs?
- Confidential Containers have their own attestation



Attestation is a very general thing

So there is More One Way To Do It™

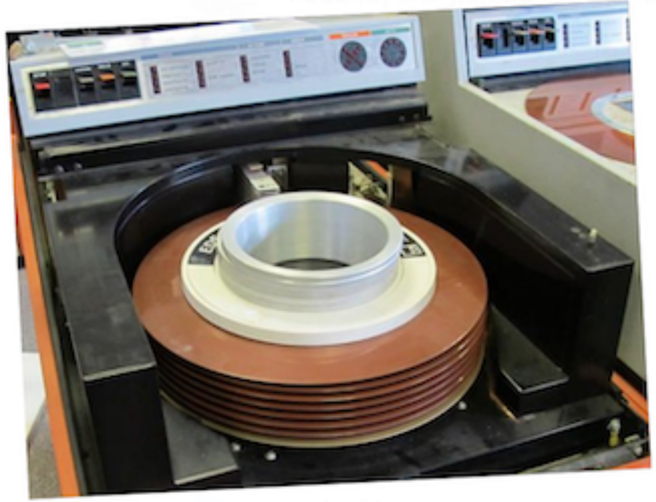
- Remote Attestation Procedures (RATS)
  - A relatively complex topic in itself
- Secure Boot and Trusted Platform Module
  - How about manufacturing transient virtual TPMs?
- Confidential Containers have their own attestation
  - Which needs to plug into all existing technologies



Attestation is a very general thing

So there is More One Way To Do It™

- Remote Attestation Procedures (RATS)
  - A relatively complex topic in itself
- Secure Boot and Trusted Platform Module
  - How about manufacturing transient virtual TPMs?
- Confidential Containers have their own attestation
  - Which needs to plug into all existing technologies
- Hide platform differences from user-space



=

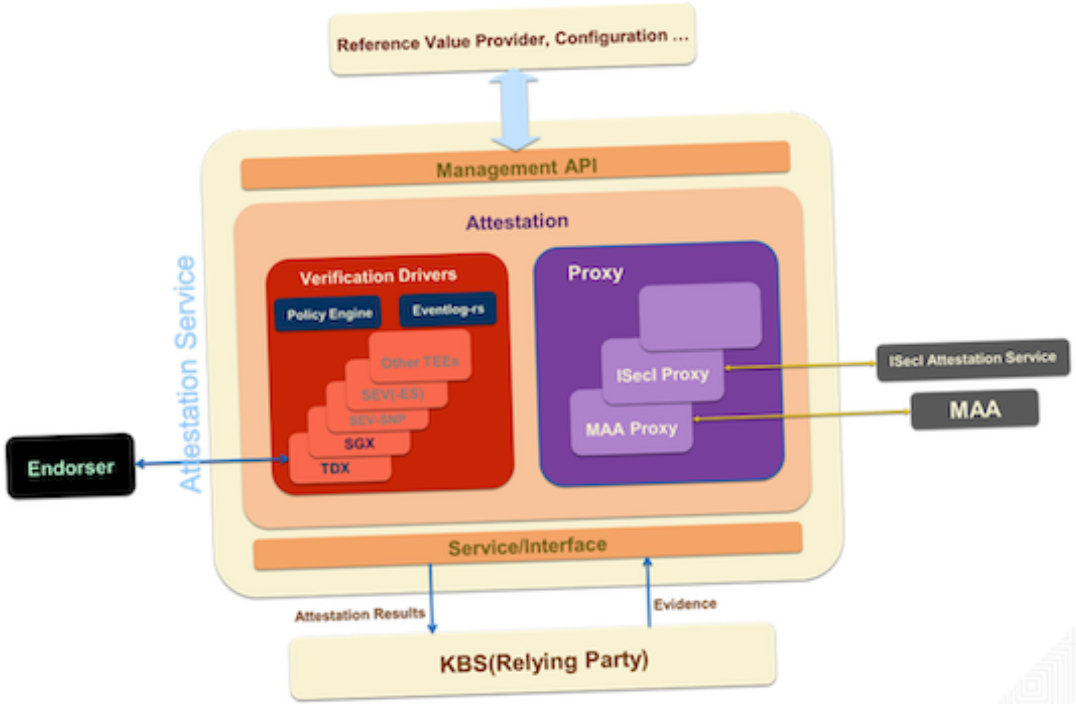
Attestation and Key Brokering

Attestation is robust only if it hands you secrets

Attestation and Key Brokering

Attestation is robust only if it hands you secrets

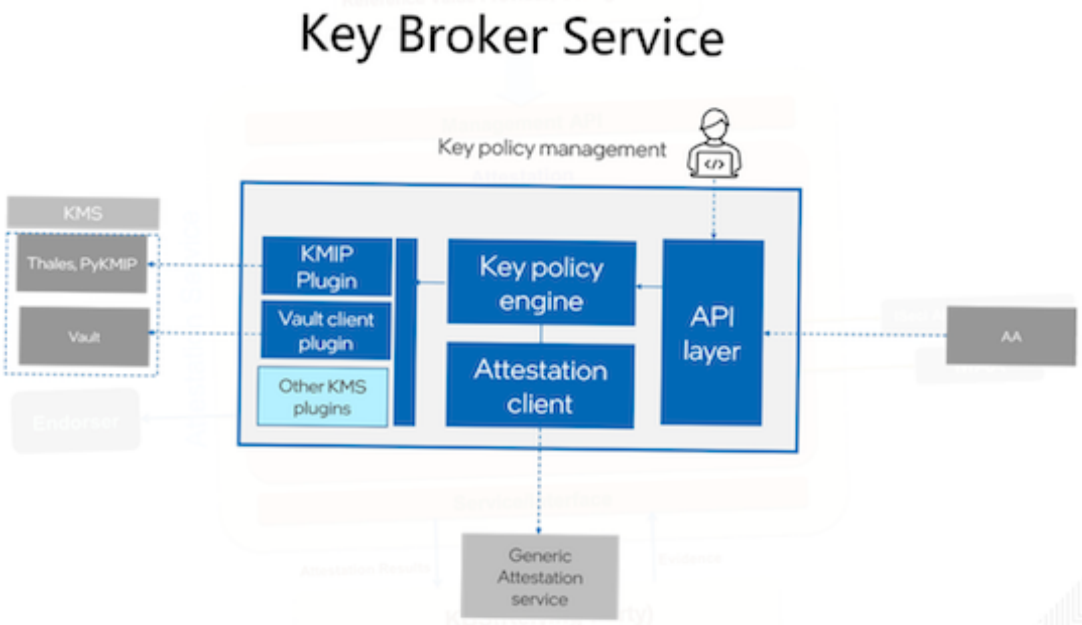
- Attestation Service (AS, aka Verifier): Checks your ID



Attestation and Key Brokering

Attestation is robust only if it hands you secrets

- Attestation Service (AS, aka Verifier): Checks your ID
- Key Broker Service (KBS): Hands over secrets



Attestation and Key Brokering

Attestation is robust only if it hands you secrets

- Attestation Service (AS, aka Verifier): Checks your ID
- Key Broker Service (KBS): Hands over secrets
- Your workload must *need* that secret

Attestation and Key Brokering

Attestation is robust only if it hands you secrets

- Attestation Service (AS, aka Verifier): Checks your ID
- Key Broker Service (KBS): Hands over secrets
- Your workload must *need* that secret

- KBS protocol defined by Confidential Containers

RCAR semantics

The semantics of attestation defined by KBS is a simple and extensible four-step model, which uses JSON structure to organize information. As follows:

1. **Request**: The KBC sends to KBS, calls the service API provided by KBS to request resources.
2. **Challenge**: After receiving the request, KBS returns a challenge to KBC and asks KBC to send evidence to prove that its environment (HW-TEE) is safe and reliable.
3. **Attestation**: KBC sends the evidence and the HW-TEE generated, ephemeral public key to the KBS.
4. **Response**: The `Response` payload includes the resource data requested when sending the initial `Request` to the specific KBS resource endpoint. The resource data is protected by the ephemeral public key passed via the previously sent Attestation payload. Within the valid time of the token, KBC can directly request resources or services from KBS by virtue of the token without going through step 2 and step 3.

Request

The payload format of the request is as follows:

(Note that the `/*...*/` comments are not valid in JSON, and must not be use

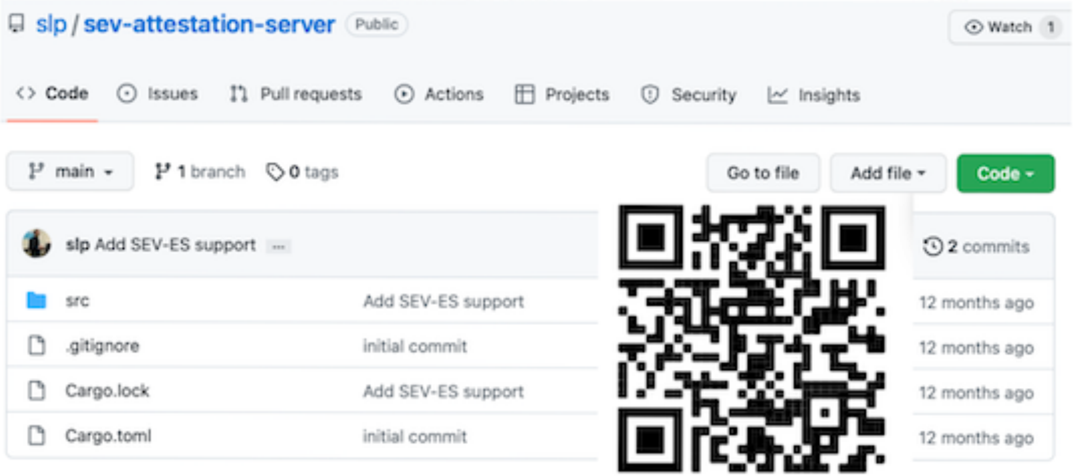
```
{
    "request": {
        /* Attestation protocol version number used by KBC */
        "version": "0.1.0",
        /* Type of HW-TEE platforms where KBC is located, such as
        "tee": "",
        /* Access token to avoid multiple attempts triggered by consecutive requests. */
        "token": "",
        /* Reserved fields are used to support some special requests sent by HW-TEE. */
        "extra-params": {}
    }
}
```

Attestation and Key Brokering

Attestation is robust only if it hands you secrets

- Attestation Service (AS, aka Verifier): Checks your ID
- Key Broker Service (KBS): Hands over secrets
- Your workload must *need* that secret

- KBS protocol defined by Confidential Containers
  - … first implemented by Sergio Lopez for libkrun



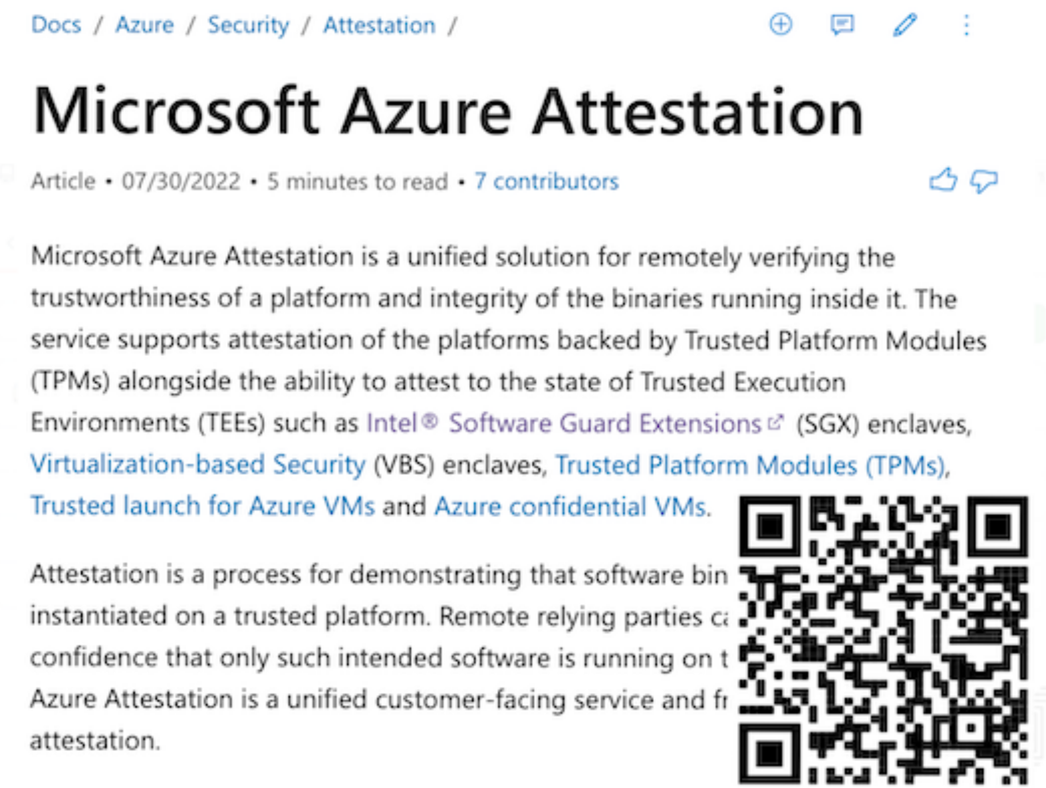
Attestation and Key Brokering

Attestation is robust only if it hands you secrets

- Attestation Service (AS, aka Verifier): Checks your ID
- Key Broker Service (KBS): Hands over secrets
- Your workload must *need* that secret

- KBS protocol defined by Confidential Containers
  - ... first implemented by Sergio Lopez for libkrun

- Services will come from different players



Docs / Azure / Security / Attestation /

Microsoft Azure Attestation

Article • 07/30/2022 • 5 minutes to read • 7 contributors

Microsoft Azure Attestation is a unified solution for remotely verifying the trustworthiness of a platform and integrity of the binaries running inside it. The service supports attestation of the platforms backed by Trusted Platform Modules (TPMs) alongside the ability to attest to the state of Trusted Execution Environments (TEEs) such as Intel® Software Guard Extensions (SGX) enclaves, Virtualization-based Security (VBS) enclaves, Trusted Platform Modules (TPMs), Trusted launch for Azure VMs and Azure confidential VMs.

Attestation is a process for demonstrating that software bin instantiated on a trusted platform. Remote relying parties c confidence that only such intended software is running on t Azure Attestation is a unified customer-facing service and fr attestation.

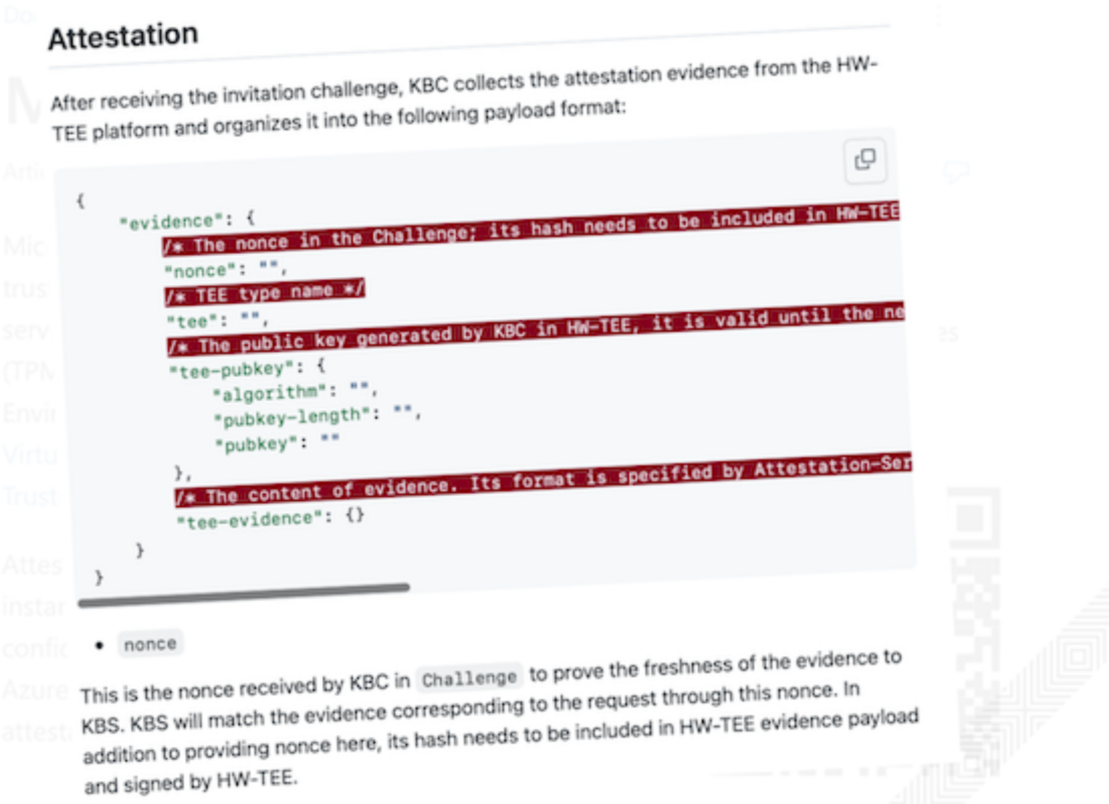
Attestation and Key Brokering

Attestation is robust only if it hands you secrets

- Attestation Service (AS, aka Verifier): Checks your ID
- Key Broker Service (KBS): Hands over secrets
- Your workload must *need* that secret

- KBS protocol defined by Confidential Containers
  - … first implemented by Sergio Lopez for libkrun

- Services will come from different players
  - So we need to define protocols, not just code



Attestation

After receiving the invitation challenge, KBC collects the attestation evidence from the HW-TEE platform and organizes it into the following payload format:

```
{
    "evidence": {
        /* The nonce in the Challenge; its hash needs to be included in HW-TEE
        "nonce": "",
        /* TEE type name */
        "tee": "",
        /* The public key generated by KBC in HW-TEE, it is valid until the ne
        "tee-pubkey": {
            "algorithm": "",
            "pubkey-length": "",
            "pubkey": ""
        },
        /* The content of evidence. Its format is specified by Attestation-Ser
        "tee-evidence": {}
    }
}
```

- `nonce`

This is the nonce received by KBC in `Challenge` to prove the freshness of the evidence to KBS. KBS will match the evidence corresponding to the request through this nonce. In addition to providing nonce here, its hash needs to be included in HW-TEE evidence payload and signed by HW-TEE.

Then there is performance

A hardware vendor's dream...

Then there is performance

A hardware vendor's dream...

A state-of-the-art calculation requires 100 hours of CPU time on the state-of-the-art computer, independent of the decade.

Edward Teller

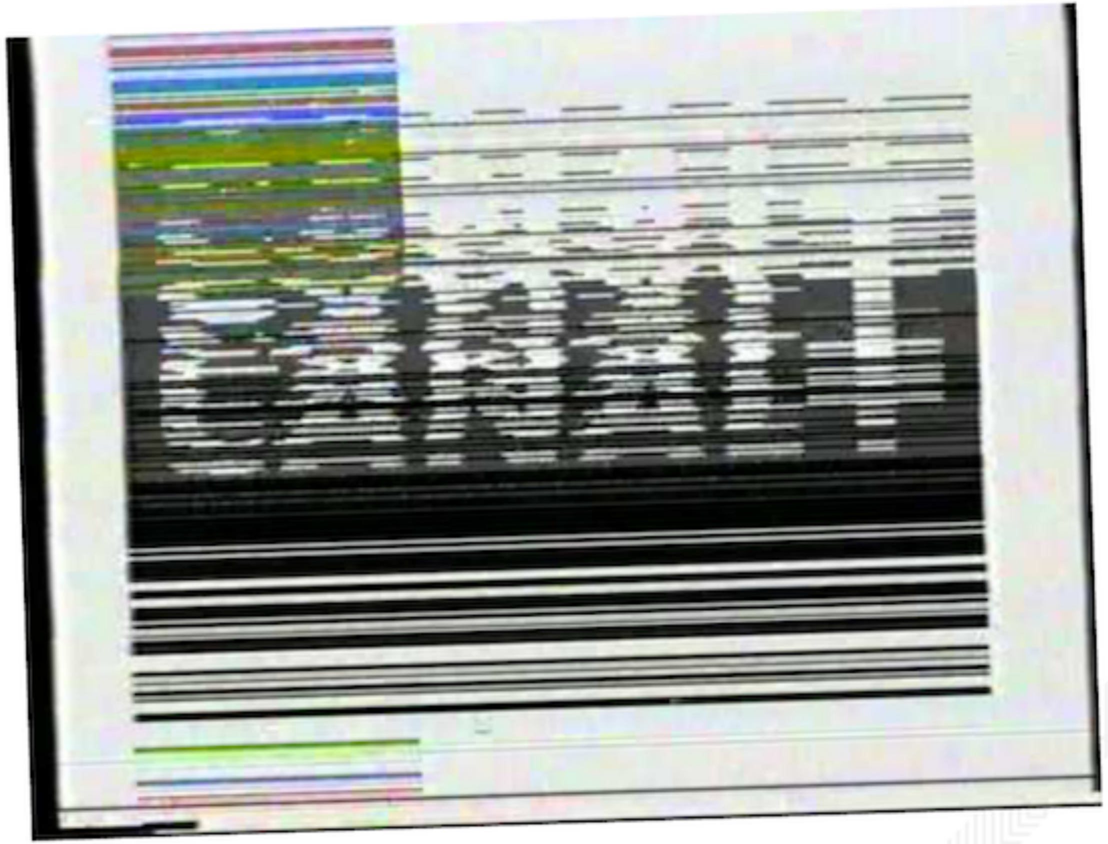
The cost of confidentiality

Confidentiality gets in the way of deduplication

The cost of confidentiality

Confidentiality gets in the way of deduplication

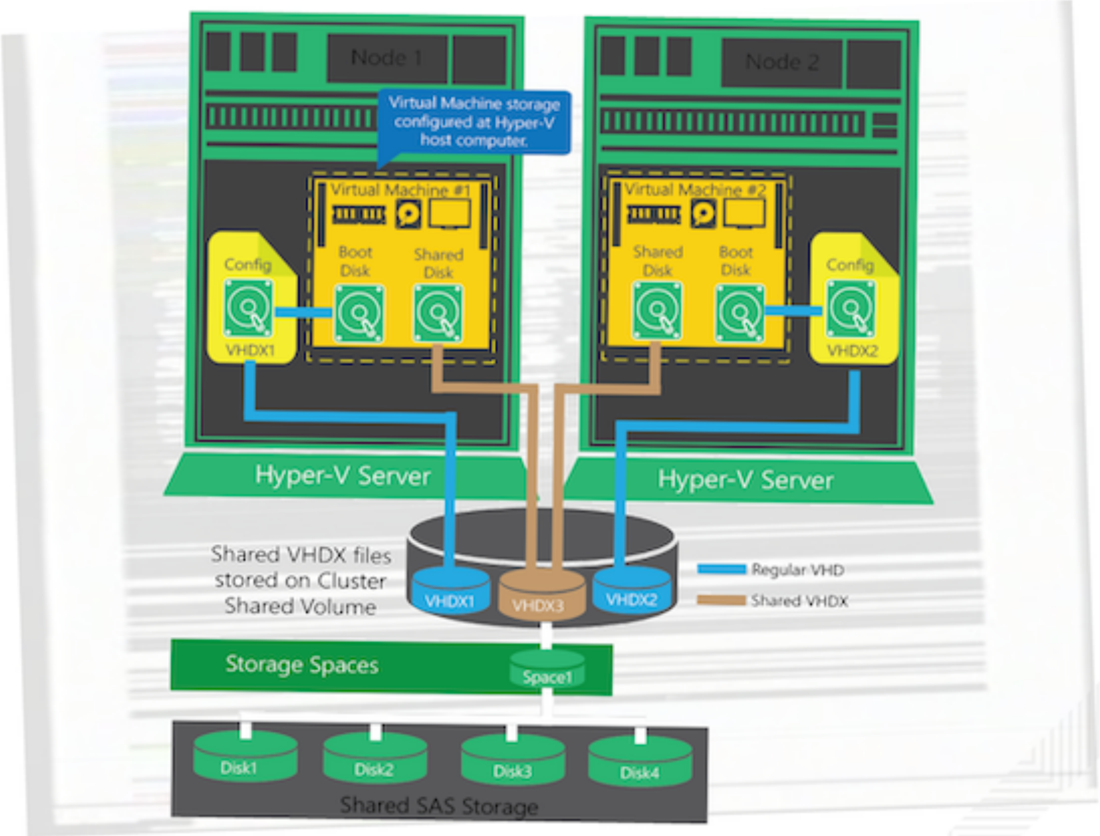
- Downloaded images are now encrypted



The cost of confidentiality

Confidentiality gets in the way of deduplication

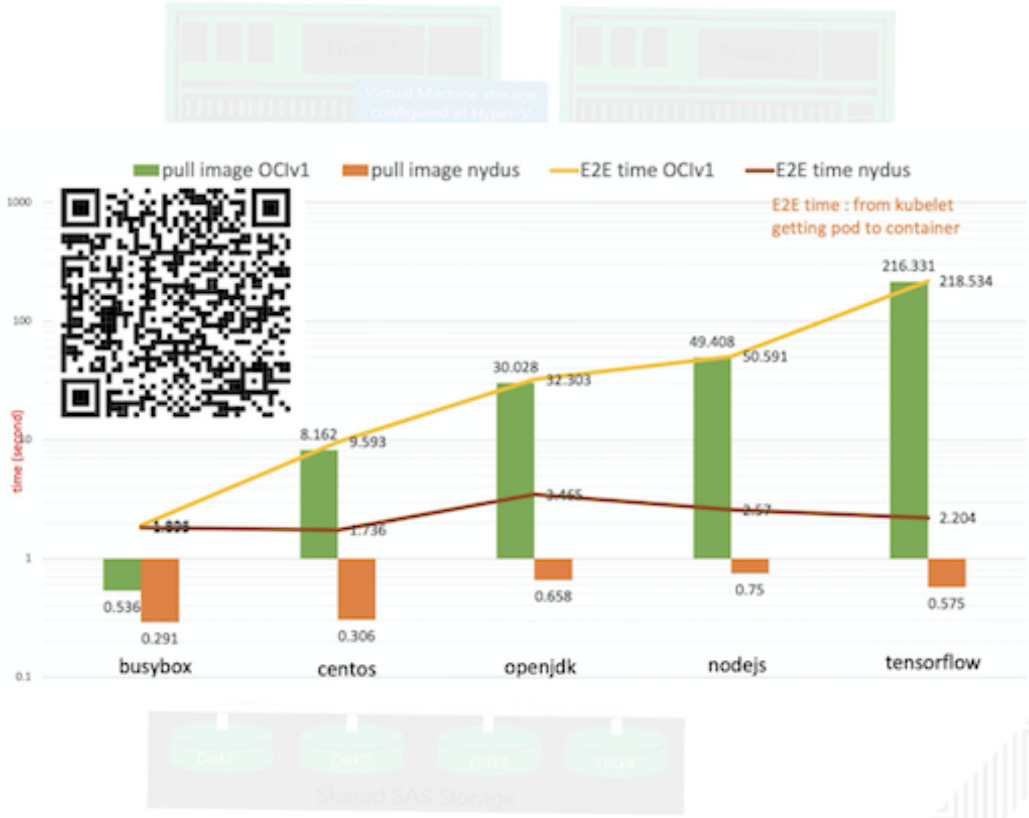
- Downloaded images are now encrypted
  - So they can't be shared between pods



The cost of confidentiality

Confidentiality gets in the way of deduplication

- Downloaded images are now encrypted
  - So they can't be shared between pods
  - This increases disk and networking costs



KVM FORUM
Red Hat

The cost of confidentiality

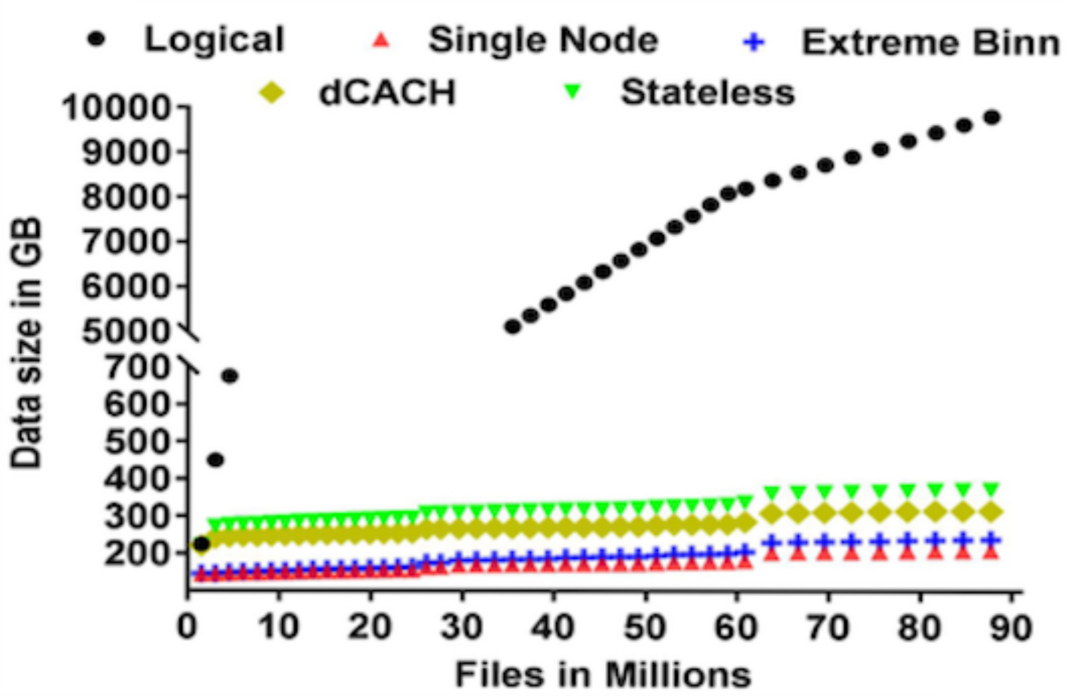
Confidentiality gets in the way of deduplication

- Downloaded images are now encrypted
  - So they can't be shared between pods
  - This increases disk and networking costs
- Disks too must be crypted

The cost of confidentiality

Confidentiality gets in the way of deduplication

- Downloaded images are now encrypted
  - So they can't be shared between pods
  - This increases disk and networking costs
- Disks too must be crypted
  - Therefore, no deduplication



The cost of confidentiality

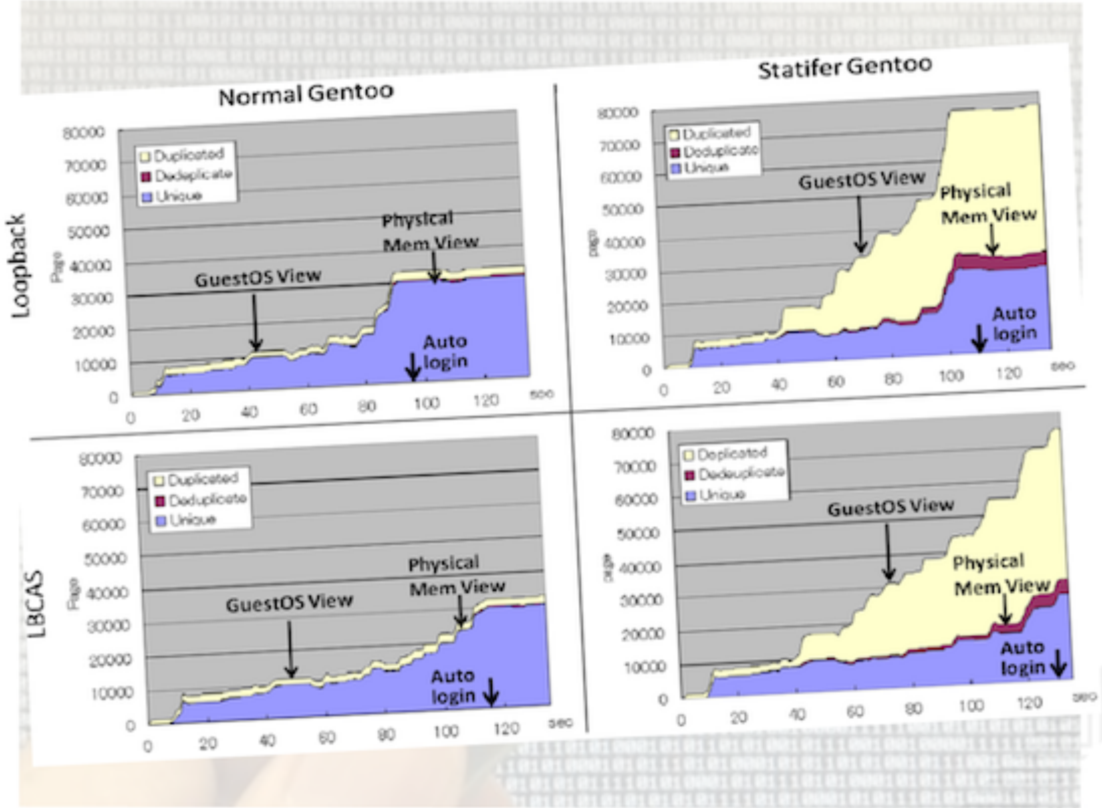
Confidentiality gets in the way of deduplication

- Downloaded images are now encrypted
  - So they can't be shared between pods
  - This increases disk and networking costs
- Disks too must be crypted
  - Therefore, no deduplication
- Memory is crypted too

The cost of confidentiality

Confidentiality gets in the way of deduplication

- Downloaded images are now encrypted
  - So they can't be shared between pods
  - This increases disk and networking costs
- Disks too must be crypted
  - Therefore, no deduplication
- Memory is crypted too
  - No kernel same-page merging



KVM FORUM
Red Hat

The cost of confidentiality

Confidentiality gets in the way of deduplication

- Downloaded images are now encrypted
  - So they can't be shared between pods
  - This increases disk and networking costs
- Disks too must be crypted
  - Therefore, no deduplication
- Memory is crypted too
  - No kernel same-page merging
- Who has a patch for Crypto KSM?

Good news: manageable runtime cost

We encrypt memory on the fly, but it's "cheap"

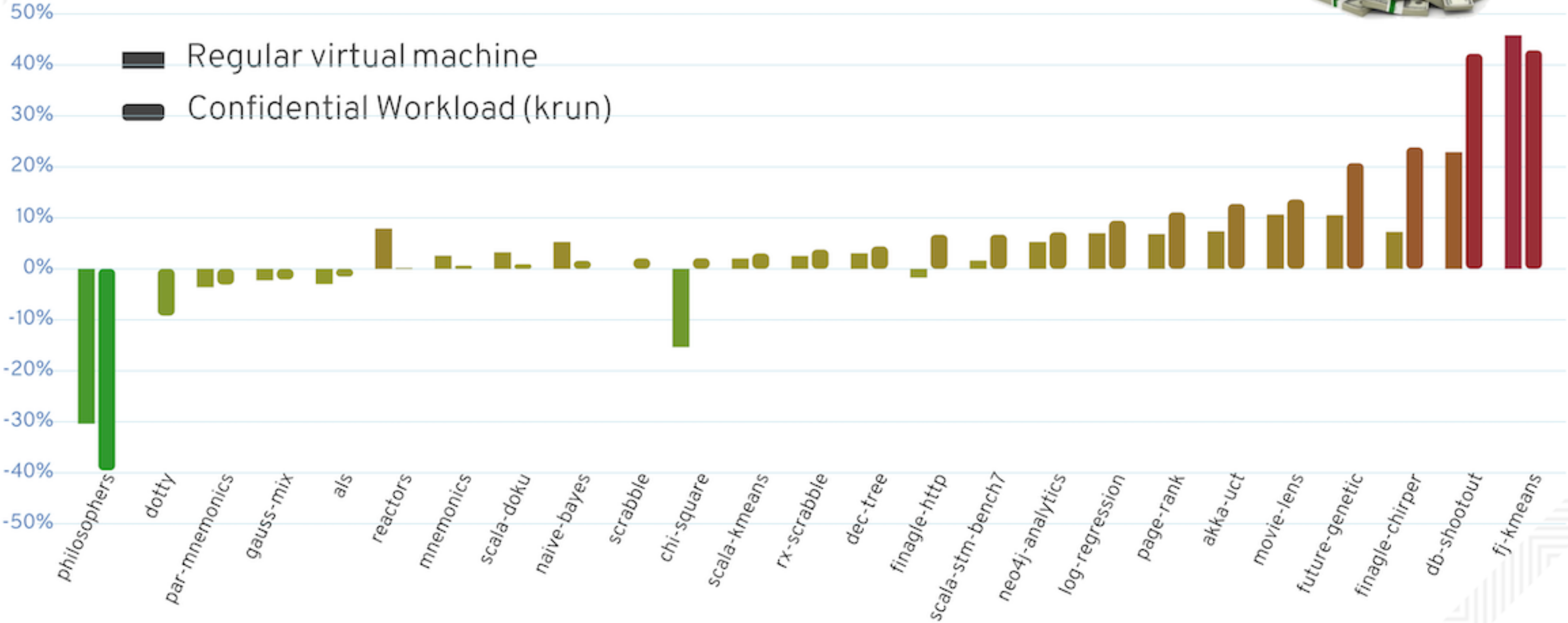


Image (re)download

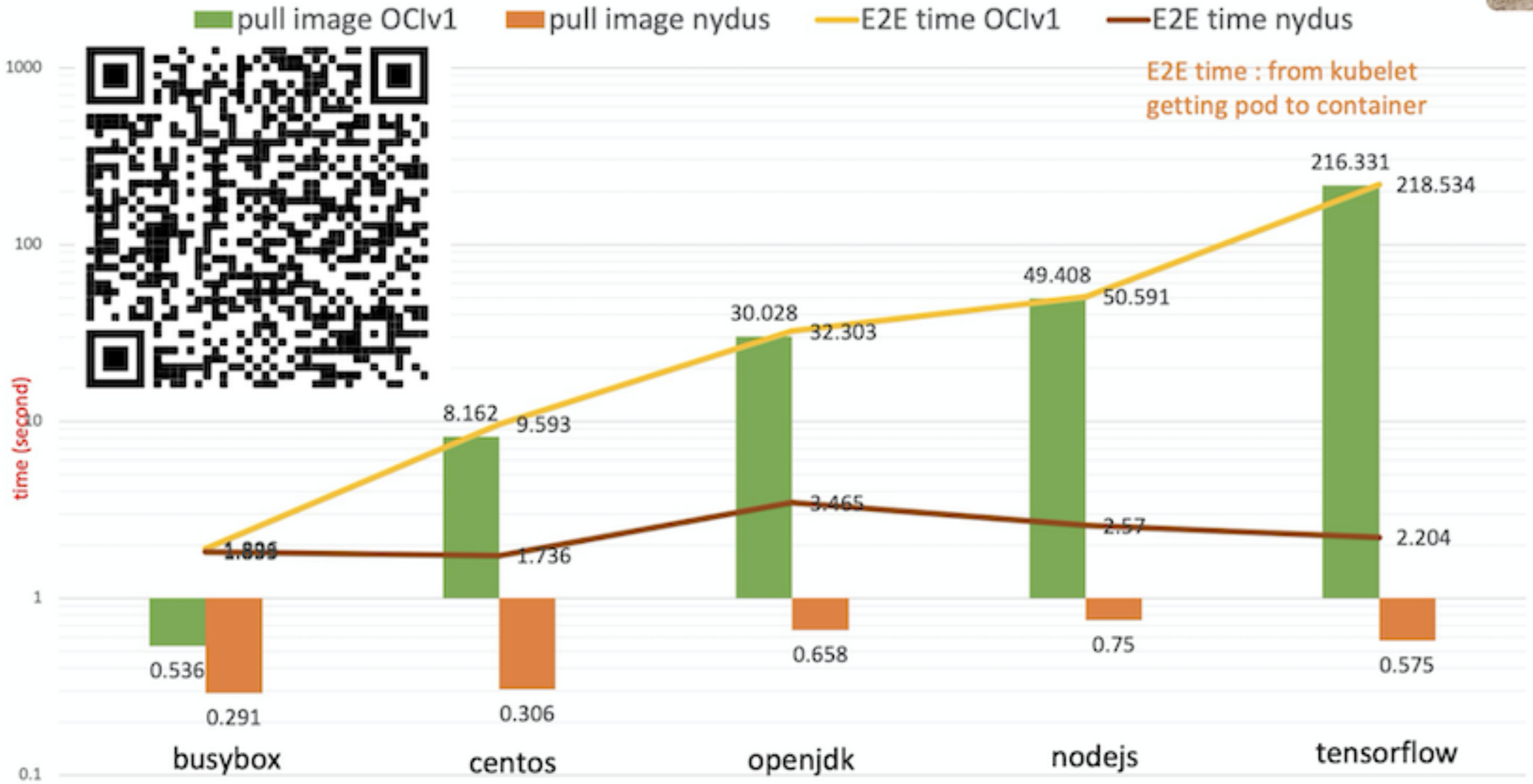
The "Lather, Rinse, Repeat" school of programming

History is always repeating itself,
but each time the price goes up

Will Durant

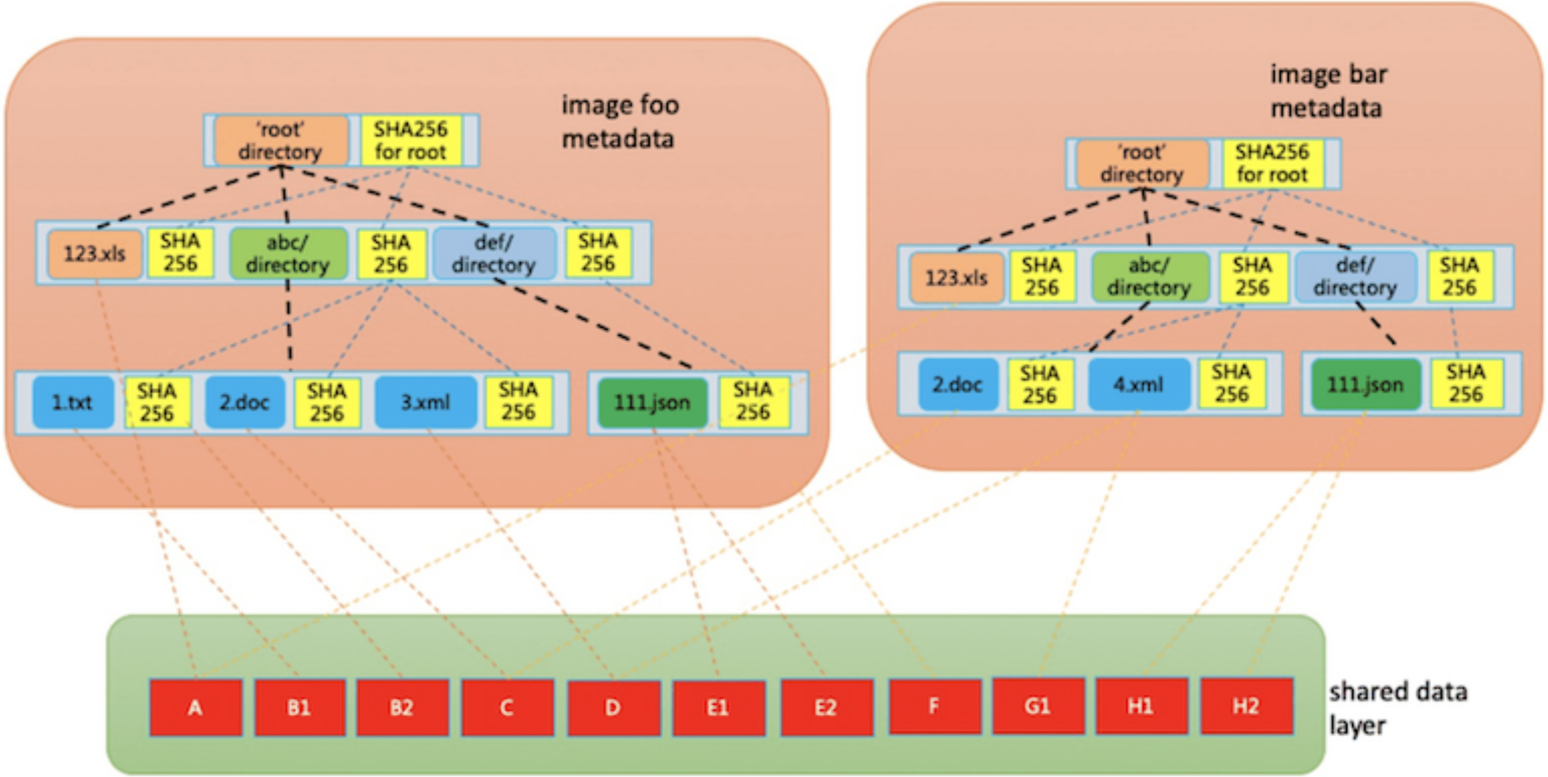
Cache + Dedup + Crypto?

Can we get something like Nydus on encrypted blocks?



Cache + Dedup + Crypto?

Can we get something like Nydus on encrypted blocks?



Cache + Dedup + Crypto?

Can we get something like Nydus on encrypted blocks?

The tree looks something like:

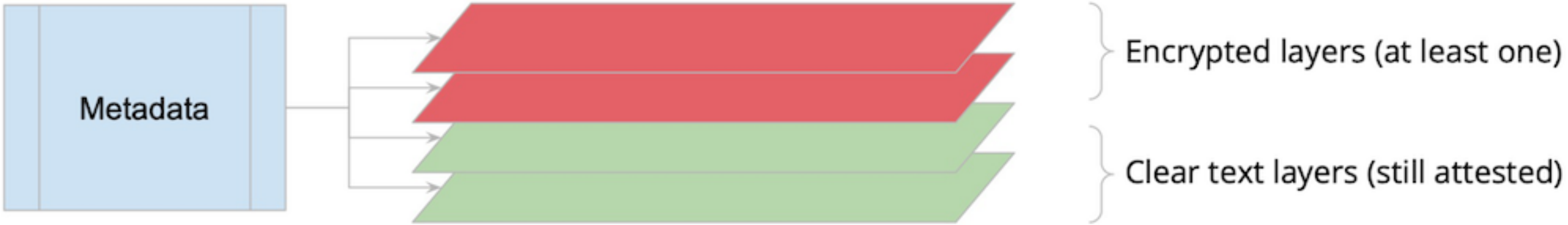
alg = sha256, num_blocks = 32768, block_size = 4096

```
                              [    root    ]
                             /    . . .    \
                [entry_0]                       [entry_1]
               /  . . .  \                       . . .   \
    [entry_0_0]   . . .  [entry_0_127]    . . . .  [entry_1_127]
      / ... \             /   . . .  \            /           \
blk_0 ... blk_127  blk_16256   blk_16383      blk_32640 . . . blk_32767
```

Source:
https://www.kernel.org/doc/html/latest/admin-guide/device-mapper/verity.html

Cache + Dedup + Crypto?

Can we get something like Nydus on encrypted blocks?



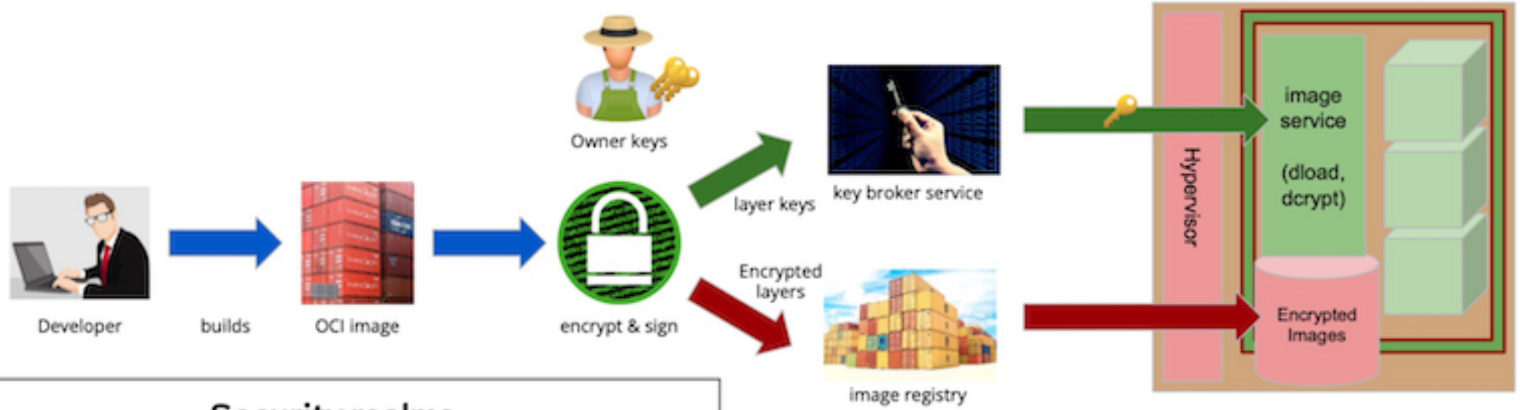
- To attest or decrypt, you need the entire layer

Cache + Dedup + Crypto?

Can we get something like Nydus on encrypted blocks?

Building encrypted & signed images

- **Generating an image** that cannot be used without a secret
- **Use OCI standard** encrypted image format – which is still actively developed
- **Build pipeline now has *two* outputs**, one to image registry, the other to key broker service



Security realms

Standard: Normal security applies (e.g. corporate)

Public: Untrusted infrastructure, encryption applies

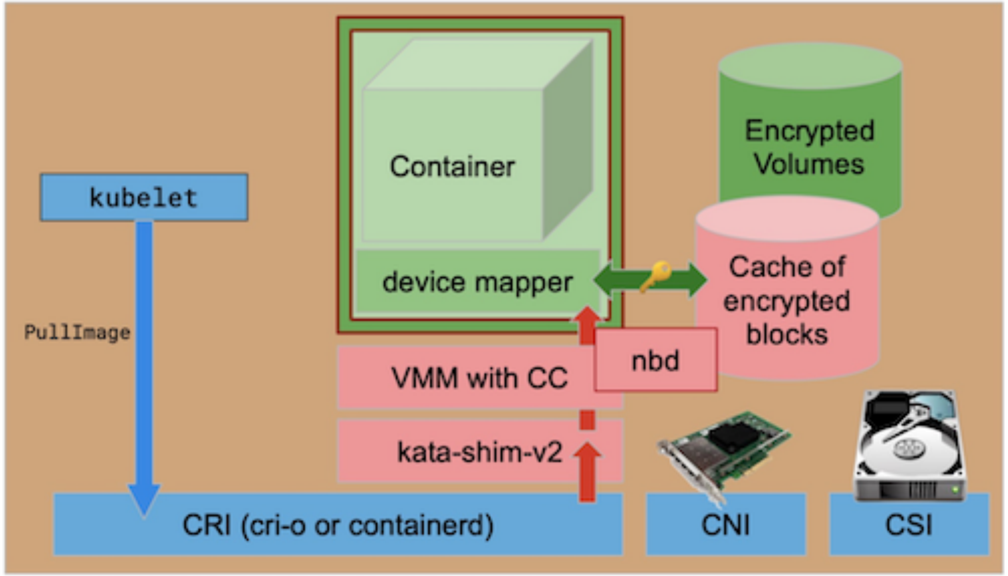
Owner: Trusted infrastructure

Cache + Dedup + Crypto?

Can we get something like Nydus on encrypted blocks?

Use the host as a block cache for image downloads by the guest

- Use **device mapper** in guest to deal with (block level) encryption and integrity
- Use **nbdkit** on the host side to deliver blocks, with local caching



Pass PullImage API to guest

Instead of storing a whole encrypted layer on local disk, we use the host + nbdkit as a cache of encrypted blocks.

This makes it possible for the host to manage the associated disk space more efficiently, e.g. share blocks across pods using the same image.

The underlying host-guest relationship becomes closer to what exists in regular kata, but using virtio-blk instead of virtiofs.

Access Control

We just need to rewrite the Kubernetes access-control model from scratch

For every problem, there is a solution that is simple, clean and wrong

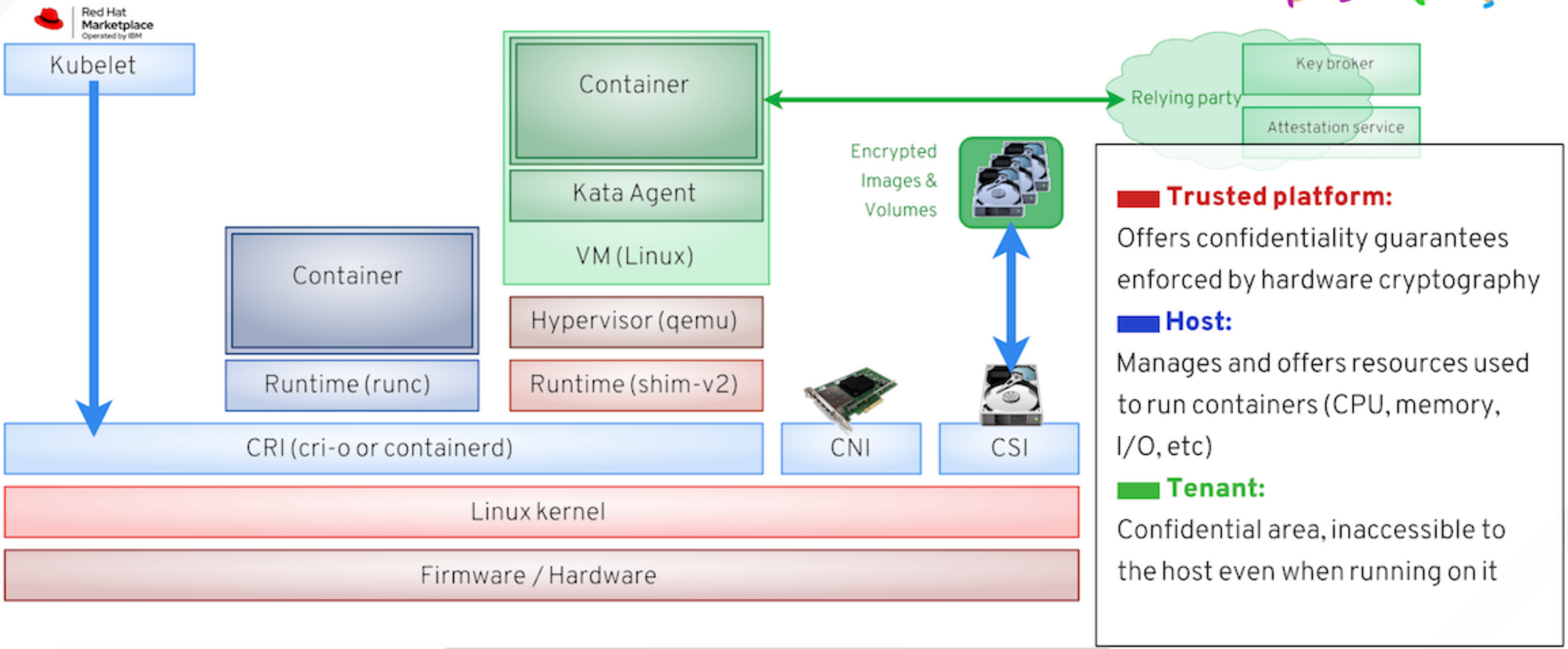
Henry Louis Mencken

Access Control

We just need to rewrite the Kubernetes access-control model from scratch

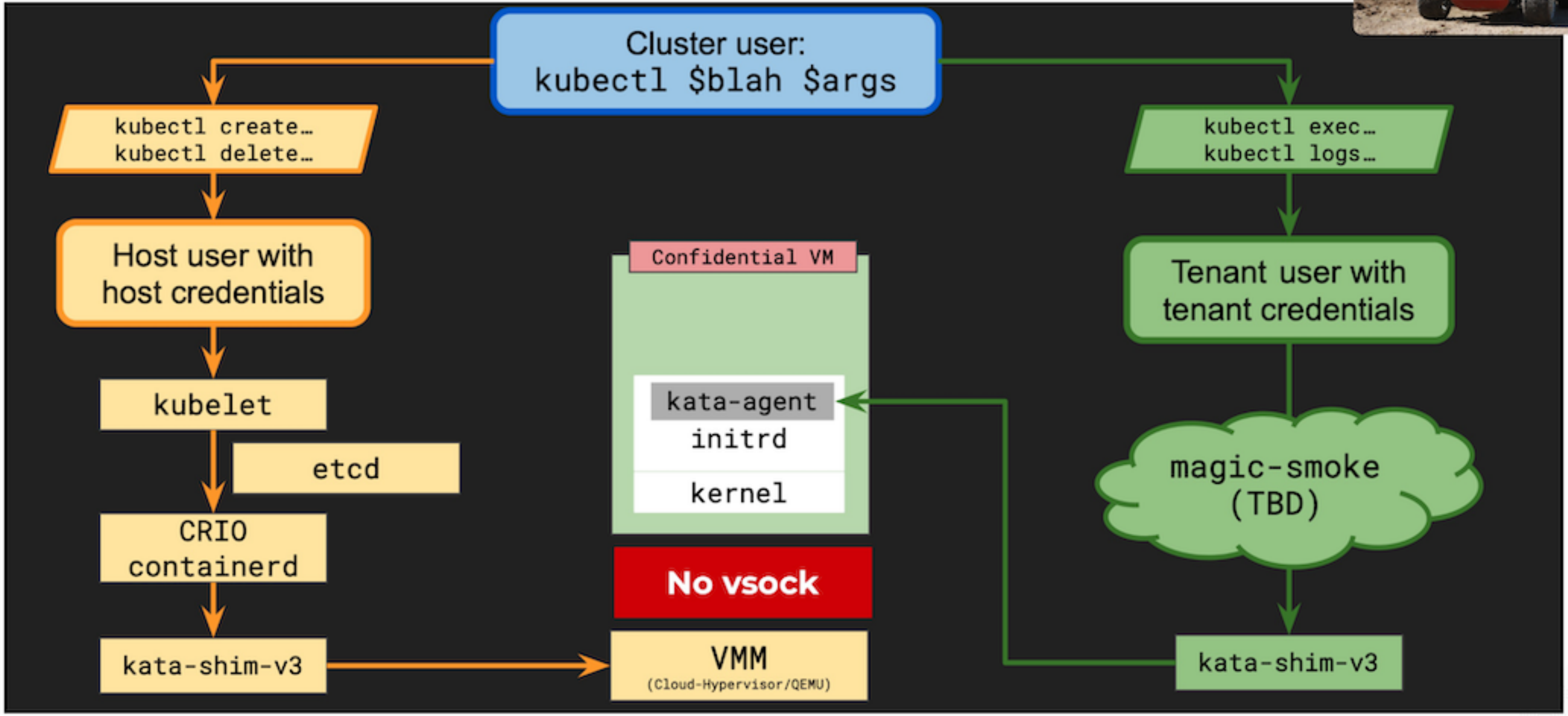
Going from two to three

It's the beginning of counting, folks



The host/tenant split API

Let's just decide which credentials we use on the fly



Debuggability

The FBI school of debugging

KVM FORUM

Debuggability

The FBI school of debugging

A nation can't solve
what the press won't let it perceive

Julian Assange

Red Hat

Tell us what you need to know

and we'll teach you why you don't need it


```
[    0.000000] Initializing [redacted]
[    0.000000] Initializing [redacted]
[    0.000000] Initializing [redacted]
[    0.000000] Linux version 3.[redacted]([redacted].org) (gcc version [redacted]
SMP Thu [redacted] 2018
[    0.000000] Command line: BOOT_[redacted]root=/dev/[redacted]
[redacted]quiet LANG=en_US.UTF-8
[    0.0[redacted] Disabled [redacted]
[    0.0[redacted] [redacted]BIOS-provided physical RAM map:
[    0.0[redacted] BIOS-[redacted]usable
[    0.0[redacted] BIOS-[redacted]
[    0.0[redacted] BIOS-[redacted]
[    0.0[redacted] BIOS-[redacted]usable
[    0.0[redacted] BIOS-[redacted]
[    0.0[redacted] BIOS-[redacted]
[    0.0[redacted] BIOS-[redacted]
[    0.0[redacted] BIOS-[redacted]
[    0.0[redacted] BIOS-[redacted]
[    0.0[redacted] BIOS-[redacted]
[    0.0[redacted] BIOS-[redacted]
[    0.0[redacted] NX (Execute Disable) protection: [redacted]
[    0.0[redacted] SMBIOS [redacted].
[    0.0[redacted] DMI: [redacted], BIOS [redacted]
[    0.0[redacted]
[    0.0[redacted]
[    0.0[redacted]
[    0.0[redacted]ns
[    0.0[redacted]
[    0.0[redacted]usable
[    0.0[redacted]0x400000000
[    0.0[redacted]type:[redacted]
[    0.0[redacted]
[    0.0[redacted]write-back
[    0.0[redacted]uncachable
[    0.0[redacted]
[    0.0[redacted]uncachable
[    0.0[redacted]
[    0.0[redacted]
[    0.0[redacted]
[    0.0[redacted]
```

Tell us what you need to know

and we'll teach you why you don't need it

It's not a suggestion, it's a rule

Really, we mean it

if you `panic()`, we offer a cryptographic, hardware-enforced guarantee that you that you won't see the logs

Any good ideas on how to address this in a sensible manner is welcome

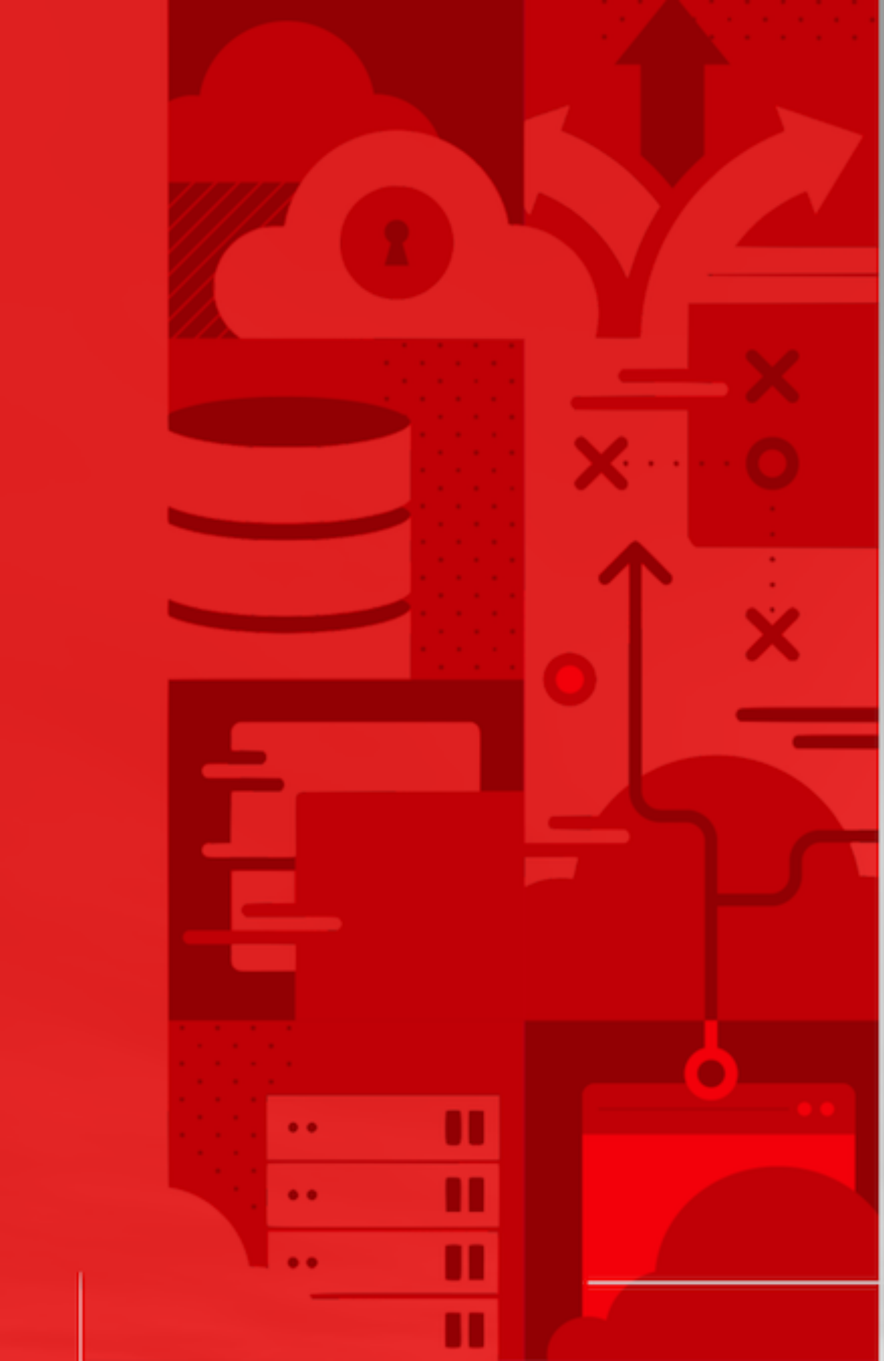
Conclusion

Confidential Containers are an opportunity but a real challenge in terms of everything

Key takeaways

High cost of confidentiality and sandboxing

- Relase 1 of confidential containers is around the corner
- Practical, portable, uniform attestation is a challenge
- Performance issues are mostly vastly enhanced resource usage
- Image download is a huge contributor to this
- Access control has to be re-thought (not in release 1)
- Debuggability or confidentiality, pick one



KVM FORUM

CONFIDENTIAL CONTAINERS

Thank you

Now is a good time for questions

This Tao3D presentation is available at
https://github.com/c3d/presentations
(branch kvm-forum-2022)

Red Hat

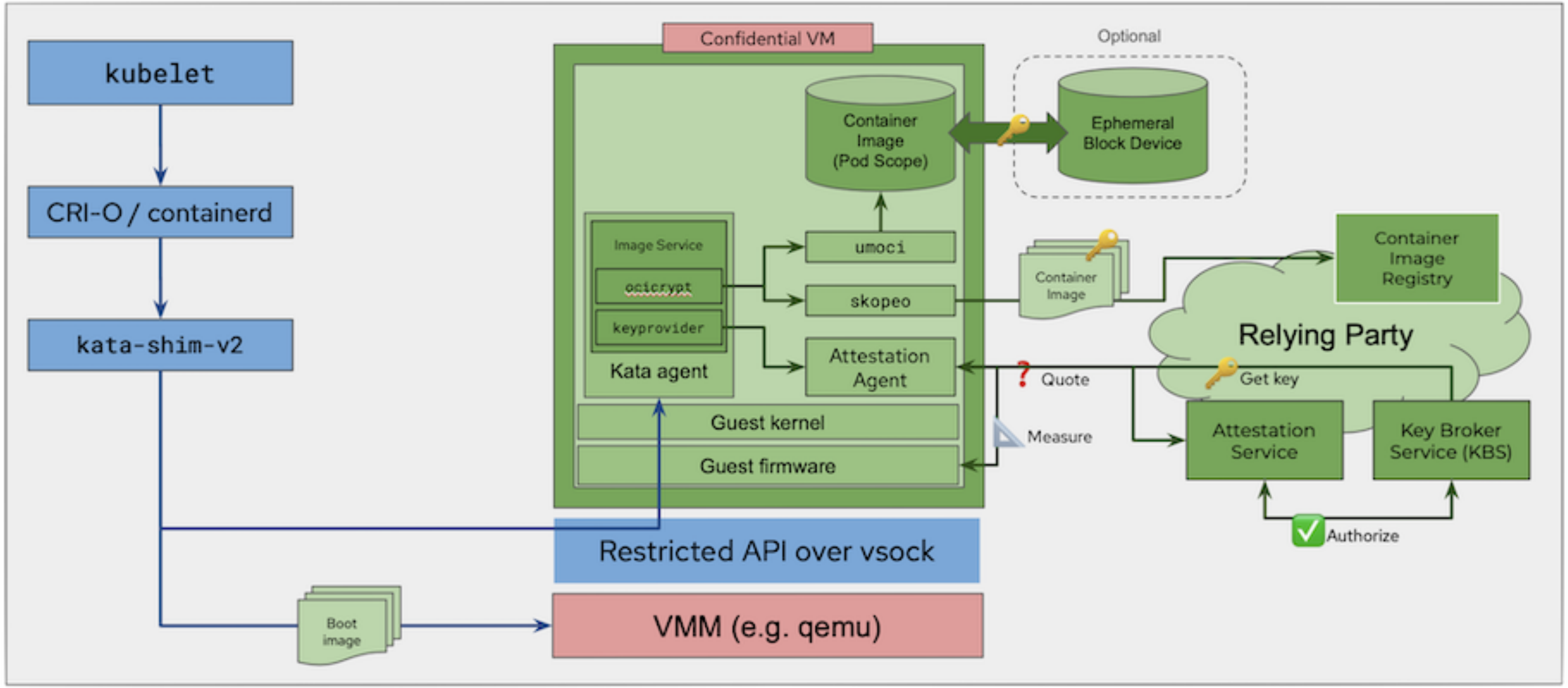
Overflow sides

https://github.com/c3d/presentations

This Tao3D presentation is available at
https://github.com/c3d/presentations
(branch kvm-forum-2022)

Attestation process

Exchanging keys with the relying party



Immutable Pods

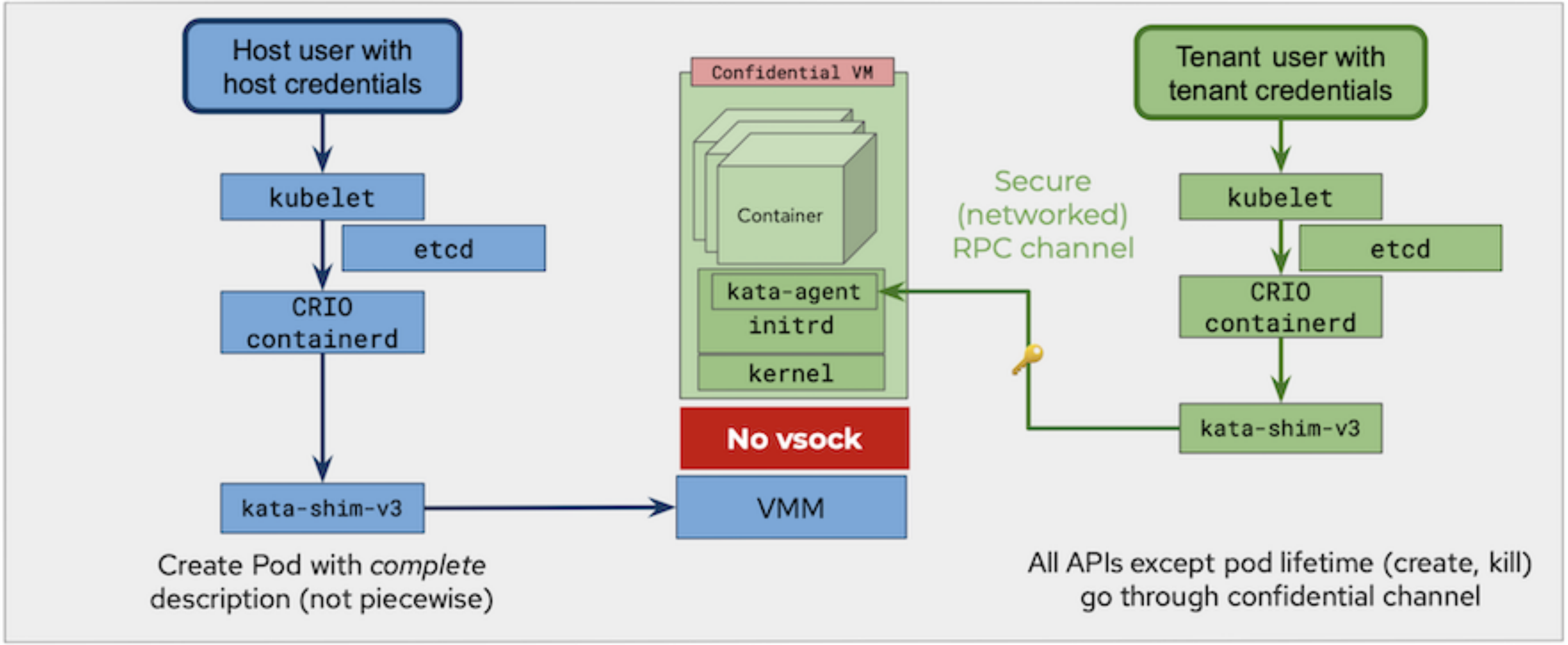
You cannot change what you certified

- **Hot Plugging** is currently used to add memory, CPU or devices to the pod
  - The Pod APIs do not give us the information about container sizes
  - Resources are dynamically added at container creation time
  - This adds a lot of complexity to the runtime, and inefficiency (e.g. fat page tables)
- **Integrity** is hard to guarantee if you can change the configuration at runtime
  - Memory hot-plugging or ballooning mechanisms conflict with encryption / validation
  - Devices, notably pass-through PCI devices with DMA, are also problematic
- **Immutable Pods** are fully defined ahead of time, before booting the virtual machine
  - This requires many changes in the existing Kubernetes APIs
  - Existing APIs may put things "in the wrong place", e.g. send logs to the *host*.
  - This will simplify and optimize the non-confidential case, e.g. remove hot-plugging

Two Control Planes

Host and Tenant security realms

- **Tenants** need their own *isolated* administrative realm (logs, container metrics, ...)
- **Hosts** manage physical resources (pod creation/destruction, raw disks, H/W metrics...)



Performance considerations

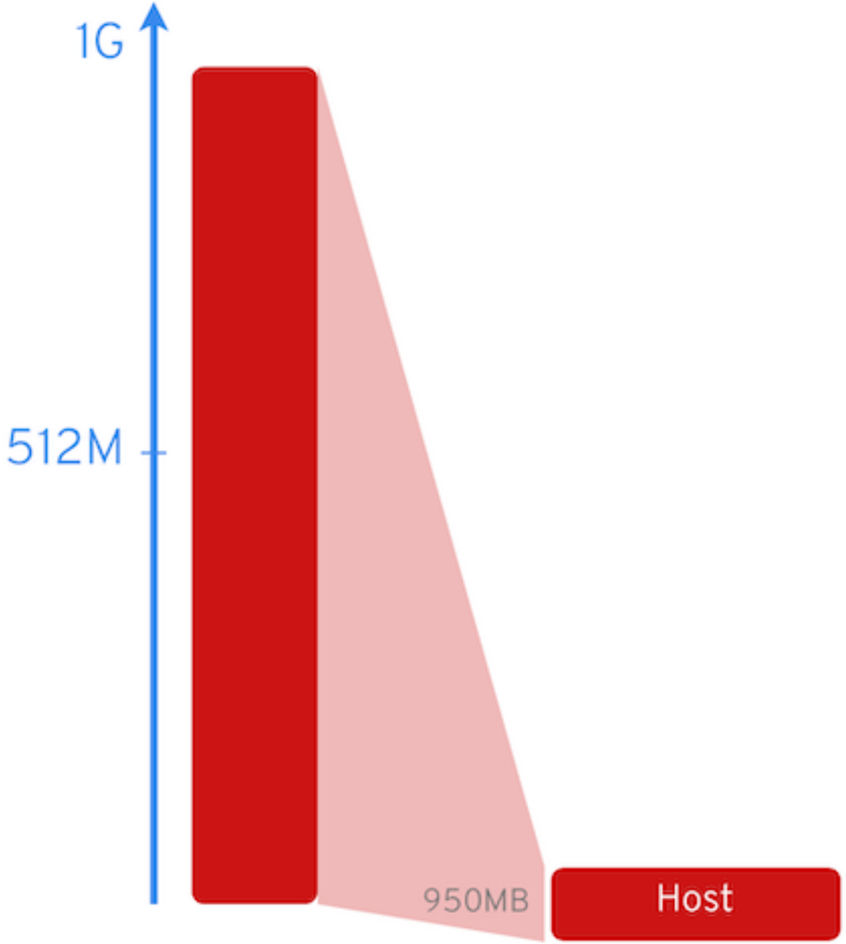
Impact of running virtualized

Memory density

How many containers can you fit?

Memory usage

Death by a thousand cuts

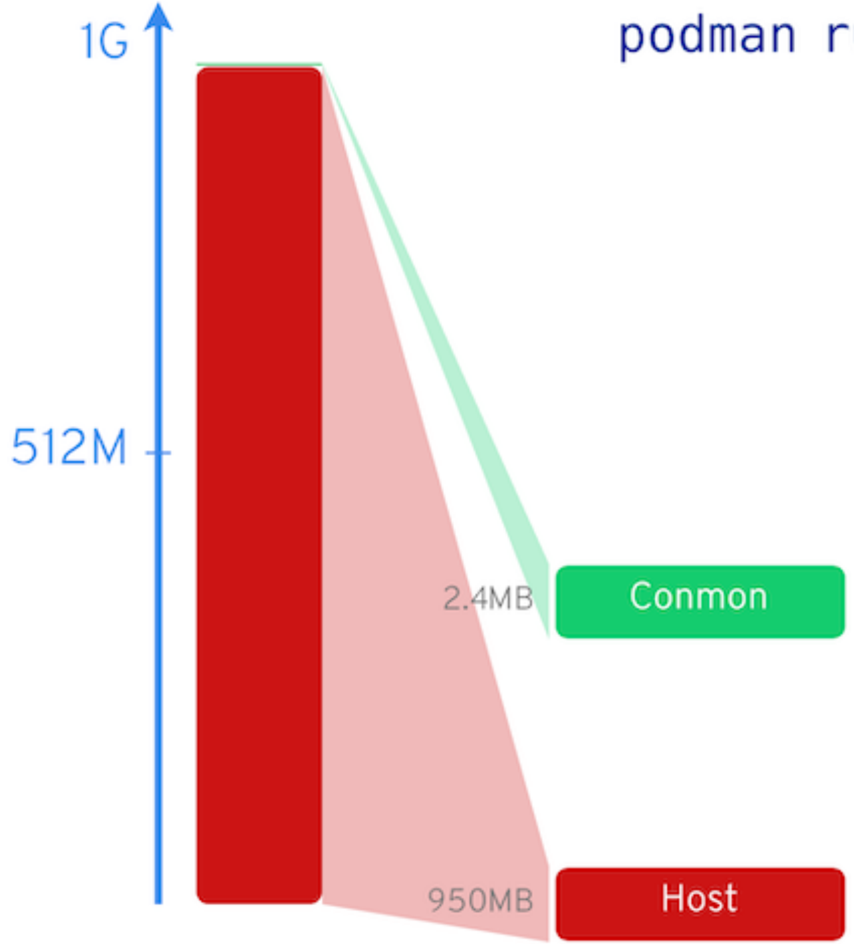


Host kernel and user-space programs

Memory usage

Death by a thousand cuts

`podman run -it --rm fedora bash`



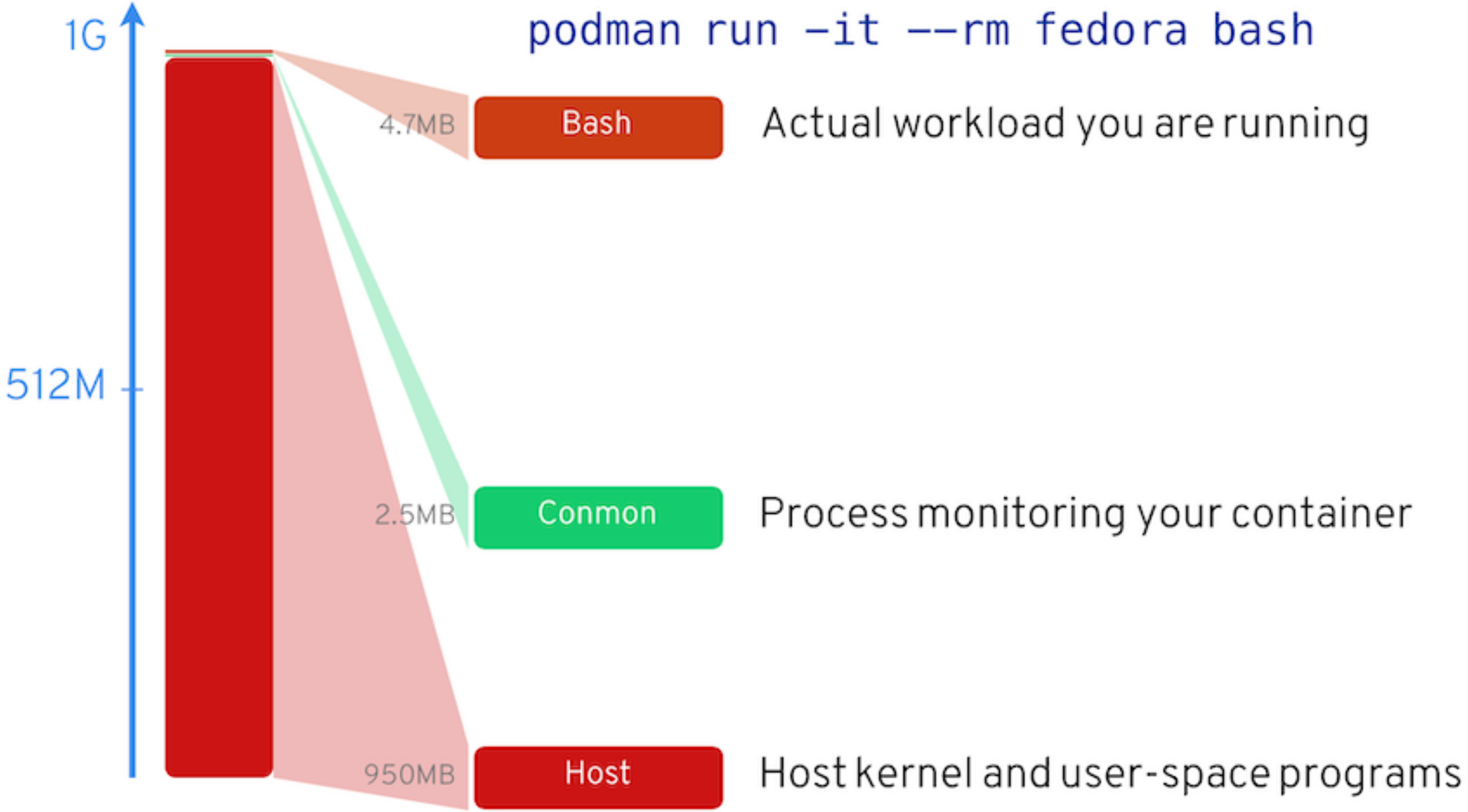
Process monitoring your container

Host kernel and user-space programs

Memory usage

Running bash

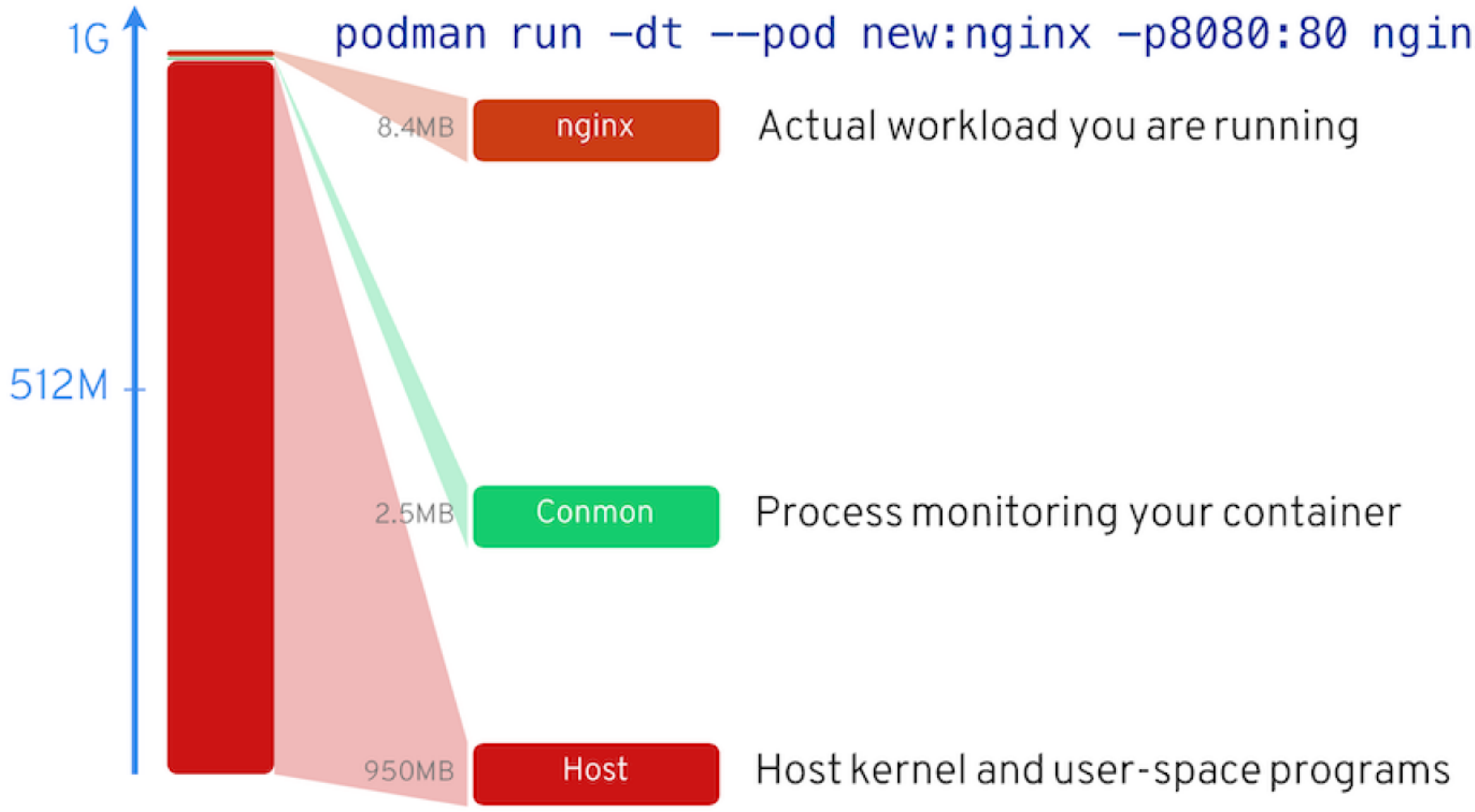
`podman run -it --rm fedora bash`



Memory usage

Running a web server

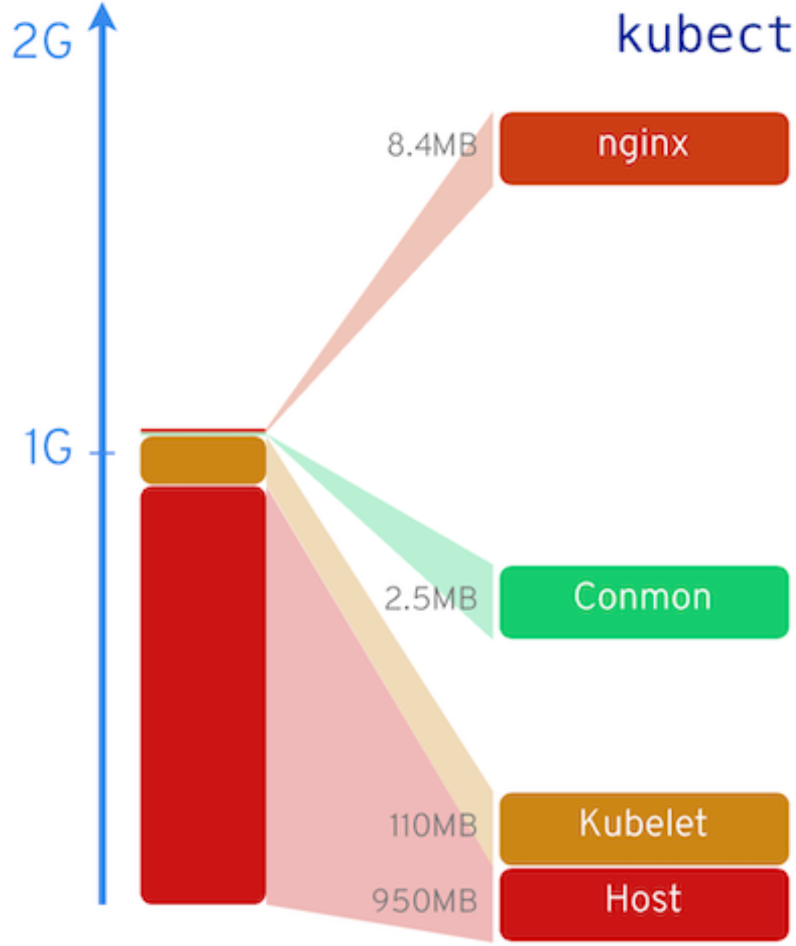
`podman run -dt --pod new:nginx -p8080:80 nginx`



Memory usage

Kubernetes overhead

`kubectl apply -f nginx.yaml`



Actual workload you are running

Process monitoring your container

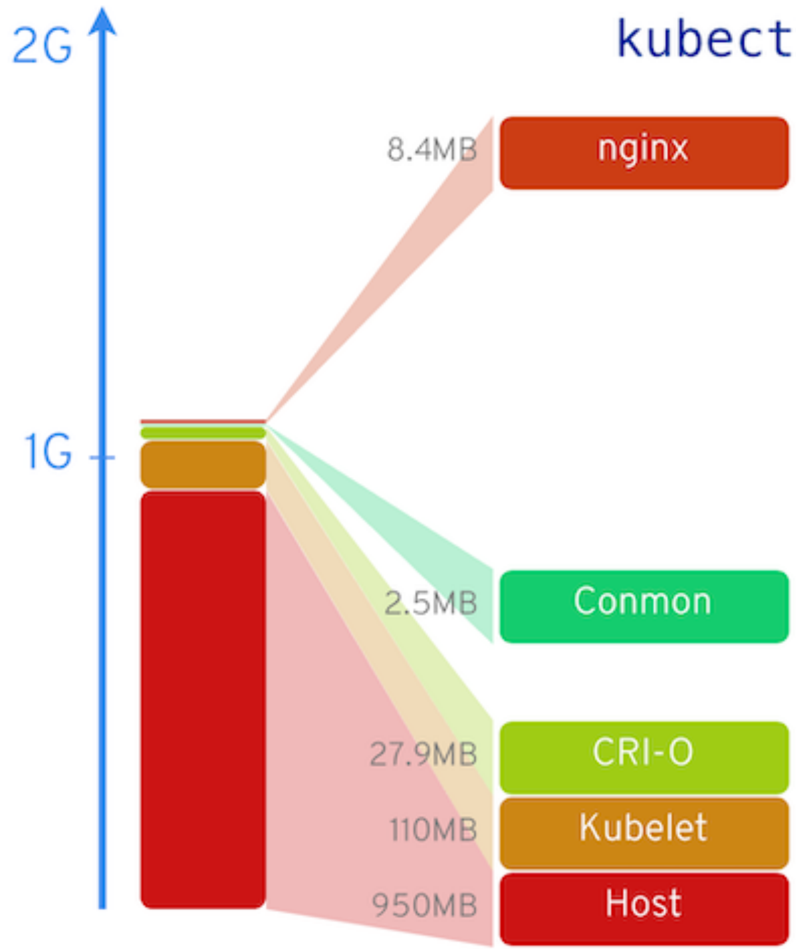
Top-level control process

Host kernel and user-space programs

Memory usage

Kubernetes overhead

`kubectl apply -f nginx.yaml`



Actual workload you are running

Process monitoring your container

Container runtime interface

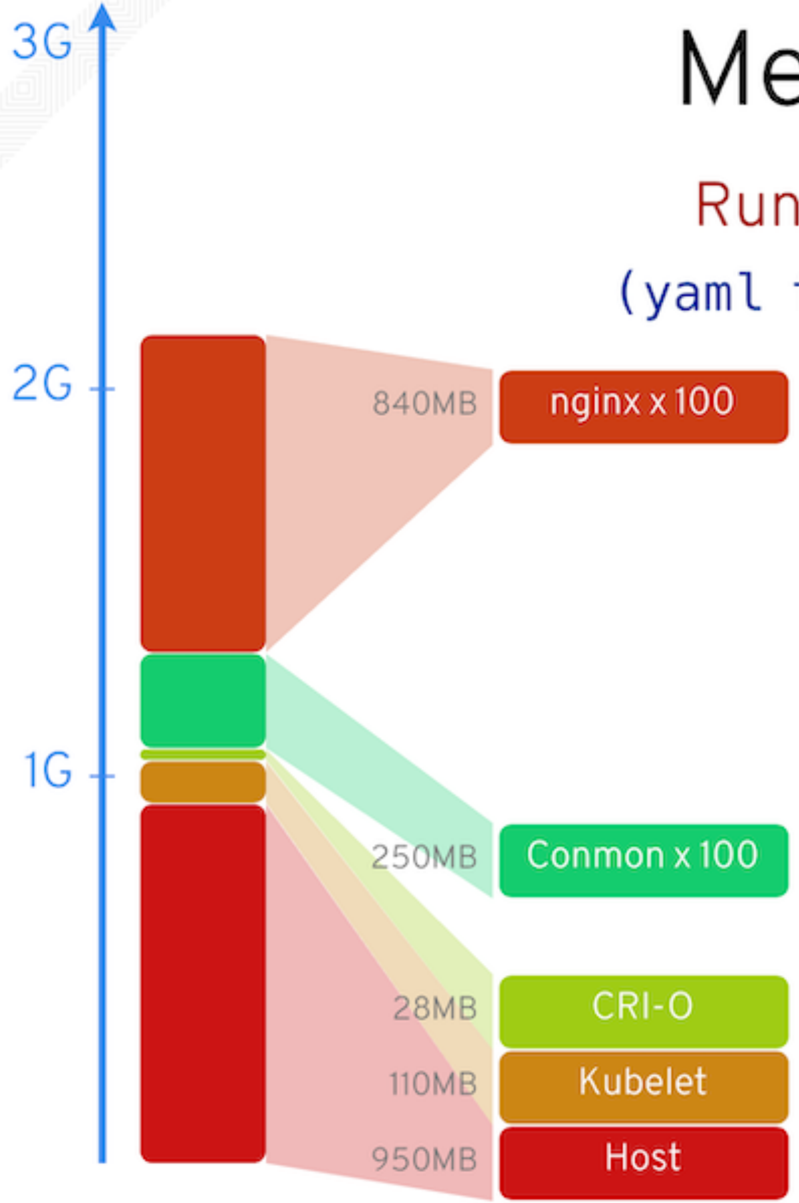
Top-level control process

Host kernel and user-space programs

Memory usage

Running 100 containers

(yaml file with 100x nginx)



Actual workload you are running

Process monitoring your container

Container runtime interface

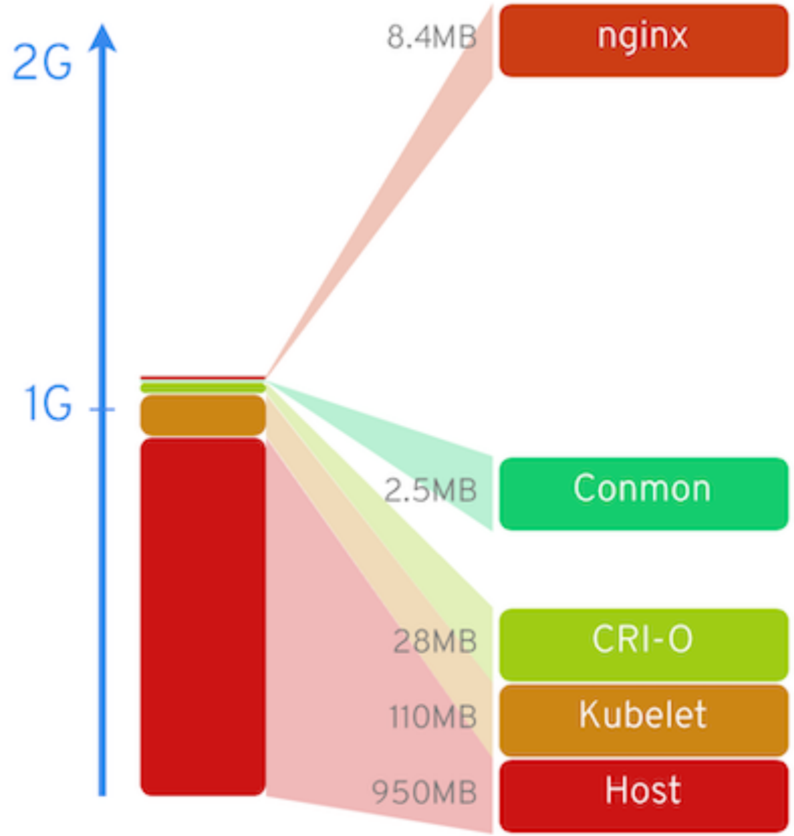
Top-level control process

Host kernel and user-space programs

Memory usage

Overhead varies with number of containers

```
kubectl apply -f nginx.yaml
```

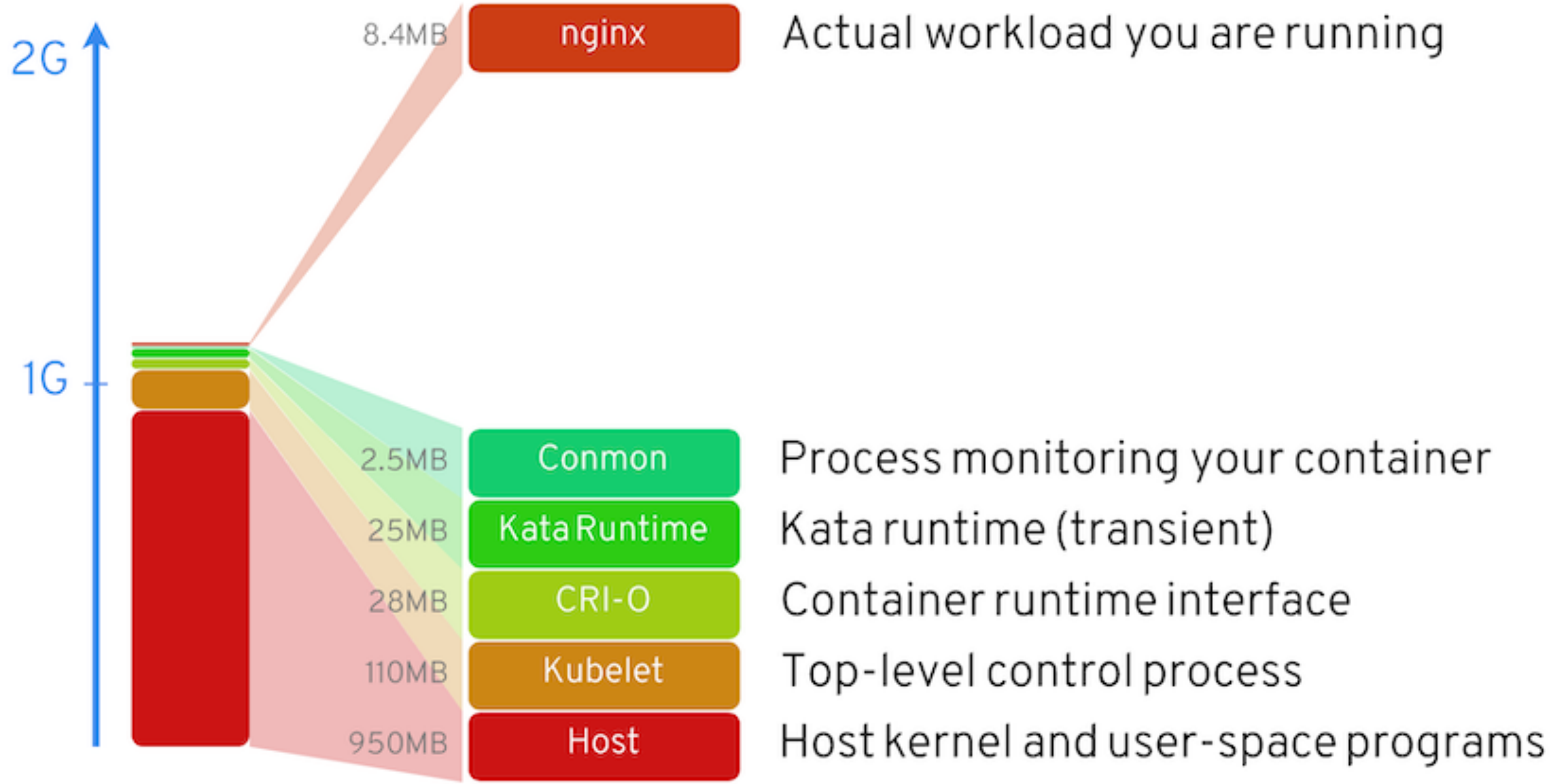


- nginx: Actual workload you are running
- Conmon: Process monitoring your container
- CRI-O: Container runtime interface
- Kubelet: Top-level control process
- Host: Host kernel and user-space programs

Memory usage

Kata adds even more overhead

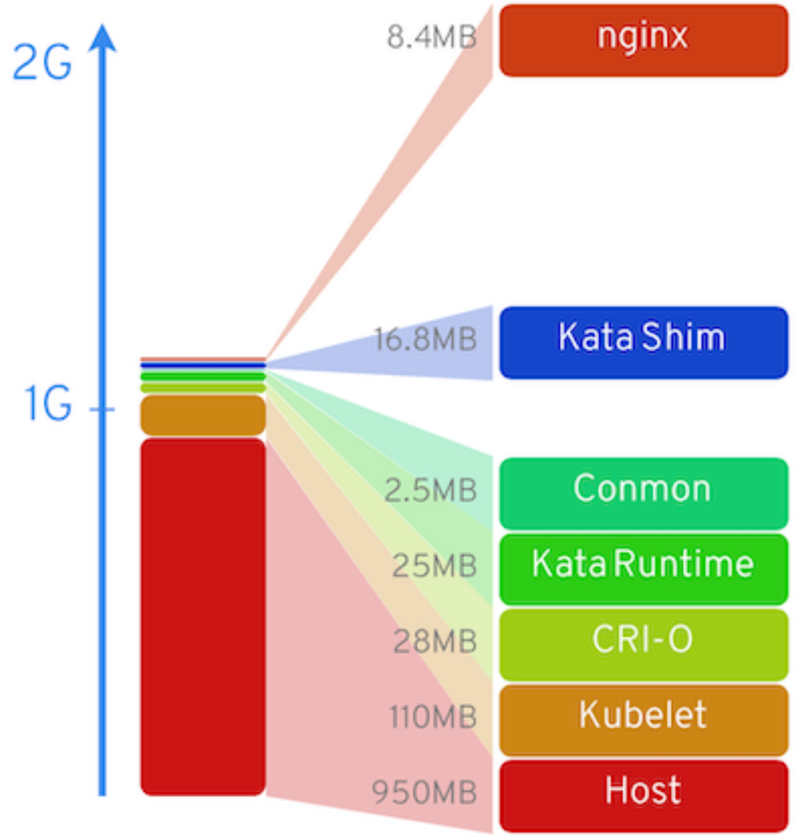
```
podman run -it --rm --runtime kata fedora bash
```



Memory usage

Kata adds even more overhead

```
podman run -it --rm --runtime kata fedora bash
```

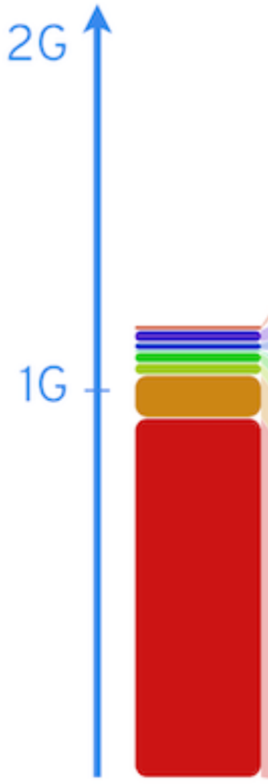


- Actual workload you are running
- I/O interface for container
- Process monitoring your container
- Kata runtime (transient)
- Container runtime interface
- Top-level control process
- Host kernel and user-space programs

Memory usage

Kata adds even more overhead

```
podman run -it --rm --runtime kata fedora bash
```

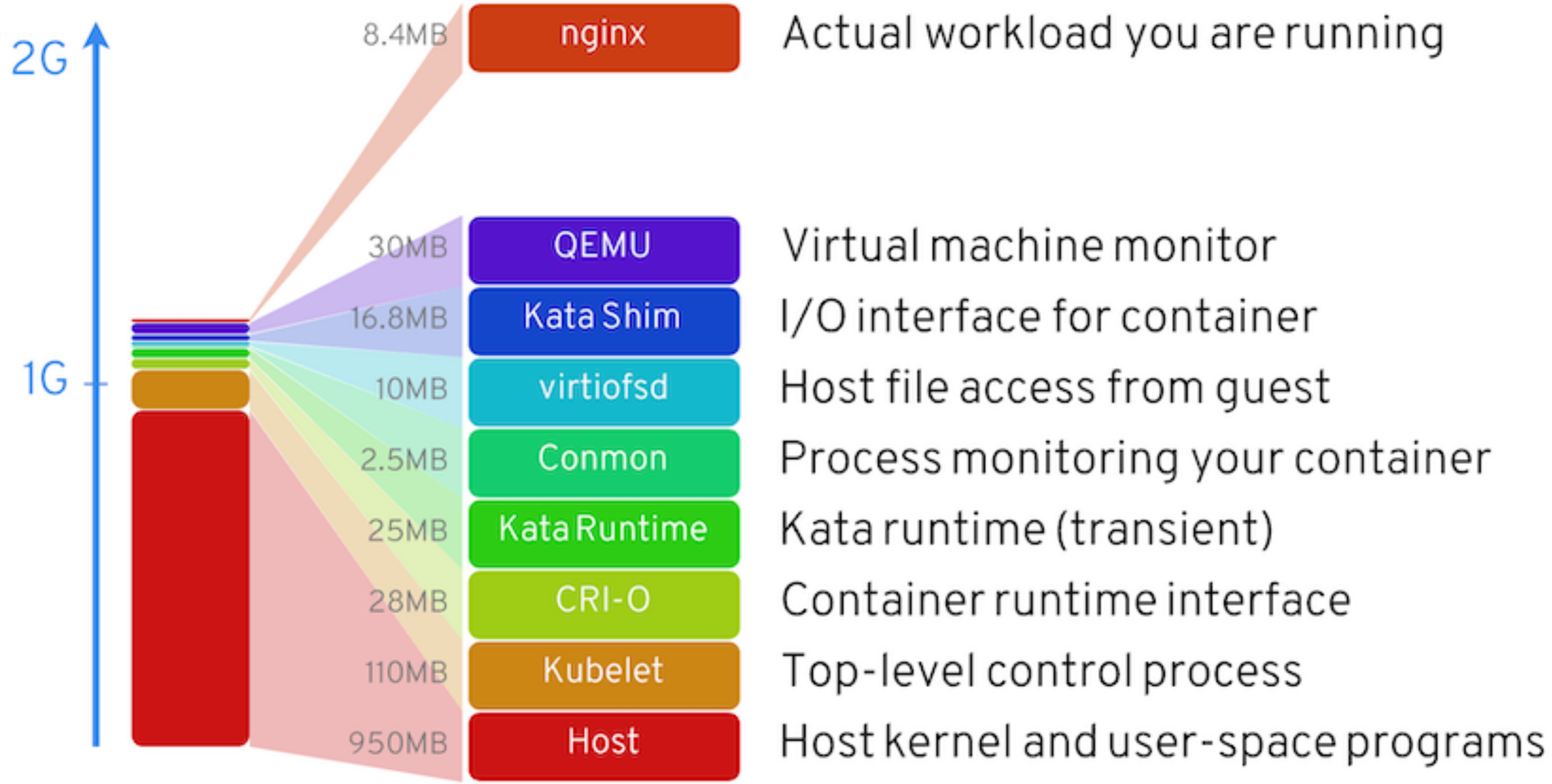


Size	Component	Description
8.4MB	nginx	Actual workload you are running
30MB	QEMU	Virtual machine monitor
16.8MB	Kata Shim	I/O interface for container
2.5MB	Conmon	Process monitoring your container
25MB	Kata Runtime	Kata runtime (transient)
28MB	CRI-O	Container runtime interface
110MB	Kubelet	Top-level control process
950MB	Host	Host kernel and user-space programs

Memory usage

Kata adds even more overhead

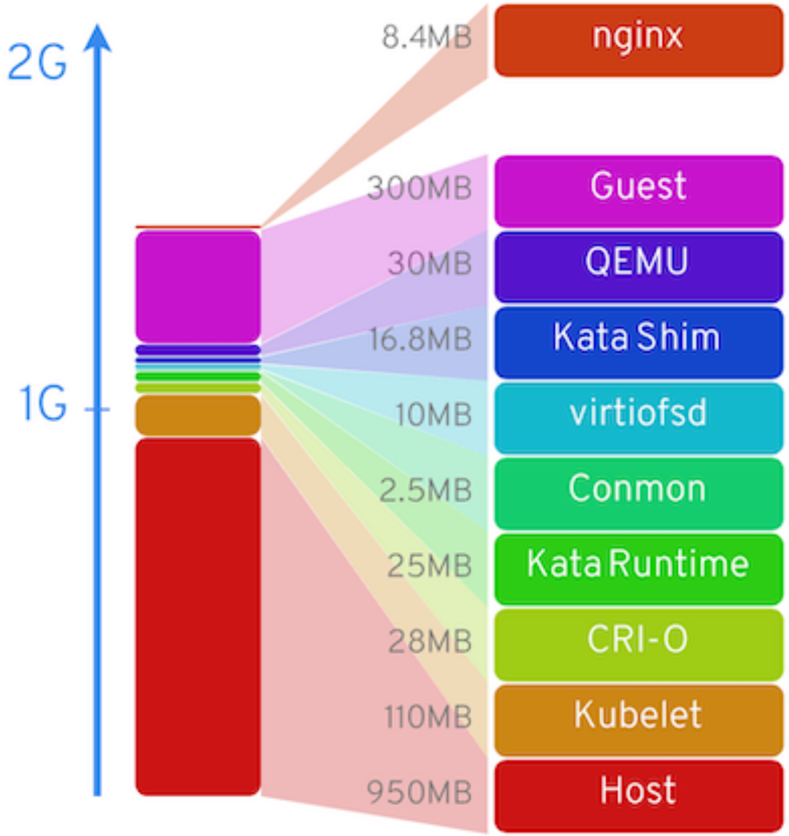
```
podman run -it --rm --runtime kata fedora bash
```



Memory usage

Kata adds even more overhead

```
podman run -it --rm --runtime kata fedora bash
```



Actual workload you are running

Guest OS kernel, data, user-space

Virtual machine monitor

I/O interface for container

Host file access from guest

Process monitoring your container

Kata runtime (transient)

Container runtime interface

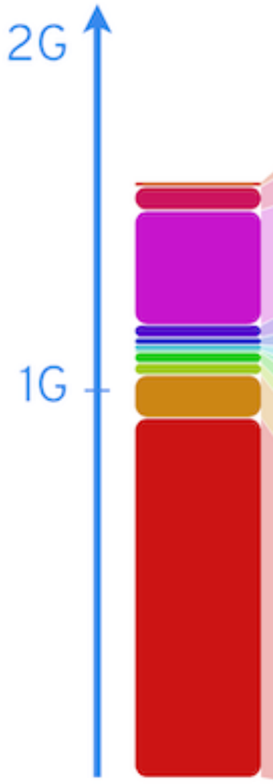
Top-level control process

Host kernel and user-space programs

Memory usage

Kata adds even more overhead

```
podman run -it --rm --runtime kata fedora bash
```

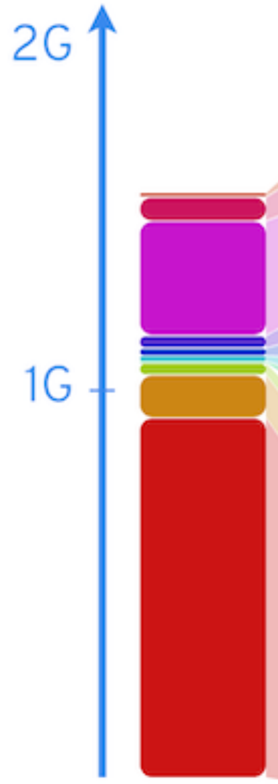


Size	Component	Description
8.4MB	nginx	Actual workload you are running
60MB	Agent	Kata agent in the guest
300MB	Guest	Guest OS kernel, data, user-space
30MB	QEMU	Virtual machine monitor
16.8MB	Kata Shim	I/O interface for container
10MB	virtiofsd	Host file access from guest
2.5MB	Conmon	Process monitoring your container
25MB	Kata Runtime	Kata runtime (transient)
28MB	CRI-O	Container runtime interface
110MB	Kubelet	Top-level control process
950MB	Host	Host kernel and user-space programs

Memory usage

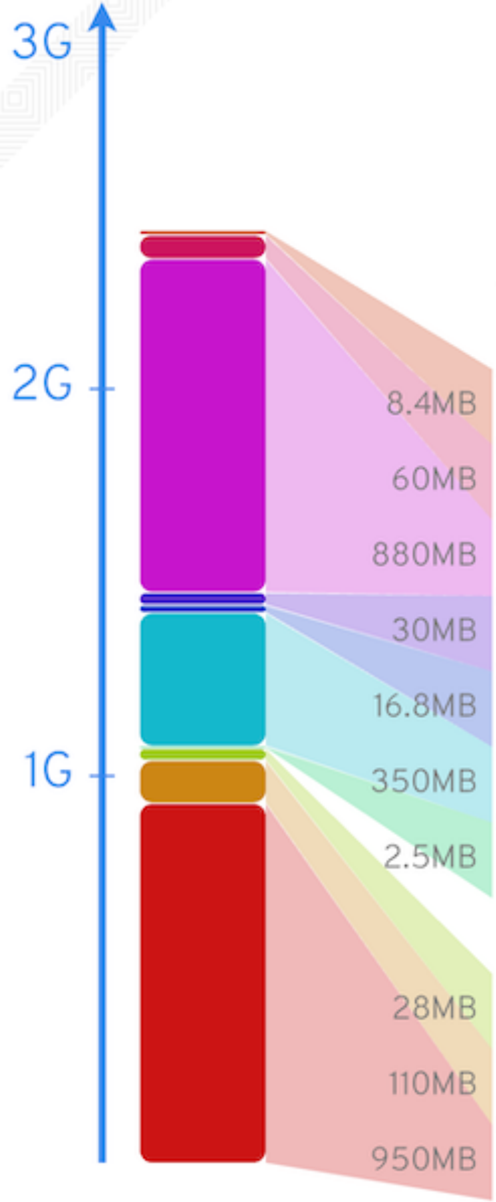
Kata Runtime is transient

`podman run -it --rm --runtime kata fedora bash`



Size	Component	Description
8.4MB	nginx	Actual workload you are running
60MB	Agent	Kata agent in the guest
300MB	Guest	Guest OS kernel, data, user-space
30MB	QEMU	Virtual machine monitor
16.8MB	Kata Shim	I/O interface for container
10MB	virtiofsd	Host file access from guest
2.5MB	Conmon	Process monitoring your container
28MB	CRI-O	Container runtime interface
110MB	Kubelet	Top-level control process
950MB	Host	Host kernel and user-space programs

Memory usage



I/Os consume memory (buffers, etc)

```
fedc07a13# dnf install -y procps-ng
```

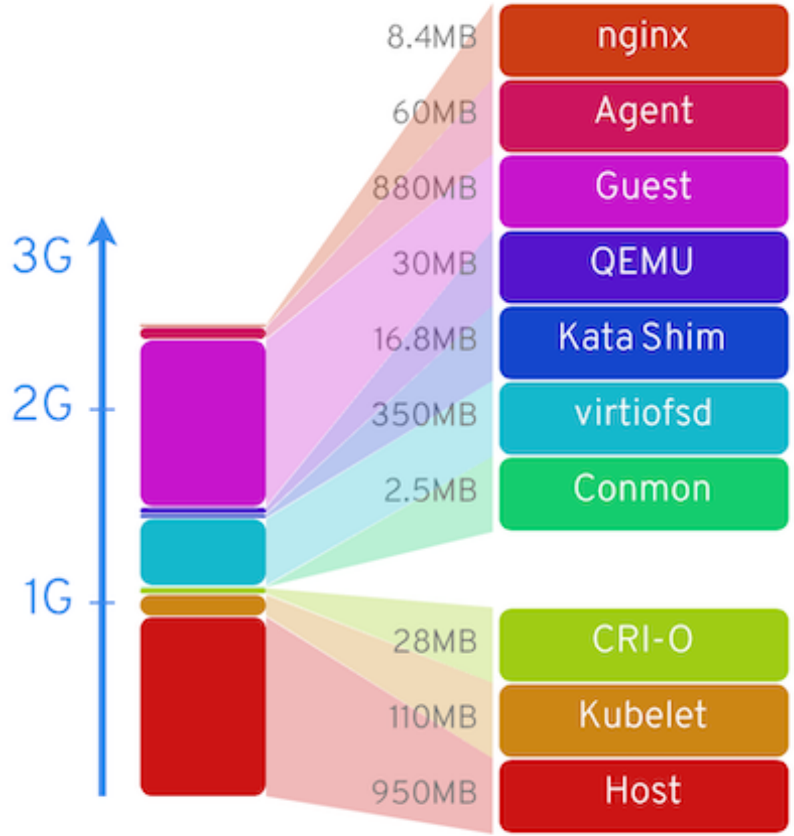
8.4MB	nginx	Actual workload you are running
60MB	Agent	Kata agent in the guest
880MB	Guest	Guest OS kernel, data, user-space
30MB	QEMU	Virtual machine monitor
16.8MB	Kata Shim	I/O interface for container
350MB	virtiofsd	Host file access from guest
2.5MB	Conmon	Process monitoring your container
28MB	CRI-O	Container runtime interface
110MB	Kubelet	Top-level control process
950MB	Host	Host kernel and user-space programs

Possible double accounting of the same physical memory between guest and virtiofsd

Memory usage

Running 100 containers

`(runtimeClass kata, 100x nginx)`



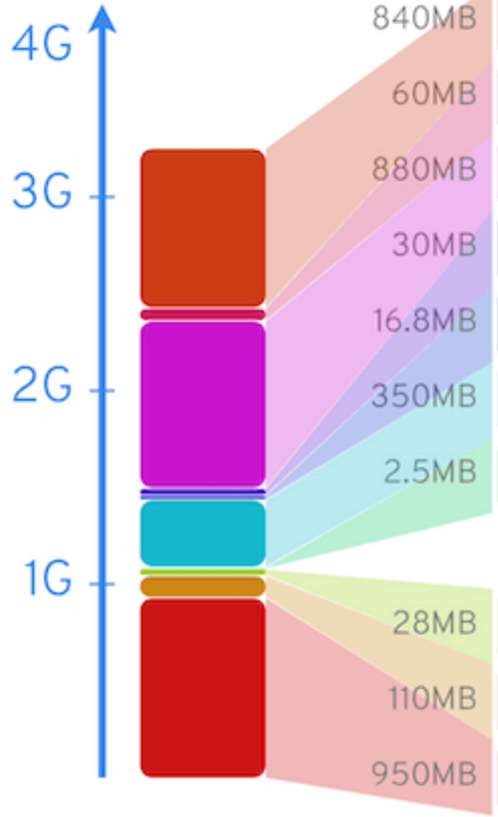
- Actual workload you are running
- Kata agent in the guest
- Guest OS kernel, data, user-space
- Virtual machine monitor
- I/O interface for container
- Host file access from guest
- Process monitoring your container
- Container runtime interface
- Top-level control process
- Host kernel and user-space programs

Possible double accounting of the same physical memory between guest and virtiofsd

Memory usage

Running 100 containers

`(runtimeClass kata, 100x nginx)`



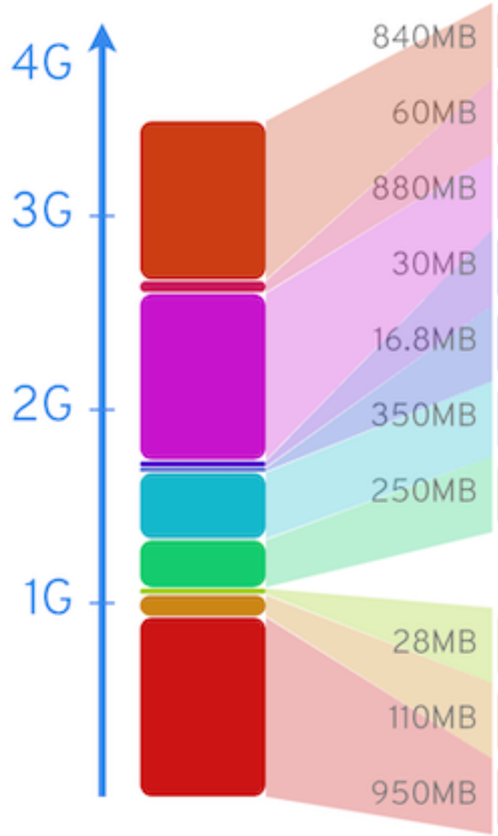
Size	Component	Description
840MB	nginx x 100	Actual workload you are running
60MB	Agent	Kata agent in the guest
880MB	Guest	Guest OS kernel, data, user-space
30MB	QEMU	Virtual machine monitor
16.8MB	Kata Shim	I/O interface for container
350MB	virtiofsd	Host file access from guest
2.5MB	Conmon	Process monitoring your container
28MB	CRI-O	Container runtime interface
110MB	Kubelet	Top-level control process
950MB	Host	Host kernel and user-space programs

Possible double accounting of the same physical memory between guest and virtiofsd

Memory usage

Running 100 containers

`(runtimeClass kata, 100x nginx)`

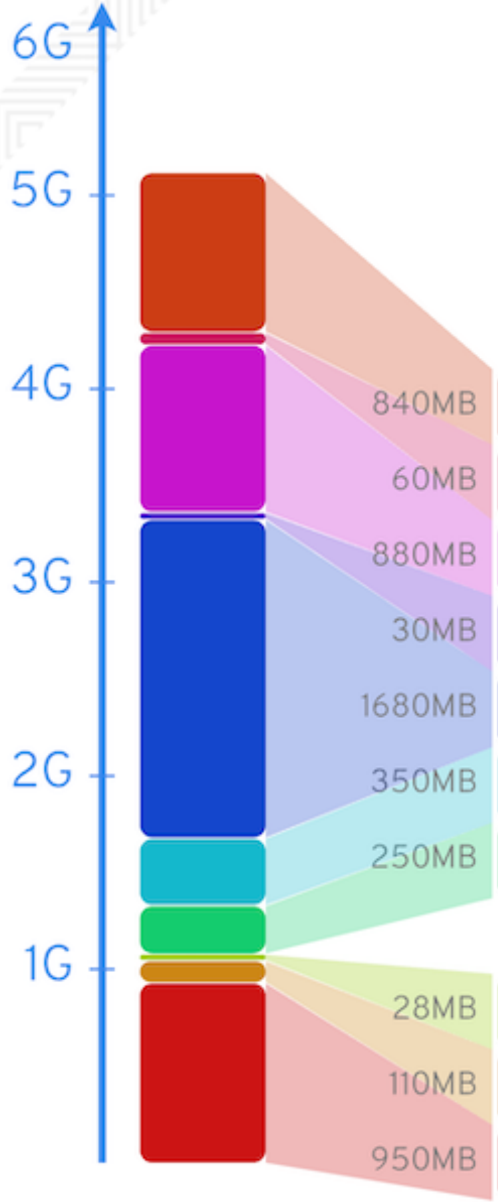


Size	Component	Description
840MB	nginx x 100	Actual workload you are running
60MB	Agent	Kata agent in the guest
880MB	Guest	Guest OS kernel, data, user-space
30MB	QEMU	Virtual machine monitor
16.8MB	Kata Shim	I/O interface for container
350MB	virtiofsd	Host file access from guest
250MB	Conmon x 100	Process monitoring your container
28MB	CRI-O	Container runtime interface
110MB	Kubelet	Top-level control process
950MB	Host	Host kernel and user-space programs

Memory usage

Running 100 containers

`(runtimeClass kata, 100x nginx)`

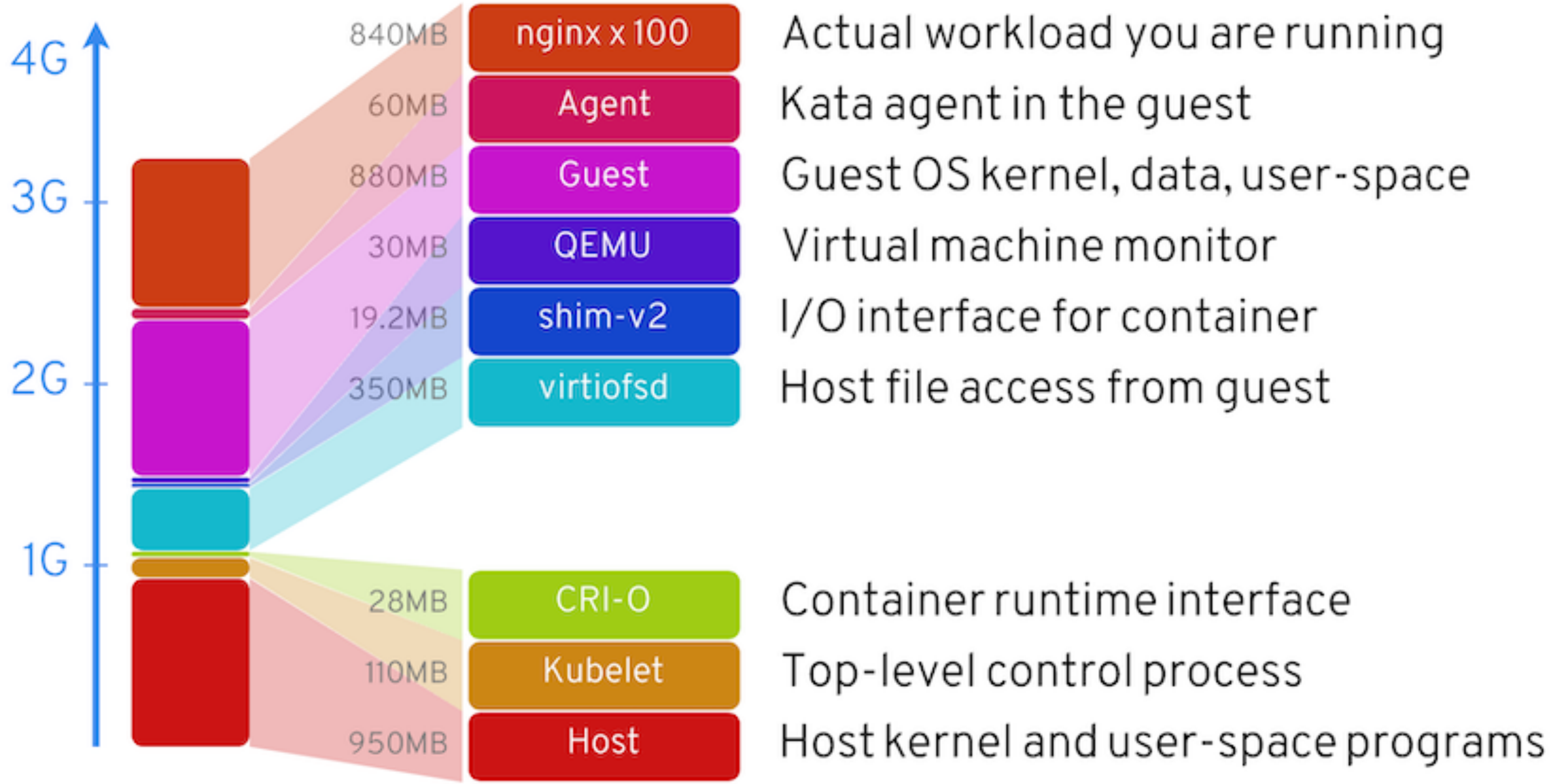


Size	Component	Description
840MB	nginx x 100	Actual workload you are running
60MB	Agent	Kata agent in the guest
880MB	Guest	Guest OS kernel, data, user-space
30MB	QEMU	Virtual machine monitor
1680MB	Kata Shim x100	I/O interface for container
350MB	virtiofsd	Host file access from guest
250MB	Conmon x 100	Process monitoring your container
28MB	CRI-O	Container runtime interface
110MB	Kubelet	Top-level control process
950MB	Host	Host kernel and user-space programs

Memory usage

Kata Containers 2.0: Containerd Shim v2

`containerd-shim-kata-v2`

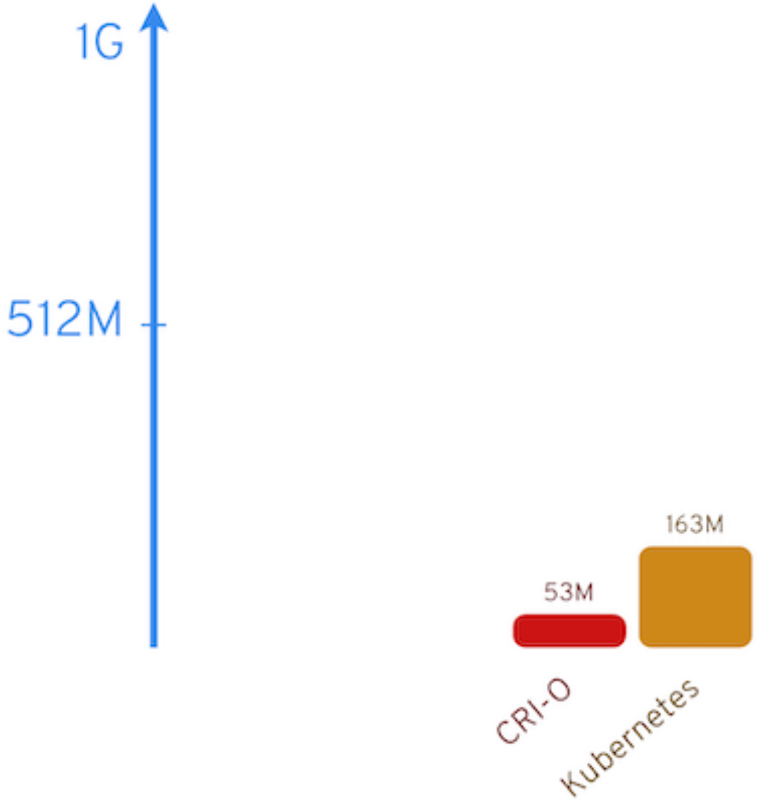


Memory - Comparing runtimes

Cost of the runtime to run 10 instances of nginx

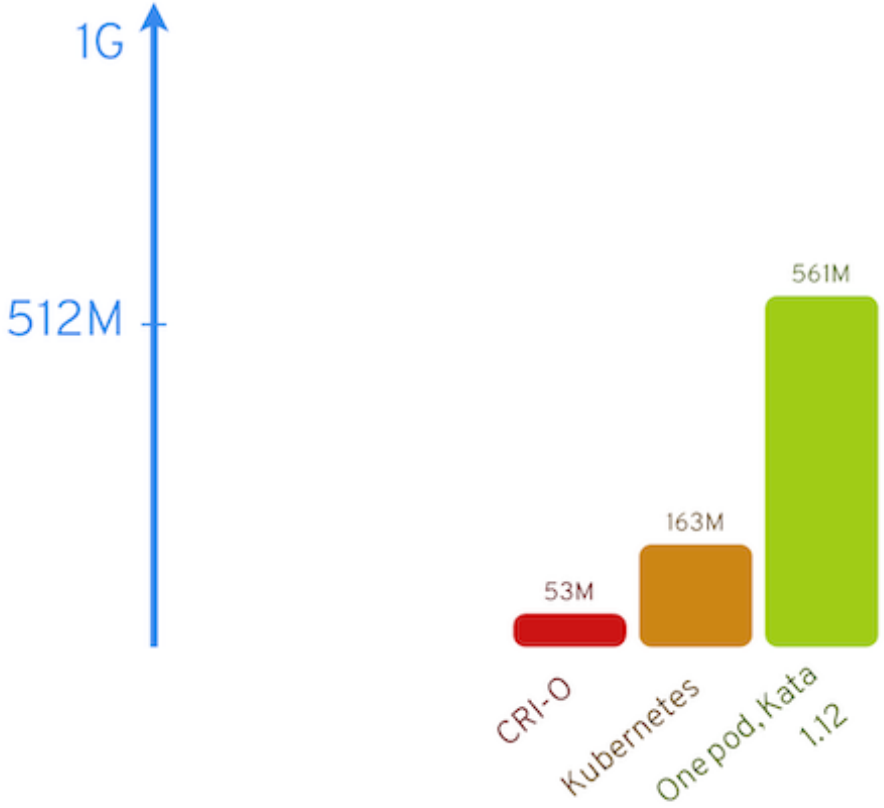
Memory - Comparing runtimes

Cost of the runtime to run 10 instances of nginx



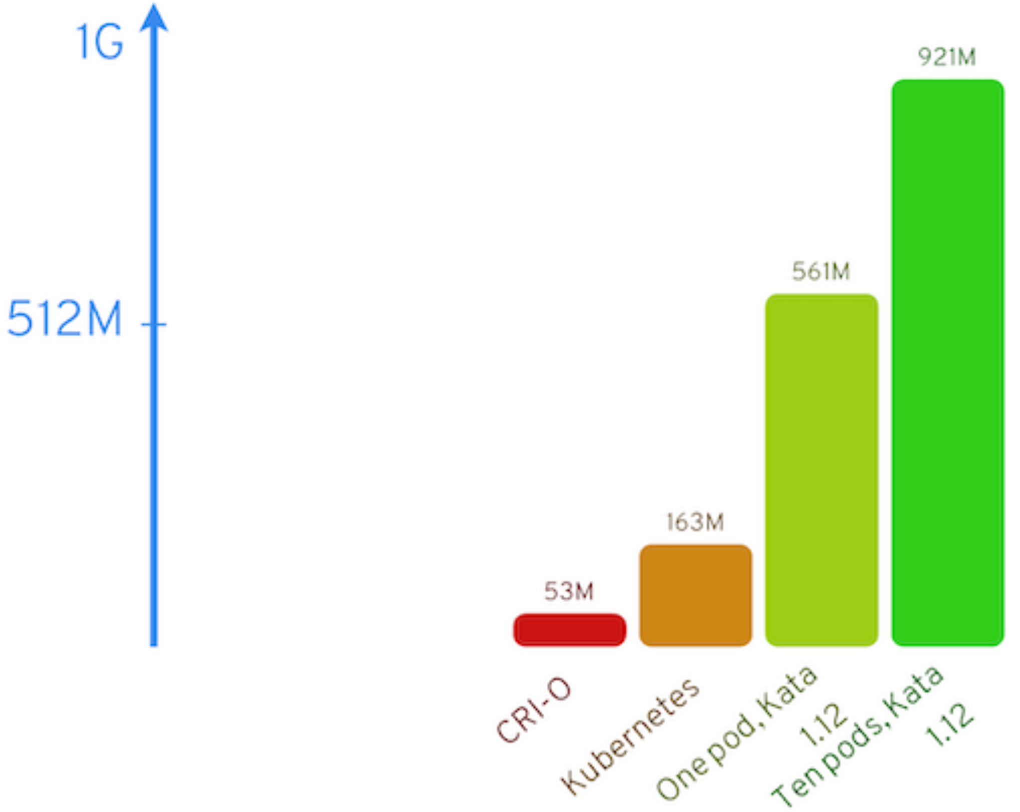
Memory - Comparing runtimes

Cost of the runtime to run 10 instances of nginx



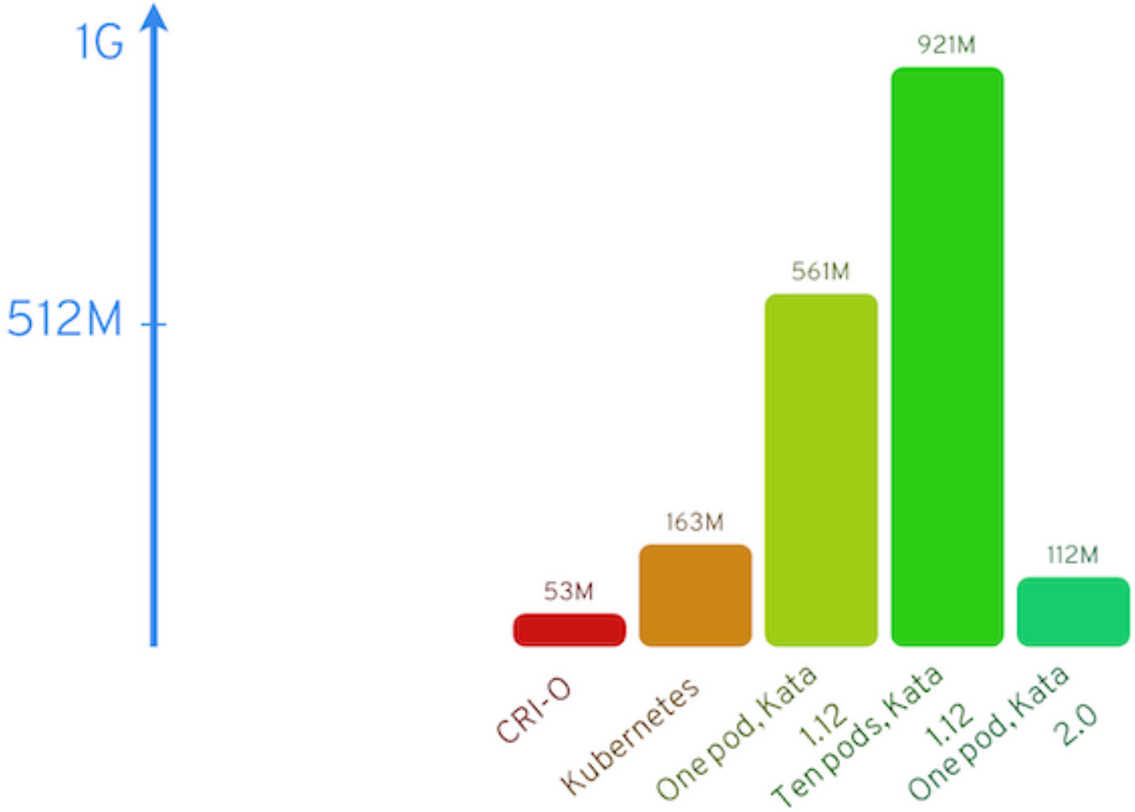
Memory - Comparing runtimes

Cost of the runtime to run 10 instances of nginx



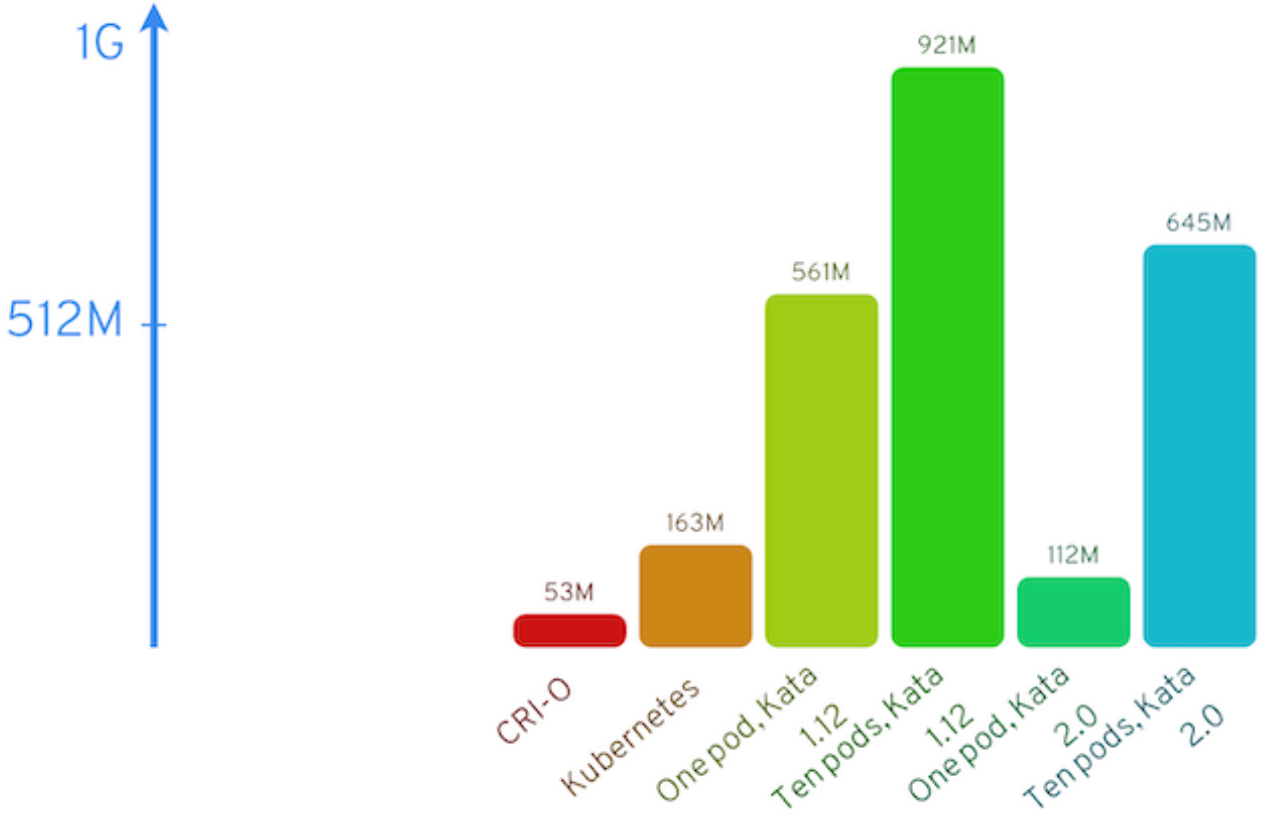
Memory - Comparing runtimes

Cost of the runtime to run 10 instances of nginx



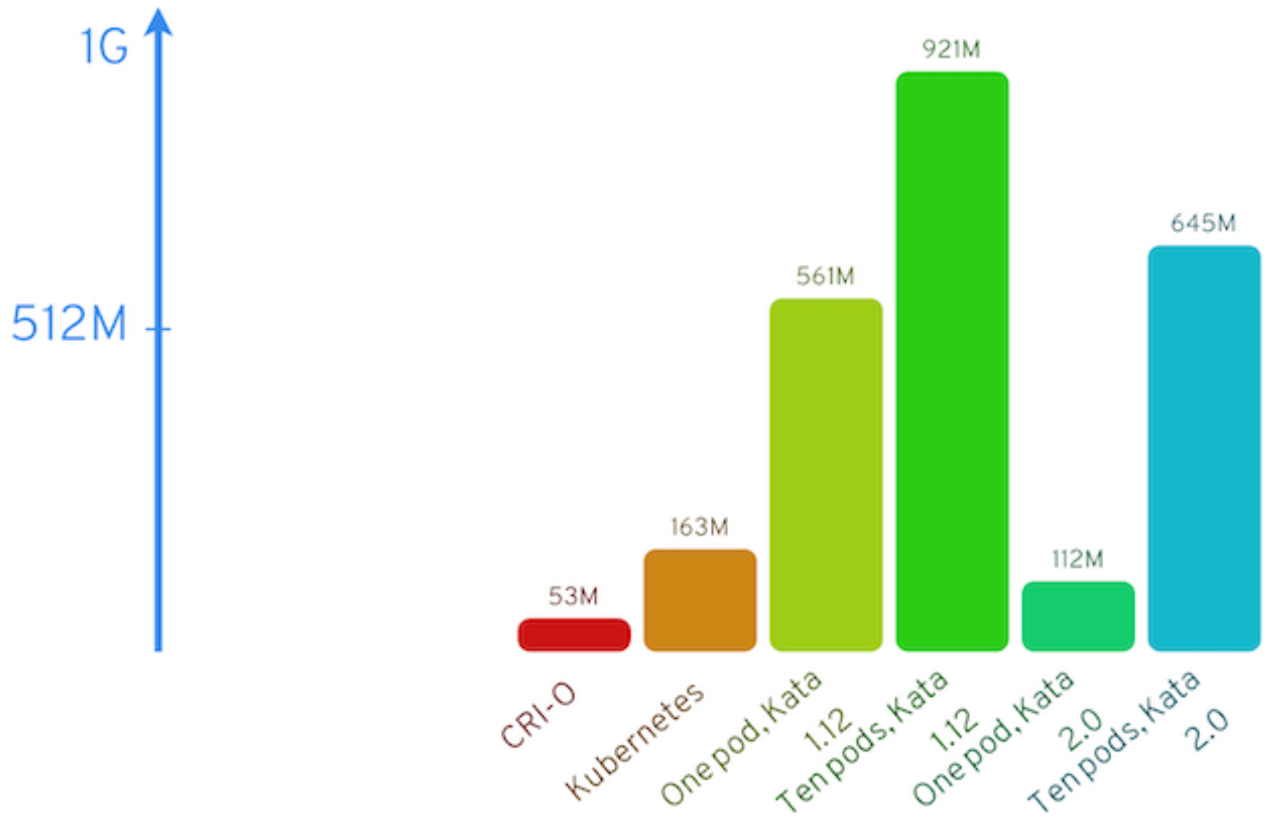
Memory - Comparing runtimes

Cost of the runtime to run 10 instances of nginx



Memory - Comparing runtimes

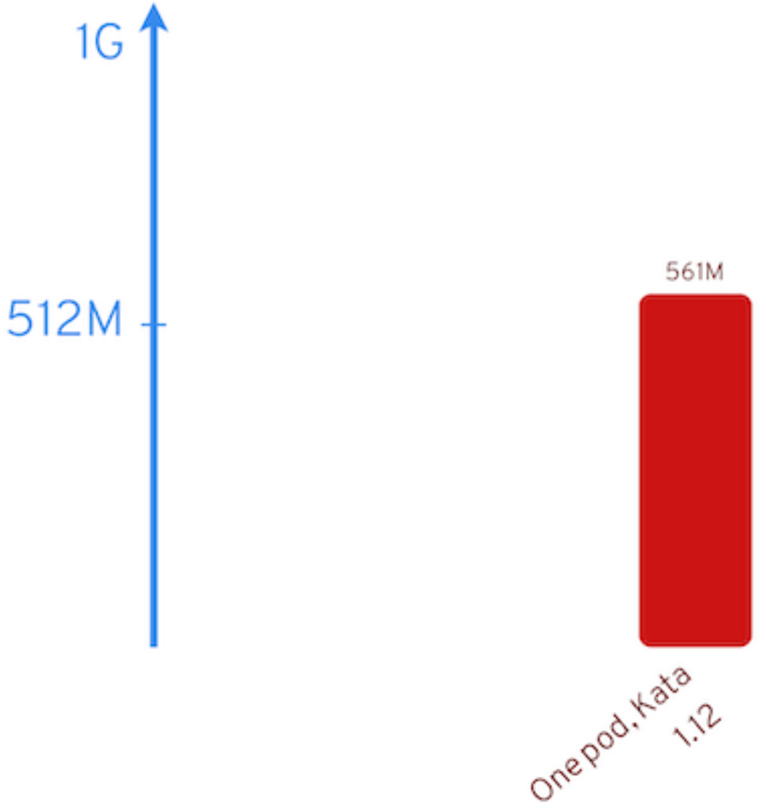
Cost of the runtime to run 10 instances of nginx



- Kata requires a lot of memory for qemu, guest kernel and virtiofs.
- Reusing pods can help, but is not common practice.

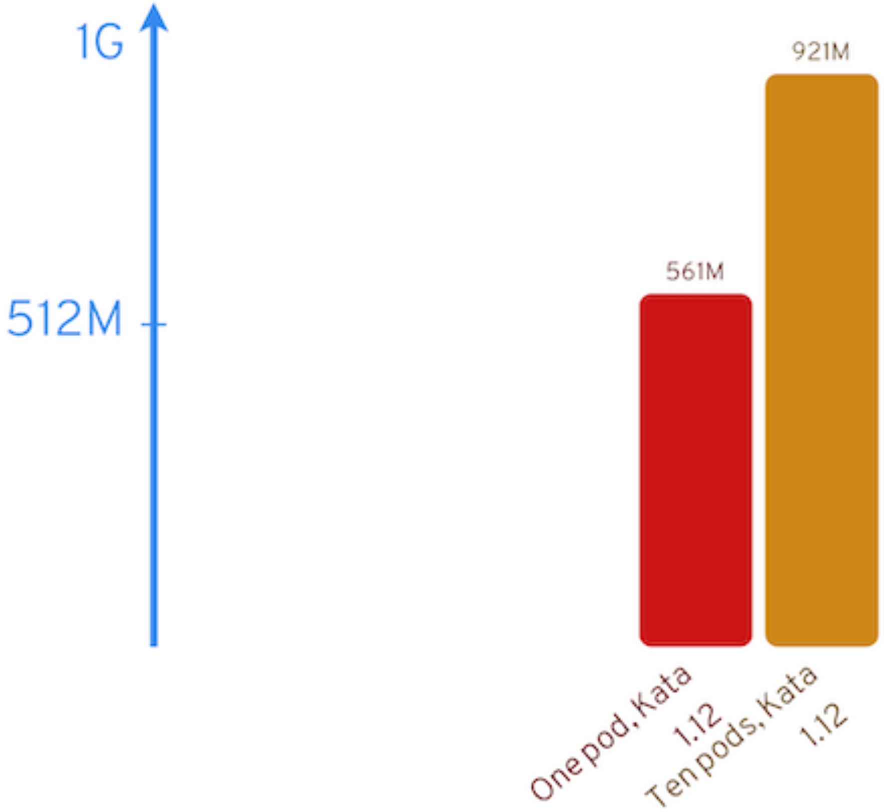
Memory - disk caches

Perform a dnf install inside the running pods



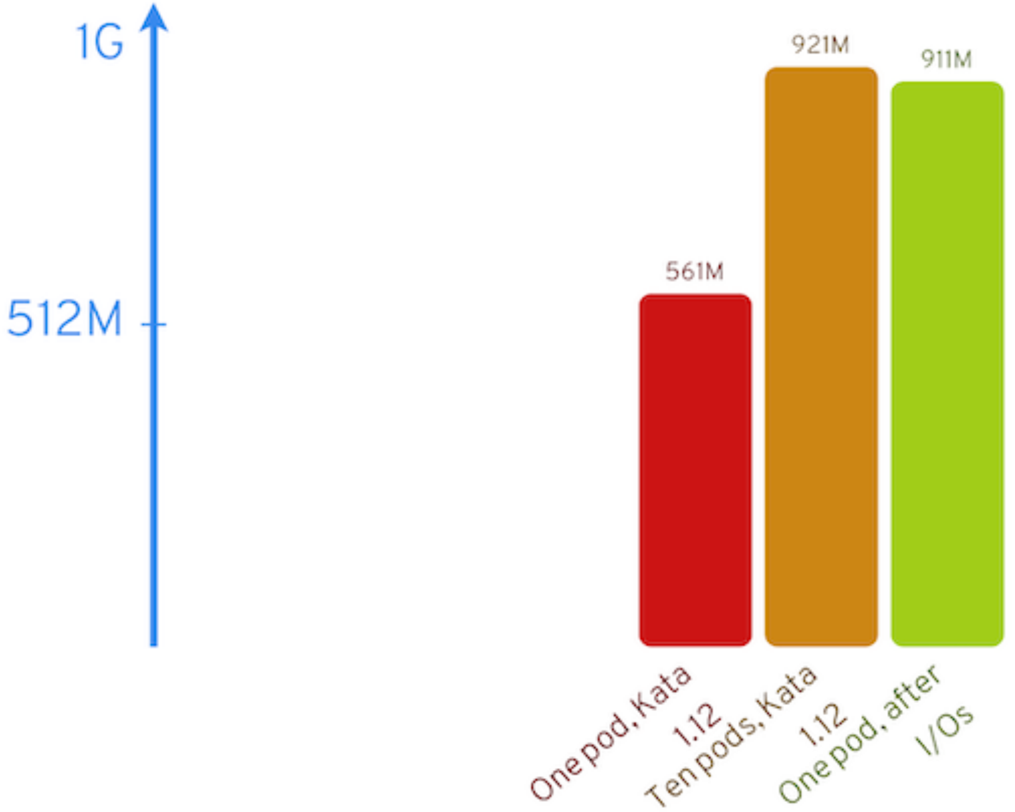
Memory - disk caches

Perform a dnf install inside the running pods



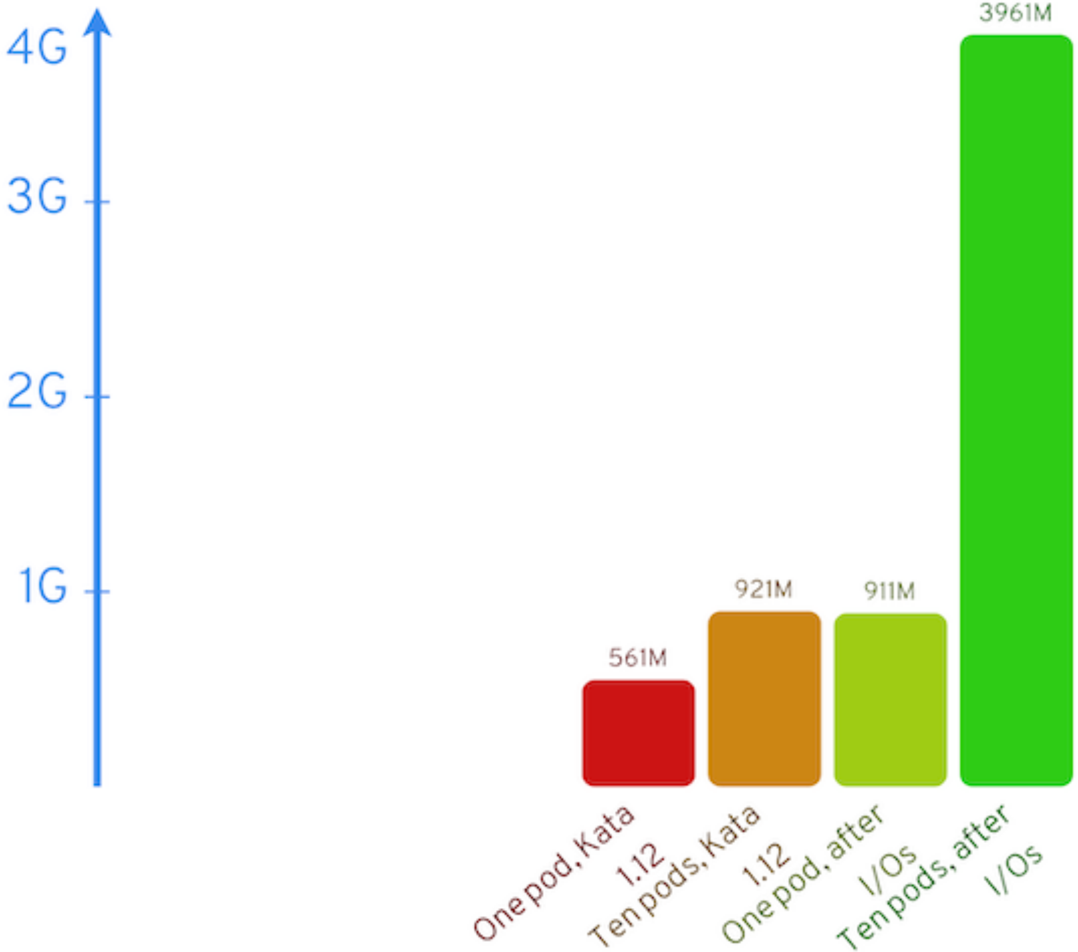
Memory - disk caches

Perform a dnf install inside the running pods



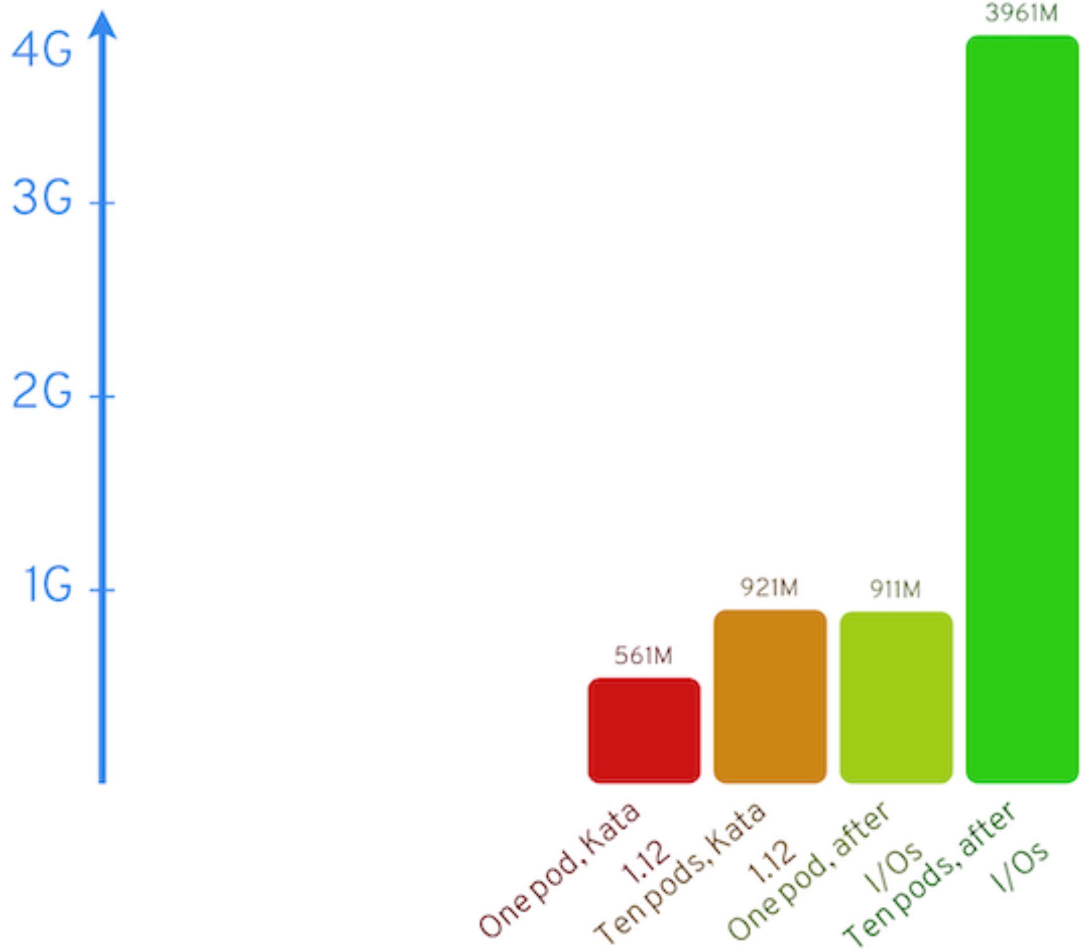
Memory - disk caches

Perform a dnf install inside the running pods



Memory - disk caches

Perform a dnf install inside the running pods



- I/O buffers consume a lot of memory.
- DAX allows host and guests to share that memory

Memory overhead - It's bad

Huge effort needed to make things better

Memory overhead - It's bad

Huge effort needed to make things better

- Too many processes

Processes

```
3949656 root      20   0   79.4m   2.2m   1.9m S   0.0   0.0   0:00.00  `- conmon
3949741 root      20   0   85.1m   4.3m   3.8m S   0.0   0.0   0:00.00      `- virtiofsd
3949788 root      20   0 4071.8m 250.6m 242.4m S   0.0   1.6   0:01.41          `- virtiofsd
3949781 root      20   0 2379.4m 594.9m 568.0m S   0.3   3.7   0:17.99      `- qemu-system-x86
3952158 root      20   0 1351.8m  14.5m   8.7m S   0.0   0.1   0:00.00      `- kata-shim
3953366 root      20   0   79.4m   2.2m   2.0m S   0.0   0.0   0:00.00  `- conmon
3953481 root      20   0 1207.8m  14.8m   9.1m S   0.0   0.1   0:00.00      `- kata-shim
3954774 root      20   0   79.4m   2.3m   2.0m S   0.0   0.0   0:00.00  `- conmon
3954896 root      20   0 1278.7m  14.9m   9.1m S   0.0   0.1   0:00.00      `- kata-shim
3956157 root      20   0   79.4m   2.3m   2.1m S   0.0   0.0   0:00.00  `- conmon
3956270 root      20   0 1350.9m  14.5m   8.9m S   0.0   0.1   0:00.00      `- kata-shim
3957451 root      20   0   79.4m   2.2m   1.9m S   0.0   0.0   0:00.00  `- conmon
3957566 root      20   0 1350.7m  14.6m   9.0m S   0.0   0.1   0:00.00      `- kata-shim
3958776 root      20   0   79.4m   2.3m   2.1m S   0.0   0.0   0:00.00  `- conmon
3958917 root      20   0 1350.7m  14.3m   8.6m S   0.0   0.1   0:00.00      `- kata-shim
3960230 root      20   0   79.4m   2.2m   2.0m S   0.0   0.0   0:00.00  `- conmon
3960373 root      20   0 1422.7m  14.7m   9.1m S   0.0   0.1   0:00.00      `- kata-shim
3961632 root      20   0   79.4m   2.3m   2.0m S   0.0   0.0   0:00.00  `- conmon
3961757 root      20   0 1207.8m  14.0m   8.4m S   0.0   0.1   0:00.00      `- kata-shim
3962991 root      20   0   79.4m   2.3m   2.1m S   0.0   0.0   0:00.00  `- conmon
3963137 root      20   0 1278.4m  14.6m   9.0m S   0.0   0.1   0:00.00      `- kata-shim
3964472 root      20   0   79.4m   2.2m   2.0m S   0.0   0.0   0:00.00  `- conmon
3964589 root      20   0 1279.5m  14.5m   8.8m S   0.0   0.1   0:00.00      `- kata-shim
3966829 root      20   0   79.4m   2.2m   2.0m S   0.0   0.0   0:00.00  `- conmon
3966963 root      20   0 1278.7m  14.7m   9.1m S   0.0   0.1   0:00.00      `- kata-shim
3968240 root      20   0   79.4m   2.2m   2.0m S   0.0   0.0   0:00.00  `- conmon
3968379 root      20   0 1278.7m  14.4m   8.6m S   0.0   0.1   0:00.00      `- kata-shim
```

Memory overhead - It's bad

Huge effort needed to make things better

- Too many processes
  - Partially addressed by 2.0

Kata Shim v2

```
779277 root   20  0 3958.9m  28.0m   4.7m S  0.0  0.4   5:39.81 `- crio
797519 root   20  0 1294.3m  19.2m   1.9m S  0.0  0.3  27:49.05 `- containerd-shim
797528 root   20  0   77.9m   0.9m   0.9m S  0.0  0.0   0:00.00    `- virtiofsd
797543 root   20  0 7175.1m 261.3m 259.5m S  0.7  3.4  87:09.12      `- virtiofsd
797538 root   20  0 4411.4m 626.2m 625.3m S  0.7  8.2 126:26.30 `- qemu-system-x86
```

Memory overhead - It's bad

Huge effort needed to make things better

- Too many processes
  - Partially addressed by 2.0
  - ... but single point of failure

Kata Shim v2

```
779277 root      20   0 3958.9m  28.0m   4.7m S   0.0   0.4   5:39.81  `- crio
797519 root      20   0 1294.3m  19.2m   1.9m S   0.0   0.3  27:49.05  `- containerd-shim
797528 root      20   0   77.9m   0.9m   0.9m S   0.0   0.0   0:00.00      `- virtiofsd
797543 root      20   0 7175.1m 261.3m 259.5m S   0.7   3.4  87:09.12          `- virtiofsd
797538 root      20   0 4411.4m 626.2m 625.3m S   0.7   8.2 126:26.30  `- qemu-system-x86
```

Memory overhead - It's bad

Huge effort needed to make things better

- Too many processes
  - Partially addressed by 2.0
  - ... but single point of failure
- A lot of memory in buffer caches

Kata Shim v2

```
779277 root      20   0 3958.9m  28.0m   4.7m S   0.0   0.4   5:39.81 `- crio
797519 root      20   0 1294.3m  19.2m   1.9m S   0.0   0.3  27:49.05 `- containerd-shim
797528 root      20   0   77.9m   0.9m   0.9m S   0.0   0.0   0:00.00     `- virtiofsd
797543 root      20   0 7175.1m 261.3m 259.5m S   0.7   3.4  87:09.12         `- virtiofsd
797538 root      20   0 4411.4m 626.2m 625.3m S   0.7   8.2 126:26.30 `- qemu-system-x86
```

Memory overhead - It's bad

Huge effort needed to make things better

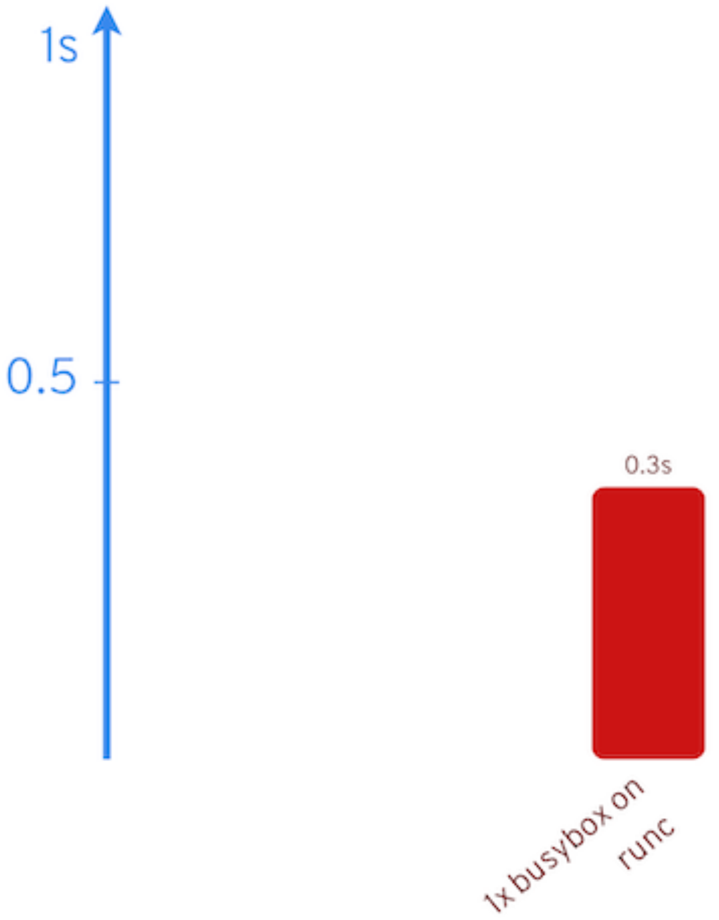
- Too many processes
  - Partially addressed by 2.0
  - ... but single point of failure
- A lot of memory in buffer caches
  - DAX allows host buffer caches to be used

Boot time

How fast does a container start?

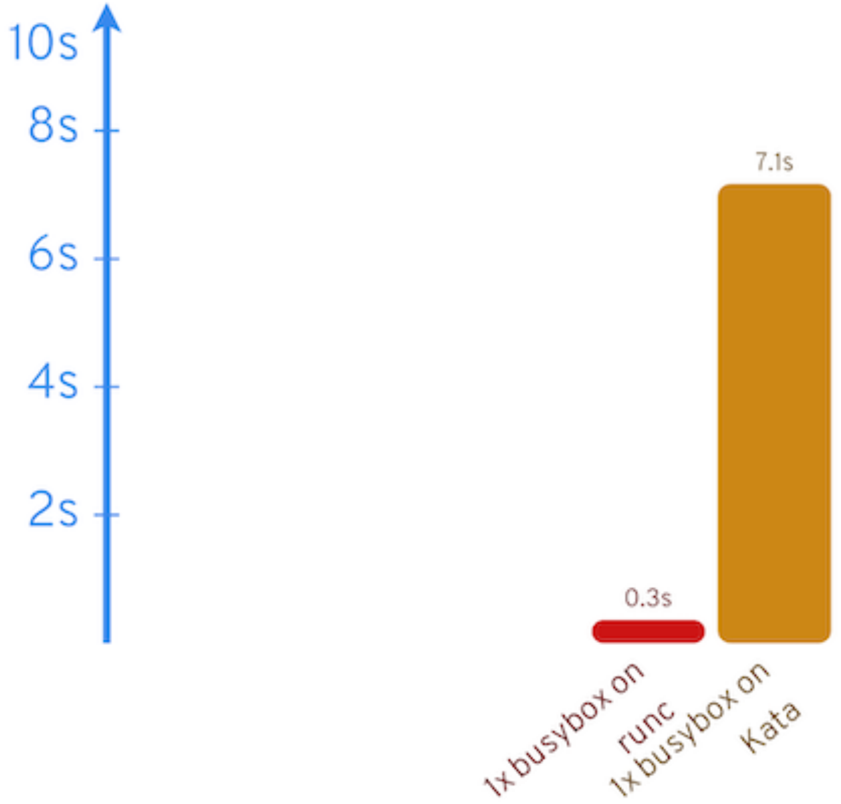
Boot time overhead

Time to boot containers



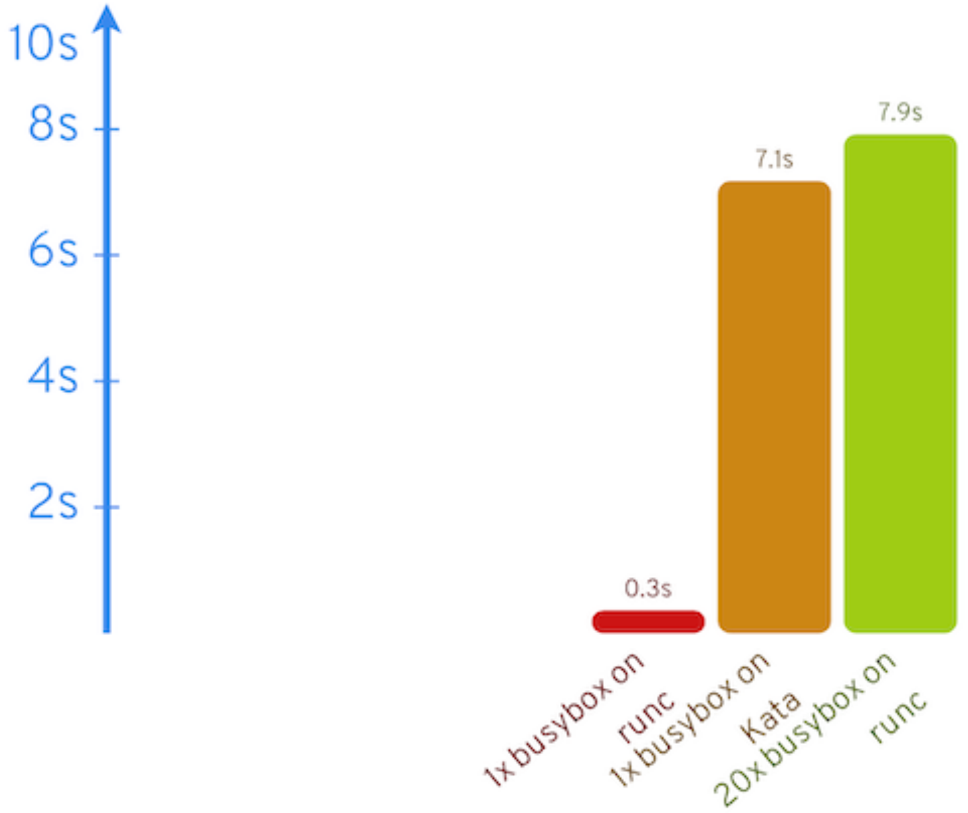
Boot time overhead

Time to boot containers



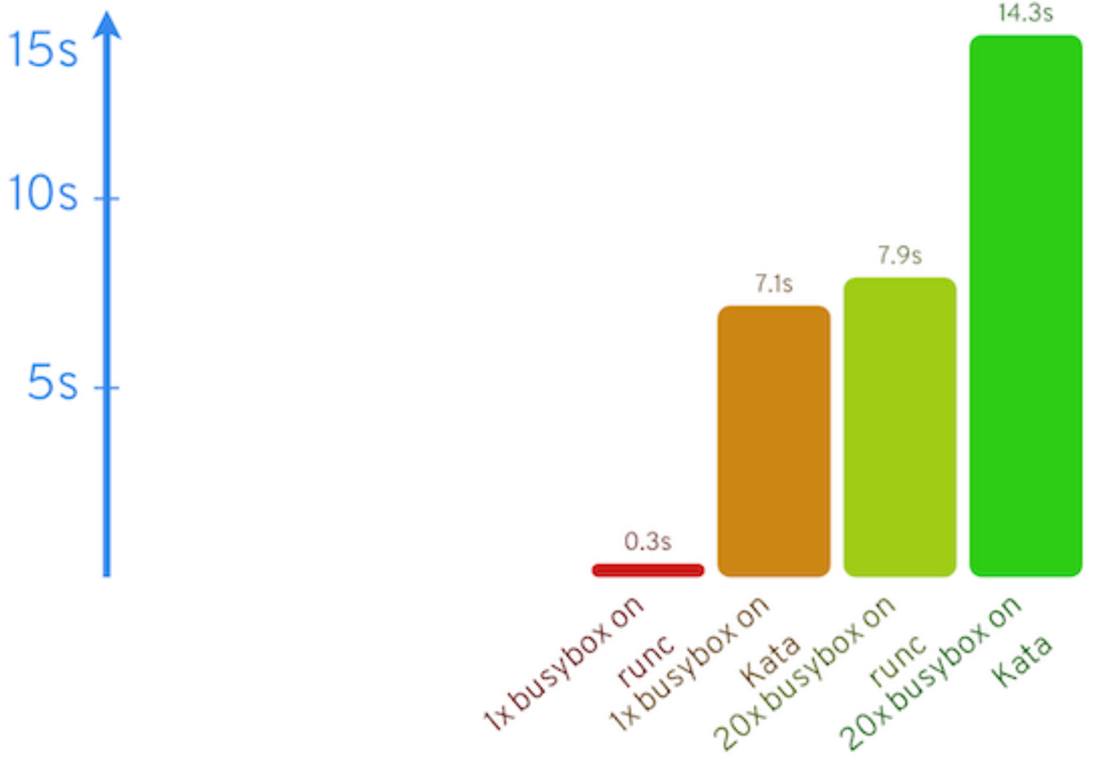
Boot time overhead

Time to boot containers



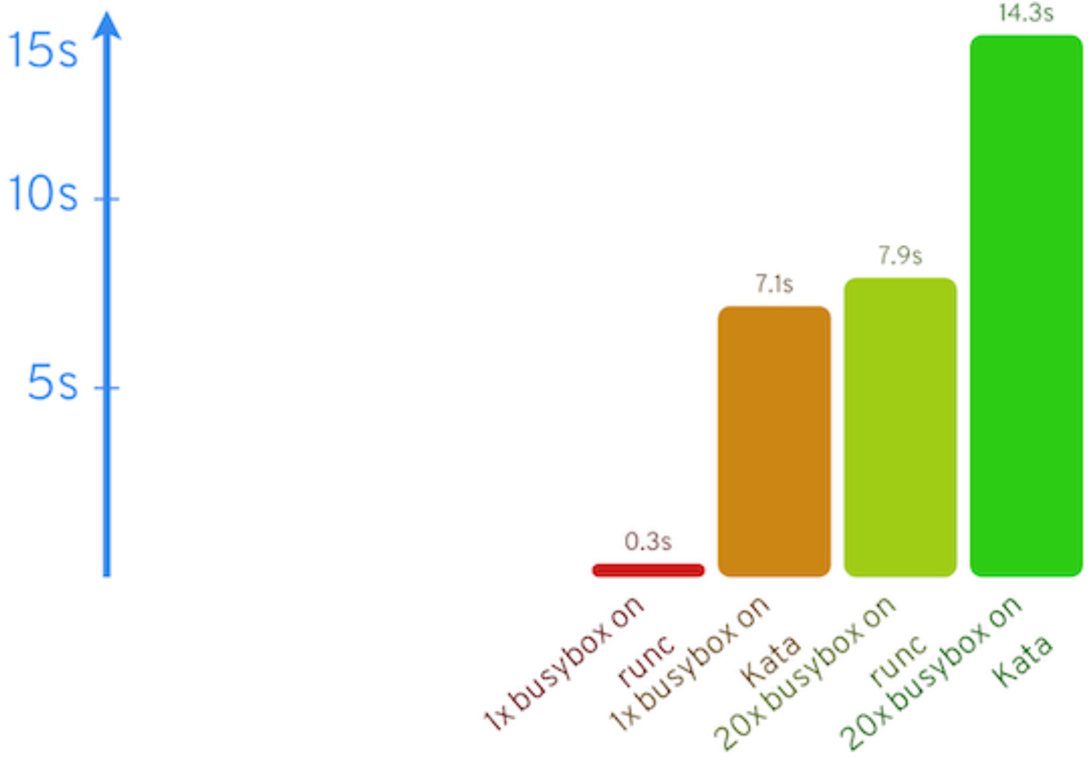
Boot time overhead

Time to boot containers



Boot time overhead

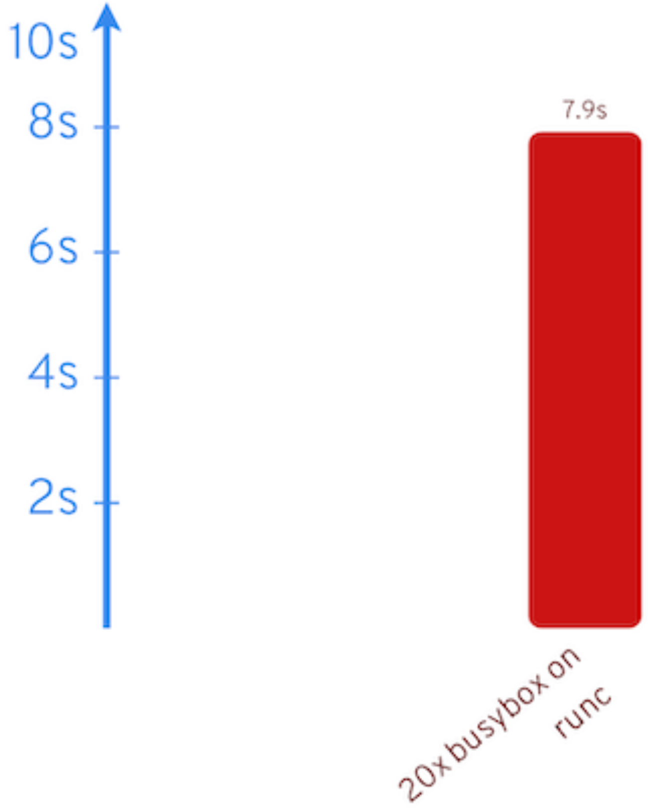
Time to boot containers



- Pod boot time is relatively constant (about 7s here)
- But can you run multiple containers in a single pod?

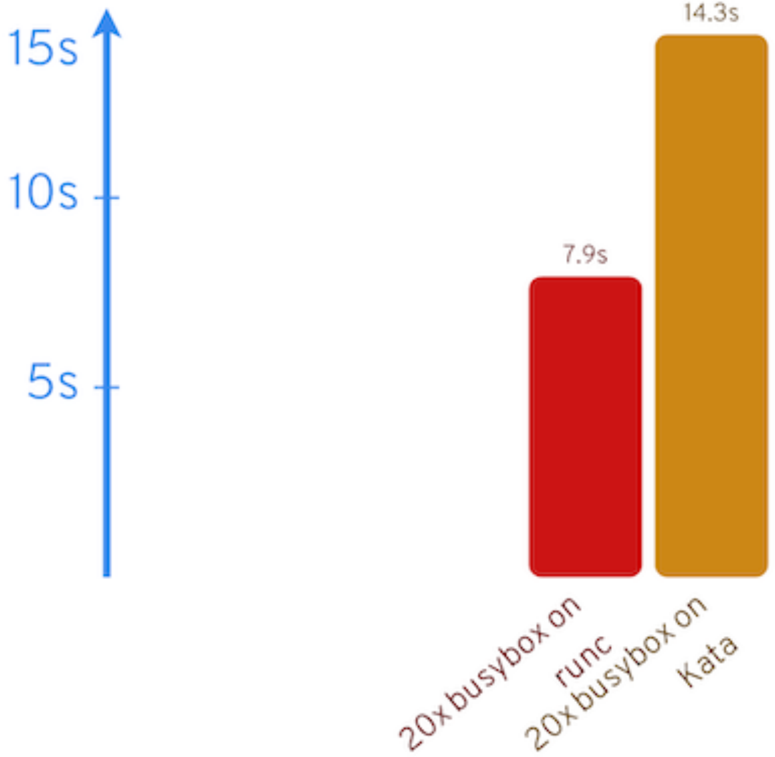
Run time overhead

Shutting down also takes time



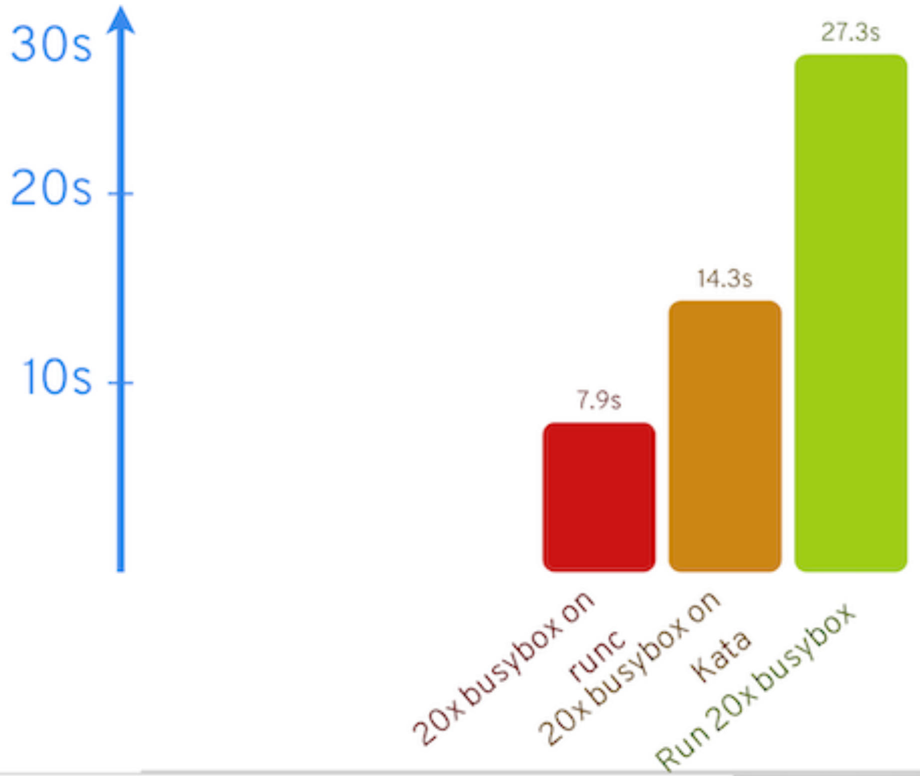
Run time overhead

Shutting down also takes time



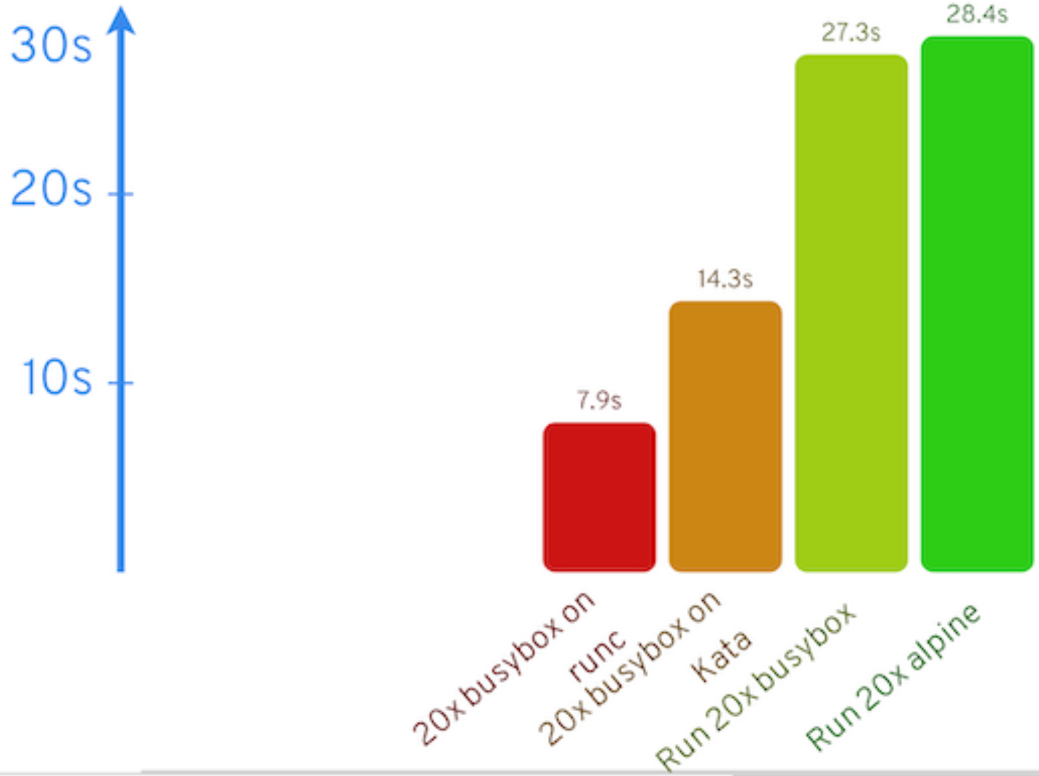
Run time overhead

Shutting down also takes time



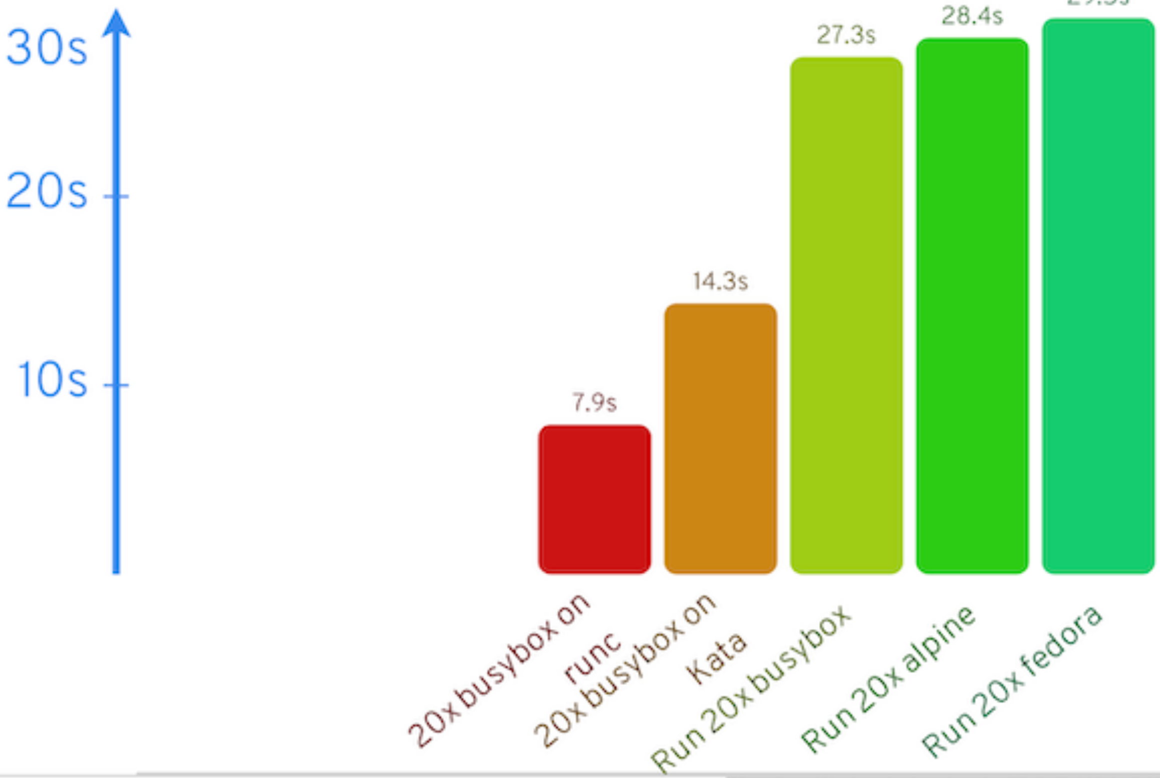
Run time overhead

Shutting down also takes time



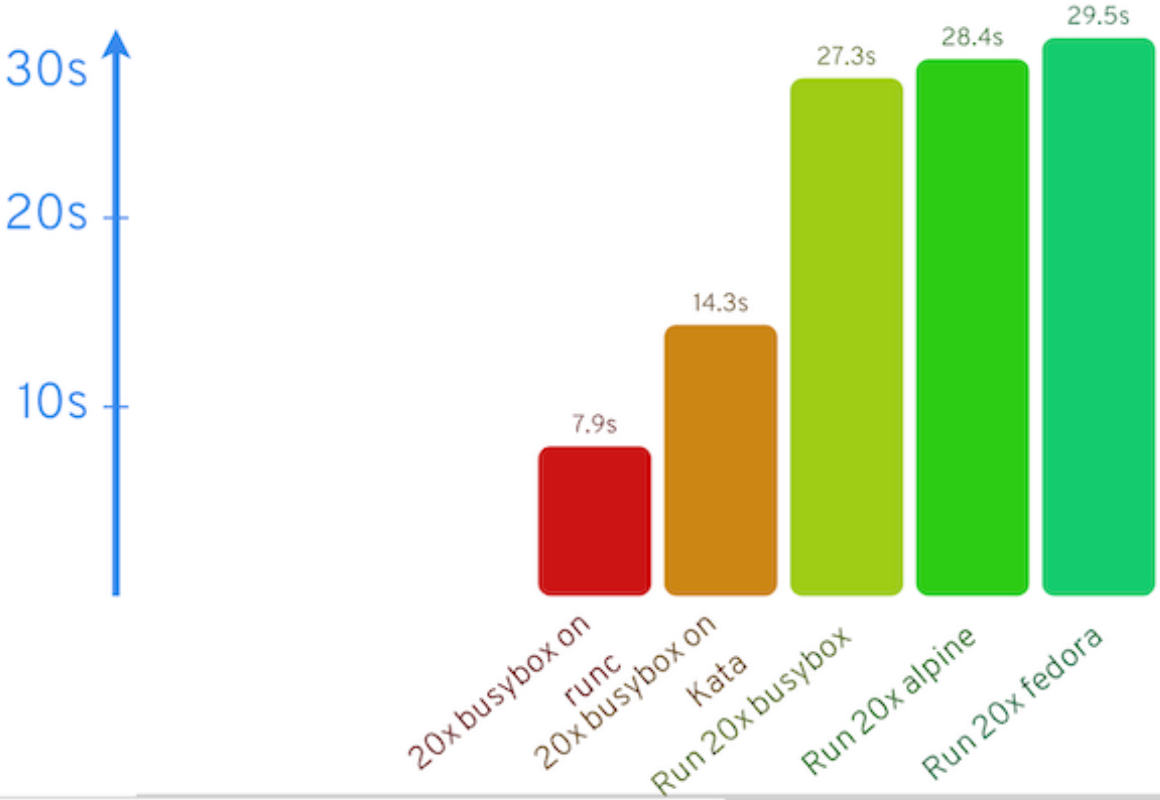
Run time overhead

Shutting down also takes time



Run time overhead

Shutting down also takes time



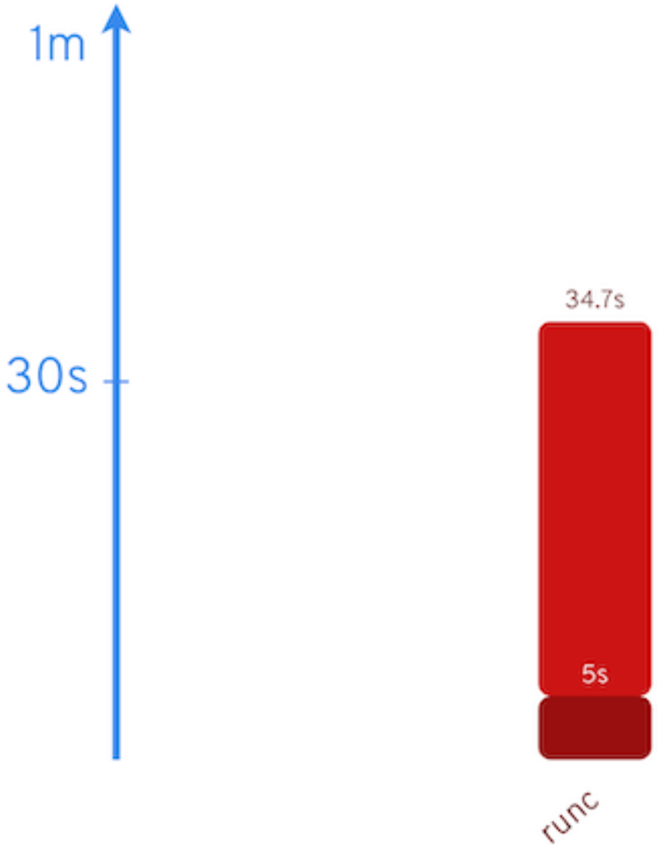
- Shutting down a Kata pod also takes time
- With Kata, the container base you use is not as relevant

Disk I/O

Disk performance with virtiofs:
improved, but still not good

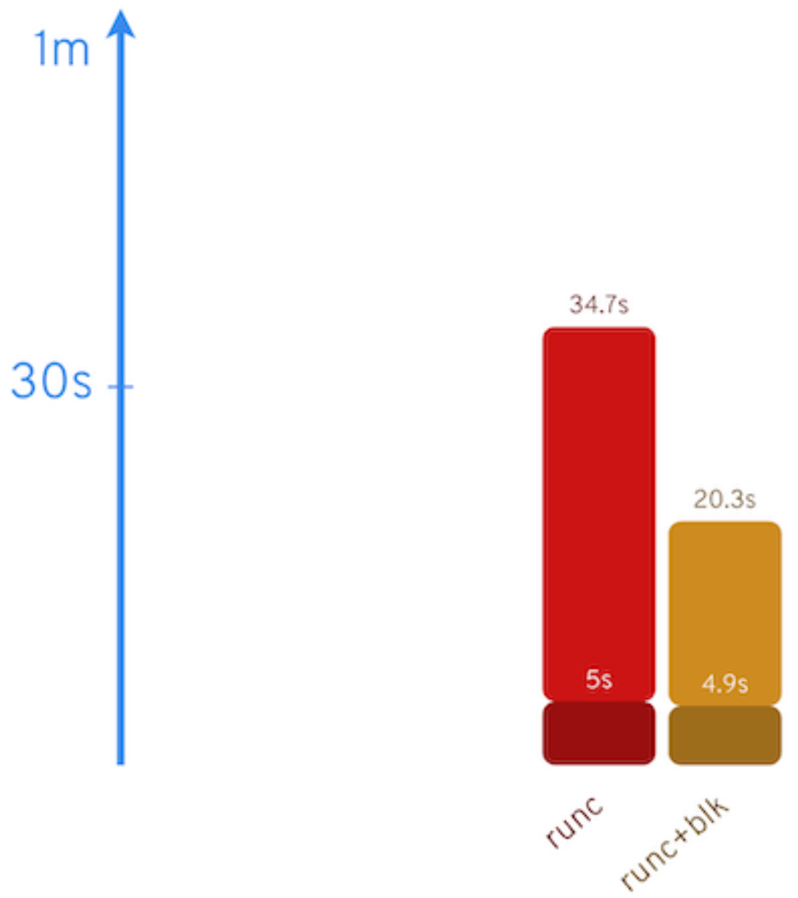
Disk read overhead

Copy files from the local file system



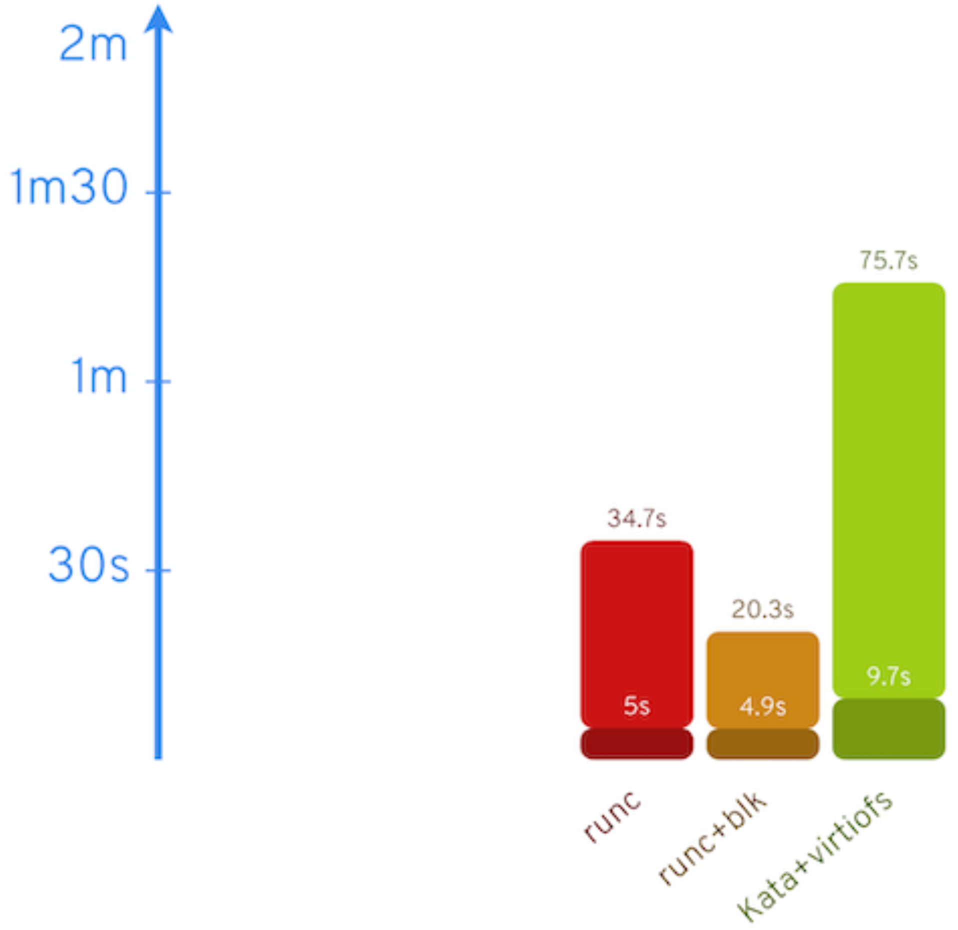
Disk read overhead

Copy files from the local file system



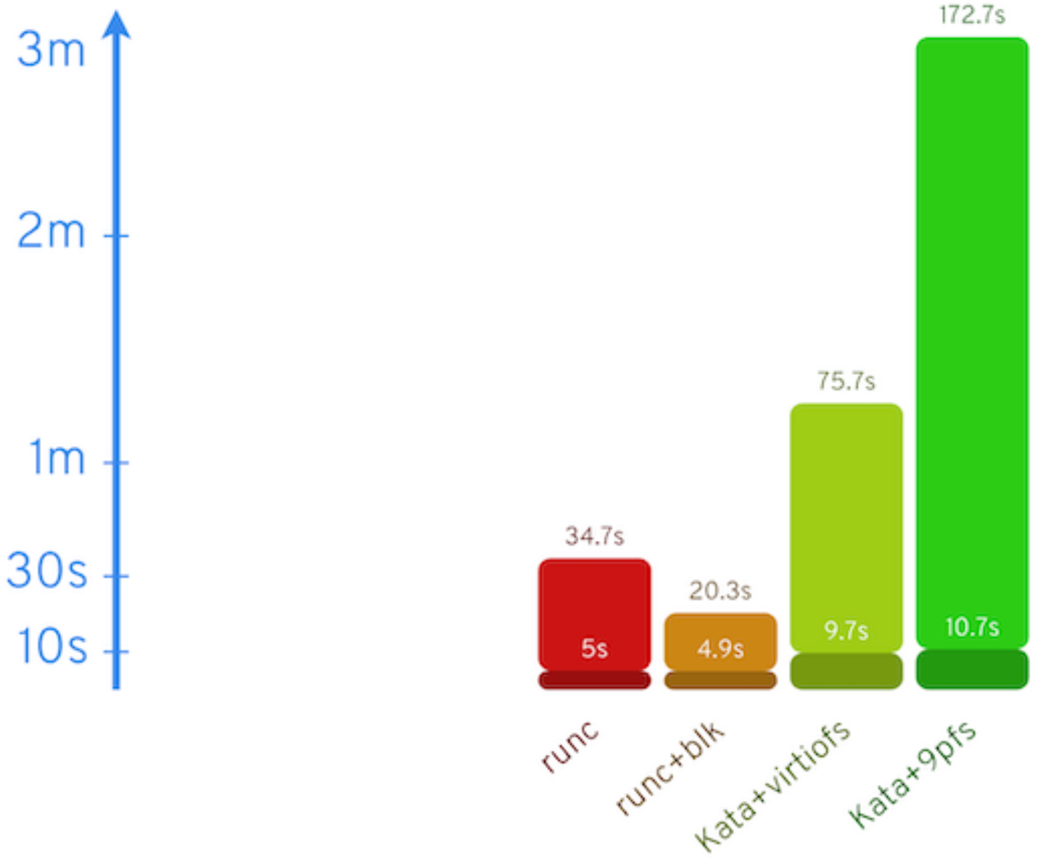
Disk read overhead

Copy files from the local file system



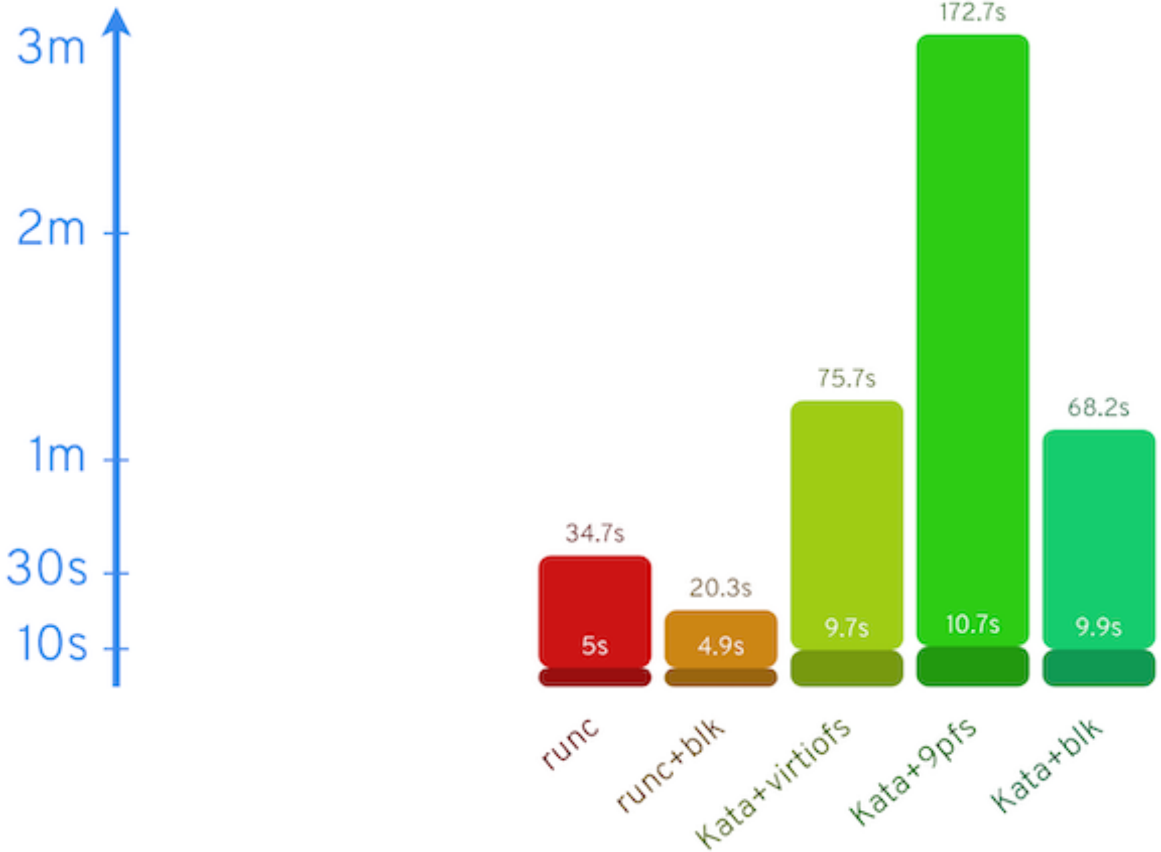
Disk read overhead

Copy files from the local file system



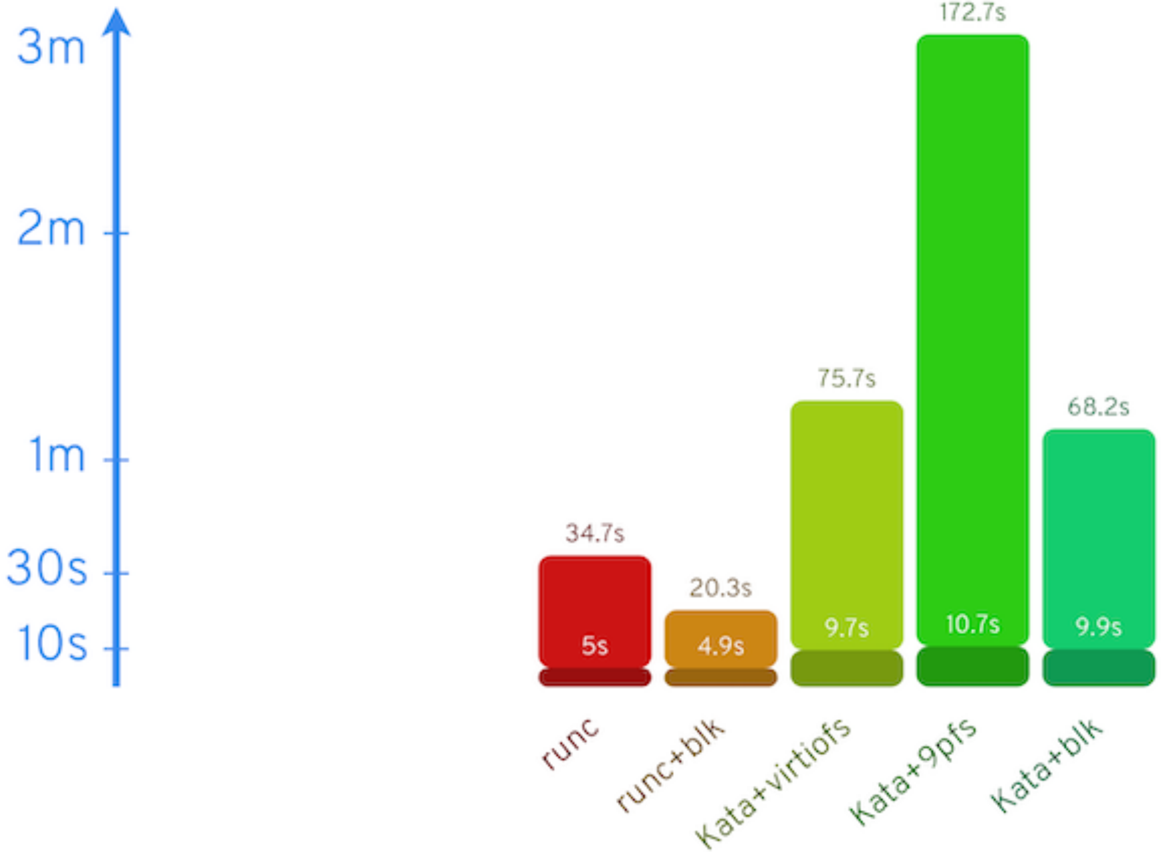
Disk read overhead

Copy files from the local file system



Disk read overhead

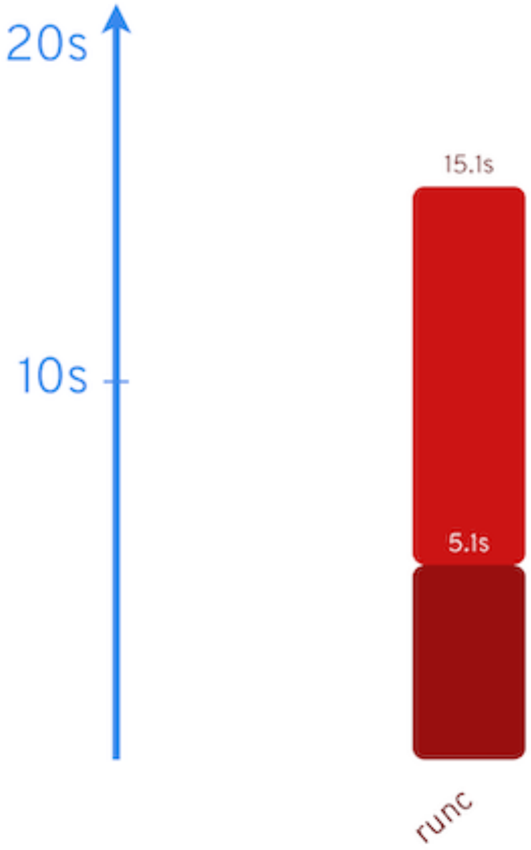
Copy files from the local file system



- File reads are markedly slower with Kata
- virtiofs improves a lot over 9pfs
- Kata benefits less from blk than runc

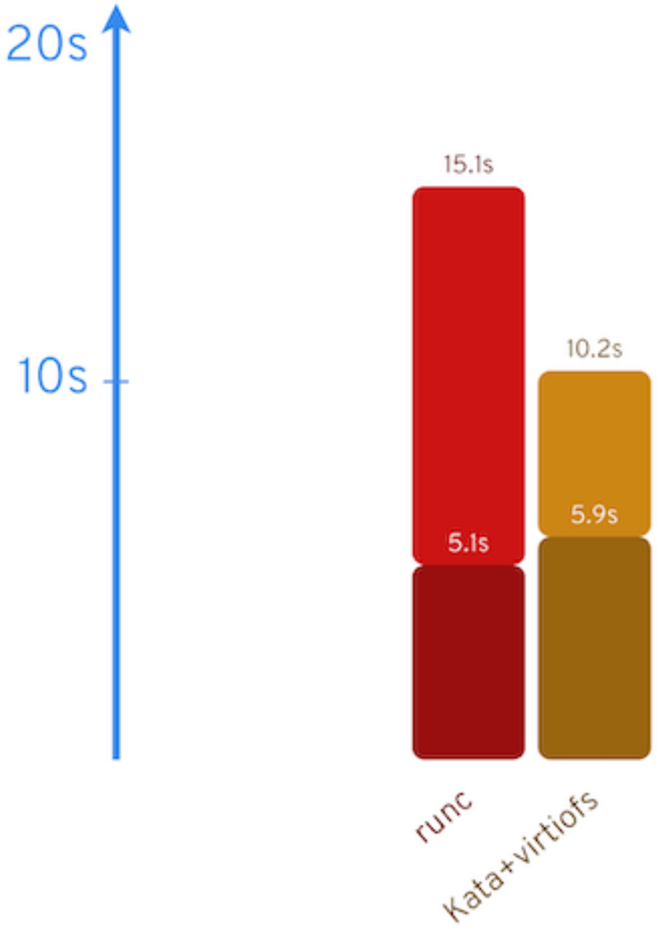
Disk write overhead

Write a 1.0G file from /dev/random



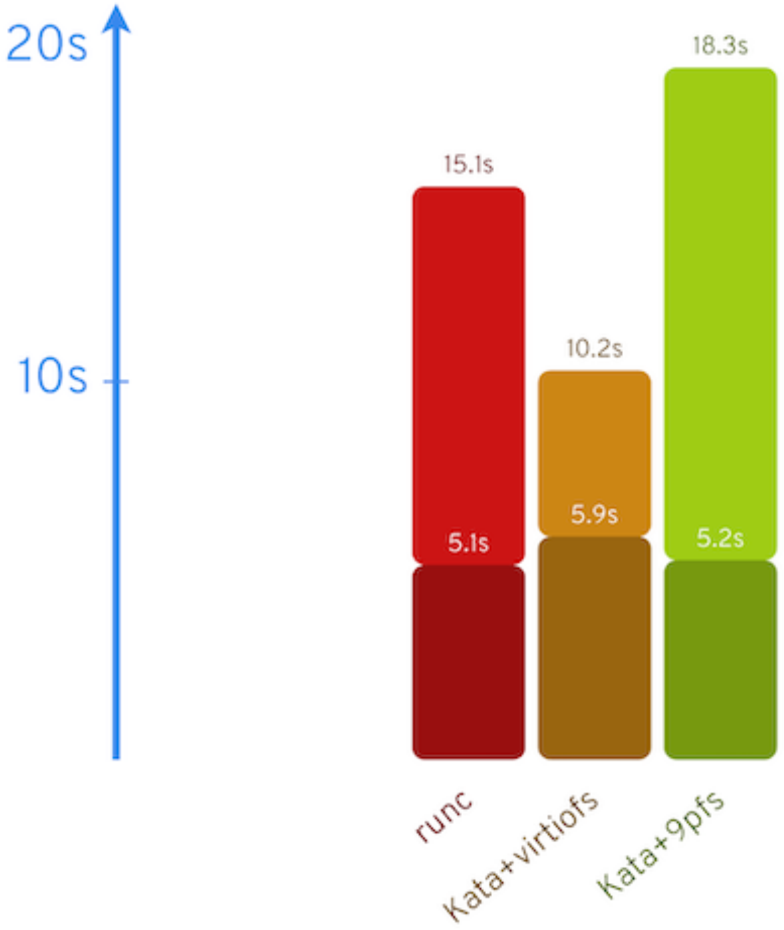
Disk write overhead

Write a 1.0G file from /dev/random



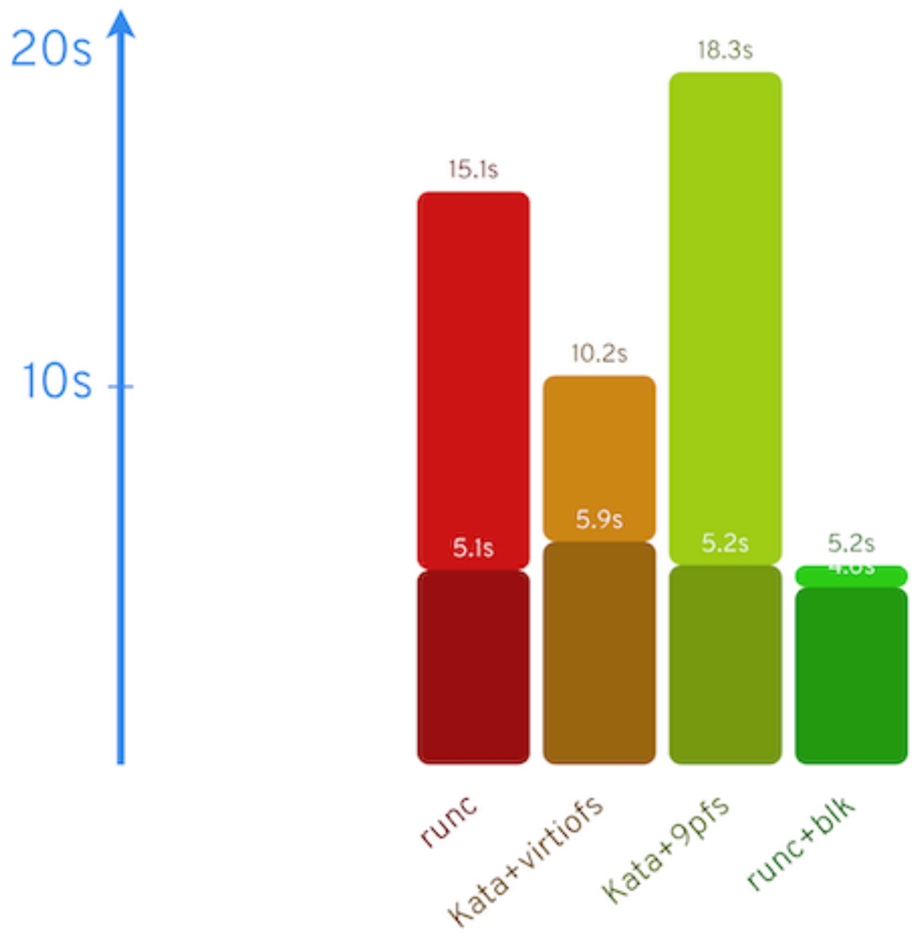
Disk write overhead

Write a 1.0G file from /dev/random



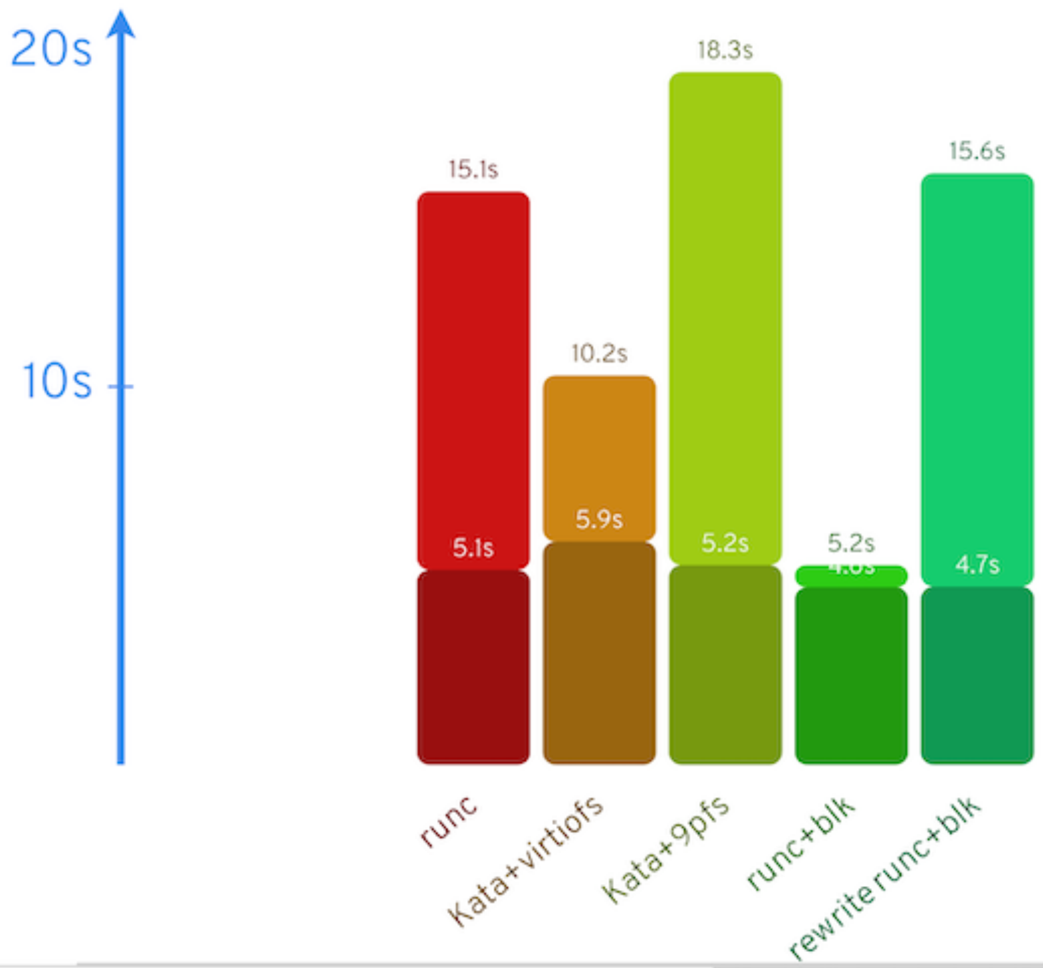
Disk write overhead

Write a 1.0G file from /dev/random



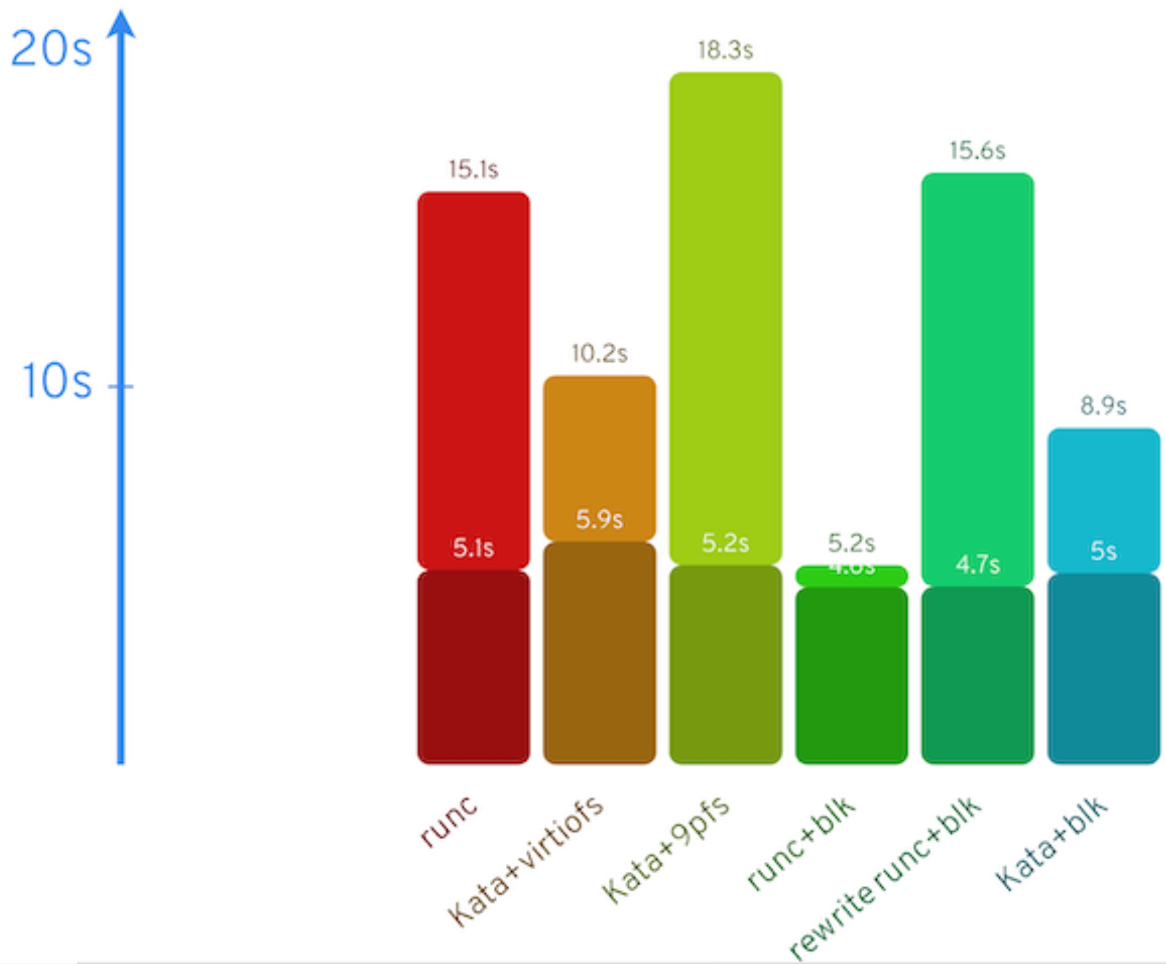
Disk write overhead

Write a 1.0G file from /dev/random



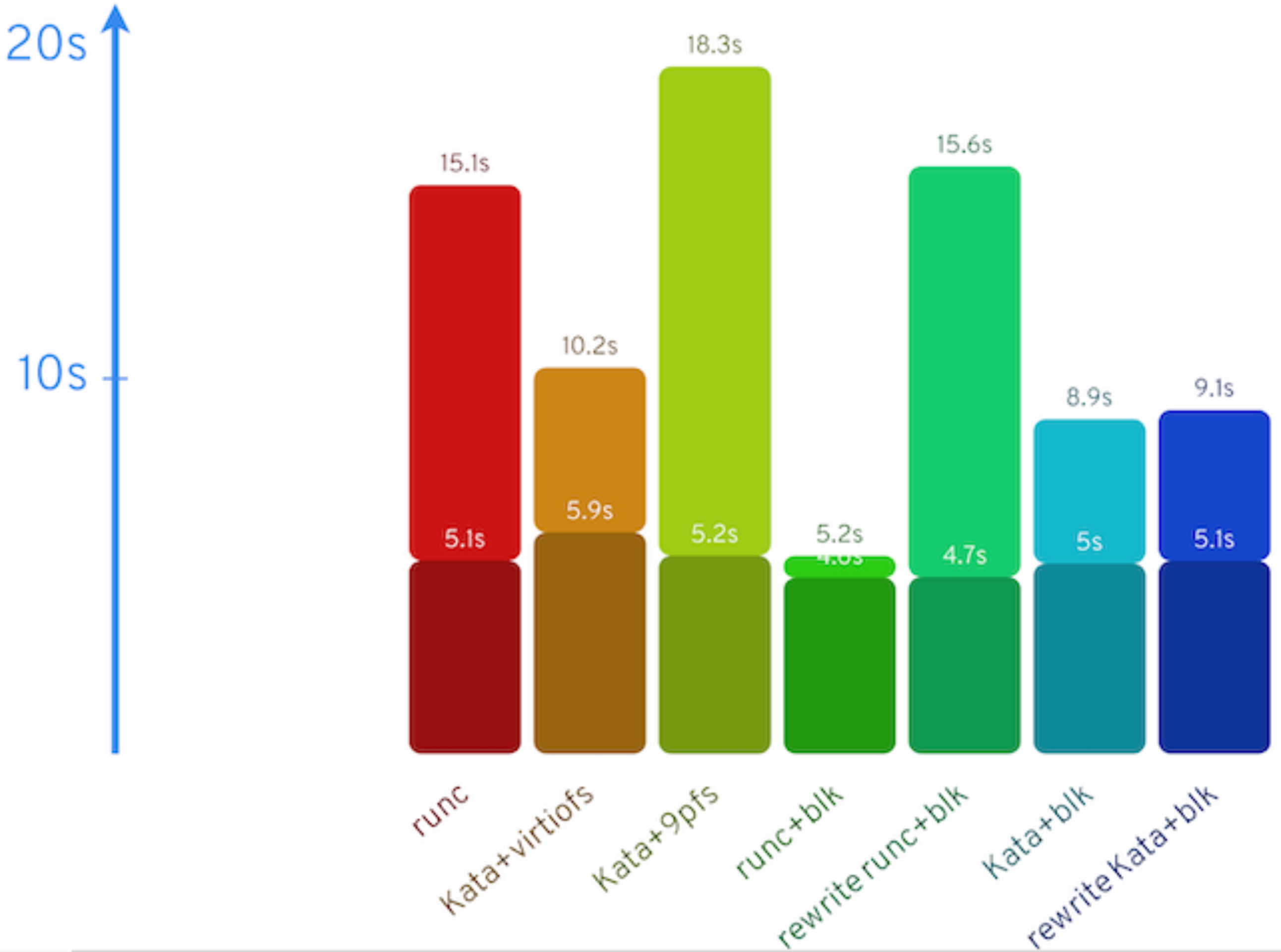
Disk write overhead

Write a 1.0G file from /dev/random



Disk write overhead

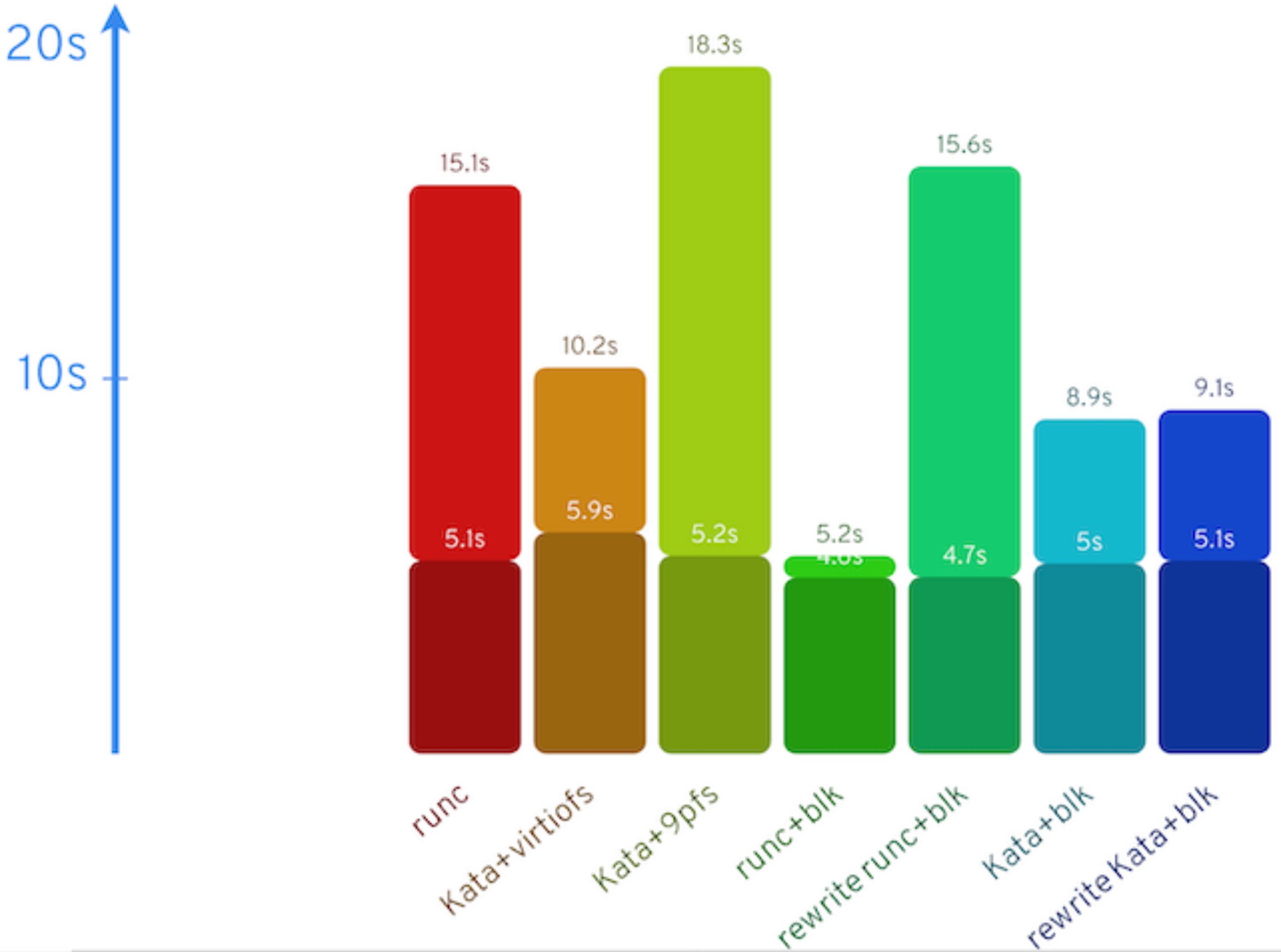
Write a 1.0G file from /dev/random



KVM FORUM Red Hat

Disk write overhead

Write a 1.0G file from /dev/random



- Kata with virtiofs outperforms runc here
- Overwriting a file is more expensive
- The effect is less marked with Kata

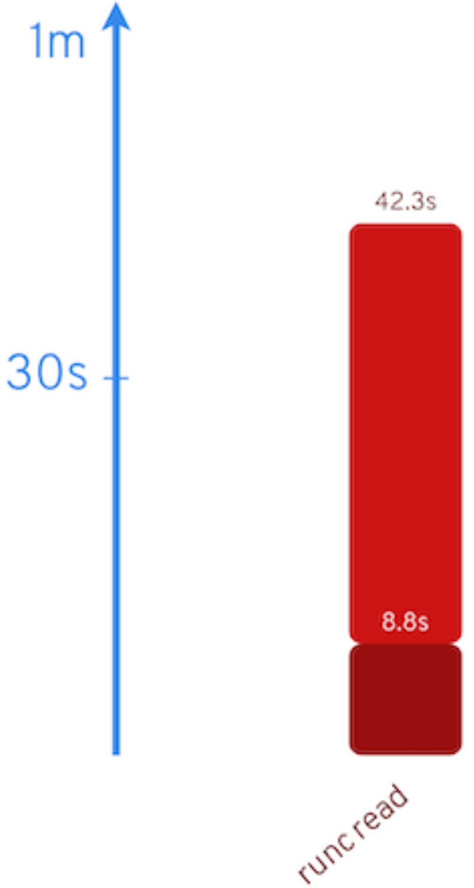
Networking I/O

Multiple networking configurations

At least some of them should perform well

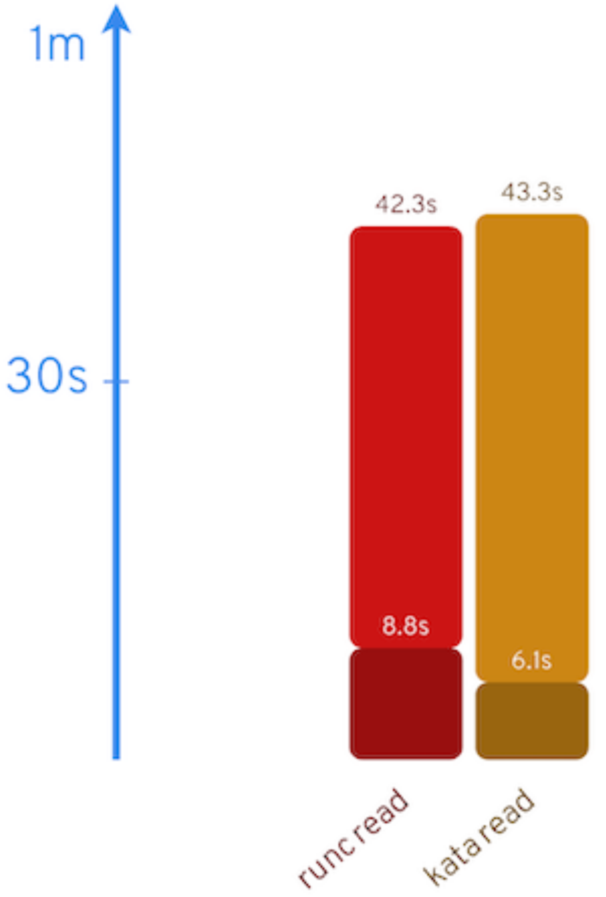
Network overhead

Read or write a 1G file from local server



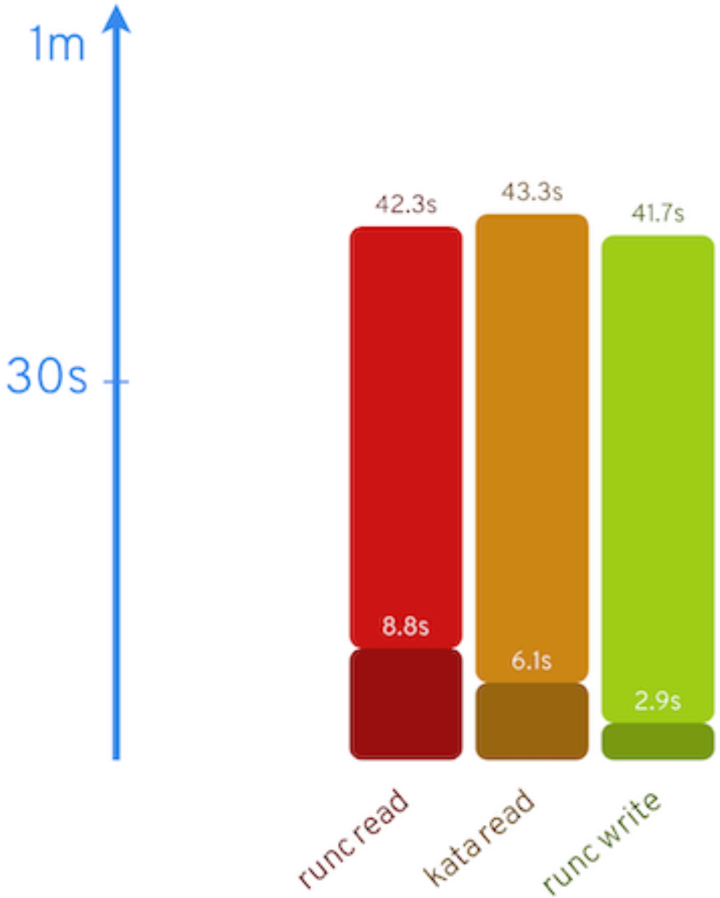
Network overhead

Read or write a 1G file from local server



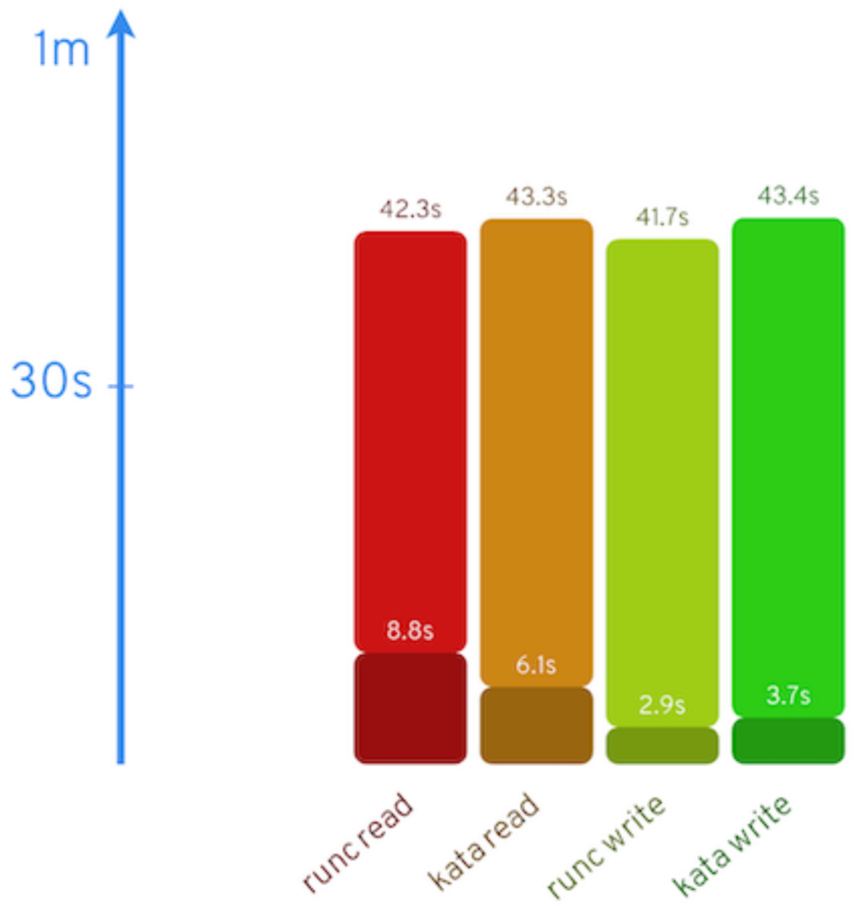
Network overhead

Read or write a 1G file from local server



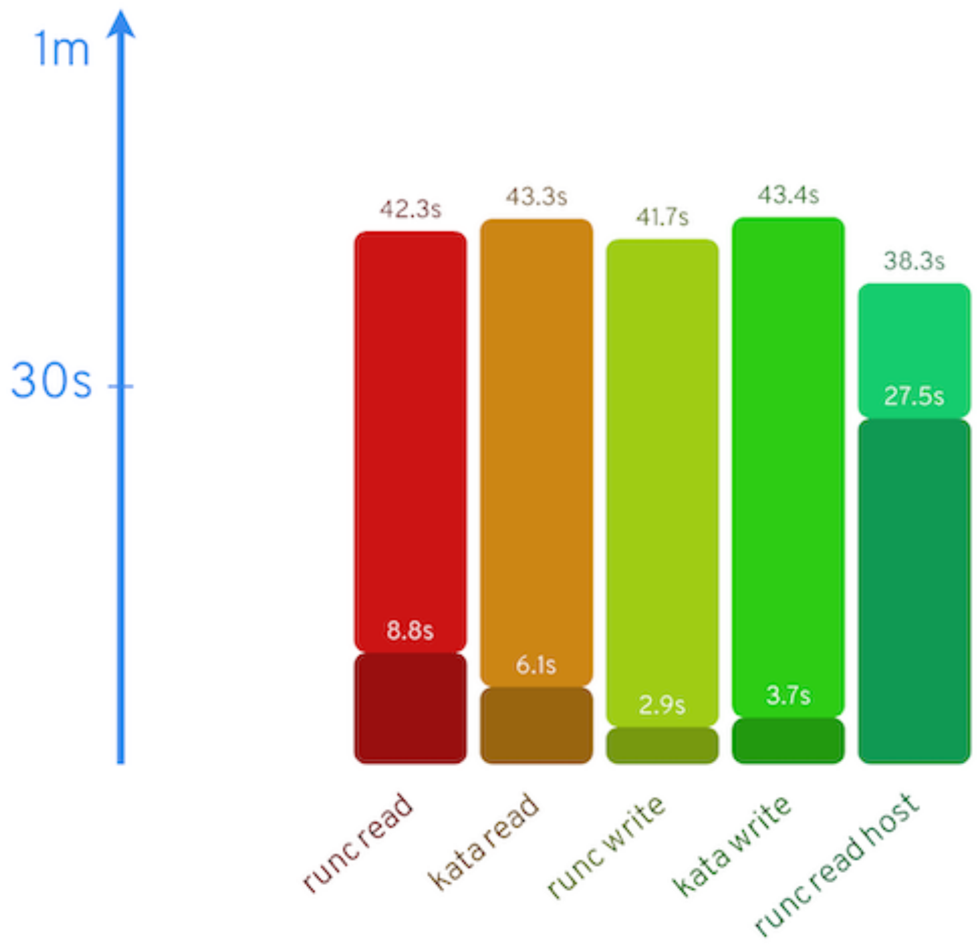
Network overhead

Read or write a 1G file from local server



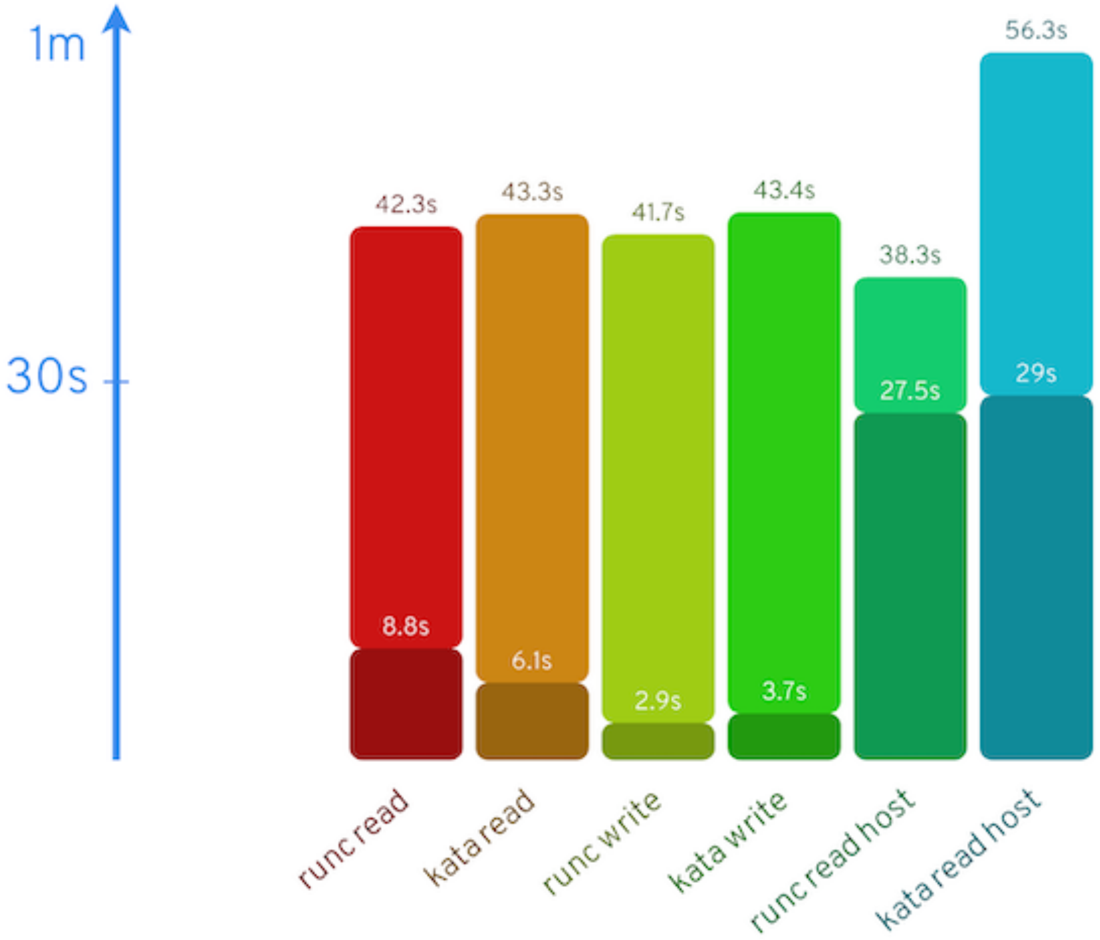
Network overhead

Read or write a 1G file from local server



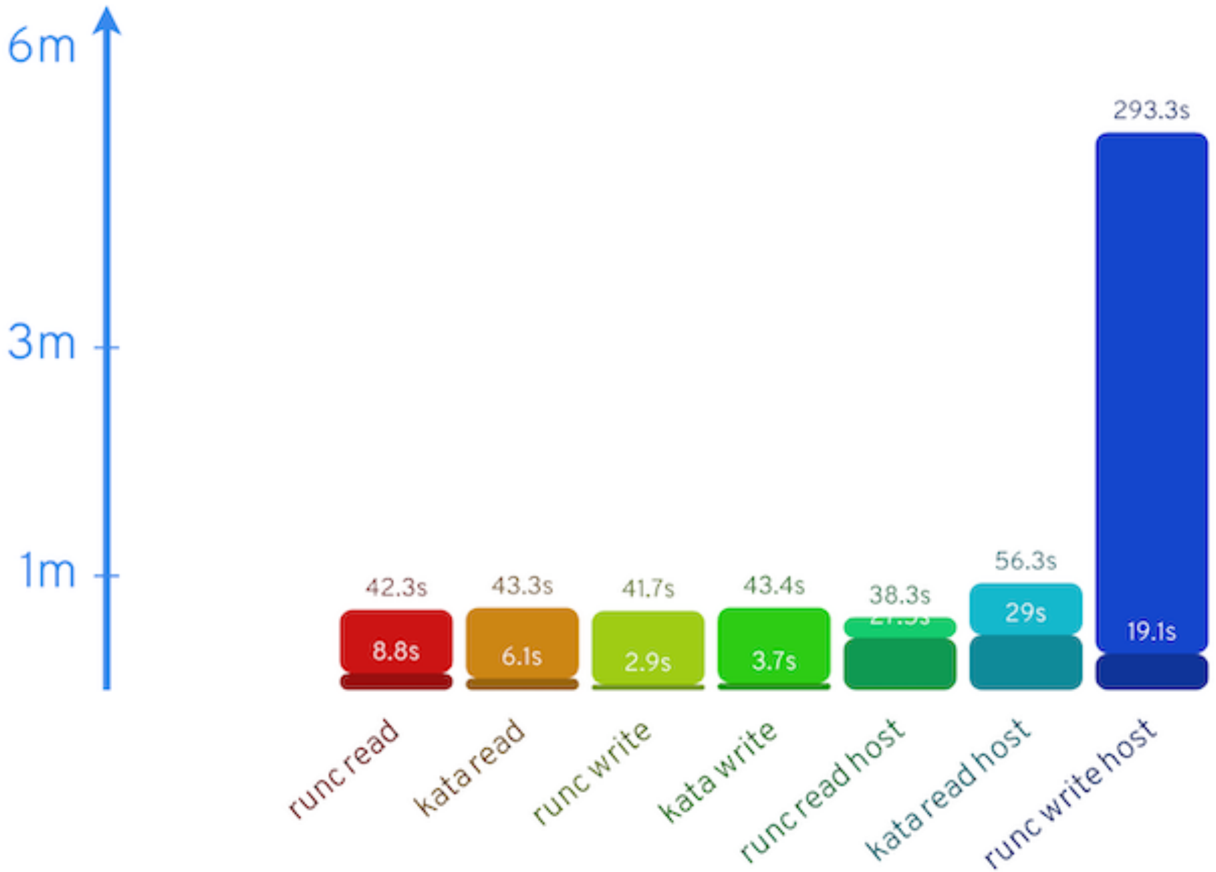
Network overhead

Read or write a 1G file from local server



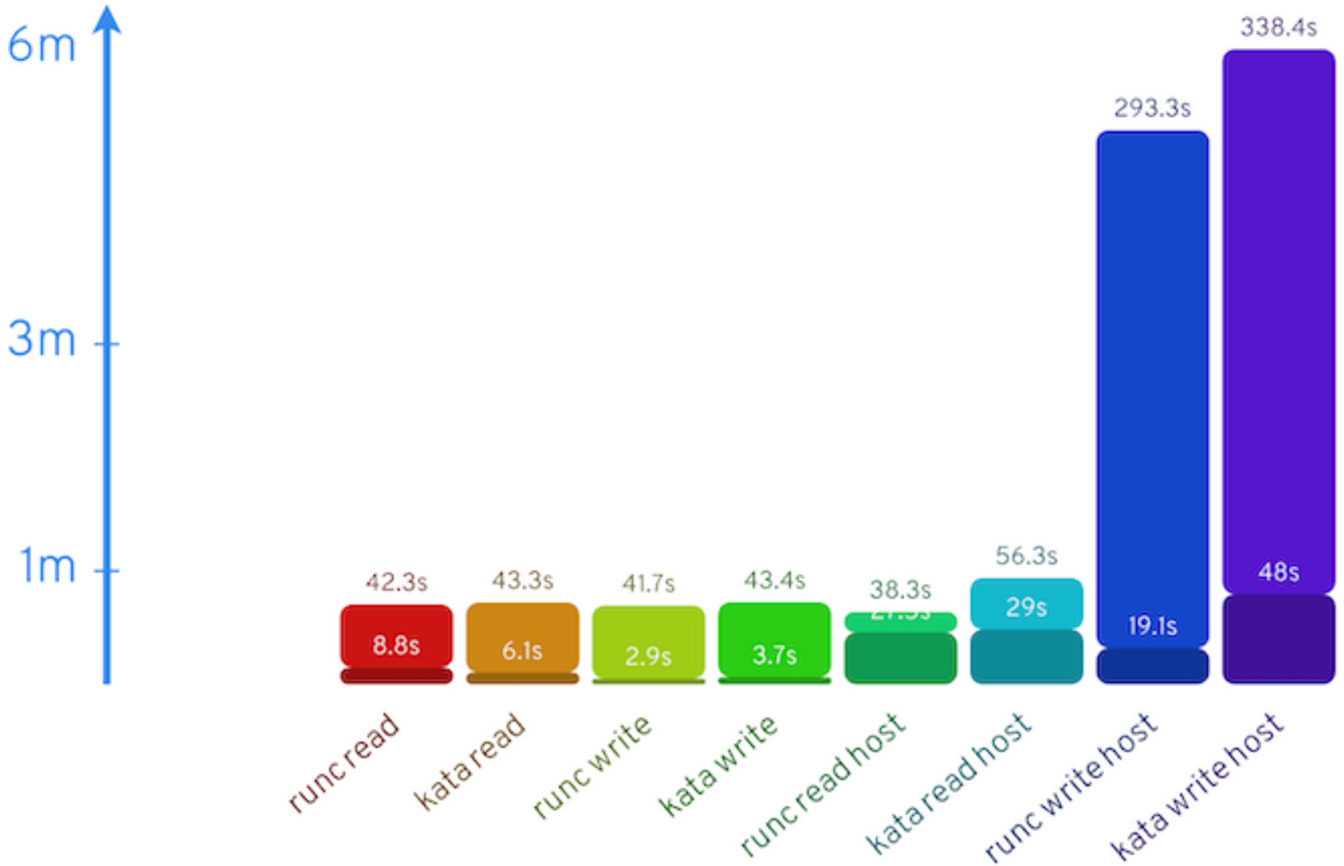
Network overhead

Read or write a 1G file from local server



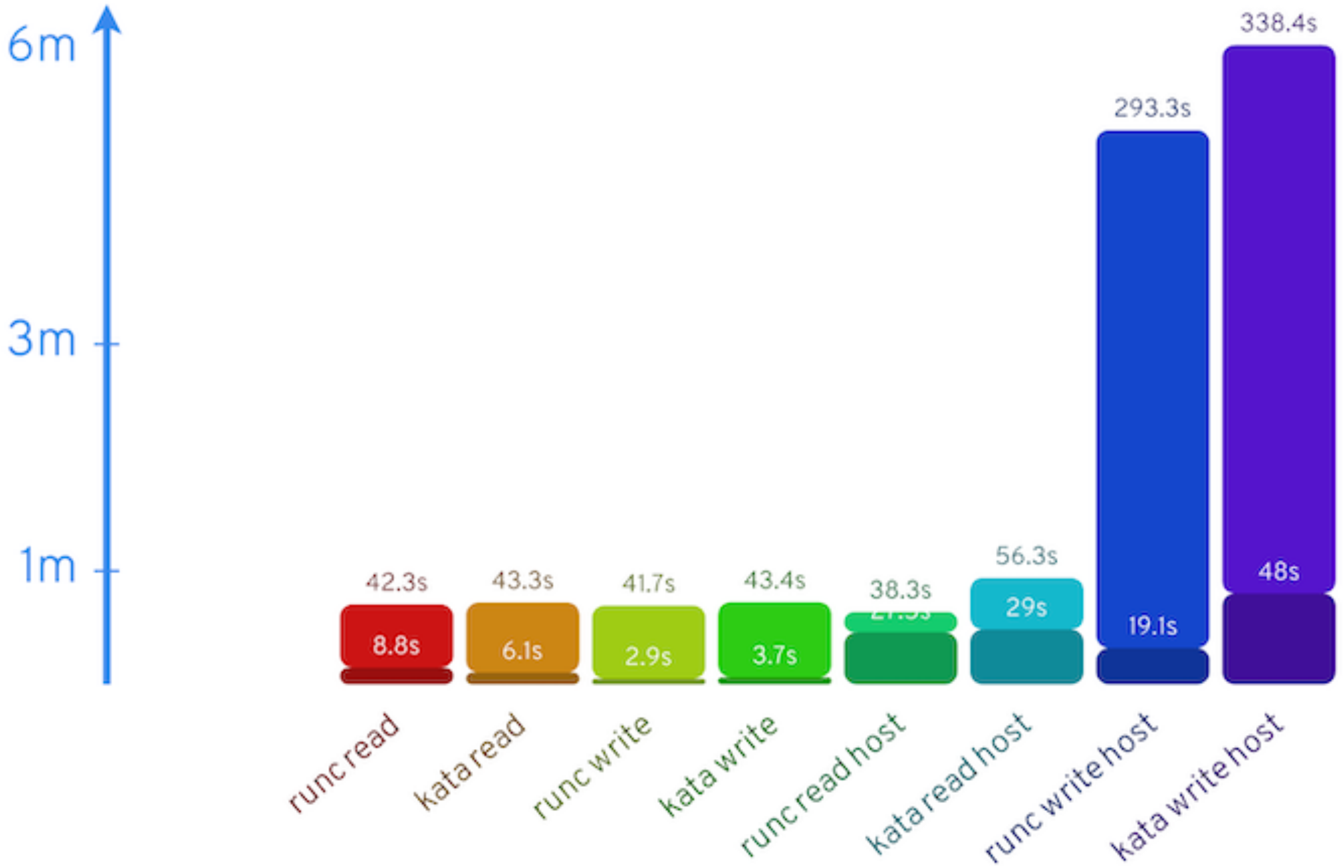
Network overhead

Read or write a 1G file from local server



Network overhead

Read or write a 1G file from local server



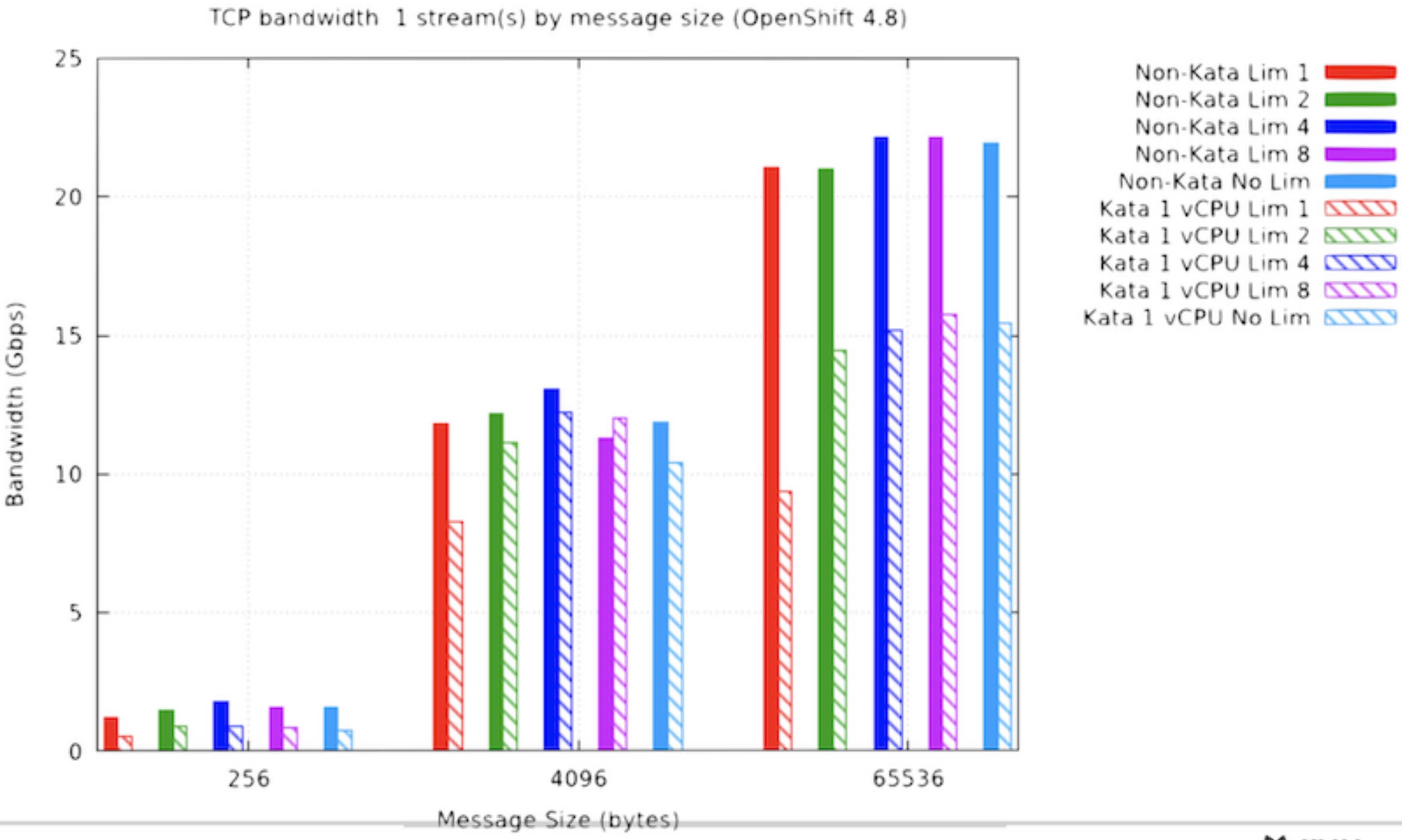
- Kata overhead is modest
  - Lower system usage in guest (moved to host)
- Host networking is rather bad for Kata
  - Writing is really bad

Network bandwidth

Path length increase make it harder to saturate link

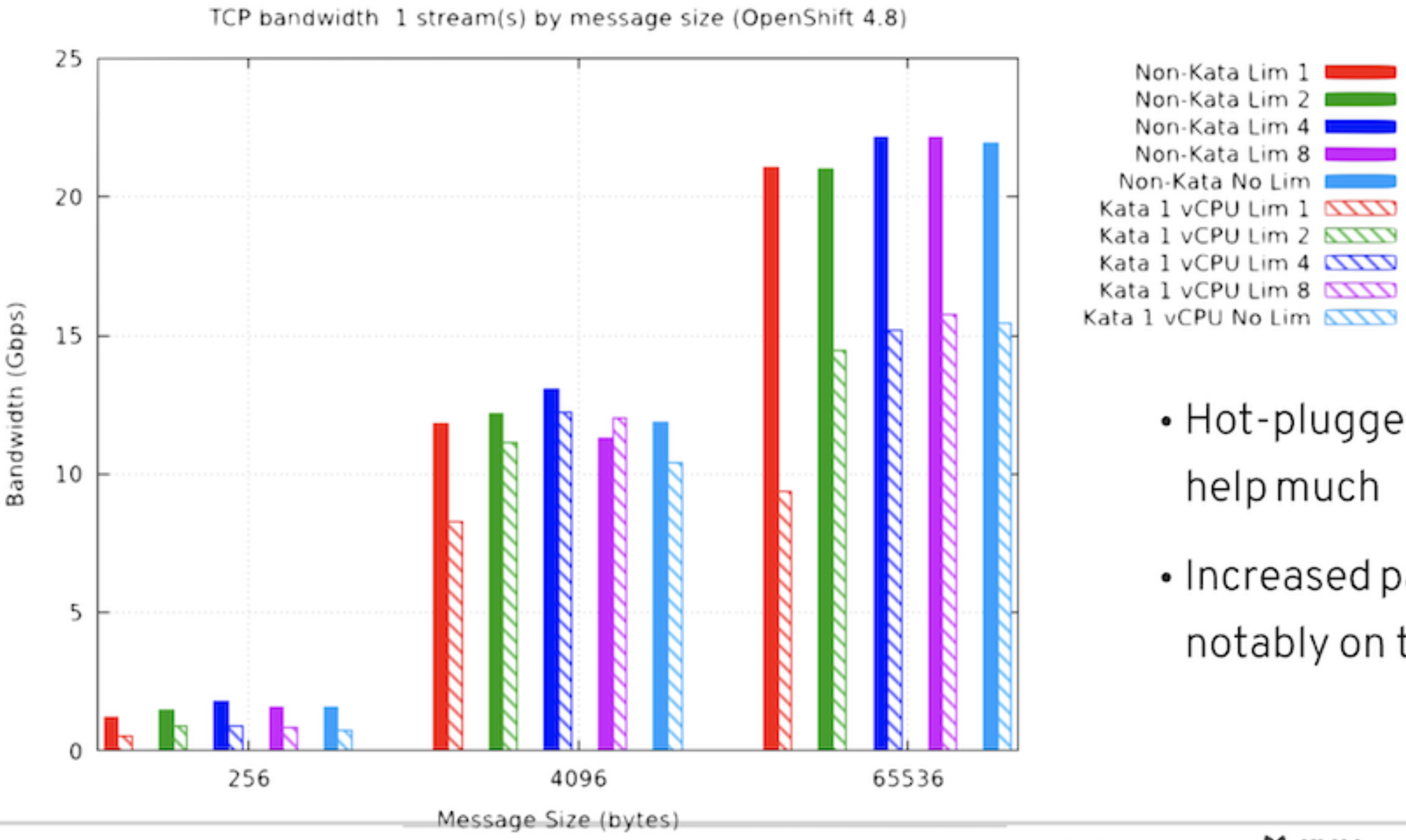
Network bandwidth

Path length increase make it harder to saturate link



Network bandwidth

Path length increase make it harder to saturate link



- Hot-plugged vCPUs don't help much
- Increased path length, notably on the host