Securing Untrusted Workloads

with Kata Containers on Kubernetes

DAVID ANGOT AND ALEX PRICE

Kubernetes at Atlassian

3 years in production

Back then very few tools were available to build clusters. We've done it "The hard way".

Dedicated K8s team

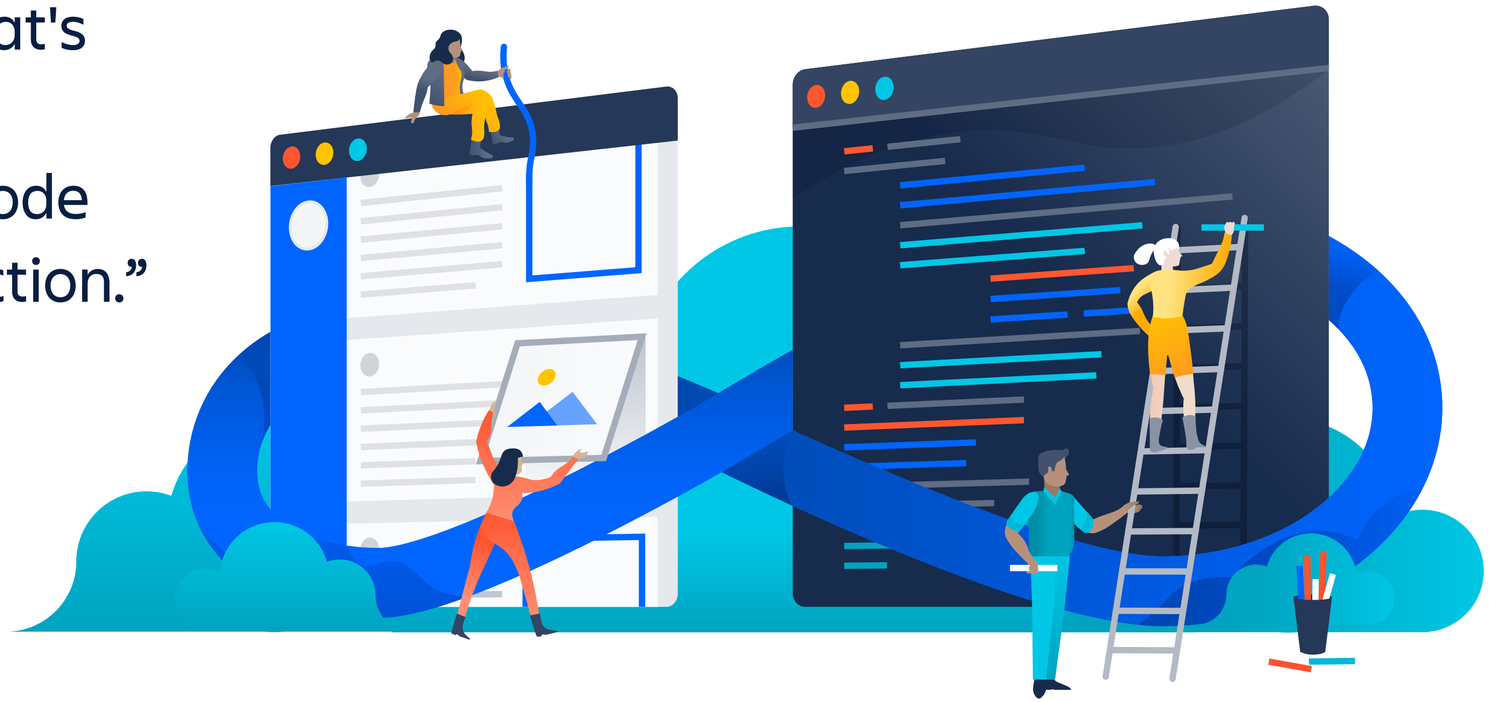
Our team is the one-stop shop for Kubernetes and containers in the company.

50+ clusters and growing

We try to avoid cluster sprawl, but multi-tenanted is not always possible.

Bitbucket Pipelines & Deployments

“Integrated CI/CD for Bitbucket Cloud that's trivial to set up, automating your code from test to production.”



BITBUCKET PIPELINES & DEPLOYMENTS

Run K8s

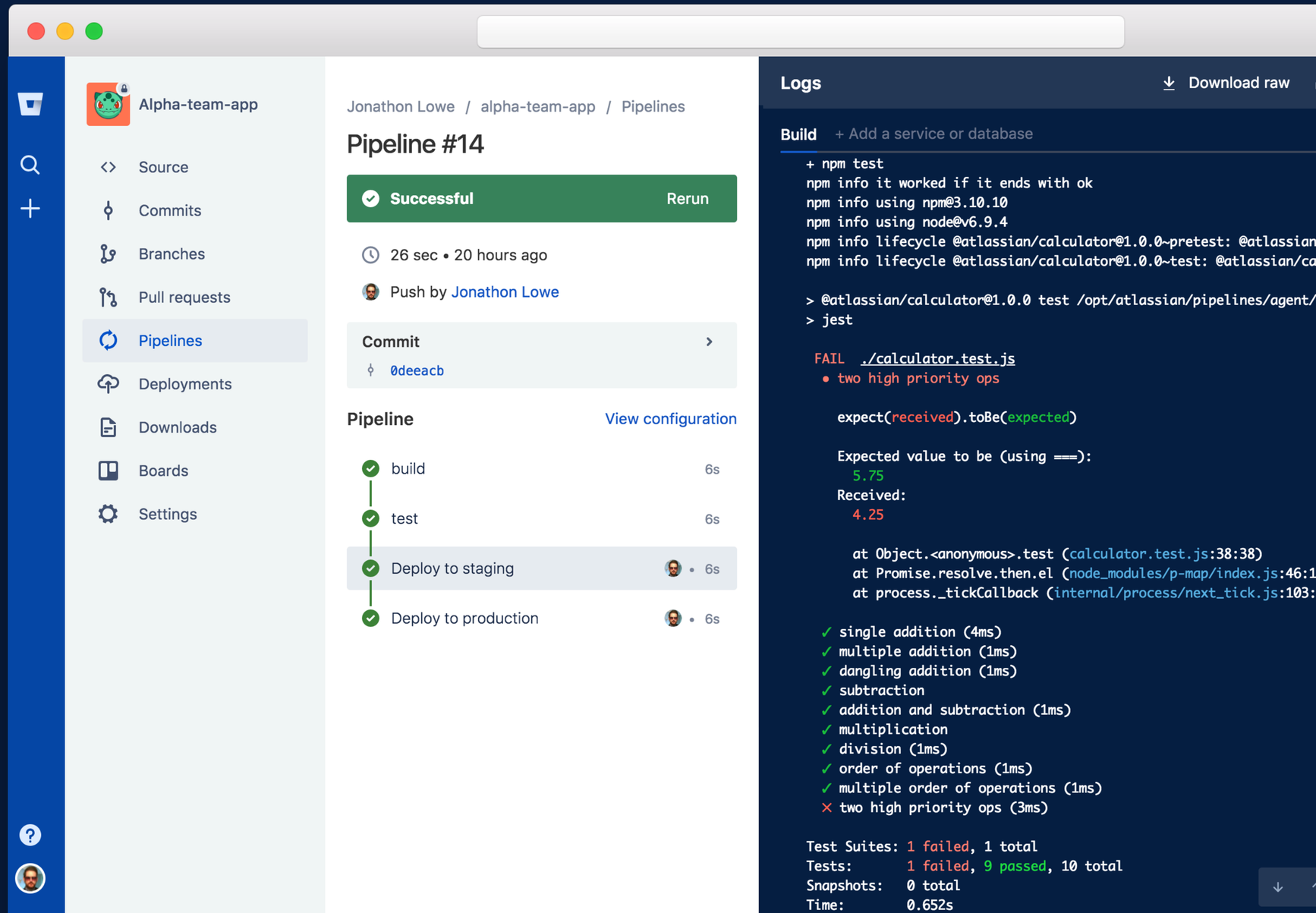
Each pipeline step is a pod, and run co-located on a pool of nodes.

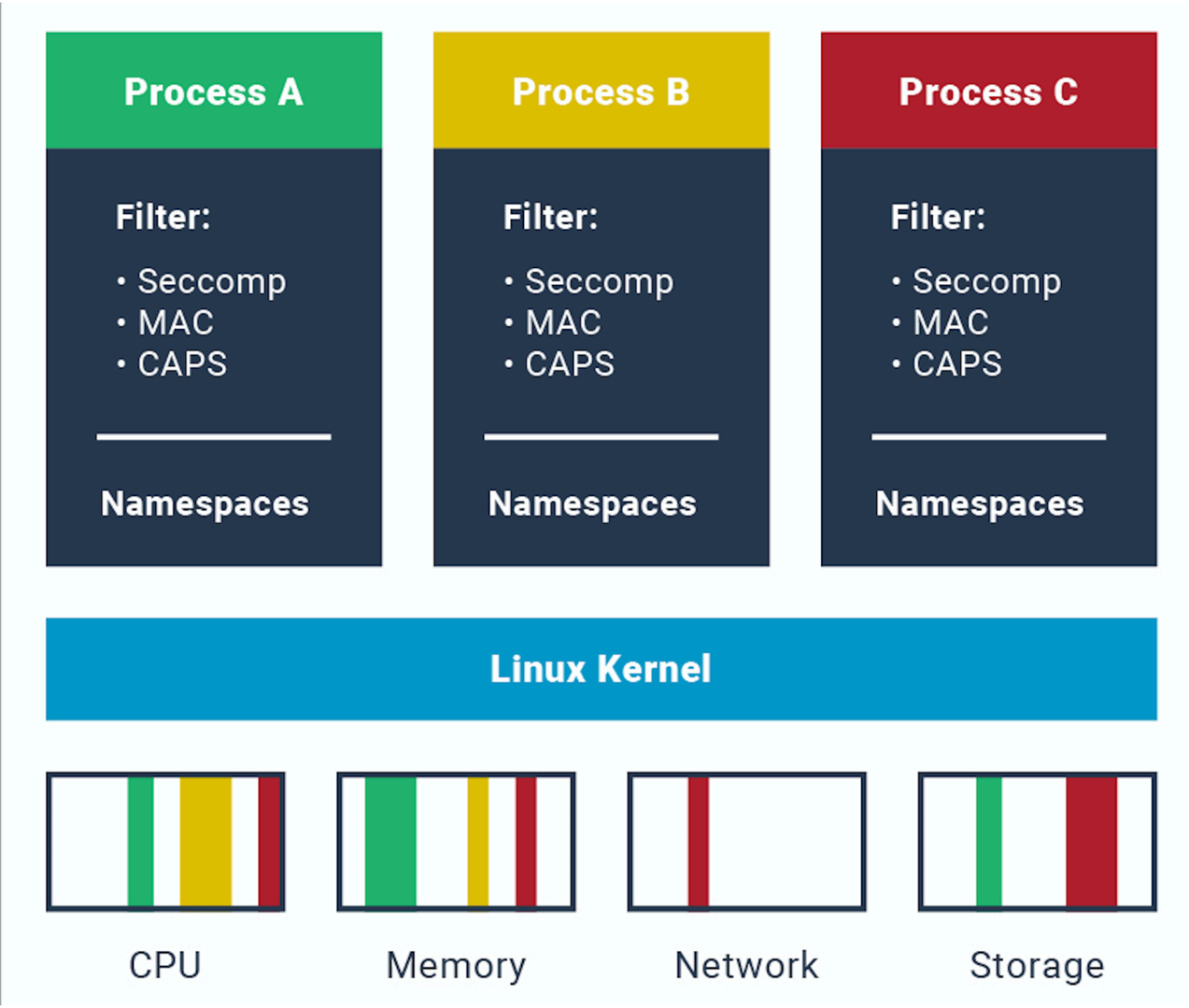
Hard multi-tenancy

Assume everything to be malicious and isolate pipelines builds from each other.

Security priority #0

Securing untrusted workload is challenging. We are always looking for improvements.





Traditional containers isolation :

Kernel namespaces
Cgroups

Additional Isolation / protection

Network Policies

Calico, Cilium, and many other tools provide network isolation for containers.

SECCOMP / Capabilities

Control what syscalls or linux capabilities are allowed.

SELinux

A well configured SELinux can limit the damage from a container exploit.

Unprivileged users

Prevent container running as root, and/or re-map users.

A shared kernel means that container escape is possible with a single vulnerability.

docker-in-docker requirements

Root user, privileged mode, extra capabilities. This makes security even more challenging.

Unpatched CVEs

Recent exploits have made container escape possible. Patching the systems is critical.

Extra layer of security is required

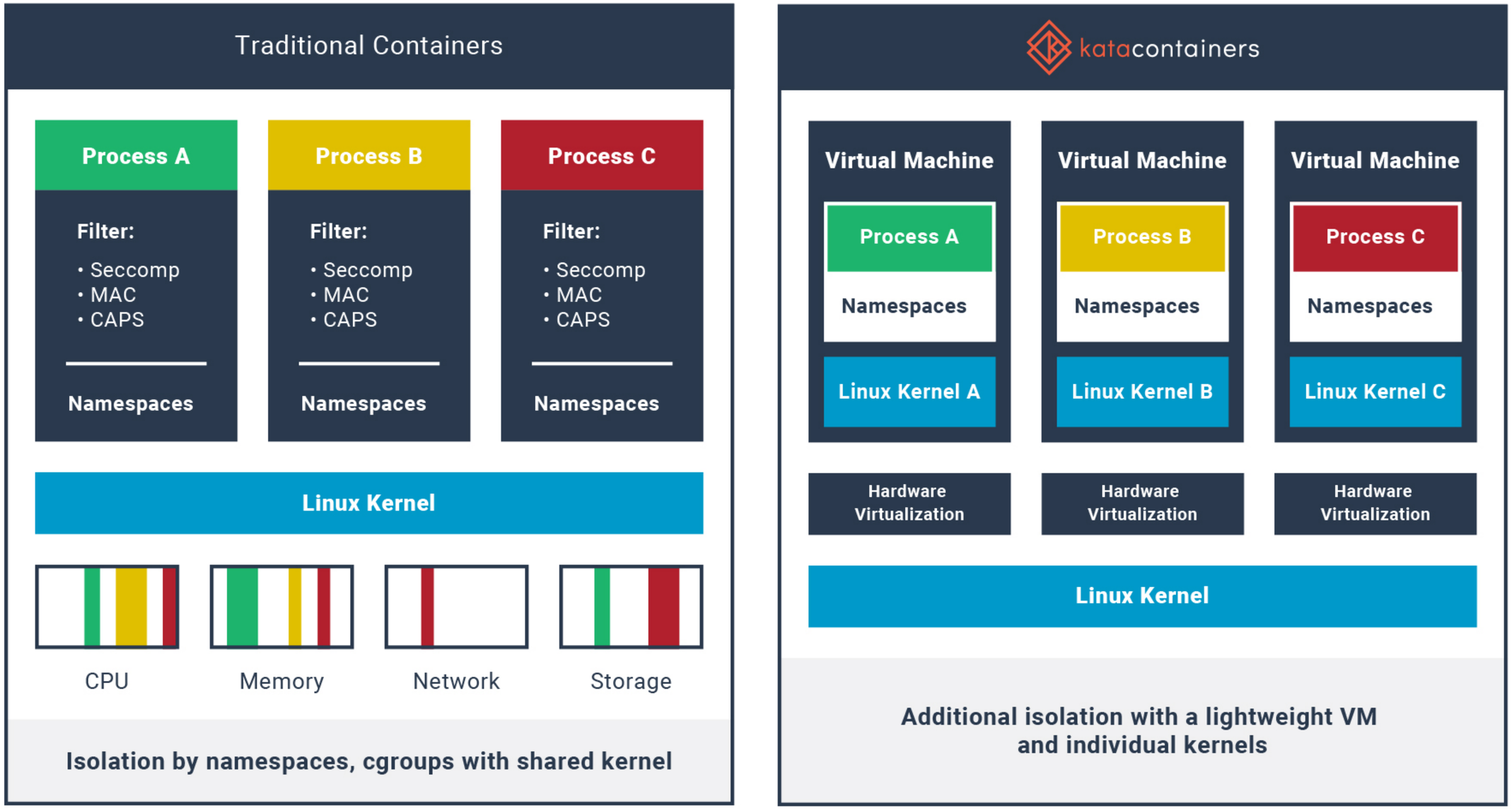
The above is why we need an extra layer of security on top of the container isolation.

“

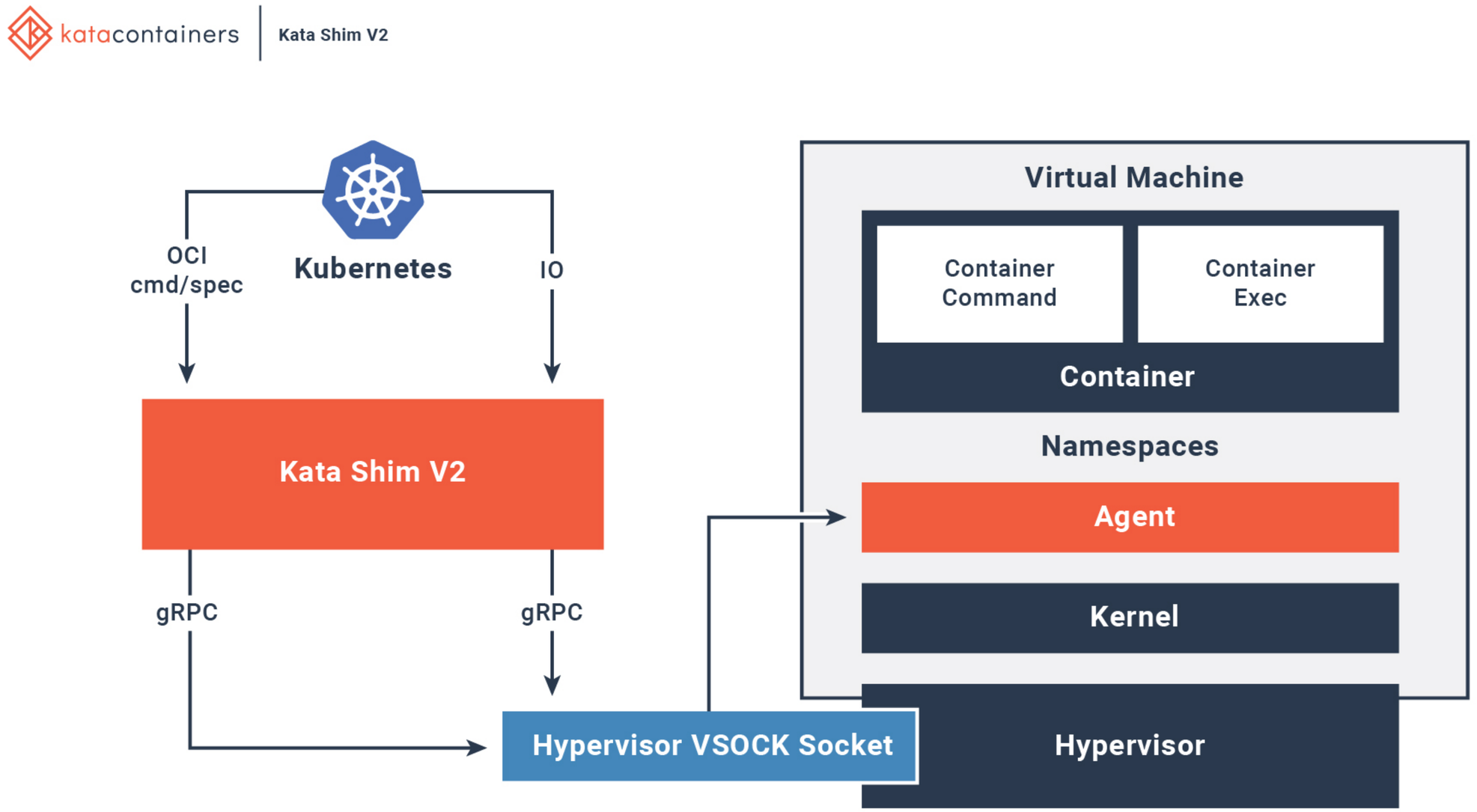
KATA : The speed of containers, the security of VMS.

KATA CONTAINERS

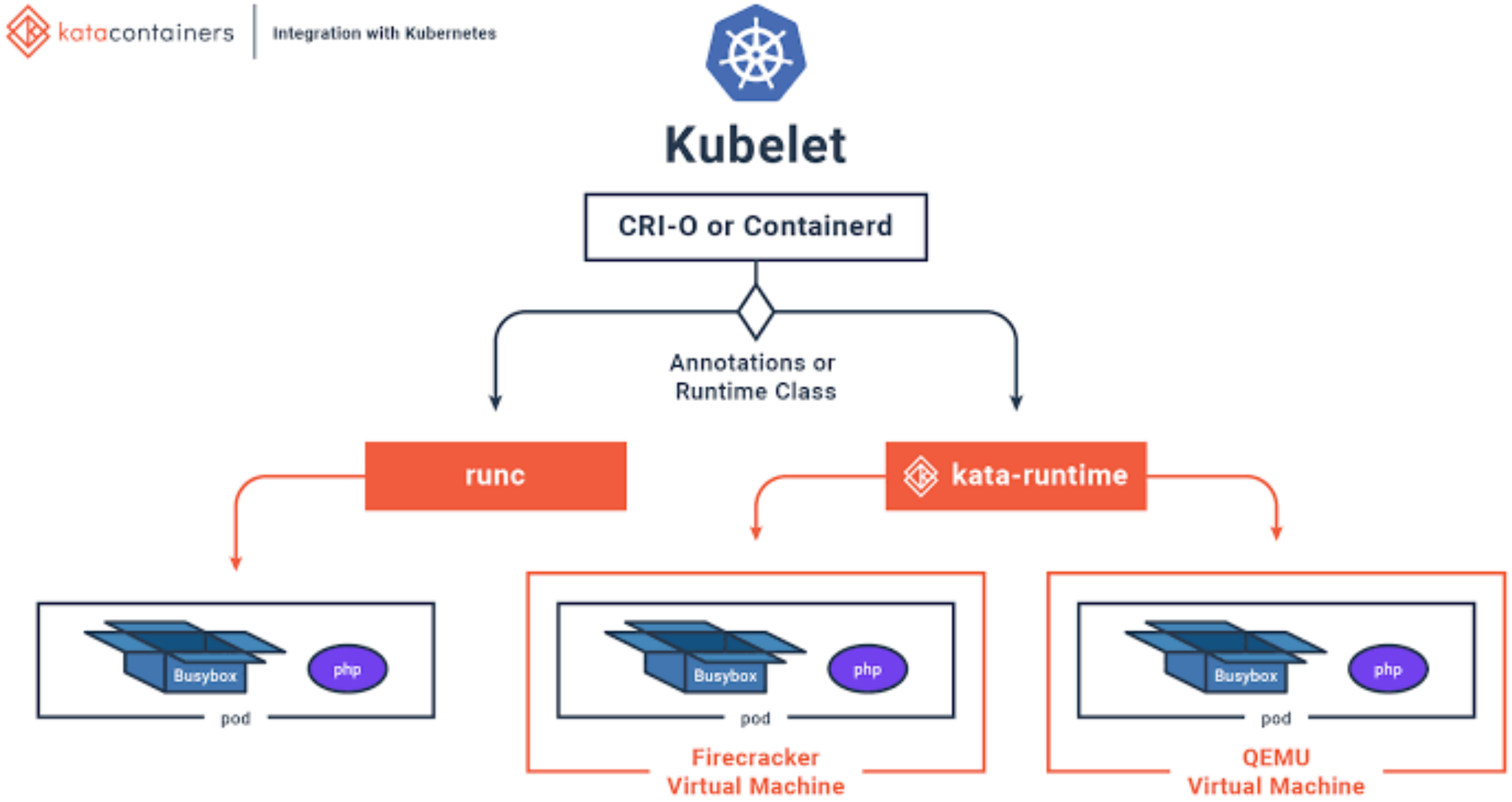
Kata container isolation



Kata Architecture



How does it work in Kubernetes



OUR PROOF OF CONCEPT WAS DONE IN 1 WEEK.

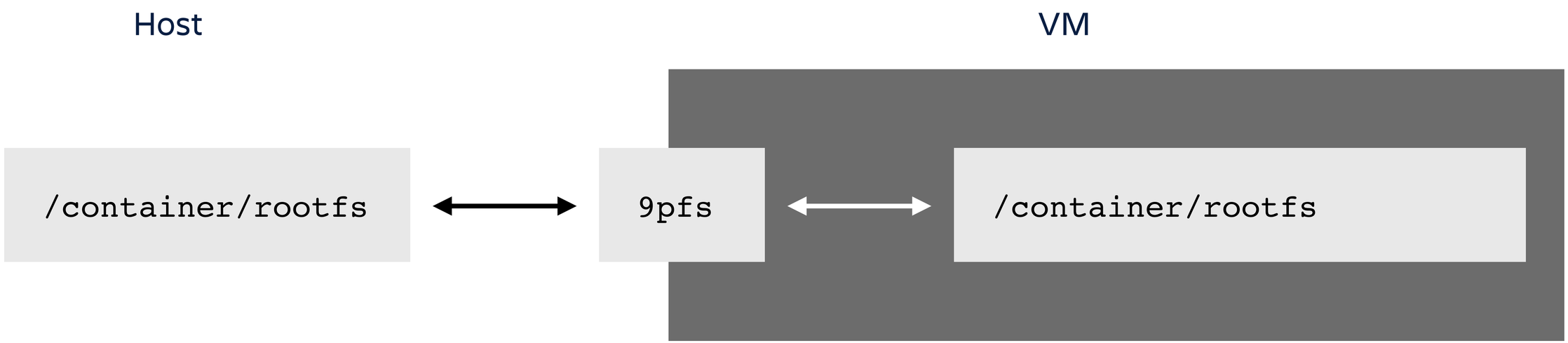
PRODUCTION 6+ MONTHS LATER.

Challenge #1: Disk performance

ROOTFS SHARING

9pfs

By default Kata uses 9pfs to share rootfs, volumes with container inside VM



BENCHMARKS

9pfs

```
sync; dd if=/dev/zero of=./tempfile bs=1M count=8192; sync
8589934592 bytes (8.6 GB, 8.0 GiB) copied, 97.2037 s, 88.4 MB/s
```

BENCHMARKS

9pfs

```
sync; dd if=/dev/zero of=./tempfile bs=1M count=8192; sync
8589934592 bytes (8.6 GB, 8.0 GiB) copied, 97.2037 s, 88.4 MB/s
```

My MacBook

```
sync; gdd if=/dev/zero of=./tempfile bs=1M count=8192; sync
8589934592 bytes (8.6 GB, 8.0 GiB) copied, 4.64459 s, 1.8 GB/s
```

SOLUTION

Old school tech to the rescue

Kata allows us to switch to a block based medium which we can then plug into the VM

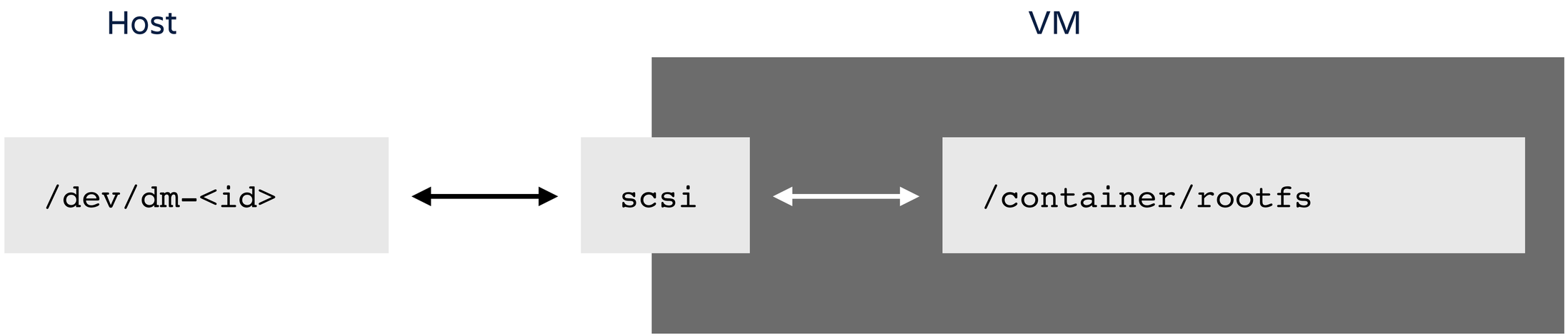
SOLUTION

Old school tech to the rescue

Kata allows us to switch to a block based medium which we can then plug into the VM

NVMe Disks

Our solution utilises an array of super fast NVMe SSD disks



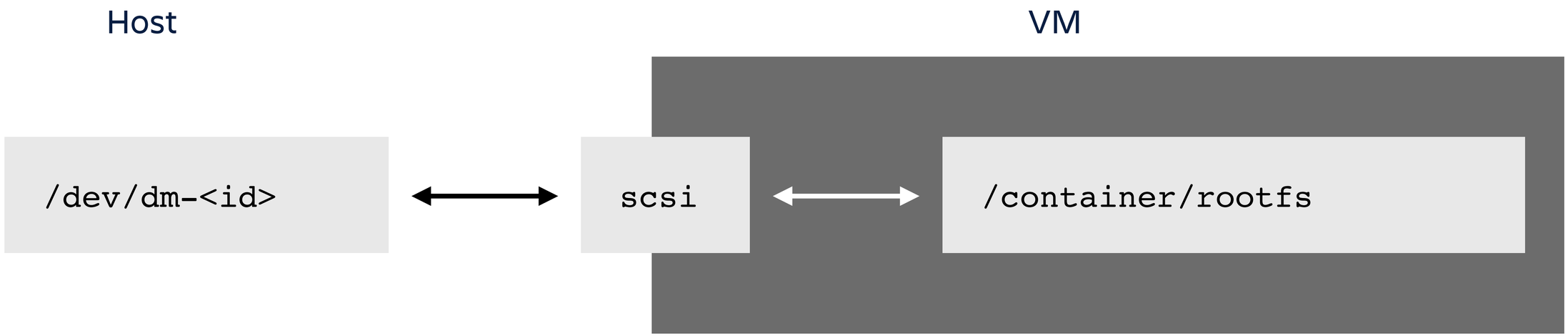
SOLUTION

Old school tech to the rescue

Kata allows us to switch to a block based medium which we can then plug into the VM

NVMe Disks

Our solution utilises an array of super fast NVMe SSD disks




```
8589934592 bytes (8.6 GB, 8.0 GiB) copied, 7.04309 s, 1.2 GB/s
```

Challenge #2: Privileged Containers

PRIVILEGED CONTAINERS

```
apiVersion: v1
kind: Pod
metadata:
  name: privileged
spec:
  containers:
    - name:  example
      image: example:1.2.3
      securityContext:
        privileged: true
```

From https://github.com/containerd/cri/blob/master/pkg/server/container_create_unix.go

INSIDE VS OUTSIDE

Host

```
[ip-10-1-40-46 ~ # uname -a
 Linux ip-10-1-40-46.us-west-2.compute.internal 4.19.68-coreos
[ip-10-1-40-46 ~ # ls -lh /dev | grep "dm-"
 brw-rw----. 1 root disk    254,   0 Nov 23 18:41 dm-0
 brw-rw----. 1 root disk    254,   1 Nov 23 18:42 dm-1
 brw-rw----. 1 root disk    254,  10 Nov 23 18:44 dm-10
 brw-rw----. 1 root disk    254,  11 Nov 23 18:44 dm-11
 brw-rw----. 1 root disk    254,  12 Nov 23 18:44 dm-12
 brw-rw----. 1 root disk    254,  13 Nov 23 18:44 dm-13
 brw-rw----. 1 root disk    254,  14 Nov 23 18:44 dm-14
 brw-rw----. 1 root disk    254,  15 Nov 23 18:44 dm-15
 brw-rw----. 1 root disk    254,  16 Nov 23 18:44 dm-16
 brw-rw----. 1 root disk    254,  17 Nov 23 18:44 dm-17
 brw-rw----. 1 root disk    254,  18 Nov 23 18:44 dm-18
```

INSIDE VS OUTSIDE

Host

```
[ip-10-1-40-46 ~ # uname -a
 Linux ip-10-1-40-46.us-west-2.compute.internal 4.19.68-coreos
[ip-10-1-40-46 ~ # ls -lh /dev | grep "dm-"
 brw-rw----. 1 root disk    254,   0 Nov 23 18:41 dm-0
 brw-rw----. 1 root disk    254,   1 Nov 23 18:42 dm-1
 brw-rw----. 1 root disk    254,  10 Nov 23 18:44 dm-10
 brw-rw----. 1 root disk    254,  11 Nov 23 18:44 dm-11
 brw-rw----. 1 root disk    254,  12 Nov 23 18:44 dm-12
 brw-rw----. 1 root disk    254,  13 Nov 23 18:44 dm-13
 brw-rw----. 1 root disk    254,  14 Nov 23 18:44 dm-14
 brw-rw----. 1 root disk    254,  15 Nov 23 18:44 dm-15
 brw-rw----. 1 root disk    254,  16 Nov 23 18:44 dm-16
 brw-rw----. 1 root disk    254,  17 Nov 23 18:44 dm-17
 brw-rw----. 1 root disk    254,  18 Nov 23 18:44 dm-18
```

Kata Container

```
[root@aprice-test:~# uname -a
 Linux aprice-test 4.19.75 #1 SMP Wed Nov 6 10:02:51 PST 2
[root@aprice-test:~# ls -lh /dev | grep "dm-"
 b--------- 1 root disk   8,  32 Nov 23 18:49 dm-0
 b--------- 1 root disk   8,  48 Nov 23 18:49 dm-1
 b--------- 1 root disk   8,  64 Nov 23 18:49 dm-10
 b--------- 1 root disk   8,  80 Nov 23 18:49 dm-11
 b--------- 1 root disk   8,  96 Nov 23 18:49 dm-12
 b--------- 1 root disk   8, 112 Nov 23 18:49 dm-13
 b--------- 1 root disk   8, 128 Nov 23 18:49 dm-14
 b--------- 1 root disk   8, 144 Nov 23 18:49 dm-15
 b--------- 1 root disk   8, 160 Nov 23 18:49 dm-16
 b--------- 1 root disk   8, 176 Nov 23 18:49 dm-17
 b--------- 1 root disk   8, 192 Nov 23 18:49 dm-18
```

ACCESSING OTHER CONTAINERS

```
[root@aprice-test:~# mkdir /foobar
[root@aprice-test:~# mount /dev/dm-25 /foobar
[root@aprice-test:~# lsblk | grep /foobar
sdu         65:64   0    32G  0 disk /foobar
[root@aprice-test:~# ls -lh /foobar/app
total 32K
-rw-r--r--.   1 root root  226 Nov 23 18:21 Dockerfile
-rw-r--r--.   1 root root 1.4K Nov 23 18:21 app.js
drwxr-xr-x.   2 root root 4.0K Nov 23 18:21 bin
drwxr-xr-x. 149 root root 4.0K Nov 23 18:24 node_modules
-rw-r--r--.   1 root root  360 Nov 23 18:21 package.json
drwxr-xr-x.   3 root root 4.0K Nov 23 18:21 public
drwxr-xr-x.   2 root root 4.0K Nov 23 18:21 routes
drwxr-xr-x.   2 root root 4.0K Nov 23 18:21 views
root@aprice-test:~#
```

WHAT NOW?

awprice commented on 6 Aug Contributor +😀 •••

This commit adds a flag to the runtime config that allows overloading of the default privileged behaviour. When the flag is enabled on a runtime, host devices won't be appended to the runtime spec if the container is run as privileged.

By default the flag is false to maintain the current behaviour of privileged.

Fixes #1213

Signed-off-by: Alex Price aprice@atlassian.com

🎉 3

Privileged in Kata

Don’t assume isolation

We assumed virtual machines would isolate workloads, but not when we used privileged

Audit container resources

Perform an audit of the resources shared with the Kata containers before going live

Finer grain control

We are interested in finer grain control when granting more privileges to containers with Kubernetes

Challenge #3: Monitoring

CONTAINER METRICS & RUNC

Cadvisor

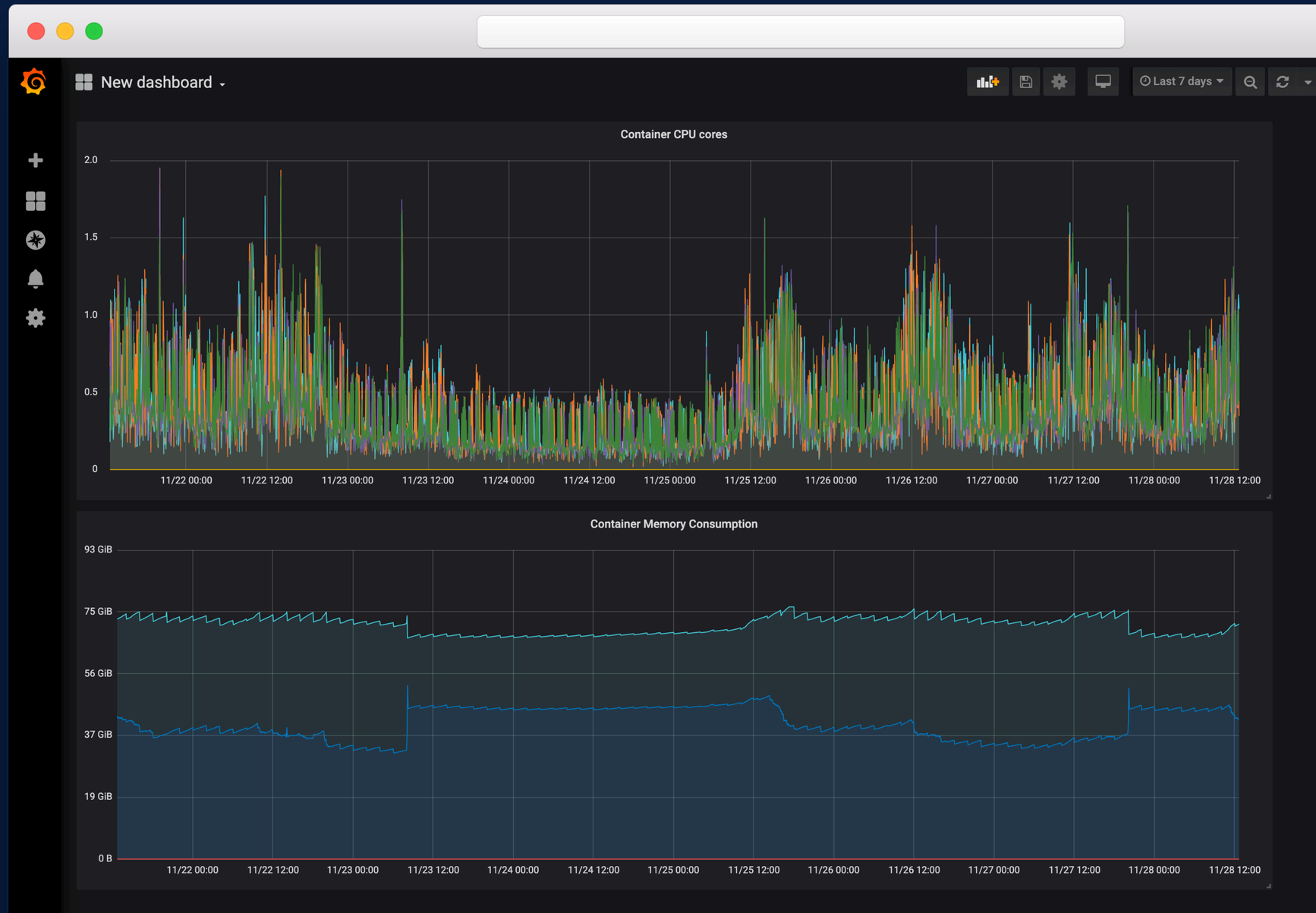
Metrics are automatically collected by Cadvisor and available from the Kubelet

Cgroups

Cadvisor uses the pod cgroups to get metrics for each container

Accurate

Cgroup reported metrics are incredibly accurate



CONTAINER METRICS & KATA

Cadvisor

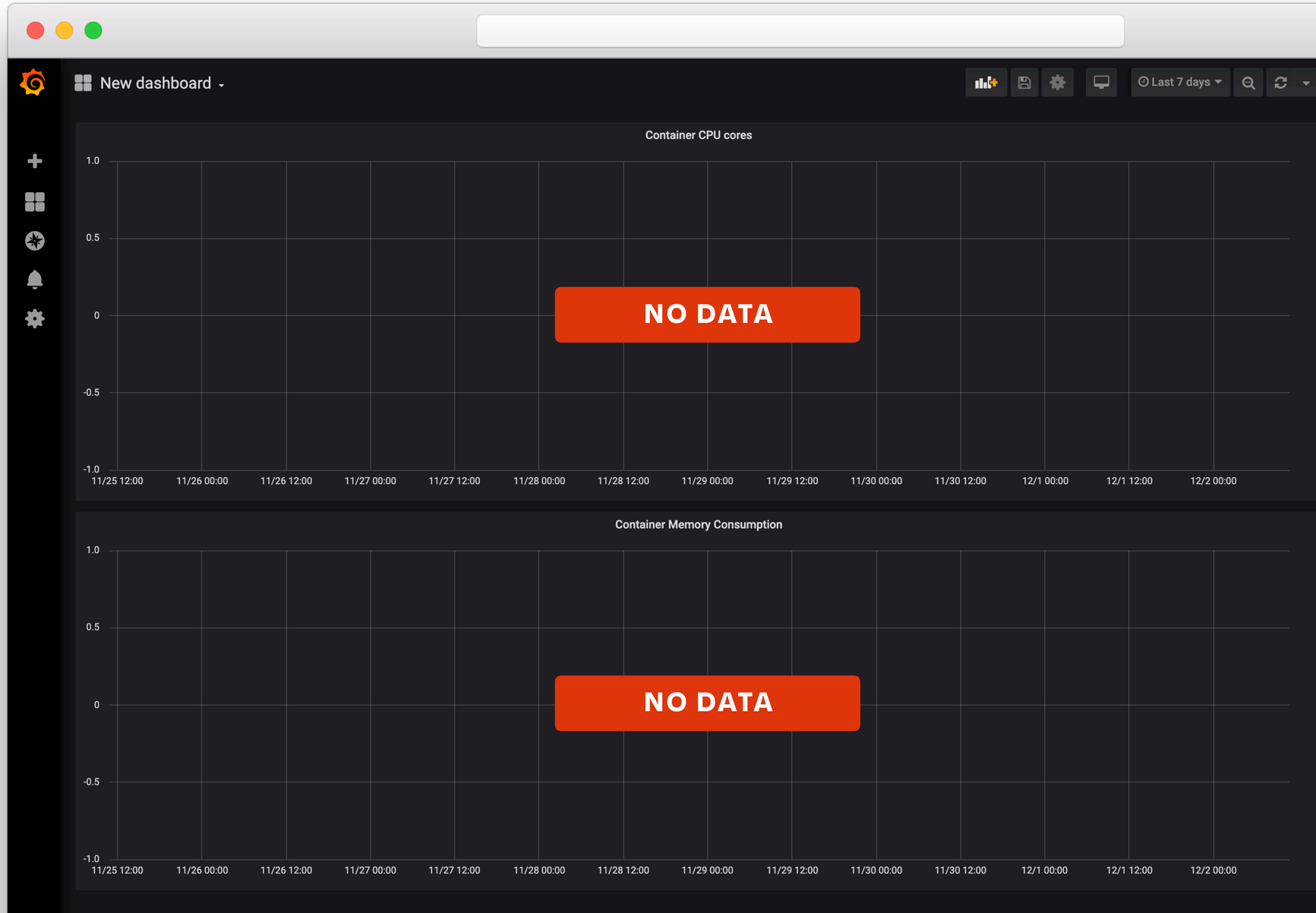
Metrics are automatically collected by Cadvisor and available from the Kubelet

Cgroups

Pod cgroups on host no longer contain the container processes, just the single VM process

Inaccurate

Pod cgroup on host metrics only have virtual machine metrics, not fine grained enough



KUBELET RESOURCE METRICS ENDPOINT

Runtime aware

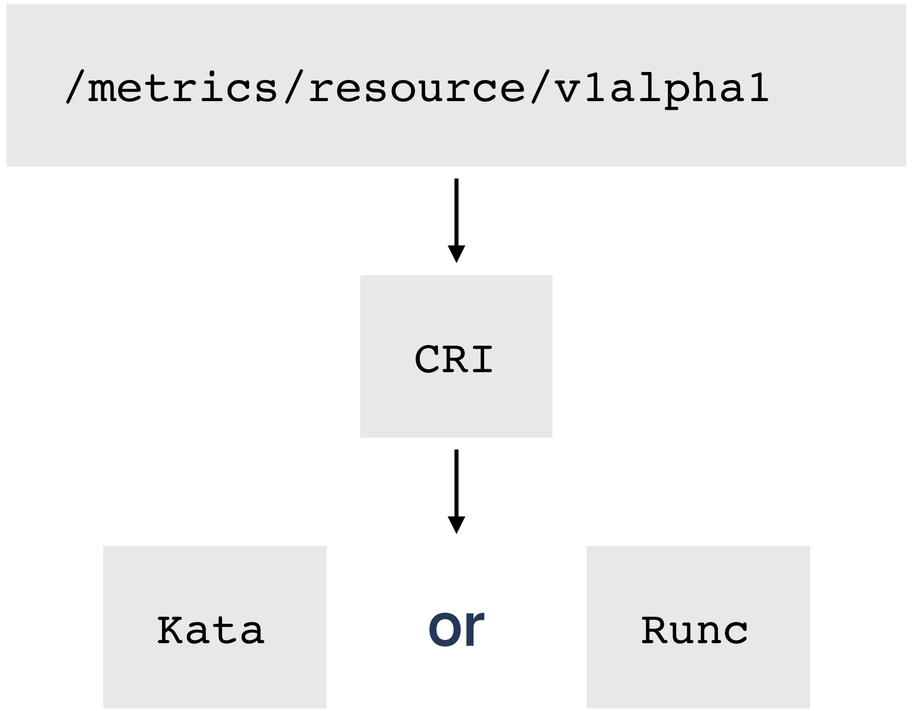
Kubelet delegates fetching of resource metrics from the runtime

Available in 1.14

Endpoint Is alpha in 1.14

Runc + Kata

Endpoint works seamlessly with both regular containers and virtual machines



From https://github.com/kubernetes/enhancements/issues/727

Whats next?

Autoscaling

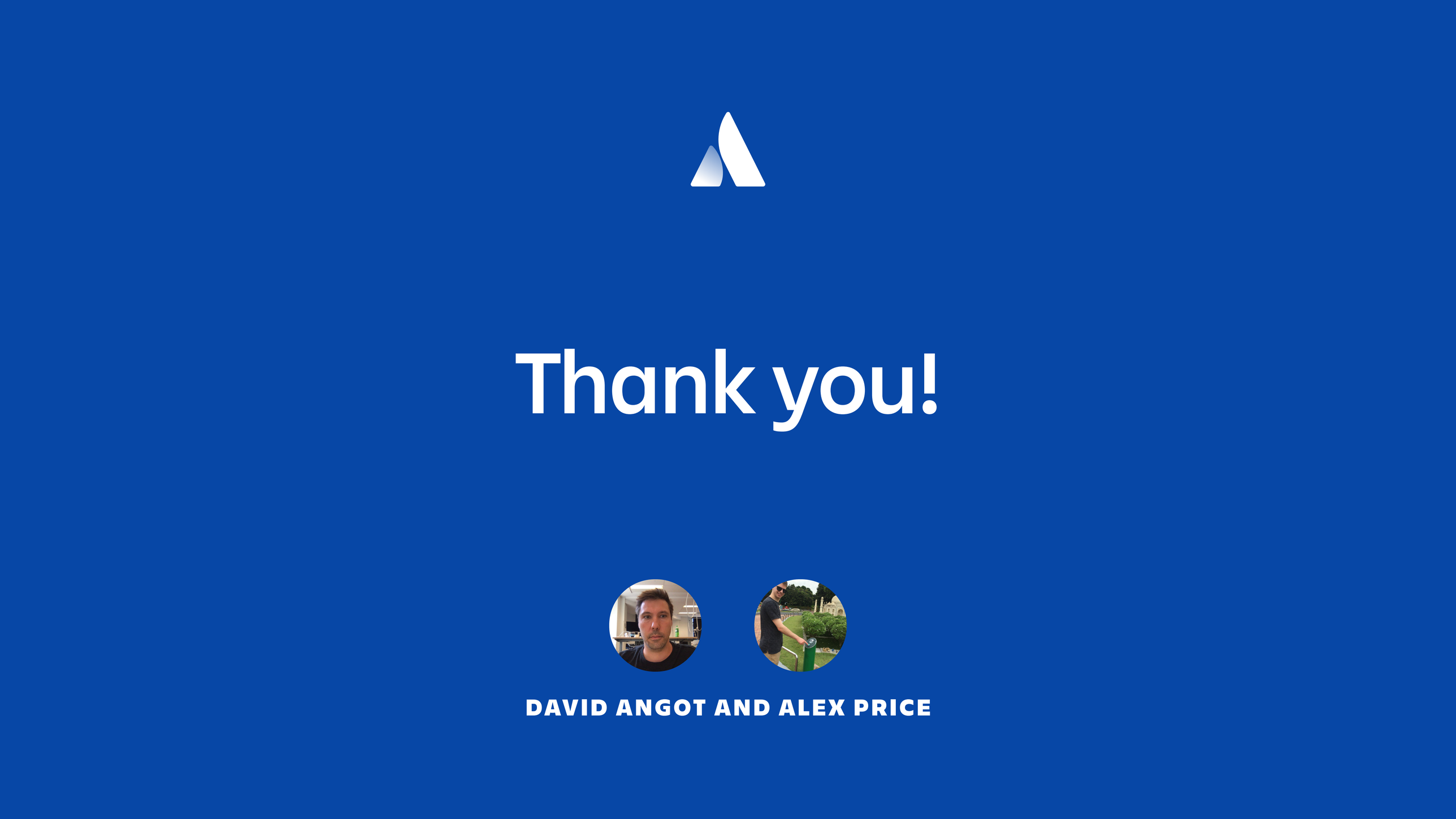
Autoscaling is tricky with such large, metal instances - wait times can be long

Annotations

New annotations in Kata allow for modifying VM configuration

Firecracker

Firecracker is an excellent candidate for isolated workloads, but does not support Kubernetes


Thank you!
DAVID ANGOT AND ALEX PRICE