Architecture Document - Multi-tenant Secure Cloud Native Platform

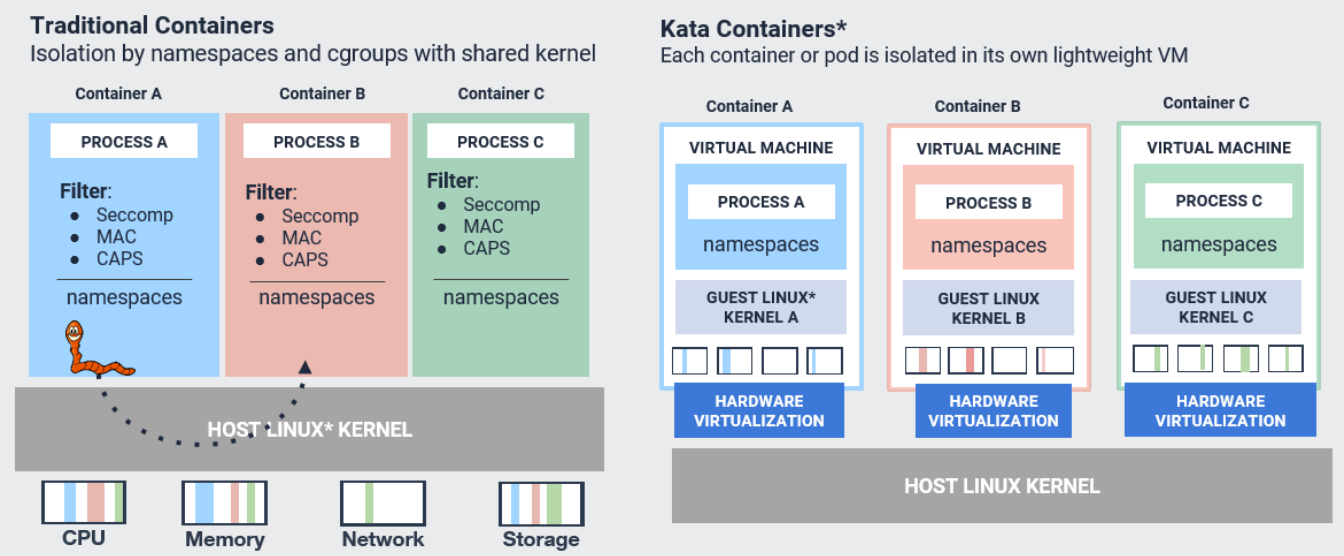
Overview

Kata Containers is an OCI-compatible container runtime that uses lightweight virtual machines (VM) to improve the isolation of workloads running inside a container. Kata Containers works seamlessly with Kubernetes through Containerd and CRI-O. It supports different architectures (including, but not limited to x86, ARM, and IBM Power) and works with different hypervisors, such as Cloud Hypervisor, and Firecracker*.

The Kata Containers project is an open-source project under the Open Infrastructure Foundation. For more information on the project and its community, you can check the Kata Containers web page and its GitHub repository.

How Kata works

In addition to the isolation capabilities that traditional containers provide, Kata Containers uses lightweight VMs to run containers. Each container runs in its own VM with a dedicated guest Kernel and guest image.



Worm image: https://clip.cookdiary.net/sites/default/files/wallpaper/worms-clipart/80947/worms-clipart-transparent-background-80947-2261480.png
* Other names and brands may be claimed as the property of others.

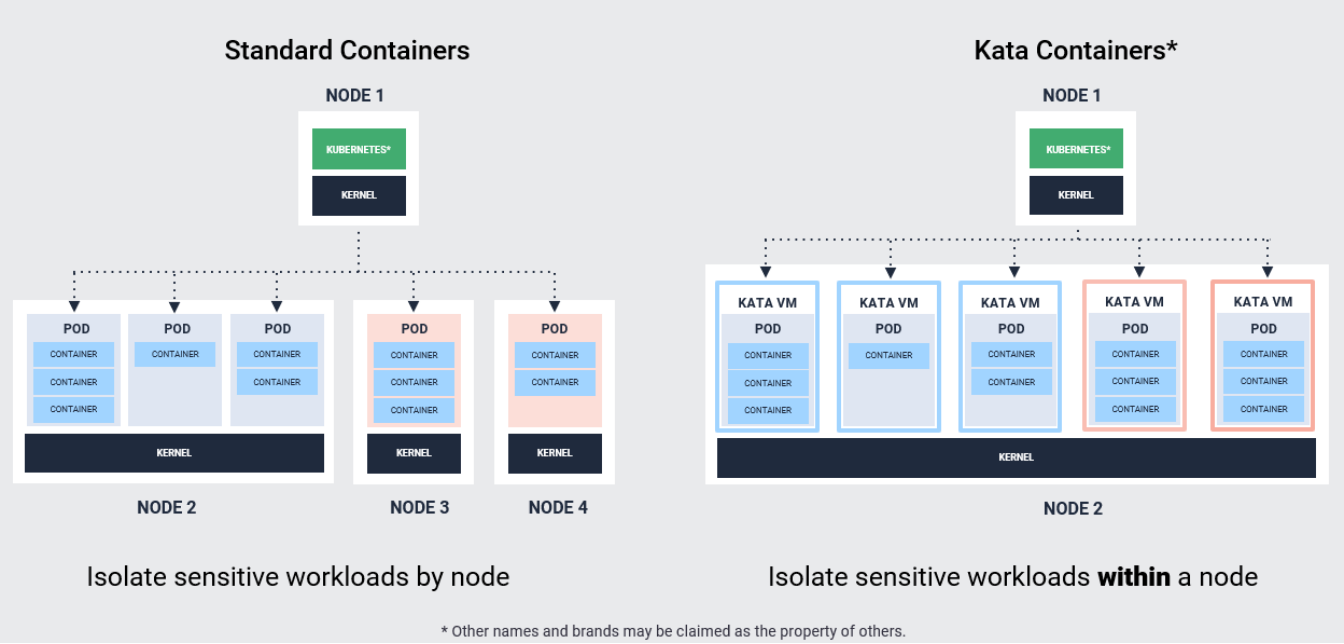
Kata Containers provides additional workload isolation with minimal performance cost. It provides default Kernel and guest image, which are highly optimized to minimize boot-time and memory footprint.

Challenges of current multitenancy Kubernetes environments

One of the main challenges in Kubernetes is how to improve the isolation between workloads of different Tenants. Some practices that you can implement are the use of namespaces or assign different worker nodes (physical or virtualized) to different tenants. When using different namespaces, you are still sharing the same kernel (host kernel) on all workloads, so any malicious pod that takes advantage of a kernel vulnerability, could affect other workloads or the node itself. Using different worker nodes for different tenants may lead to inefficient usage of resources in your infrastructure.

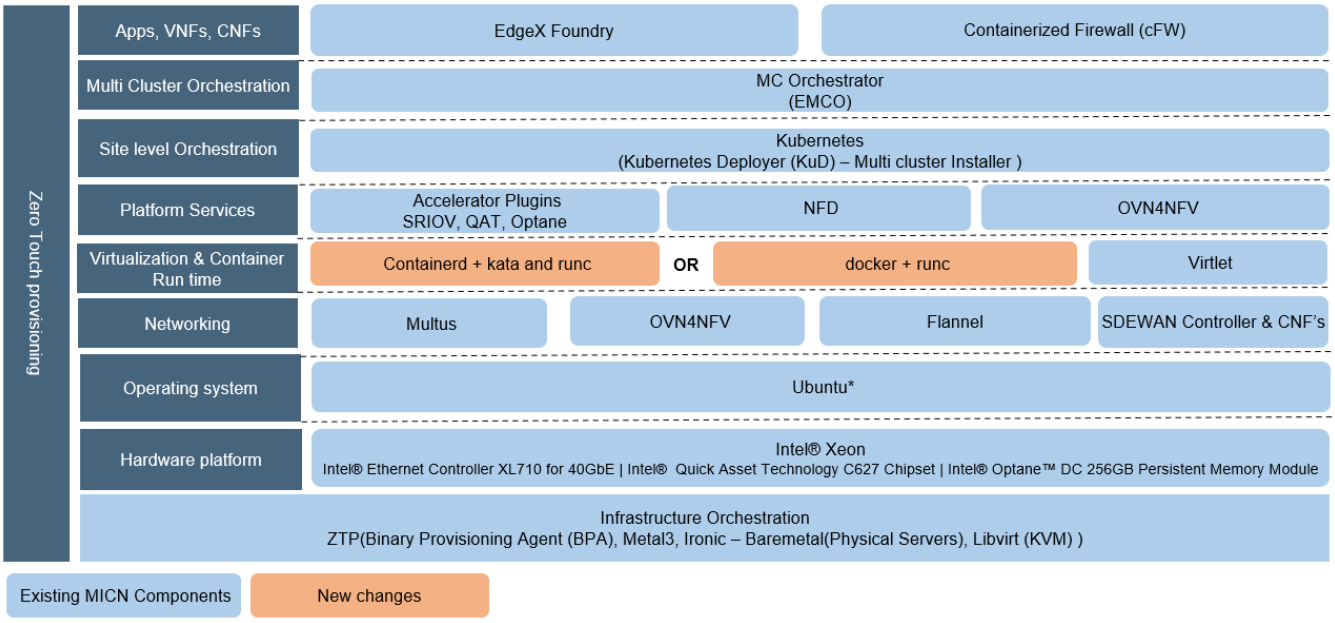
How Kata can help

Kata Containers helps to solve this inefficiency by providing VM level isolation of workloads within a node leading to greater flexibility in multitenant scenarios. Using Kata Containers, you'll enhance the multitenant capabilities of your Kubernetes environment because it will allow you to run trusted and untrusted workloads on the same node. Untrusted workloads will be isolated in VMs that do not share the kernel with other workloads or the host, so even if a malicious workload tries to take advantage of a vulnerability in the Kernel, the vulnerability could break out of the container isolation but will still be isolated by the VM.



How Kata Containers is integrated into the Integrated Cloud Native (ICN) stack

In the current Integrated Cloud Native (ICN) stack, Kubernetes uses Docker and dockershim to launch containers using the runc runtime. As Kata Containers was designed with the CRI specification in mind, it does not support dockershim. Because of this, we are introducing Containerd as an alternative CRI runtime in this stack. With Containerd, Kubernetes will be able to launch trusted workloads using runc and untrusted workloads using Kata Containers in the same environment. The image below shows the changes we are making to the current ICN stack.



Kata Containers will be available in the ICN stack through **kata-deploy**. Kata-deploy installs Kata Containers artifacts on the worker nodes of an already deployed Kubernetes cluster and configures it, including containerd with needed roles and runtime classes.

Kubernetes features that will help you manage Kata Containers seamlessly

Runtime Class

RuntimeClass is a Kubernetes feature that allows you to select different container runtimes to run a POD. This is useful as you can choose to run trusted workloads using the default runc runtime and use kata-containers for workloads that need to be executed in isolation. Containerd must also be configured with RuntimeClass so that it knows how to manage the requests. For more information on this feature, please refer to the Kubernetes documentation.

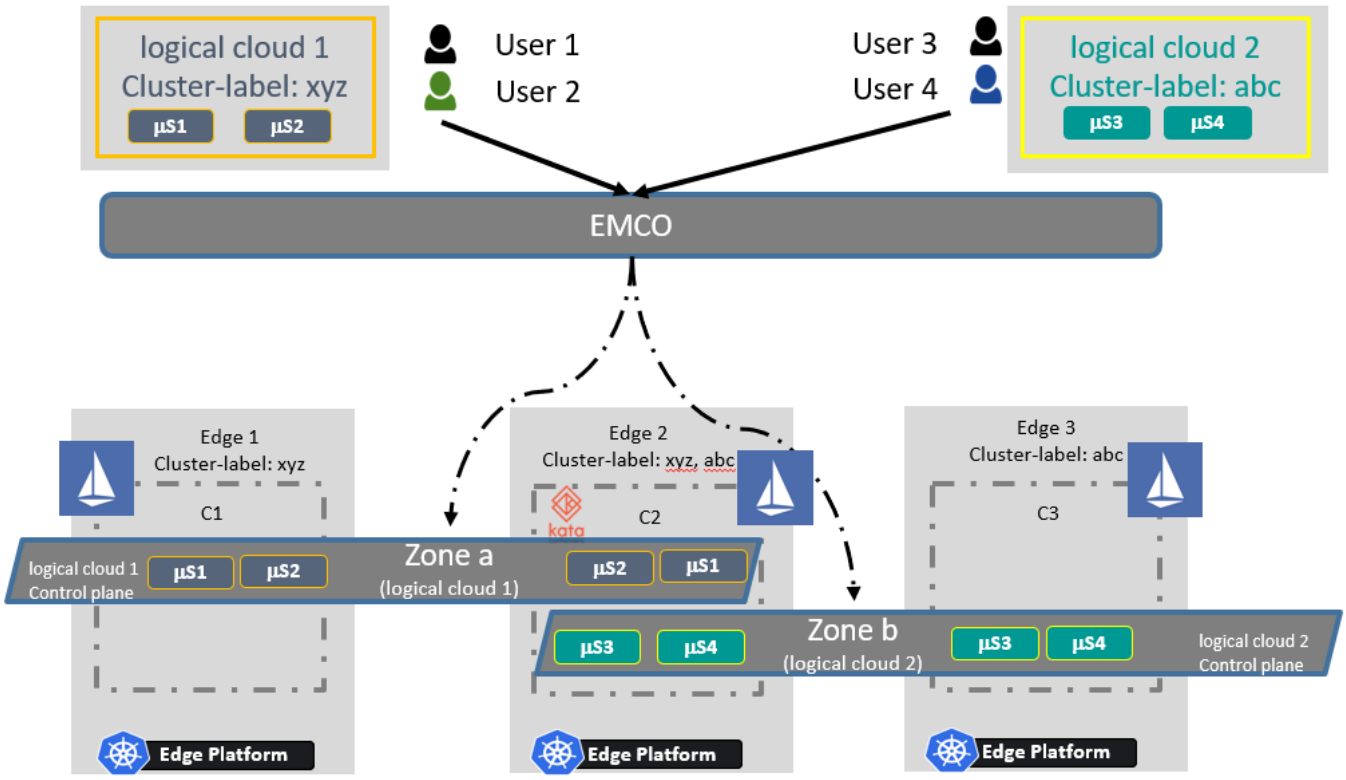
Pod Overhead

PodOverhead is a Kubernetes feature that allows you to specify the number of resources that a container needs in a POD, mainly CPU and memory. As Kata Containers uses lightweight VMs, of the VM to better manage the resources in your Kubernetes environment. For more information on this feature, please refer to the Kubernetes documentation.

In this ICN stack, through kata-deploy, default values for PodOverhead are provided, but the runtime classes could be updated, or new runtime classes could be added with different values to handle different types of workloads.

Hard Multi-Tenancy use cases with Kata Containers in EMCO

The edge multi-cluster orchestrator (EMCO) provides multitenancy capabilities that can be used along with Kata Containers to enhance the isolation for workloads that belong to different tenants. The diagram below shows that EMCO would be able to schedule workloads from different tenants on the same Kubernetes cluster relying on the additional isolation between workloads that Kata Containers provides.


logical cloud 1
Cluster-label: xyz
µS1
µS2
User 1
User 2
User 3
User 4
logical cloud 2
Cluster-label: abc
µS3
µS4
EMCO
Edge 1
Cluster-label: xyz
C1
Edge 2
Cluster-label: xyz, abc
C2
kata
Edge 3
Cluster-label: abc
C3
logical cloud 1
Control plane
µS1
µS2
Zone a
(logical cloud 1)
µS2
µS1
µS3
µS4
Zone b
(logical cloud 2)
µS3
µS4
logical cloud 2
Control plane
Edge Platform
Edge Platform
Edge Platform