



Level Up Your Container Sec Game [Cheat Sheets]

Download our free cheat sheets and master Kubernetes and container security best practices. Get instant access to expert-curated tips, tricks, and essential guidelines to safeguard your containerized environments.

[Download Containers Cheat Sheet](#)

[Download Kubernetes Cheat Sheet](#)

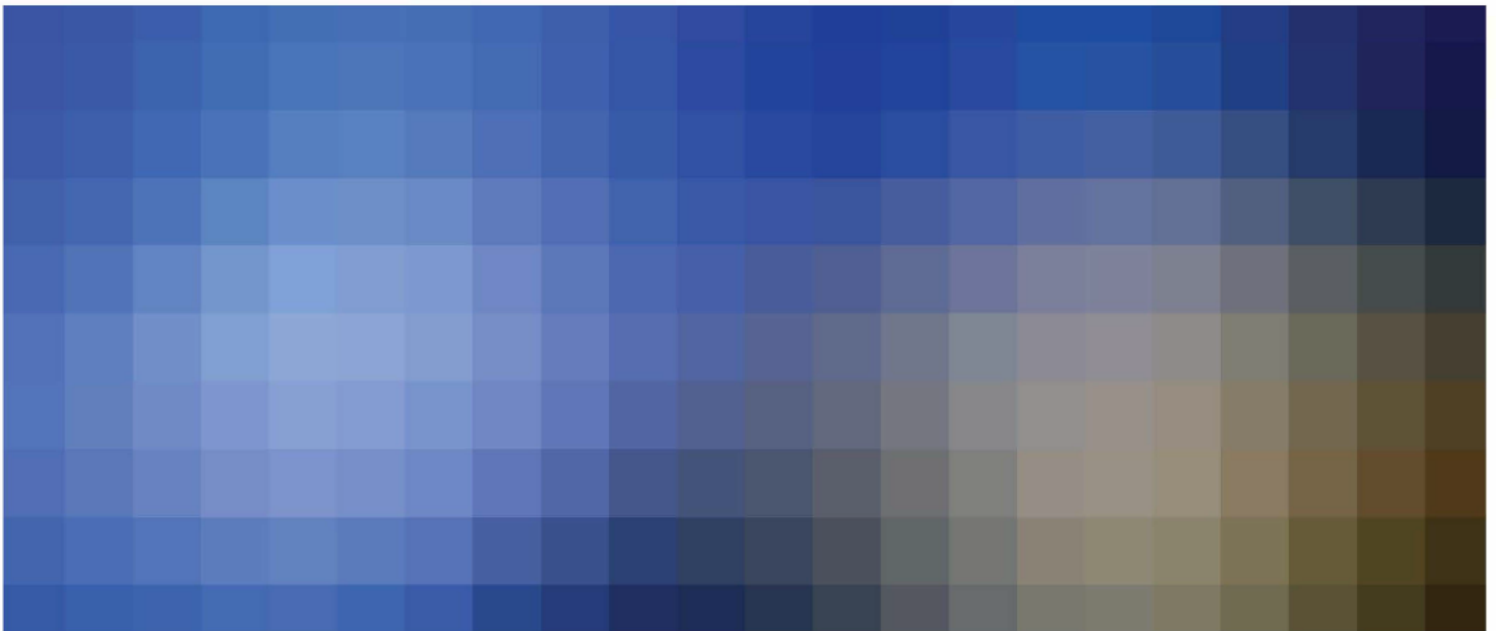
[← All articles](#)

Containers vs. VMs: What's the difference?

In a nutshell, containers and virtual machines (VMs) are two inherently different approaches to packaging and deploying applications/services in isolated environments.

Wiz Experts Team
April 8, 2024

6 minute read



In a nutshell, containers and virtual machines (VMs) are two inherently different approaches to packaging and deploying applications/services in isolated environments.

Isolation came into the mainstream when organizations used physical servers. At that time, it was not possible to define resource limits for applications, which led to many practical flaws, as seen below.



Orca Security Ltd.
Exhibit 2075
Wiz v. Orca
IPR2024-01109, -01190, -01191
Ex. 2075-001

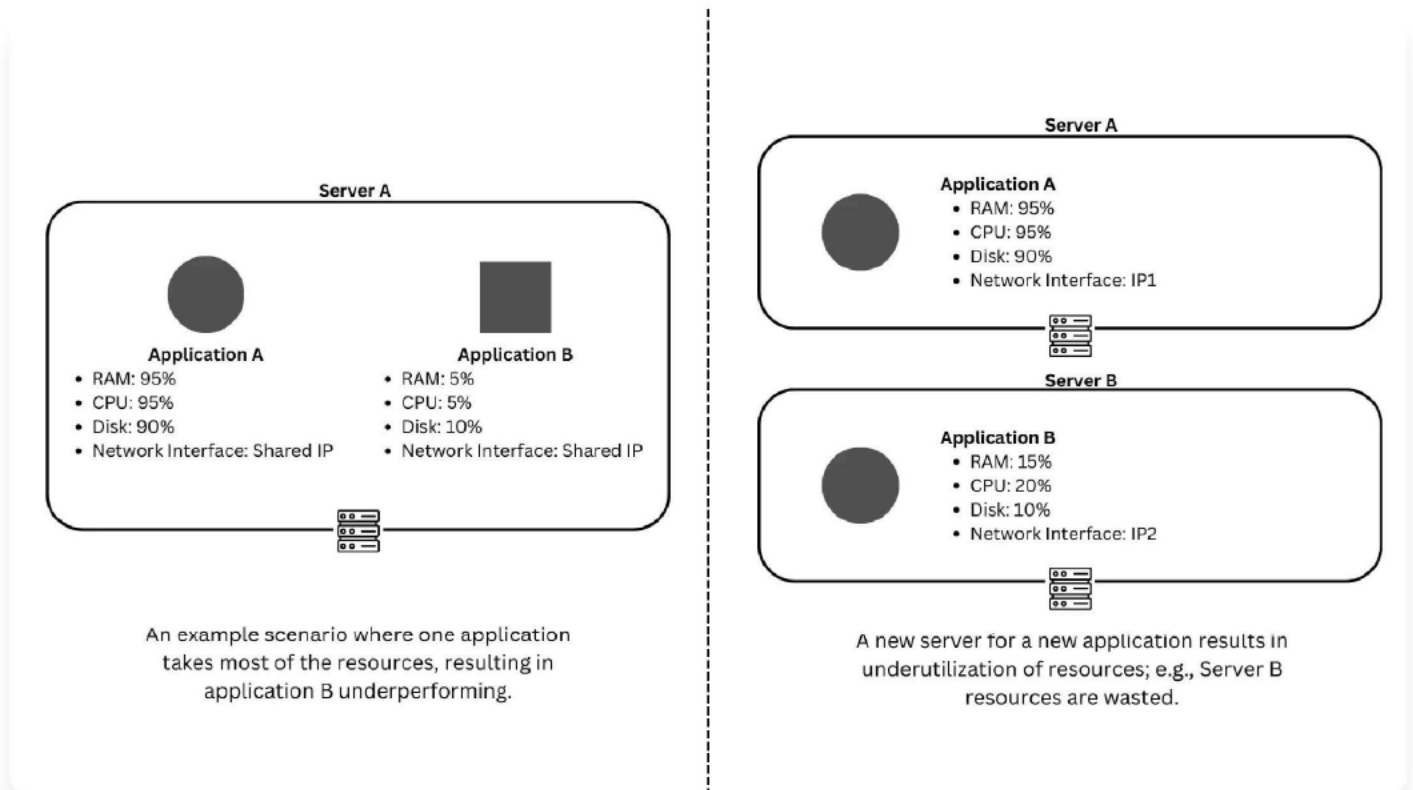


Figure 1: Practical flaws with improper use of resource isolation

Imagine a scenario where multiple applications are running on a single server: One application could end up consuming most of the resources, causing the other apps to underperform. Because of this, it was common to purchase new servers whenever there was a new application. However, the performance requirements of the new application were often unknown, meaning developers would frequently rely on guesswork.

This uncertainty and inefficiency drove the need for a more precise approach to managing resources, which led to the invention of the resource isolation techniques in use today. This post will discuss the distinctions between containers and virtual machines (VMs), examining their roles in resource isolation, efficiency, and security within digital environments.

WIZ ACADEMY

What is Container Security? [Securing Containers 101]

[Read more](#)



Virtualization and containerization in modern computing

Virtualization and **containerization** both entail the abstraction of computing resources from the hardware layer (Figure 2). This allows multiple virtual environments and containers to run parallelly on a single host machine, significantly reducing the need for businesses to purchase new hardware every time they need a new application. This separation of computing resources is the key building block in modern cloud computing.

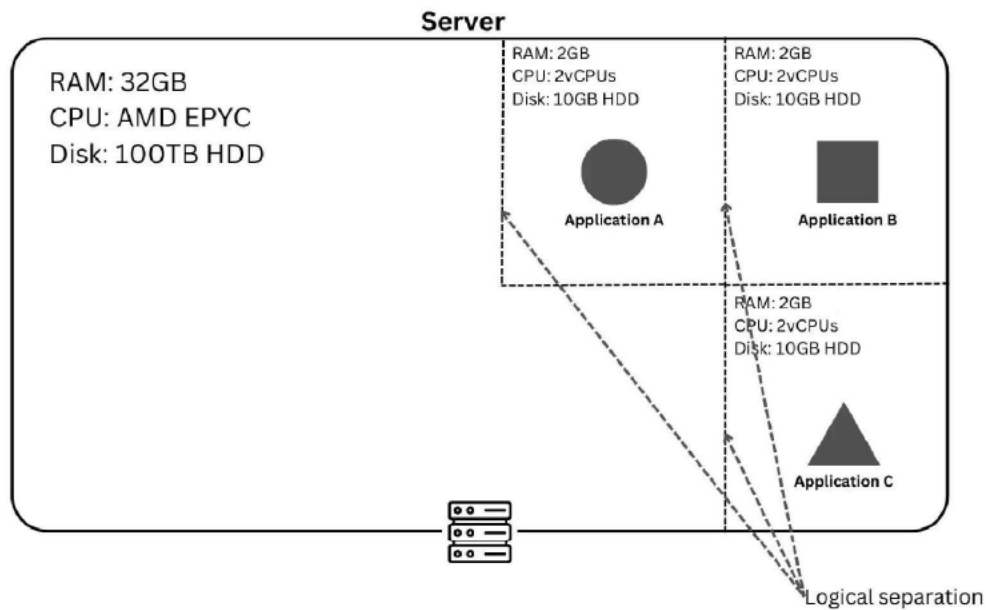


Figure 2: Logical separation of the resources

All existing cloud providers, including [AWS](#), [Azure](#), and [Google](#), use virtualization and containerization under the hood to maximize resource utilization, reduce the cost of ownership, offer more flexibility, etc.

Advanced Container Security Best Practices [Cheat Sheet]

This cheat sheet goes beyond the no-brainer container security best practices and explores advanced techniques that you can put into action ASAP. Use this cheat sheet as a quick reference to ensure you have the proper benchmarks in place to secure your container environments.

[Download now](#)



How VMs work

Virtual machines use virtualization to achieve resource isolation. In virtualization, host machines consist of separate VMs with separate operating systems and dedicated virtual hardware. This level of logical separation is made possible by a hypervisor, a software layer that manages the physical resources and allocates these resources to the VMs as required.

As illustrated below, hypervisors play an important role in the process of emulating underlying hardware.

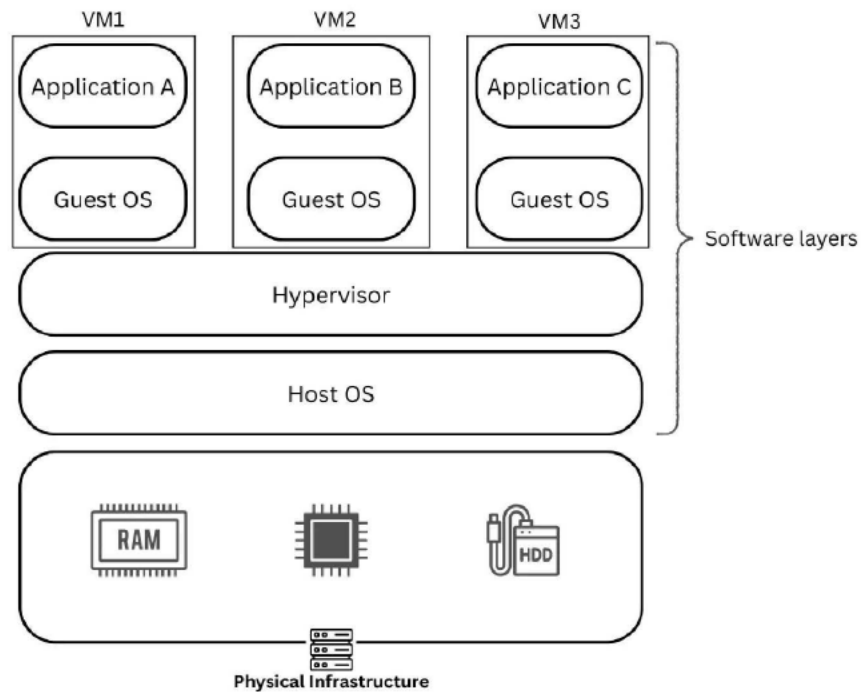


Figure 3: Virtualization

Essentially there are two types of hypervisors: embedded/hosted or bare metal:

- **Embedded/hosted hypervisors:** Run as software applications, requiring an underlying guest operating system to operate. Examples: VMware Workstation and Oracle VM VirtualBox.
- **Bare metal hypervisors:** Do not require an operating system, but instead run directly on physical hardware; faster and more secure than embedded/hosted hypervisors. Examples: VMware ESX/ESXi, Citrix Hypervisor, Red Hat Virtualization (RHV), Kernel-based virtual machine (KVM), and Microsoft Hyper-V.

Hypervisors are designed to monitor and control all the resources while managing VMs, making them a frequent target for hackers. The most recent incident was [CVE-2021-21974](#), a [ransomware attack on the VMware ESXi hypervisor](#). It's important to have a strong understanding of existing vulnerabilities, as well as security measures in place, to monitor the server environment.

How containers work

Similar to virtualization, container technology also serves as a method for achieving resource isolation. The key distinction between virtualization and containerization is that containers do not require their own dedicated OS. Instead, one operating system hosts multiple individual containers, saving physical resources (e.g., RAM, storage, CPU), lowering licensing fees, and reducing other overhead costs.

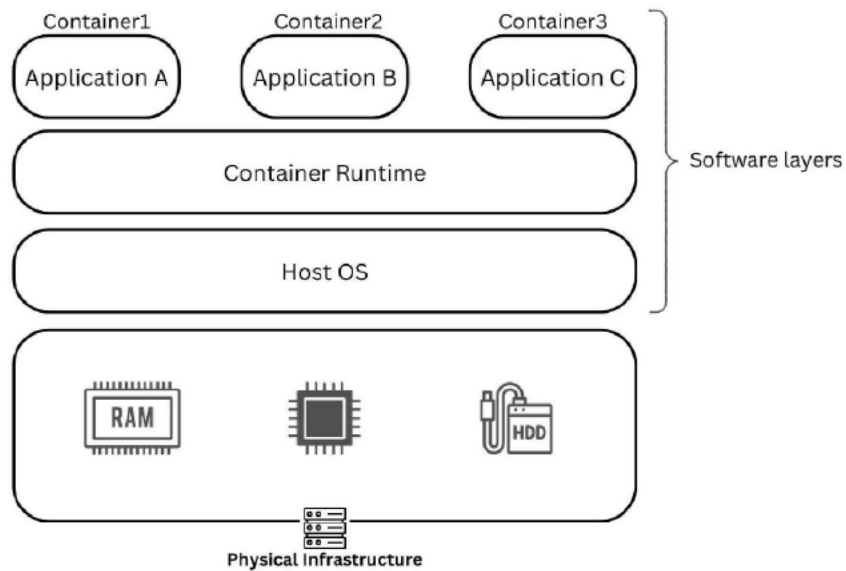


Figure 4: Containerization

Figure 4 shows how containerization works from a bird's-eye view. But to understand this better, we need to dig into the role of the container runtime engine, which runs on top of the host operating system. [Docker](#) is the de facto container runtime engine, but there are many others including [CRI-O](#) and [rkt](#).

Also, it's worth mentioning that all the container runtimes follow one governance structure known as the [Open Container Initiative \(OCI\)](#). OCI consists of three primary specifications:

- **image-spec:** Describes the physical structure of the [container images](#)
- **runtime-spec:** Describes how the container can run in a given [container runtime](#)
- **distribution-spec:** Describes how the API works

When combined, these offer a thorough framework for creating, distributing, and running containers; they also guarantee the portability and level of isolation in containerized apps.

WIZ ACADEMY

Container Security Best Practices

[Read more](#)



Key differences between containers and VMs

As we discussed, virtualization and containerization offer resource isolation, but with different architectures. The two approaches can coexist and outperform each other depending on the scenario, so it's important to know what each has to offer.

Isolation levels and security

Containers and VMs both provide isolation and hence security, but on different levels.

VMs

Even though VMs offer full isolation, they are still vulnerable to attack. Their advantage is that once compromised, VMs remain completely isolated from neighboring VMs. Still, in virtualization, the hypervisor is a more common target for most attack vectors because it gives the attacker full control of VMs managed by the hypervisor.

Containers

Containers offer more flexible isolation. For example, in Docker, there are different network configurations depending on the level of isolation required: bridge, host, and none network types.

- **Bridge network:** Docker's internal IP address management (IPAM) assigns a subnet, allowing for communication between all the containers assigned to that bridge network via their own IP addresses.
- **Host network:** All the containers share the network namespace of the host operating system; this is less secure because there is no network isolation.
- **None network:** Contrary to the host network, containers here will not attach to any networks, meaning they will not be able to communicate and provide full isolation.

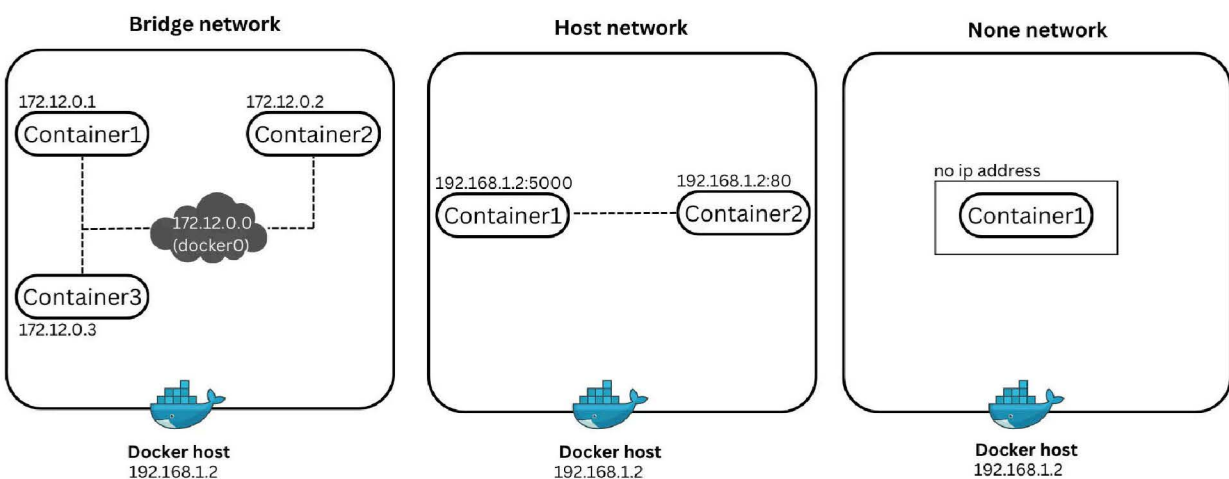


Figure 5: Configurable network isolation in Docker

With containers, the processes, file system, and resources (CPU, RAM, disk) also have flexible isolation, making containers more prone to vulnerabilities. With good security measures and scanning tools like Wiz and Kubescape, you can, however,

make the environment secure.

Performance overhead and efficiency

Containers are less resource-intensive compared to VMs because they share the physical machine hardware and OS kernel with other containers. This leads to better performance and also less startup time.

[A comprehensive study conducted by Martin Lindström at the Blekinge Institute of Technology](#) highlights that overall, container-based systems perform better than virtual machines, for instance, in terms of disk performance and memory usage. However, in a few tests (e.g., memory bandwidth benchmark test), VMs outperformed containers.

Scalability and resource utilization

Each VM requires resources equivalent to a full operating system instance, which means they come with significant resource overhead compared to containers.

Containers, on the other hand, require only application dependencies, making their resource utilization very efficient. They are also lightweight, which means that Docker containers can be started almost instantaneously, allowing for rapid scaling up or down as application demand changes.

Deployment, orchestration, and management

For containers, orchestration tools like Docker Swarm and Kubernetes automate container deployment and management; this includes scaling, service discovery, load balancing, state management, etc. This automated orchestration is essential for microservice applications with high complexity.

On the contrary, VMs follow more traditional configuration management and use less automation, making them mostly suited for running legacy apps.

Kubernetes Security Best Practices [Cheat Sheet]

This 6 page cheat sheet goes beyond the basics and covers security best practices for Kubernetes pods, components, and network security.

[Download Cheat Sheet](#)



Choosing between containers and VMs

Despite their popularity, containers will never replace VMs. Both are needed for different requirements. Make sure you know what factors to consider when choosing between the two.

Use cases for containers

- **Microservice applications:** These require ease of scaling and ease of communication between the services, which makes containers an ideal candidate.
- **DevOps and agile environments:** Containers support CI/CD and agile development practices by providing consistent environments, from development through production.
- **Highly scalable applications:** Containerization makes it comparatively easy to dynamically scale environments based on demand.

Use cases for VMs

- **Legacy applications:** These often require a specific OS or sometimes have complex dependencies that are difficult to containerize.
- **High-security environments:** VMs provide strong isolation compared to containers, which makes them ideal for applications with high-security requirements.
- **Stable workloads:** Applications with predictable, stable workloads that do not require frequent scaling can be efficiently deployed on VMs.

Hybrid approaches

There are instances where leveraging both technologies will provide a versatile, efficient, and secure environment for deploying applications. That's where tools like RancherVM, Red Hat OpenShift® Virtualization, and KubeVirt are especially helpful, as they manage virtual machines *and* containers.

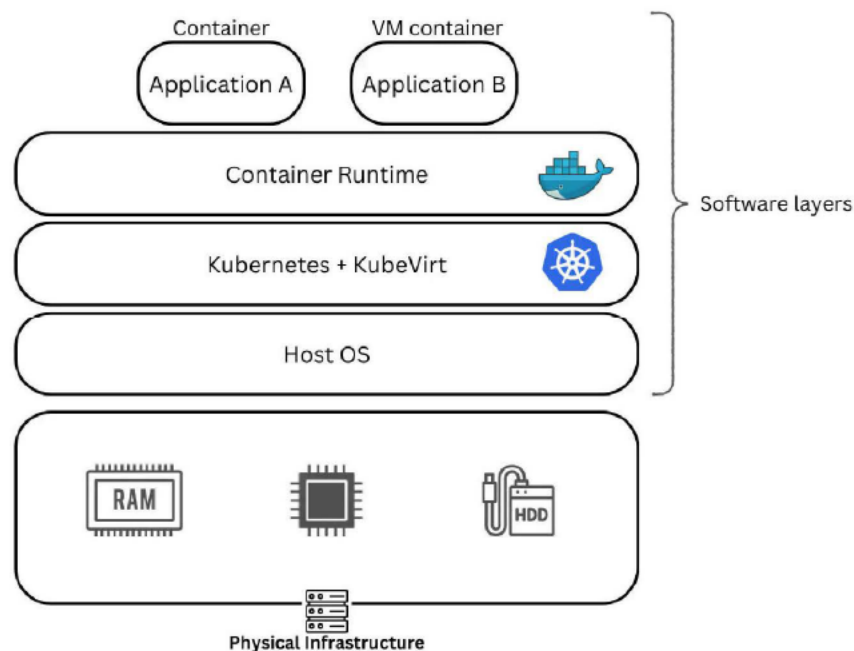


Figure 6: Architecture of KubeVirt

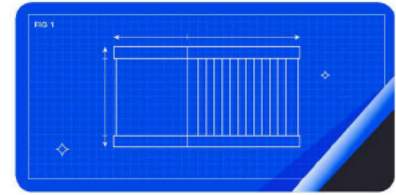
As seen above, the basic architecture of KubeVirt allows users to leverage Kubernetes' powerful orchestration capabilities for both containerized and virtualized workloads, making it easier to manage a hybrid environment.

With this integration, you can manage containers and VMs side by side using the same set of tools and APIs; this simplifies the deployment and management of mixed workloads in a cloud-native environment.

WIZ ACADEMY

Container Architecture: A Security Review

[Read more](#)



Conclusion

The debate between containers and VMs extends beyond a mere technical comparison to a strategic consideration of application deployment and management. The choice between the two technologies hinges on specific project requirements, security considerations, and scalability needs.

Whichever you go with, it's essential to maintain a strong security posture. Organizations must continuously evaluate their deployment strategies not only for performance and scalability but also for their ability to withstand emerging security threats.

This is where solutions like [Wiz](#) come into play, offering comprehensive security and compliance monitoring across both containers and VMs. Wiz's ability to provide deep insights into the security health of your deployments ensures that regardless if you use containers or VMs, your infrastructure will remain secure and compliant.

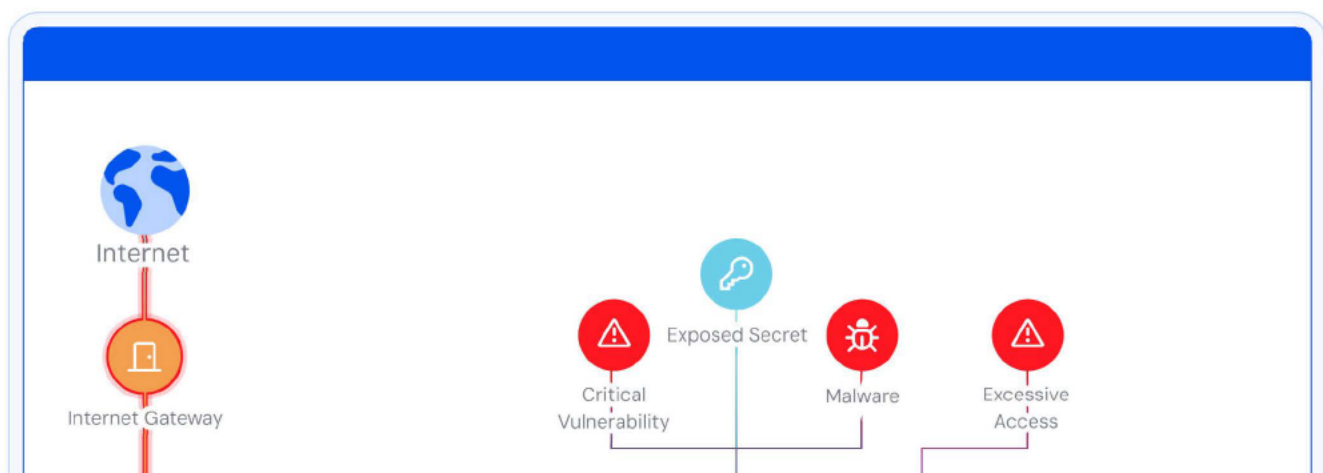
See for yourself how our industry-leading platform can secure your containers, VMs, and the rest of your cloud infrastructure.

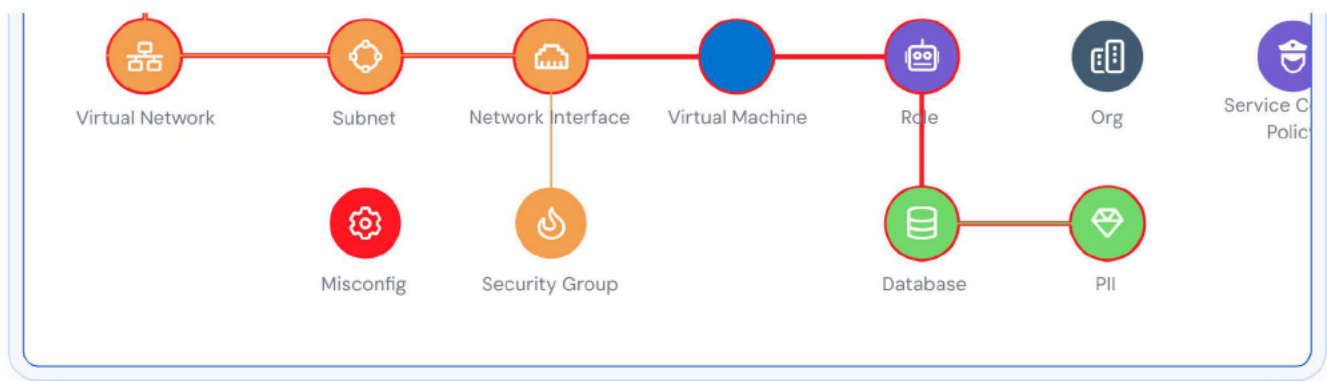
[Schedule a demo today.](#)

What running in your containers?

Learn why CISOs at the fastest growing companies choose Wiz to uncover blindspots in their containerized environments.

[Get a demo](#)





Explore more on this topic

Introduction to Containers and Containerization

- [Containerization 101: Transforming Application Development and Deployment >](#)
- [What is a Container Engine? >](#)
- [Containers vs. VMs: What's the difference? >](#)
- [Container Architecture: A Security Review >](#)
- [Linux containers: A security review >](#)
- [Container Runtimes Explained >](#)
- [Container Images Explained >](#)

PLATFORM

Wiz CNAPP

Wiz Code

Wiz Cloud

Wiz Defend

Integrations

Environments

Documentation

LEARN

Customer stories

Train Your Team For Cloud

Blog

CloudSec Academy

Resources Center

Cloud threat landscape

Cloud Security Assessment

COMPANY

About Wiz

Join the team

Newsroom

Events

Contact us

Trust Center

Our partners



https://arttrk.com/pixel/?ad_log=ua-00044119-
91738093010349&tag=impl&src=\${CREATIVE_ID}&msg=uid=\${CACHEBUSTER}%7Crev=1%7Cxndr_ctrk=\${CREATIVE_ID}!\${CREATIVE_CODE}!adxccl!%7C
6777-4afc-84b7-43d1d03528a8&idfa=\${DEVICE_APPLE_IDA}&device_id=\${DEVICE_AAID}