CLOUD SECURITY AND THE CONTAINERS APPROACH

By Igor Beliaiev

softserve

As elusive as the cloud can be for most people, the concept of the containers approach can be even more so. What is a container and where does it fit into cloud security?

“Containers” and “microservices” are an approach to developing applications for a cloud environment. The word “container” most likely brings to mind a shipping container, or box. As the word applies to the cloud, it’s not far off. A container is an easily moveable “box” of services that supports the applications that run within it, while providing a connection to the cloud host environment. Microservices are the applications that run inside the containers.

Why Containers?

The concept of containers is somewhat similar to the idea of a virtual machine itself. But containers are much more lightweight than virtual machines; more flexible, scalable, and easier to use. They can pack more applications into a given physical infrastructure than virtual machines, and without relying on a guest operating system or replicate. Container technologies can also be spun up instantly, and don’t virtualize entire servers, but rather the applications themselves.

The Benefits of Using Containers

For many years, software developers were building full programs that were installed on a computer and ran on a specific OS. But since all of the functions within the program weren't independent, a failure in any part of the program could crash the entire system. Booting up a new virtual machine with the service could take a few minutes, or even a few hours, for particularly large services.

Containers and microservices are an improvement on this old system, and provide a better way of building software for cloud deployment. Containers "spin up" in less than a second, so failure recovery is nearly instantaneous. Contrary to writing a giant, monolithic program of all user functions, developers are building microservices that each perform a very specific task. Making the functions fully independent from one another also allows the development team to use the best programming language for each function, rather than choosing one language for all functions.

Containers were conceived as the answer to a matrix of restriction and incompatibilities.

By providing a common interface to migrate applications between environments, you don't need to rebuild the application to ensure compatibility in the new environment. Various microservices come together to form a complete cloud-based service for the user.

Containers and Digital Transformation

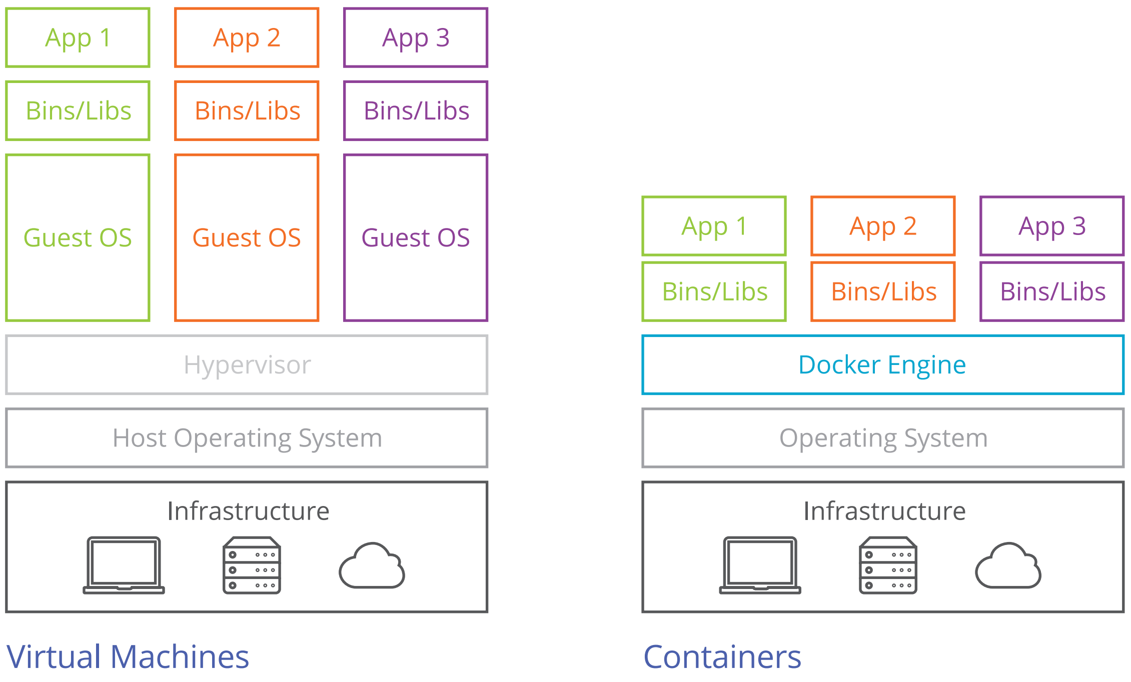
Containers and microservices help the most innovative software companies to increase the efficiency of their businesses, build more reliable software, improve a continuous delivery process, and transform their application development processes and IT infrastructure. As companies embrace new technologies like these, security must be more than an afterthought. In the era of digital transformation, the focus needs to be shifted from securing network perimeters to securing data spread across all systems, devices, and the cloud.

The world's largest technology companies, including Microsoft and Google, are now using cloud containers — and they're increasing their presence in production environments every day.

According to a **Red Hat survey**, 67 percent of organizations are planning to start using containers in production environments over the next two years. According to another survey by **Cluster HQ**, a container data management company, 73 percent of enterprises are currently using containers for development and testing.

Containers vs VMs

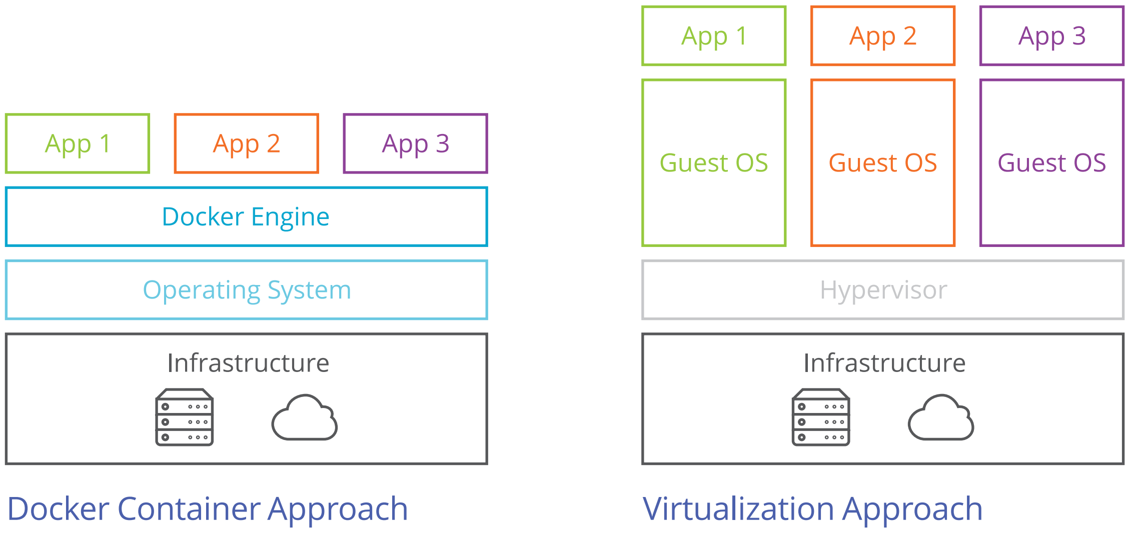
One of the most common misconceptions about containers is that they act as light virtual machines (VMs). Containers create an isolation boundary at the application level rather than at the server level. This isolation means that if anything goes wrong in that single container it only affects that individual container and not the whole VM or whole server. That leads many people to think they are perfectly isolated — but they're not. A malicious container can influence the execution of other containers through the common kernel, by exploiting a kernel vulnerability or leveraging the privileges of the compromised container.



Because containers roll an application together — with its dependencies and interfaces — into a single re-deployable unit, a container can be run on any host system with the appropriate kernel components while shielding the application from behavioral inconsistencies due to variances in software installed on the host. Multiple containers can be run on a single host OS without the use of a hypervisor, while still being isolated from neighboring containers. This layer of isolation introduces consistency, flexibility, and portability that enables rapid software deployment and testing.

Recognizable Container Software

One of the best examples of recognizable container software is Docker. At its core, Docker is an open-source project that uses several resource isolation features of the Linux kernel to sandbox an application, its dependencies, and interfaces inside of an atomic unit. "Your application is really more secure when it's running inside a Docker container," said Nathan McCauley, director of security at Docker, which currently dominates the container market.



It does not require any configuration off the bat — just one simple command to install it and you are in. But Docker is not a security tool meant to be used out of the box. Its default configuration provides a lot of settings that allow it to become much more secure and reliable. While it may not be for beginners, Docker is a great tool for developers.

Although containers aren't as isolated from one another as virtual machines, they are more secure than applications that run by themselves. And because containers are an easy way to package and distribute applications, many are doing just that.

But not all containers software available on the web can be trusted, and not all libraries and components included in those containers are secure, patched, and up-to-date.

Container Security

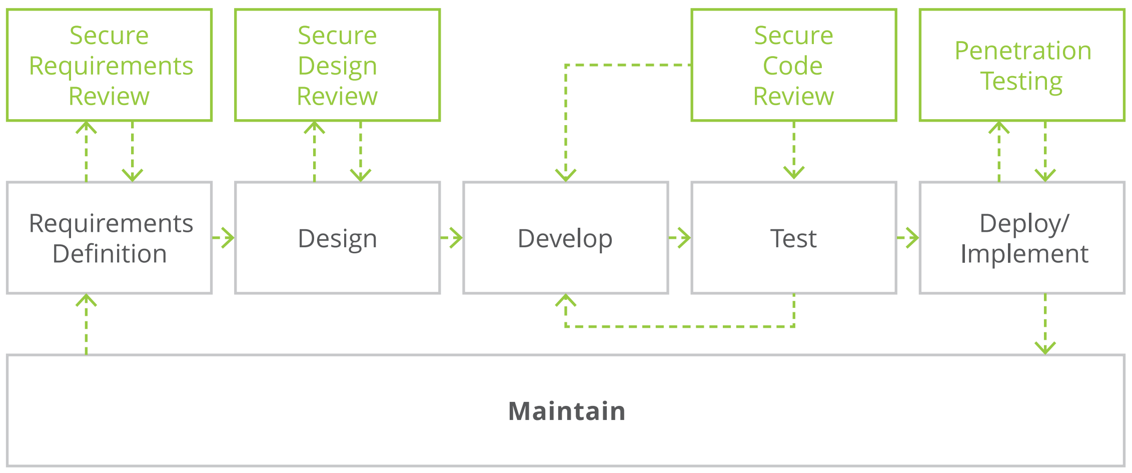
Since cloud containers became popular, one of the biggest concerns has become how to keep them secure. The best practices for securing container environments are not only about hardening containers or the servers they run on after security has been breached — they focus on securing the entire environment right off the bat. Security must be considered from the moment container images are pulled from a registry up until the containers are spun down from a runtime or production environment.

SecDevOps: Starting Security at the Beginning

The best way to make your cloud containers secure is to keep security in mind from step one.

At SoftServe, we use a specific security lifecycle:
Security + DevOps = SecDevOps. In other words, DevOps moves security closer to the beginning of the software lifecycle, making it an integral part of the secure development lifecycle (SDLC).

Security in the SDLC



More and more businesses realize that initiating security at QA time, or as a reaction to a security threat, is too late. Putting security earlier in the development lifecycle process causes a much higher rate of success and much higher throughput.

SecDevOps seeks to embed security inside the development process as deeply as DevOps has done with operations. Classic DevOps is designed to automate software development, accelerating the process to satisfy the needs of operations to acquire code that immediately works in production. SecDevOps, in contrast, automates the secure coding component of development to satisfy the needs of the security team, establishing and maintaining software that is immediately secure in production.

In most cases, the correct model for container security is a collaborative one with well-defined responsibilities that SecDevOps experts provide. Keep in mind, however: DevOps teams are not security experts and security teams are not usually as well-versed in DevOps. Security teams want to ensure that software is delivered without vulnerabilities, no matter how long it takes, and DevOps aims to deliver the best quality software as fast as possible. That being said, the future of this collaboration lies in the ability of security and development teams to communicate with empathy.

Putting security in the process from the beginning provides the opportunity to secure architecture, infrastructure, environment, and application from scratch, instead of looking at the entire application and trying to understand its potential risks. This approach allows for the discovery and correction of issues before going into production, and provides the ability to automate security processes into the development cycle that reduce human mistakes and improve risk management.

With large and complex applications, it’s necessary to ensure the security of the whole entity, rather than its bits and pieces. With the challenge of securing microservices, it’s necessary to protect a large number of services and their implementations one by one. They interact with each other through various APIs and data flows. More services means more elements exposed to cyber-attacks or leakage, so a larger span of security means better protection to cover all of them, rather than each at a time.

Best Practices for Secure Containers

Reducing the attack surface is a basic goal of security. Containerization has specific structural and operational elements that require scrutiny. Specifically, the underlying shared kernel architecture of containers requires special attention beyond securing the host; it requires maintaining standard configurations and container profiles. Here are some best practices for keeping your containers secure.

Keep It Simple

The first step is pretty straightforward: Keep it simple. Try to keep your container ecosystem as simple as possible. You should run processes in separate containers, even if these are services that depend on one another. You should also use the container-linking feature to connect two containers rather than combining them in the same one. You should also focus on keeping the footprint of containers small — don't load unnecessary packages or services that just waste resources — and make sure that your containers are designed to be easily replaceable.

Comprehensive Vulnerability Management

Going back to major areas for reviewing container security risks, we need to mention the importance of having a comprehensive vulnerability management program. Vulnerability management goes far beyond scanning images when they are first downloaded from a registry. Containers can easily pass through the development cycle with access controls or other policies that are too loose, resulting in corruption that causes the application to break down or a compromised runtime.

Proactive Checks

Implementing proactive checks throughout the lifecycle is another measure that ensures secure containers. Part of managing security through the container lifecycle helps to ensure the integrity of the container images in the registry and to enforce controls as they have been deployed into production. Image signing or fingerprinting could also be used, allowing you to verify the integrity of the containers. Ensure that only approved images are running in your environment.

Least Privileges in Runtime

Try to have the least privileges in runtime. This is a basic security best practice that applies equally in the world of containers. When a vulnerability is exploited, it generally provides the attacker with access and privileges that are equal to those of the application or process that has been compromised. Ensuring that containers operate with the least privileges and access required to get the job done reduces your exposure to risk.

Enforce Network Segmentation

On running containers, make sure to enforce network segmentation or micro-segmentation to segregate clusters or zones of containers by application or workload. In addition to being a highly effective best practice, network segmentation is a must-have for container-based applications that are subject to PCI DSS.

Monitor, Monitor, Monitor

And as we already know, monitoring is a key. Do an active monitor of container activity and user access as is it realized in any typical IT environment to quickly identify any suspicious or malicious activity. Log all administrative user access to containers for auditing.

Things to Watch Out For

Proactive security is preferable to reactive security in any situation, but sometimes it pays to keep an eye on particular problem areas.

Here are few major areas to consider when reviewing container security:

Kernel-Level Threats

In containers, the kernel is shared among all containers and the host, making the kernel more vulnerable. Kernel exploits range from privilege escalation, to arbitrary code execution, and even Denial of Service. Ensure the host operating system is hardened, up-to-date, and that it leverages kernel security patching features.

Bypassing Isolation

Attackers who gain access to one container can gain access to other containers or even to the host itself. If the hacker can get root privileges inside a containerized app, he can potentially gain root access to the host. This can impact potential privilege escalation attacks where the user gains higher privileges, such as those for the root user, and can be done through a bug in application code. This is the primary concern surrounding containers and but can be avoided by employing a comprehensive approach based on system hardening, strengthening default configurations, and patching.

Insecure Images

If container images aren't downloaded from trusted sources, there is a high risk of downloading malicious images into your environment. Those images could contain malware and backdoors, or even outdated software with known vulnerabilities. If an attacker can trick and convince you to run his image, both the host and your data are in danger. In an open-source environment, images created by any organization's developers are often updated and accessible. This can create an endless stream of uncontrolled code that may harbor vulnerabilities or unexpected behaviors. To prevent this, always download images from trusted sources and ensure that images do not contain any software with known vulnerabilities.

Compromising Secrets

When a container accesses any service, it will likely require a secret, like an API key or username and password. An attacker who can get access to this secret will also have access to the service. This problem becomes more acute in a microservice architecture in which containers are constantly stopping and starting. The best practices to keep your secrets safe include: encrypting secrets while in transit; encrypting secrets at rest; preventing secrets from unintentionally leaking when consumed by the final application; and strictly adhere to the principle of least-privilege, where an application only has access to the secrets that it needs — no more, no less.

Denial-of-Service (DoS) Attacks

A container could behave in a way that effectively creates a DoS attack on other containers. For example, opening sockets repeatedly will quickly bring the entire host machine to a crawl and eventually cause it to freeze up. DoS attacks may include scenarios where one container seizes control of all available system resources to stop other containers from operating properly. If a single container can monopolize access to certain resources – including memory and more secret resources such as user IDs – it can starve out other containers on the host, resulting in a denial of service, whereby legitimate users are unable to access part or all of the system. To avoid this issue in most cases, try limiting the amount of resources, CPU, and memory allocated to the container. The implementation of monitoring tools can also help to track and deal with DoS attacks.

Cross-Container Threats

Cross-container threats usually arise due to weak network defaults such as a bridge configuration. An application on one container may be able to compromise another container on the same host or on the same local network (for example gaining unauthorized access to the database container on the same host). To prevent cross-container threats, use proper networking management with custom configuration and user-defined networks.

Default Configurations

Systems that use default configurations have a high potential for information security breaches, because cyber-criminals tend to look for common standard configuration. Make sure to customize your containers, not only to better suit your own needs but also to decrease the likelihood of compromised security. Do the additional work to harden your container-based environment and make it more secure and robust.

Conclusion

Containerized applications can be inherently more secure than their predecessors with the right attention to security. Microservices provide a high level of risk mitigation due to the fact that they are container-based, feature the ability to easily rollback if needed, and provide greater system resilience.

Organizations can improve their security with the use of containers, without adding incremental overhead to their application infrastructure. Containers provide isolation for applications from their host and from each other. They minimize the use of resources from the underlying infrastructure, and reduce the surface area of the host itself when configured properly. While applications secured in containers are already more secure, there are endless ways to make your containers even safer.

Additional Resources

Docker Secure Deployment Guidelines
AWS: **Docker on AWS**
Docker Docs: **Docker Security**
Redhat: **Container Security Guide**

ABOUT US

SoftServe is a global digital authority and consulting company, operating at the cutting edge of technology. We reveal, transform, accelerate, and optimise the way large enterprises and software companies do business. With expertise across healthcare, retail, media, financial services, software, and more, we implement end-to-end solutions to deliver the innovation, quality, and speed that our clients' users expect.

SoftServe delivers open innovation – from generating compelling new ideas, to developing and implementing transformational products and services. Our work and client experience is built on a foundation of empathetic, human-focused experience design that ensures continuity from concept to release.

Ultimately, we empower businesses to re-identify their differentiation, accelerate market position, and vigorously compete in today's digital, global economy.

Visit our **website**, **blog**, **Facebook**, **Twitter**, and **LinkedIn** pages.

USA HQ

201 W 5TH STREET, SUITE 1550
AUSTIN, TX 75703
+1 866 687 3588

EUROPEAN HQ

One Canada Square
Canary Wharf
London E14 5AB
+44 (0)800 302 9436

info@softserveinc.com
www.softserveinc.com

softserve