Twistlock
Enterprise Security. DevOps Agility.

STRATEGIES FOR SECURING HIGHLY SCALABLE CONTAINERIZED ENVIRONMENTS

OVERVIEW

If you ask a Docker expert to identify the main benefits of containers, scalability -- that is, the ability to increase or decrease the size of a software environment -- will undoubtedly be on the list. In contrast to virtual machines and bare-metal servers, containers enable organizations to build tremendously scalable environments. This is because containers make it easy to copy instances of an application quickly, and to move instances between different environments easily.

To create a truly scalable containerized environment for production, however, it’s necessary to think beyond the scalability of containers themselves. Containers are the building-blocks that enable a scalable environment, but they are not the only part of the software stack that needs to scale. The tools that your organization uses to manage, monitor and secure containers must also have the ability to scale as seamlessly as the environment.

The latter category of tools -- security -- is in many respects the most difficult to scale. This is because the security tools that many organizations still use were not designed with containers in mind, and are ill-prepared to support the massive scalability requirements of a modern computing environment.

This whitepaper discusses this challenge. It explains why enabling scalability for security tools is an essential component of building an effective containerized environment -- and why the security layers of a containerized software stack are in many respects the most difficult parts to scale. It then explains how organizations can meet the challenge by adopting security tools and practices that are truly scalable and free them from the constraints that can undercut the benefits of Docker adoption.

SCALABILITY BASICS

Before delving into the details of the relationship between security and scalability in containerized environments, it is important to define what scalability means and why it is important -- especially in a containerized environment.

As noted above, scalability means the ability to increase -- or, just as important, decrease -- the size of a software environment in response to fluctuations in demand.

Achieving scalability in production environments is important for two main reasons. The first is ensuring high availability. In order to make certain that applications and services remain available and stable as the number of users accessing them increases, you must be able to scale your environment up instantly whenever demand increases. To do this, you must add more objects to your environment by creating additional containers or add instances of your application by implementing additional deployments.

The second benefit of scalability centers on the cost savings and efficiency that an organization derives from being able to decrease resource consumption quickly whenever demand decreases. Running more instances of an application or service than are required to meet demand at any given moment is wasteful. Environments that are larger than they need to be not only create unnecessary costs, but also burden your team with more infrastructure to manage and expose a larger-than-necessary surface to potential attack.

Scalability and containers

Containers' ability to enable scalability has been one of the chief selling-points of container platforms like Docker since its emergence in 2013. Containers can start in seconds, whereas virtual machines and bare-metal servers generally take minutes to boot. This means that a containerized application or service can not only be launched much more quickly than one hosted on a virtual or physical server, but also that additional instances of that application or service can be created almost instantaneously.

The easy portability of containers also helps to make containers highly scalable. With a scant few exceptions, a Dockerized application or service can run inside any type of host environment that supports Docker.

This portability makes it easy for a team to add or remove physical host infrastructure from a containerized environment quickly, without having to spend time configuring the host environment for the application. It also enables organizations to create additional deployments of an application or service by copying a containerized environment and starting it on other infrastructure.

THE SECURITY CAVEAT:
HOW SECURITY CREATES SCALABILITY CHALLENGES

While it is easy to scale containerized applications and services up or down, achieving true scalability in a production environment requires scaling more than just containers and the code running inside of them. It's also essential to be able to scale the tools that your organization uses to help manage and secure the containers.

After all, a Dockerized environment in which the number of containers can increase or decrease easily, while the scale of the tools used to keep the environment stable and safe cannot be changed quickly, is not actually scalable. It's like a submarine moored in a shallow pond, or a cheetah confined to a cage: Its agility and versatility are undercut by the limitations of the context in which it exists.

In many instances, security tools are particularly constraining on the scalability of a Dockerized environment. While the management and orchestration solutions that organizations typically deploy alongside Docker have been designed with containers in mind, there is a good chance that the security tools they use were not.

That is a problem because security tools and considerations present a number of special challenges when building a scalable containerized environment. Those challenges include:

- **Absence of baselines.** In an environment that constantly fluctuates in scale -- either as the number of objects within the environment increases or as the number of deployments grows -- it is difficult or impossible to predict what the environment should look like at any given moment. This makes it hard to determine normal operating conditions and to establish baselines for security purposes.
- **Lack of comprehensive security monitoring.** On an individual basis, the various tech nologies used to construct a typical container environment -- a host operating system, the Docker daemon, an orchestrator such as Kubernetes or Swarm, a container registry and so on -- are designed to be secure. However, the security needs and functionality of each individual component tend not to be well integrated. By default, container stacks lack security tools that monitor every layer. While it is possible in small environments to perform this type of integrated, comprehensive security monitoring manually, doing so becomes impossible when the environment grows large in scale.
- **Lack of active defenses on host systems.** The security strategy inside Docker host environments -- that is, the specialized operating systems created for hosting a Docker environment -- generally focus on hardening the host environment against attack, rather than providing active security. Tools such as AppArmor and SELinux help to make attacks more difficult, but they are not designed for detecting or mitigating attacks once they are in progress. Here, too, organizations face a scalability challenge because active security is possible to implement manually in small-scale environments, but it cannot be achieved when environments scale up.

- **Different security strategies.** The security strategies and policies that work in a small environment may not work in a large one because the nature of the environments is different. For example, in a small environment, an organization may not need to rely on analytics or machine learning to detect threats because those threats can instead be detected manually. In a large environment, however, machine learning and analytics are essential. They are the only way to build an automated, active security strategy that works at any scale.
- **Manual configuration fails at scale.** In small environments, security policies and configurations can be set manually. This approach is unfeasible in environments that are comprised of thousands of containers, or in which applications are constantly updated. For this reason, too, achieving scalability requires adopting a security strategy that includes the automation necessary to work in large-scale environments.
- **Faster updates.** The larger an environment, the harder it is to make sure that every update that is pushed out will not introduce vulnerabilities. In small-scale environments where updates are implemented on an infrequent basis, container images can be checked manually and sequentially. In a continuously delivered large-scale application deployment, however, this approach fails. In the latter context, security must be bricked into all stages of the delivery chain -- from development and pre-production testing to runtime -- in order to maximize the organization’s ability to find and respond to threats within a highly dynamic environment. In other words, security must be shifted to the “left” of the delivery chain.
- **Efficiency.** As noted above, scalability means the ability to decrease the size of an environment as well as increase it. When an environment shrinks, it is important to ensure that security processes, and the amount of resources they consume, shrink with it. Otherwise, an organization risks wasting resources on security overhead that is not necessary at times when an environment is smaller.

For all of these reasons, traditional security tools are poorly positioned to meet the scalability requirements of fast-changing containerized environments.

HOW TO SCALE SECURELY

Yet by taking a more modern and innovative approach to security, it is possible to implement a highly scalable containerized environment that is also secure.

The following strategies and practices will allow your organization to meet the challenges described above and achieve the full scalability that containers are capable of enabling:
in his tracks, rather than waiting until the damage is done to shut him out.

Last but not least, by leveraging static analysis and machine learning to automate security policy creation, containers obviate the need for developers and security analysts to collaborate manually on security policy. For this reason, containers enable a much greater degree of scalability by removing dependencies on human actors. They also help to prevent configuration rot by ensuring that security policies are updated automatically, in real time.

All of the above means that when you migrate to microservices and containers, your security strategy no longer boils down to securing your software environment's perimeter and hoping for the best. You can instead adopt a proactive, continuous approach to security that takes advantage of special characteristics that are available only inside a containerized environment.

- **Embrace security add-ons.** The need for comprehensive, across-the-platform security, which the security features that are built into container tools cannot deliver, means that adding extra security tools to your environment is not just an option. It's the only way to make sure that your environment is truly secure at all layers.
- **Leverage automation to maximize time-to-value.** Security in a fast-changing container environment requires automation. For this reason, organizations should deploy security tools that can automatically create and adjust security policies. This removes manual tasks from your operations in order to maximize time-to-value and ensure that your team can focus its energies doing what matters most -- putting their expertise to work -- rather than configuring policies that could be handled automatically.
- **Automate policy configuration.** Your security policies should be configured and enforced automatically. Automation prevents policies from becoming outdated or inconsistent with the needs of your environment as it scales. This prevents configuration rot that could lead to vulnerabilities.
- **Leverage machine learning.** In a large-scale environment, humans alone cannot identify and interpret security threats quickly enough to prevent breaches. This is why machine learning is essential for mitigating vulnerabilities and threats in a scalable containerized environment.

- **Maintain strict access control.** As the number of systems and deployments in your environment increases, protecting access becomes more complicated. This is all the more true when scaling up means increasing the number of people who help to manage an environment (and scaling down requires removing the access of those who no longer need it). Strict access control needs to be a core part of your security strategy in a containerized environment.
- **Secure all layers.** Keeping environments secure at scale means securing all layers of the environment. In a containerized environment, this requires implementing hardening and active security features from the host operating system and registry to the container engine and application services.
- **Adopt consensus-based best practices.** Consensus-based best practices mean those that have been recognized in the industry as the most effective for securing containerized environments. They are defined in publications such as NIST SP 800-190, which describes best practices for application container security. These standards are essential for scalability because, in large and rapidly changing environments, it is not realistic to identify healthy baselines and best practices yourself.
- **Integrate security across the SDLC.** In a highly scalable environment, addressing security problems in production is not always possible. In order to prevent security challenges from interfering with scalability, you should integrate security operations across the Software Development Life Cycle (SDLC). By shifting security “left” -- that is, performing security tests and hardening earlier in the software delivery chain -- you can find and address security issues before they reach production, freeing your production apps and services to scale seamlessly.
- **Implement tools that can scale down.** In order to avoid paying for more than you need, you must ensure that your security tools can scale down as well as up. This ability keeps your containerized environment lean and efficient.

CONCLUSION

Security may not be the first consideration that comes to mind when you think about scalability. Traditionally, security has not been a major part of the conversation about scale.

In a containerized environment, however, ensuring that security tools and operations can scale as quickly as containerized applications and services is the only way to realize the full benefits of containers. This is difficult because traditional security tools were not designed with the challenges of scalability in mind.

Yet by taking a new approach to security -- by adopting tools that were designed from the ground up to handle the scalability demands and other challenges of containerized environments -- you can scale rapidly and stay secure at the same time.

ABOUT TWISTLOCK

ENTERPRISE SECURITY. DEVOPS AGILITY.

Twistlock protects today's applications from tomorrow's threats with advanced intelligence and machine learning capabilities. Automated policy creation and enforcement along with native integration to leading CI/CD tools provide security that enables innovation by not slowing development. Robust compliance checks and extensibility allow full control over your environment from developer workstations through to production. As the first end-to-end container security solution, Twistlock is purpose-built to deliver modern security.

LEARN MORE AT

Twistlock.com