SPONSORED BY

PRISMA™
BY PALO ALTO NETWORKS

GEEK GUIDE

Calculating the ROI of DevSecOps

LINUX™ JOURNAL

Table of Contents

GEEK GUIDES:
Mission-critical information for the most technical people on the planet.

Copyright Statement

About the Sponsor

Palo Alto Networks' mission is to protect our way of life in the digital age by preventing cyberattacks with our pioneering Security Operating Platform, providing highly effective cybersecurity in the cloud, across networks, and for mobile devices.

Calculating the ROI of DevSecOps

PETROS KOUTOUPIS

Introduction

In the beginning came DevOps. By streamlining both Software Development and IT Operations, it merged two extremely important roles to deliver software effectively and efficiently. The DevOps role shortened the development lifecycle to deliver vital bug fixes, software updates and much needed features quickly. DevOps reduced the bottleneck of the entire process with the exception of one key component: *security*.

DevSecOps

DevSecOps expands beyond the practice of DevOps by introducing the practice and mindset of security into the process. Its primary goal is to distribute security decisions safely at the necessary speed and scale while not sacrificing the required security. Remember, DevOps is centered around development and operations. If you need to take advantage of both the agility and responsiveness that DevOps offers, security needs to play a role in the software lifecycle.

Why DevSecOps? Typically, security becomes an afterthought in the software development/delivery lifecycle, and it's often pushed off to the final stages of the process. Before the DevOps concept emerged, when the entire process consumed many months to even years, this was not considered problematic. Now that more companies have adopted Continuous Delivery/Continuous Integration (CD/CI) models, releases tend to occur a lot more frequently. I'm talking about weeks, if not days, before a new revision of an application drops into the public domain. Waiting until the very last minute to ensure that the application is safe and secure to deploy destroys the entire process and potentially could derail the delivery of the application. What could have been a few weeks, might end up being a few months of development, testing and integration.

What does DevSecOps look like? Basically with

DevSecOps, security is designed into the application or feature at the onset of the process. A good strategy is to determine risk tolerance and conduct a risk analysis of a given feature. How much security are you willing to provide the feature? And how consistent are you going to be with that requirement throughout the lifecycle of that same feature? Now, what happens when you scale that model across multiple features, sometimes being worked on simultaneously? Automation certainly will help out a lot here. Ideally, this automation would maintain short and frequent development models while also integrating your security measures with minimal to no disruptions to your operations.

DevSecOps introduces many other advantages, including but not limited to the following:

- Increased speed and agility for security teams.
- Decreased response time to address change and needs.
- Increased or better collaboration and communication across teams.
- Increased opportunities for automated builds and quality assurance testing.
- Early identification of vulnerabilities in application code.

- Resources and talent are freed to work on high-value work.

DevSecOps is a critical component in markets where software updates already are performed multiple times a day. Older security models just cannot keep up.

The six most important components that make up the DevSecOps approach are:

1. The ability to deliver code in small chunks so vulnerabilities are identified quickly.

2. Increased speed and efficiency to source code management, determining whether a recently submitted change is good or bad.

3. Being in a constant state of compliance (that is, audit-ready).

4. The ability to identify potential emerging threats with every code update and then being able to respond quickly.

5. The ability to identify new vulnerabilities with code analysis and then being able to understand how to respond and patch the affected code.

6. Always being up to date with training engineers on

security guidelines for set routines.

Some may argue that the "security" piece is nothing more than a mindset or philosophy. Even if that were the case, a large part of the challenge is identifying risks early on and using the right tools to guide you through the entire process—from the very beginning to the very end.

Containers

Containers and container technologies have redefined the way many organizations conduct business. The technology brings unprecedented agility and scalability. It should come as no surprise that container technologies are widely adopted and continue to thrive in the wild. They even form the foundations to many of the cloud native, mobile and cross-platform applications that we take for granted today. Knowing this, it does raise the question, how can you be sure that each deployment is safe and secured?

The Benefits of Containers To recap, containers decouple software applications or services (often referred to as microservices) from the operating system, which gives users a clean and minimal Linux environment while running the desired application(s) in one or more isolated "containers". Containers were and still are an ideal technology for the ability to isolate processes within a respective container. This process isolation prevents a misbehaving application in one container from affecting processes running in another

container. Also, containerized services are designed not to influence or disturb the host machine.

Another key feature of containers is portability. This is typically accomplished by abstracting away the networking, storage and operating system details from the application, resulting in a truly configuration-independent application, guaranteeing that the application's environment always will remain the same, regardless of the machine on which it is enabled. With an orchestration framework behind it, one or more container images can be deployed simultaneously and at scale.

Containers are designed to benefit both developers and system administrators. The technology has made itself an integral part of many DevOps toolchains. Developers can focus on writing code without having to worry about the system ultimately hosting it. There is no need to install and configure complex databases or worry about switching between incompatible language toolchain versions. Containers streamline software delivery and give the operations staff flexibility, often reducing the number of physical systems needed to host some of the smaller and more basic applications.

The beauty of containers is that they are completely platform-agnostic. As a result of their portability, they can be deployed on-premises in local data centers or out in the

cloud. Under the same management framework, they can be managed and monitored seamlessly across both hybrid and multi-cloud environments. You even can run containers within virtual machines or serverless in cloud native applications. The possibilities are endless.

Container Adoption According to a 2018 survey conducted by the Gartner research firm, by the year 2020, more than 50% of the IT organizations that were surveyed will be using container technologies. This is up from less than 20% in the 2017 survey. Without a doubt these and many other organizations are seeing the value in using containers.

In addition, a 2017 Forrester report, "Containers: Real Adoption and Use Cases in 2017", commissioned by Dell EMC, Intel and Red Hat, revealed that of the 195 US/European managers or IT decision-makers responsible for public/private cloud decisions surveyed, at least 63% used containers with more than 100 instances deployed. That number was projected to grow in the coming years. The very same survey listed "security" as the *number one* roadblock to container technology deployment (37%).

Think about it. A "build once, run everywhere" application can be affected (alongside the many other container applications) by an infected or vulnerable kernel hosting it. It also can be affected by the applications and libraries it's packaged with. This would not be the case in a virtual

machine, as the application would be fully isolated from the other(s). And now that more workloads have moved to the cloud, where organizations have less control over the system(s) hosting their containers and cloud native applications, this becomes more of a risk.

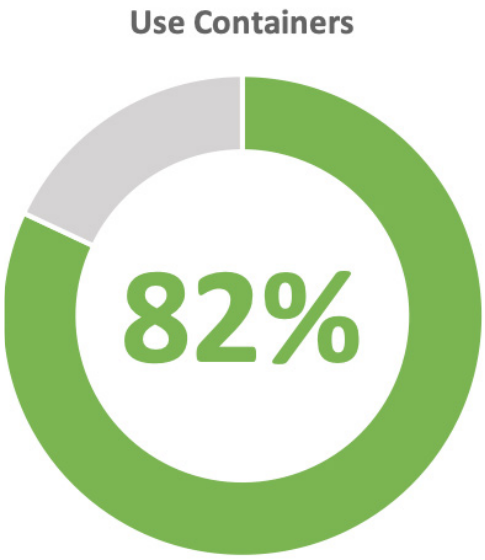

FIGURE 1. IT Professionals Already Using Container Technologies

What's driving this adoption? A Portworx "2018 Container Adoption Survey" may provide the answer. Out of the 519 IT professionals that were surveyed, nearly 82% were already running container technologies, and 84% of those who were running them were running them in production. And of those, 30.2% claimed it was to enable their applications to run on multiple cloud platforms and to avoid vendor lock-in. The rest stated that it was to increase developer efficiency (32.3%), save on their infrastructure costs (25.9%) and support microservices architectures (11.6%).

For those hosting their containers in the cloud, 12.8% were running them in three separate clouds (Google + Azure + AWS), while 22.5% were running them in two clouds (AWS + Google, Google + Azure or Azure + AWS).

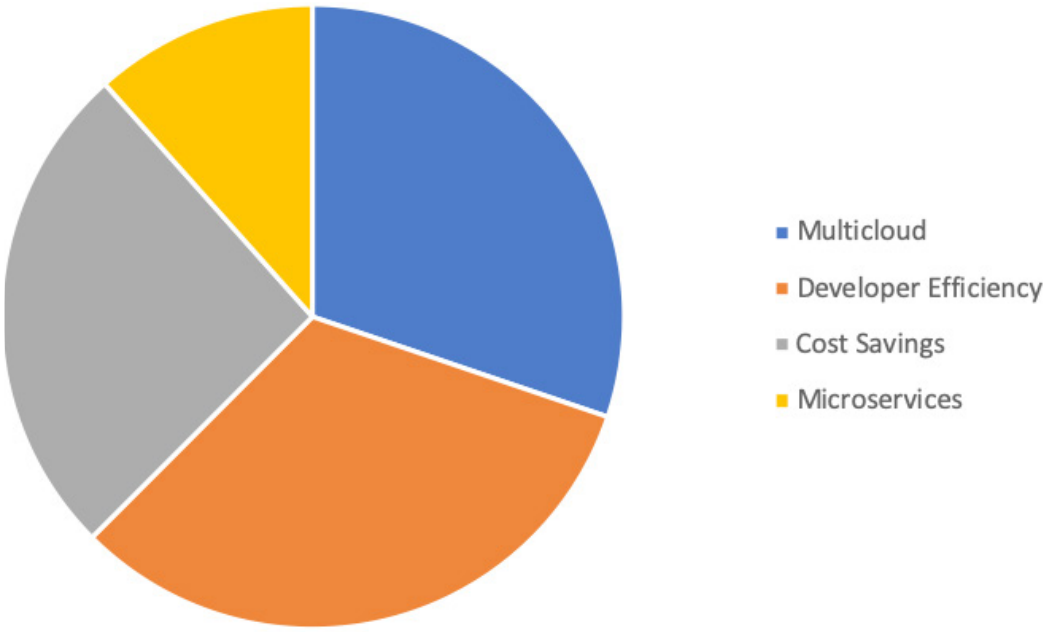

FIGURE 2. Reasons for Using Containers

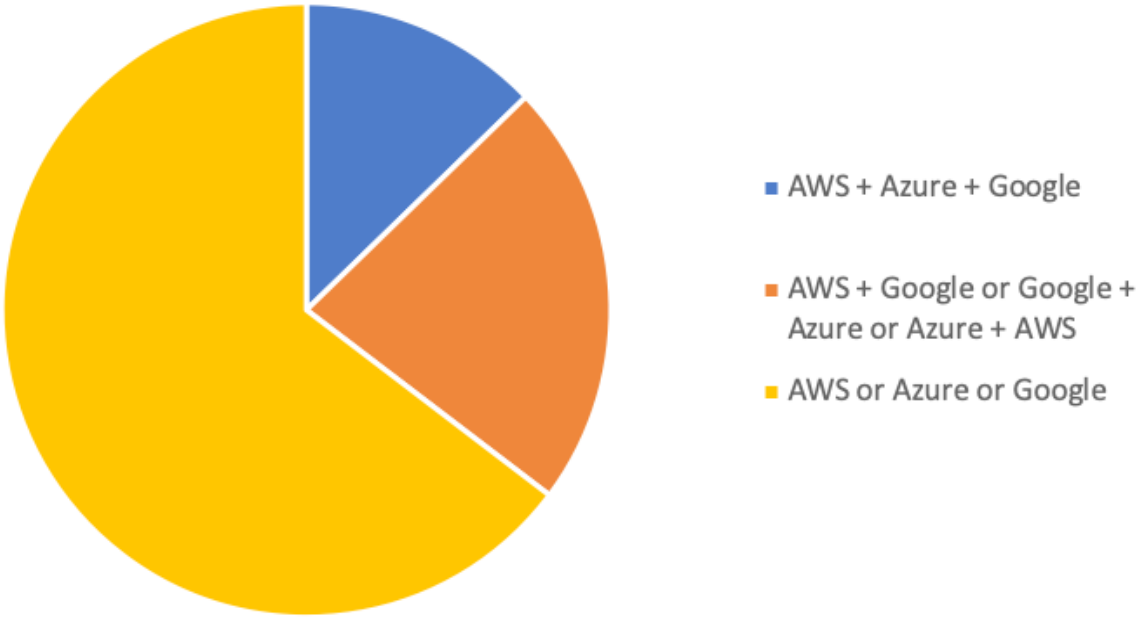

FIGURE 3. Single or Multicloud Deployments of Containers

Container Security Considerations Although container technologies bring an added layer of security for running applications in an isolated environment, containers alone are not an alternative to taking proper security measures. Unlike traditional hypervisors, a container can have a more direct path to the host operating system's kernel, which is why it is standard procedure to drop privileges as quickly as possible and run all the services as non-root wherever possible. Also, note that whenever a containerized process requires access to the underlying filesystem, you should make it a good habit of mounting that filesystem as read-only.

The state of a container image also raises concern, which is why it's improper to run containers (be it Docker or anything else) from an untrusted source. When deploying an unknown or unofficial image, you increase the risk of running vulnerable or buggy code in your environment. And if that container is configured to host a privileged process, any attack exposing a potential vulnerability eventually could cost the data center an entire host system (and maybe more).

There also exists the potential for a container to run system binaries that it probably shouldn't be touching in the first place, at least without your knowledge. Another similar scenario is when a rogue application or attacker gains container access through an application vulnerability and replaces some of the underlying system binaries with one

that does not belong or was not intended to run in that container image—all of which will continue to run during the life of that container. This can result in additional system and network compromises or worse.

Having the right tools to enable the Sec in DevSecOps will go a long way and can potentially save your firm or your customer tons of hours of headache (and downtime), and in turn, lots of dollars to repair the damage done. Damage is not confined only to software or data. It can also destroy reputation.

A Return on Investment?

Costs are one of the key factors to container adoption. At least, that's what 37% of respondents stated according to the Survata "Container Adoption and Drivers" survey conducted in 2016. Another 21% cited the increase in frequency of software releases. Regardless of how you look at it, the main takeaway is a decrease in spending and increase in profit.

Clearly, if it were not for the many benefits, industries would not be deploying container technologies. Such benefits cited in the same survey include improved flexibility for IT infrastructure (63%), overall IT cost savings (53%), increased speed/productivity for developers deploying code (52%), greater responsiveness to business needs (40%) and more ROI from the cloud (38%).

At the time, two-thirds of IT professionals expected their company to save at least 16% on IT costs by using containers, while one-fifth indicated that their savings would exceed 30%.

Where Does the Money Go? At the end of the day, the initial investment into building a container-friendly infrastructure can be quite expensive. Costs include the following:

- Commercially supported and managed container products.
- The hardware servers, storage, network switches and so forth.
- Container orchestration/management tools (to enable multi-cloud or hybrid clusters).
- The hardware and software to support and manage the image registry.
- Experienced personnel to manage/maintain and even consult or design the services around containers.

I'm talking about an up-front Capex with an ongoing Opex here, much like any other technology deployment inside the data center. Either way, it's important to assess the ROI for these Capex and Opex charges.

Digging deeper into the key aspects of a container-friendly infrastructure, you need to consider the following aspects of containers:

- **Runtime engine:** the runtime engine operates and manages (for example, clone, suspend and snapshot) the deployed container. Often, you will find container runtime engines included in modern operating system distributions and virtualization platforms.

- **Image repository:** an image repository will provide a single location for container image distribution. It also will provide long-term storage and version control for those same container images.

- **Orchestration framework or workload manager:** a container management system (such as Kubernetes, OpenShift, Rancher and so on) will automate the deployment of container images across multiple hosts, balance workloads across those systems, restart containers on crashes and provision additional copies of a container to handles increased application usage.

- **Virtual network overlay:** to enable inter-container communication, you must enable a virtual network overlay over shared physical network interfaces.

- **Hardware infrastructure:** one of the most important

pieces to building a container-friendly infrastructure is provisioning and configuring the right amount of hardware with the right amount of horsepower and enough room for expansion or growth. At the end of the day, no matter how abstracted a container is from the underlying hardware, the application itself must eventually be deployed on physical (or virtual) machines—that is, servers, switches and storage systems to hold the persistent application data. These workloads can live on-premises, in one or more public clouds or both, which leads to a significant investment in meta-management tools to manage those same workloads across multiple disparate platforms.

- **Support and expertise:** various elements are required to run production-scale container deployments. And although the previous sections of this ebook cover a large portion of the upfront costs involved in deploying container technologies either locally, in the cloud or both, there will be a time when you need to make an investment on the operations piece to support the infrastructure, and finding the talent to maintain or debug said infrastructure can be a challenge all on its own. Most organizations tend to seek consultants or vendor professional services to assist with container strategies, architectural design, implementation and support.

Once you make all the investments to define, implement,

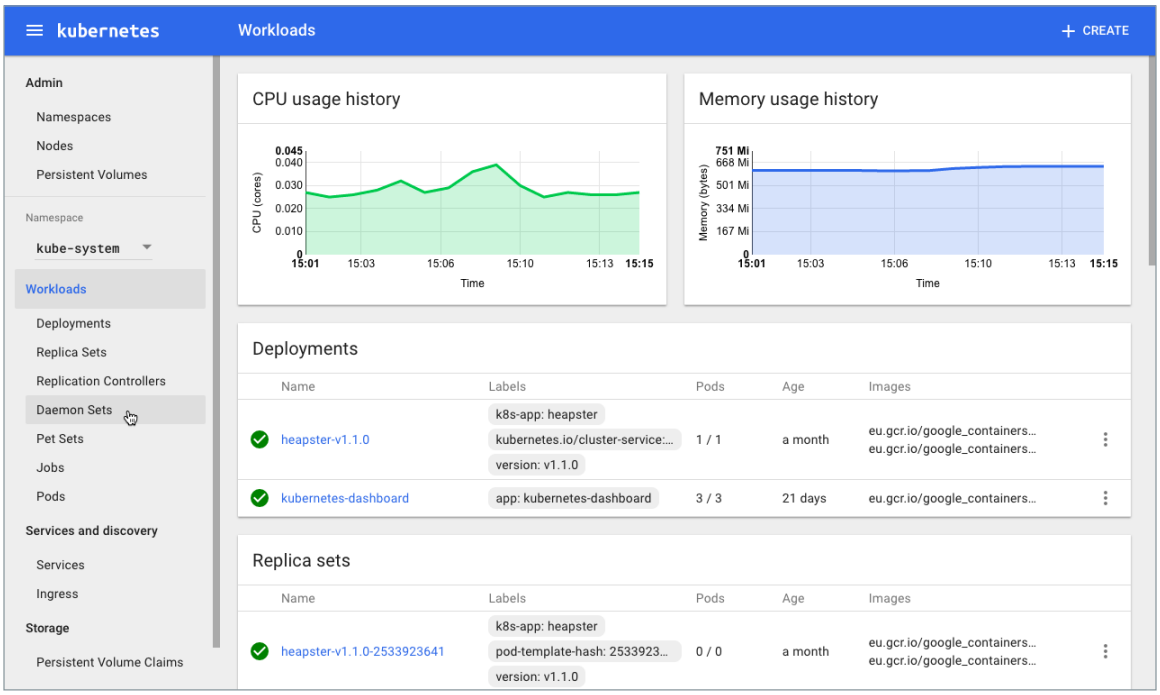

FIGURE 4. The Kubernetes Web UI Dashboard Source: kubernetes.io

secure and support a container-ready infrastructure properly, ROI will immediately follow. The amount of time it takes for a return on that investment is reduced further once you enable the security piece of DevSecOps. It's simple. The less resources spent on investigating, debugging and addressing production code, the less money spent in general.

Bringing the Sec to DevSecOps

Now, what do you look for to add security to your DevOps ecosystem? You'll need a product that focuses on container security across an application's lifecycle—

one that's fully committed to providing enterprise security with DevSecOps agility and able to integrate with any modern CI tool or registry. It also should be designed to be deployed alongside your virtual machines, containers and cloud native applications.

I'm talking about an end-to-end security solution built for containerized environments that protects against software exploits, malware and active threats through its advanced intelligence and machine-learning capabilities. One that will profile expected container behavior automatically, and create and enforce security models at runtime. The goal would be to build security models of expected behavior and enforce them automatically via whitelisting. Ideally, security can be introduced much earlier in the development lifecycle to identify and block threats from developer workstations through to production.

The following are some key features to look for:

- **Runtime protection:** defends your containers against detected exploits, compromises, application flaws and configuration errors, and actively monitors container activities and detects policy violations. With reporting of all anomalous behaviors while also taking the appropriate actions to disconnect or isolate them, runtime protection prevents disruption to any and all other containers across the Kubernetes cluster (or other workload

manager). The solution should identify when a container does something it shouldn't be doing. For instance, if a container running nginx suddenly invokes netstat, and netstat isn't a whitelisted process for that image, the security platform should detect it.

- **Vulnerability management:** constant scanning of container images in registries, workstations and servers for known vulnerabilities and misconfigurations is a must with detected vulnerabilities being reported. How nice would it be to break the Docker image apart and parse each individual layer, specifically searching for these threats? The platform would take remediation actions based on the severity of the vulnerability during runtime and provide users with granular control when managing the types of vulnerabilities beyond their severity ratings. You can block individual CVEs explicitly while ignoring others.

- **Continuous integration:** to integrate directly into your CI process (such as Jenkins). This way, it can find and report problems before they ever make it into production. In some cases, when a package with an open CVE is reported, it would be an excellent feature to receive a report with the package version that has the fix, giving developers clear insight into the vulnerabilities present in every build. These plugins should allow you to define and enforce your vulnerability policies at build time.

- **Compliance:** the Center for Internet Security (CIS) Docker and CIS Kubernetes Benchmarks provide guidance for establishing a secure configuration of a Docker container. In short, this benchmark provides the best security practices for deploying Docker. Having a solution with built-in checks to validate the recommended practices from this benchmark is another must. In parallel to this, the solution should include an extensive list of configuration checks for the host machine, Docker dæmon, Docker files and directories. Organizations would be able to enforce Trusted Registries and Trusted Images. And when configured, it should enforce that the images from these trusted lists are the only ones deployed on production servers.

- **Cloud native firewalls:** as workloads move to hybrid or cloud deployments, you'll need a platform to enable security teams to move beyond the manual management of whitelisted IP addresses by offering firewalls for cloud native environments—that is, having both layer 3 and layer 7 firewalls that automatically learn the network topology of your applications and provide application-tailored microsegmentation for all of your microservices.

- **Access control:** the ability to define and enforce policies governing user access to both container and workload management resources, limiting specific users to individual functions or APIs. This allows you to specify

access policies to container resources without the need to create new identities and groups. You can monitor detailed user access audit trails, action types, services requested and more from the console.

- **Analytics:** to visualize all relevant data and enable you to enforce standard configurations, container best practices and recommend deployment templates. This way your containers will remain compliant to industry or company policies.

Summary

DevSecOps is a natural and necessary response to the bottleneck effect introduced by older security models layered on top of modern CD pipelines. Its goal is to bridge the gap between IT and security while also ensuring fast and safe delivery of code. It is meant to address the security concerns in every phase of the development lifecycle. As more organizations rely on containerized applications to keep operations up and running, security efforts outside traditional methods are crucial to prevent costly downtimes. ■

PETROS KOUTOUPIS, ***LJ*** **Editor at Large, is currently a senior performance software engineer at Cray for its Lustre High Performance File System division. He is also the creator and maintainer of the RapidDisk Project. Petros has worked in the data storage industry for well over a decade and has helped pioneer the many technologies unleashed in the wild today.**