Docker containers in DevOps: Risks and Best Practices

Rafi Youssef: rafiy@kth.se

2nd May 2022

Abstract

The DevOps paradigm ensures that a company's growth and processes are in sync. In terms of security, DevOps security technologies are useful for transmitting information on external attackers that can be linked back to development, allowing for faster vulnerability eradication. It's an especially important step in cloud installations, as release intervals might be as short as 24 hours. This essay discusses the differences between DevOps and DevSecOps. On the other hand, it presents the challenges and best practices of Docker Containerization of DevOps.

Contents

1 Introduction

Software development (Dev) and IT operations (Ops) are combined in DevOps. Its goal is to abbreviate the systems development life cycle and provide high-quality software delivery on a continual basis [1]. DevOps is an add-on to Agile software development, with numerous DevOps features originating in the Agile approach. Figure 1 shows the life cycle of DevOps [2].

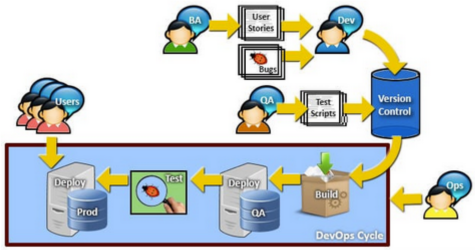

Figure 1: DevOps cycle

A good DevOps cycle would begin with:
• The developer is writing code.
• In a QA environment, creating and delivering binaries.
• In a single seamless flow, test cases are executed and then deployed to production [2].
Obviously, automation of Build, Deployment, and Testing is a big part of this method. In a DevOps cycle, the use of Continuous Integration (CI) tools and Automation Testing tools becomes the norm. DevOps is linked to Agile software development because DevOps was proposed by Agile practitioners as a way to extend the approach into production. The approach has even been dubbed a counterculture to the ITIL-endorsed IT service management approaches. There is no recognised structure for DevOps [3].

Organizations should search for DevOps toolchains that increase collaboration, eliminate context switching, incorporate automation, and employ observability and monitoring to ship better software, quicker. An all-in-one toolchain or an open toolchain are the two main approaches to a DevOps toolchain. An all-in-one DevOps solution offers a full solution that does not usually require the use of extra third-party technologies. With multiple tools, an open toolchain can be customised for a team's needs. An open toolchain, according to Atlassian, is the best solution since it can be modified with best-of-breed technologies to meet an organization's specific needs. This method frequently results in increased time effciency and a shorter time to market [4].
Regardless of the DevOps toolchain used by a business, the correct tools must be used to handle the main phases of the DevOps lifecycle:
• Plan,
• Build,

• Continuous integration and deployment,
• Monitor,
• Operate,
• Continuous feedback.

The selected technologies in an open DevOps toolchain touch several aspects of the DevOps lifecycle. The sections that follow highlight some of the most popular DevOps tools, however this list is constantly updated due to the nature of the business. Every quarter, suppliers announce new integrations and, in some cases, condense their solutions to focus on a single problem for their users [4].

Containerization is the process of putting a software component, as well as its environment, dependencies, and configuration, into a separate unit known as a container. This allows an application to be deployed reliably in any computer environment, whether on-premises or in the cloud [5]. Virtualization is used by DevOps teams to construct virtual machines (VMs), which are emulation of hardware and software setups. Figure 2 clarifies the difference between Containers and virtual machines [6].

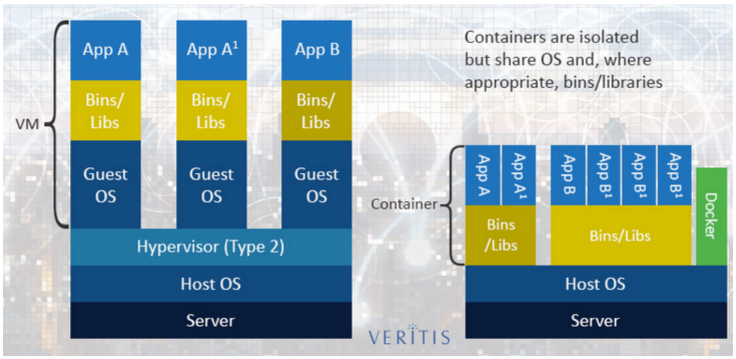

Figure 2: Containers vs. VMs

Docker is a software framework for constructing, executing, and managing containers on servers and in the cloud. It is a subset of the Moby project [7]. Docker containers are an excellent fit for DevOps, offering considerable benefits over virtualization and bare-metal deployment. They are simpler and quicker to deploy, use fewer resources to run, are easier to maintain, and are more versatile in general. These benefits enable DevOps teams to break down applications into microservices, each of which can be updated and delivered quickly, increasing development speed and agility. Containers also allow DevOps teams to standardise the packaging, delivery, and deployment of programmes throughout the development lifecycle [8].

In this essay, the similarities and differences between DevOps and DevSecOps have been presented. Also, the risks and best practices Docker containers in security have been discussed.

2 DevOps Methodology

Its goal is to bring together development, quality assurance, deployment, and integration activities. DevOps is a philosophy that tries to create a culture of collaboration amongst previously disparate teams. Traditionally, software development and deployment have been handled by two different people or departments. DevOps strives to increase productivity by removing the barriers that exist between these two stages of software development. DevOps is a vital factor in optimising time and resources for greater productivity, comprehension, and training for any technical specialist [9].
DevOps is a way for bringing programmers and system administrators closer together during the software development process. A DevOps engineer is a specialist who works at the crossroads of these two disciplines. The fundamental goal of a DevOps engineer is to improve the predictability, effciency, and security of software development to the highest feasible level. DevSecOps is a branch of the DevOps idea that, in addition to automation, includes code quality and reliability assurance [10]

3 Which Is Better: DevSecOps or DevOps?

3.1 Similarities

Both DevOps and DevSecOps may be able to leverage AI to automate phases in the application development process. It can be done in a DevOps style using auto-complete code and anomaly detection across several devices. Automated and regular security checks, as well as anomaly recognition, can assist in proactively identifying vulnerabilities and security threats, even in complex and distributed environments, thanks to DevSecOps [11].
Continuously recording and monitoring application data to resolve faults and promote improvements is a key component of DevOps and DevSecOps methodologies. Access to real-time data is critical for improving application performance, minimising the attack surface of the application, and strengthening the organization's overall posture [12].
DevOps and DevSecOps require a collaborative culture to achieve development goals such as rapid iteration and development that does not jeopardize the application environment's health and security. Both of these solutions include bringing together previously compartmentalized teams to increase visibility across the application's lifecycle, from planning to execution monitoring. Let's look at the differences between DevOps and DevSecOps in the section below [11].

3.2 Differences

Throughout the application development and deployment process, DevOps emphasizes collaboration between development and testing teams. Teams from development and operations work together to implement unified KPIs and tools. The goal of a DevOps approach is to increase the frequency of deployment while

maintaining the application's consistency and productivity. A DevOps engineer thinks about how to deploy updates to an application with the least amount of disruption to the user experience. Because they are so focused on improving delivery speed, DevOps teams don't always focus on avoiding security threats along the way, which can put the application and the organization's resources at risk [12].

As teams recognized that the DevOps approach didn't adequately handle security concerns, DevSecOps evolved from DevOps. Rather than retrofitting security into the build, DevSecOps arose as a method for managing security before, throughout, and after the development cycle. Application security is started at the start of the build process rather than at the conclusion of the development pipeline with this method. A DevSecOps expert uses this new methodology to ensure that programs are secure against cyberattacks before they are delivered to the customer and that they remain secure during application updates. DevSecOps emphasizes the need for developers to write code with security in mind, and it aims to address the security challenges that DevOps does not address. Understanding the differences between DevOps and DevSecOps will only aid you in determining which methodology is best for the projects your company works on [13].

The Figure 3 compares between DevOps and DevSecOps [14].

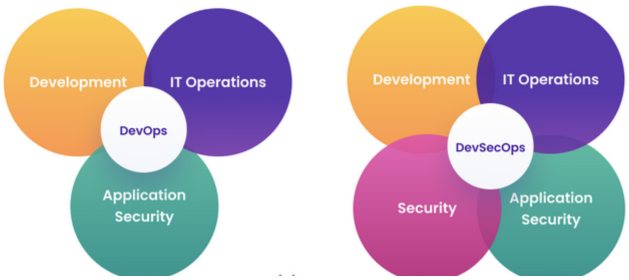

Figure 3: DevOps vs. DevSecOps

4 Challenges for Docker containers

Companies have long used virtual machines (VMs) or bare-metal servers to run their applications. For that infrastructure, security meant safeguarding your program and the host it was running on, as well as protecting the application while it was operating. Containerization brings with it a slew of new issues that must be handled.
(1) Microservices are enabled by containers, which increases data volume, network complexity, and access control complexity.
(2) Containers rely on a base image, and determining whether the image is secure or insecure can be diffcult. Vulnerabilities in images can also spread to all containers that employ the susceptible image.
(3) Containers have brief lives, making it challenging to keep track of them, particularly during runtime. A lack of visibility into a constantly changing container environment poses another security vulnerability.
(4) Unlike virtual machines, containers aren't always isolated from each other. The breach of one container can lead to the compromise of others.
(5) Another place where security issues exist is container configuration. Is it possible that containers are running with elevated rights when they shouldn't be? Is it possible that images are launching services that extend the attack surface? Are images capable of storing secrets?
(6) Given the fast-paced nature of container settings, compliance, as one of the most important security drivers, can be especially diffcult. In a Docker context, many of the conventional components that helped verify compliance, such as firewall rules, take on a new meaning.
(7) Finally, traditional server workload security solutions are unprepared to address container security threats and issues. [15]

5 Best practises for Docker security

(1) Always use the most recent Docker version.
(2) Make sure only trusted users are members of the Docker group to give them power over the Docker daemon.
(3) Make sure you've set up rules to provide an audit trail for Docker daemons, files, and directories.
(4) To reduce the possibility of traffic interception, utilize registries that have a valid registration certificate or use TLS.
(5) Run your containers as a non-root user as a best practice (UID not 0). Containers are started with root rights as the root user inside the container by default.
(6) Create a robust governance policy that mandates regular image scanning. Before continuing to the build step, stale photographs or images that haven't been scanned in a while should be discarded or rescanned.
(7) Do not execute containers with the privileged flag, as this type of container will have access to the majority of the underlying host's capabilities. This flag

also overrides any CAP DROP or CAP ADD rules you've established.[16]

6 Conclusion

DevOps is the way of the future. Software development models go through a continual improvement cycle from time to time. You must accept it, comprehend it, and instill it in your mind. So that your automation efforts bring value to the chain and are lean enough to swiftly respond to changes, you must know the various automation and continuous integration solutions. You may be working on projects that require alpha, beta, and user acceptance testing (UAT) before being launched in production.

There are some dificulties in implementing DevSecOps. The first issue is one of people and culture. You may need to retrain members of your DevOps teams in order for them to grasp security best practices and how to use your new security tooling. In terms of culture, your teams must actually believe that they are equally accountable for the security of the software they design and deploy as they are for its features, functions, and usability. Finding the correct security tooling and integrating it into your DevOps workflow is a second difficulty. The more automated and integrated your DevSecOps tools are with your CI/CD pipeline, the less training and culture shift you'll need to undertake.

Choosing a more automated version of the security tools you've been using for years isn't always the best solution. Why? Because your development environment has most certainly changed significantly in recent years. Opensource software makes up 70% of the average modern software application.
Unfortunately, standard security methods were not built to effectively find vulnerabilities in open-source software. Modern cloud-native programs, likewise, run in containers that can be instantly started and stopped. Traditional security solutions, especially those claiming to be cloud security tools, are unable to effectively analyze the dangers of applications running in containers. Finally, What talents should you foster to fulfill DevSecOps' aims of releasing better software faster and detecting and responding to software defects in production faster and more efficiently? What KPIs should you use to assess the quality of your DevSecOps initiatives?

References

[1] Mike Loukides (7 June 2012). *What is DevOps?. O'Reilly Media.*

[2] *DevOps Testing Tutorial: How DevOps will Impact QA Testing?.* URL: https://www.softwaretestinghelp.com/devops-and-software-testing/. (visited on 28/04/2022).

[3] *What is DevOps? The ultimate guide.* URL: https://www.techtarget.com/searchitoperations/definition/DevOps. (visited on 27/04/2022).

[4] *Considerations for your DevOps toolchain.* URL: https://www.atlassian.com/devops/devops-tools/choose-devops-tools. (visited on 28/04/2022).

[5] *What are containers?.* URL: https://cloud.google.com/learn/what-are-containers. (visited on 28/04/2022).

[6] *Containers Vs VMs: Glance at Security Pros and Cons.* URL: https://www.veritis.com/blog/containers-vs-vms-glance-at-security-pros-and-cons/. (visited on 24/04/2022).

[7] *The Role of Docker in DevOps.* URL: https://dev.to/kodekloud/the-role-of-docker-in-devops-1con. (visited on 28/04/2022).

[8] *Use containers to Build, Share and Run your applicanions.* URL: https://www.docker.com/resources/what-container/. (visited on 22/04/2022).

[9] J. Noppen S. Jones and F. Lettice. *Proceedings of the 2nd International Workshop on Quality-Aware Dev Ops-QUDOS 2016.* pp. 7-11.

[10] Patrick Debois Gene Kim. *, John Willis, Jezz Humble. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations , 2016.*

[11] *What is the Difference Between DevOps and DevSecOps?.* URL: https://www.appdynamics.com/blog/product/devops-vs-devsecops/. (visited on 24/04/2022).

[12] *Devops Vs DevSecOps Comparison.* URL: https://www.wallarm.com/what/devops-vs-devsecops-comparison. (visited on 28/04/2022).

[13] *DevOps vs DevSecOps: What is the Difference?.* URL: https://www.bunnyshell.com/blog/devops-vs-devsecops. (visited on 25/04/2022).

[14] *DevOps Security best practices.* URL: https://snyk.io/learn/devops-security/. (visited on 28/04/2022).

[15] L. Chen. "Continuous Delivery: Huge Benefits," in: *IEEE Software,* 32.2 (2015), pp. 50–54.

[16] R. T. Yarlagadda. "DevOps and Its Practices." In: *International Journal of Creative Research Thoughts (IJCRT),* 9 (March 2021).