Docker containers in DevOps: Risks and Best Practices

Rafi Youssef: rafiy@kth.se

23rd 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 **4**

2 DevOps Methodology **6**

3 Which Is Better: DevSecOps or DevOps? **6**
3.1 Similarities 6
3.2 Differences 7

4 Challenges for Docker containers **8**

5 Best practises for Docker security **8**

6 Conclusion **9**

References **10**

1 Introduction

Software development (Dev) and IT operations (Ops) are combined in De- vOps. Its purpose is to shorten the system operations and ensure consistently strong delivery process.[1] . DevOps is an add-on to Agile software devel- opment, with numerous DevOps features originating in the Agile approach. Figure 1 shows the life cycle of DevOps [2].

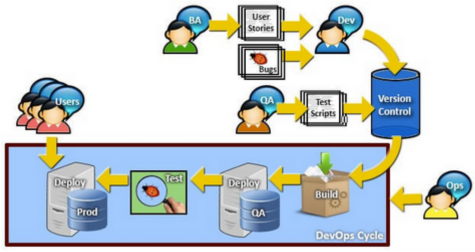

Figure 1: DevOps cycle

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].

Clearly, distribution,testing as well as automation of Build is a big part of this method. In a DevOps cycle, Automation Testing and Continuous Integration tools uses becomes the criterion.As a result of DevOps that was pro- posed by Agile practitioners, DevOps is linked to Agile software development in order to broaden the approach into production. The approach also been dubbed as a counterculture to the ITIL-endorsed IT service management approaches. There is no recognised structure for DevOps [3].

To release better software faster, organisations should look for DevOps toolchains that enlarge collaboration, exclude context switching, embrace au- tomation, and use observability and monitoring.

DevOps toolchain have two main approaches. The first approach are an all-in-one toolchain, this all-in-one DevOps solution mean that it can give full solution and it does not need to third-party technology. The second approach are an open toolchain that can be used to fit the need of the team's by using many tools. Ac-cording to Atlassian, an open toolchain approach is the greatest answer and the reason behind that because it can be used to customized with best-of-breed technology to adapt company's re-quirements. This strategy typically leads to a shorter time to market and higher productivity [4].

Whatever DevOps toolchain a company employs, the right tools must be utilized to handle the following key parts of the DevOps lifecycle:
”

•Build
•Monitor
• Continuous deployment, integration and feedback
•Plan
• Operate .”[4]

Numerous parts of the DevOps lifecycle are touched by the technologies in an open DevOps infrastructure. The parts that precede showcase several of the most common DevOps tools; however, owing to the sensitivity of the company, this list is frequently monitored. Manufacturers announce the new interfaces quarterly, and in certain situations, they provide simpler solutions to make sure that they can focus more on customers certain problems. [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].

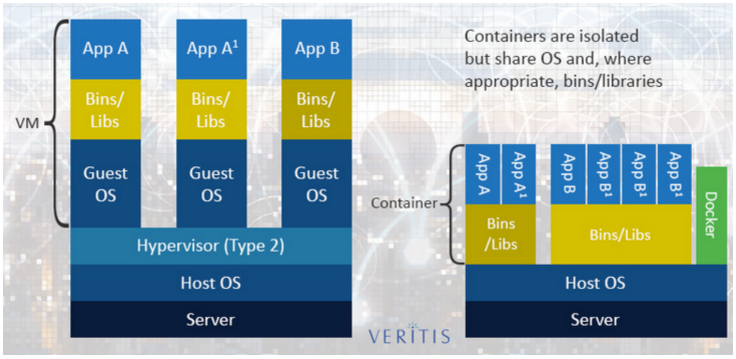

Figure 2: Containers vs. VMs

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 ben- efits 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 De- vOps teams to stan-

dardise the packaging, delivery, and deployment of pro- grammes throughout the development lifecycle [8].
In this essay, the similarities and differences between DevOps and DevSec- Ops have been presented. Also, the risks and best practices Docker containers in security have been discussed.

2 DevOps Methodology

Its purpose is to bring together tasks such as "quality assurance ,evolution , deployment, and incorporation activities"[9]. DevOps is a methodology aimed at fostering collaboration across previously distinct teams. By reducing the hurdles that arise between software development phases, the work performed for distribution and software development takes place through two separate individuals or teams. In this way, DevOps aims to boost output. DevOps is a critical aspect in maximizing time and resources for higher efficiency, training, and knowledge for experts. [9].
Through the software development process DevOps can be a way that bring system administrators and programmers closer to each other. The engineers who works in DevOps are experts that works at the crossroads of these two disciplines. The fundamental goal of a DevOps technician's job is to aim best level by increase software design reliability, ability, and protection. DevSecOps is an offshoot of DevOps that, in addition to automation, includes high quality code and improvement [10].

3 Which Is Better: DevSecOps or DevOps?

3.1 Similarities

Both DevSecOps and DevOps 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].

A crucial component of DevSecOps and DevOps techniques is constantly capturing and analyzing data files to rectify errors and stimulate enhance- ments. Actual data is essential for increasing software quality, reducing the application's security vulnerabilities, and bolstering the company's core position. [12].

DevSecOps and DevOps require a collaborative culture to achieve develop- ment goals such as rapid iteration and development that does not jeopardize the application environment's health and security. These approaches involve integrating formerly siloed groups together to improve transparency across the application development lifecycle, from beging to the end. The section below will discuss

the differences between DevSecOps and DevOps [11].

3.2 Differences

DevOps promotes interaction among testing and development all across the software development and implementation cycle. Developers and management teams collaborate to provide consistent KPIs and tools. DevOps strategys aim to enhance deployment speed while retaining software integrity and efficiency. A DevOps engineer considers how to distribute changes to a system with the minimal possible of downtime for users. DevOps teams do not often seek to avoid security vulnerabilities along the process because they are so concentrated on boosting delivery time, which can place the application and the institution's capabilities at risk [12].

As teams recognized that the DevOps approach didn't adequately handle security concerns, DevSecOps evolved from DevOps. Rather than retrofitting security in structure, DevSecOps arose as a method for managing security before, throughout, and after the development cycle. With such a strategy, application security is initiated at the beginning of the structure phase and not at the completion of the development cycle. This innovative approach is used by a DevSecOps specialist to verify that programmes are safe versus thefts until they are delivered to the client, as well as during development authority.

It is important in DevSecOps that developers write code with security, and they aim to address security challenges that do not occur in DevOps. Lastly it is important to realize the difference between DevSecOps and DevOps,with purpose to be able to decide which method is best for the projects your company is working on. [13].

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

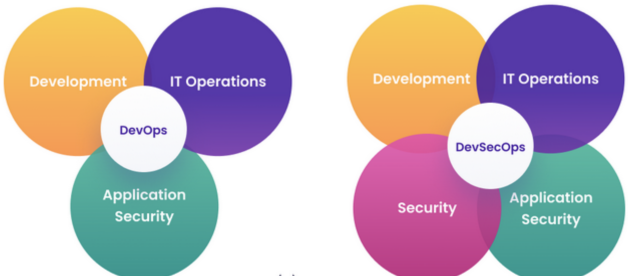

Figure 3: DevOps vs. DevSecOps

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 appli- cation 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 difficult. 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 chang- ing 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 difficult. 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 scan- ning. 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 con- tainer will have access to the majority of the underlying host's capa- bilities. 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, com- prehend it, and instill it in your mind.

You must explore the fundamental automating and secure development technologies such that your automated initiatives that improve the chain and also are thin to adjust rapidly changes faster. You might be conducting a research study that needs alpha, beta, and user acceptability testing (UAT) before going live.

There are some difficulties 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. Open- source 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 in- stantly started and stopped. Traditional security solutions, especially those claiming to be cloud security tools, are unable to effectively analyze the dan- gers 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).