WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

End to End Automation on Cloud with Build Pipeline

[1]Shreehari H S, [2] M Sreehari

[1]Assistant Professor, Dept. of ECE, SJCIT, Chikkaballapur, India, shreehari.harthi@gmail.com
[2]Student, Dept. of ECE, SJCIT, hikkaballapur, India, sreeharireddy536@@gmail.com

ABSTRACT –

Continuous integration/continuous deployment, or CI/CD, is a development and deployment technique that is commonly recommended by modern commercial software development organisations for new releases. This approach is frequently implemented using Kubernetes, a distributed application platform built on clusters. Although the general idea of CI/CD is pretty well established, there are many different ways to execute it. Systems might not be completely automated, and resources are dispersed among on-premises and cloud-based services. Additionally, even if a development pipeline could be designed to guarantee the security of the produced product, during execution the artefact might not be shielded from outside observers or cloud providers.

This article presents a full CI/CD pipeline based on Kubernetes that fills four implementation holes. First off, the pipeline encourages clear division of labour across responsibilities in development, security, and operations (also known as DevSecOps).

INTRODUCTION

The security of both the deployed item that is finished and the pipeline that creates it have come under more scrutiny as applications have grown more complicated. The software development community has embraced several new paradigms, with the inclusion of security within the "DevOps" framework being the most notable. DevSecOps, a mashup of the words "development," "security," and "operations," is the combination of best practises from these three fields with an emphasis on automation. These systems produce "pipelines" that are automated or semi-automatic, where both source code and build artefacts (such as built executables) are examined by several processes to guarantee their overall security against recognised attack vectors. The primary categories of modern industry software development and deployment pipelines include a (CD). Automated testing is frequently used in continuous integration (CI) to verify that changes do not damage anything, i.e., that the code compiles and passes its test suite [2]. Peer review, lint tools that enforce readability and adherence to a coding standard, static analysis for known critical vulnerabilities (CVEs), and dynamic analysis of build artefacts for security or performance concerns have all been added to this process in more recent times.

BLOCK DIAGRAM

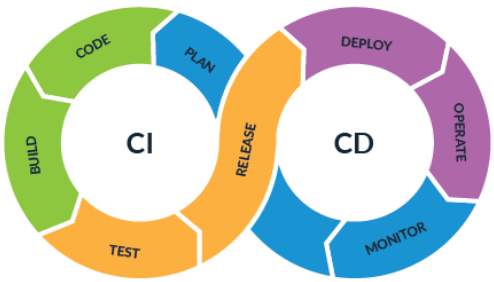

Fig: Continuous CI & CD

Our two-stage pipeline's CI stage. When a developer pushes to the remote repository, the pipeline automatically produces and sends a finalised artefact to the container registry (Harbor) (Harbor). Prior research has demonstrated scanning by Clair / Trivy; however, we do not use it in our prototype.

administrative rights on the actual system. Adoption is unfortunately difficult since application developers must change current code. Developers may use TEEs without having to re-architect their code bases thanks to Asylo's simple-to-use framework that acts as a "wrapper" around the code. This article illustrates how a CI/CD pipeline can be expanded with a new step that guarantees a completed binary may run in a secure enclave.

The contributions of our work are as follows:

- Our CI/CD pipeline is architected using Kubernetes, an increasingly popular Platform as a Service (PaaS) offering, and can be migrated trivially to different cloud providers or to on-premise infrastructure.
- We extend the CI/CD pipeline with an additional step that automatically deploys artifacts using Asylo, a TEE framework, to ensure the confidentiality of deployment artifacts.
- We provide a comprehensive evaluation that focuses on the security, performance, and limitations of our CI/CD pipeline.

To the best of our knowledge, this work is the first to describe a comparable CI/CD pipeline in detail, particularly one that is portable across cloud service providers. Furthermore, we have released our implementation as an open source project to allow others to extend our work.

DESIGN

This article demonstrates a CI/CD process that is repeatable, portable, and secure that is localised to a Kubernetes cluster. In a massive, clustered, distributed system, Kubernetes is an open-source tool for application development and deployment [8]. Although an open-source project, Kubernetes is overseen by the Cloud Native Computing Foundation (CNCF), which selects the system's life cycle and future feature set through a number of steering committees and special interest groups. In the PaaS architecture of cloud computing technologies (Infrastructure as a Service (IaaS), PaaS, and Software as a Service), Kubernetes is best referred to be a "platform." SaaS, etc. While a number of the largest cloud service providers currently offer managed Kubernetes-as-a-Service, such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform, it is also possible for individuals and organisations to run a Kubernetes cluster in their own on-premise systems, using a commercial cloud provider's IaaS offering, or on purpose-built offerings, such as those designated for use by governments.

Many corporate companies are worried about the overall security of their software development pipeline and final artefacts, as well as the overall repeatability and portability of their solutions [9] in order to prevent vendor lock-in [10]. In order to standardise CI/CD across contexts (such as several cloud providers) and technologies, the open source framework Tekton abstracts implementation details. Tekton and Kubernetes are used in our.

Continuous Integration

A software developer starts the CI process by committing and sharing changes, as shown in Figure 1. (i.e., pushing to a remote repository). Event listeners then automate each subsequent procedure in the pipeline, from creating the artefact to storing it for later usage. All of these processes are automated by Tekton. More specifically, a Dockerfile in the repository is used to create an image by a Tekton event listener using the most recent commit that was just submitted by the developer. Kaniko, a tool for creating containers inside of a Kubernetes cluster, builds the image while utilising Asylo to guarantee the privacy of the image. If all goes well, the final artefact (the image) is uploaded to Harbor, a local container registry for the Kubernetes cluster.

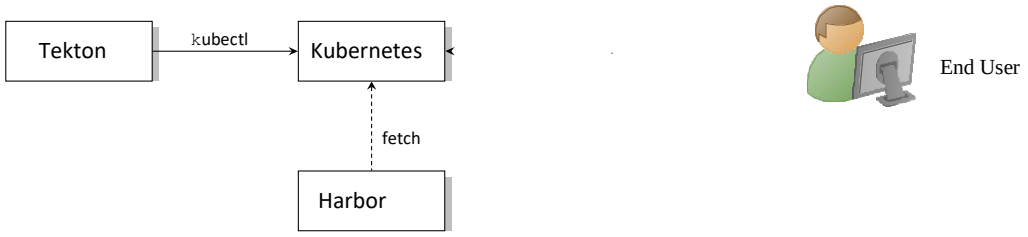

Fig. 2. The CD stage of our two-stage pipeline. Tekton triggers a second action and artifacts previously published to Harbor will be pulled and (re)deployed to our production namespace.

CD stage) begins execution immediately; otherwise, the CI pipeline fails, denotes the cause of the failure (i.e., the action that failed), and stops execution.

Continuous Deployment

Many companies distinguish between "delivery" and "deployment," with the latter referring to the automation of the actual deployment of final artefacts to production [14], [15]. Definitions of a CD pipeline vary. We might end the discussion by just entering the item into a container registry if "continuous delivery" serves as the definition. The CI step of our pipeline is the only one represented by this definition. Instead, our CD step sets our pipeline apart from earlier efforts (such as [12]) by deploying completed artefacts to operational production systems. After the CI stage has been successfully completed, Tekton starts the CD stage by making a reference to the most recent artefact pushed to Harbor. This process either begins the artifact's first deployment to the Kubernetes cluster or its redeployment.the utilize of centers gotten from a few other source such as a core/IP merchant or a foundry.

Implementation

Our pipeline is implemented using a variety of open-source applications and for-profit cloud services. On a Kubernetes cluster run by Google Kubernetes Engine, the majority of these components are installed (GKE). GKE is a Google Cloud Platform managed "Kubernetes-as-a-Service" feature. A CNCF-compliant Kubernetes cluster with master nodes overseen by the provider (in this case, Google Cloud Platform) and worker nodes is available to users

after provisioning. versions of the / cluster chosen by the user. Although the version control system we use, GitHub, is not cluster-scoped, GitLab may be utilised to include cluster-scoped version control into our workflow.

The implementation of our CI/CD pipeline is thoroughly explained in the next paragraphs of this section.

Advantages

- Automated deployments and standardized production environments are outcome of agility of business and IT collaboration
- Faster delivery of builds, features, and bug fixing thereby creating a continuous build pipeline
- Accelerated customer feedback cycles and collaboration results into high quality for feature/function delivery
- Continuous innovation because of continuous development of new ideas
- End-To-End Automation and Configuration Management for deployment pipeline using open source tools in days rather than in weeks
- Rapid delivery in Dev, Test, Pre-production and Production environments: Elimination of manual and repeatable processes based deployment

Applications

DevOps unifies the application delivery process into a continuous flow that incorporates planning, development, testing, deployment and operations, with the goal of delivering applications more quickly and easily. Provides an overview of this delivery flow and its emphasis on continuous, integrated services.

CONCLUSION

Binaries may now be delivered using containers to a Kubernetes cluster that supports secure enclaves offered by TEEs thanks to our CI/CD workflow, which is almost fully contained within a Kubernetes cluster. We specifically address the CD stage and the security of artefacts from untrusted system administrators, such as those of the cloud provider, in contrast to earlier work. By lowering the amount of external, external entities that need to be trusted, scoping our CI/CD locally within the Kubernetes cluster boosts the overall security of the system. Additionally, it enhances the system's repeatability and portability, two features that many corporate companies want for a range of security and commercial objectives.
using GitHub as an external, controlled version control system

REFERNCES

1. N. Tomas, J. Li, and H. Huang, “An empirical study on culture, automa- tion, measurement, and sharing of devsecops,” in 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security). New York, NY: IEEE, 2019, pp. 1–8.
2. M. Meyer, “Continuous Integration and Its Tools,” IEEE Software, vol. 31, no. 3, pp. 14–16, 2014.
3. J. Humble and D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Upper Saddle River, NJ: Addison-Wesley Professional, 2010.
4. W. R. Claycomb and A. Nicoll, “Insider Threats to Cloud Computing: Directions for New Research Challenges,” in 2012 IEEE 36th Annual Computer Software and Applications Conference. New York, NY: IEEE, 2012, pp. 387–394.
5. M. Sabt, M. Achemlal, and A. Bouabdallah, “Trusted Execution Environment: What It is, and What It is Not,” in 2015 IEEE Trust-com/BigDataSE/ISPA, vol. 1. New York, NY: IEEE, 2015, pp. 57–64.
6. P. Jauernig, A.-R. Sadeghi, and E. Stapf, “Trusted Execution Envi- ronments: Properties, Applications, and Challenges,” IEEE Security & Privacy, vol. 18, no. 2, pp. 56–60, 2020.
7. F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, “Innovative Instructions and Software Model for Isolated Execution,” in Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, ser. HASP ’13. New York, NY: Association for Computing Machinery, 2013. [Online]. Available: https://doi.org/10.1145/2487726.2488368
8. B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, “Borg, omega, and kubernetes: Lessons learned from three container-management systems over a decade,” Queue, vol. 14, no. 1, p. 70–93, January 2016. [Online]. Available: https://doi.org/10.1145/2898442.2898444
9. P. Perera, R. Silva, and I. Perera, “Improve Software Quality through Practicing DevOps,” in 2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer). New York, NY: IEEE, 2017, pp. 1–6.