ATLANTIS Integrated Framework: DevSecOps Approach

Ioannis Oikonomidis, Netcompany
Dr. Georgia Dede, Netcompany

Co-funded by the European Union

This project has received funding from the European Union's Horizon Europe Research and Innovation Programme under the Grant Agreement No. 101073909.

ATLANTIS Integrated Framework: The DevSecOps Approach

Ioannis Oikonomidis and dr. Georgia Dede (Netcompany)

The present paper analyses the integration approach that is followed in the context of the ATLANTIS project. The DevSecOps approach that is adopted in the continuous integration and deployment (CI/CD) process is further described. DevSecOps and CI/CD are vital in software integration by automating testing, ensuring security, and enabling seamless collaboration. They facilitate the smooth incorporation of diverse software components, promoting efficiency, reliability, and consistent quality in integrated systems. The ATLANTIS integrated framework, based on a DevSecOps principle, introduces security aspects into the components' development lifecycle and is implemented as a security- enhanced CI/CD process. Implementing security into the software development lifecycle is of paramount importance for reliable, robust, and resilient operation of Critical Infrastructures (CI).

1. Introduction

In the context of large-scale projects, software integration is vital as it enables seamless collaboration between diverse systems and applications, ensuring they work together harmoniously. It streamlines workflows, enhances efficiency, and fosters a cohesive environment where data and functionalities can be shared across the project, reducing redundancy, and maximizing the overall effectiveness. Software integration is an extremely difficult task that becomes even more complex when multiple, geographically dispersed development teams are interacting remotely. Thus, integration planning is critical towards ensuring that distant peers collaborate, interact, and contribute effectively. It also provides techniques and resources that facilitate, if not actually enable, their interaction. With the goal of framing, facilitating, and automating the development, integration, and validation tasks for the duration of the project, this paper may serve as a guidance to ATLANTIS developers.

Development, Security, and Operations (DevSecOps) is essential as it merges development, operations, and security, embedding security practices throughout the software development lifecycle [1]. This approach ensures early threat detection, faster response to vulnerabilities, and a culture of continuous security, crucial in safeguarding against evolving risks in modern digital environments. The goal of the presented and implemented ATLANTIS DevSecOps lifecycle design is to automate and integrate security throughout the entire ATLANTIS software development process. Furthermore, the concepts of Continuous Integration and Continuous Delivery (CI/CD) are integrated into the ATLANTIS development process in order to generate consistent and dependable software results [2]. In detail, GitLab CI/CD is used to implement the ATLANTIS CI/CD workflow, with the necessary security enhancements added to guarantee security presence throughout the entire DevSecOps cycle. Utilizing cutting-edge container and container orchestration technologies, the ATLANTIS CI/CD workflow is implemented through a private GitLab CI/CD instance hosted on the deployment platform used.

2. The Current State of Affairs in DevSecOps

CI is a developer practice that maintains a functioning system through incremental changes and frequent (usually daily) mainline integration using the right tools that support automation and a large number of automated tests. CI is pivotal in software development, allowing developers to merge code changes into a shared repository regularly. It automates testing and builds, enabling early bug detection, faster identification of integration issues, and ensuring that the codebase remains stable, facilitating efficient collaboration among team members. CI requires developers to apply Test-Driven Development (TDD) along with ongoing refactoring. In this way, a developer makes sure his local copy of the code runs consistently when he is unit-testing and driving it.

The automated deployment of new system releases into the production environment is known as continuous deployment, or CD. After the above-described continuous integration process, CD takes care of automatically updating an already-running version of the system to minimize downtime when a system reaches a maturity level (as indicated by specific, predefined criteria).

When combined, CI/CD creates a pipeline that receives new developments and delivers a system that is updated and operational, hosted in a predetermined environment. CI/CD automates the entire software release process, from code integration and testing (CI) to deployment and delivery (CD). It ensures a rapid, reliable, and iterative approach to delivering software, fostering a streamlined workflow and consistent, high-quality releases. A CI/CD environment with GitLab has already been set up in ATLANTIS. The high-level CI/CD pipeline of GitLab is shown in Figure 1.

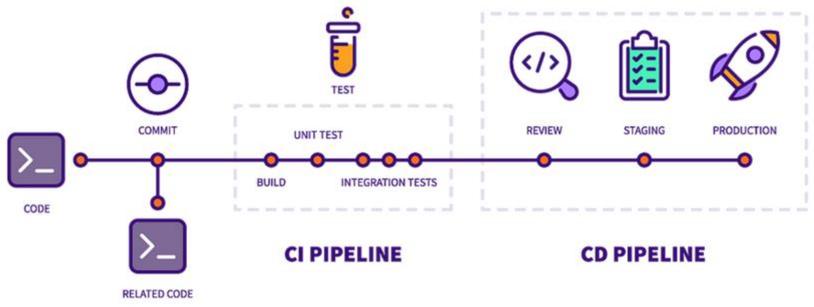

Figure 1. CI/CD pipeline (source: about.gitlab.com ***Error! Reference source not found.****).*

3. The Role of the ATLANTIS Integrated Framework

Modern software development leverages an agile-based Software Development Lifecycle (SDLC) to accelerate the development and delivery of software releases, including updates and fixes. DevOps and DevSecOps use the agile framework for different purposes. DevOps focuses on the speed of app delivery, whereas DevSecOps augments speed with security by delivering apps that are as secure as possible as quickly as possible. The goal of DevSecOps is to promote the fast development of a secure codebase.

Towards supporting the platform's integration efforts and the administration of the anticipated ATLANTIS platform releases, ATLANTIS implements DevSecOps practices. DevSecOps represents the integration of security practices into every phase of the software development lifecycle, emphasizing collaboration between development, operations, and security teams. This approach aims to embed security measures early on, enabling continuous security testing, threat detection, and remediation. Its ultimate goal is to deliver high-quality software continuously while expediting the systems development life cycle. Infrastructure security is integrated from the start and by design into DevOps, thanks to the integration of security by design provided by DevSecOps. Based on DevSecOps, agile DevOps techniques and methods, like CI/CD, incorporate security as a fundamental component.

The DevSecOps approach followed in the context of ATLANTIS is presented in the following figure, namely Figure 2.

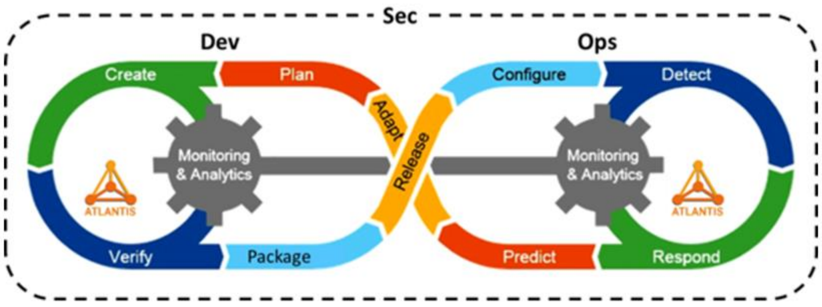

Figure 2. ATLANTIS DevSecOps.

4. The Research and Development Path in ATLANTIS

In the context of the ATLANTIS platform, Gitlab CI/CD pipeline is used and further adapted and extended for the purposes of the project. In software development based on continuous method, the developers continuously build, test, and deploy iterative code changes. This iterative process helps reduce the chance that they develop new code on buggy previous version. This method will lead to less human intervention from the development of new code until its deployment. The proposed CI/CD pipeline is illustrated in more details in Figure 3.

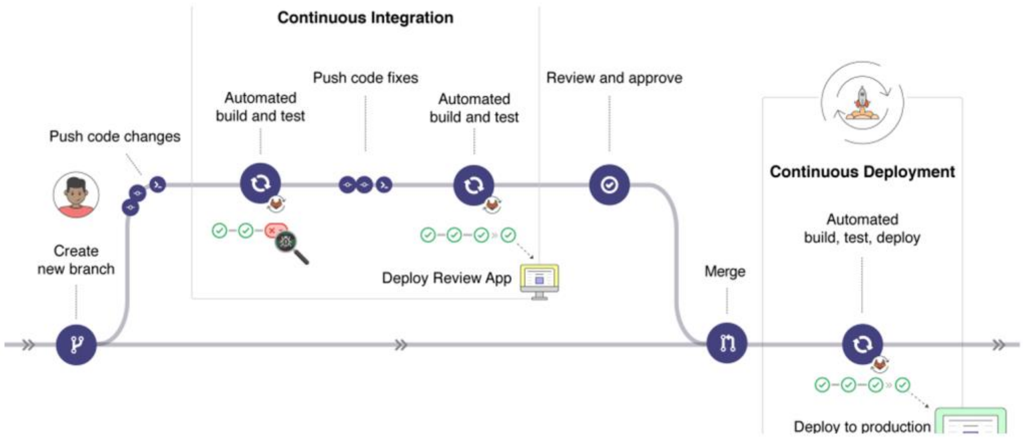

Figure 3. CI/CD pipeline steps (source: docs.gitlab [3]).

As shown in Figure 3, the main steps of GitLab's CI/CD process are further explained as follows:

- The software developer integrates a specific piece of software with the goal of integrating it with a module functionality.
- The developer presents unit tests for the pertinent code segment, assessing the consistency of the module's outputs.
- The developer pushes and commits changes to their local repository's code to a remote GitLab repository branch that is designated as a development branch within the CI infrastructure.
- Code merging is requested by the developer.
- The project-specific CI/CD pipeline is activated, and the CI platform runs the selected unit test or tests.
- The code is subjected to additional integrated system testing if unit testing is successful.
- Should the integrated system testing prove effective, the merge-request will be approved, incorporating the recently committed source code into the code master branch.
- If not, the merge-request is denied. In this situation, the developer needs to either update the unit testing itself or update the code to comply with the procedures.
- After a successful CD, i.e., building and deploying the software package through the source code management platform, the new code is running on the deployment infrastructure and is subject to user testing and production.

5. The Challenges and Barriers

During the development of the ATLANTIS integrated platform, factors such as the requirement for dependable and simple component installation act as major motivators due to the inherent complexity of ATLANTIS and its components. The introduction of the cloud-native Kubernetes [3] approach also brings up several integration-related issues that have to be fixed to ensure that an application is easy to install and maintain over time, as well as robust and integrity during deployment operations.

Initially, before deploying an application as a container (or group of containers) to Kubernetes, a set of Yet Another Markup Language (YAML) files are required to be created. These YAML files describe various components of the deployment [4], exposed service if any, employed storage, config maps, and secrets with important deployment information (e.g. values for environment variables) and jobs for necessary initialization and other recurring tasks. The deployment of the custom applications requires these files. In the context of the applications as well as external components like Keycloak and Kafka, an automated configuration and deployment flow is used to necessitate manual file authoring.

An automated workflow for the simple and default configuration and deployment of the ATLANTIS platform has been developed and the aforementioned factors have been appropriately considered.

6. The Benefits and Impact

In addition to the DevOps component, ATLANTIS applies DevSecOps by integrating security improvements into the CI/CD workflow. In this context, security constraints are present throughout the entire software lifecycle, and security picks are introduced even before the design phase. DevSecOps fosters early vulnerability detection, integrates security across the development lifecycle, and cultivates a collaborative culture. This approach results in minimized risks, faster threat response, and a proactive security stance, enhancing the overall security and reliability of software products. In this context the main benefits are secure hosting and security in the DevOps lifecycle.

Secure Hosting

The company hosting the GitLab CI/CD instance (the ATLANTIS partner: Synelixis Solutions S.A.) upholds information security regulations in line with ISO/IEC 27001:2019 [5].

Security in DevOps Lifecycle

SDLC is being revolutionized by DevOps, which applies manufacturing quality control knowledge to application design and production. DevOps increases consistency through process optimization and a decrease in human error by emphasizing the use of automation to handle many of the processes involved in developing, testing, and deploying software. There are numerous benefits to this method for SDLC security as well.

By automating security testing at key stages of the development lifecycle with tools like Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST), security in DevOps can be strengthened (DevSecOps approach). Developers can quickly and consistently create software error-free by incorporating security into DevOps, which helps to shorten release cycles and enhances the calibre of each release [6].

Application security testing techniques like SAST and DAST are used to identify security flaws that could leave an application open to attack:

Known as “white box testing”, SAST enables developers to identify security flaws in the source code of the application early in the software development life cycle. Without running the code, SAST is used to verify it. Additionally, without running the underlying code, it guarantees compliance with coding standards and guidelines. By adding a static analyser to the continuous integration/continuous development loop, it is possible to prevent programming errors from arising at an advanced stage of the development process.

"Black box" testing, or DAST, looks for flaws and security vulnerabilities in an application that is currently in use. Since it needs an application that has been built and tested, it is completed later in the development lifecycle. The application’s source code, as well as the technologies and frameworks, are unknown to the tester. DAST is a software testing tool that examines the security of developed software by feeding it malicious data in an attempt to find security flaws and determine which ones are connected to the actual deployment of an application.

There are advantages and disadvantages in both SAST and DAST testing methodologies. They are most useful at different stages of the software development life cycle and identify various kinds of vulnerabilities. SAST needs to be run frequently and against all files that contain source code. One the other hand DAST requires to be run in a production-like environment on an application that is currently in use. It seems therefore that incorporating both SAST and DAST into an application security testing program is the best course of action. The techniques of SAST and DAST are complementary, and both should be implemented for thorough testing [7].

7. Future Outlook

DevSecOps plays a pivotal role in CI/CD by ensuring that security measures are seamlessly integrated into the automated processes of continuous integration and continuous deployment. By embedding security practices into every stage of CI/CD pipelines, DevSecOps minimizes vulnerabilities, enables automated security testing, and facilitates quick remediation of issues. This integration ensures that the rapid development and deployment cycles of CI/CD remain fortified against potential security threats, promoting a culture of proactive security while maintaining the agility and speed of the development and release process.

Integrating DAST and SAST into CI/CD workflow involves incorporating these techniques seamlessly into the development pipeline. SAST tools analyse source code for vulnerabilities without executing the program, and they can be integrated into CI/CD by automating scans during the build process, providing immediate feedback to developers on code changes. DAST, on the other hand, tests running applications for weaknesses by simulating attacks and can be integrated into CI/CD by automating tests against staging environments, ensuring that applications are assessed for security issues before reaching production. Orchestration tools within CI/CD pipelines schedule and execute these scans at relevant stages, while unified reporting and policy enforcement ensure that identified vulnerabilities are promptly addressed and compliance standards are met before deployment.

CI/CD is a technique that involves automating app development stages in order to regularly deliver apps to customers. Continuous Integration, Continuous Delivery, and Continuous Deployment are the key ideas associated with CI/CD [8].

Security testing can be added to the CI/CD process for the project's different code repositories, as depicted in Figure 4 below. All the SAST/DAST checks will be carried out on a Gitlab instance that hosts the code repositories.

Taking into consideration that source code might not be feasible to be shared by uploading it on ATLANTIS GitLab, it can be the case that only DAST will be employed and not SAST.

ATLANTIS DevSecOps in CI/CD demonstrates scalability by accommodating increased workloads and diverse applications while ensuring security measures are consistently applied. Its adaptability lies in the ability to swiftly integrate new security tools, update policies, and adjust testing methodologies to address evolving threats and compliance standards. This ensures a flexible, responsive framework capable of meeting changing security needs without disrupting the development and deployment processes.

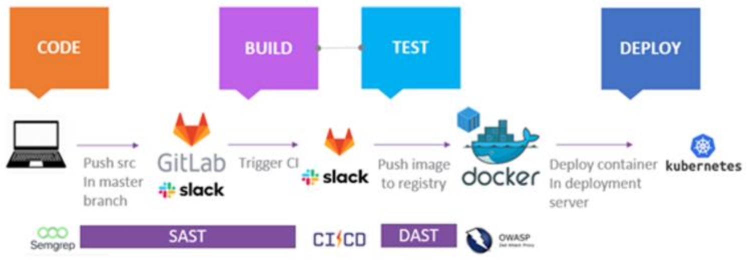

Figure 4. SAST/DAST in the development lifecycle.

8. Conclusions

This paper describes the ATLANTIS integrated framework focusing on the DevSecOps approach in CI/CD. DevSecOps and CI/CD are integral in the software development lifecycle, ensuring efficient, secure, and agile practices. They facilitate continuous integration, testing, and deployment, expediting delivery, enhancing security, and fostering a culture of collaboration and improvement throughout the development process. One of the ATLANTIS project's key pillars is the automation and integration of security at every stage of the ATLANTIS software development lifecycle; for this reason, the DevSecOps methodology receives particular attention.

Additionally, this paper outlines a strict plan for taking part in and managing the DevSecOps process in the context of the ATLANTIS integrated framework. The ATLANTIS Consortium provides the infrastructure for the DevSecOps environment, which is hosted on a dedicated host platform. Furthermore, it is determined that the concept of CI/CD has to be incorporated into the ATLANTIS development process in order to generate consistent and trustworthy software results. Towards this end, specialized tools and workflows should be used, and a highly dependable development, testing, and deployment environment will be set up. Automation is critical to this process, since it will expedite the completion of repetitive tasks that have to be completed each time a new feature of ATLANTIS is added, expanded, or integrated with the platform as a whole. Without excluding local development and deployments on partners' premises, ATLANTIS integrated framework uses Docker for application containerization, Kubernetes for container orchestration, and GitLab solutions for continuous integration and continuous delivery, source code management, and issue tracking.

References

[1] E. S. G. S. J. d. C. D. A. &. K. U. Soares, "The effects of continuous integration on software development: a systematic literature review," *Empirical Software Engineering,* pp. 27(3), 78, 2022.

[2] N. Singh, https://itnext.io/autoscale-gitlab-runners-in-aws-gcp-and-azure-b9b6f4d7e17, 2022.

[3] "Kubernetes," 2023. [Online]. Available: https://kubernetes.io/.

[4] M. L. Bernardi, https://docs.gitlab.com/ee/ci/index.html, 2022

[5] "ISO 27001," 2022. [Online]. Available: https://www.iso.org/standard/27001.

[6] M. A. Ç. C. E. &. S. A. Aydos, "Security testing of web applications: A systematic mapping of the literature," *Journal of King Saud University-Computer and Information Sciences,* pp. 34(9), 6775-6792, 2022.

[7] "Red Hat," 2022. Available: https://www.redhat.com/en/topics/devops/what-is-ci-cd.

[8] Veracode, 2022. [Online].