# Building a Secure DevOps

**Karishma Pooj**

***Abstract*— DevOps is an emerging paradigm that aims at integrating developer and operations personnel together in order to continuously improve the collaboration, productivity, automated workflows and performance of the application.**

**Introducing security in DevOps will allow the application to be built in a manner that will enable it to withstand security breaches at a better level. Security can be integrated right from the inception of the DevOps cycle. This paper tries to cover the various security considerations one must evaluate before adopting DevOps at various phases of SDLC.**

***Index Terms*— Continuous Integration, DevOps, Security, Software Development Life Cycle.**

## I. INTRODUCTION

DevOps is a lean and agile principle based approach in which the business management, development, operations and quality assurance department collaborate to deliver software in a continuous manner to enable the business to expand more quickly, thereby seizing market opportunities and reducing customer feedback time. DevOps isn't a tool or a product; it's an approach to operations. By uniting the Development and Operations teams to automate and standardize the processes for infrastructure deployment, you achieve faster innovation, accelerated time to market, improved deployment quality, better operational efficiency, and more time to focus on your core business goals. It’s an organizational shift which has changed the traditional system of distributed groups performing functions separately to cross- functional teams working on continuous operational feature deliveries. This approach reduces problems due to miscommunication between team members and accelerates problem resolution that directly helps deliver output faster and continuously. Based on business requirements DevOps can be applied to a selected delivery model by tailoring the environment and product architecture. Since enhancement and bug fixes are frequently deployed within short periods of time in this process, lesser importance is given to security aspects. In a normal development life cycle security is considered just before deployment. This means that the application does not undergo rigorous security testing making it vulnerable to attacks.

Application security must be incorporated into all supporting business processes and a dedicated security team must establish alignment with the business, automating the discovery of vulnerabilities, and providing automated application security testing. Additional complexities with application security arise when organizations move towards automation. To improve the protection of applications from vulnerabilities and potential attacks, there is a need to identify ways to integrate application security within their DevOps model. Incorporating security from the start of the development process is the basic solution to achieve a secure application [Fig. 1.1].

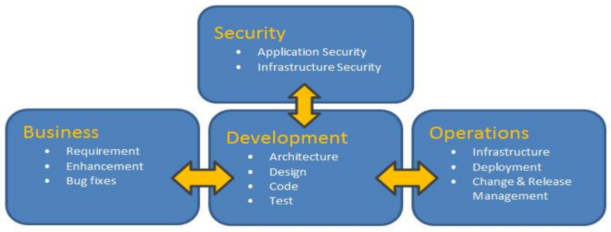

Fig. 1.1 Security in DevOps

## II. LITERATURE REVIEW

Following papers were referred to as a part of literature review:

### *A. Understanding DevOps*

Manish Virmani [1] proposed how various principles of DevOps, if adopted in release lifecycle can really optimize the organization’s capability of software delivery. This paper also highlights the fact that organizations who adopt DevOps principles have an edge over organizations not riding DevOps.

### *B. Strengthening harmonization of DevOps*

Syed W Hussaini [2] proposed a systemic approach of an IT DevOps strategy which ensures that the program’s core objective is focused on the program lifecycle and its timely delivery. Systematic models to ensure maximum harmonization between Dev and Ops was described in order to craft an end to end framework.

### *C. Devops Automation and Integration*

Johannes Wettinger [3] explained the problem by combining artifacts of different kinds in order to automate the deployment of Cloud applications using various components. This problem was addressed using an automated transformation framework to generate TOSCA standard-based modeling artifacts from different kinds of existing DevOps artifacts.

### *D. Application Security Testing for DevOps*

WhiteHat Security [4] have developed and explained the WhiteHat Sentinel platform that assists security teams in automating their application security efforts while embedding application security into the DevSecOps model. This paper also highlights how to incorporate application security throughout the software development lifecycle (SDLC) with complete transparency to DevOps team.

### *E. Rackspace DevOps Advisory Service*

Rackspace professionals [5] have explained through their architecture how various tools have been integrated to build their DevOps. It explains how developers can use Jenkins to build and test a code continuously. Tools like Chef, Puppet and SaltStack are used for configuration management which can be managed by GitHub. New Relic gives detailed performance metrics for every aspect of environment, in

**Karishma Pooj**, Department of Information Technology (Information Security), K. J. Somaiya College of Engineering, Mumbai University

real-time. This will help developers know how the entire application is performing and help them identify bottlenecks.

## III. APPLYING SECURITY PRACTISES TO DEVOPS

Achieving security in following DevOps process [Fig. 3.1]:

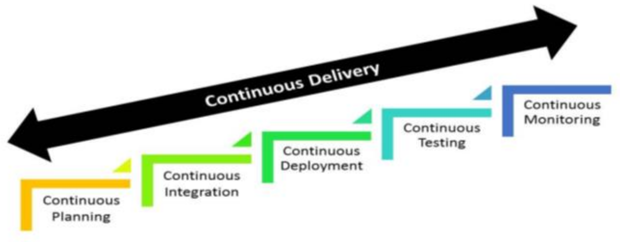

Fig.: 3.1 DevOps Process

### *A. Continuous Planning*

Business plans and requirements should be agile so that they can be quickly adjusted to the changing market conditions. From the security point of view the phase is ideal for research on the security technologies and processes that can be run throughout the continuous development and deployment process. Decisions of code, frameworks, libraries and costing for the elements that will be used throughout the cycle are made at this stage. Adding security practices at this early stage of planning will enhance the security of an application. Having prior knowledge of the requirements and -plan will make it easy to add security in the DevOps cycle.

### *B. Continuous Integration*

DevOps places the developer at the center of the process. Communicating security importance to the developers has always been a concern. This continuous interaction between the developers and security team can be bridged using an automated portal to review the code. Source code review can be done at this stage. Developers can write their codes, upload them on the portal and perform continuous code review instantly with issues. This stage can be achieved by automating the process where developers can add their code on a portal to be scanned rather than manually sending it for review which is time consuming. This will be a repeatable continuous process all across the development cycle.

### *C. Continuous Deployment*

This is the heart of DevOps and forms the critical piece of overall software delivery optimization. Security should be an integral part of the continuous deployment process. Automating the infrastructure vulnerability assessment can enhance the deployment cycle. Majority of delays in deployment depend on the infrastructure readiness. Having a continuous automated security review of infrastructure where the operations teams instantly get results from vulnerability assessment scans on the servers will cut the delay time in getting the production servers ready. This continuous review for infrastructure will work towards achieving continuous deployment.

### *D.Continuous Testing*

Automation is the building block of DevOps. Security testing should first fit into the standard automated continuous process along with the tools capable to produce results in short time. Automating security test cases will enable the completion of tasks at a faster rate. Various build tools such as Jenkins; Bamboo etc. can be integrated with Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST). This automated integration testing will help achieve goals and eliminate maximum vulnerabilities in the least possible time.

### *E. Continuous Monitoring*

Planning a replica of the production system can help monitor the continuous behavior of the application from security aspects. Penetration tests and Vulnerability assessment can be planned accordingly to monitor the impact on production servers. Once the replica servers are hardened the penetration activity can be performed on the same. This can eliminate the downtime required for performing these activates and provide a study of the production server's behavior based on these activities without performance impact. Planning and testing a replica of the production system early on allows for an opportunity to observe various quality parameters throughout.

## IV. PROPOSED ARCHITECTURE

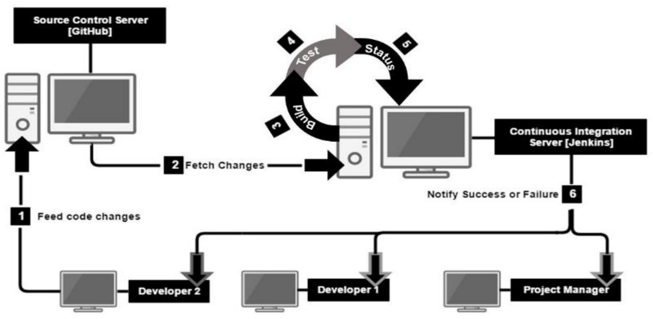
Source Control Server [GitHub]
Test
Status
Build
Continuous Integration Server [Jenkins]
Fetch Changes
Notify Success or Failure
Feed code changes
Developer 2
Developer 1
Project Manager

Fig. 4.1 SecureDevOps Architecture (a)

Fig. 4.1 highlights the architecture proposed which is an implementation of open source tools to implement Secure DevOps. The initial phase of the architecture is development of the application or any code changes. Code changes are evident in regards to customer's requirement, enhancements in the current application or any bug fixes. Once the application build is ready or the changes are made successfully, it is passed on to GitHub. GitHub acts as a centralised repository which is responsible for tracking all the code changes. Later on, the code will be provided as an input to Jenkins.

Jenkins tool is used to build the code as and when decided by the developer; can be taken on a weekly basis or monthly basis.

In the above archiecture, Jenkins is responsible for performing 3 opertions namely, Code build, Testing build and notifying the development team regarding the status of test performed.

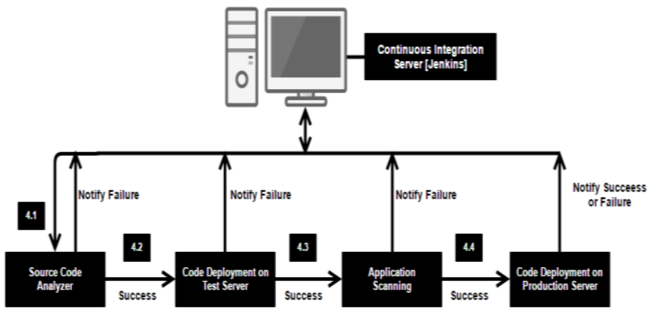
Continuous Integration Server [Jenkins]
Notify Failure
Notify Failure
Notify Failure
Notify Success or Failure
4.1
4.2
4.3
4.4
Source Code Analyzer
Success
Code Deployment on Test Server
Success
Application Scanning
Success
Code Deployment on Production Server

Fig. 4.2 SecureDevOps Architecture (b)

Fig. 4.2 highlights the test performed by Jenkins after successfully building the code. Once the code is successfully built, it is forwarded to the source code analyser tool. The tool will analyze the code supplied to it and detect and expose any security vulnerabilities that exist in the code. A threshold will be set based on logic to decide whether to continue with the current code or to make changes in the current code in order to mitigate the defects exposed by source code analyzer tool.
Once the code passes the defined threshold value, it is deployed on a test server. Code is finally tested by an application scanner (dynamic testing). If the code passes this test, it will be deployed on to the production servers using any existing code deployment tool. Any failures in the above steps would be notified to the stakeholders by Jenkins.

## V. IMPLEMENTATION

For implementation of the proposed model, open source tools were used. Jenkins is installed and configured to integrate other tools. This implementation details only include how to incorporate security elements into DevOps.

### *A. GitHub Integration with Jenkins*

GitHub is used for tracking the code changes as made by the developers. With the help of the GitHub plugin, Jenkins can easily pull source code from any GitHub repository that the Jenkins build node can access. Jenkins will be configured to "build when a change is pushed to GitHub". Also Jenkins hook URL is to be specified in the configuration in order to automatically build the code in Jenkins once a code change is detected in GitHub.

### *B. Grails installation*

This is a tool used to build the code once given as an input to Jenkins. For implementation the version used is 2.4.4.
Once Jenkins is downloaded and service is started, Jenkins can be configured to add Grails installation by adding details about Grails' installation location. Post build action can be implemented to call the SAST scan (VCG project as specified below).

### *C. VisualCodeGrepper Integration with Jenkins*

VisualCodeGrepper is a source code analyzer tool used for code review. There are no plugins in Jenkins to integrate this tool. So integration was done by invoking VisualCodeGrepper command line through Jenkins.
Create a project called VCG in Jenkins. Add a build step "Execute Windows Batch Command". A batch file is to be created which contains the command to invoke VisualCodeGrepper. This batch file is to be stored in the workspace folder of the VCG project in Jenkins.
Post build actions are to be created to save the results of the scan as well as to display a message in case of a successful and unsuccessful scan. Also to call DAST scan (ZAP project as specified below).

### *D. OWASP ZAP Integration with Jenkins*

OWASP ZAP is an application scanner used to dynamically scan the applications to perform dynamic application security testing. For this implementation, v2.4.0 is used. Create a persistent session of Zap. Set the context and policies for scan.
Create a project called Zap in Jenkins. Add build step "Execute ZAProxy". The Zap session created should be pasted in the workspace folder of the ZAP project in Jenkins. Post build actions are to be created to save the results of the scan as well as to display message in case of successful and unsuccessful scan.

## VI. CONCLUSION

DevOps is an emerging concept which deals with a continuous automated Application development cycle. The adoption of security solutions within this process can be leveraged as the foundational platform for a successful and secure DevOps solution. By embedding application security into the DevOps framework organizations will be able to deploy secure solutions continuously. The above implementation can be also materialized using proprietary tools for source code review as well as for dynamic application testing.

## ACKNOWLEDGEMENT

I am pleased to acknowledge my indebtedness to Prof. Sonali Patil for her invaluable guidance and perspective suggestions at every stage. She has been a perennial source of inspiration throughout paper work.

I offer my profound gratitude towards all staff members of K. J. Somaiya College of Engineering, Vidyavihar, Mumbai for providing me all academic facilities required to complete the work.
I would like to thank my colleagues, who have contributed to ease the understanding of this project and this paper by giving their time and taking keen interest in making this a success.

## REFERENCES

[1] Manish Virmani, "Understanding DevOps & Bridging the gap from Continuous Integration to Continuous Delivery", Fifth International Conference on Innovative Computing Technology(INTECH 2015), pages 78-82

[2] S. W. Hussaini, "Strengthening harmonization of Development (Dev) and Operations (Ops) silos in IT environment through systems approach", 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), October 2014, pages 178-183

[3] J Wettinger, U. Breitenbucher, F. Laymen, "Standards-based DevOps Automation and Integration Using TOSCA", IEEE/ACM 7th International Conference on Utility and Cloud Computing, 2014, pages 59-68

[4] White hat security, "Application Security Testing as a foundation for Secure DevOps", Dec 2016.

[5] Rackspace. "DevOps Configuration Management and Continuous Deployment Services." Internet: www.rackspace.com/en-in/devops, Dec. 12, 2013

[6] S. Sharma and B. Coyne, *DevOps for Dummies*. Hoboken, N.J : J. Wiley and Sons, 2015.

**Karishma Pooj**, Department of Information Technology (Information Security), K. J. Somaiya College of Engineering, Mumbai University