# A deep comparison of dependabot and snyk and how to integrate them into an organisation.

Harald Hobbelhagen, Jaldu Suvarchala

April 23, 2023

**Abstract**

In this essay, the reader gets an overview into the usage of software bots in modern development cycles. First, the usage of software bots is motivated. Later on, two bots are compared. These bots are then evaluated against challenges and benefits of bot usage in software development.

We certify that generative AI, incl. ChatGPT, has not been used to write this essay. Using generative AI without permission is considered academic misconduct.

## 1 Introduction

This essay aims to showcase how bots may help to overcome the gap between security and feature engineering in development cycles. To accomplish this, two bots (dependabot and snyk) are identified and compared with each other. Dependabot and Snyk both help address security matters in software development by managing dependencies and identifying security vulnerabilities. We will also explain how these tools can integrate into an organization.

## 2 Software-Bots as State of the Art in DevOps

Bots are rapidly becoming means of interacting with software services. The extensive use of messaging apps (like Facebook Messenger, Slack, etc.), as well as the advancement of natural-language processing, which many bots leverage to communicate with their users. The development of machine-learning algorithms that can analyze data from multiple domains using big data is one of influence the growth factors of big data. Developers may easily create a user interface(UI) for engaging with these algorithms and data using bots. There are several bots that can be used in each phase of the DevOps cycle such as Code Review bots, Testing bots, Integration bots, etc. [LSZ18]

One of the biggest issues in contemporary software development is weak dependencies. it is not only crucial but also difficult. Software projects often rely on a variety of external dependencies, making it difficult for developers to regularly monitor and check for corresponding security flaws that affect their project dependencies. [ACSM21] To address security flaws in dependencies, software bots may help developers and organisations to address security flaws on time. This paper will showcase two bots which scan projects to locate security flaws in the project dependencies and offers automatic solutions to assist developers in addressing those flaws.

## 3 Introducing the tools

Before comparing Snyk and dependabot, we will introduce the tools shortly and explain their purpose. This illustrates the limitations of each tool and the reasons for choosing one over the other or use them in combination.

### 3.1 Dependabot

Dependabot is a tool offered by GitHub that checks dependency files. In case of the identification of security vulnerabilities, dependabot can be configured to do certain actions like creating pull requests.

It can deal with various programming languages or configuration and deployment tools like Terraform and docker. [depb]

Figure 1 shows a demonstrative workflow with dependabot. It checks releases in many package managers and compares it with the dependency configuration in the repository. Afterwards, it may create a pull request with a corresponding change. Organisations may review the pull requests or execute automated tests on it.

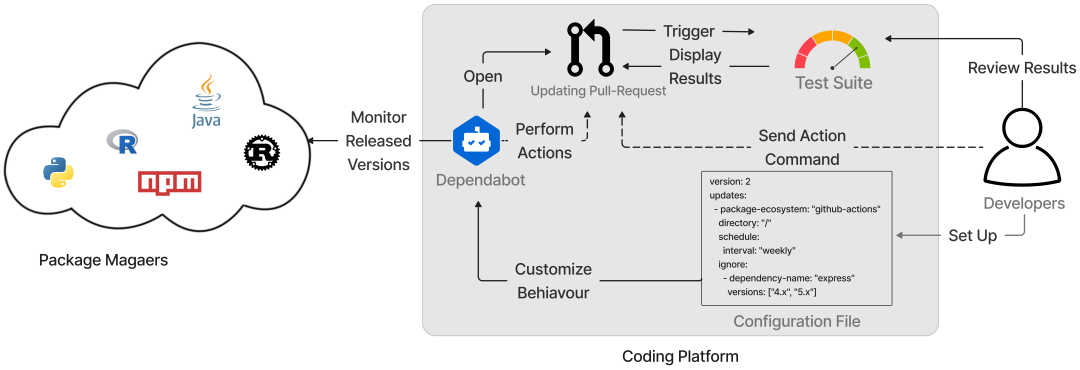


Figure 1: Demonstrative setup of dependabot inspired by [CH22]

Dependabot-Core is the library that checks the security and version updates in Dependabot. Use it to produce automated pull requests, updating dependencies for Java, JavaScript, Python, PHP, .NET, Elm, Go, Rust, Ruby, Dart, and Elixir projects. Moreover, it has the ability to update Terraform, Docker, and submodules of git.

One may activate it by browsing its GitHub repository and configuring dependabot security updates under the security section. [ghD]

## 3.2 Snyk

[ITW21] Snyk can be run either via the command line or in a managed webpage console. Further, it has integration into several IDEs. Snyk further has a tool that offers to scan project dependencies as a bot like service. It does this by scanning dependency files and comparing them with an internal vulnerability database. [ITW21] Snyk can be run either via the command line or in a managed webpage console. Further, it has integration in several IDEs [snyc] and can be integrated well into version control systems as well as CI/CD tools [snyd]. Furthermore, Snyk offers container security scanning and security checks on infrastructure as code configurations. As a result, Snyk tries to offer a whole suite of security products and services around development from code to production deployment.

Figure 2 shows a demonstrative workflow with Snyk using the whole suite of products. Especially, it is shown the snyk bot analysing the Git repository and the integration into the CI/CD pipeline. The bot further monitors production environments.

## 3.3 Comparison

After describing the tools isolated, this section will have a look at the differences and similarities between these two tools.

Dependabot is deeply integrated into GitHub and its repositories. [ghD] In comparison, Snyk provides a variety of services as a platform. Starting from integrations in IDEs to CI/CD tools and analysis of docker containers running on cloud instances. Hence, it seems more service is driven for organizations wanting full-fledged security software rather than an open-source tool making sure that its dependencies are non-vulnerable.

As Snyk can be seen as a service offered to organizations as a full-fledged service provider, it has pricing. It offers three different plans with a limited free plan. [snye] Dependabot on the other hand is included in the free tier of GitHub and can therefore be considered a free service. [ghP]

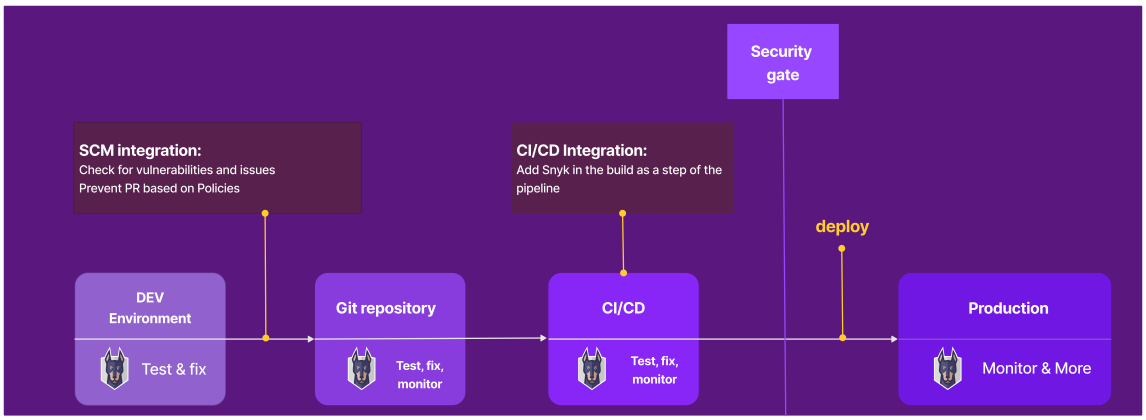


Figure 2: Snyk Workflow inspired by [syn]

Further important for security scanning is the analysis of the whole dependency tree. When dependencies are declared in a separate dependency file, these dependencies are classified as direct dependencies. Usually, such direct dependencies rely on other dependencies as well which are then called transitive dependencies. The dependencies can now be stored in form of a tree. As one has seen in the critical log4j vulnerability, a lot of dependencies used this dependency transitively. Snyk analyses the whole dependency tree and matches it against its own vulnerability database whereas dependabot does this just in case of the existence of lock files, like package-lock.json in the case of npm. [ITW21] [snyb]

# 4 Benefits of adoption

Of course, software bots have a reason to exist and mostly they take over tasks humans would do otherwise. The benefits of software bots may be split up into three parts (automation, information integration, and improvements beyond humans) [ENL20].

In the case of automation, humans may automate time-consuming and repeatable tasks which may be tedious, too. This may be the manual check of security vulnerabilities in current project dependencies. This process is repetitive and can be automated. On top, a machine is usually faster than a human for this task. [ENL20]

Another important step is information gathering from various sources. The benefits of this part lie in multiple areas. First, having all information from various systems in one place helps to get a clear overview. In case of an incident, a security advisor may see all information behind several firewalls and access rules suited to what is needed in one place. Further, it allows machines to combine data points that are not connected by humans themselves because the amount of points to connect has risen. [ENL20]

Another improvement is the constant working hours of bots as well as the reduction of human errors. This may lead to higher-quality in results and improvements in the workflow. [ENL20] An example would be anomaly detection in software systems.

# 5 Challenges adopting to Software Bots

This section will focus on challenges when adopting software bots for organizations. Each of the challenges will be mapped in the next section to dependabot and snyk and how these tools deal with these challenges.

[ENL20] point out several challenges when using software bots in an organization. One of these challenges is noise which is defined as an overload of information and messages by the bot. This can be expressed when bots post too much information under GitHub pull requests or send more Slack messages than humans would do. Further to noise, one may define interruption when bots ask

developers for more information even though they are not ready for providing more feedback which results in an interruption in their workflow as the bot is requesting more input too early.

Another important aspect of bots is trust. Bots performing more complex tasks cause more frequent failed builds and deployments. [ENL20] A solution to this problem may be appropriate test infrastructure as well as limiting the usage of bots to behaviors where the bot can be sure it does the right actions.

Further, the usability of bots needs to be taken into consideration. Users would like to communicate with bots in a natural language way. On the other hand, this opens up a range of interpretations causing the bot not to act as someone wants it to. [ENL20]

# 6 Setup with dependabot or snyk

In this section, the focus is set on how to set up dependabot and snyk to leverage the above benefits and minimize the mentioned challenges.

## 6.1 Automation of tasks and collecting information

As we have seen in section 4, one benefit of using bots is the automation of tasks. One may even leverage their usage further by combining them with information gathering. For dependabot, the main task is keeping track of all project dependencies by showing whether these have vulnerabilities. It further allows to automatic create pull requests to update dependencies (see section 4). Here, dependabot analyses frequently dependencies and checks for vulnerabilities which is a tedious task. Further, it is deterministic and may be automated rather easily. In this setup, dependabot combines automation with information collection. Mapping this feature now to the challenges identified in section 5, one may investigate on how to reduce the number of pull requests and alarms dependabot creates to reduce information overflow. Organisaitons may set the following settings from the dependabot documentation [depa], to increase their productivity in bot usage as seen in listing 6.1.

- schedule.interval: Is defined as the interval dependabot checks for updates. When this property is set to daily, an organization may adapt quickly to new releases but will cause a lot of alarms for developers.
- ignore: Tells dependabot which dependencies to ignore. If an organization has specific reasons to ignore warnings for certain dependencies, this may reduce the number of duplicate alarms.

Listing 1: Example dependabot.yml

```
version: 2
updates:

- package-ecosystem: "github-actions"
  directory: "/"
  schedule:
    # Check for updates to GitHub Actions once week
    interval: "weekly"
  ignore:
    - dependency-name: "express"
      # For Express, ignore all updates for version 4 and 5
      versions: ["4.x", "5.x"]
```

Looking now at the same case by using snyk, snyk offers security checks on connected repositories, too. [snya] In contrast to dependabot, Snyk offers various ways to integrate into the development cycle to automate the task of analyzing dependencies with reducing the information overflow for the developers. Snyk offers the following options:

- Static code analysis: One-time scan which can be started as needed
- Warnings in the IDE: Directly integrated while coding. Hence, just the developer gets the information during development

- Code check-in pull request: Kickoff security analysis when the developer wants to push data into certain environments.

Snyk is here capable of combining its functions as it offers a broader range of functions rather than the isolated capabilities of dependabot. Organisations may try to combine these tools to increase their level of automation and tackle the challenges pointed out in section 5. These include the trust in certain tasks done by snyk as well as using the tools to distract developers the least.

### 6.2 Communication style

[ENL20] define communication with bots as a challenge. As dependabot and snyk automate certain workflows and even act upon their findings, they face the same issues. This may be combined with another issue of information overflow by for example writing more Slack messages than a human would do.

In our tests, dependabot opened for every vulnerability it found a new pull request with a fix for one dependency. As a result, the tested repository had suddenly ten open pull requests which caused a flow of emails triggered because of the changes. In case of further integrations to messaging tools like Slack, this gets even more distributed. It has to be mentioned that one may change the notification settings of Github. To reduce this information overflow, organizations may configure dependabot notification settings according to their needs so developers do not start ignoring its messages because of the overflow. https://docs.github.com/en/code-security/dependabot/dependabot-alerts/configuring-notifications-for-dependabot-alerts

Snyk has its own integration with messaging tools like Slack or ticket solutions like Jira. It shows identified vulnerabilities on one page and gives an overview of its findings. The optional Slack integration however sends messages for each vulnerability.

Both do not offer to be configured via natural language. Hence, snyk is configurable via their admin interface and does not necessarily require coding. Dependabot however requires a committed configuration file. Even though certain parts are pre-defined, the configuration has to be done via code.

## 7 Conclusion

In this essay, we have seen why software bots gained importance in software development cycles. Further, we introduced two tools on the market, dependabot, and snyk, and compared them with each other. With this background knowledge in mind, the benefits and challenges of adopting to software bots are identified. The results are mapped against snyk and dependabot, highlighting their advantages and how to configure them to mitigate the challenges and leverage their advantages.

We have seen that both dependabot and snyk have their right to exist as bots to increase software development productivity. We have identified configurations for both tools to mitigate challenges organisations phase when using bots and identified advantages for each bot.

Both bots show which benefits organisations may gain when using bots in their DevOps processes.

## References


[ACSM21] Mahmoud Alfadel, Diego Elias Costa, Emad Shihab, and Mouafak Mkhallalati. On the use of dependabot security pull requests. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pages 254–265, 2021.

[CH22] F. R. Cogo and A. E. Hassan. Understanding the customization of dependency bots: The case of dependabot. *IEEE Software*, 39(05):44–49, sep 2022.

[depa] Configuration options for the dependabot.

[depb] Dependabot github.

[ENL20] Linda Erlenhov, Francisco Gomes de Oliveira Neto, and Philipp Leitner. An empirical study of bots in software development: Characteristics and challenges from a practitioner's perspective. New York, NY, USA, 2020. Association for Computing Machinery.

[ghD] Configuring dependabot security updates.

[ghP] Github pricing.

[ITW21] Nasif Imtiaz, Seaver Thorn, and Laurie Williams. A comparative study of vulnerability reporting by software composition analysis tools. Association for Computing Machinery, 2021.

[LSZ18] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky. Software bots. *IEEE Software*, 35(1):18–23, 2018.

[snya] Exploring and working with the snyk code results.

[snyb] Snyk direct and indirect dependencies.

[snyc] Snyk ide-tools.

[snyd] Snyk.io ci-cd integrations.

[snye] Snyk.io plans.

[syn] Snyk git-repository-and-ci-cd-integrations-comparisons.