Improving monitoring automation with Icinga

Pau Feixa Morancho
paufm@kth.se

May 4, 2023

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

Contents

1 Introduction

The scope of many projects grows rather fast so, expecting a developer to monitor the whole infrastructure is not feasible. Citing Theo Schlossnagle[1]: "Monitoring is about observing and determining the behavior of systems, often trying to determine correctness. Its purpose is to answer the question. Are my systems doing what they are supposed to?". That is why monitoring is a very important term when talking about DevOps, because every developer wants to answer that question. So, to do that, many applications fully dedicated to monitoring come into play.

In this paper, one of those tools, Icinga, will be discussed, with the objective of pointing out why one of the key aspects of a project is monitoring everything, as well as how this can be done automatically.

2 Monitoring

2.1 Definition and usages

According to the IEEE Computer Society [2], in DevOps, monitoring is one of the stages in the DevOps chain (see Figure 1), and it is referred to measuring the performance and health of systems and applications in order to help developers identify and solve software problems. If those potential problems are detected in the first stages, the benefits are even higher, as it can reduce the effect of degradation of service.

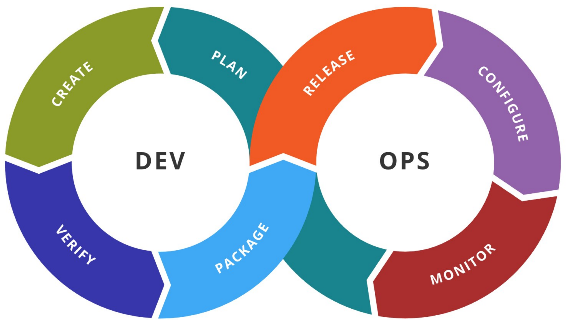

Figure 1: Stages in the DevOps tool chain.

At this point, another concept emerges, which is continuous monitoring [3][4]. The idea behind this new term is the same as monitoring, which is that of tracking the systems to see how they behave, but it goes one step further. This new term embraces the time aspect, so checks are performed regularly, and typically are automated, involving software which helps in carrying out this task. As it has been said, the main goal is to identify threats early so that they do not become an issue.

Once the definition of monitoring is known, it is appropriate to list some of the different usages that appear [5]:

- Testing
- Problem and anomaly detection
- Performance analysis and optimization
- Debugging
- Security

2.2 Golden Signals

The term Golden Signals was introduced by Google [6] to refer to the essential metrics that are needed to measure in applications. These signals are metrics that offer a view of a service from a user perspective, so that it is possible to detect some issues affecting its behavior. These Golden Signals are defined by 4 metrics, which will be now briefly explained:

1. Latency: It is the time it takes to service a request.

2. Traffic: It is a measure of how much demand is being placed on the system.
3. Errors: It is the rate of requests that fail.
4. Saturation: It is a measure of how full the service is.

2.3 What to expect in the near future?

Recent studies done on the market have shown that it has been a consistent growth in the open-source monitor market and it will continue like that for years to come (see Figure 2). More specifically, the market is expected to grow up to \$50 billion in 2027 from the \$27.6 billion in 2020.

It goes without saying that this aspect just proves the point of how important monitoring is and how important it will continue to be, as it saves time, which can be translated to money savings for the companies.

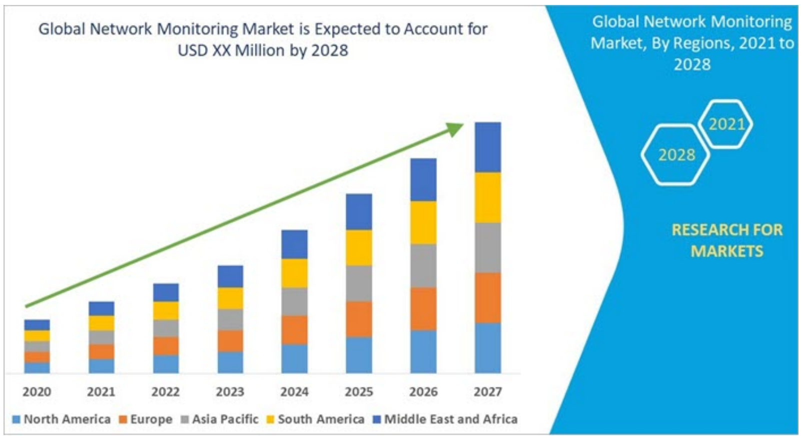

Figure 2: Global Network Monitoring Market Tendency from Data Bridge Market Research.[7]

3 Icinga: Monitor your entire infrastructure

3.1 What is it?

Icinga [8] is an open-source monitoring system that checks the availability of network resources. It also notifies users when any problem takes place, reporting all the data that has been measured. The good aspect of this application in particular is that the monitoring automation can be improved, automating checks whenever the infrastructure changes. By checking the whole infrastructure in a single place, full visibility can be gained, so errors can be found quickly. In other words, Icinga can monitor large, complex environments across multiple locations. Just out of curiosity, the word ”icinga” can be translated from Zulu to English as ”searching”, which is one of the main things that this tool does.

3.2 The 6-in-1 Stack

The Icinga stack spans six areas to cover all aspects of monitoring your entire infrastructure. These different areas can be seen in Figure 3. Each color in the stack is associated with one area, whose correspondences are:

1. Orange: Infrastructure monitoring
2. Blue: Cloud monitoring
3. Violet: Metrics & Logs
4. Yellow: Analytics
5. Red: Notifications
6. Green: Monitoring automation

Figure 3: The 6-in-1 Stack defined by Icinga.

3.2.1 Infrastructure monitoring

It is needless to say that a lot of data is produced by the infrastructure informing about its health and performance. Using this data is how it can be known which is the state of the different parts of the project so that they can be running. This area of the stack includes monitoring in:

- Server monitoring: Avoid interruptions and increase the availability by monitoring the server infrastructure.
- Application monitoring: Apps should run smoothly and respond fast.
- Network Monitoring: Reduce the network downtime and find problems affecting the bandwidth.

3.2.2 Cloud monitoring

By using the Icinga Stack, it is possible to monitor everything, even the clouds, in a single monitoring system, which allows to have full visibility, as it was explained when first introducing Icinga.

The previous item in the stack, the infrastructure monitoring takes care of server monitoring among others, but this one is able to monitor servers in the cloud, like AWS or Azure, or even multiple cloud providers.

3.2.3 Metrics & Logs

With all the data and metrics that will be present, it has to be ensured that they can be collected, stored and visualized. This topic can be associated with observability. Even though sometimes monitoring and observability are mistakenly exchanged by one another, they are not the same concept, as the latter is a broader concept. However, this is not the main topic of this paper, so just a few more details on observability will be given.

In the Metrics & Logs area, it is allowed to monitor metrics and logs, which show the full picture of everything that is happening. Having all the monitoring data from the diverse parts of the infrastructure in a single interface, along with other aspects, is what defines observability. Another advantage of taking all the metrics and logs and storing it in the same place, is that there is the possibility of creating alarms and notifications depending on the results obtained, which avoids repetitive patterns and some frequent trends.

3.2.4 Analytics

This element from the 6-in-1 Stack is just a continuation from the previous one, as it consists on analysing the data collected from the whole infrastructure. With the objective of simplifying the analysis of all the monitoring data that will be stored, it is possible to perform detailed and customizable searches so that it is easier to identify problems, as well as being able to filter that data in order to narrow things. Not only that, but also the possibility of creating several dashboards for different use cases helps separating the analysis by teams in the same project.

3.2.5 Notifications

It goes without saying that a developer will not be constantly checking and monitoring everything, so there has to be a way of receiving fast notifications when a problem pops up. Icinga notifies the developer through any desired channel, including email, which helps in reducing the response time.

In larger projects, it should not be desired that a single person receives all the notifications, so this tool helps in assigning different people to different parts of the project, as well as choosing the channel in which these notifications will be received. This part has to be well automated, because Icinga has to inform of any infrastructure change by itself.

3.2.6 Monitoring automation

Finally, this last area of the stack is one of the most important parts, and it is one of the main topics in this paper. So, why is it so relevant?

In large projects, the number of objects that have to be checked is considerable, which require more effort. Icinga can monitor thousands of servers and services, and automate all areas of it. This helps in managing everything in an effortless and fastest way, reducing also potential human errors. Icinga is such a powerful tool that it is adaptable, so automating the monitoring also means automating the generation of monitoring checks and updating those checks. For instance, Icinga will notice when parts of the infrastructure evolve (removing a server, for example) and it will adapt the monitoring configuration (removing the monitoring checks, in this case).

3.3 How does it work?

In Figure 4, there is a representation of a generalization of an Icinga scenario [9]. In that diagram, it can be easily seen that each node has one role, being the possible roles Master, Satellite or Agent. In this tree-scenario, the Master is the one at the top, and its purpose is to collect all the metrics from the nodes below, usually from the mentioned Satellites.

These nodes, the Satellites, are used when there are many servers to monitor, each in a different private subnet, so that all of them can be accessed. Lastly, there are the Agents, which are the machines that are being monitored and are executing monitoring scripts.

It is safe and sound to confirm that this scenario can be quite unreliable, as if the Master fails, everything else is practically useless because, although data is still going to be taken, it will not be collected in a single interface, which is the whole point of this scenario. For that, one solution can be creating two instances registered as Master, so that one can substitute the other in case of failure. This thought can be extended for Satellites too, so that important nodes have redundancy, which implies lower downtime and more availability.

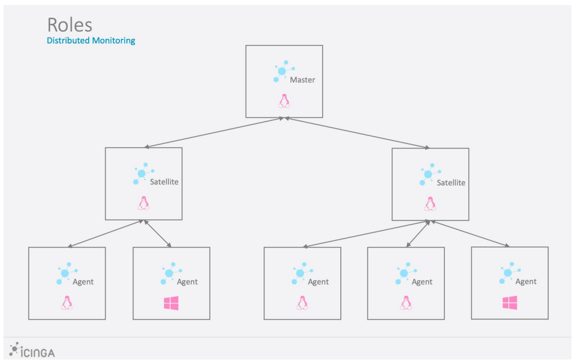

Figure 4: Icinga scenario. Source: ITGIX.

3.4 Possible scenarios

In the previous section, a general scenario has been shown, so that the different components that take part could be identified and briefly explained. Now, three other possible scenarios [10] will be explained, with the objective of demonstrating the flexibility of Icinga and the range of possibilities it gives.

3.4.1 Independent Icinga Instances

As its name indicates, in this scenario many Icinga instances will be running, which are independent from each other. The problem with this kind of scenario is that, as instances do not exchange information, some checks can be redundant, which could trigger the alarms more than once. Moreover, as it has already been explained, one of the key aspects of Icinga is that it is possible to have all the data collected in a single interface, so running independent instances may not be the best solution.

3.4.2 Central Instance with Distributed Instances

This second scenario is more similar to the general case, as there are several instances monitoring specific entities, but all reporting to a central instance, which would be the Master. The Master would be collecting data for the user, so then notifications can be sent and all the stuff that was explained in the 6-in-1 Stack. This solution is not perfect either, as the principal problem appears when there are a lot of entities to monitor, because there is just one central point that has to be aware of all the entities. Nevertheless, in most of the cases it would be a better scenario than the previous one with independent instances.

3.4.3 Central Icinga Instance with Distributed Workers

As in the previous case, there is a central Icinga instance that has to collect the results, but the main difference is that, in this scenario, the so-called ”workers” would be performing the checks. In any case, the central instance is still a point of failure, because if it fails, there is not any other instance in charge of collecting the data. A possible solution, as it has been previously mentioned in Section 3.3, setting up a cluster of identical machines running in active-passive mode would solve that, as if one instance crashes, the other one can take over.

4 Conclusions and Reflections

To sum up, it is important to highlight that monitoring is really important from a DevOps point of view because it gives full visibility of everything that is happening, which allows offering better services or remediate issues in a faster way. Moreover, the adaptability that tools like Icinga have make it easier for developers to focus on tasks that other tools cannot do for them. Furthermore, a company that is offering some kind of service, would surely like to get notified as soon as a problem appears, and Icinga can take care of that.

This paper is based on explaining the performance capabilities that Icinga offers, focusing on the monitoring aspect. As it has been stated in several sections, Icinga makes it easy for developers to focus on developing. By using monitoring automation and collecting the data in a single interface, errors can be found easier, patterns can be detected sooner, etc., all leading to an improvement in the resistance against service degradation. For this reason, the idea that should be extracted from this paper is that it has to be highly encouraged to start using this kind of tools from early stages of the project, so that errors that can be solved do not generate severe problems in the future.

References

[1] Theo Schlossnagle. Monitoring in a devops world. page 2, 2017.

[2] Tanhaz Kamaly (IEEE Computer Security). Observability vs monitoring: The key differences devops should know. URL: https://www.computer.org/publications/tech-news/trends/observability-vs-monitoring-the-key-differences-devops-should-know, 2022. Last accessed 3 May 2023.

[3] Arfan Sharif. What is devops monitoring? URL: https://www.crowdstrike.com/cybersecurity-101/observability/devops-monitoring/, 2023. Last accessed 3 May 2023.

[4] Arfan Sharif. What is continuous monitoring? URL: https://www.crowdstrike.com/cybersecurity-101/observability/continuous-monitoring/, 2022. Last accessed 3 May 2023.

[5] Martin Monperrus. Course notes on software monitoring & tracing. *KTH course for Automated Testing and DevOps*, page 3, Created April 2019, last updated April 6, 2023.

[6] Rob Ewaschuk and Betsy Beyer. Monitoring distributed systems. URL: https://sre.google/sre-book/monitoring-distributed-systems/, 2016. Last accessed 3 May 2023.

[7] Data Bridge Market Research. Global network monitoring market – industry trends and forecast to 2029. URL: https://www.databridgemarketresearch.com/reports/global-network-monitoring-market, 2022. Last accessed 3 May 2023.

[8] Icinga. Official webpage. URL: https://icinga.com/, 2023. Last accessed 4 May 2023.

[9] Ralitsa Dimitrova. Introduction to icinga monitoring tool. URL: https://itgix.com/blog/icinga-monitoring-tool-tutorial/, 2022. Last accessed 4 May 2023.

[10] C. Haen, E. Bonaccorsi and N. Neufeld (CERN). Distributed monitoring system based on icinga. pages 1149–1150, 2011.