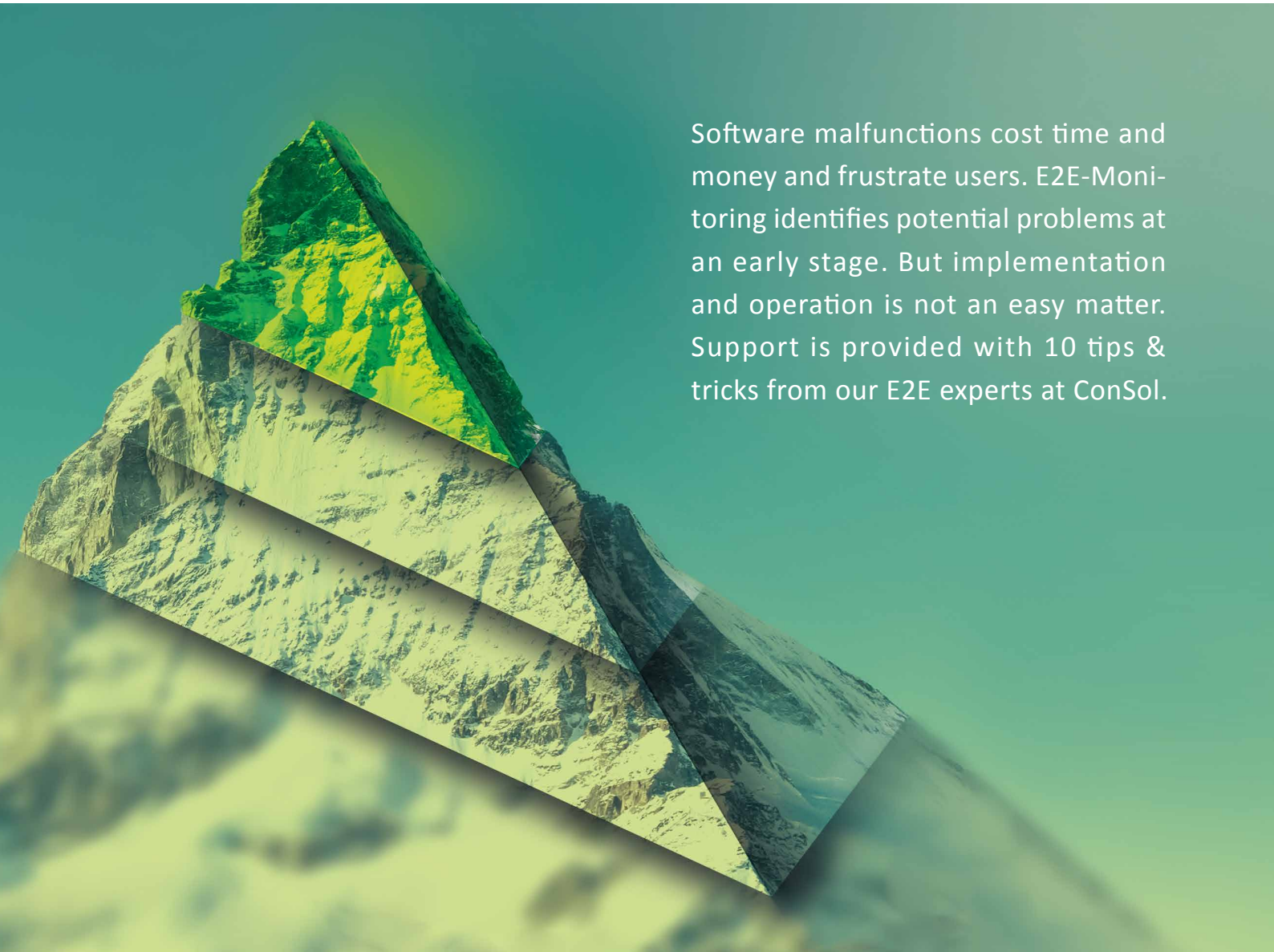
sakuli

10 Step Guide to End-2-End-Monitoring



Software malfunctions cost time and money and frustrate users. E2E-Monitoring identifies potential problems at an early stage. But implementation and operation is not an easy matter. Support is provided with 10 tips & tricks from our E2E experts at ConSol.

Monitoring is a powerful tool to track system health. Observing data like the efficiency of a database, the CPU load or the general availability of servers is a task “traditional monitoring” is doing perfectly fine. But is this enough to ensure your complex system architecture is performing at its best?

Everybody knows this situation: trying to buy something awesome online but being stuck in endless page-load loops, error messages and buttons that seem to not do anything. Nowadays customers can just switch to another online-supplier, leaving the shop-owner without a turnover and a bad reputation.

End-2-End Monitoring helps to ensure all functionality, track the performance, and thereby the user experience of your system, enabling operations to act fast in case of an issue, even before the user will notice anything malicious. But setting up real End-2-End Monitoring can be complex and overwhelming. Choosing the right technologies, identifying your business-critical processes, configuring proper threshold and using smart alerts are only some of the tasks to consider.

Application Performance Management vs. End-2-End Monitoring

If you are looking for an End-2-End monitoring solution, you are probably going to come across application performance management (APM).

With the rise of cloud-based microservice applications, many companies are facing the same issues. Tracking the performance and availability of the application has become increasingly difficult. APM is about discovering anomalies in the application and its components. Tracking the hardware, virtual machines, containers or infrastructures such as databases, caches or the network to find errors and bottlenecks and proposing solutions is its key task. This approach only covers system analysis from a technical perspective - without integration and communication between all those services. In contrast, the purpose of End-2-End monitoring is about the availability and performance of the application facing the end-user - running checks on the application (in the best-case scenario, simulating a real user with tools like https://sakuli.io) to mimic real user experience and observing the performance of an application, implicitly all its subsystems, at all times.

Keep the Checks Small

Compared to technical monitoring of, for example, VM CPU and memory usage, End-2-End checks are more complex: They can be resource-intensive, high maintenance and a dedicated test machine, test farm or container setup is required to execute End-2-End monitoring cases regularly. Therefore, it is important to prioritize important business workflows and focus on the essential and business-critical paths of your application. As a result, End-2-End monitoring ensures incredibly fast reaction times in case “something” is wrong with the system, thus ensuring a stable revenue-stream and conversion-rate.

Be Where the Party is

It is best to run End-2-End monitoring on production systems to measure user experience. Of course, running on another stages is possible, but these rarely have the same setup, nor could a stage environment be as important for revenue generation as production. Therefore, the checks should run on production to derive maximum utility from the results.

Choose your Thresholds Wisely

An important task is to choose the right thresholds. If the performance of your application drops under a certain threshold, your throughput and conversion-rate will be dramatically reduced as the system feels “too slow” for the expectations of a user nowadays. There are a few studies on this topic, according to one of them, “the presence of feedback prolongs Web users’ tolerable waiting time and the tolerable waiting time for information retrieval is approximately 2 seconds.”(Nah 1) Therefore, it is important to specify the thresholds in consultation with the business managers to keep users happy and the business up and running 24/7/365.

Don’t Panic!

Even if an alert has been sent with a warning, don’t panic. With correctly set thresholds, alarms will go off early enough to be able to react. In complex systems, a peak of concurrent users, network traffic or database and messaging reconnections might cause fluctuations in the systems’ performance without rising to a serious issue. Therefore, a warning might not require instant reaction, but observation whether the system stabilizes itself or requires action. The following aspects are worth keeping in mind when rating a warning:

- How critical is the step that failed regarding business?
- Which functionality was checked, and how many systems were involved?
- How often did the check fail in the last hours?
- Is there any planned maintenance or release going on?

Location Matters

If your application is used by an international audience, set up End-2-End monitoring in each region/market your application is targeted on. It might sound costly but will generate insight into why and where multi-region routing could be necessary. Don’t underestimate latency, routing efficiency, processing overhead, congestion etc. to affect the user experience negatively.

Read vs. Write

There is not much of a difference between a monitoring system having read or write access to an application. As soon as there is an interaction, some data will be stored, hit-rates and page-loads will be manipulated, etc. It is therefore very important to ensure, that check-data does not impair your daily business.

As soon as data is made persistent within a business process (e.g. a checkout process within the web-shop), the following actions should be considered. To avoid interference with daily business, business intelligence or other analytical systems, close coordination with all involved parties is necessary. Test-transactions need to be excluded by accounting and third-party systems by using identifiable test-data and dummy credentials. Before writing data automatically into a productive system, it is important to think about the possible consequences.

Test Data on Production

Test management meets monitoring – namely test data management. Test data and test accounts in production should be named unique to separate them from customer data. In addition, this data should be managed and documented transparently. The responsible team maintaining the data and its documentation should be the same setting up the checks for the End-2-End monitoring.

Be Compliant

Another point to consider is compliance. Imagine you create a test account on production. Over time, many people might access the test account for debugging purposes or check development. When someone leaves the company, they can still log in to the test account if the password was not changed or completely disabled. Changing the passwords, altering test cases and informing people are few of the necessary tasks, which cause a lot of effort. Therefore, do not hand out credentials for the test users. Modern End-2-End testing and monitoring frameworks come with security mechanisms – like secrets in Sakuli – where credentials are fully encrypted. If you do not hand out technical user credentials, you will not have to change them later on. In the case of debugging and check development, create personalized accounts for everybody working with the monitoring suite. Those accounts can easily be disabled and exchanged with the credentials of the technical user when checks go live.

Look at the Bigger Picture

One red test between thousands is not an indicator of your application failing. It might have been a hiccup at your ISP, infrastructure, internal services, or network. So rather than inspecting results individually, search for patterns from the overall results. If the duration of the login process increased dramatically after a new release, it might be a good idea to look into that. Therefore, pay attention to certain events like future release dates and major changes to the overall system.

Avoid Conditional Statements in Checks

Conditional statements are key elements in every programming language. However, using these in monitoring checks might cause inconsistent results. Monitoring is designed to collect metrics, to track certain performance, to load indicators and to display the collected data graphically to highlight issues of the system and its infrastructure. However, if checks contain conditional steps, the collected metrics of the application are incommensurable as the performed check changes based on the implemented condition. This makes it impossible to compare information about the performance and availability of the application reliably. Therefore, conditional statements should be avoided in End-2-End monitoring checks.

Conclusion

End-2-End Monitoring can be a challenge! Especially subtleties, like choosing the right threshold or what to cover in a check, are not obvious at first glance. Experience with End-2-End monitoring technologies is essential to set it up efficiently. Our tips & tricks provide an overview of the most important topics to consider but there is much more to be aware of as soon as you start implementing the mentioned concepts.

For more information about Testautomation, Monitoring and RPA, visit sakuli.io, follow us on Twitter @sakuli_e2e or visit consol.com

ConSol
Consulting & Solutions Software GmbH

St.-Cajetan-Straße 43
D-81669 München
Tel.: +49-89-45841-100
info@consol.de
www.consol.de
Folgen Sie uns auf Twitter: @consol_de

ConSol
Austria Software GmbH

Mooslackengasse 17
A-1190 Wien, Österreich
Tel.: +43-1-9971392
info-austria@consol.com
www.consol-software.at
Folgen Sie uns auf Twitter: @consol_at