ROOKOUT

5 ways to slash debugging time throughout your code development lifecycle

It's no secret that powerful new tools and methods make it faster and easier than ever before to develop and scale applications, but they also significantly reduce the visibility developers and devops need to understand how services behave and to debug them when issues arise.

In this e-book, we describe five situations in which engineers using these new technologies and methods face tough and time-consuming challenges when developing, debugging or refactoring apps, and show how Rookout empowers them to get the tasks done and quickly move on.

1. Debugging across the accelerated dev lifecycle
2. Debugging cloud-native applications
3. Breaking down a monolith
4. Debugging other people's code
5. Debugging in production

1.

Debugging across the accelerated dev lifecycle

THE CHALLENGE

Software development is undergoing a remarkable revolution. CI/CD and DevOps allow faster work and larger scale changes with less fear of failure -- or at least with confidence to fail fast and recover quickly.

Along with great benefits, these new methods bring with them significant challenges: Developers must adopt new tools and procedures, and keep pace with the relentless cycle of code testing, automating deployment flows, and assessing how code plays in production, then refining and starting again.

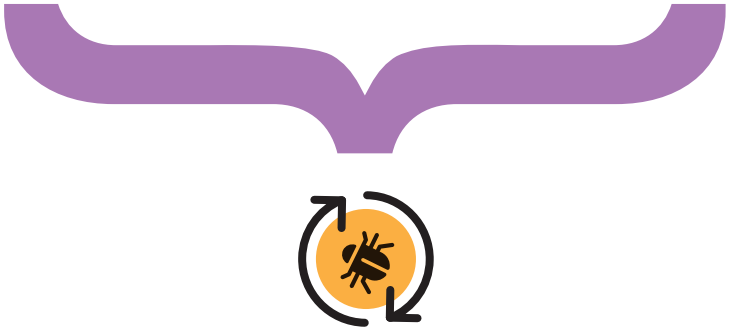
While debugging has always been a key part of development, the accelerated, automated dev cycle makes it more challenging in a number of ways:

- Developers must master and apply three distinct sets of debugging tools and methods -- for local debugging, for remote tests or staging environments, and for troubleshooting production issues.
- Every code change in the debugging flow, even small changes like adding a code line or a monitor, requires time-consuming application rebuilding and redeployment, as well as in some cases expensive hardware costs.
- When debugging breaks the development flow it breaks developers' focus as well, resulting in frustration and efficiency lost to context switching and delays.

HOW ROOKOUT HELPS

Rookout DevOps-ready debugging works seamlessly with all phases of the accelerated dev cycle - locally, in remote staging environments, and in production.

Rookout enables data collection for debugging without breaking to re-build and re-deploy. It accelerates the debugging process substantially and allows developers to remain focused, so they can keep pace with the automated build-deploy-test flow and overall efficiency increases.



2.

Debugging cloud-native applications

THE CHALLENGE

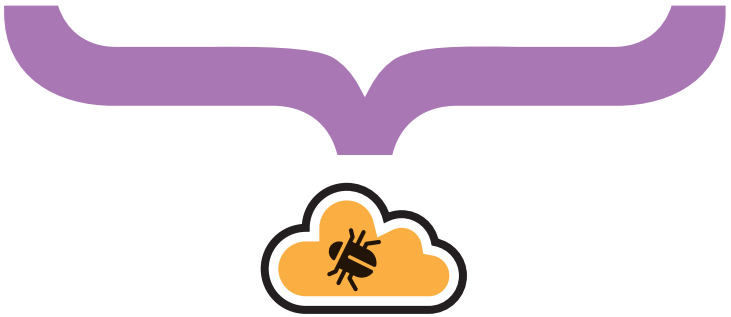
Many organizations have already started to transition some or all of their applications to the cloud. By replacing local, physical hardware with public, private or hybrid cloud platforms, they aim to enable greater scalability and enhanced flexibility, while reducing hardware costs. The extreme, cutting-edge version of this approach is transitioning to serverless deployment, under which code and business logic are completely decoupled from hardware.

While serverless deployment increases agility, simplifies operations and reduces capital expense, it also makes applications much harder to troubleshoot. Traditional troubleshooting entails installing an agent, fetching logs from a specific host, and changing configurations to fix the issue (or at least try to). Leveraging agents for cloud applications is a challenge, and changing configurations is even more so. For serverless applications, these traditional troubleshooting tools are completely irrelevant.

HOW ROOKOUT HELPS

Rookout cross-environment debugging operates at the code level, and requires no agent installation, log fetching or configuration changes. 'Rooks' are deployed within the code itself, and enable rules to be added and data fetched with a click of a button.

If your team is test driving new serverless capabilities, or if you are working on moving core business flows to cloud deployments, you need Rookout so your teams can continue debugging its apps as if they were running on the server next door.



3.

Breaking down a monolith

THE CHALLENGE

Applications are often built initially as monoliths, structured as a single process to be deployed on a single server. With time and growth, however, the monolith becomes a bottleneck that makes development inefficient. With each code change, even for something as minor as adding a log snippet to get data for debugging, the complete application must be re-built and re-deployed, resulting in significant waiting time and real hardware cost.

To streamline development, monolithic apps may be divided into smaller microservices, each of which performs a well-defined subset of the functions previously performed by the monolith. These microservices are easier to deploy and may be set or updated independently.

Refactoring a monolith into microservices is a large, risky and often overwhelming project that can span months and demand years of engineer and developer time.
The process requires engineers to

- Map existing functionality and break it into logical components.
- Build a new service for each component, and develop new ways to deploy it.
- Develop new data flows, through which the various new services communicate with each other.

Throughout all these changes, engineers must be sure not to break established business flows performed by the monolith.

As new microservices are broken out, new challenges arise:

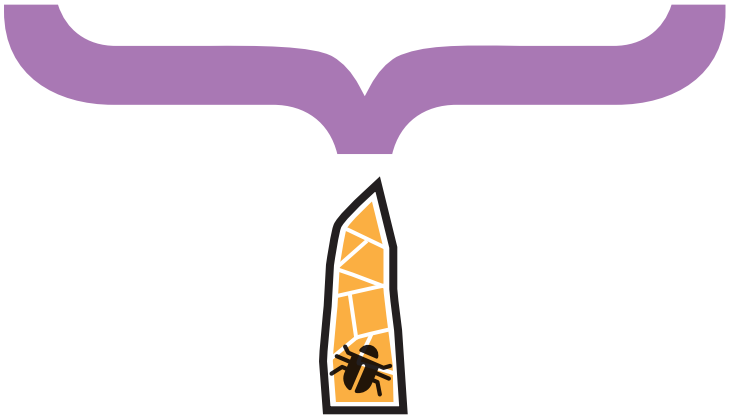
- As devs work on and debug microservices simultaneously, in parallel, they need visibility how other developer activity might impact their functions.
- Ensuring smooth communication and handshakes between microservices that are deployed on different servers, written in different languages, and/or use different databases.
- Tracing problematic requests as they make their way from one service to another.

HOW ROOKOUT HELPS

Rookout zero-touch debugging reduces refactoring time and complexity. It provides engineers with the visibility needed to learn and understand monoliths in production, without changing code and without re-deploying, by simply inserting code snippets. As microservices are created, Rookout enables seamless debugging of the modular system, just as it does for single applications.

Rookout allows rapid debugging of the newly created services wherever and however they're deployed and assists developers in tracing problems across each other's code.

At every stage of the monolith-to-microservice transition, Rookout helps make the formidable task more manageable and efficient, and a good deal less risky.



4.

Debugging other people's code

THE CHALLENGE

Debugging your own code is hard. But it doesn't hold a candle to the challenges inherent in debugging code that other developers wrote. Perhaps, in the best case, the author sits in the next seat, and the code you're debugging is still fresh in his mind, and you can simply ask him about any issues that stump you. More likely he's from another team, sits on another floor, or works in another country. Often it's code written a long time ago, by a developer who no longer works for the company.

Still more difficult is debugging open source projects or third-party apps that you're integrating in your code. There's no identifiable developer to ask and posting on forums is perilously close to a song and a prayer. When the "third party" bugs you are tackling are in, for example, a Microsoft app, it's just you and your tools, working alone.

As a general rule, debugging someone else's code is like locating your car keys in some stranger's house. In the dark. While wearing mittens. You simply cannot think your way into an unknown stranger's thought patterns with enough precision to find the key data you're seeking. Without knowing what their objects represent or the logic they used, debugging will cost even the savviest, smartest (and luckiest) dev dearly in time and effort that could be better spent coding. And the only way to speed up the process is to shine a bright light onto as much of the data and processes as you possibly can.

HOW ROOKOUT HELPS

Rookout high-visibility debugging provides full observability into any app code, no matter who wrote it, when, and for what purpose. It makes debugging code written by your fellow developers, as well as third parties and open sources, easier, faster and most of all, more successful. And because two heads are better than one, Rookout enables collaborative debugging so two or more developers can tackle the same issue simultaneously.

5.

Debugging in production

THE CHALLENGE

Most teams simply write off debugging in production as impossible. At the very most, they troubleshoot problems as they occur by monitoring production issues using APM. By adding log snippets, they get insight into what's happened after the fact. Once an issue has been identified, however, teams invest significant time and effort in trying to reproduce it locally so they can find the root cause and actually debug.

However, issues can't always be reproduced locally. The environment and network configuration are different; application deployment is different; and user behavior and data are different.

To make matters worse, in many organizations, development teams can't access logs and monitors directly, but must request them from operations teams -- a process that can take days or even weeks in some cases. If a monitor or log is not already available, the operations team must likewise be called on to push the log or monitor snippet into production, and troubleshooting the issue is further delayed.

Under even the best circumstances, resolving production issues may drag on for weeks or months, delaying milestones and distracting developers from scheduled tasks.

HOW ROOKOUT HELPS

Rookout production debugging gives development teams instant visibility into the production environment, simply, without pushing changes or depending on operations teams. Instead of struggling to reproduce issues in a similar environment, Rookout allows developers to quickly and thoroughly investigate the issue where and under the exact conditions in which it occurred.

The result? Quicker time to resolution, a more efficient development team and a stable production environment.

ABOUT ROOKOUT

Rookout, the rapid debugging company, dramatically slashes debugging time and effort across the full product cycle, from dev through staging and production. It enables on-the-fly data visibility and collection without restarts, redeployments or added code, and with real-time delivery to APM, logging and, alerting and other platforms.

Rookout provides the essential live-code observability that developers and DevOps need, whether in monolith applications with long CI/CD cycles, complex microservice infrastructures or hybrid/serverless environments.

For more information, visit www.rookout.com