Observability

100-level live demo script

Introduction

Narration
In this demo, I'll show how IBM Instana helps quickly identify, debug, and resolve an incident in a microservices-based application.

To set the context, our application is called Stan's Robot Shop, and it is a modern, cloud native application with microservices leveraging various technologies such as Java, Python, and MySQL and deployed in containers on top of Kubernetes cluster.

Such applications create a serious challenge for managing application performance because components are dynamic and loosely coupled. They use different technologies, so they usually require broad knowledge and multiple tools to diagnose. Instana, with a single agent deployed per host, automates the discovery process.

Application components are discovered and observed as they are deployed. Over 200 technologies are supported with zero or minimal configuration, releasing you from installing and configuring multiple tools or plugins. The discovered components can be grouped into an application perspective, giving the application owner an easy overview of key metrics ("golden signals") like traffic, errors, and latency on a single pane of glass.

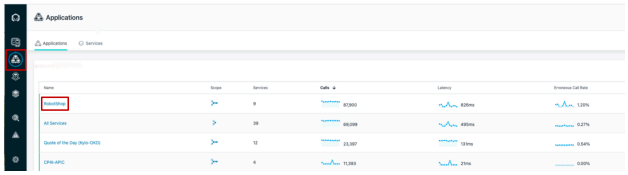
This is a visualization of all the dependencies within the robot shop application. Instana automatically discovered the relationships between the services and correlated them into this dynamic graph. We can see how requests are moving through the application in real time. Instana is able to do this because it tracks every request that flows through the application.

We can tell there are some problems with the application because several services are highlighted in yellow and red.

But you wouldn't normally be looking at the dashboard when something like this happens, so let me walk you through what it looks like from the SRE/IT operator's point of view when an incident occurs.

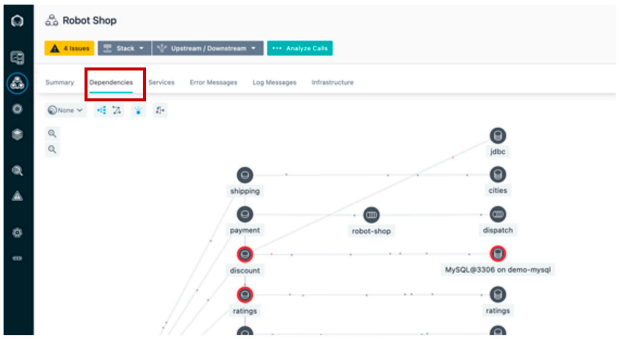
Action 0.1.1

- From the sidebar menu, click the **Applications** icon and choose **RobotShop**.



Action 0.1.2

- Click the **Dependencies** tab.



1 - Getting an incident alert

1.1 - Automatically assess events and alerts

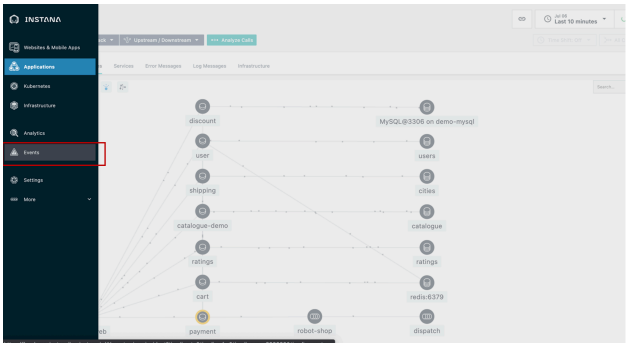
Narration
We've just gotten an alert from Instana that there has been a sudden increase in erroneous calls on our 'discount' service, which is part of the robot shop application.

Although I don't have it connected right now, the alert would show up via one of the configurable alert channels, like PagerDuty, Microsoft Teams, Slack, and many others (full list).

It's important to note here that you're not getting alerts for just anything. Behind the scenes, Instana is determining which events and issues are related, and it only sends alerts if a problem is likely to affect end users.

Let's go into the details for this incident.

Action 1.1.1
- Click the **Events** icon (triangle) on the sidebar menu.



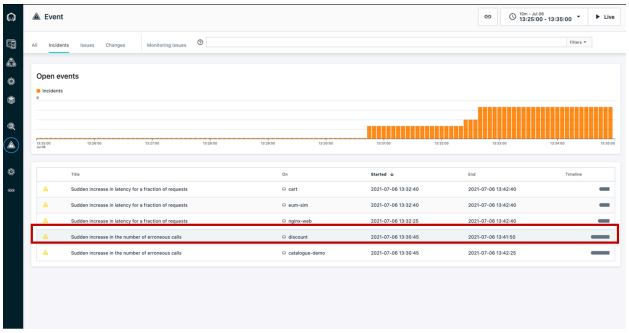
2 - Inspecting auto-correlated incident details

2.1 - Gather information from the incident detail page

Narration
Instana recognized that the sudden increase in the number of erroneous calls was something important to alert on, so we did not have to do any configuration or set thresholds in order to get this alert. We get key information right away when we come into this incident detail page. There's a timeline of the incident, the event that triggered Instana to create the incident, and all of the related events.

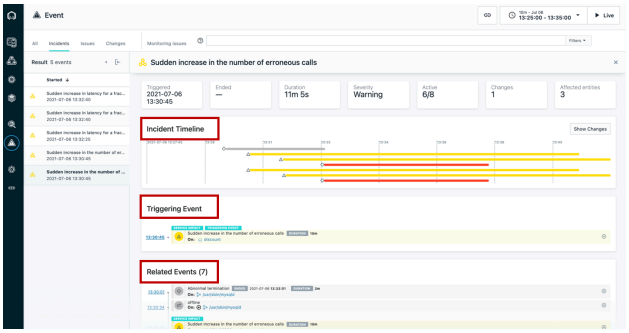
Action 2.1.1

- Click the incident called **Sudden increase in the number of erroneous calls** on the 'discount' service.



Action 2.1.2

- You will see the **Incident Timeline**, **Triggering Event**, and **Related Events**.



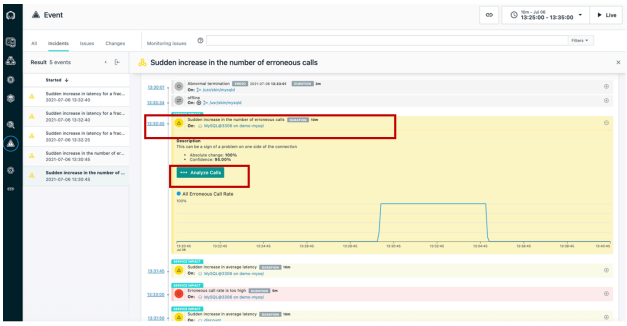
3 - Debugging the incident by inspecting calls

3.1 - Understand the incident

Narration
It looks like the abnormal termination of the MySQL database caused the problem. It shows how one data store issue rippled out to affect a number of directly and indirectly connected services. Instana's automatic root cause analysis uses the relationship information from the dynamic graph to accurately collate the individual issues into one incident. This completely eliminates alert storms, providing your DevOps engineers and SREs with a single notification of actionable information to enable them to promptly restore normal service. Let's look at some related traces for this.

Action 3.1.1

- Under **Related Events**, click the event that says **Sudden increase in the number of erroneous calls**. Then, click **Analyze Calls**.



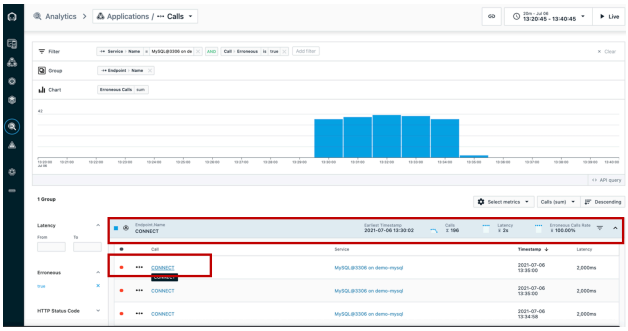
3.2 - Examine the details

Narration
Now, we moved to the Analytics view. You can see how Instana UI allows for easy navigation between different views, keeping the time span and context. At the top, you can see the filter that was applied to all collected traces. All filtered requests are grouped by endpoint (in this case, it is the database CONNECT exposed by the MySQL server).

There is only one endpoint here, but if there were multiple, you'd see a list. Endpoints are automatically discovered and mapped by Instana. We can go into the details for each erroneous call to MySQL via this endpoint (CONNECT).

Action 3.2.1

- Click the endpoint named **CONNECT**. Then, click the first call (also named **CONNECT**).



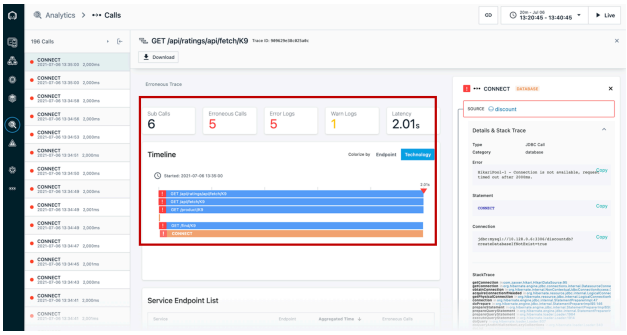
4 - Drilling down with end-to-end traces

4.1 - View the call via the visual dashboard

Narration
Clicking an individual call takes us to a view of the call in the context of the end-to-end trace. We can see where the request began and each call that was made along the way.

Everything is presented in an easy-to-navigate visual dashboard, so we can drill into increasingly detailed information to pinpoint the problem, without using multiple tools or navigating back and forth to lots of dashboards.

Action 4.1.1
- Click the first call on the list.



4.2 - Understand the impact and source of the incident

Narration
In the call stack, we can click each span to see more information, including the complete stack trace. We can see the source, in this case the ‘discount’ service, and [scroll down] the destination, which in this case is CONNECT of MySQL.

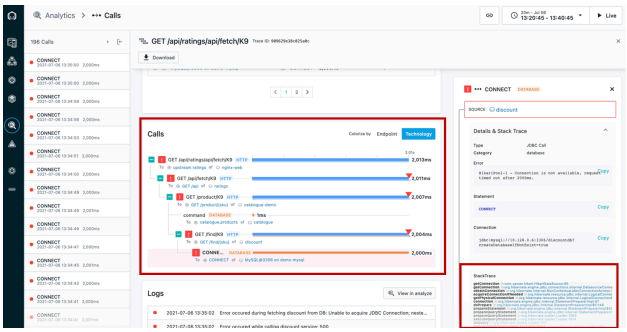
It’s useful to have this context because we can easily see how the calls go from one service to another, just by clicking them. We can also see how the error (red triangle) propagated up the call stack, in this case beginning with the MySQL database.

So we can confirm that the root cause of the incident that affected the ‘discount’ service was with the MySQL database. The abnormal termination of the database caused a connection error, which then flowed back through the application.

When we bring MySQL back online, it will fix the problem.

Action 4.2.1

- Scroll down to the section labeled **Calls**.



5 - Confirming the incident resolution was successful

5.1 - Metrics for the robot shop have returned to normal

Narration
Now that MySQL is working again, we can go back and confirm that the problems with the robot shop have been repaired.

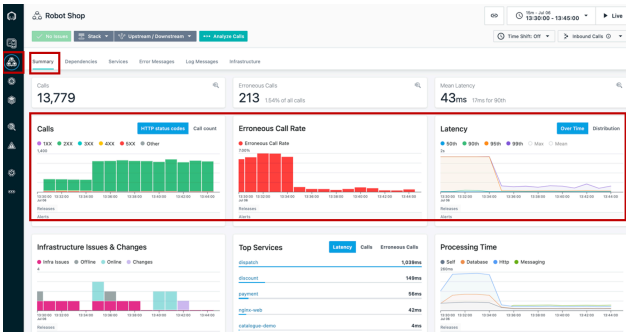
You should see that the call volume has increased, the number of erroneous calls decreased, and latency also decreased.

If you're giving the demo in real time, the incident should have reset itself by the time you're done demo'ing. If not, this part can be skipped.

If you set the timeframe at the beginning of the demo, you can set it again to begin at 0:30 minutes past the hour and end at 0:45 minutes past the hour.

Action 5.1.1

- Navigate to **Applications** in the sidebar menu, choose **Robot Shop**, and click the **Summary** tab.



Summary

Now, we can see that the metrics for the robot shop have returned to normal: the call volume has increased again, the erroneous call rate has decreased, and latency has decreased.

We've fixed the problem with the robot shop and restored normal service!

Hopefully, you've seen that Instana can help make the process of identifying problems and finding the root cause of those problems very frictionless. Since Instana automates so many of the manual and labor-intensive aspects of the process, you can focus on getting other work done and not worry about instrumenting observability or constantly monitoring for problems. And when problems do arise, all the trace data is there at your fingertips to dig into.