# Observability

300-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. Our application is called Stan's Robot Shop, and it uses various technologies such as Java, Python, and MySQL to sell robots online.

Let's get started!

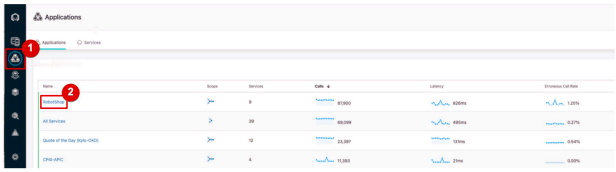
# 1 - Accessing RobotShop's dependencies, events, and alerts using Instana

**Note**: Before proceeding, ensure you have Instana open, as per instructions in the **Demo preparation**.

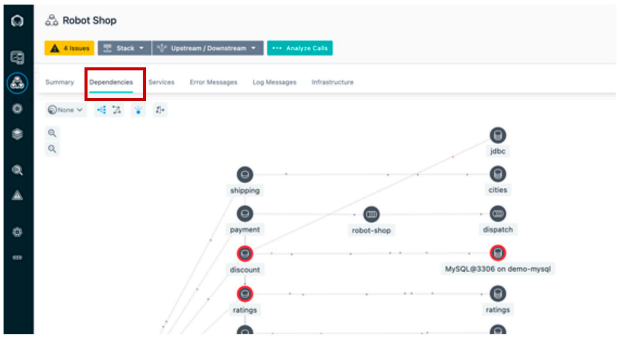
## 1.1 - Navigate to RobotShop's dependencies

**Narration**
We're going to use Instana to view all the dependencies within the RobotShop application. Instana automatically discovers the relationships between the services and correlates them into a dynamic graph.

**Action** 1.1.1
- From the sidebar menu, click the **Applications** icon (1) and choose **RobotShop** (2).




**Action** 1.1.2
- Click the **Dependencies** tab.




**Narration**
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.

**Action** 1.1.3

- Hover the cursor over a few of the icons to show info on what technology they are built on. For example, we can see how the catalogue-demo is built on HTTP, Java Virtual Machine (JVM), and Spring Boot, while the catalogue database is built on MongoDB.


## 1.2 - Automatically assess events and alerts

**Narration**

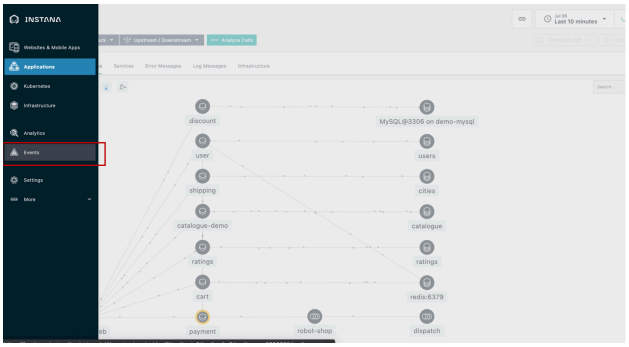
Since you wouldn't normally be looking at the dashboard when something like this happens, let's see the SRE/IT operator's point of view when an incident occurs.

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).

**Action** 1.2.1

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



**Narration**
It’s important to note here that you’re not getting alerts for just anything. Instana automatically groups related events and issues into incidents. It determines what events and/or issues are related using the dynamic dependency graph that we just looked at.

Instana also continuously assesses the groups of events and issues to determine which ones are impacting end users or posing an imminent risk of impacting end users. Those are the ones that Instana will alert on, so as a SRE/IT operator, you will not be interrupted constantly for things that are not very important.

Let’s go into the details for this incident.

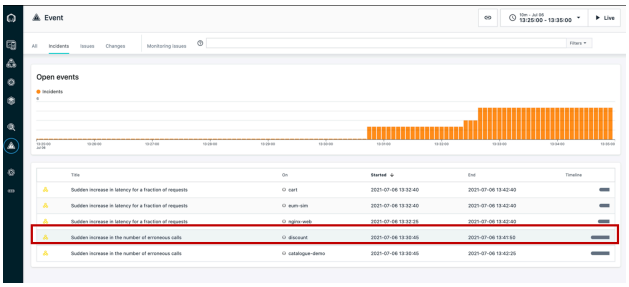
# 2 - Inspecting auto-correlated incident details

## 2.1 - Gather information from the incident detail page

**Narration**
Instana recognized that the sudden increase 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.

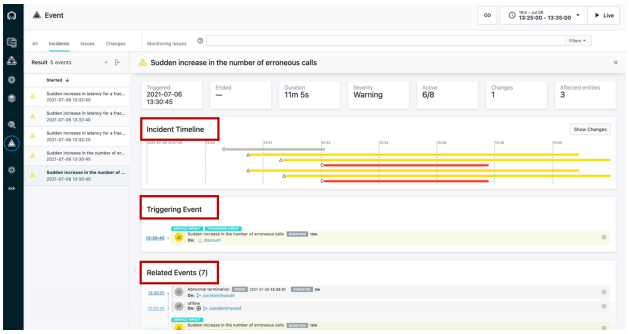
Let's click the incident on the 'discount' service for more details.

**Action** 2.1.1
- Click the incident called **Sudden increase in the number of erroneous calls** on the **discount** service.




**Narration**
We can now see a timeline of the incident, the event that triggered Instana to create the incident, and all of the related events.

**Action** 2.1.2
- Hover the cursor over to show **Incident Timeline**, **Triggering Event**, and **Related Events**.



# 3 - Debugging the incident by inspecting calls

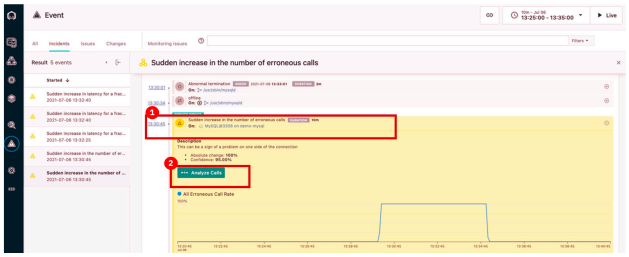
## 3.1 - Understand the incident

**Narration**
Let's take a closer look.

By inspecting the related events, it looks like the abnormal termination of the MySQL database caused the problem. We can go into more detail about each call that failed to connect to the database. Simply select the event that relates to the sudden increase in erroneous calls, then go in to analyze calls. Going into the actual trace for a request that resulted in an error will help us confirm that MySQL is really the source of the incident.

**Action** 3.1.1

- Under **Related Events**, click the event that says **Sudden increase in the number of erroneous calls** (1). Then, click **Analyze Calls** (2).

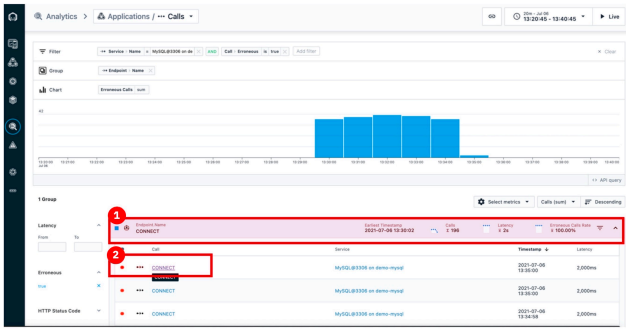



## 3.2 - Examine the details

**Narration**
All the calls are grouped by endpoint. There is only one endpoint showing, but if there were more you'd see a list here. 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** (1). Then, click the first call that is also named **CONNECT** (2).

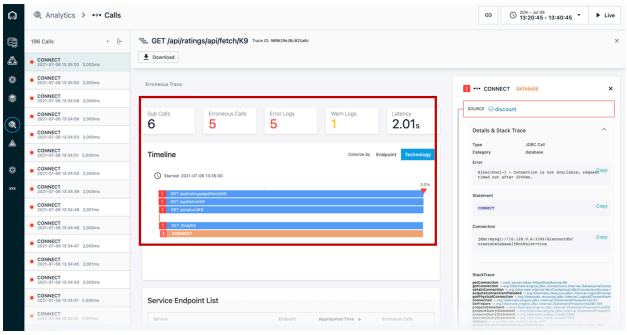


# 4 - Drilling down with end-to-end traces

## 4.1 - View the call via the visual dashboard

**Narration**
Now that we've clicked on an individual call, in this case the first call in the list, we can view the call in the context of the end-to-end trace.

**Action** 4.1.1
- Highlight the middle dashboard area showing elements of the first call on the list. You will need to scroll down to the timeline view, and then change focus to the right column.




**Narration**
We can see where the request began and each call that was made along the way. The timeline view gives a quick overview of the time spent on each span, as well as key performance indicators, such as the number of erroneous calls in this trace, the number of warning logs, and total latency.

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.

In the call stack [move to right column], 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.

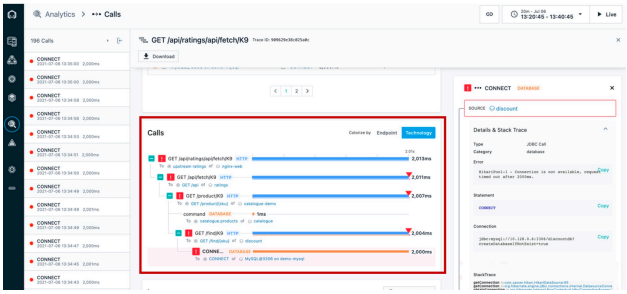
It's useful to have this context because we can easily see how the calls go from one service to another, just by clicking them. We can also see how the error (red triangle) propagated up the call stack, in this case beginning with the MySQL database.

## 4.2 - Understand the impact and source of the incident

**Narration**
Let’s identify the root cause of the incident affecting the ‘discount’ service.

**Action** 4.2.1

- In the middle area, scroll down to the section labeled **Calls** until you reach the MySQL database row at bottom.




**Narration**
We’ve confirmed that the root cause of the incident that affected the ‘discount’ service was with the MySQL database. We can see the abnormal termination of the database caused a connection error, which then flowed back through the application.

# 5 - Confirming the incident resolution was successful

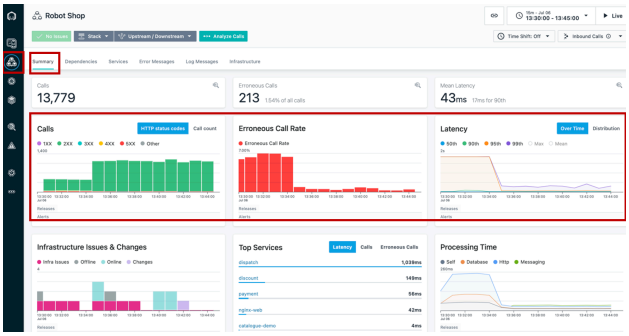
## 5.1 - Observe that metrics for the robot shop have returned to normal

**Narration**
When we bring MySQL back online, it will fix the problem. For the sake of this demonstration, we will assume it is brought back online, rather than depicting it.

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

**Action** 5.1.1

- Navigate to **Applications** in the sidebar menu, and choose **Robot Shop**. If you are not already on the **Summary** tab, click the tab.




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

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

**Action** 5.1.2

- If you're giving the demo in real time, the incident should have reset itself by the time you're done demo'ing.
- 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. Refer to the **1 - Environment setup (past incident)** instructions in the **Prepare to give the demo** section of the **Demo preparation** if you need a refresher of how to do this.

## Summary

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.

I'm happy to take any questions or go back to any part of the demo.