



IBM Instana



5-min video demo <https://youtu.be/0Bs1G2f8RGU>

DBmarlin demo for IBM Instana

Walkthrough linking to <https://play.dbmarlin.com/>



Walkthrough

Demo Steps



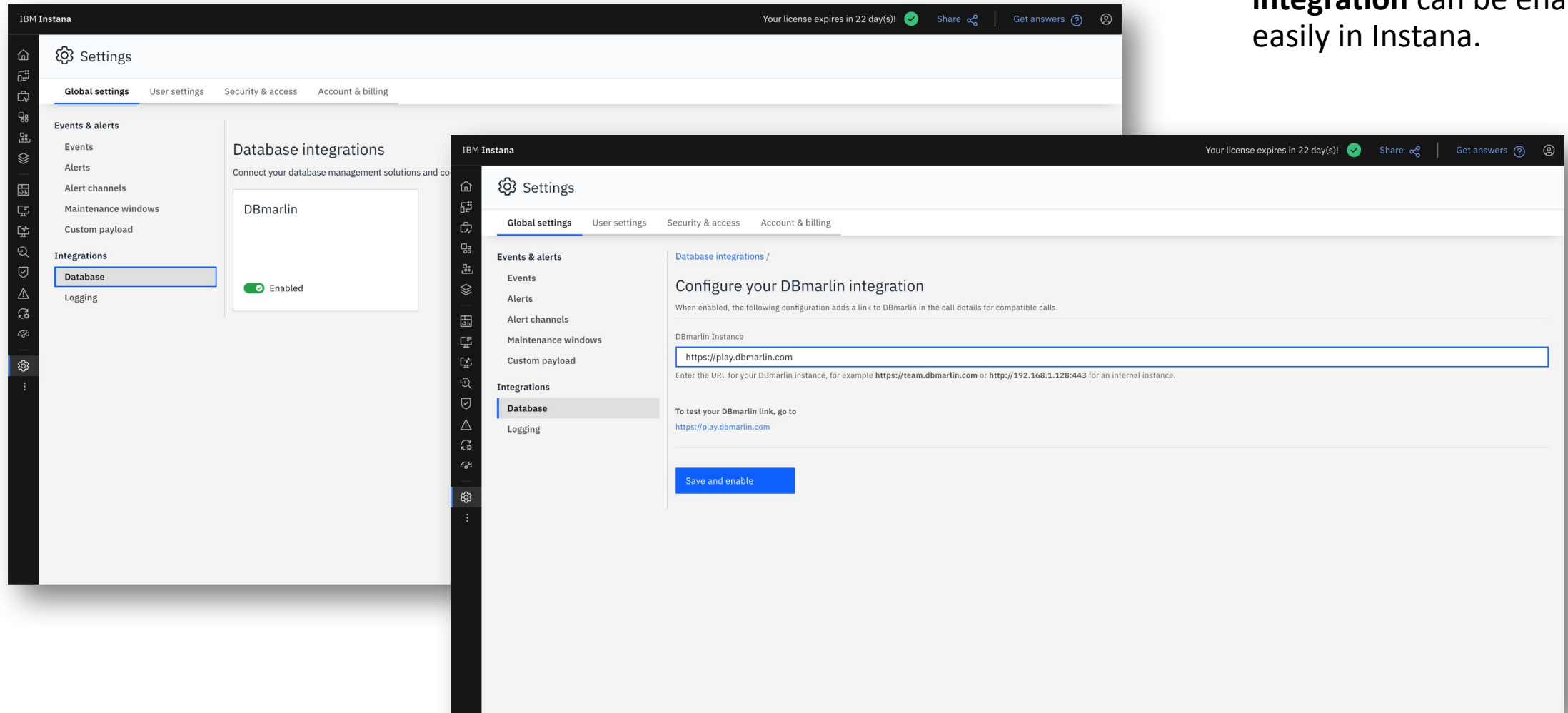
Login to IBM Instana

Switch to either of these environments which has the **BikeStore** demo application:

1. <https://demo-partner.instana.io/>
2. <https://ibmdemo-instanaibm.instana.io/>

Settings -> Integration

1. Show how DBmarlin **integration** can be enabled easily in Instana.



Note that if you don't have the right level of permissions in Instana you might not be able to show this screen so explain that it just requires the URL for the DBmarlin and move to next step.

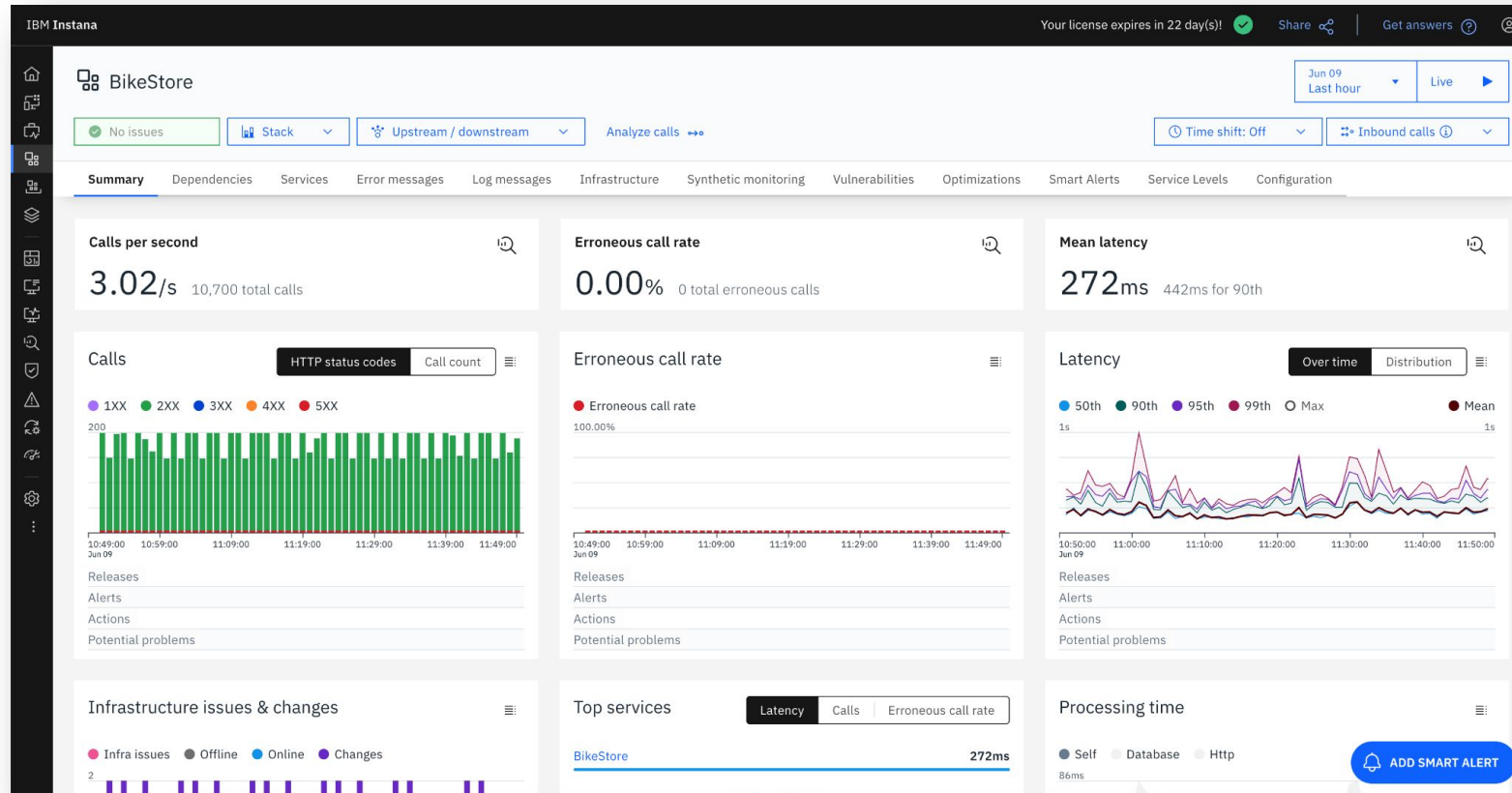
Applications -> BikeStore

1. Click into the **Applications** and then click into **BikeStore** application.

The screenshot shows the IBM Instana web interface. On the left, a dark sidebar contains a navigation menu with items like Home, Websites & ..., Business pro..., Applications, Platforms, Infrastructure, Custom dash..., Logs, Synthetic mo..., Analytics, Vulnerabilities, Events, Automation, Service levels, Settings, and More. A red arrow points to the 'Applications' menu item. The main content area displays the 'Applications' page, which includes a table of monitored applications. The 'BikeStore' application is highlighted with a red border. The table columns are Name, Scope, Services, Calls, Latency, Erroneous call rate, and Health. The 'BikeStore' row shows 10,658 calls and a latency of 272ms.

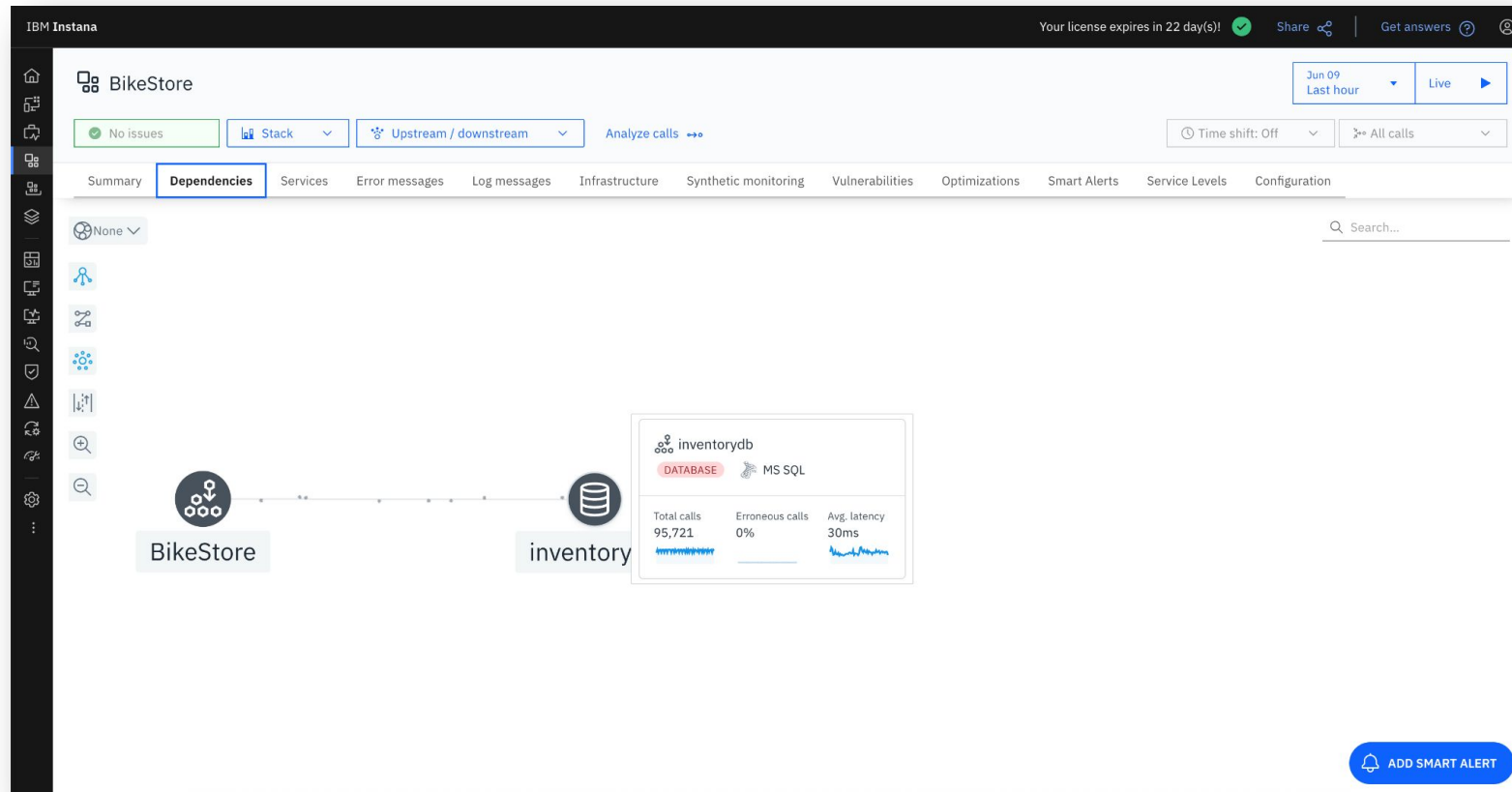
Name	Scope	Services	Calls	Latency	Erroneous call rate	Health
DBmarlin staging1		1	279,606	2ms	0.00%	✓
WebTuna Prod		1	221,831	7ms	0.00%	✓
BikeStore		1	10,658	272ms	0.00%	✓
BikeStore EKS		0	0	0ms	0.00%	✓
Robot Shop (local)		0	0	0ms	0.00%	✓
AP websites		0	0	0ms	0.00%	✓
Robot Shop		0	0	0ms	0.00%	✓
WebTuna Staging		0	0	0ms	0.00%	✓
DBmarlin staging3		0	0	0ms	0.00%	✓
NY Cabs		0	0	0ms	0.00%	✓

BikeStore application



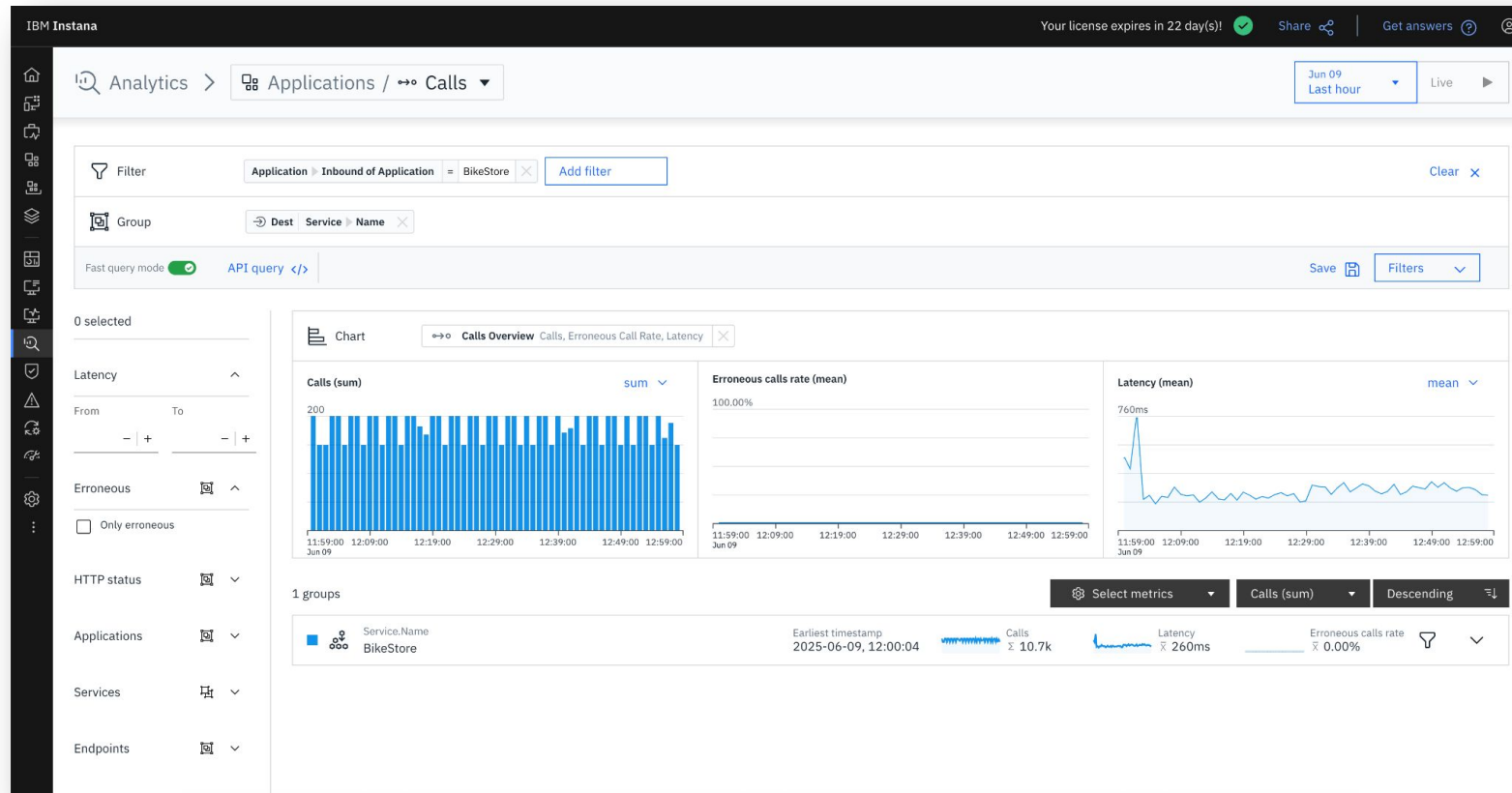
1. Explain that this app has around **3 calls per second** and the latency chart shows the average looks ok but the **99th percentile** show transactions taking **over 1 second**.
2. Next click **Dependencies** tab

Dependencies tab



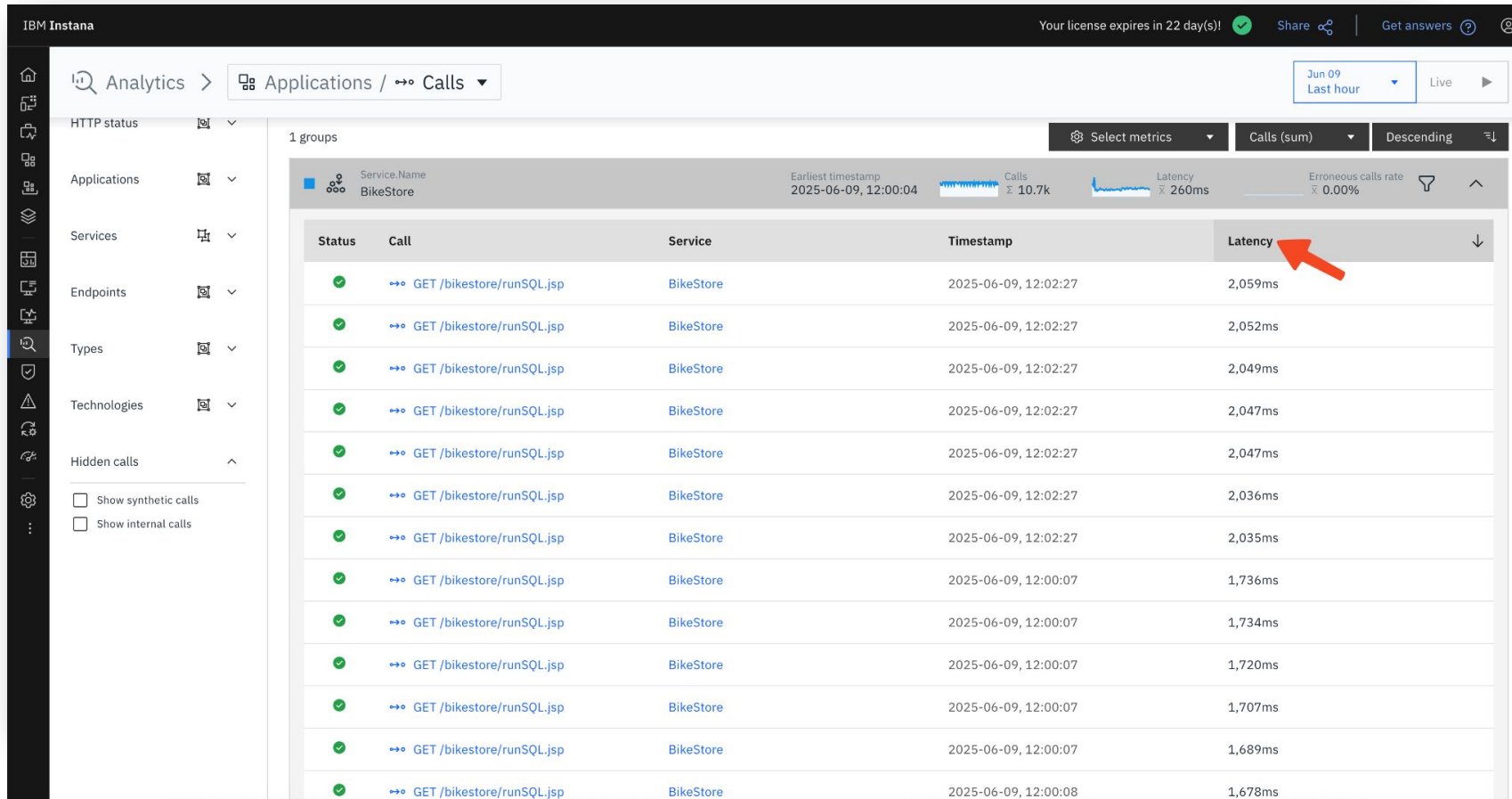
1. Navigate to **Dependencies tab**
2. Mouseover the app tier and database tiers to show more details.
3. Click on **Analyze calls** at the top of the screen

Analytics



1. Expand the group for Service.name=BikeStore to see the individual traces.

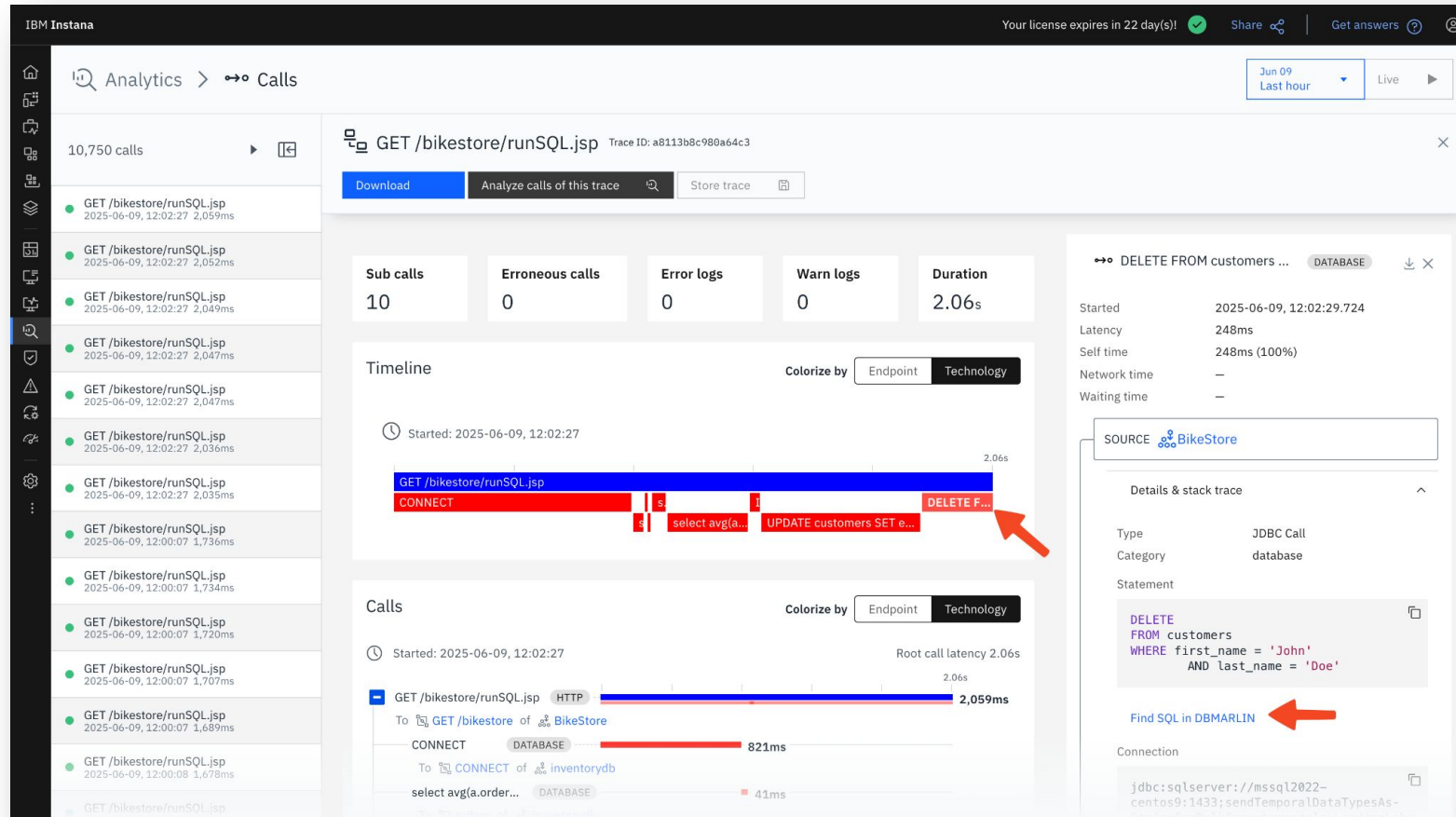
Analytics



Sort the traces by **Latency** descending and you should find some slow ones.

Then click into the top one.

Trace view



1. The trace has a very long response time which is dominated by the long **red** bar which is the database call.
2. Click on the **red** bar for the `DELETE` statement to see the SQL statement text on the right
3. You will see a **Find SQL in DBmarlin** button below it.
4. Click the button which will open `play.dbmarlin.com` in a new browser tab.

DBmarlin SQL Search results

The screenshot displays the DBmarlin SQL Search interface. On the left is a dark blue sidebar with navigation links: ANALYSIS (Database Instances 14, Hosts 11, Event History, SQL Search, Recommendations), REPORTS (Activity Comparison, SQL Stats Explorer, SQL Stats Comparison, Parameter Comparison), and SETTINGS (Database Instances, Hosts, SQL Statistics, Alert Rules, Snapshots, Integrations, Event Types). The main panel is titled 'SQL Search' and shows a search input with the text 'DELETE FROM customers WHERE first_name = 'John' AND last_name = 'Doe''. Below the input is a 'Search for Statement' button. The results section, titled 'Matching Statements 1', shows a table with columns 'Instance', 'Statement', and 'Similarity'. The table contains one entry: 'mssql2022-centos9' with the same DELETE statement and a 100% similarity score. Navigation controls like 'First', 'Back', '1', 'Next', and 'Last' are visible at the bottom of the results table.

SQL Search

Analysis > SQL Search

DELETE FROM customers WHERE first_name = 'John' AND last_name = 'Doe'

Select Instance (Optional)

Search for Statement

Matching Statements 1

Export Clear Search

Instance	Statement	Similarity
mssql2022-centos9	DELETE FROM customers WHERE first_name = 'John' AND last_name = 'Doe'	100%

20 First Back 1 Next Last

COPYRIGHT © APPLICATION PERFORMANCE LTD. ALL RIGHTS RESERVED. 5.5.0

1. DBmarlin will open on the SQL Search screen with the results of the search.
2. Explain that the results ordered by Similarity score and it is doing a fuzzy match since the statement text could be slightly different between app and db due to formatting like white spaces, or quotation marks.
3. Click into it to see the details

DBmarlin Statement and Plans

The screenshot displays the DBmarlin interface for a specific database instance. The top navigation bar includes the DBmarlin logo, instance name 'mssql2022-centos9', environment 'Staging', application 'Orders', and customer 'abc'. The left sidebar contains sections for ANALYSIS (Database Instances, Hosts, Event History, SQL Search, Recommendations), REPORTS (Activity Comparison, SQL Stats Explorer, SQL Stats Comparison, Parameter Comparison), and SETTINGS (Database Instances, Hosts, SQL Statistics, Alert Rules, Snapshots, Integrations, Event Types).

The main content area is divided into two tabs: 'SQL Text and Plans' and 'SQL Performance'. The 'SQL Text and Plans' tab is active, showing the SQL statement:

```
DELETE FROM
customers
WHERE
  first_name = 'John'
  AND last_name = 'Doe'
```

Below the SQL text, the 'Execution Plans' section is visible. It includes a title 'The execution plan below is for the period 09 Jun 2025 12:06 to 13:00 for the database and username .' and a diagram of the execution plan. The plan starts with a 'DELETE' operation, followed by a 'Sequence' operation, then a 'Table Spool (Eager Spool)' operation. The main part of the plan is a 'Clustered Index Delete (Delete)' operation, which is further detailed in a pop-up window. This window shows the 'Clustered Index Scan (Clustered)' operation, which is a 'Physical Operation' and 'Logical Operation'. It includes a table with the following data:

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.0095029 (10%)
Estimated I/O Cost	0.0075694
Estimated CPU Cost	0.0019335
Estimated Subtree Cost	0.0095029
Estimated Number of Executions	1
Estimated Number of Rows to be Read	1615
Estimated Number of Rows	1615
Estimated Row Size	15 B
Ordered	True
Node ID	9

The pop-up window also shows the 'Output List' with the following columns: [inventorydb].[dbo].[orders].order_id, [inventorydb].[dbo].[orders].customer_id, and [inventorydb].[dbo].[orders].customer_id.

1. DBmarlin will show the Statement text and any **Execution Plans**.
2. Execution plans show the steps the database optimizer has chosen to execute to return the results. It can be very useful when tuning SQL.
3. Next click **Send to DBmarlin Co-pilot** to get suggestions on how to optimize the SQL.

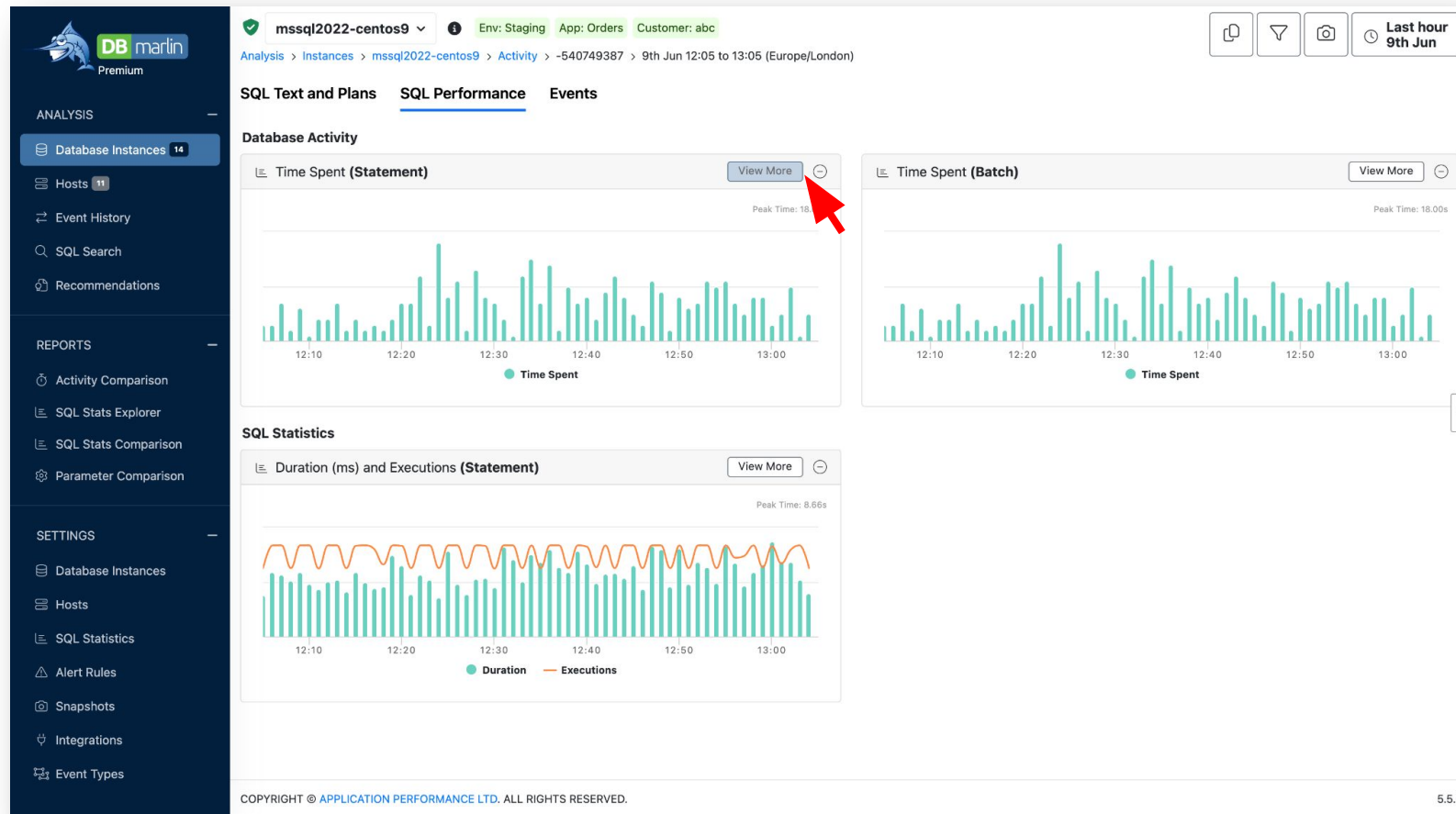
DBmarlin Co-pilot for SQL Tuning Advice

The screenshot displays the DBmarlin Co-pilot interface, which is integrated into the DBmarlin Premium dashboard. The dashboard's left sidebar contains navigation menus for ANALYSIS (Database Instances, Hosts, Event History, SQL Search, Recommendations), REPORTS (Activity Comparison, SQL Stats Explorer, SQL Stats Comparison, Parameter Comparison), and SETTINGS (Database Instances, Hosts, SQL Statistics, Alert Rules, Snapshots, Integrations, Event Types).

The main interface is divided into three panels. The left panel, titled 'SQL Text and Plans', shows the SQL statement: `DELETE FROM customers WHERE first_name = 'John' AND last_name = 'Doe'`. The middle panel, titled 'Execution Plans', displays a detailed execution plan for the statement, including steps like 'Sequence', 'Table Spool (Eager Spool)', 'Clustered Index Delete (Delete)', 'Clustered Index Delete (Delete)', 'Hash Match (Inner Join)', 'Nested Loops (Inner Join)', and 'Table Spool (Eager Spool)'. The right panel, titled 'DBmarlin Co-pilot', provides AI-generated tuning suggestions. It includes a 'Key Observations from Your Plan' section with bullet points about the query's performance, a 'Tuning Suggestions' section with numbered advice on indexes and foreign keys, and a 'Statistics' section. A code block shows the recommended index creation: `CREATE INDEX idx_customers_name ON customers (first_name, last_name)`. The bottom of the interface features a message input field, a 'Clear History' button, and an 'AI Settings' button.

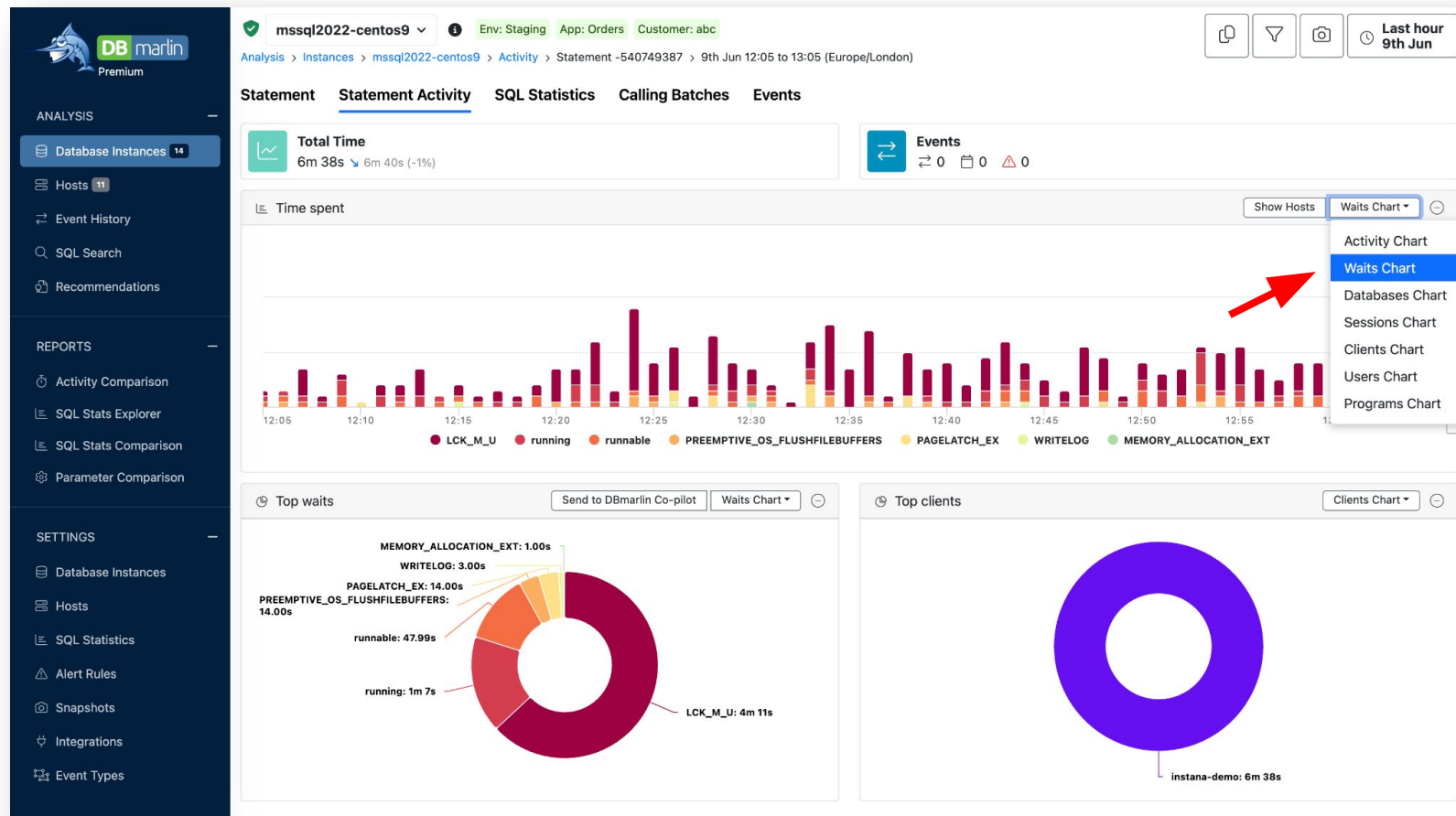
1. DBmarlin Co-pilot is an AI assistant which is available on any screen within DBmarlin.
2. In this case it has analyzed the SQL statement text and plan and made a recommendation to use an index.
3. Click to slide the Co-pilot panel away when you are done.

SQL Performance



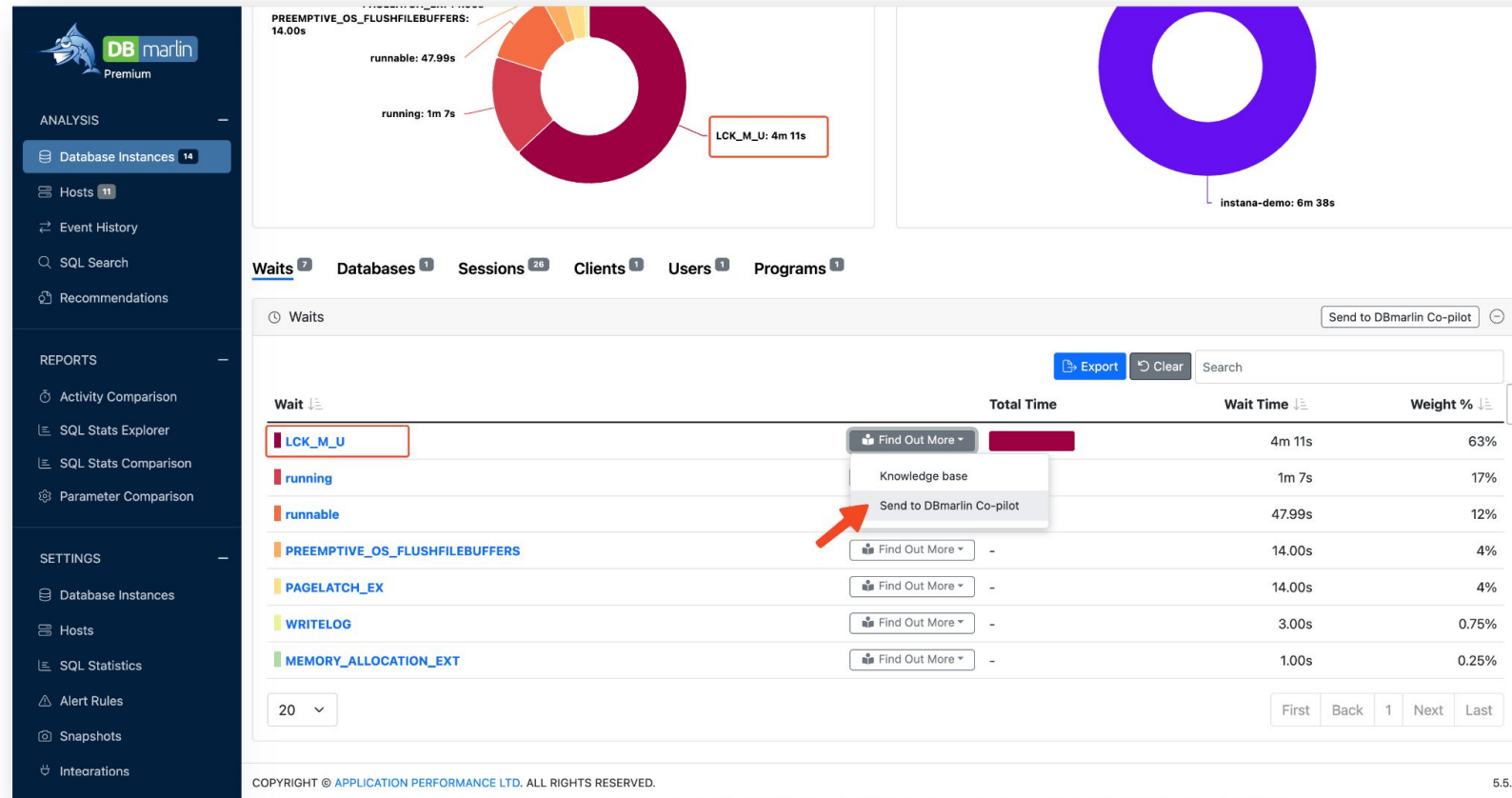
1. Click on the **SQL Performance** tab to see charts for SQL Activity and SQL Statistics.
2. In the top right corner of the Time Spent (Statement) chart click the **View more** button.

DBmarlin Statement Activity



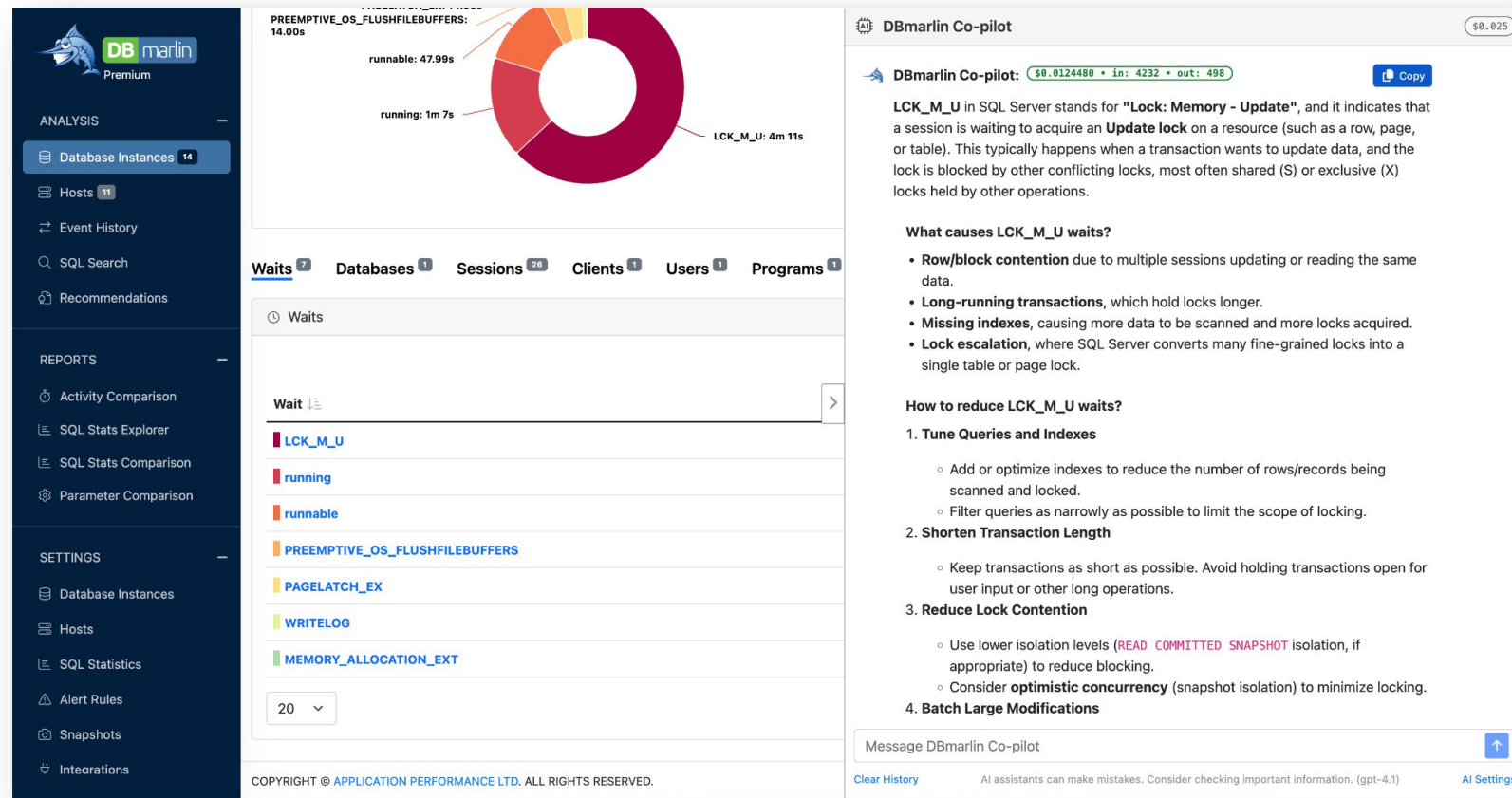
1. Statement Activity tab show the activity for the statement over time.
2. From the Dropdown select **Waits Chart** to see wait-events over time. You will see the DELETE statement spends more than half it's time waiting on a lock **LCK_M_U**
3. Scroll down to the table below

DBmarlin Statement Wait time



1. For the wait event **LCK_M_U** you can click to ask the **Knowledge base** or **Co-pilot** for more information.
2. Click co-pilot for an explanation of the wait and what can be done to reduce it.

DBmarlin Co-pilot for Wait Event Advice



1. Co-pilot can explain the LCK_M_U wait event and what can be done to reduce it.

Further information

- 5-min video demo <https://youtu.be/0Bs1G2f8RGU>
- 1 FREE license for IBM Instana customers <https://www.dbmarlin.com/instana-offer>



IBM Instana

