Service Analysis

Dynatrace Training Module

dynatrace

Agenda

- Service Drilldowns
  - Details
  - Key Requests
  - Multidimensional Analysis Views
  - Compare
- Understand Dependencies
  - Service Flow
  - Service Backtrace
- Analyze Transactions
  - View Web Requests
  - Response Time Distribution
  - Response Time Hotspots
  - Failure Analysis
  - Exception Analysis
  - PurePaths
- Service Analysis during a Problem

Service Drilldowns

Service Drilldowns

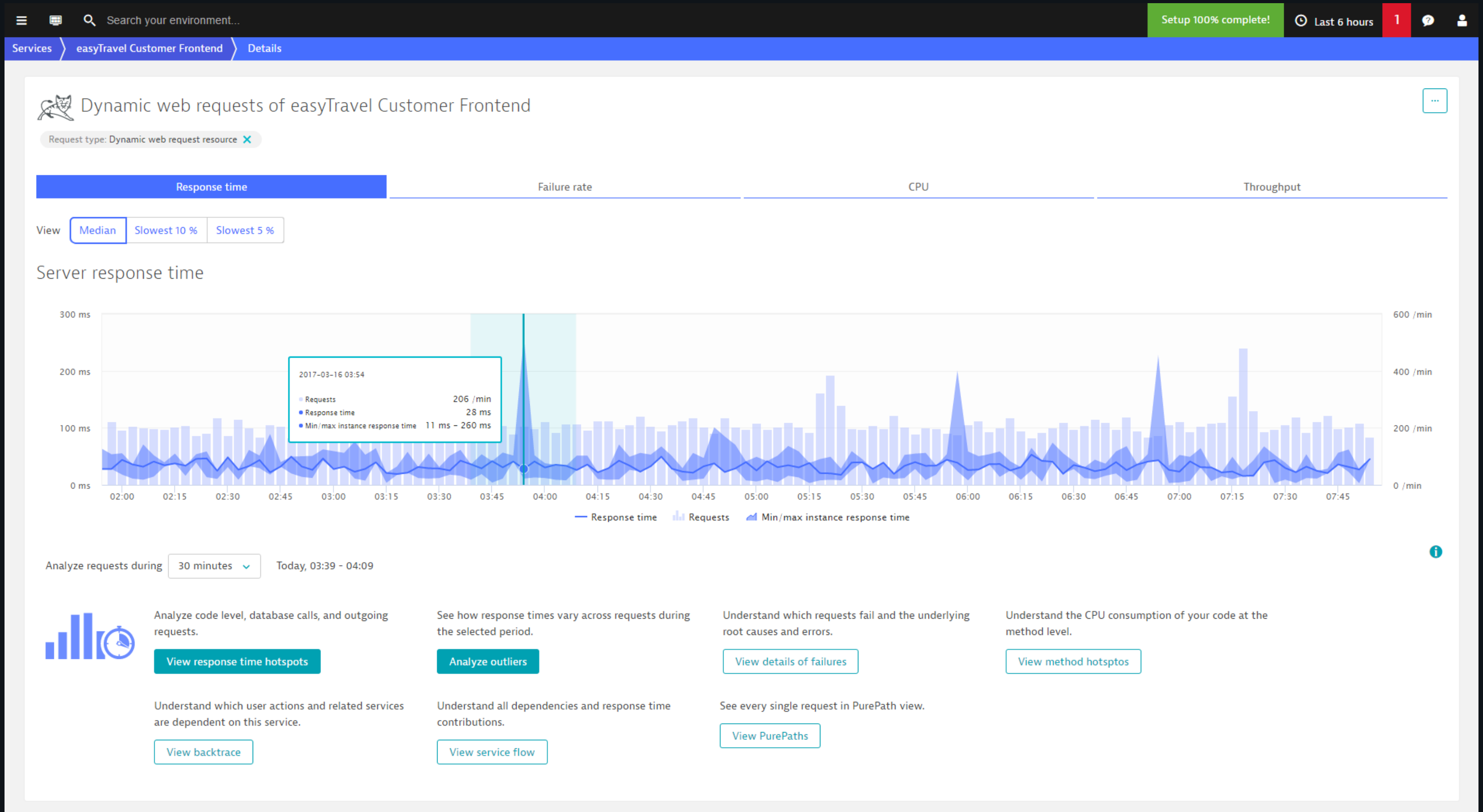
Details

Key Requests

Multidimensional Analysis Views

Compare

Details



Services 〉 easyTravel Customer Frontend 〉 Details

Dynamic web requests of easyTravel Customer Frontend

Request type: Dynamic web request resource

View: Median | Slowest 10 % | Slowest 5 %

Server response time

2017-03-16 03:54
Requests 206 /min
Response time 28 ms
Min/max instance response time 11 ms – 260 ms

Analyze requests during 30 minutes Today, 03:39 - 04:09

Analyze code level, database calls, and outgoing requests.

See how response times vary across requests during the selected period.

Understand which requests fail and the underlying root causes and errors.

Understand the CPU consumption of your code at the method level.

Understand which user actions and related services are dependent on this service.

Understand all dependencies and response time contributions.

See every single request in PurePath view.

Details

- What is it?
  - Detailed overview of a service's performance
  - Starting point for further analysis
- When would I use it?
  - Understand the overall performance over time
  - Beginning manual hotspot and failure analysis
  - Landing view for several problem root causes

Service Drilldowns

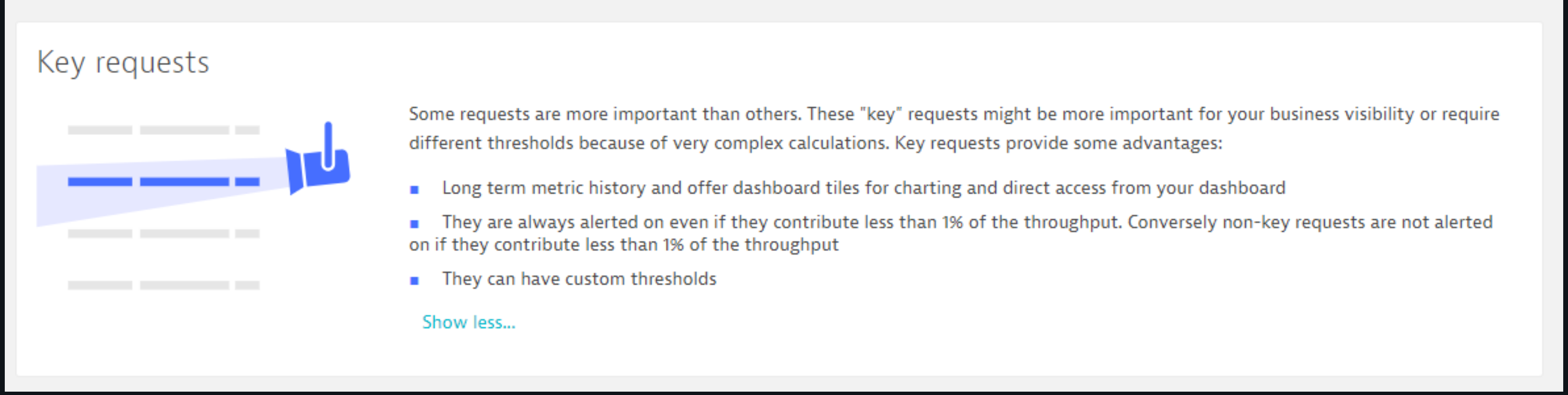
Details

Key Requests ◀

Multidimensional Analysis Views

Compare

Key Requests



Key requests

Some requests are more important than others. These "key" requests might be more important for your business visibility or require different thresholds because of very complex calculations. Key requests provide some advantages:

- Long term metric history and offer dashboard tiles for charting and direct access from your dashboard
- They are always alerted on even if they contribute less than 1% of the throughput. Conversely non-key requests are not alerted on if they contribute less than 1% of the throughput
- They can have custom thresholds

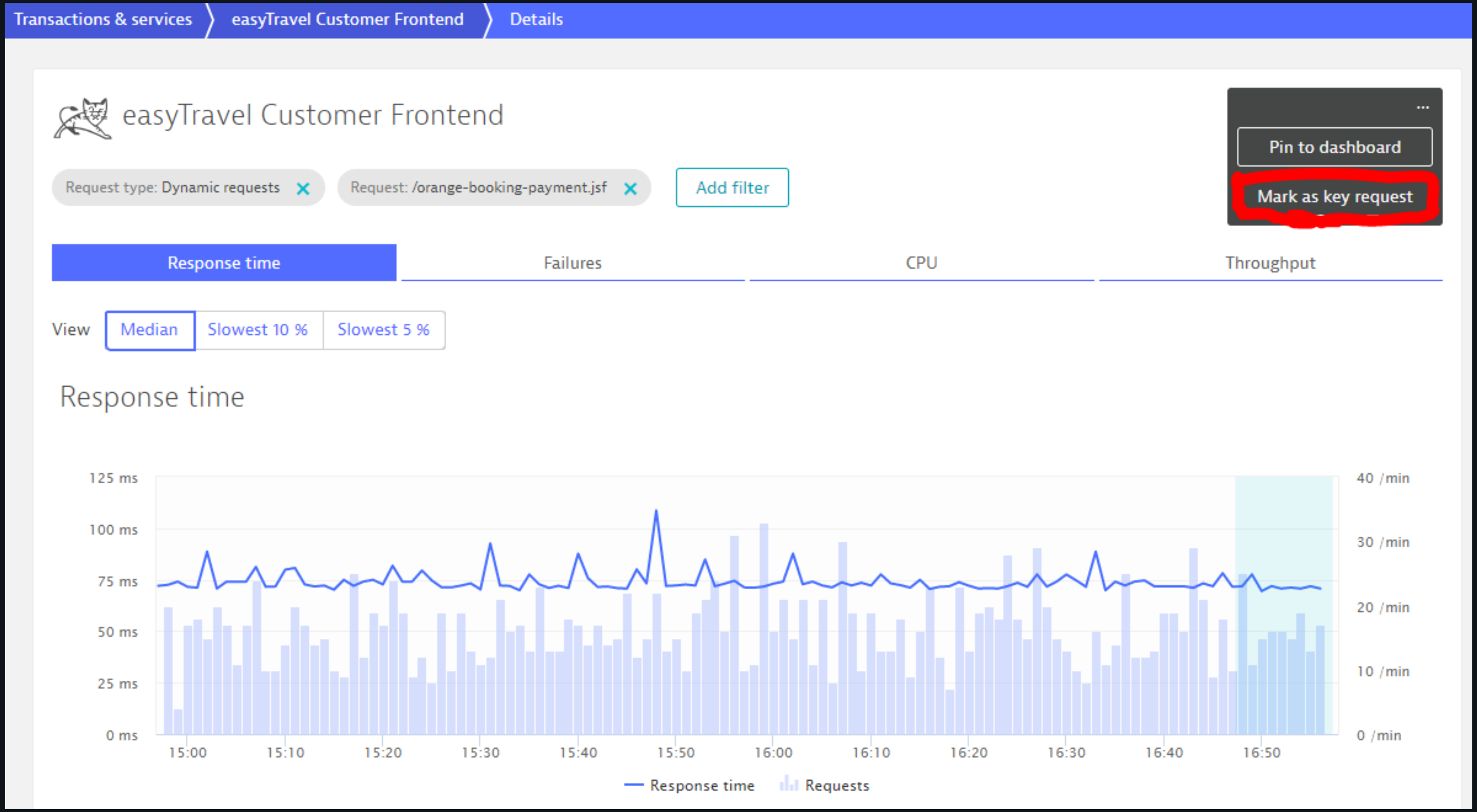
Show less...

Key Requests

- Not every request has the same importance to the success of your digital business. For example, visits to critical landing pages and shopping cart views can be particularly vital to your business operations and your customers’ experience with your application.
- To give such critical requests the attention they deserve, Dynatrace enables you to flag certain requests as key requests. You can define your own response time and failure rate thresholds for key requests, access long-term monitoring data, and pin key requests to your dashboards in the form of dedicated tiles.
- Custom anomaly detection for key requests. (Discussed later)

Configure a Request as Key

- From the service details screen view the 'top requests'
- In the list of requests, select the request you'd like to make 'key' and filter on it
- When analyzing a single request, you can select the button 'mark as key request'



Transactions & services 〉 easyTravel Customer Frontend 〉 Details

easyTravel Customer Frontend

Request type: Dynamic requests ✕

Request: /orange-booking-payment.jsf ✕

Service Drilldowns

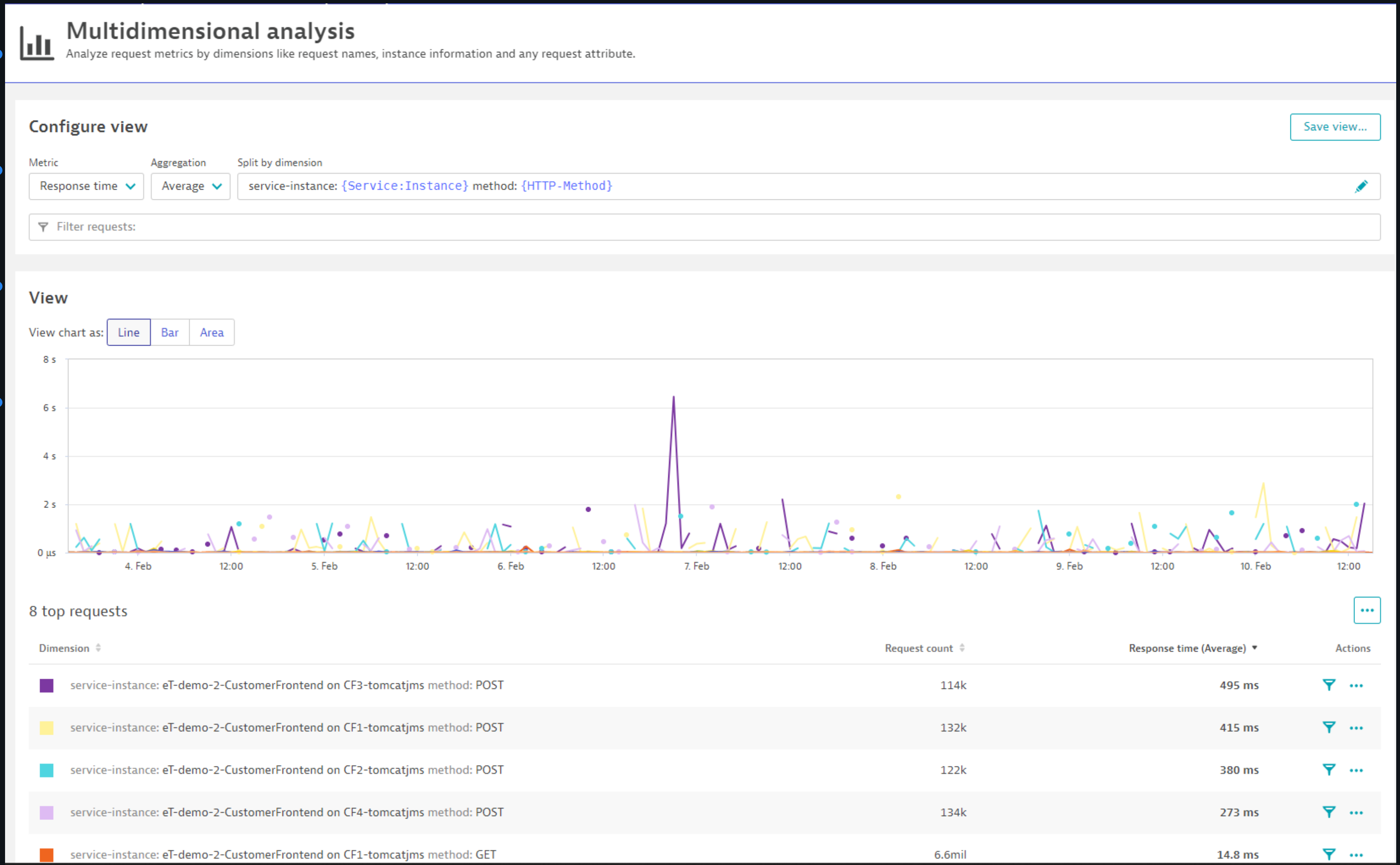
Details

Key Requests

Multidimensional Analysis Views

Compare

Multidimensional Analysis Views



Multidimensional analysis

Analyze request metrics by dimensions like request names, instance information and any request attribute.

Configure view

Metric: Response time

Aggregation: Average

Split by dimension: service-instance: {Service:Instance} method: {HTTP-Method}

Filter requests:

View

View chart as: Line | Bar | Area

8 top requests

	Dimension	Request count	Response time (Average)	Actions
🟪	service-instance: eT-demo-2-CustomerFrontend on CF3-tomcatjms method: POST	114k	495 ms	▼ •••
🟨	service-instance: eT-demo-2-CustomerFrontend on CF1-tomcatjms method: POST	132k	415 ms	▼ •••
🟦	service-instance: eT-demo-2-CustomerFrontend on CF2-tomcatjms method: POST	122k	380 ms	▼ •••
🟪	service-instance: eT-demo-2-CustomerFrontend on CF4-tomcatjms method: POST	134k	273 ms	▼ •••
🟧	service-instance: eT-demo-2-CustomerFrontend on CF1-tomcatjms method: GET	6.6mil	14.8 ms	▼ •••

Service Drilldowns

Details

Key Requests

Multidimensional Analysis Views ◀

Compare

Multidimensional Analysis Views

- What is it?
  - Book marked trending analysis of specific metrics relative to selected service
  - Drill into long-term trends
- When would I use it?
  - Bookmark metrics for others to view
  - Combine metrics for deeper understanding of transaction behavior
  - Set filters
    - Show only specific instance of service request
- Environment wide MDA view available under *diagnostic tools*
- Soon you will be able to define calculated service metrics from this view

Service Drilldowns

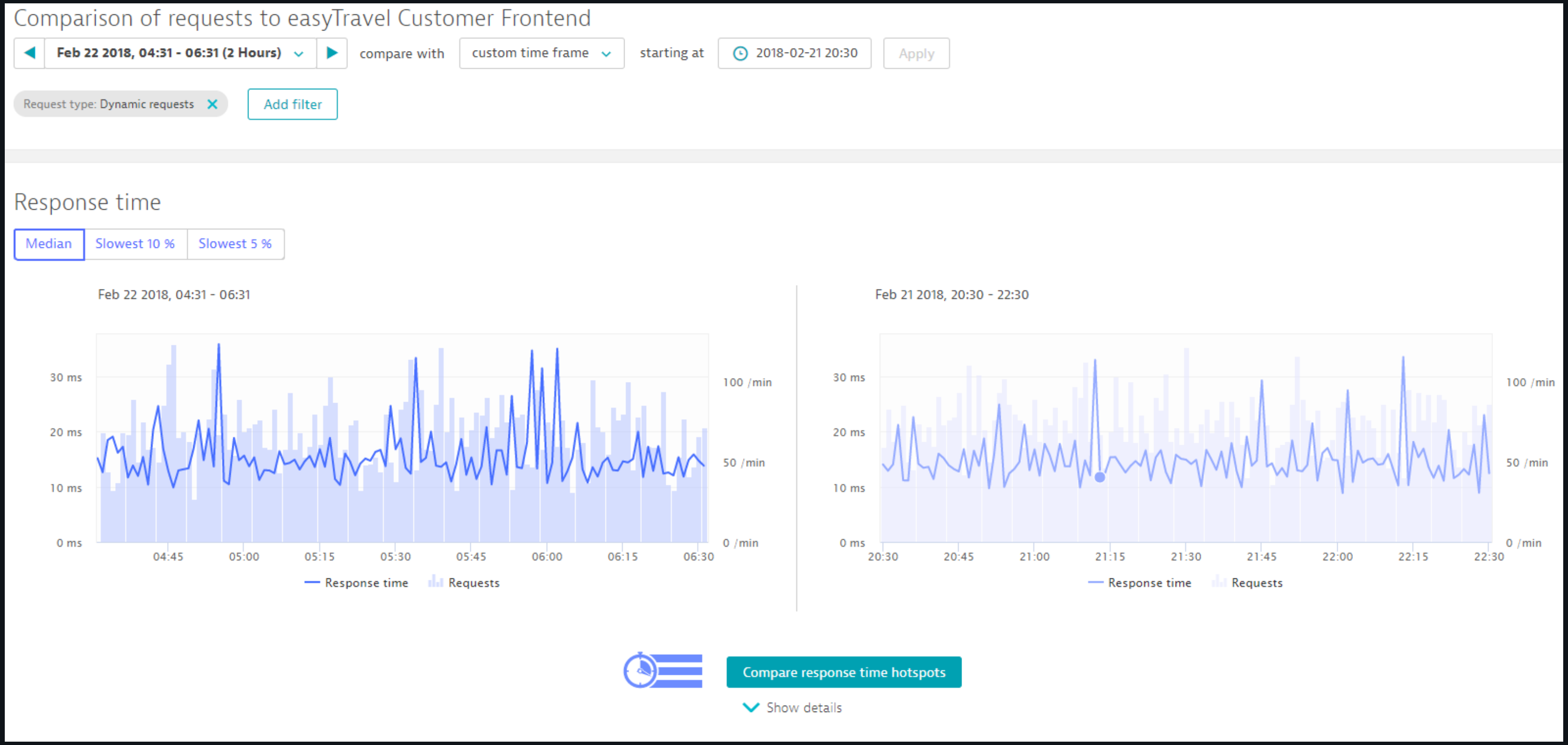
Details

Key Requests

Multidimensional Analysis Views

Compare ◀

Compare



Comparison of requests to easyTravel Customer Frontend

◀ Feb 22 2018, 04:31 - 06:31 (2 Hours) ▶ compare with custom time frame starting at 2018-02-21 20:30 Apply

Request type: Dynamic requests ✕ Add filter

Response time

Median | Slowest 10 % | Slowest 5 %

Feb 22 2018, 04:31 - 06:31

Feb 21 2018, 20:30 - 22:30

Compare response time hotspots

Show details

Compare

- What is it?
  - Compare view enables you to compare critical service-request metrics across two time frames
- When would I use it?
  - View performance metrics for Requests, Attributes, or Instances
    - Response time
    - Failures
    - CPU
    - Load

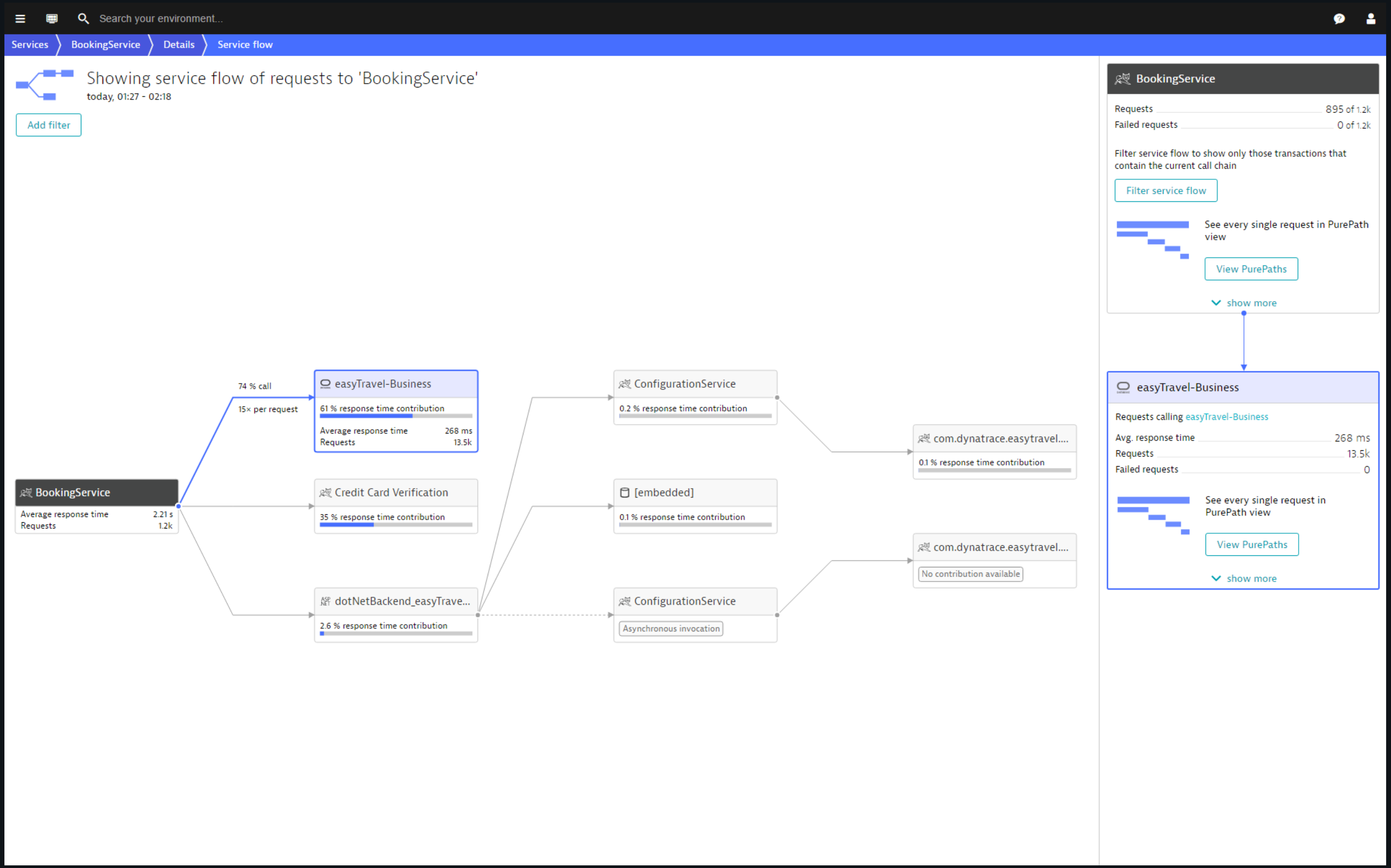
Understand Dependencies

Understand Dependencies

Service Flow

Service Backtrace

Service Flow



Search your environment...
Services
BookingService
Details
Service flow
Add filter

Showing service flow of requests to 'BookingService'

today, 01:27 - 02:18

BookingService
Average response time 2.21 s
Requests 1.2k
74 % call
15× per request
easyTravel-Business
61 % response time contribution
Average response time 268 ms
Requests 13.5k
Credit Card Verification
35 % response time contribution
dotNetBackend_easyTrave...
2.6 % response time contribution
ConfigurationService
0.2 % response time contribution
[embedded]
0.1 % response time contribution
ConfigurationService
Asynchronous invocation
com.dynatrace.easytravel....
0.1 % response time contribution
com.dynatrace.easytravel....
No contribution available

BookingService

Requests 895 of 1.2k

Failed requests 0 of 1.2k

Filter service flow to show only those transactions that contain the current call chain

See every single request in PurePath view

easyTravel-Business

Requests calling easyTravel-Business

Avg. response time 268 ms

Requests 13.5k

Failed requests 0

See every single request in PurePath view

Understand Dependencies

Service Flow ◀

Service Backtrace

Service Flow

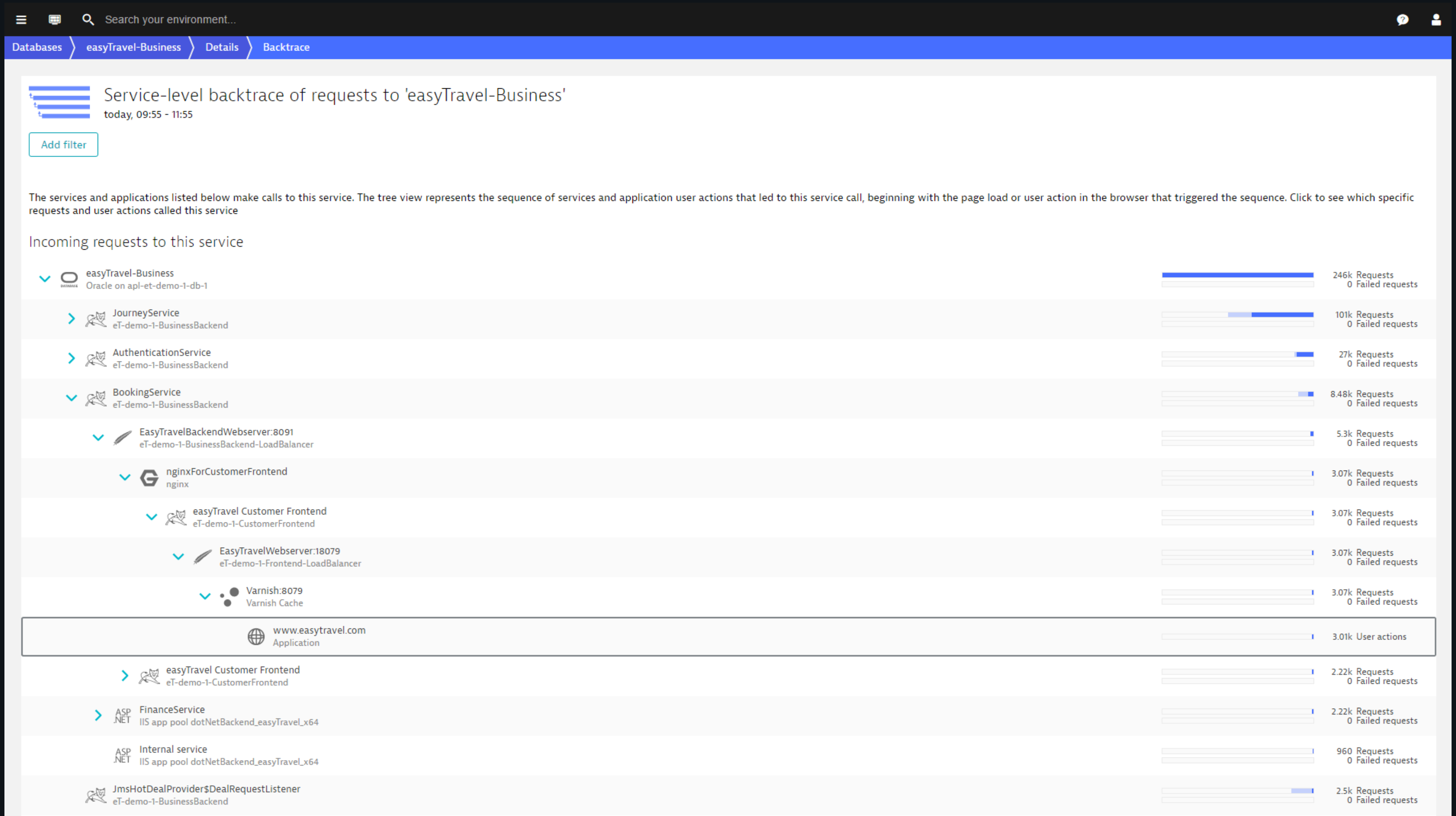
- What is it?
  - Overview of all services and queues that a selected service makes requests to and the time spent within those services
- When would I use it?
  - Understand the call chain sequence of a service
  - View all the response time contributors for a service
  - View affected tiers during active or resolved problems

Understand Dependencies

Service Flow

Service Backtrace

Service Backtrace



Service-level backtrace of requests to 'easyTravel-Business'

today, 09:55 - 11:55

The services and applications listed below make calls to this service. The tree view represents the sequence of services and application user actions that led to this service call, beginning with the page load or user action in the browser that triggered the sequence. Click to see which specific requests and user actions called this service

Incoming requests to this service

easyTravel-Business Oracle on apl-et-demo-1-db-1	246k Requests	0 Failed requests
JourneyService eT-demo-1-BusinessBackend	101k Requests	0 Failed requests
AuthenticationService eT-demo-1-BusinessBackend	27k Requests	0 Failed requests
BookingService eT-demo-1-BusinessBackend	8.48k Requests	0 Failed requests
EasyTravelBackendWebserver:8091 eT-demo-1-BusinessBackend-LoadBalancer	5.3k Requests	0 Failed requests
nginxForCustomerFrontend nginx	3.07k Requests	0 Failed requests
easyTravel Customer Frontend eT-demo-1-CustomerFrontend	3.07k Requests	0 Failed requests
EasyTravelWebserver:18079 eT-demo-1-Frontend-LoadBalancer	3.07k Requests	0 Failed requests
Varnish:8079 Varnish Cache	3.07k Requests	0 Failed requests
www.easytravel.com Application	3.01k User actions	
easyTravel Customer Frontend eT-demo-1-CustomerFrontend	2.22k Requests	0 Failed requests
FinanceService IIS app pool dotNetBackend_easyTravel_x64	2.22k Requests	0 Failed requests
Internal service IIS app pool dotNetBackend_easyTravel_x64	960 Requests	0 Failed requests
JmsHotDealProvider$DealRequestListener eT-demo-1-BusinessBackend	2.5k Requests	0 Failed requests

Understand Dependencies

Service Flow

Service Backtrace ◀

Service Backtrace

- What is it?
  - A view that shows information about who makes calls to a particular service
- When would I use it?
  - Understand what services call the selected service
  - Analyze the performance of a service from the perspective of the calling clients

Analyze Transactions

Analyze Transactions

View Web Requests

Response Time Distribution

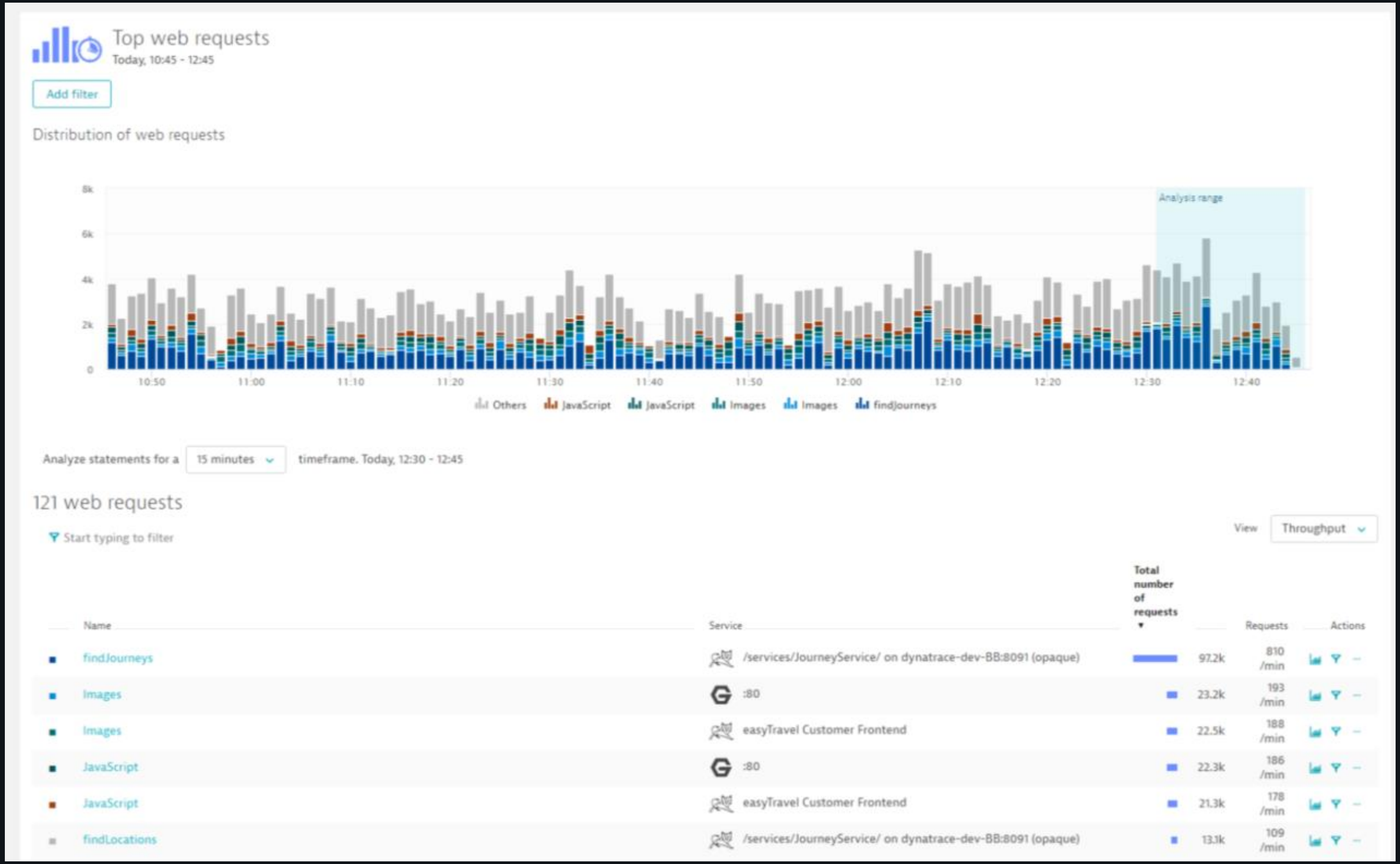
Response Time Hotspots

Failure Analysis

Exception Analysis

PurePaths

View Web Requests



Top web requests

Today, 10:45 - 12:45

Distribution of web requests

Analyze statements for a 15 minutes timeframe. Today, 12:30 - 12:45

121 web requests

	Name	Service	Total number of requests	Requests	Actions
■	findJourneys	/services/JourneyService/ on dynatrace-dev-BB:8091 (opaque)	97.2k	810 /min	
■	Images	:80	23.2k	193 /min	
■	Images	easyTravel Customer Frontend	22.5k	188 /min	
■	JavaScript	:80	22.3k	186 /min	
■	JavaScript	easyTravel Customer Frontend	21.3k	178 /min	
■	findLocations	/services/JourneyService/ on dynatrace-dev-BB:8091 (opaque)	13.1k	109 /min	

Analyze Transactions

View Web Requests ◀

Response Time Distribution

Response Time Hotspots

Failure Analysis

Exception Analysis

PurePaths

View Web Requests

- What is it?
    - A feature that allows you to quickly view all of the web requests within the currently selected timeframe
- When would I use it?
    - Find a specific request by filtering on different aspects, such as:
        - Response Time
        - HTTP Method
        - HTTP Response Code
        - Request Attribute

Analyze Transactions

View Web Requests

Response Time Distribution

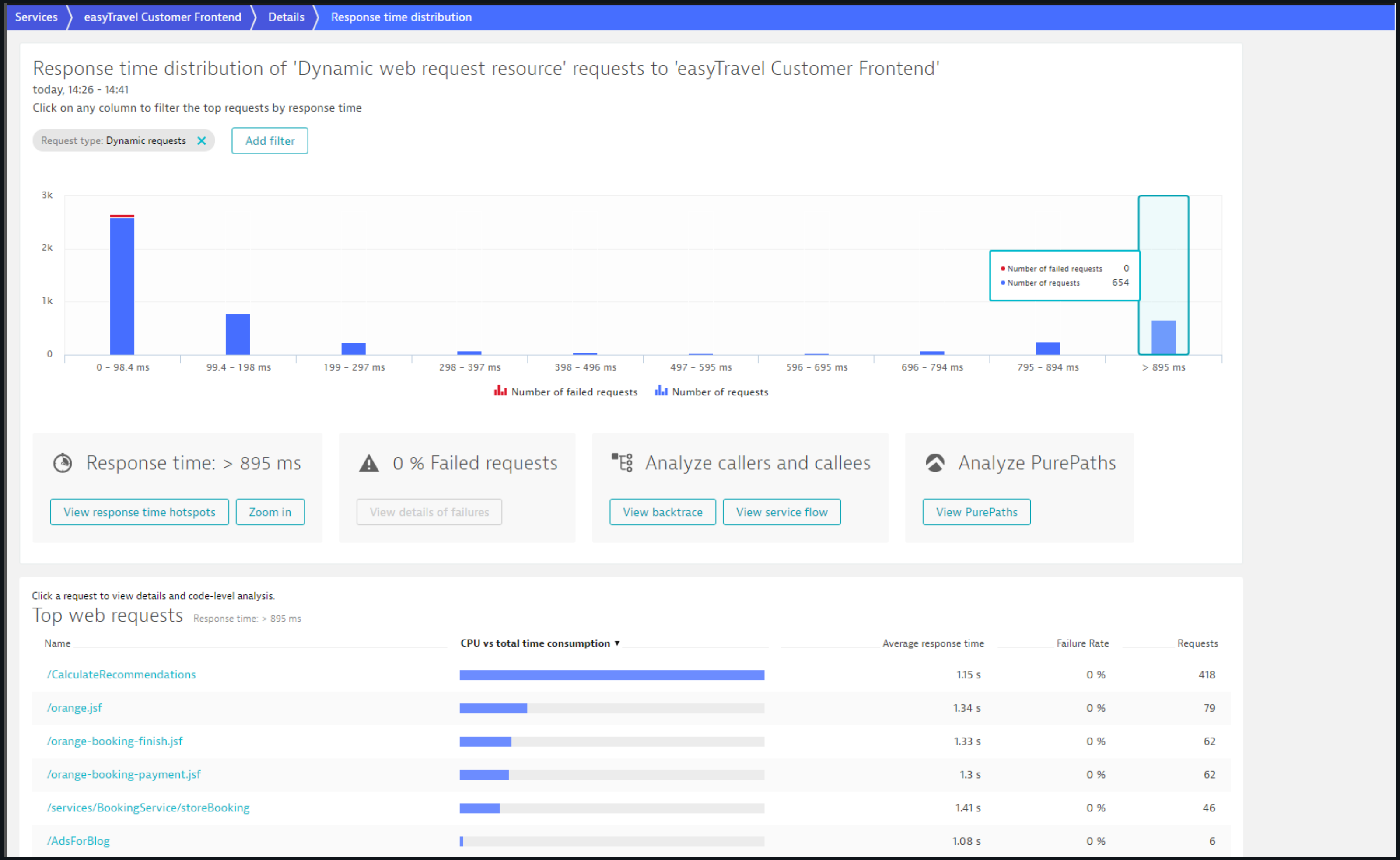
Response Time Hotspots

Failure Analysis

Exception Analysis

PurePaths

Response Time Distribution



Response time distribution of 'Dynamic web request resource' requests to 'easyTravel Customer Frontend'

today, 14:26 - 14:41

Click on any column to filter the top requests by response time

Click a request to view details and code-level analysis.

Top web requests Response time: > 895 ms

Name	CPU vs total time consumption ▾	Average response time	Failure Rate	Requests
/CalculateRecommendations		1.15 s	0 %	418
/orange.jsf		1.34 s	0 %	79
/orange-booking-finish.jsf		1.33 s	0 %	62
/orange-booking-payment.jsf		1.3 s	0 %	62
/services/BookingService/storeBooking		1.41 s	0 %	46
/AdsForBlog		1.08 s	0 %	6

Analyze Transactions

View Web Requests

Response Time Distribution ◀

Response Time Hotspots

Failure Analysis

Exception Analysis

PurePaths

Response Time Distribution

- What is it?
  - A feature that allows you to quickly view the variance in request duration
- When would I use it?
  - Easily view performance outliers and pick them out for deeper analysis
  - Quickly view changes in performance duration during problems vs normal behavior

Analyze Transactions

View Web Requests

Response Time Distribution

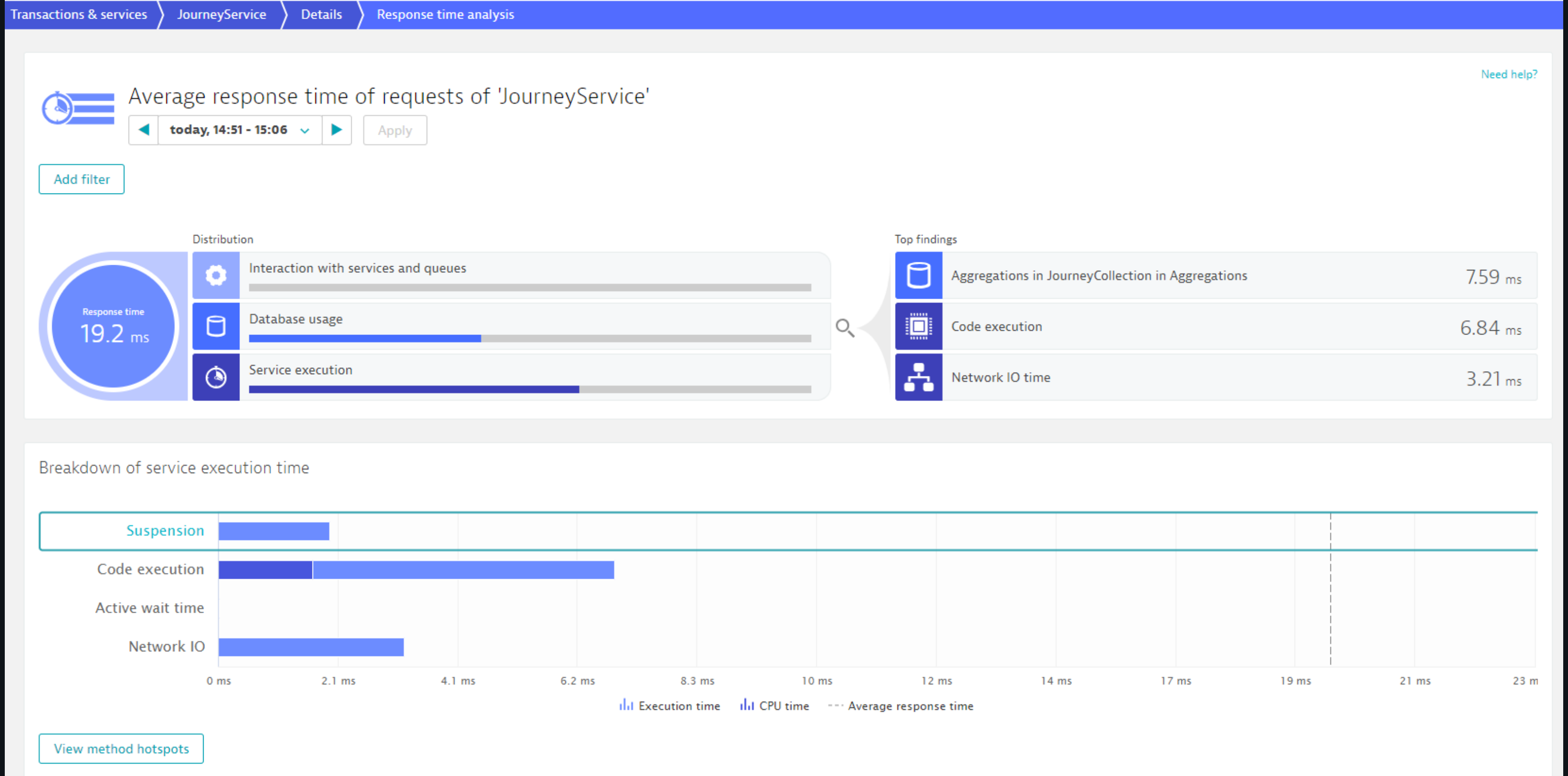
Response Time Hotspots

Failure Analysis

Exception Analysis

PurePaths

Response Time Hotspots



Transactions & services > JourneyService > Details > Response time analysis

Average response time of requests of 'JourneyService'

today, 14:51 - 15:06 | Apply

Add filter

Distribution

Response time 19.2 ms

- Interaction with services and queues
- Database usage
- Service execution

Top findings

Aggregations in JourneyCollection in Aggregations	7.59 ms
Code execution	6.84 ms
Network IO time	3.21 ms

Breakdown of service execution time

Response Time Hotspots

- What is it?
  - A feature that allows you to breakdown time spent in any service or even individual requests
- When would I use it?
  - Performance analysis of any instrumented service
  - Understand total impact of code, DB queries, calls to other services
  - Analyzing performance degradation during problems related to a service or request

Analyze Transactions

View Web Requests

Response Time Distribution

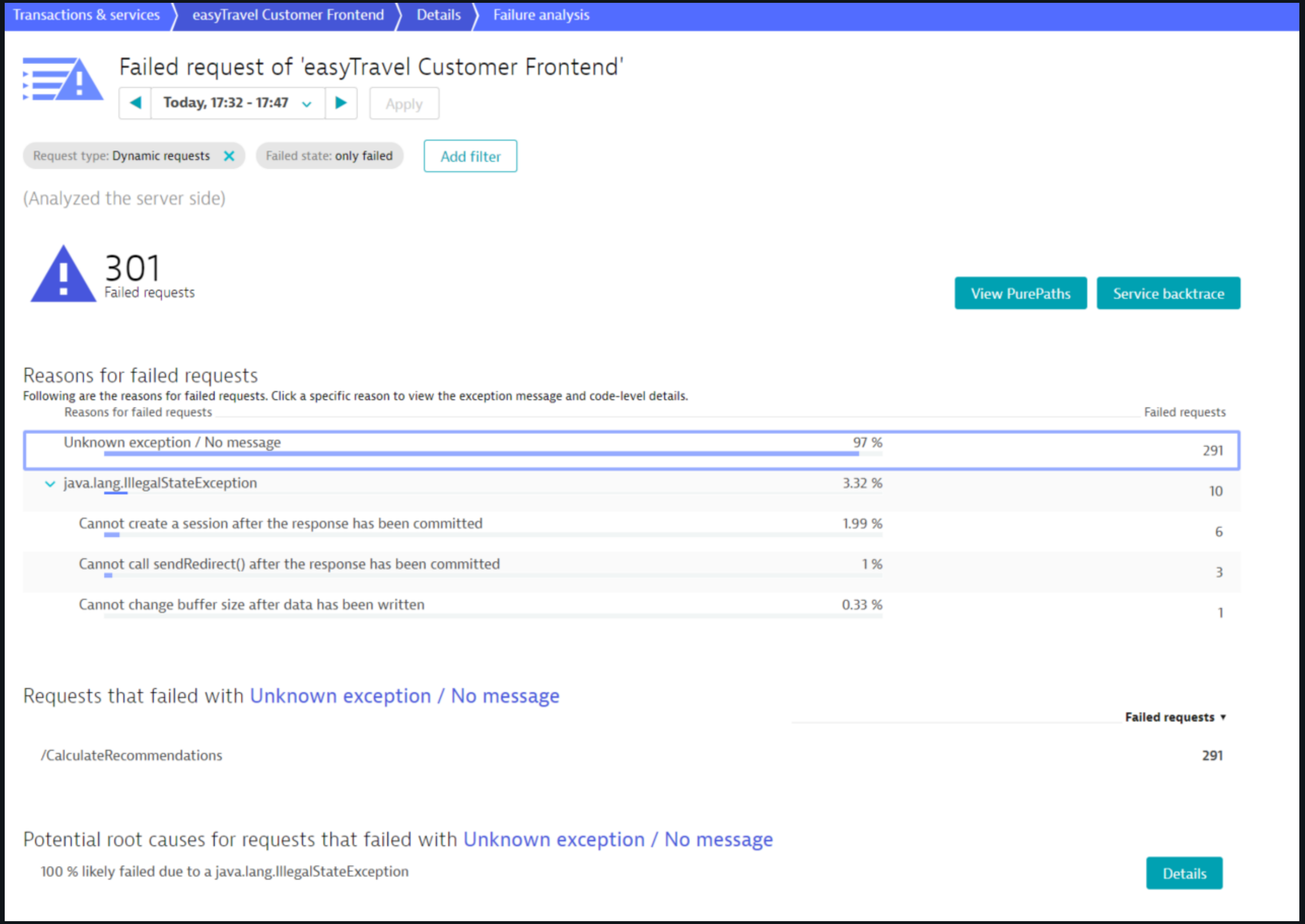
Response Time Hotspots

Failure Analysis

Exception Analysis

PurePaths

Failure Analysis



Failed request of 'easyTravel Customer Frontend'

Today, 17:32 - 17:47

Request type: Dynamic requests ✕

Failed state: only failed

(Analyzed the server side)

301
Failed requests

Reasons for failed requests

Following are the reasons for failed requests. Click a specific reason to view the exception message and code-level details.

Reasons for failed requests		Failed requests
Unknown exception / No message	97 %	291
java.lang.IllegalStateException	3.32 %	10
Cannot create a session after the response has been committed	1.99 %	6
Cannot call sendRedirect() after the response has been committed	1 %	3
Cannot change buffer size after data has been written	0.33 %	1

Requests that failed with Unknown exception / No message

	Failed requests ▾
/CalculateRecommendations	291

Potential root causes for requests that failed with Unknown exception / No message

100 % likely failed due to a java.lang.IllegalStateException

Analyze Transactions

View Web Requests

Response Time Distribution

Response Time Hotspots

Failure Analysis ◀

Exception Analysis

PurePaths

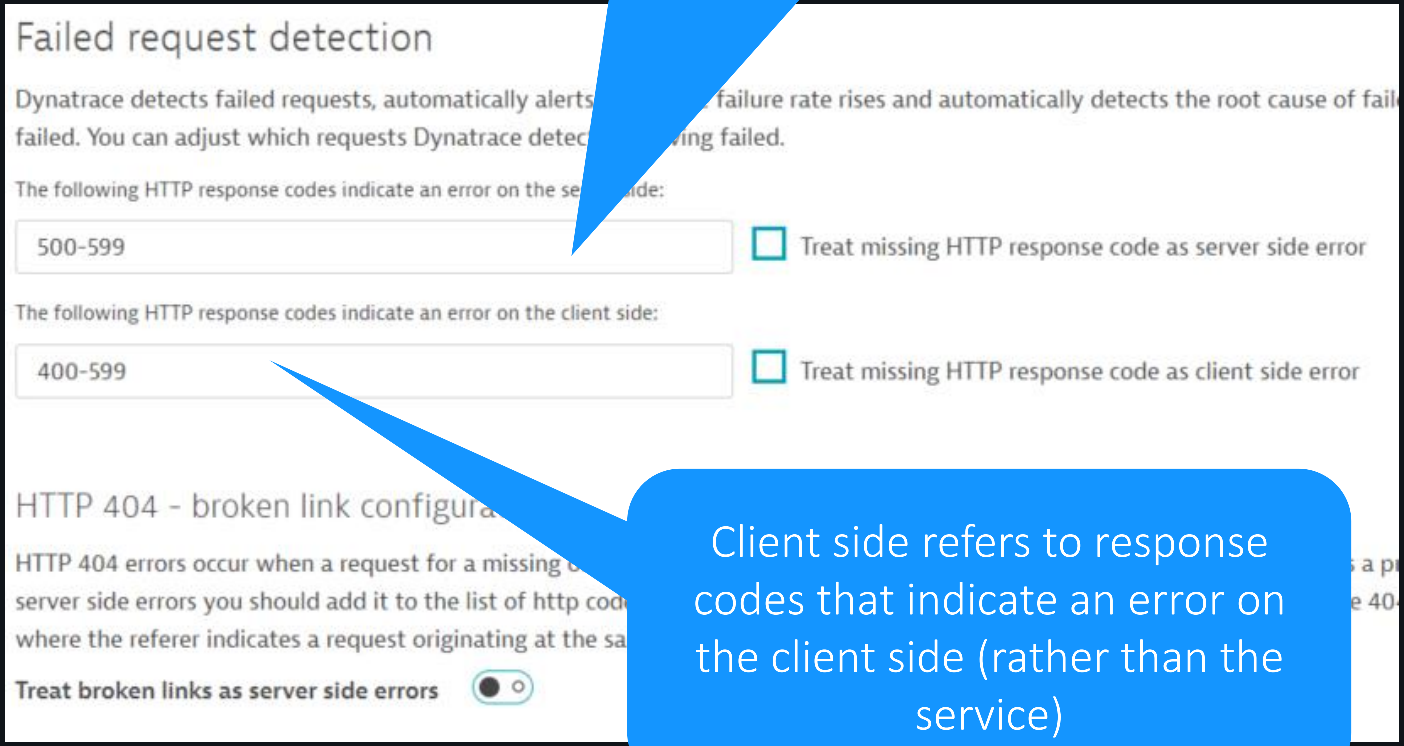
Failure Analysis

- What is it?
  - View the details of failures occurring at any instrumented tier
- When would I use it?
  - Ad hoc failure analysis of any instrumented service
  - Analyzing failure rate increase during problems related to a service or request

What counts as a Failure?

Custom Error detection

- Dynatrace automatically captures HTTP errors
- It also detects programming exceptions as the reason for failed requests when the exceptions result in the abort of service calls
- Many web containers provide error pages for handled exceptions, which are also detected



Failed request detection

Dynatrace detects failed requests, automatically alerts failure rate rises and automatically detects the root cause of fail failed. You can adjust which requests Dynatrace detec ving failed.

The following HTTP response codes indicate an error on the se ide:

500-599 □ Treat missing HTTP response code as server side error

The following HTTP response codes indicate an error on the client side:

400-599 □ Treat missing HTTP response code as client side error

HTTP 404 - broken link configura

HTTP 404 errors occur when a request for a missing server side errors you should add it to the list of http code where the referer indicates a request originating at the sa

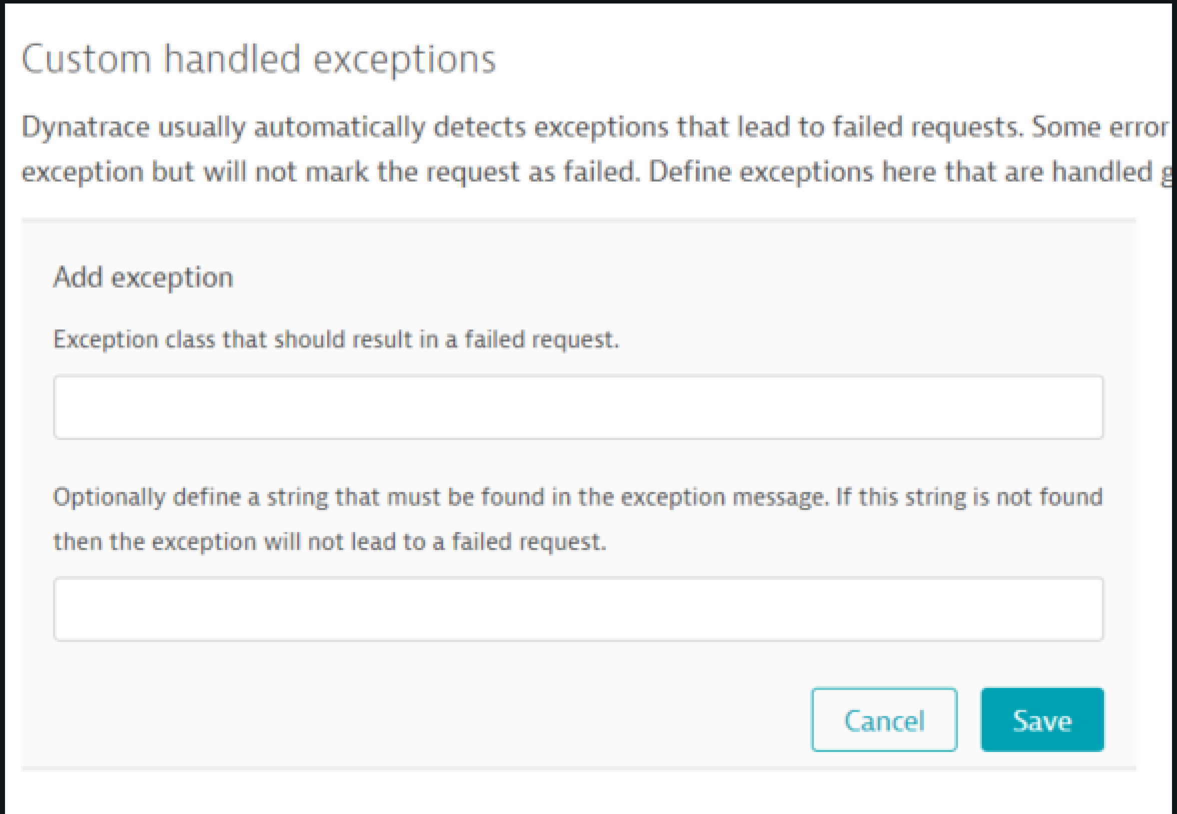
Treat broken links as server side errors ●○

Server side refers to response codes that indicate an error with the service itself

Client side refers to response codes that indicate an error on the client side (rather than the service)

Custom Exceptions

- Beyond this, however, there are situations where application code handles exceptions gracefully in a manner that isn't detected
- When this happens, Dynatrace doesn't detect failed requests or alert you to errors
- Dynatrace can find the defined exception (and optional defined exception message) on any request and mark it as a failure



Custom handled exceptions

Dynatrace usually automatically detects exceptions that lead to failed requests. Some error exception but will not mark the request as failed. Define exceptions here that are handled g

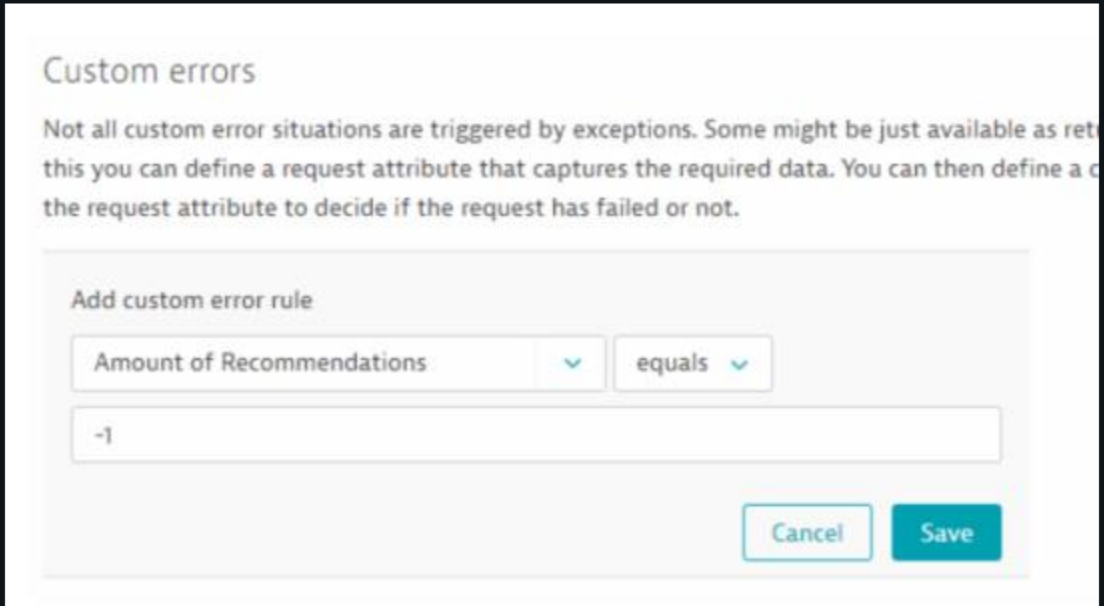
Add exception

Exception class that should result in a failed request.

Optionally define a string that must be found in the exception message. If this string is not found then the exception will not lead to a failed request.

Custom Errors

- There are many cases where requests fail for reasons that are related to business logic
- While such situations often aren’t detectable via exceptions or HTTP response codes, they are nevertheless indicative of problems
- To handle these situations, Dynatrace now allows you to use request attributes as indicators for error situations



Custom errors

Not all custom error situations are triggered by exceptions. Some might be just available as ret
this you can define a request attribute that captures the required data. You can then define a c
the request attribute to decide if the request has failed or not.

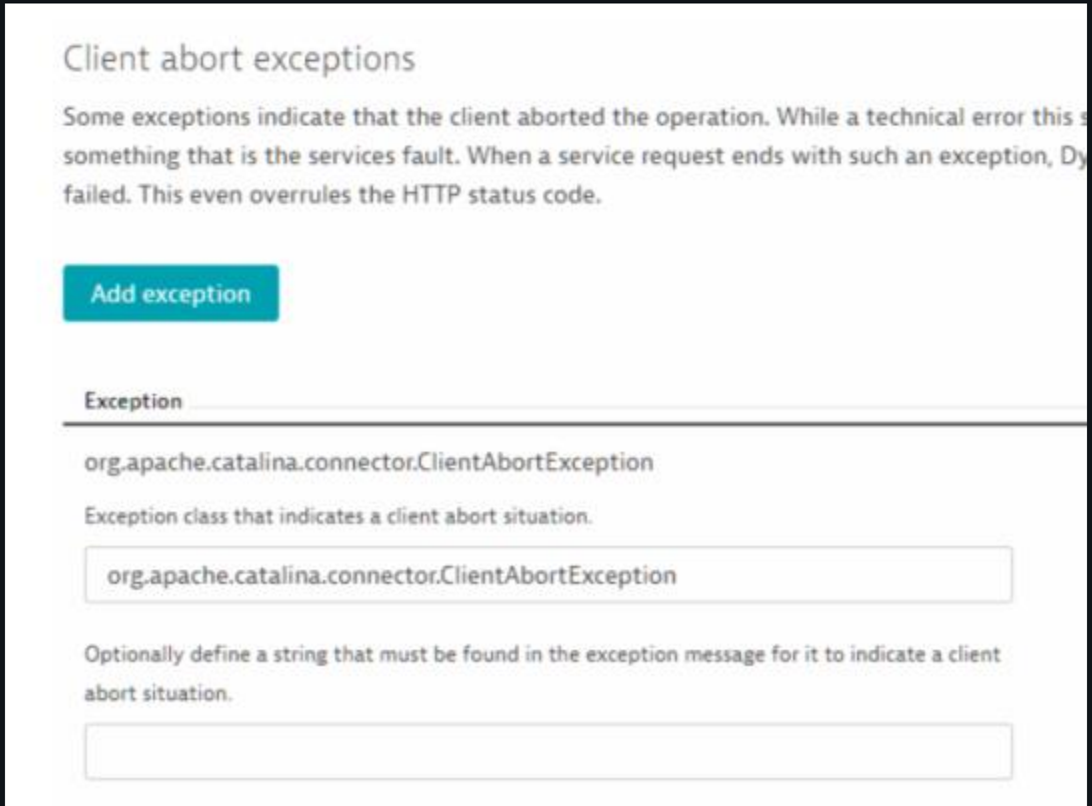
Add custom error rule

Amount of Recommendations | equals

-1

Client Abort Exceptions

- There are exceptions that indicate a call was aborted and as such shouldn't be considered as failed under any circumstances



Client abort exceptions

Some exceptions indicate that the client aborted the operation. While a technical error this s
something that is the services fault. When a service request ends with such an exception, Dy
failed. This even overrules the HTTP status code.

org.apache.catalina.connector.ClientAbortException

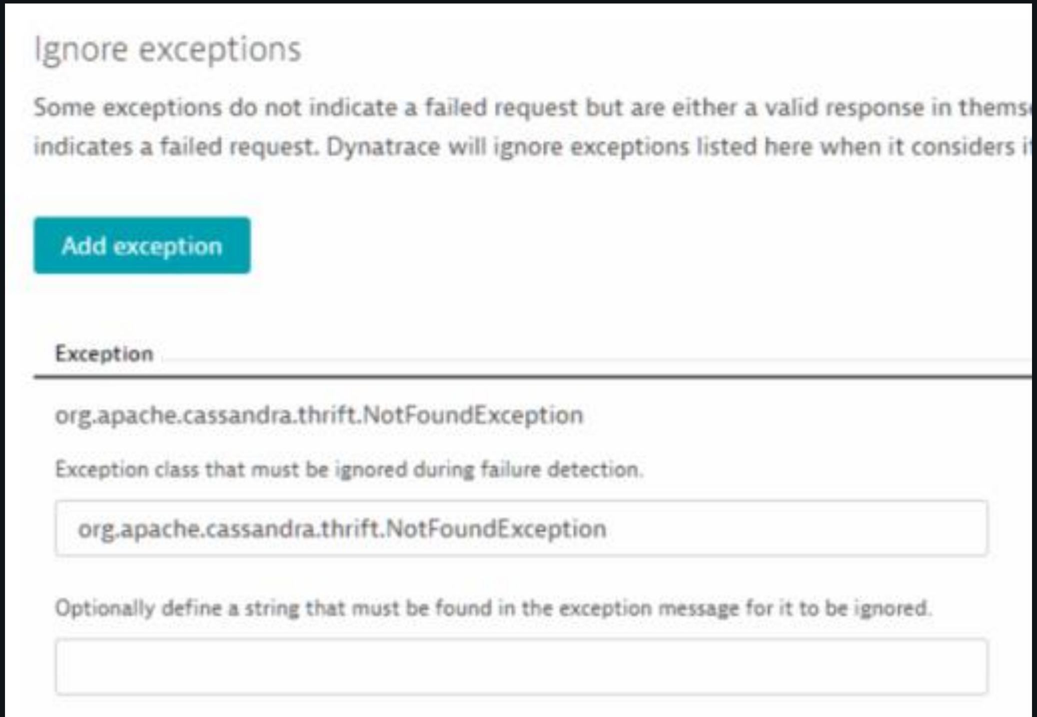
Exception class that indicates a client abort situation.

org.apache.catalina.connector.ClientAbortException

Optionally define a string that must be found in the exception message for it to indicate a client abort situation.

Ignore Exceptions

- In a perfect world, every request that triggers an exception would be considered a failed request.
- There are however cases where your code returns exceptions that indicate a certain response and not an error



Ignore exceptions

Some exceptions do not indicate a failed request but are either a valid response in thems
indicates a failed request. Dynatrace will ignore exceptions listed here when it considers i

Exception

org.apache.cassandra.thrift.NotFoundException

Exception class that must be ignored during failure detection.

org.apache.cassandra.thrift.NotFoundException

Optionally define a string that must be found in the exception message for it to be ignored.

Analyze Transactions

View Web Requests

Response Time Distribution

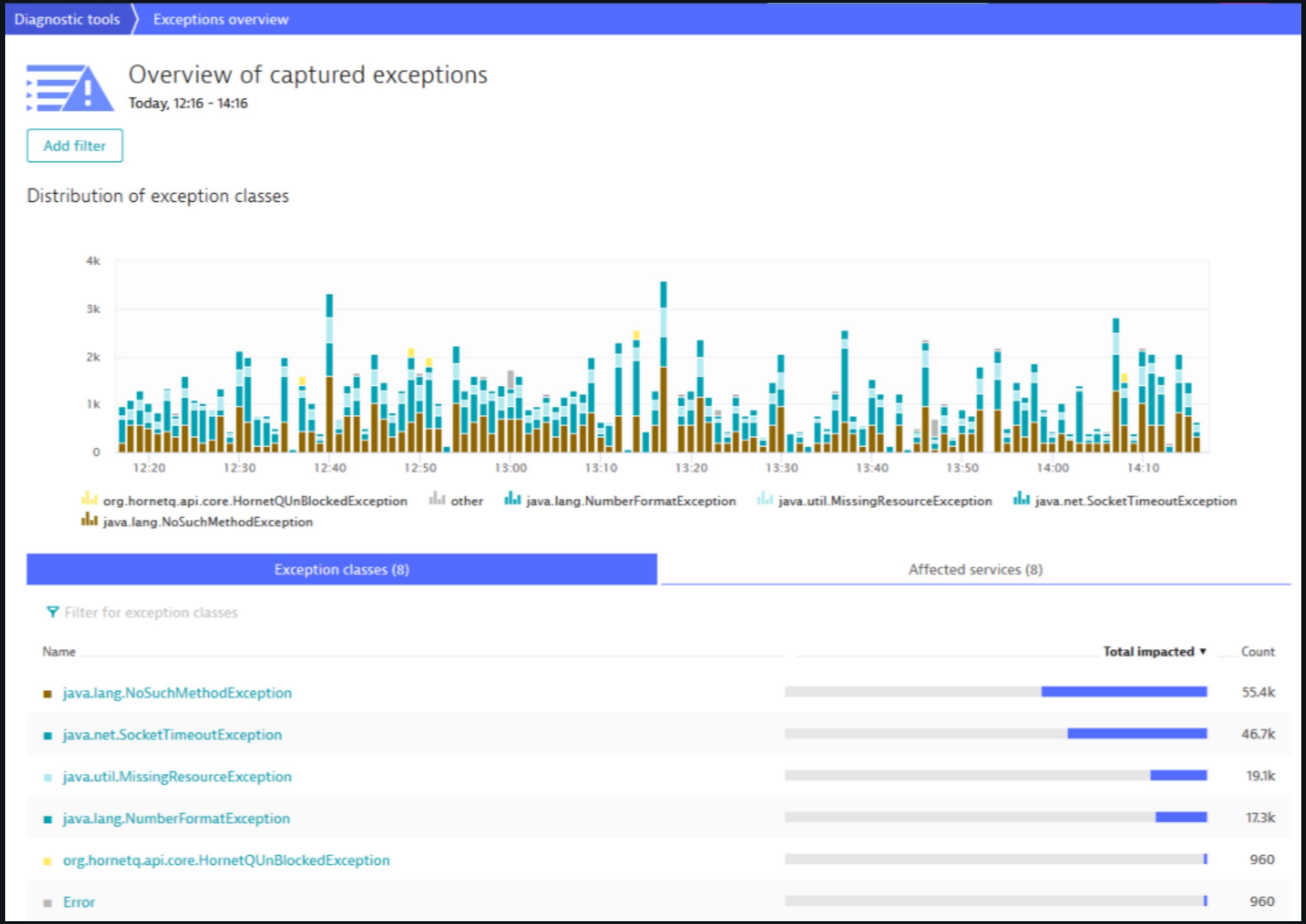
Response Time Hotspots

Failure Analysis

Exception Analysis

PurePaths

Exception Analysis



Diagnostic tools
Exceptions overview
Add filter
4k
3k
2k
1k
0
12:20
12:30
12:40
12:50
13:00
13:10
13:20
13:30
13:40
13:50
14:00
14:10
org.hornetq.api.core.HornetQUnBlockedException
other
java.lang.NumberFormatException
java.util.MissingResourceException
java.net.SocketTimeoutException
java.lang.NoSuchMethodException
Exception classes (8)
Affected services (8)
Filter for exception classes

Overview of captured exceptions

Today, 12:16 - 14:16

Distribution of exception classes

Name	Total impacted ▾	Count
java.lang.NoSuchMethodException		55.4k
java.net.SocketTimeoutException		46.7k
java.util.MissingResourceException		19.1k
java.lang.NumberFormatException		17.3k
org.hornetq.api.core.HornetQUnBlockedException		960
Error		960

Analyze Transactions

View Web Requests

Response Time Distribution

Response Time Hotspots

Failure Analysis

Exception Analysis ◀

PurePaths

Exception Analysis

- What is it?
  - View the details of exceptions that are happening in your application and how it changes over time
- When would I use it?
  - Identify the most commonly triggered exception
  - Filter down to the services that contain exceptions
  - Drilldown into the stacktrace of a particular exception

Analyze Transactions

View Web Requests

Response Time Distribution

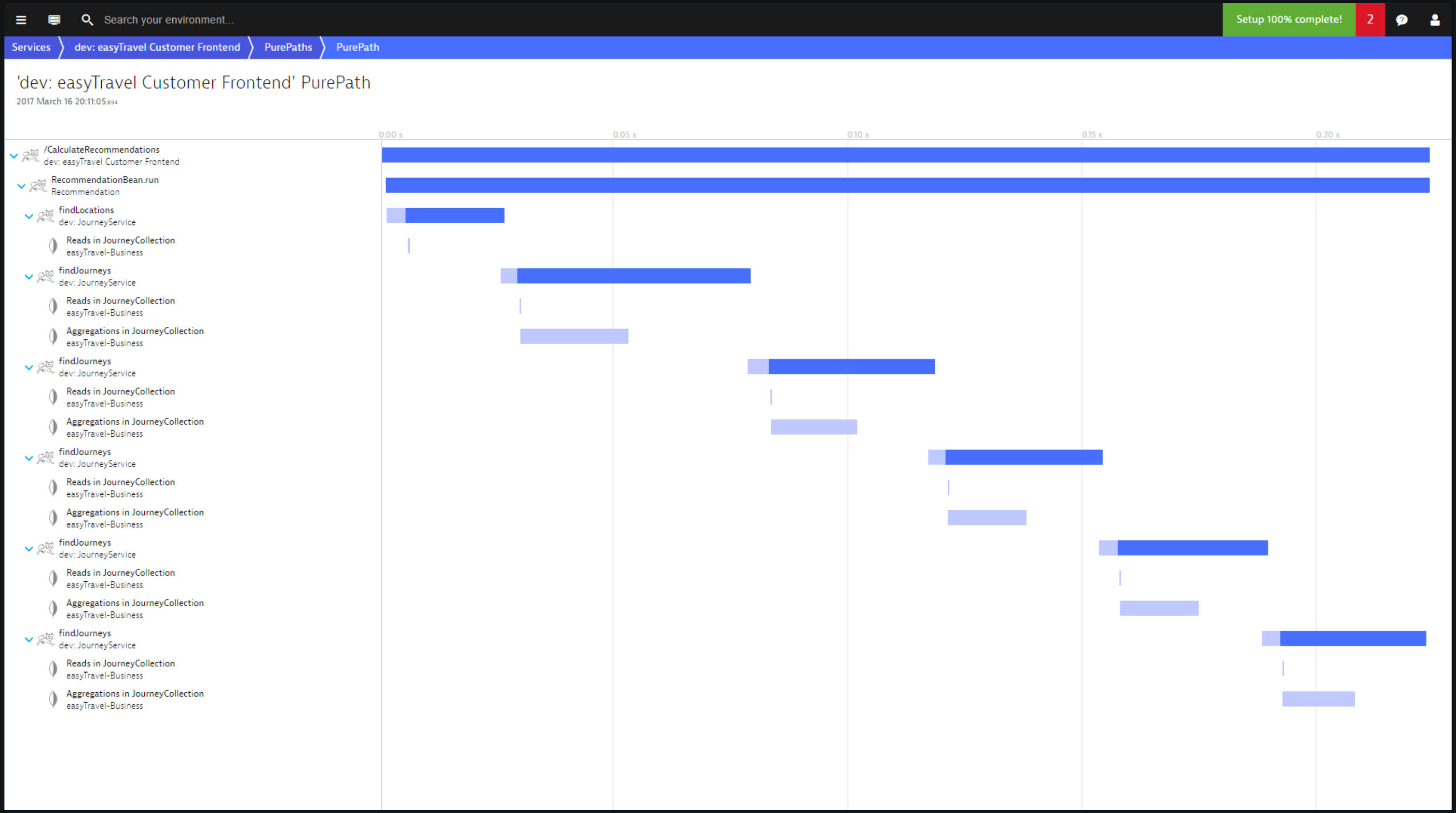
Response Time Hotspots

Failure Analysis

Exception Analysis

PurePaths

PurePaths



'dev: easyTravel Customer Frontend' PurePath
2017 March 16 20:11:05.894

- /CalculateRecommendations
  dev: easyTravel Customer Frontend
  - RecommendationBean.run
    Recommendation
    - findLocations
      dev: JourneyService
      - Reads in JourneyCollection
        easyTravel-Business
    - findJourneys
      dev: JourneyService
      - Reads in JourneyCollection
        easyTravel-Business
      - Aggregations in JourneyCollection
        easyTravel-Business
    - findJourneys
      dev: JourneyService
      - Reads in JourneyCollection
        easyTravel-Business
      - Aggregations in JourneyCollection
        easyTravel-Business
    - findJourneys
      dev: JourneyService
      - Reads in JourneyCollection
        easyTravel-Business
      - Aggregations in JourneyCollection
        easyTravel-Business
    - findJourneys
      dev: JourneyService
      - Reads in JourneyCollection
        easyTravel-Business
      - Aggregations in JourneyCollection
        easyTravel-Business
    - findJourneys
      dev: JourneyService
      - Reads in JourneyCollection
        easyTravel-Business
      - Aggregations in JourneyCollection
        easyTravel-Business

Analyze Transactions

View Web Requests

Response Time Distribution

Response Time Hotspots

Failure Analysis

Exception Analysis

PurePaths ◀

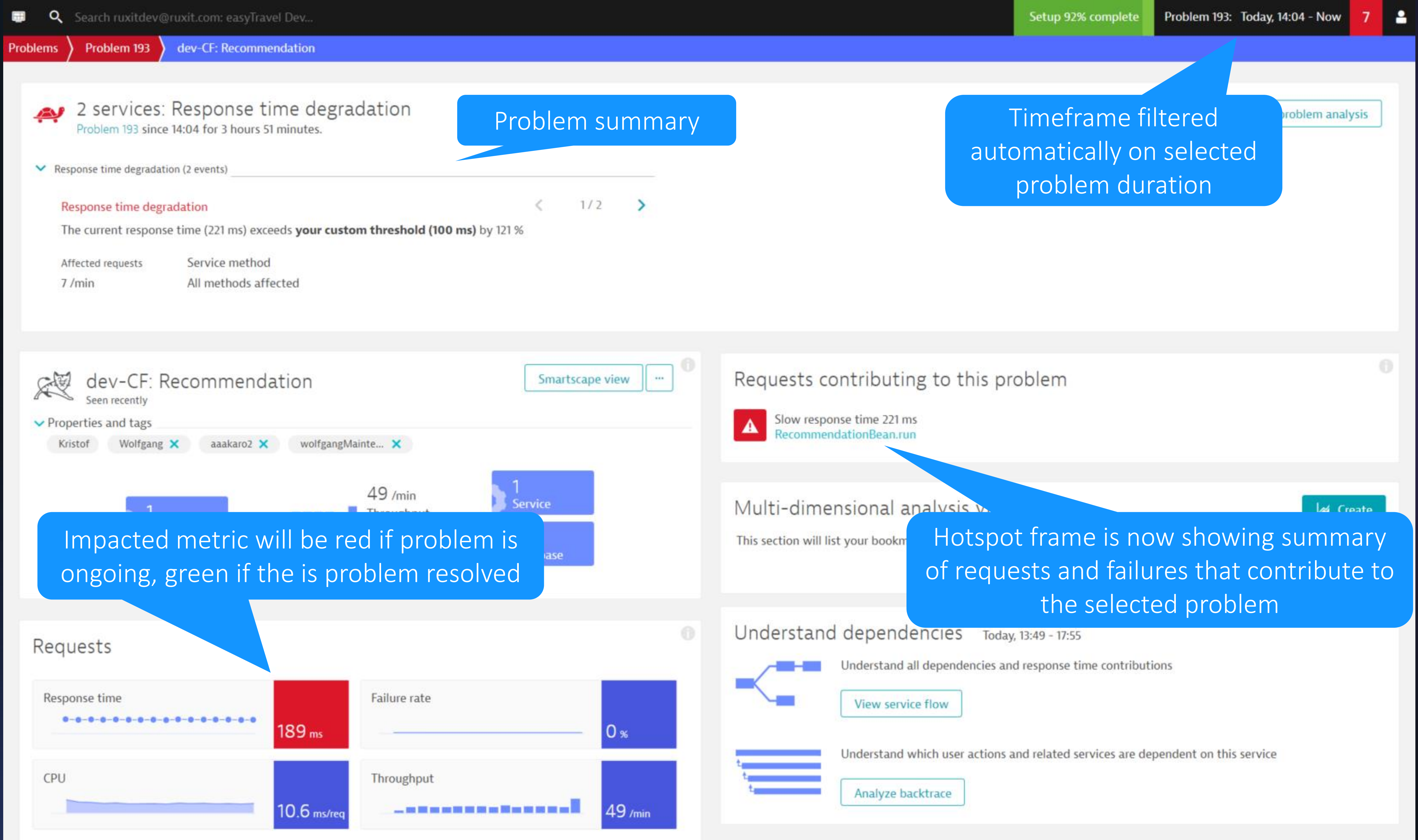
PurePaths

- What is it?
  - Deep dive breakdown of a single transaction.
  - Waterfall breakdown of where time is spent

Service Analysis during a Problem

Service Analysis – Normal vs Abnormal

- To know what is ‘abnormal’, we must first understand what is ‘normal’
- A few of the previous views dynamically change based on detected abnormalities
  - Service Details
  - Response Time Hotspots
  - Failure Analysis



Search ruxitdev@ruxit.com: easyTravel Dev...
Setup 92% complete
Problem 193: Today, 14:04 - Now
7
Problems
Problem 193
dev-CF: Recommendation

2 services: Response time degradation

Problem 193 since 14:04 for 3 hours 51 minutes.

Response time degradation (2 events)

Response time degradation

The current response time (221 ms) exceeds **your custom threshold (100 ms)** by 121 %

Affected requests	Service method
7 /min	All methods affected

dev-CF: Recommendation

Seen recently

Requests

Response time	189 ms
Failure rate	0 %
CPU	10.6 ms/req
Throughput	49 /min

Requests contributing to this problem

Slow response time 221 ms
RecommendationBean.run

Multi-dimensional analysis v

This section will list your bookm

Understand dependencies Today, 13:49 - 17:55

Understand all dependencies and response time contributions

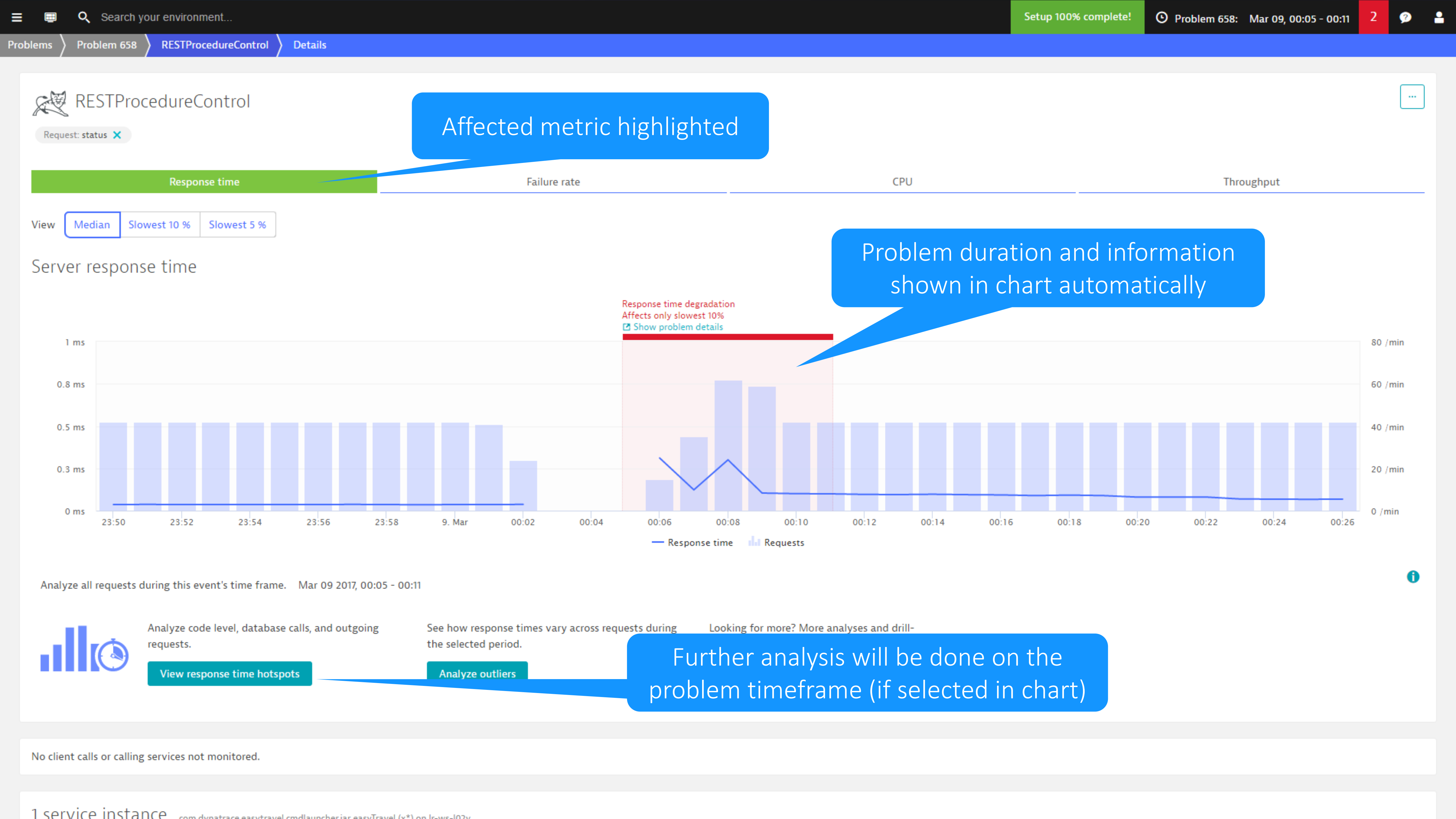
Understand which user actions and related services are dependent on this service

Problem summary

Timeframe filtered automatically on selected problem duration

Impacted metric will be red if problem is ongoing, green if the is problem resolved

Hotspot frame is now showing summary of requests and failures that contribute to the selected problem



RESTProcedureControl

Request: status ✕

Affected metric highlighted

Server response time

Problem duration and information shown in chart automatically

Analyze all requests during this event's time frame. Mar 09 2017, 00:05 - 00:11

Analyze code level, database calls, and outgoing requests.

View response time hotspots

See how response times vary across requests during the selected period.

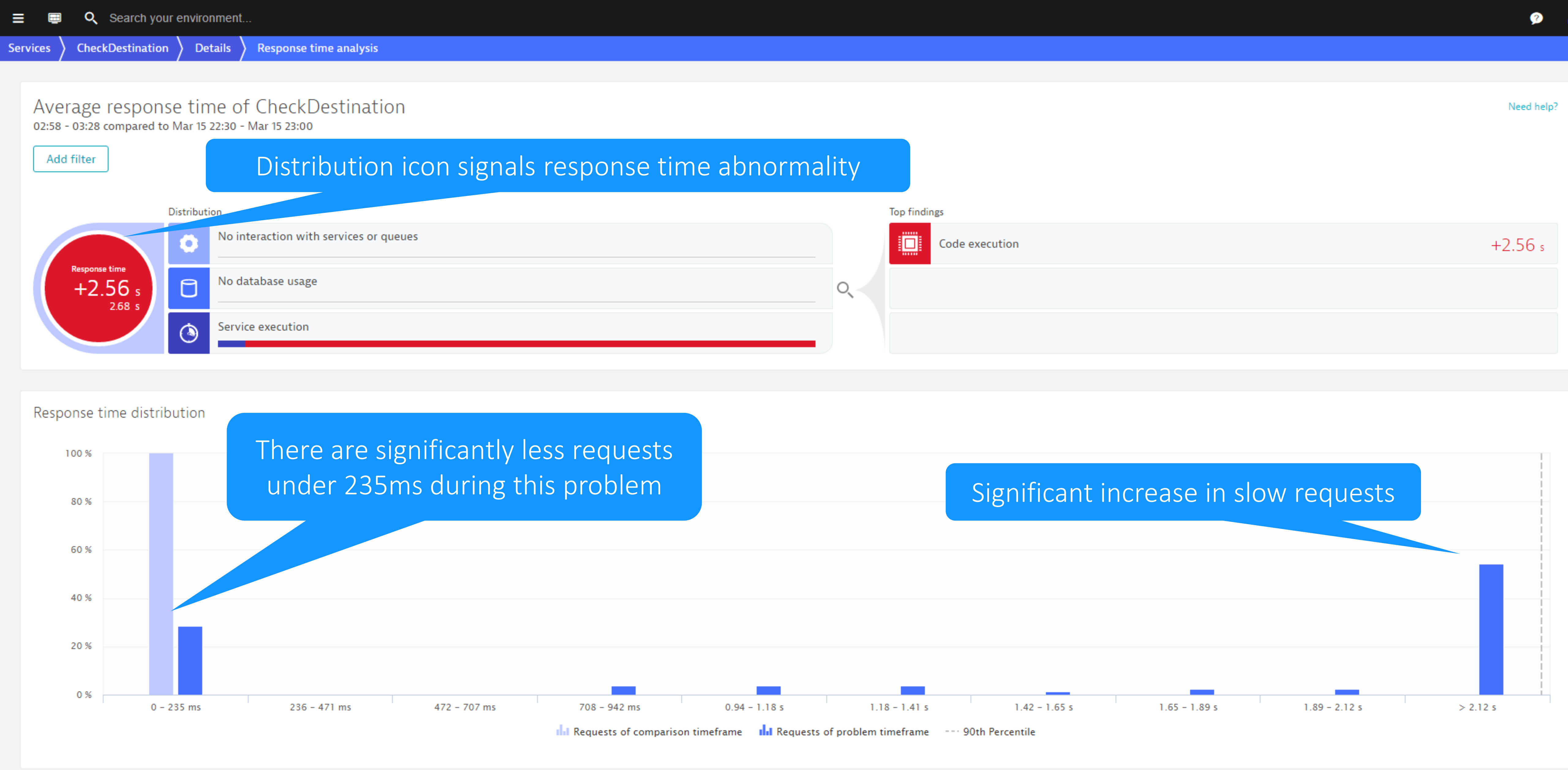
Analyze outliers

Looking for more? More analyses and drill-

Further analysis will be done on the problem timeframe (if selected in chart)

No client calls or calling services not monitored.

1 service instance com.dynatrace.easytravel.cmdlauncher.jar easyTravel (x*) on lr-ws-l02v



Average response time of CheckDestination

02:58 - 03:28 compared to Mar 15 22:30 - Mar 15 23:00

Distribution icon signals response time abnormality

No interaction with services or queues

No database usage

Service execution

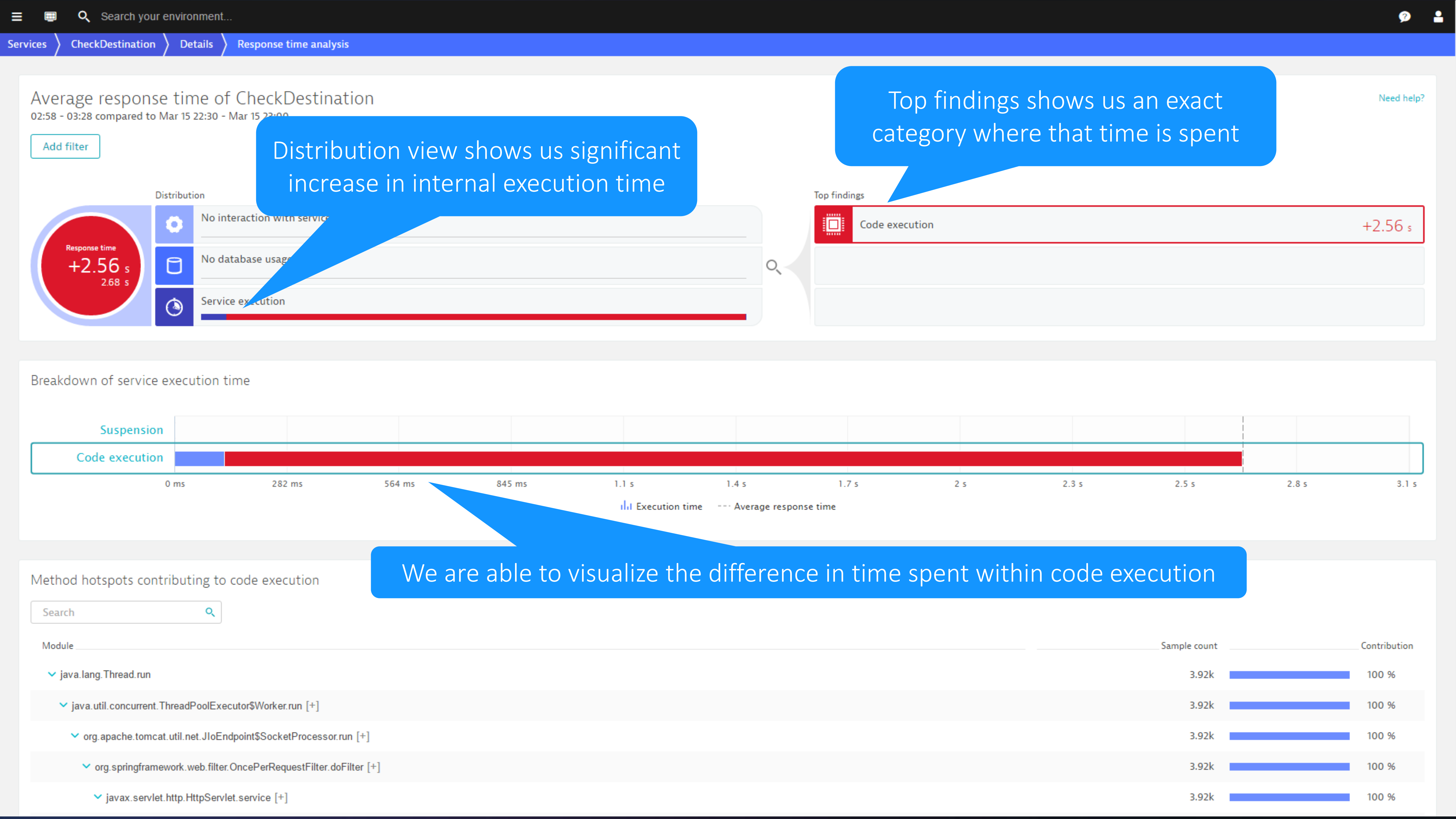
Top findings

Code execution	+2.56 s

Response time distribution

There are significantly less requests under 235ms during this problem

Significant increase in slow requests



Average response time of CheckDestination

02:58 - 03:28 compared to Mar 15 22:30 - Mar 15 23:00

Add filter

Need help?

Distribution

- No interaction with servic
- No database usag
- Service execution

Top findings

Code execution	+2.56 s

Distribution view shows us significant increase in internal execution time

Top findings shows us an exact category where that time is spent

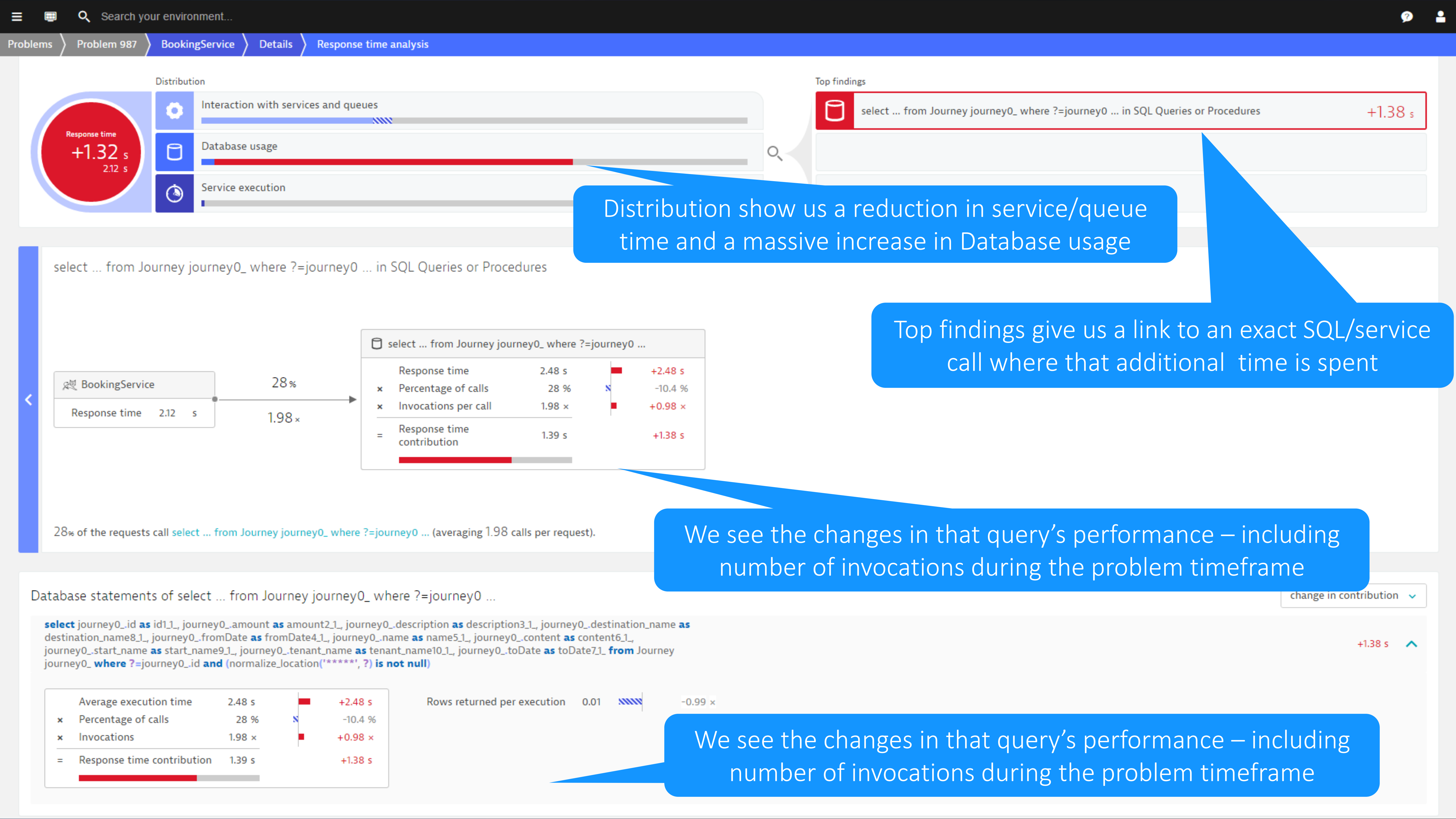
Breakdown of service execution time

We are able to visualize the difference in time spent within code execution

Method hotspots contributing to code execution

Search

Module	Sample count	Contribution
java.lang.Thread.run	3.92k	100 %
java.util.concurrent.ThreadPoolExecutor$Worker.run [+]	3.92k	100 %
org.apache.tomcat.util.net.JIoEndpoint$SocketProcessor.run [+]	3.92k	100 %
org.springframework.web.filter.OncePerRequestFilter.doFilter [+]	3.92k	100 %
javax.servlet.http.HttpServlet.service [+]	3.92k	100 %



Distribution

Response time +1.32 s 2.12 s

- Interaction with services and queues
- Database usage
- Service execution

Top findings

select ... from Journey journey0_ where ?=journey0 ... in SQL Queries or Procedures +1.38 s

Distribution show us a reduction in service/queue time and a massive increase in Database usage

Top findings give us a link to an exact SQL/service call where that additional time is spent

select ... from Journey journey0_ where ?=journey0 ... in SQL Queries or Procedures

BookingService

Response time 2.12 s

28 %

1.98 ×

select ... from Journey journey0_ where ?=journey0 ...

	Response time	2.48 s	+2.48 s
×	Percentage of calls	28 %	-10.4 %
×	Invocations per call	1.98 ×	+0.98 ×
=	Response time contribution	1.39 s	+1.38 s

28% of the requests call select ... from Journey journey0_ where ?=journey0 ... (averaging 1.98 calls per request).

We see the changes in that query's performance – including number of invocations during the problem timeframe

Database statements of select ... from Journey journey0_ where ?=journey0 ...

change in contribution

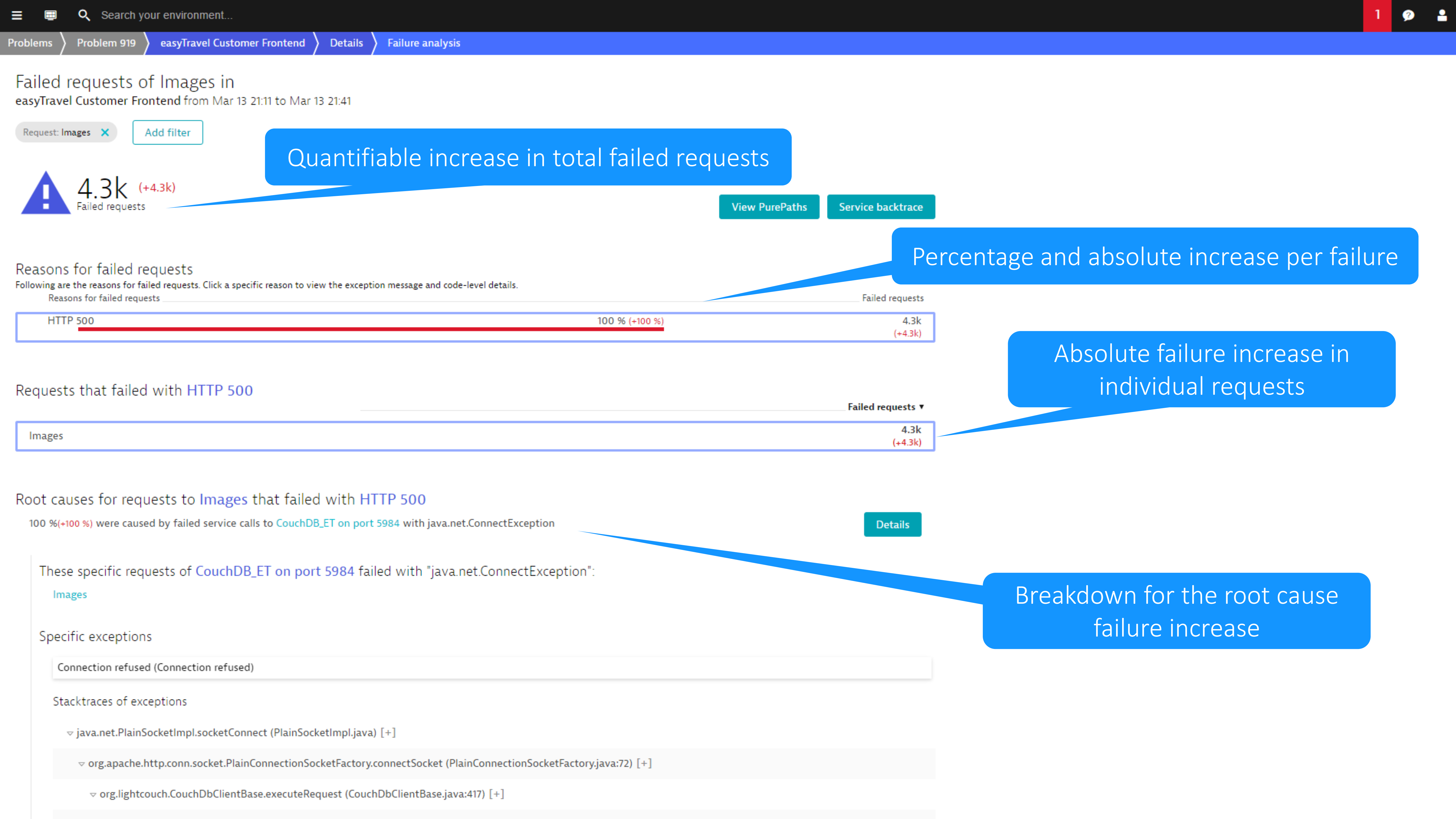
```
select journey0_.id as id1_1_, journey0_.amount as amount2_1_, journey0_.description as description3_1_, journey0_.destination_name as
destination_name8_1_, journey0_.fromDate as fromDate4_1_, journey0_.name as name5_1_, journey0_.content as content6_1_,
journey0_.start_name as start_name9_1_, journey0_.tenant_name as tenant_name10_1_, journey0_.toDate as toDate7_1_ from Journey
journey0_ where ?=journey0_.id and (normalize_location('*****', ?) is not null)
```

+1.38 s

	Average execution time	2.48 s	+2.48 s
×	Percentage of calls	28 %	-10.4 %
×	Invocations	1.98 ×	+0.98 ×
=	Response time contribution	1.39 s	+1.38 s

Rows returned per execution 0.01 -0.99 ×

We see the changes in that query's performance – including number of invocations during the problem timeframe



Failed requests of Images in

easyTravel Customer Frontend from Mar 13 21:11 to Mar 13 21:41

Request: Images ✕ Add filter

4.3k (+4.3k)
Failed requests

View PurePaths Service backtrace

Reasons for failed requests

Following are the reasons for failed requests. Click a specific reason to view the exception message and code-level details.

Reasons for failed requests		Failed requests
HTTP 500	100 % (+100 %)	4.3k (+4.3k)

Requests that failed with HTTP 500

	Failed requests ▾
Images	4.3k (+4.3k)

Root causes for requests to Images that failed with HTTP 500

100 %(+100 %) were caused by failed service calls to CouchDB_ET on port 5984 with java.net.ConnectException

Details

These specific requests of CouchDB_ET on port 5984 failed with "java.net.ConnectException":

Images

Specific exceptions

Connection refused (Connection refused)

Stacktraces of exceptions

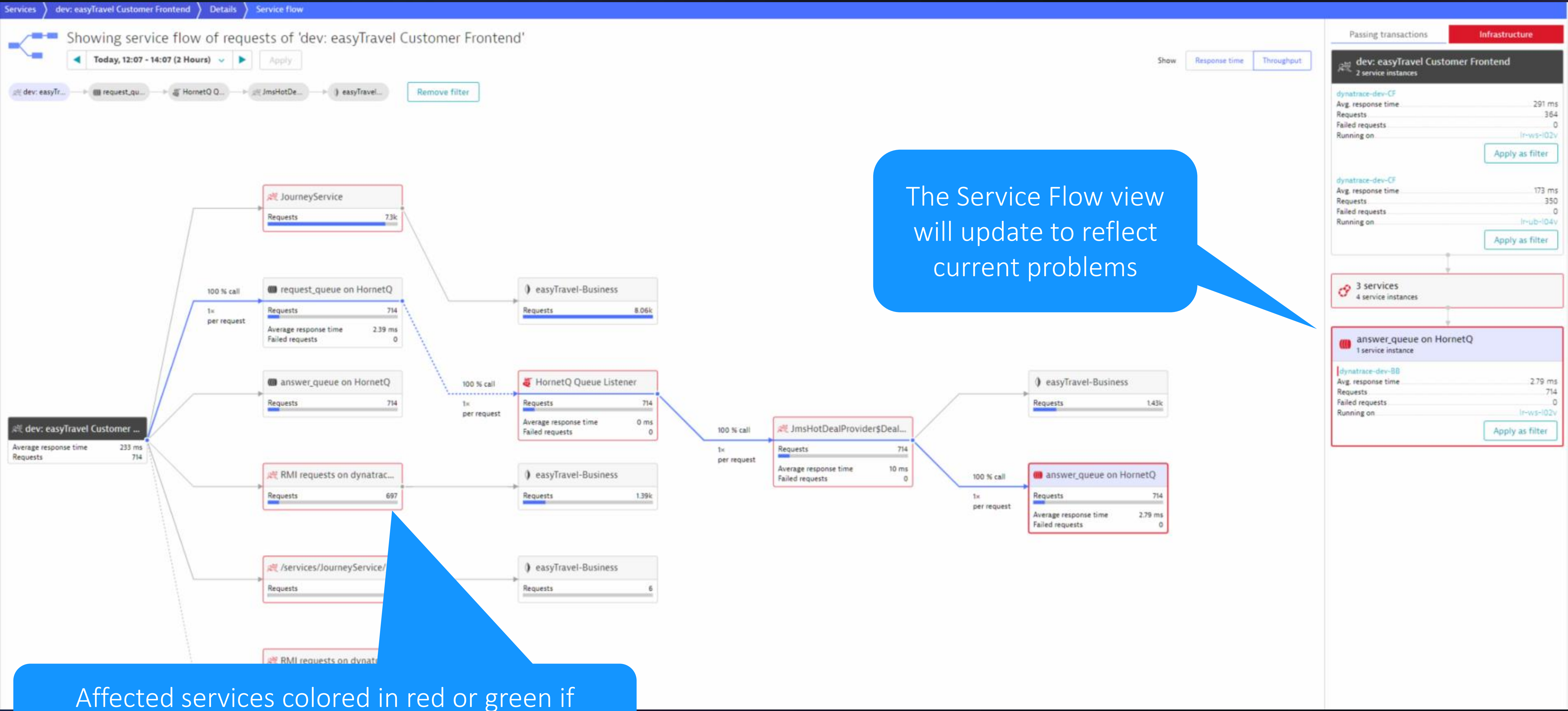
```
▿ java.net.PlainSocketImpl.socketConnect (PlainSocketImpl.java) [+]
  ▿ org.apache.http.conn.socket.PlainConnectionSocketFactory.connectSocket (PlainConnectionSocketFactory.java:72) [+]
    ▿ org.lightcouch.CouchDbClientBase.executeRequest (CouchDbClientBase.java:417) [+]
```

Quantifiable increase in total failed requests

Percentage and absolute increase per failure

Absolute failure increase in individual requests

Breakdown for the root cause failure increase



The Service Flow view will update to reflect current problems

Affected services colored in red or green if viewing within problem analysis mode.

Questions?

dynatrace

Simply smarter clouds