

Fearless from Monolith to Serverless with Dynatrace

George Mao

Specialist Solutions Architect, Serverless, Amazon Web Services

Wayne Segar

Solutions Architect, Dynatrace

198249

© 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved.

From Monolith to Serverless with Dynatrace

What is Serverless?

Evolution of Serverless Computing

From Monolith to Serverless with Dynatrace

What is Serverless?

Build and run applications
without thinking about servers



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

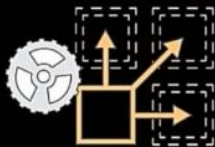
Challenges with Servers



Operations and management



Scaling

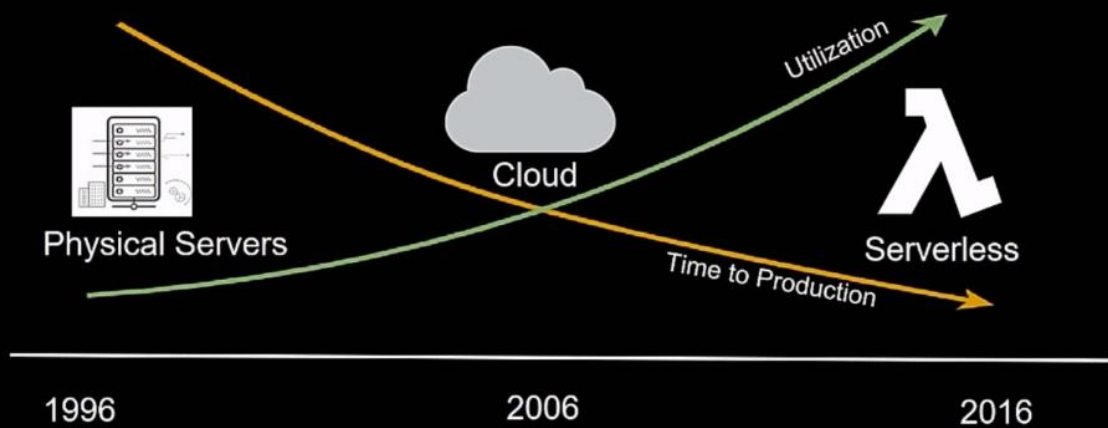


Provisioning and utilization



Availability and fault tolerance

Serverless Revolution



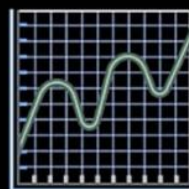
AWS Lambda: Serverless Compute



- AWS Lambda lets you run code without managing servers.
- **Don't Pay for Idle time:** You only pay for compute you consume



No Servers to Manage



Continuous Scaling



Subsecond Metering

How it Works

EVENT SOURCE



Changes in data state



Requests to endpoints



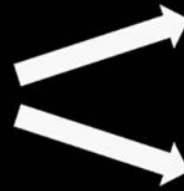
Changes in resource state



FUNCTION



Node.js
Python
Java
C#
Go



SERVICES (ANYTHING)



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.


Splitting
Monoliths
Into
Components

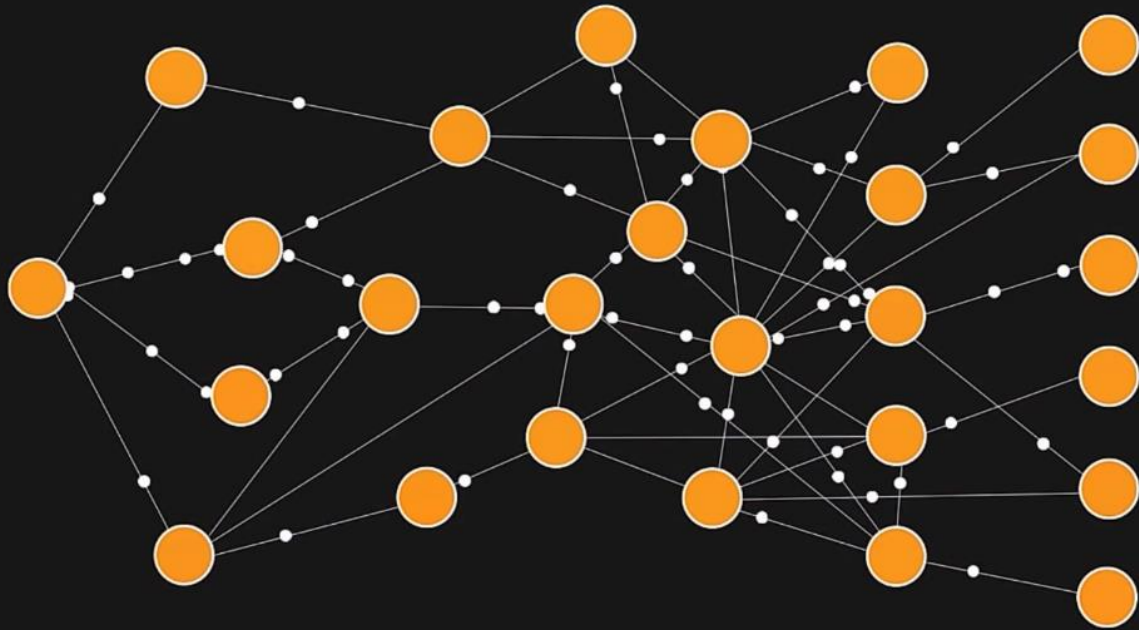



Courtesy of AWS

Into Services



Courtesy of 



Courtesy of 

AWS Serverless

Leveraging standard service and platform capabilities




Courtesy of 

AWS Serverless


Leveraging standard service and platform capabilities



Courtesy of 

Microservices to Functions



Courtesy of 


Microservices for message delivery



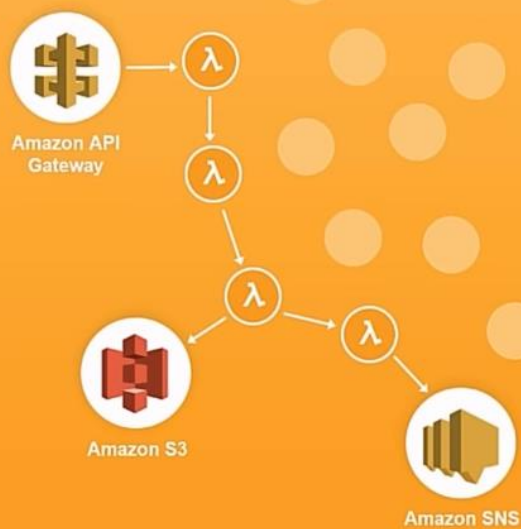
Courtesy of 

Microservices for data persistence



Courtesy of  aws

Microservices for workflow management



Courtesy of  aws

Microservices On Lambda Are Ephemeral



Amazon API Gateway



When the system is idle, it shuts down and costs nothing to run



Amazon SQS



Amazon DynamoDB




Amazon Kinesis



Amazon S3



Amazon SNS

Courtesy of 

Who is dynatrace



Advanced
Technology
Partner

DevOps Competency
Migration Competency
Public Sector Partner
Marketplace Seller
SaaS Partner

- Dynatrace is at the forefront of AI-based monitoring platform
- 8 years a leader in Gartner's MQ for Application Performance Monitoring Suites, 5th consecutive year leader in terms of market share*
- Runs on the AWS Cloud, monitoring up to 100,000+ hosts
- 72 of the Fortune 100, trust Dynatrace incl. Verizon, Citrix
- AWS Competency Partner and available on AWS Marketplace
- Dynatrace digital assistant davis using Amazon Technology

*Gartner: Market Share Analysis: Performance Analysis Software, Worldwide, 2016



© 2018, Amazon Web Services, Inc. or Its Affiliates. All rights reserved.

Dynatrace Journey! It shaped our product!

More Agile

~120

Code commits / day

340

Stories per sprint

26

Feature Releases / Year

More Quality

31000

Unit & Int Tests / hour

60h

UI Tests per Build

93%

Production bugs found by Dev

More Stability

450

Global EC2 Instances

99.998%

Global Availability

500

Deployments / Day



Questions / Challenges on that “Migration” Journey

From
On-Premise Monolith



- Where to start?
- What to break out?
- Dependencies?

Via
Containerized Services



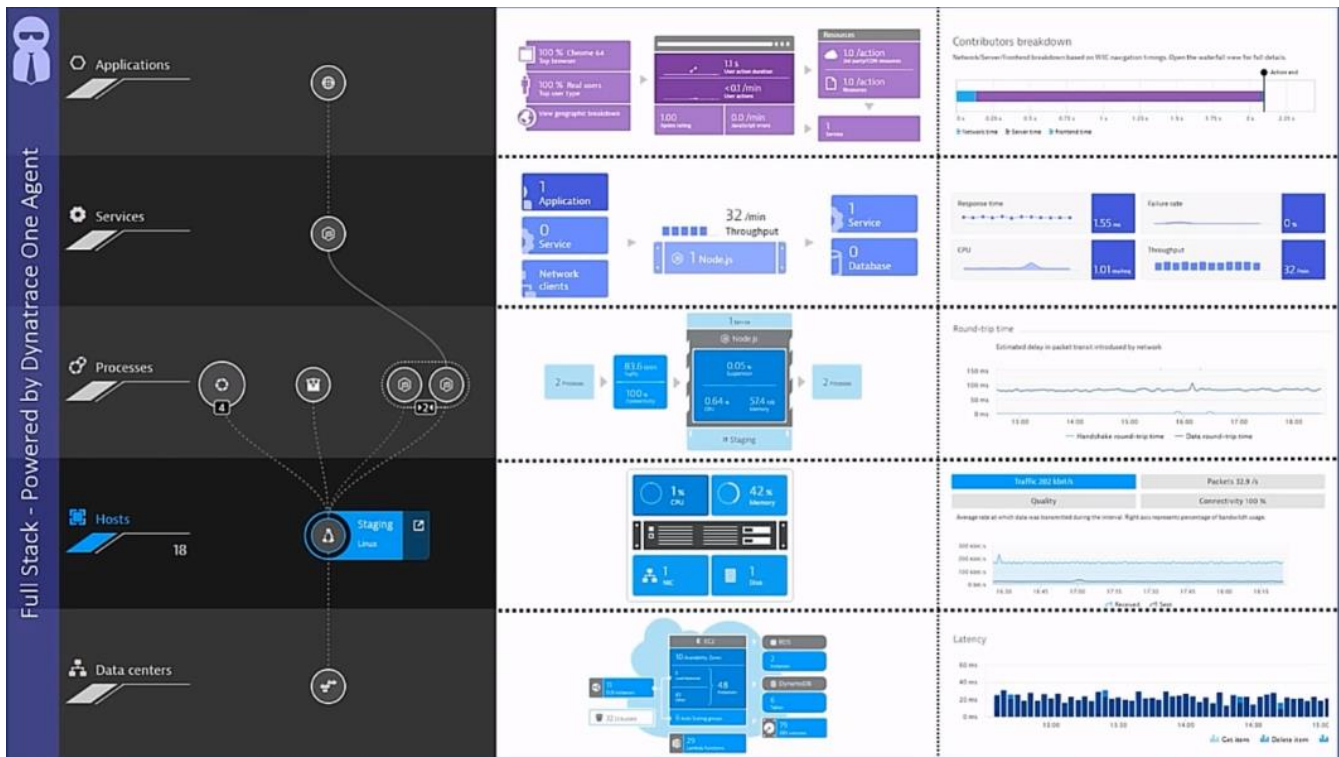
- Works as expected?
- Users happy?
- Does it scale?
- Does it perform?

To
Functions & Cloud Services



- Works as expected?
- Users happy?
- How to optimize?
- How to automate?





Dynatrace enables Break, Shift, Re-Platform

Dynatrace Smartscape

We tell you **WHAT** to migrate and **WHERE** to break the monolith!

Dynatrace AI

Automatic End User, Service and Infrastructure Root Cause Analysis

Dynatrace Automation API

Automate Rollout in your Containers & Functions. Automate Shift-Left and Self-Healing



Where to start?
What to break out?
Dependencies?



Works as expected?
Does it scale?
Does it perform?

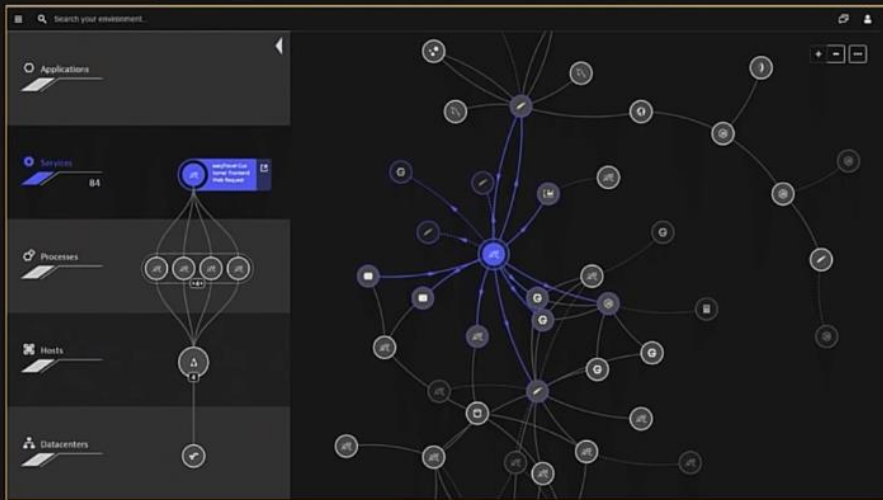


Works as expected?
How to scale? How to optimize?
How to automate?



Dynatrace Helps to Decide *What* to Migrate/Shift

Automate T-Shirt Size Categorization for your Migration Plan

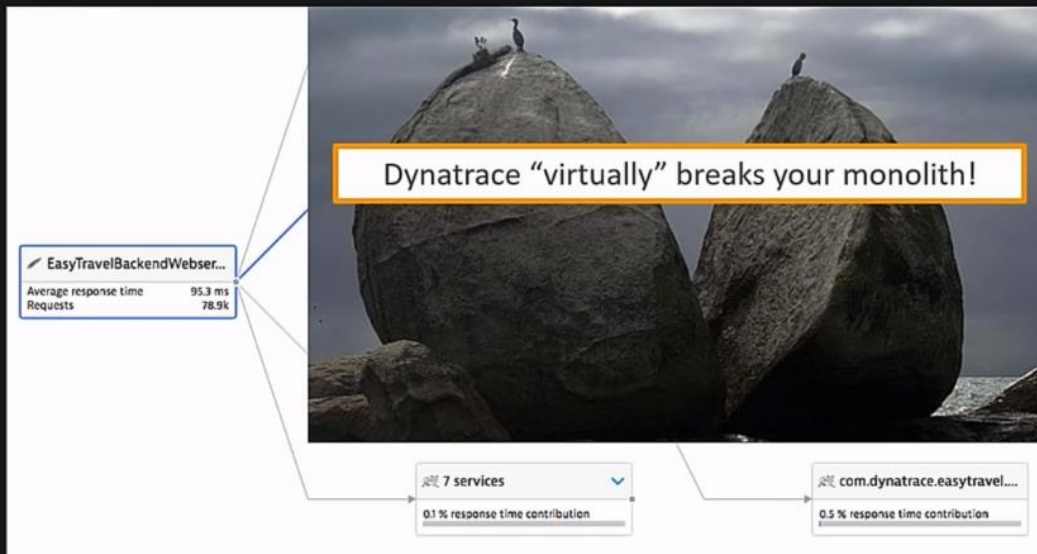


Dynatrace Smartscope

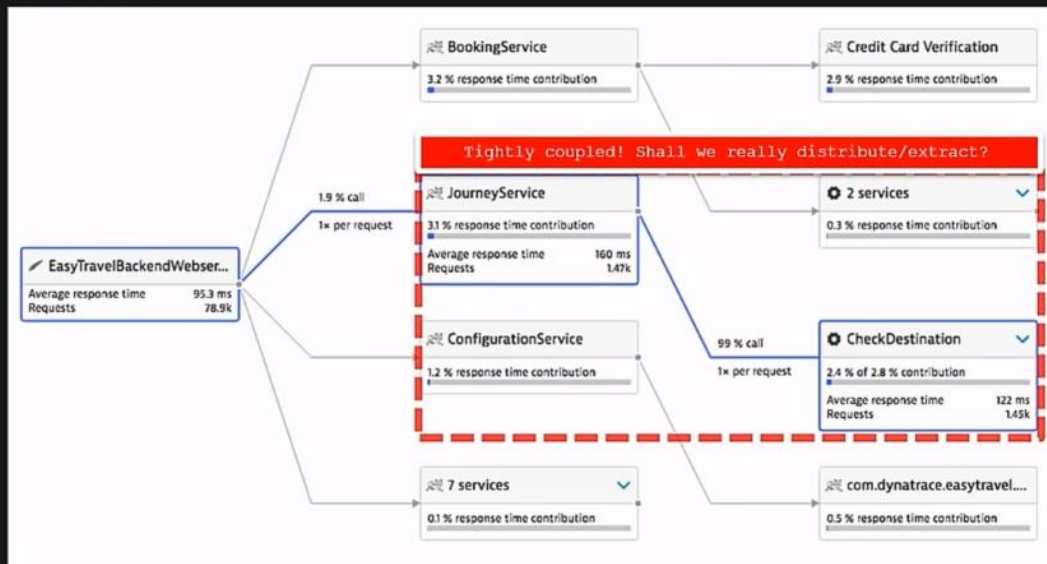
- **Auto-dependency** mapping of internal & external services
- **Auto-load** detection and baselining
- **Auto-resource** consumption detection



Dynatrace helps to decide **WHERE** to break the monolith



Dynatrace helps to decide WHERE to break the monolith



Getting to know the Monolith Architecture

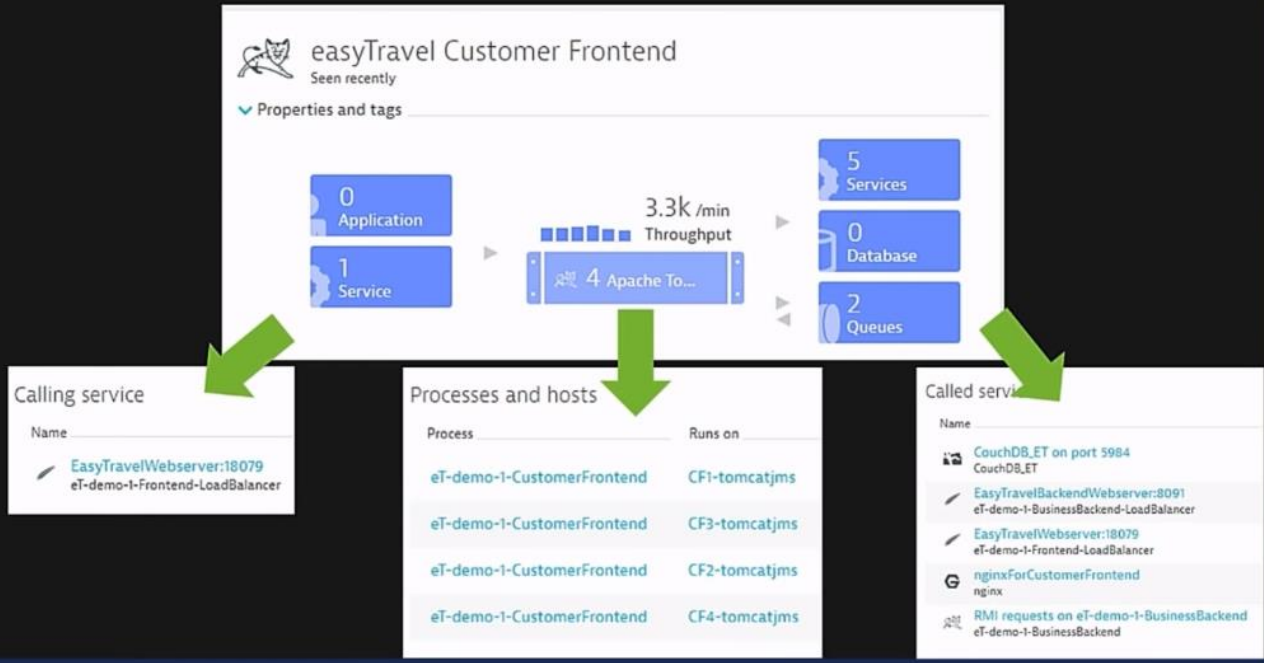
Service
Endpoints

Service Flow

Depending
Services



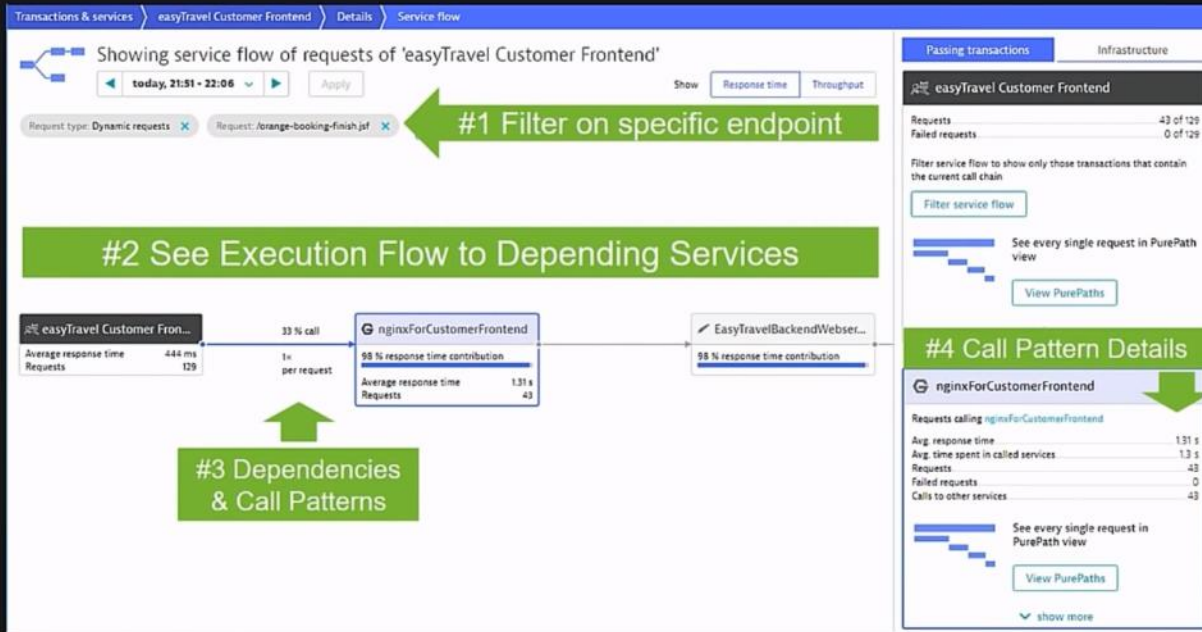
Step #1 Dependency Information



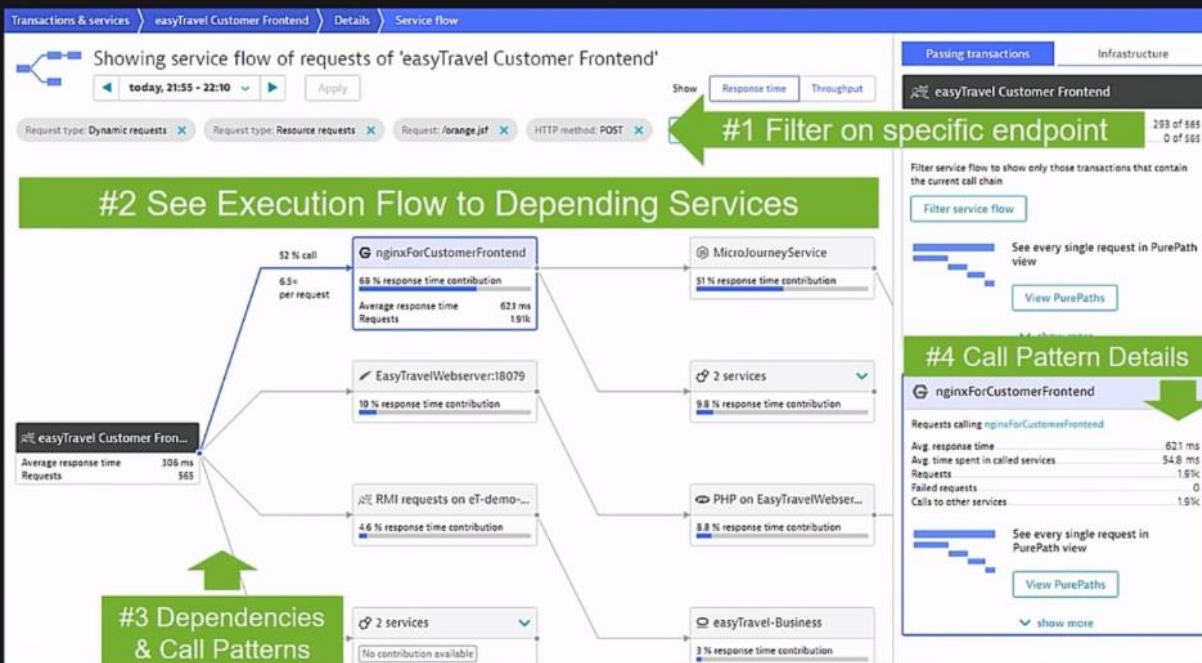
Step #2 Detecting Service Endpoints

Name	Contribution	Throughput	CPU
/orange-booking-review.jsf easyTravel Journ... easyTravel User		21.6 /min	11.4 ms/req
/orange-booking-payment.jsf easyTravel Journ...		15.1 /min	6.78 ms/req
/orange-booking-finish.jsf - Key request easyTravel Journ... easyTravel Desti...		11.9 /min	5.51 ms/req
/orange-trip-details.jsf easyTravel Journ...		5.77 /min	8.14 ms/req
/legal-orange-mobile.jsf		1.5 /min	0.74 ms/req

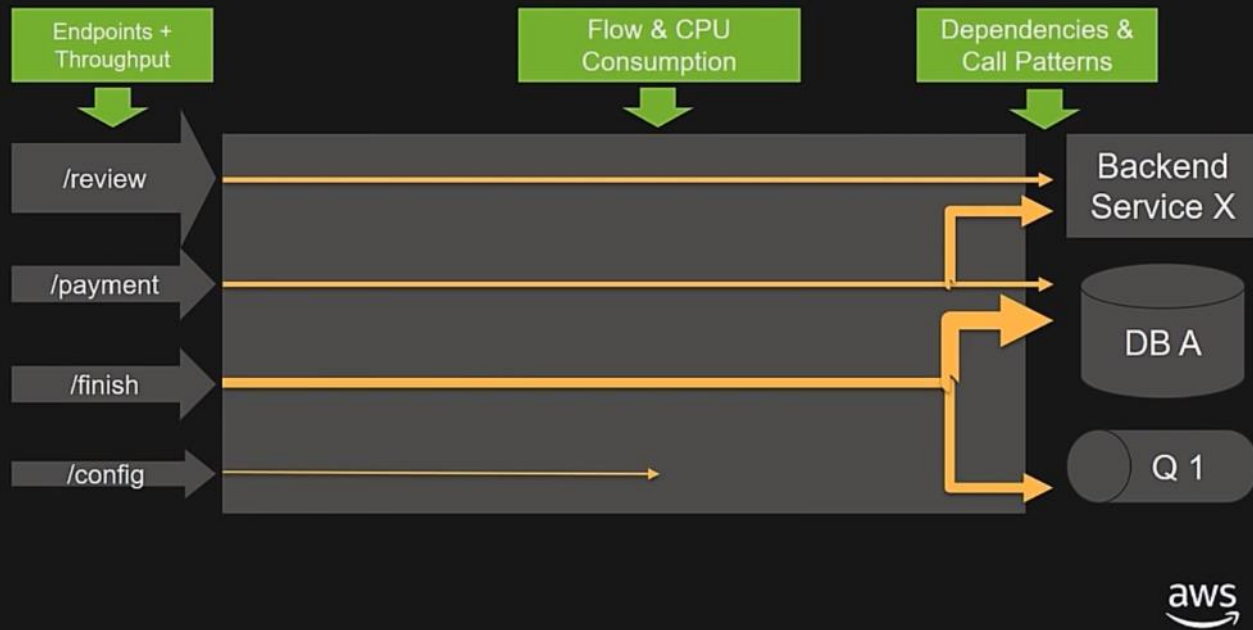
Step #3 Understanding Service Flow per Endpoint



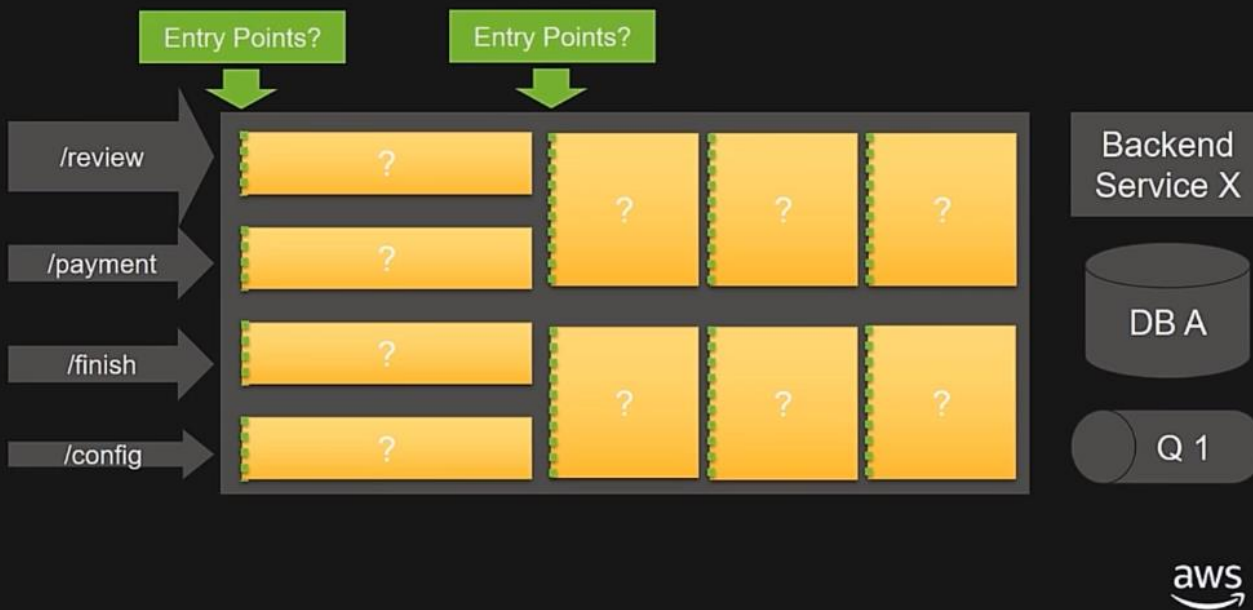
Step #3 Understanding Service Flow per Endpoint



What We Learned So Far!



Next: Where to Break the Monolith?



Step 4: Finding Entry Points with CPU Sampling

Transactions & services > easyTravel Customer Frontend > Details > Hotspots

Request type: Dynamic requests ✕ Request: /orange-booking-finish.jsf ✕ **#1 Filter on specific endpoint**

easy **#2 Search for specific code**

Method	Sample count	Contribution
java.lang.Thread.run	21	100 %
java.util.concurrent.ThreadPoolExecutor\$Worker.run [+]	21	100 %
org.apache.tomcat.util.net.JIoEndpoint\$SocketProcessor.run [+]	21	100 %
javax.faces.webapp.FacesServlet.service	21	100 %
com.sun.faces.lifecycle.LifecycleImpl.render [+]	15	71.4 %
com.sun.faces.lifecycle.LifecycleImpl.execute	6	28.6 %
com.dynatrace.easytravel.frontend.beans.BookingBean.performBooking [-] #3 Found Entry Point to Package		
com.dynatrace.easytravel.frontend.data.DataProvider.storeBooking	2	9.52 %
com.dynatrace.easytravel.business.client.BookingServiceStub.storeBooking	2	9.52 %

aws

Step #5 Define Custom Service Entry Points

Settings > Server-side service monitoring > Custom service detection > Define custom Java service

Settings

Monitoring
Setup and overview
Monitoring overview
Monitored technologies

Process groups
Detection and naming

Web and mobile monitoring
Global settings and configuration

Cloud and virtualization
Connect vCenter, OpenStack or AWS

Server-side service monitoring
Manage & customize service monitoring

Custom service detection
Merged service monitoring
Service naming rules
Request attributes

Log analytics
Setup log-based events and log storage

Anomaly detection
Configure detection sensitivity

Alerting
Configure alerting settings

Integration

Define custom Java service Save Cancel

Name your custom service **#1 Service Name**
BookingService

Entry points
Entry points for your custom service can be methods of a specific class or implementations of an interface. Each non-recursive call to this method will represent a single request to your custom service. [More...](#)

Optionally restrict custom service rules by process groups

☐ This service is used inside a busy loop to process dequeued messages of a queue
Dynatrace automatically detects event-based message queue handlers such as JMS MessageListener or RabbitMQ DefaultConsumers. If your code doesn't use such a message queue handler [More...](#)

Find another entry point Define entry point manually

Active	Entry point	Methods	Delete	Edit
<input checked="" type="checkbox"/>	com.dynatrace.easytravel.frontend.beans.BookingBean	performBooking #2 Method		

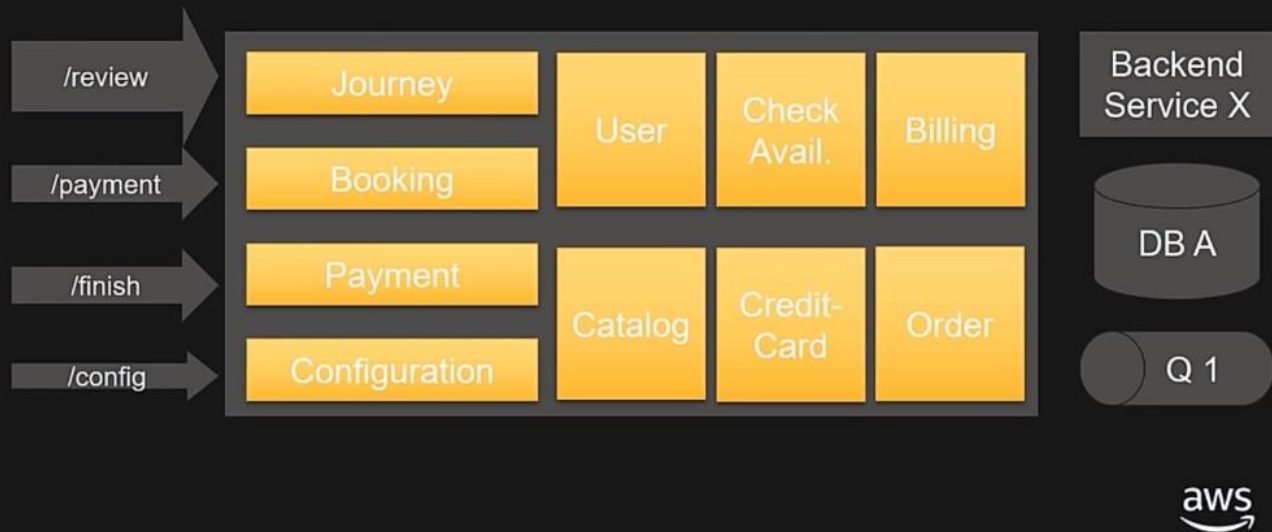
Fully qualified class name
com.dynatrace.easytravel.frontend.beans.BookingBean

Methods
public java.lang.String performBooking ✕

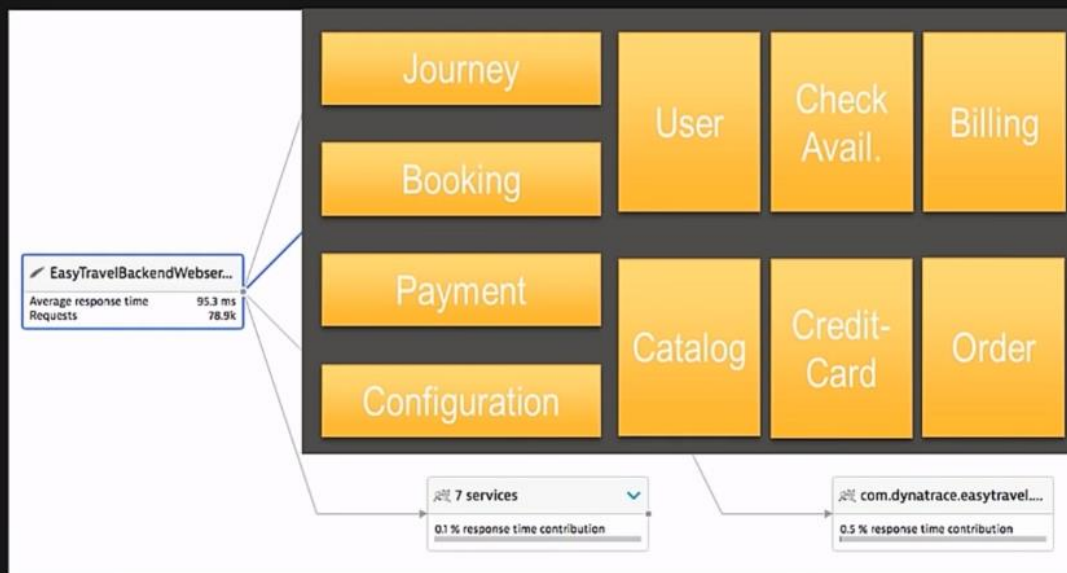
Search for methods Add method manually

aws

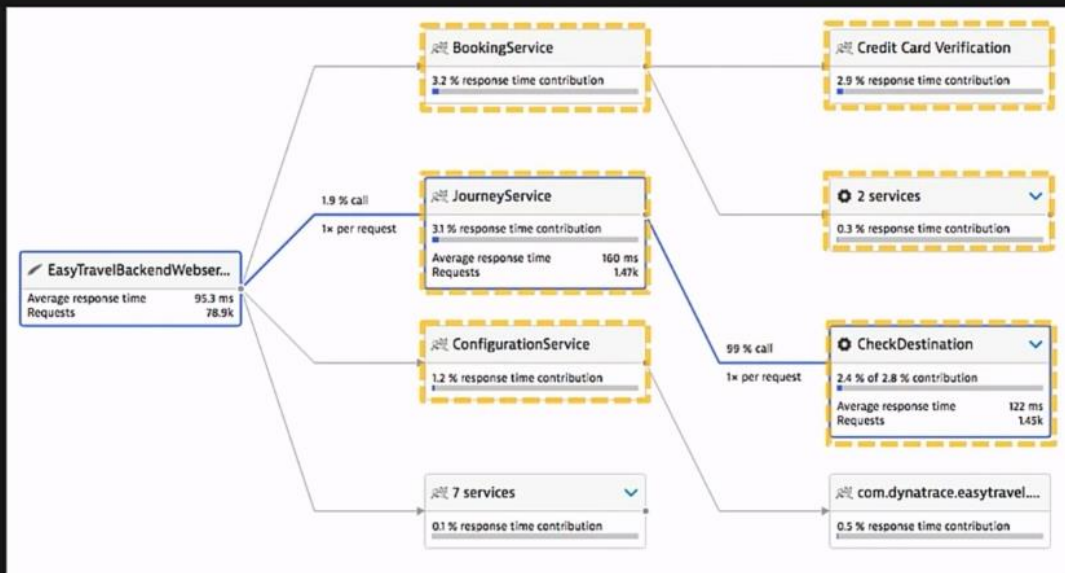
We “virtually” broke the monolith!



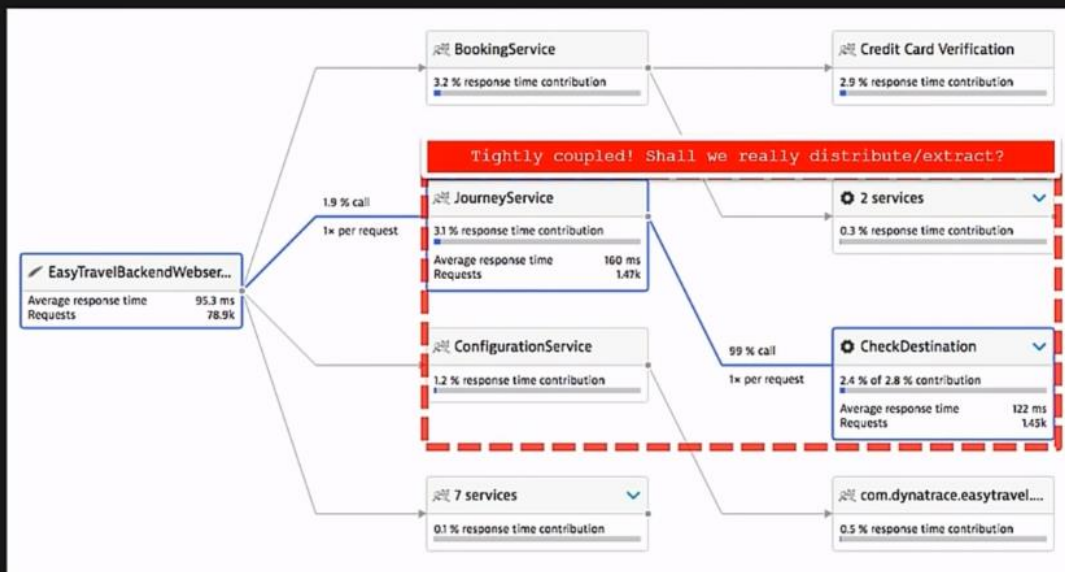
Step #6 Learn from Dynatrace Data



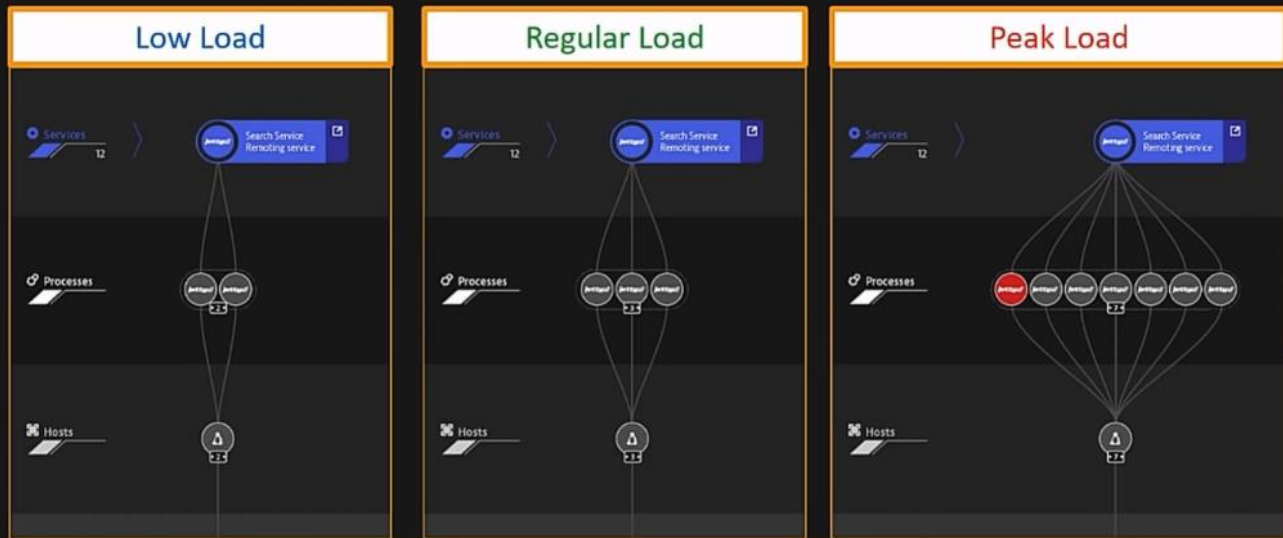
Step #6 Learn from Dynatrace Data



Step #6 Learn from Dynatrace Data



Dynatrace validates & optimizes scalability



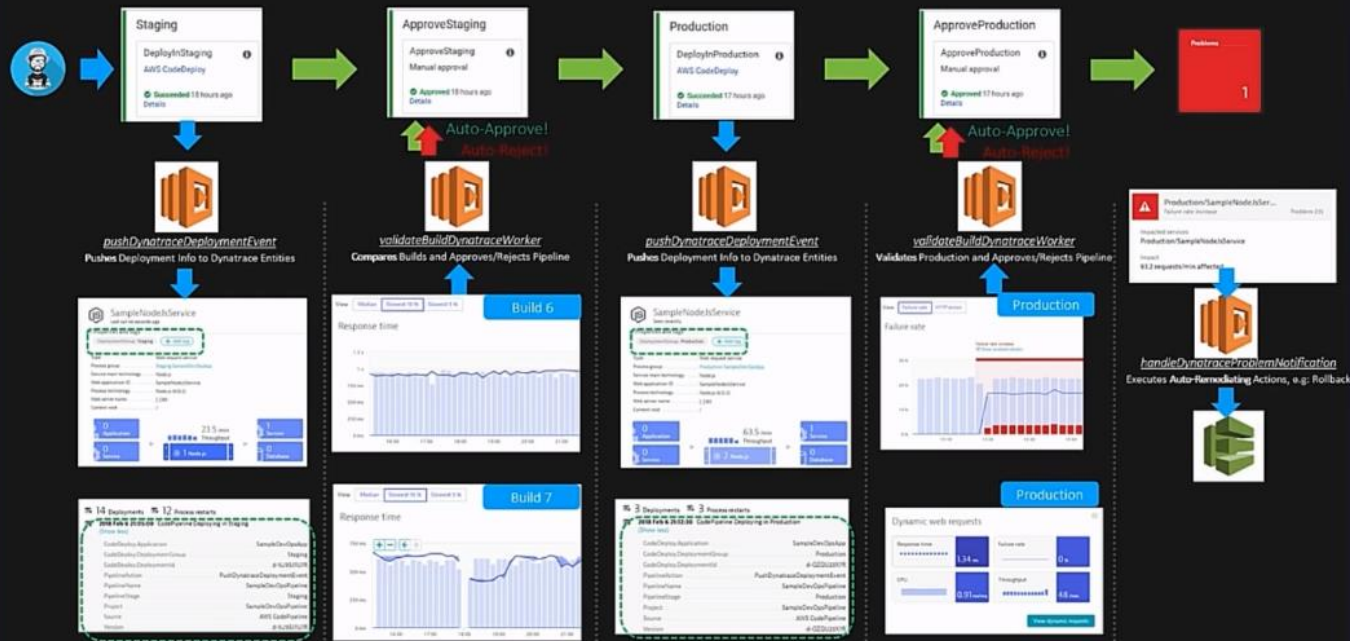
How does each service scale? Does it correctly fail over??

Dynatrace Baselineing Validates Before vs After!



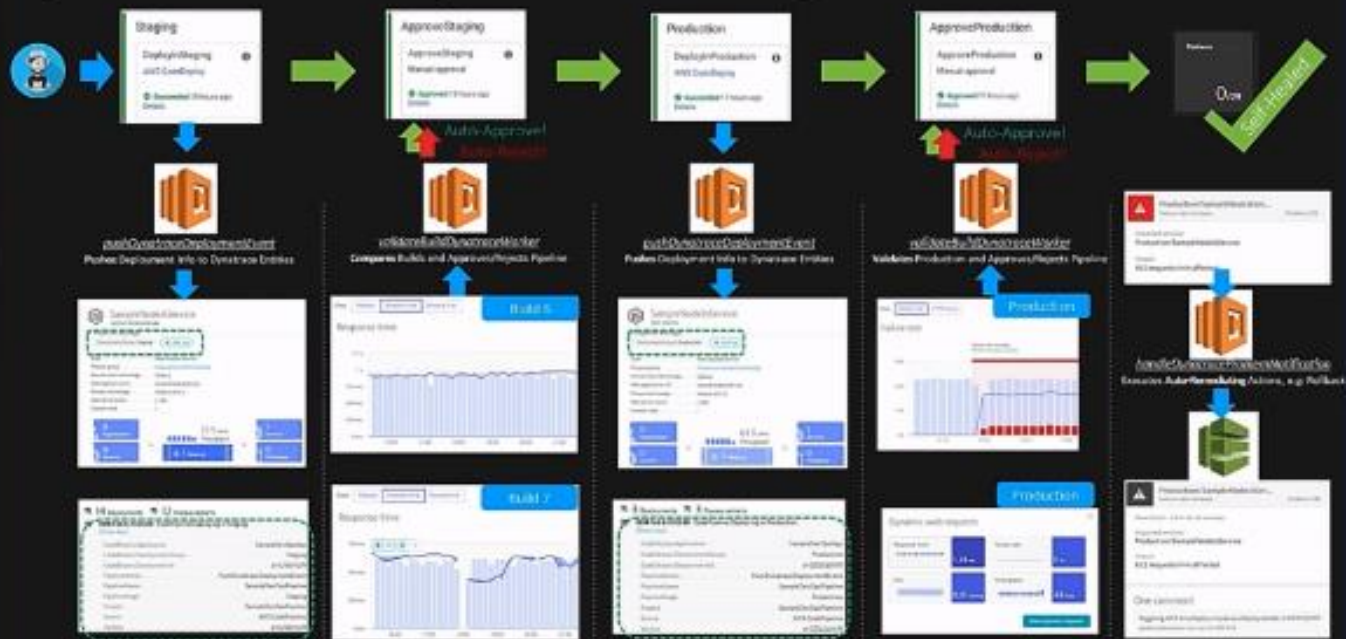
How is Performance & Resource Consumption per Service Endpoint?

Dynatrace APIs: How to DevOps? Shift-Left? Self-Healing?



<https://github.com/Dynatrace/AWSDevOpsTutorial>

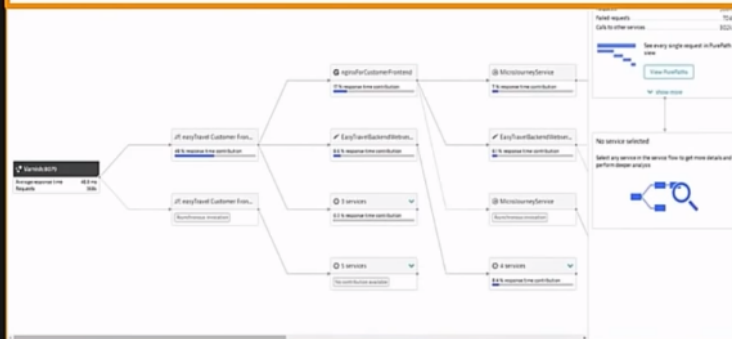
Dynatrace APIs: How to DevOps? Shift-Left? Self-Healing?



<https://github.com/Dynatrace/AWSDevOpsTutorial>

Dynatrace Service Flow Validates your Architecture!

Planned Architectural Service Flow

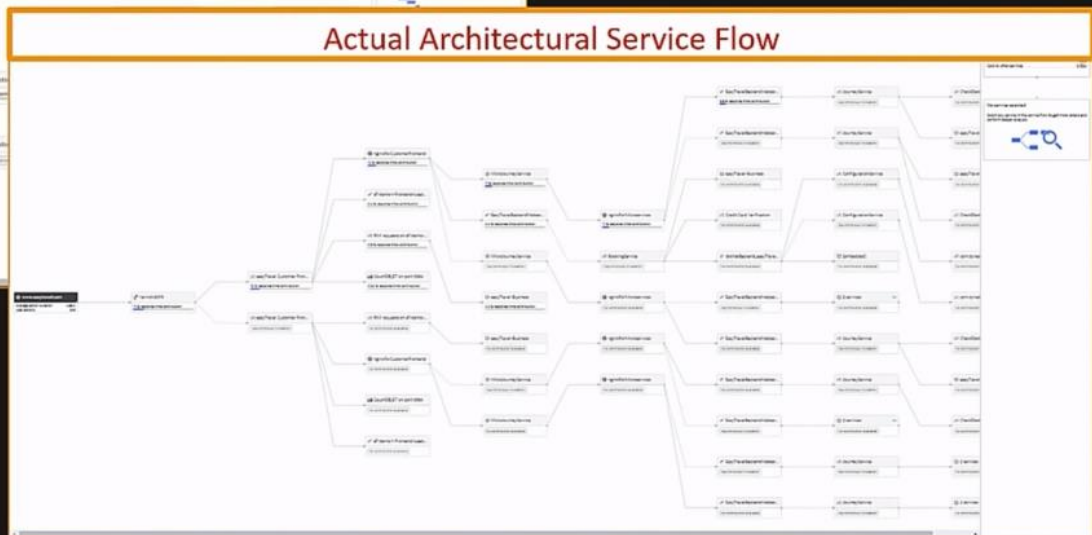


Dynatrace Service Flow Validates your Architecture!

Planned Architectural Service Flow



Actual Architectural Service Flow

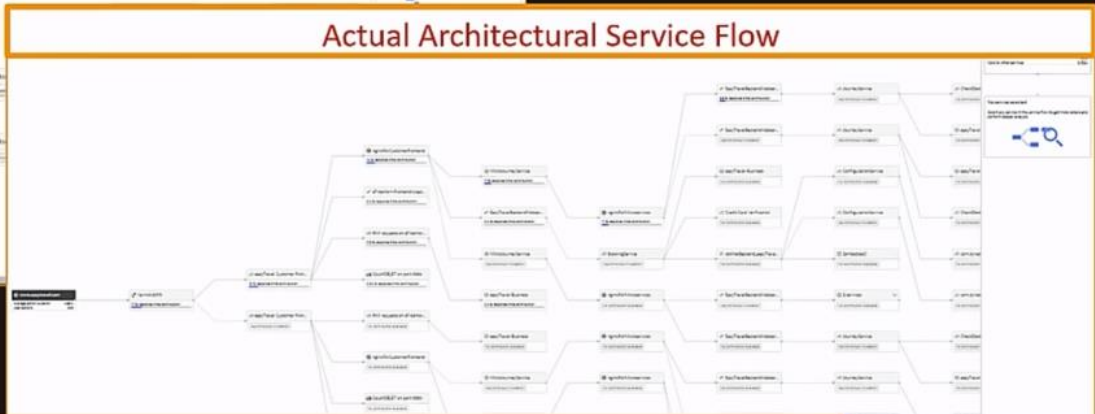


Dynatrace Service Flow Validates your Architecture!

Planned Architectural Service Flow



Actual Architectural Service Flow



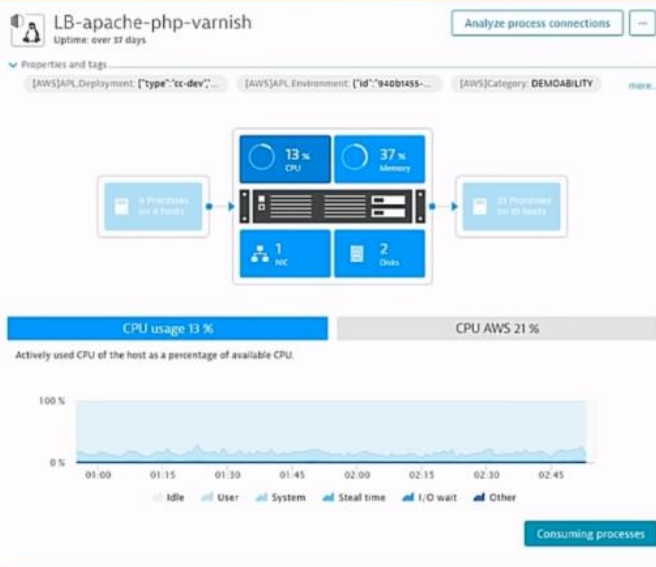
Identify / Optimize Architectural Patterns

Recursive Calls, N+1 Call Pattern, Chatty Interfaces, No Caching Layer ...

Dynatrace Cloud Monitoring Validates Cloud Services

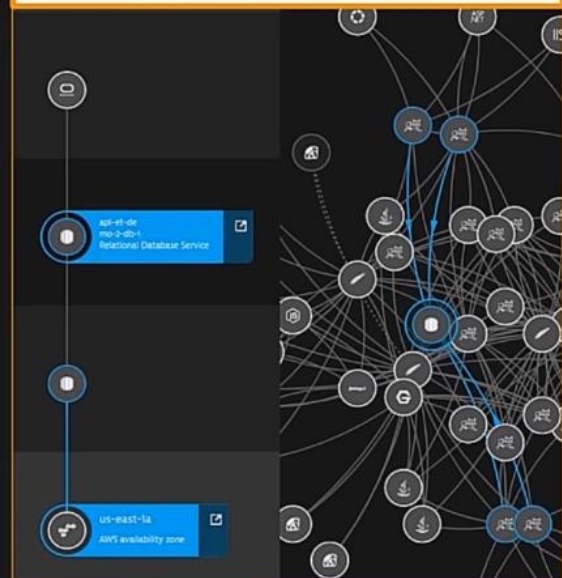
WHAT RUNS WHERE?

Monitoring Data and Tags for each EC2 instance through CloudWatch

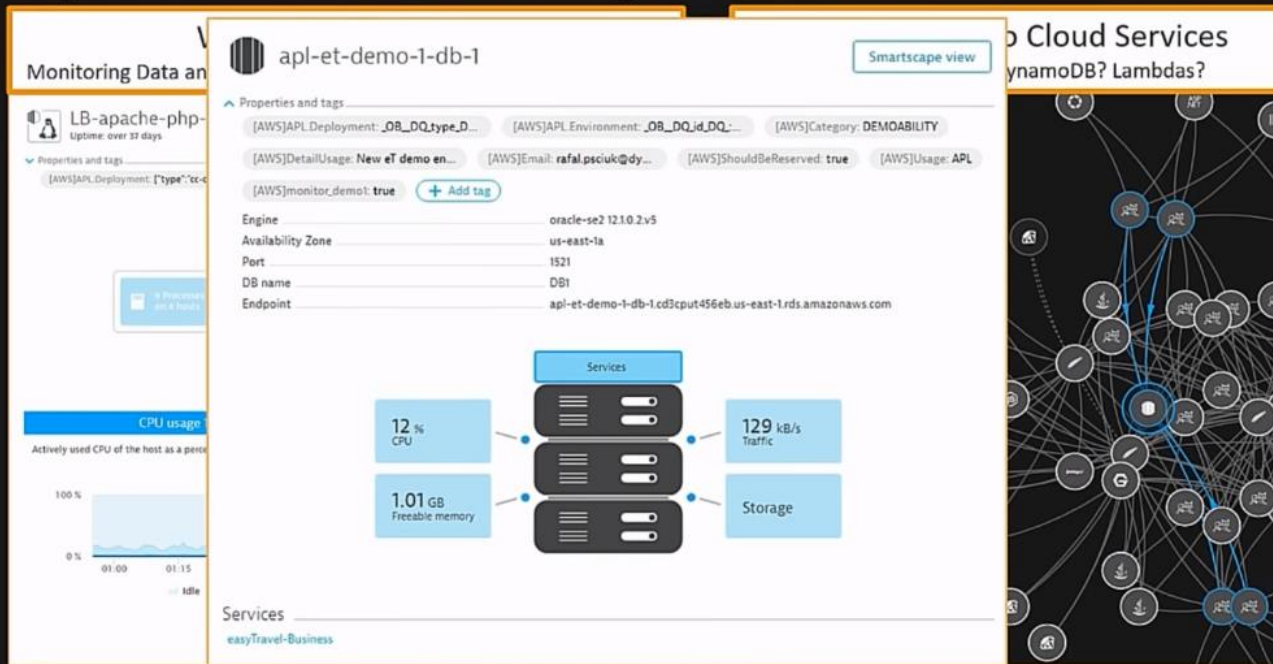


Dependencies to Cloud Services

Who is calling RDS? DynamoDB? Lambdas?

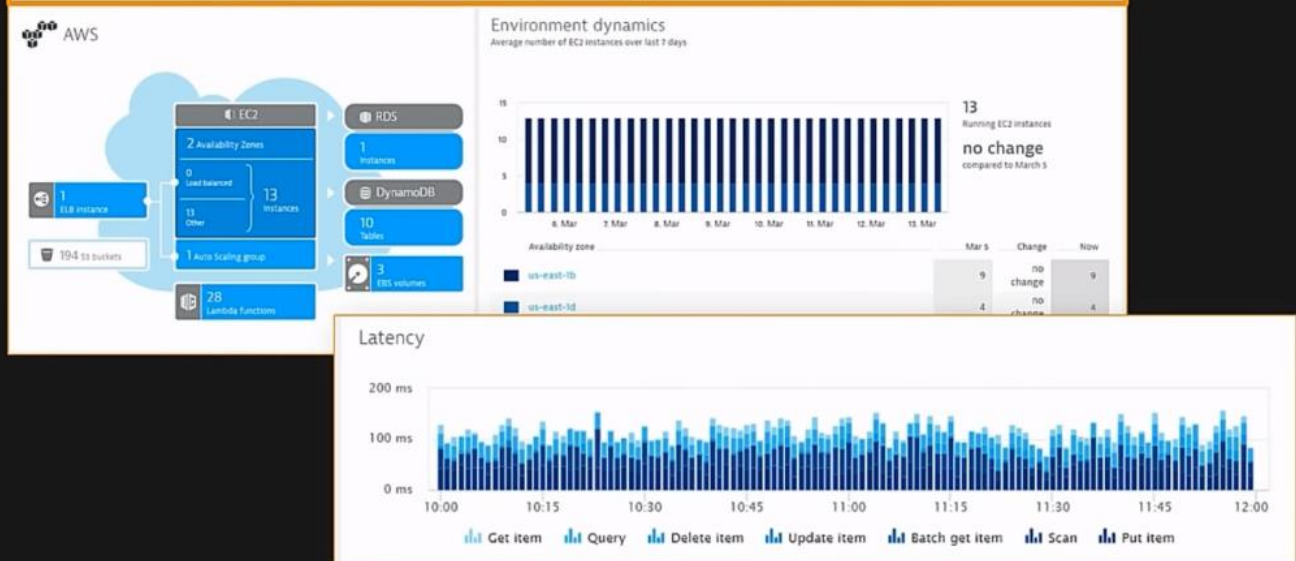


Dynatrace Cloud Monitoring Validates Cloud Services



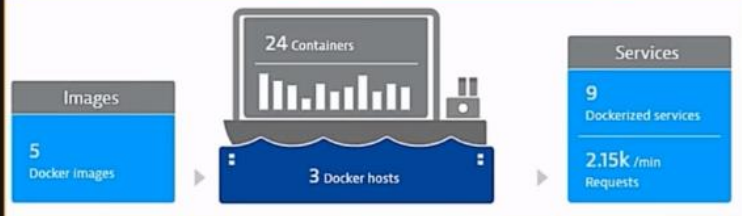
Dynatrace Cloud Monitoring Validates Cloud Resource Usage

Automated Cloud Resource Usage and Error Detection

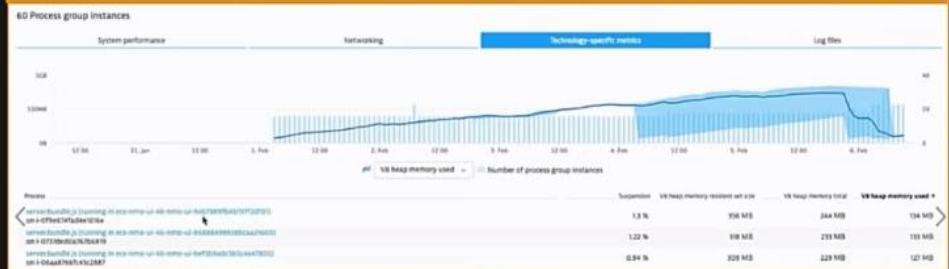


Dynatrace Container Monitoring

Automatic Injection & Meta Data Extraction!

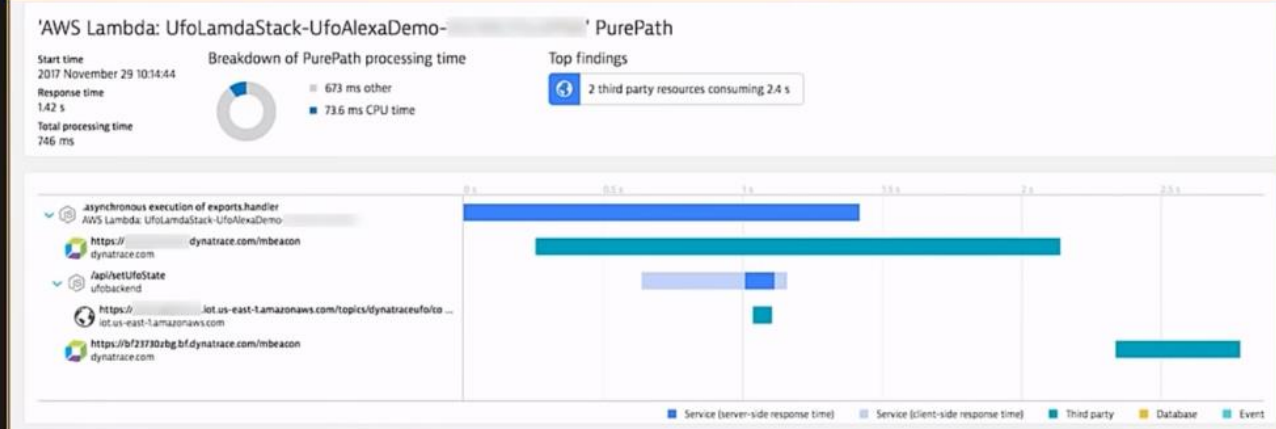


Automatic Service Monitoring within the Containers



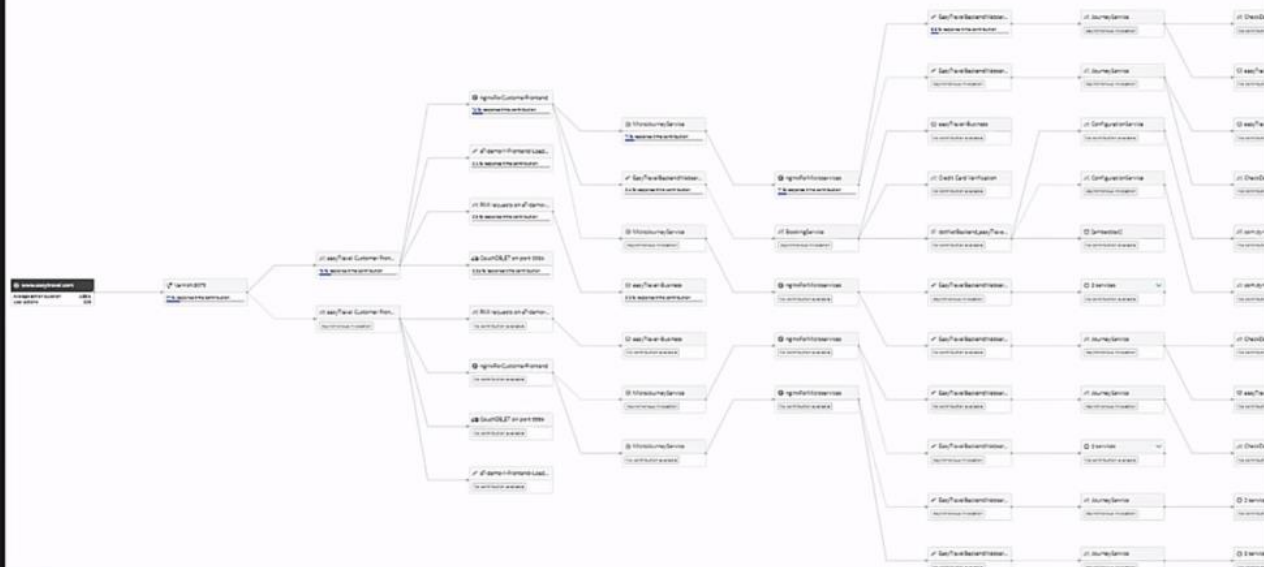
Dynatrace End-to-End through Lambda

Method Level Visibility: No Code Change Required!



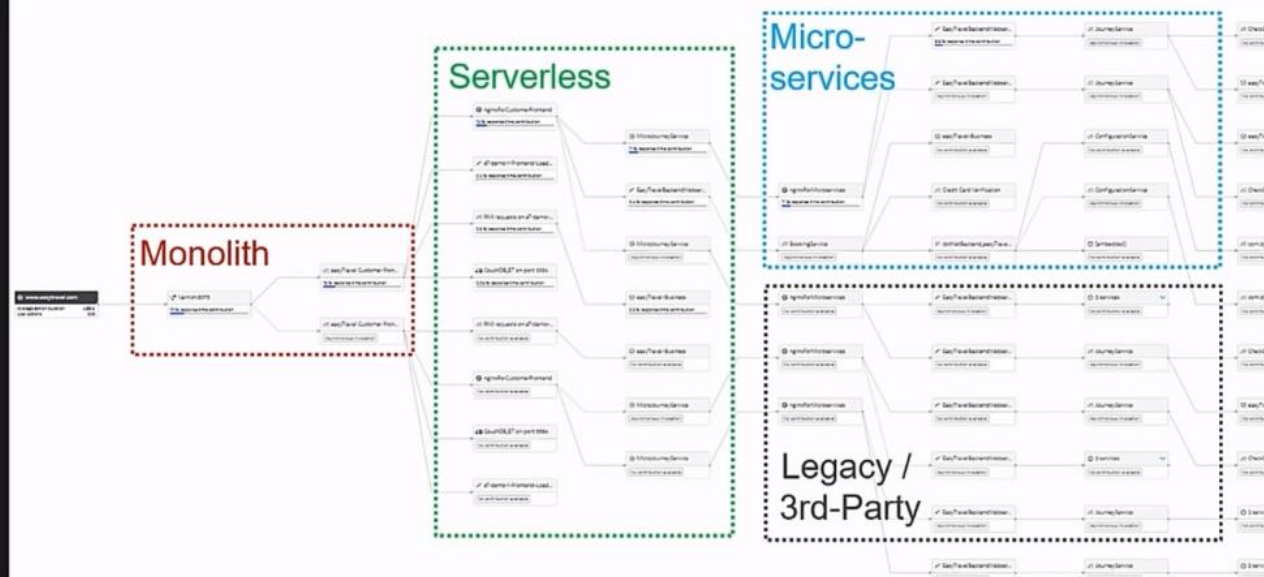
Dynatrace End-to-End through your Stack

From Monolith via Microservices to Serverless – and into the Mainframe



Dynatrace End-to-End through your Stack

From Monolith via Microservices to Serverless – and into the Mainframe



Dynatrace RUM Validates User Behavior Before vs After!

How are Conversions, Bounce Rates, Interaction Behavior, JavaScript Errors, Crashes?

www.dynatrace.com

User behavior

Active sessions, Actions per session, Entry/Exit actions, Bounce rate, and Conversion goals.

The dashboard displays the following metrics:

- 96 % New users** (Top users)
- 76 % Synthetic** (Top user types)
- View geographic breakdown**
- 41.3 /min** (Active sessions)
- 2.1** (Actions per session)
- 77.8 %** (Bounce rate)
- 4.6 s** (Action duration)
- 29.6 s** (Action duration)
- 3.8 s** (Action duration)
- 3.0 %** (Overall conversion)

Conversion trend

Shows overall rate of success toward your conversion goals. Each converted session met at least one of your conversion goals.

The chart shows the conversion rate (purple bars) and converted sessions (grey bars) from November 8 to December 8. The conversion rate fluctuates between approximately 30% and 60%, while converted sessions range from about 1 to 2.5 per minute.

— Conversion rate ■ Converted sessions

No problems in last 72 hours

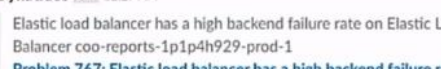
Frequent issues: User action duration degradation, JavaScript error rate increase
 Dynatrace reports recurring problem patterns as "frequent issues." Alerts are sent out only if severity increases.

Top conversion goals

See the conversion rate of your defined goals.

[Start Free Trial](#) 3.1 % [View full details](#)

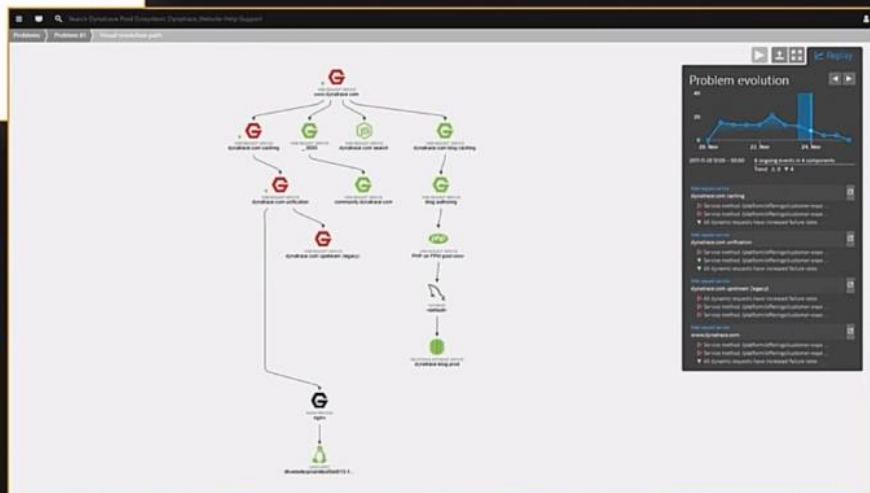
Dynatrace AI: When something fails, find out: Where, Why and How *Faster*



The screenshot shows a Dynatrace dashboard interface. At the top, the Dynatrace logo and the text 'APR 12:27 AM' are visible. The main content area displays a problem entry with the title 'Elastic load balancer has a high backend failure rate in environment: Production'. Below the title, the identifier 'coo-reports-1p1p4h929-prod-1' is shown. The problem is dated 'Jan 13th'. A snippet of the problem description is visible, showing the text 'Elastic load balancer has a high backend failure rate' repeated twice. On the right side of the dashboard, a partial view of a search bar and a list of other problems is visible.

If you break things!

ChatOps with the AI



Dynatrace AI: When something fails, find out: Where, Why and How Faster



Dynatrace APP 12:27 AM

Elastic load balancer has a high backend failure rate on Elastic Load Balancer coo-reports-1p1p4h929-prod-1

Problem 767: Elastic load balancer has a high backend failure rate in environment: Production

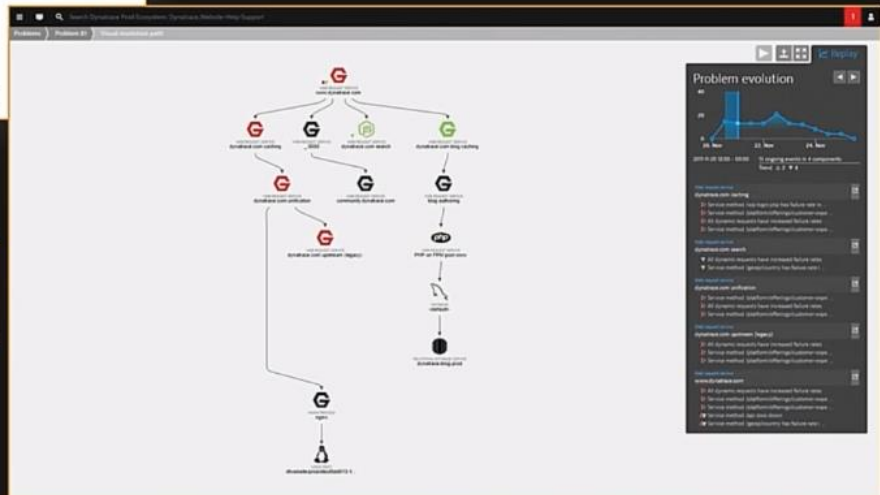
Jan 13th

coo-reports-1p1p4h929-prod-1

Elastic load balancer has a high backend failure rate

Elastic load balancer has a high backend failure rate

If you break things!
ChatOps with the AI



Try dynatrace

- Signup through AWS Marketplace
- Run as SaaS or On-Premises
- Visit us at our booth
- Follow us @dynatrace

aws partner network

Advanced
Technology
Partner

DevOps Competency

Migration Competency

Public Sector Partner

Marketplace Seller

SaaS Partner