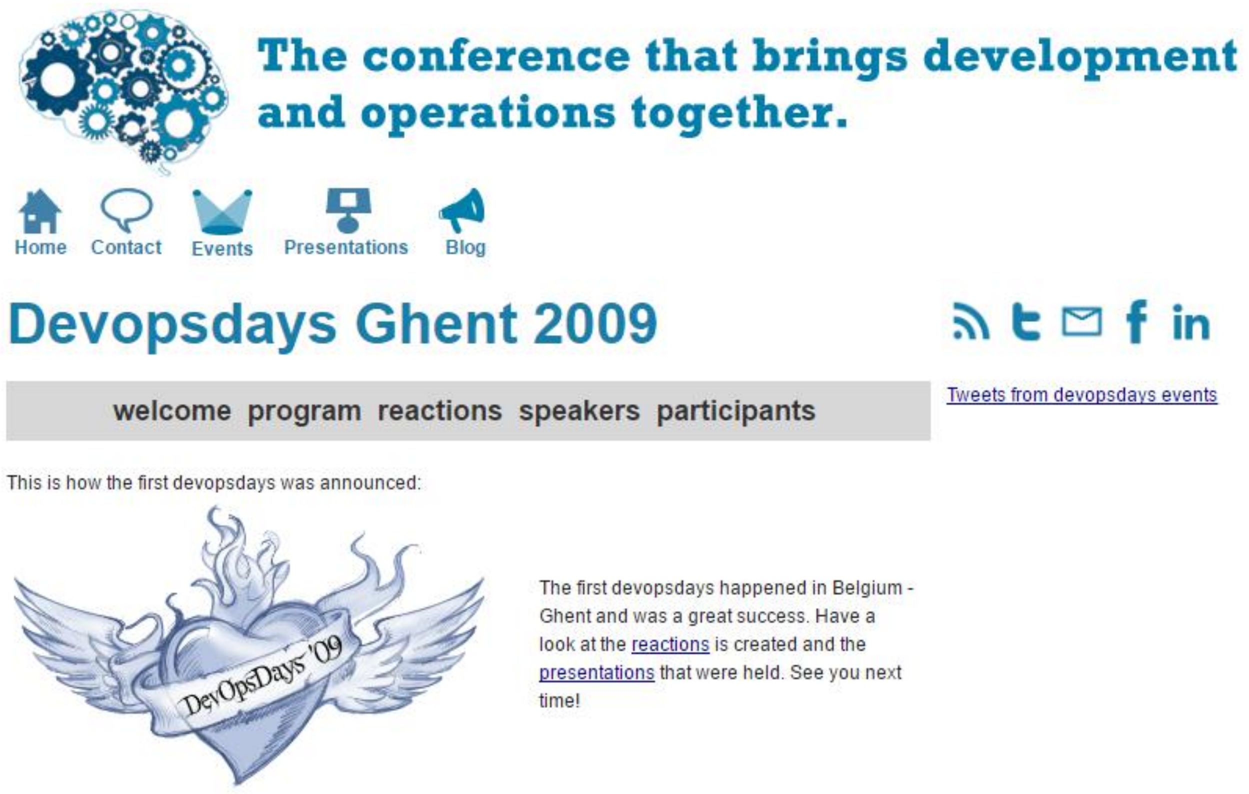
Continuous Innovation through DevOps Pipelines

Andreas Grabner: @grabnerandi, andreas.grabner@dynatrace.com
Slides: http://www.slideshare.net/grabnerandi
Podcast: https://www.spreaker.com/show/pureperformance

The Story started in 2009



The conference that brings development and operations together.

Home Contact Events Presentations Blog

Devopsdays Ghent 2009

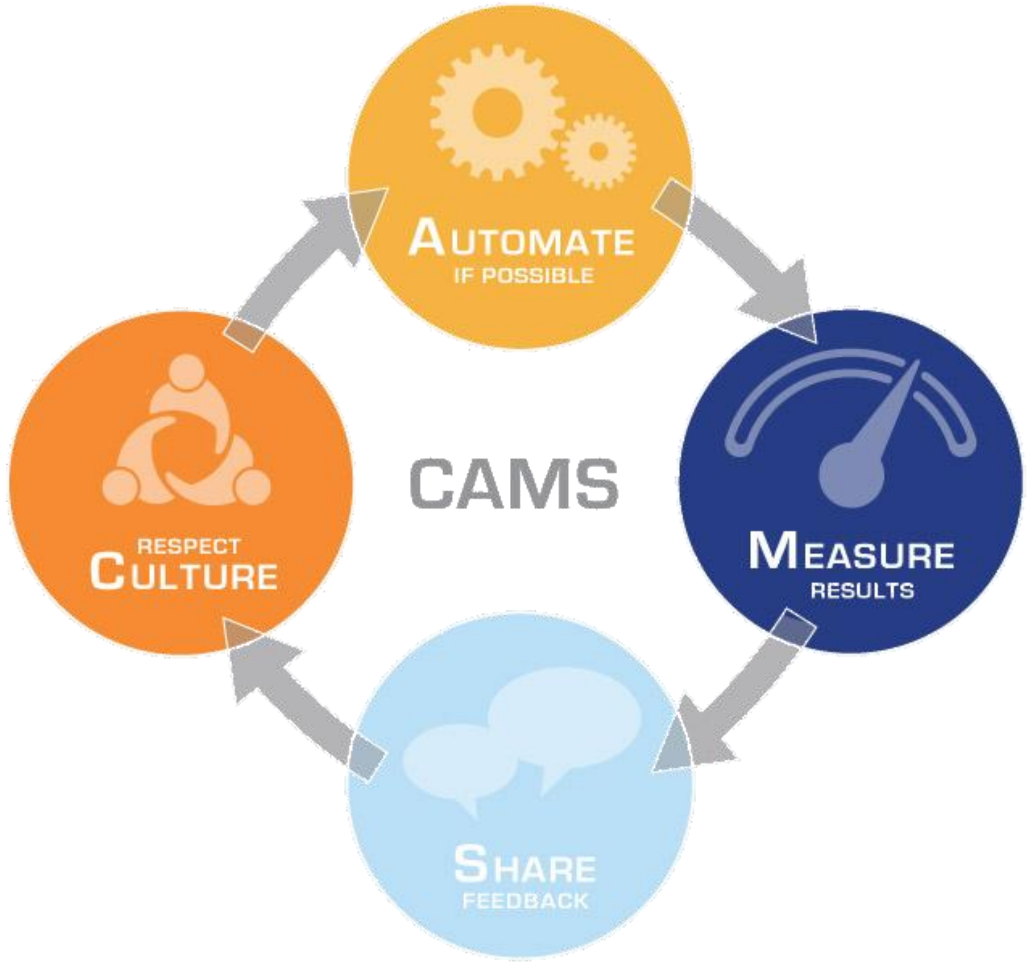
welcome program reactions speakers participants

Tweets from devopsdays events

This is how the first devopsdays was announced:

The first devopsdays happened in Belgium - Ghent and was a great success. Have a look at the reactions is created and the presentations that were held. See you next time!

@grabnerandi

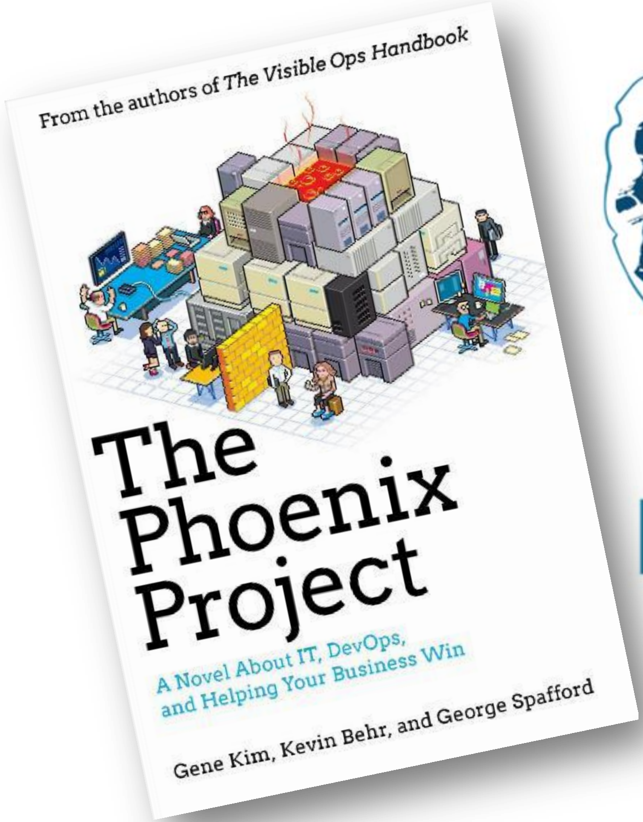


@grabnerandi

“The ***stuff we did*** when we were a ***Start Up*** and we ***All*** were ***Dev***s, ***T***esters and ***Ops***”

Quote from Andreas Grabner back in 2013 @ DevOps Boston

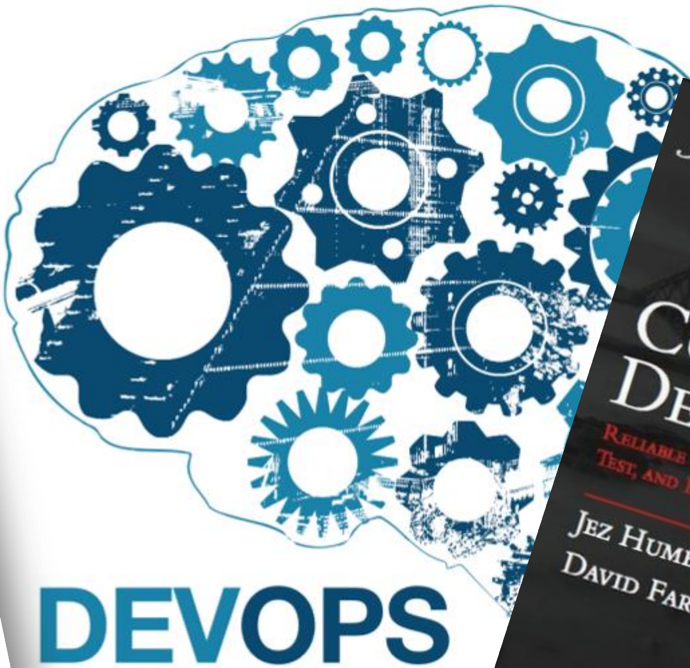
@grabnerandi


From the authors of *The Visible Ops Handbook*

The Phoenix Project

A Novel About IT, DevOps, and Helping Your Business Win

Gene Kim, Kevin Behr, and George Spafford



The Addison-Wesley Signature Series

Continuous Delivery

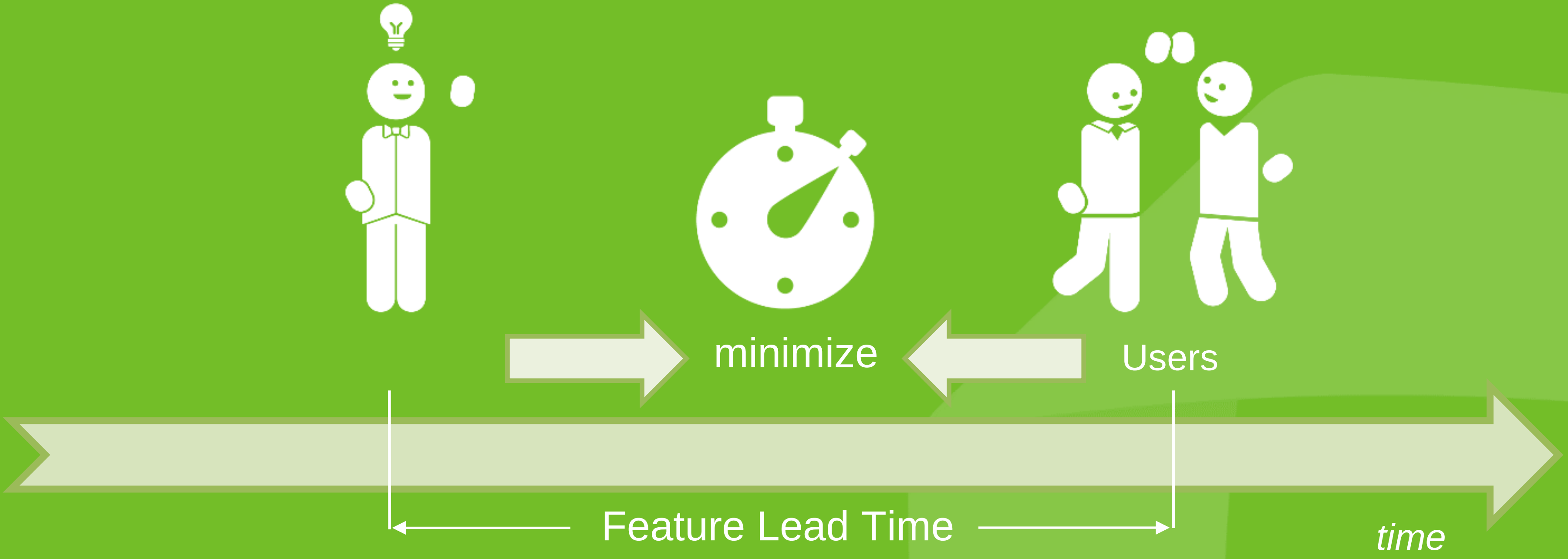
Reliable Software Releases through Build, Test, and Deployment Automation

Jez Humble, David Farley

A Martin Fowler Signature Book

@grabnerandi

Goal: Optimize Lead Time



24 "Features in a Box"

Ship the whole box!

Very late feedback ☹

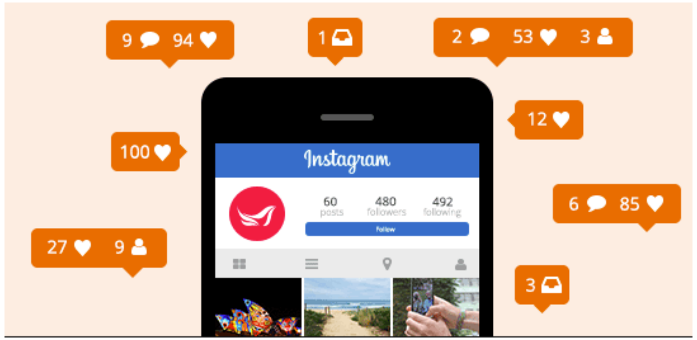
Frustration!

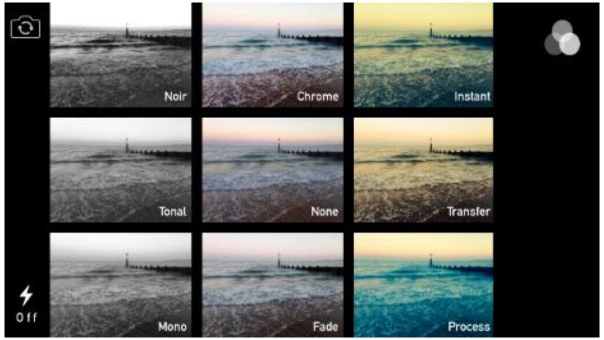
Continuous Innovation and Optimization

„1 Feature at a Time“

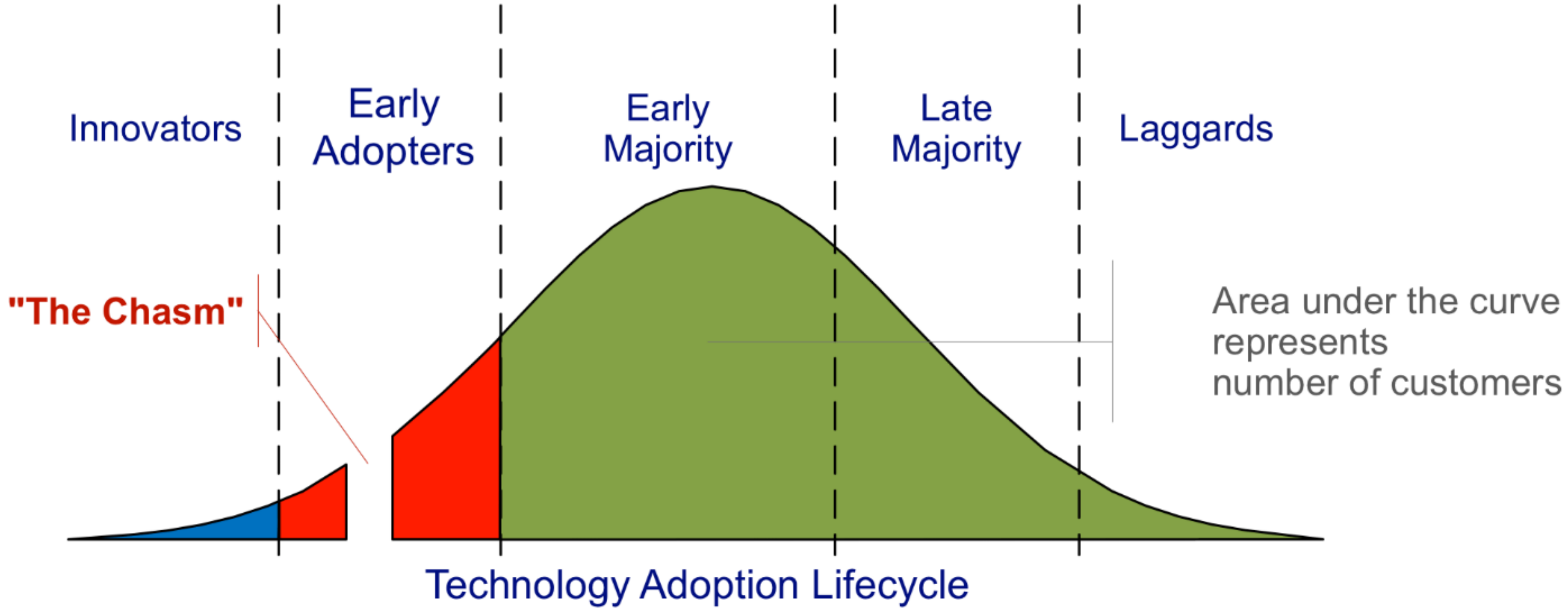
„Immediate Customer Feedback“

„Optimize before Deploy“



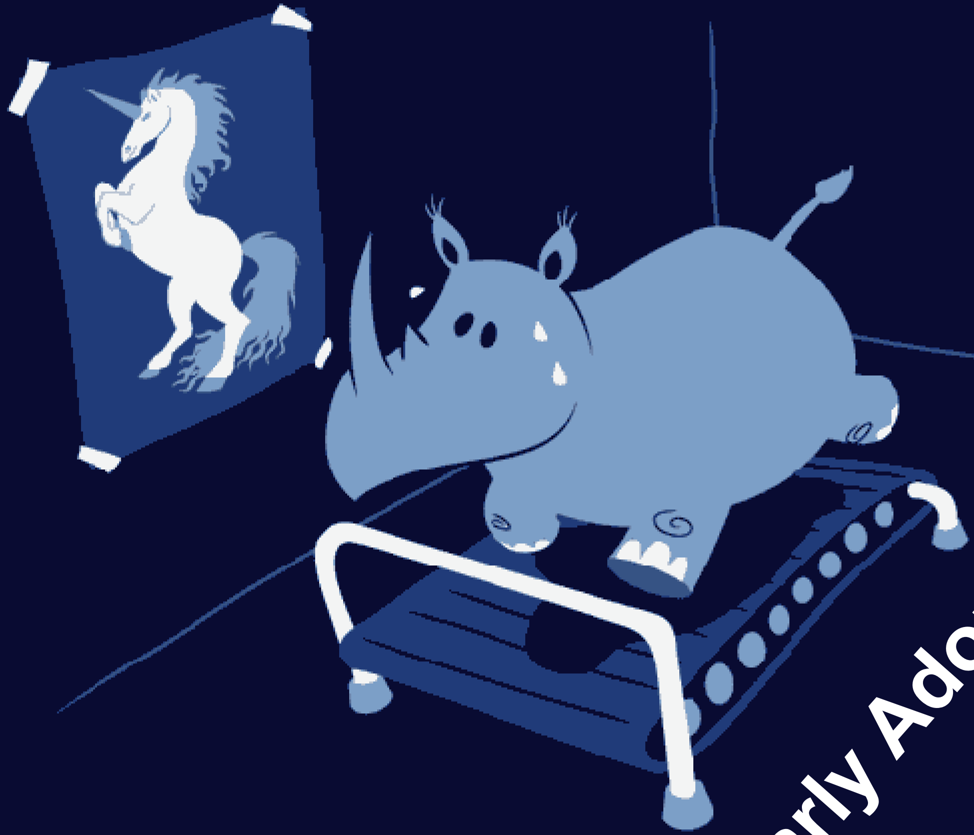


DevOps Adoption


Innovators
Early Adopters
Early Majority
Late Majority
Laggards
"The Chasm"
Area under the curve represents number of customers
Technology Adoption Lifecycle

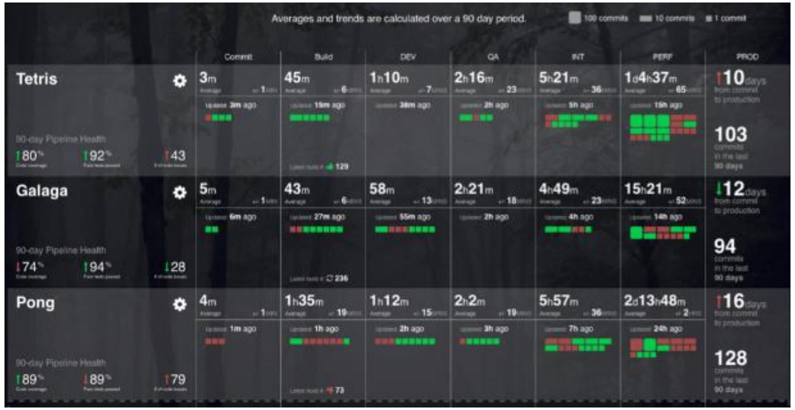
Innovators (aka Unicorns): Deliver value at the *speed of business*

cars.com	→	*700 deployments / YEAR*
flickr	→	*10 + deployments / DAY*
Etsy	→	*50 – 60 deployments / DAY*
amazon.com	→	*Every 11.6 SECONDS*



Early Adopters

“We Deliver ***High Quality Software, Faster*** and ***Automated*** using ***New Stack***“


Averages and trends are calculated over a 90 day period. 100 commits 10 commits 1 commit

	Commit	Build	DEV	QA	INT	PERF	PROD
Tetris	3m Average 1	45m Average 6	1h10m Average 7	2h16m Average 23	5h21m Average 36	1d4h37m Average 65	↑10 days from commit to production
	Updated 3m ago	Updated 15m ago	Updated 38m ago	Updated 2h ago	Updated 5h ago	Updated 15h ago	103 commits in the last 90 days
90-day Pipeline Health ↑80% Code coverage ↑92% Pass tests passed ↑43		Latest build # 129					
Galaga	5m Average 1	43m Average 6	58m Average 13	2h21m Average 18	4h49m Average 23	15h21m Average 52	↓12 days from commit to production
	Updated 6m ago	Updated 27m ago	Updated 55m ago	Updated 2h ago	Updated 4h ago	Updated 14h ago	94 commits in the last 90 days
90-day Pipeline Health ↓74% Code coverage ↑94% Pass tests passed ↓28		Latest build # 236					
Pong	4m Average 1	1h35m Average 19	1h12m Average 15	2h2m Average 19	5h57m Average 36	2d13h48m Average 2	↑16 days from commit to production
	Updated 1m ago	Updated 1h ago	Updated 2h ago	Updated 3h ago	Updated 7h ago	Updated 24h ago	128 commits in the last 90 days
90-day Pipeline Health ↑89% Code coverage ↓89% Pass tests passed ↑79		Latest build # 73					

„***Shift-Left Performance*** to ***Reduce Lead Time***“
Adam Auerbach, Sr. Dir DevOps

“… deploy some of our ***most critical production*** workloads on the ***AWS platform*** …”, Rob Alexander, CIO

https://github.com/capitalone/Hygieia & https://www.spreaker.com/user/pureperformance

2011

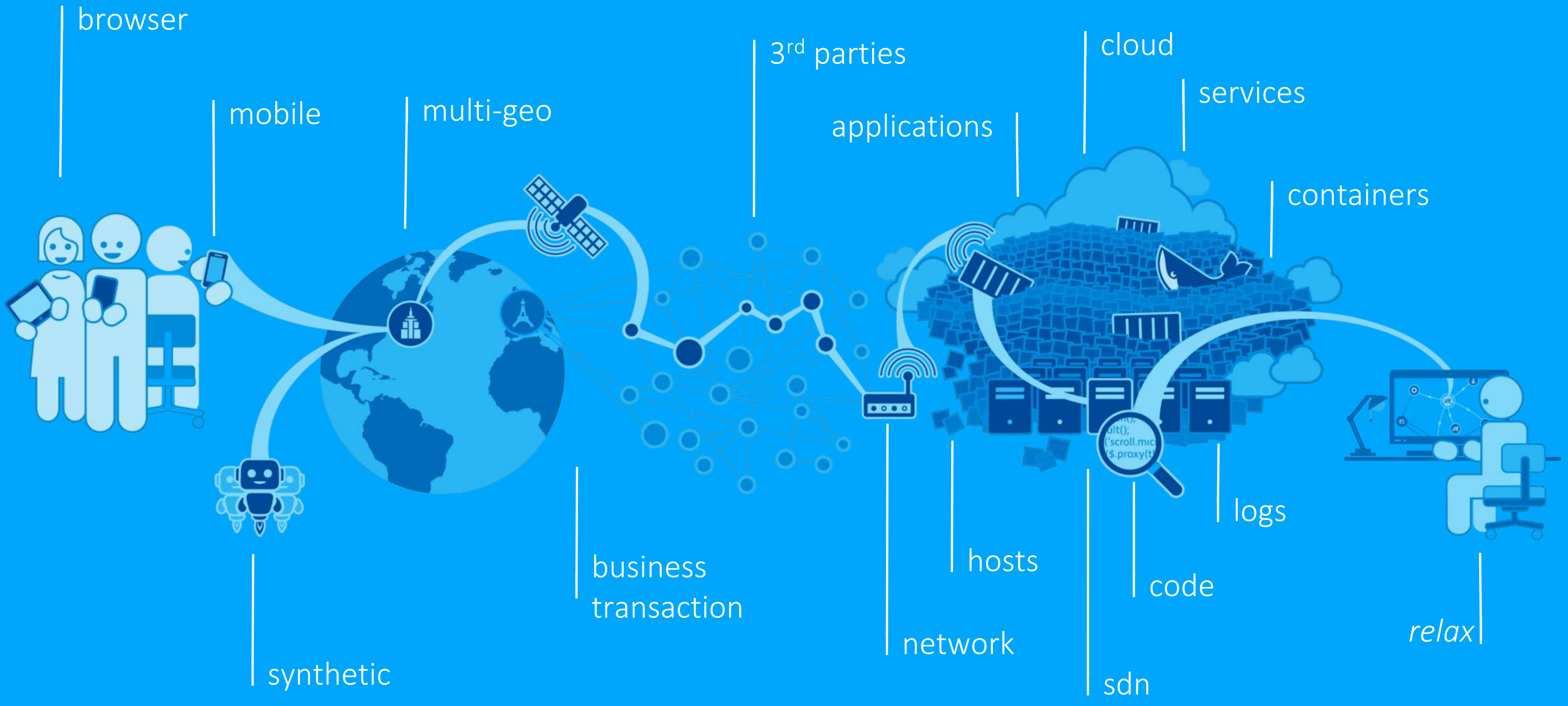
2 major releases/year
customers deploy & operate **on-prem**

2016

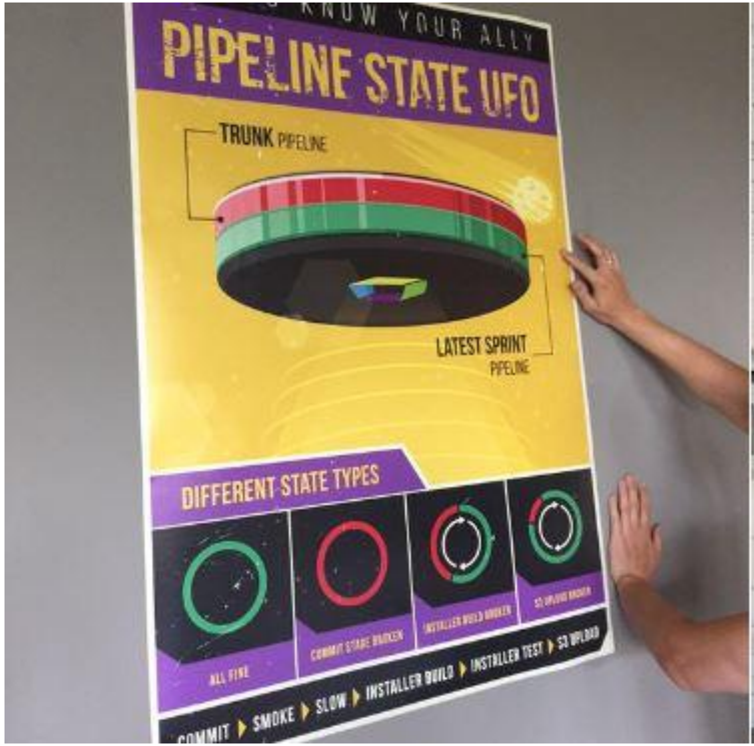
26 major releases/year
170 prod deployments/day
self-service online sales
SaaS & Managed

dynatrace

full-stack, broad, hyper-scale



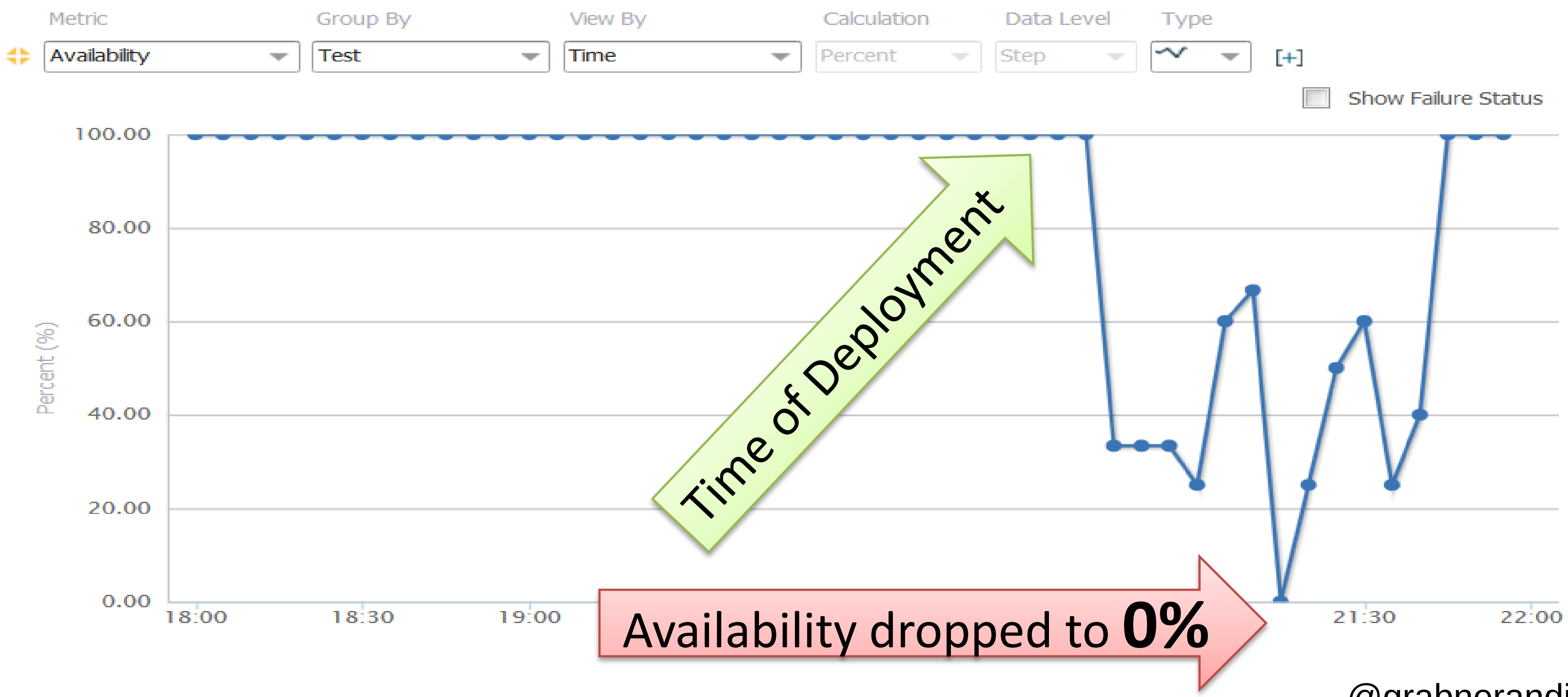
“In Your Face” Data!



https://dynatrace.github.io/ufo/

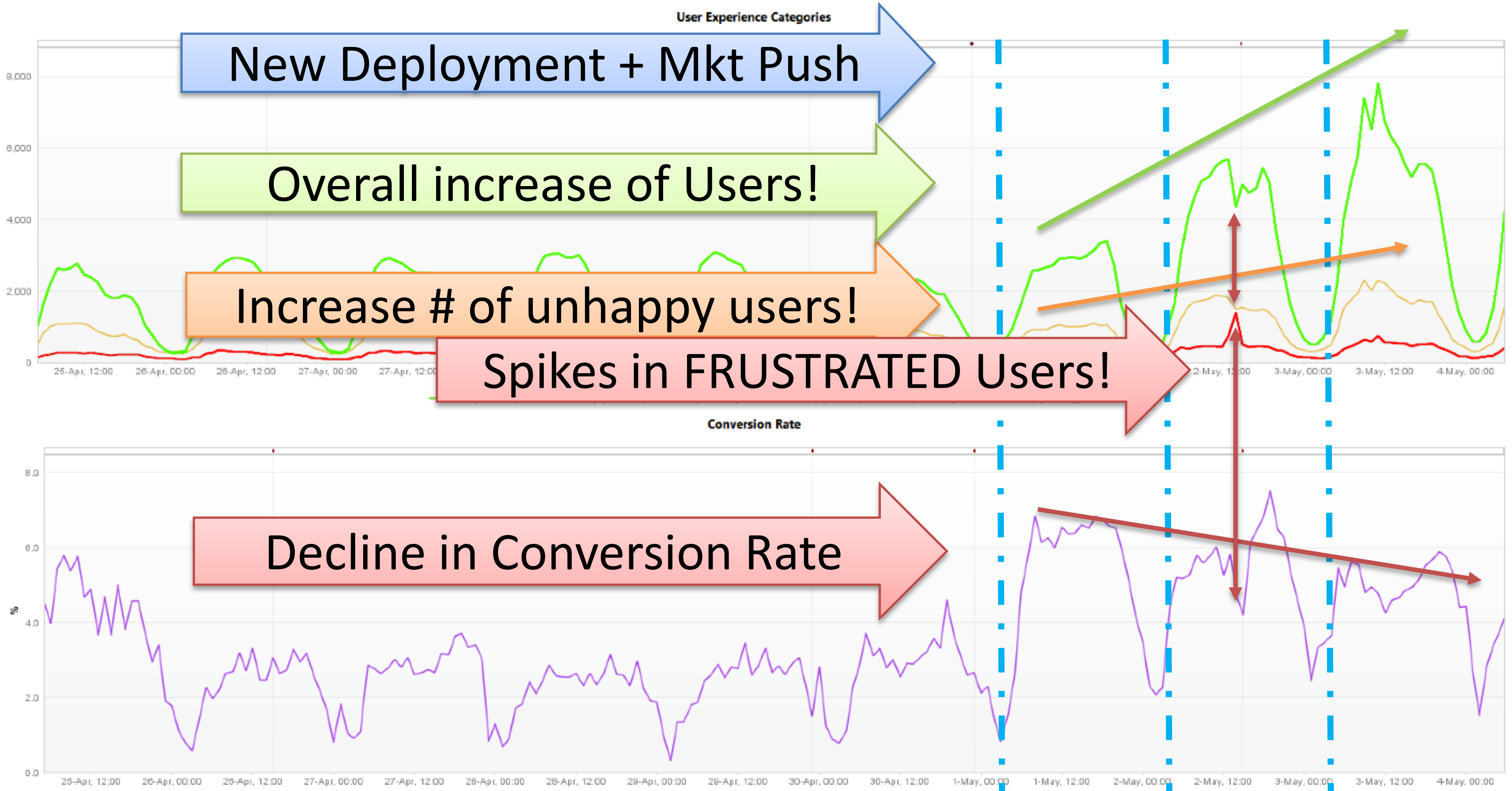
@grabnerandi

#1: Availability -> Brand Impact

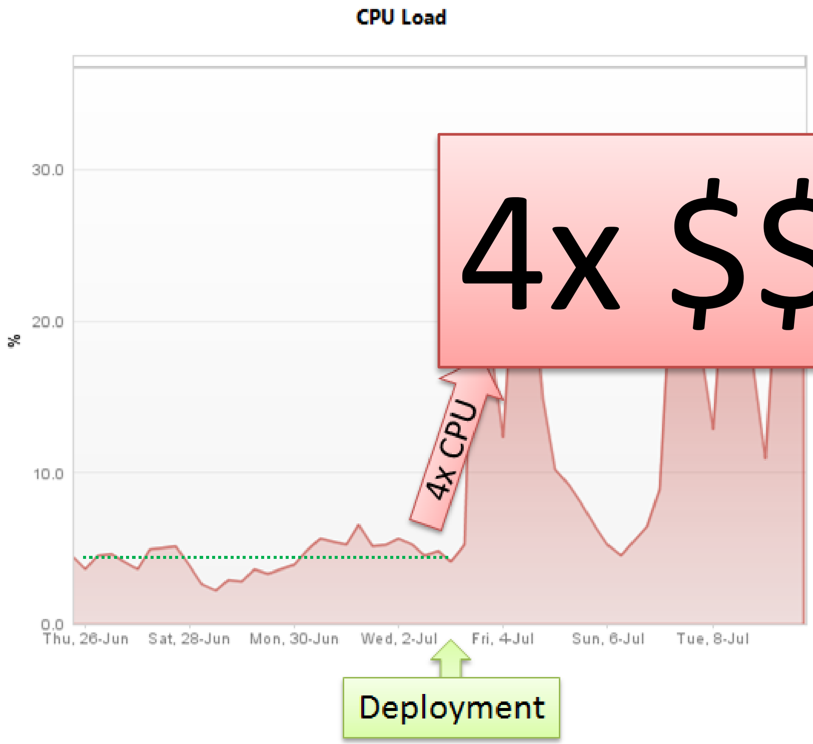


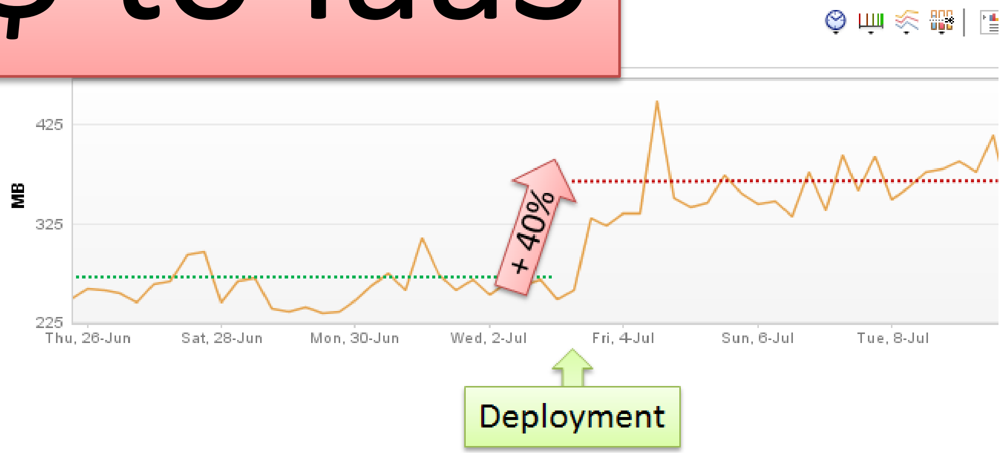
@grabnerandi

#2: User Experience -> Conversion



#3: Resource Cons -> Cost per Feature

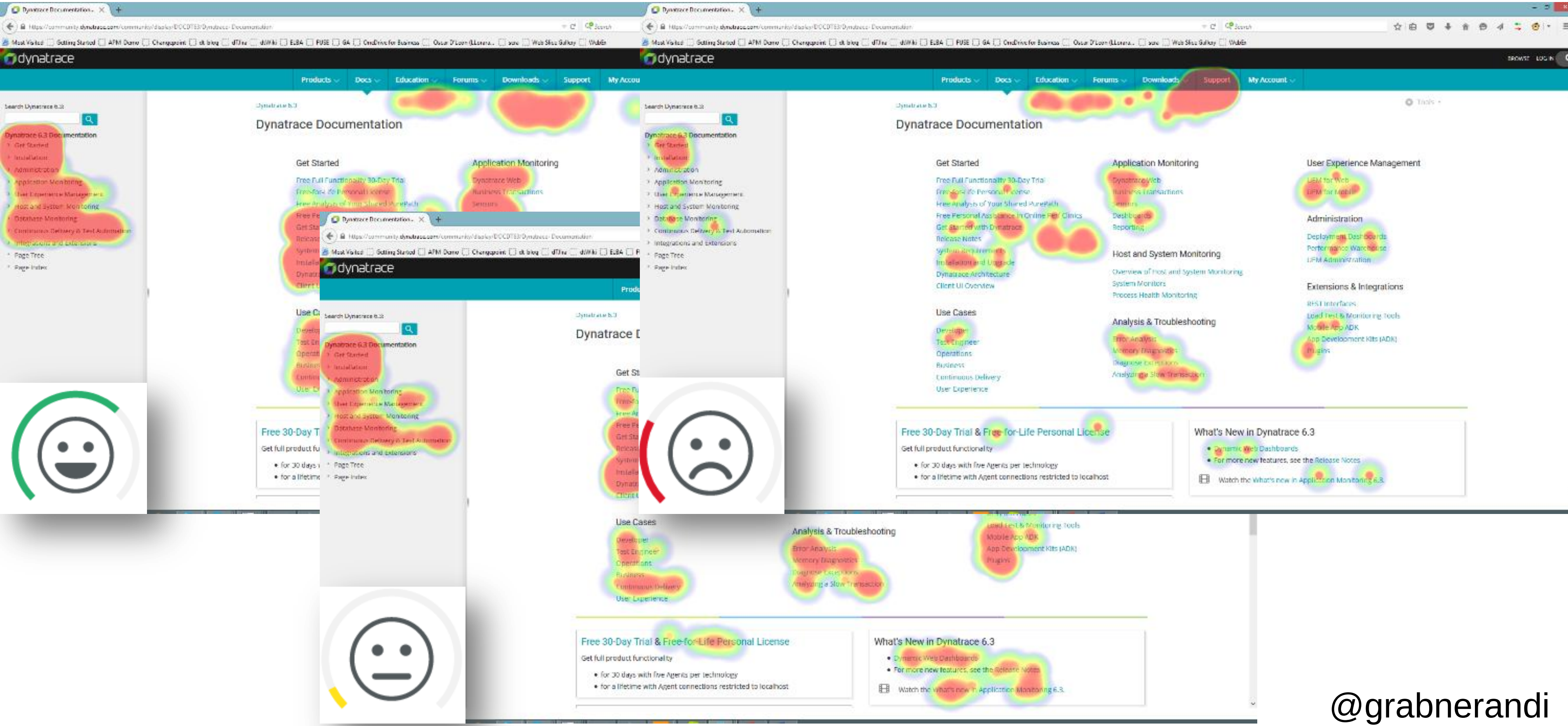




4x $$$ to IaaS

@grabnerandi

#4: Performance -> Behavior


dynatrace
Products
Docs
Education
Forums
Downloads
Support
My Account
Dynatrace Documentation
Get Started
Application Monitoring
User Experience Management
Administration
Host and System Monitoring
Extensions & Integrations
Analysis & Troubleshooting
Use Cases
Free 30-Day Trial & Free-for-Life Personal License
What's New in Dynatrace 6.3

@grabnerandi

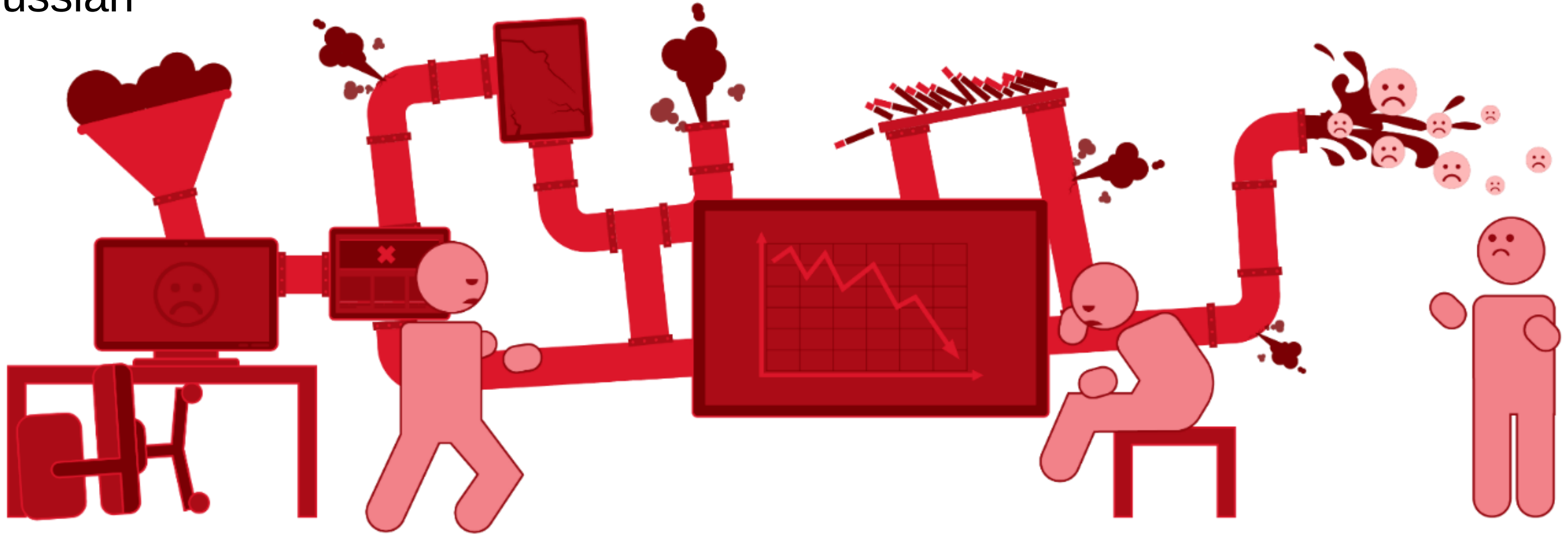
Not every Sprint ends without bruises!

Understanding Code Complexity

- 4 Millions Lines of Monolith Code
- Partially coded and commented in Russian

From Monolith to Microservice

- Initial devs no longer with company
- What to extract withouth breaking it?



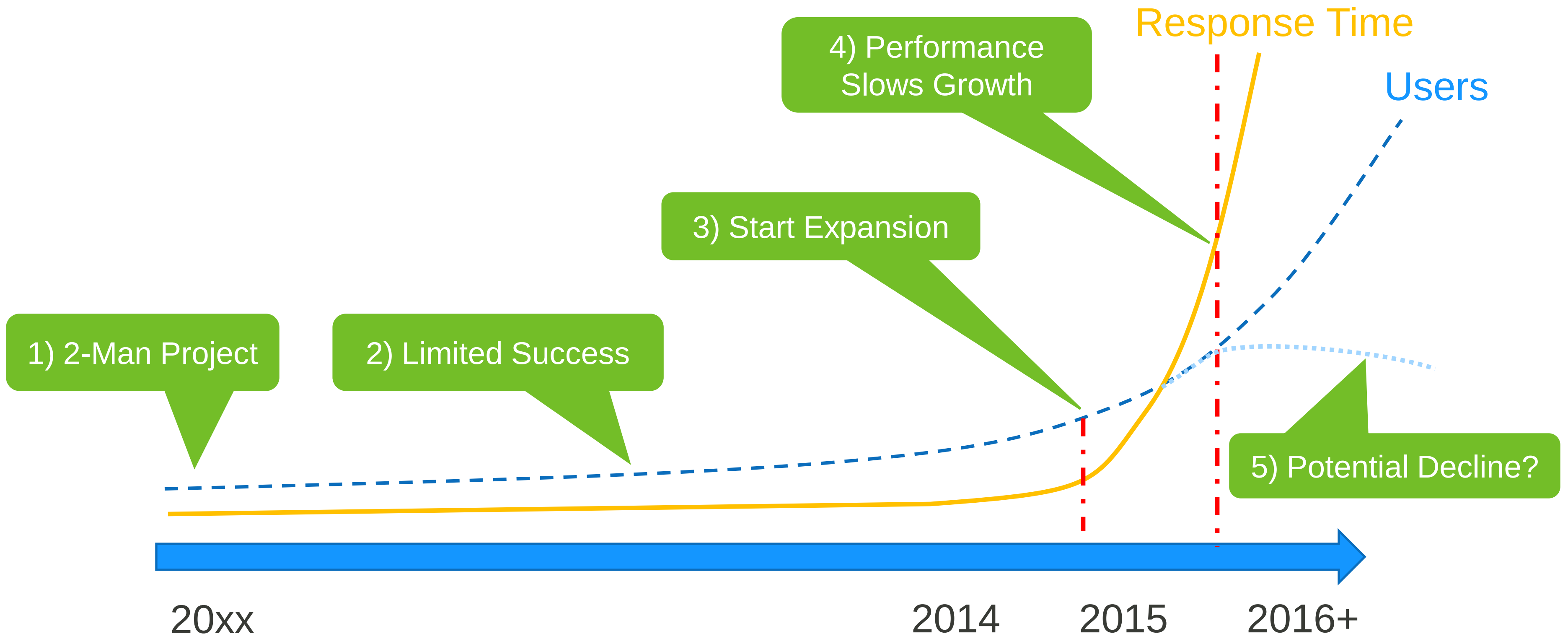
Shift Left Quality & Performance

- No automated testing in the pipeline
- Bad builds just made it into production

Cross Application Impacts

- Shared Infrastructure between Apps
- No consolidated monitoring strategy

Scaling an Online Sports Club *Search Service*

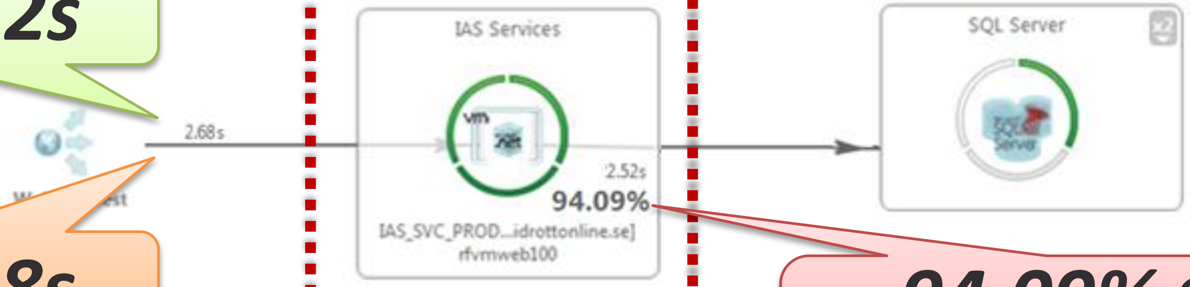


@grabnerandi

Early 2015: Monolith Under Pressure

April: 0.52s

May: 2.68s

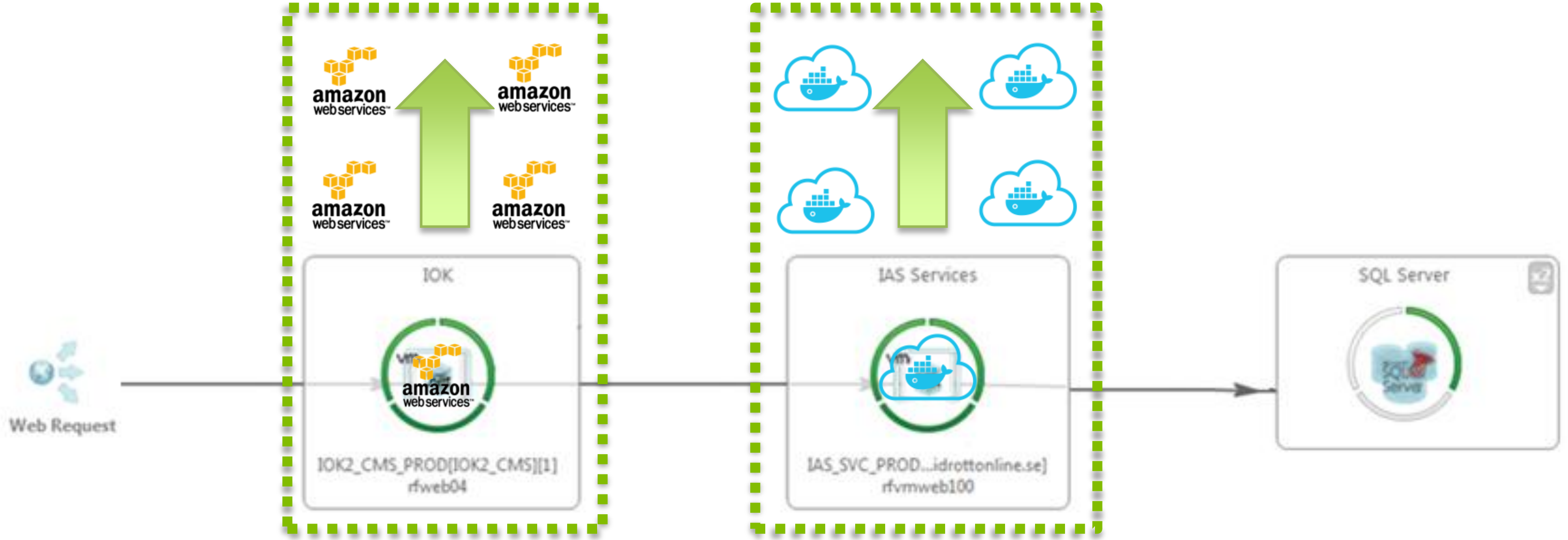


94.09% CPU Bound

Can‘t scale vertically endlessly!

@grabnerandi

From Monolith to Services in a Hybrid-Cloud

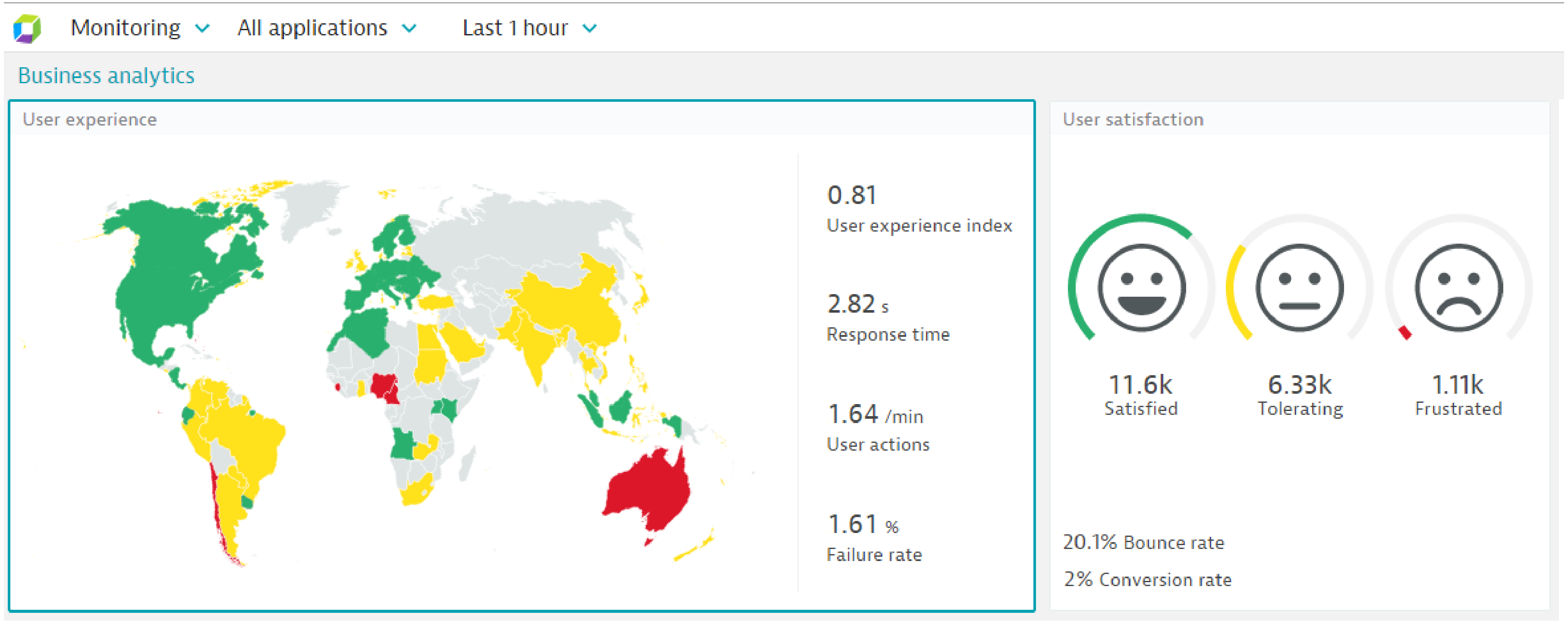


Front End in Geo-Distributed Cloud

Scale Backend in Containers On Premise

@grabnerandi

Go live – 7:00 a.m.



User experience

0.81
User experience index

2.82 s
Response time

1.64 /min
User actions

1.61 %
Failure rate

User satisfaction

11.6k
Satisfied

6.33k
Tolerating

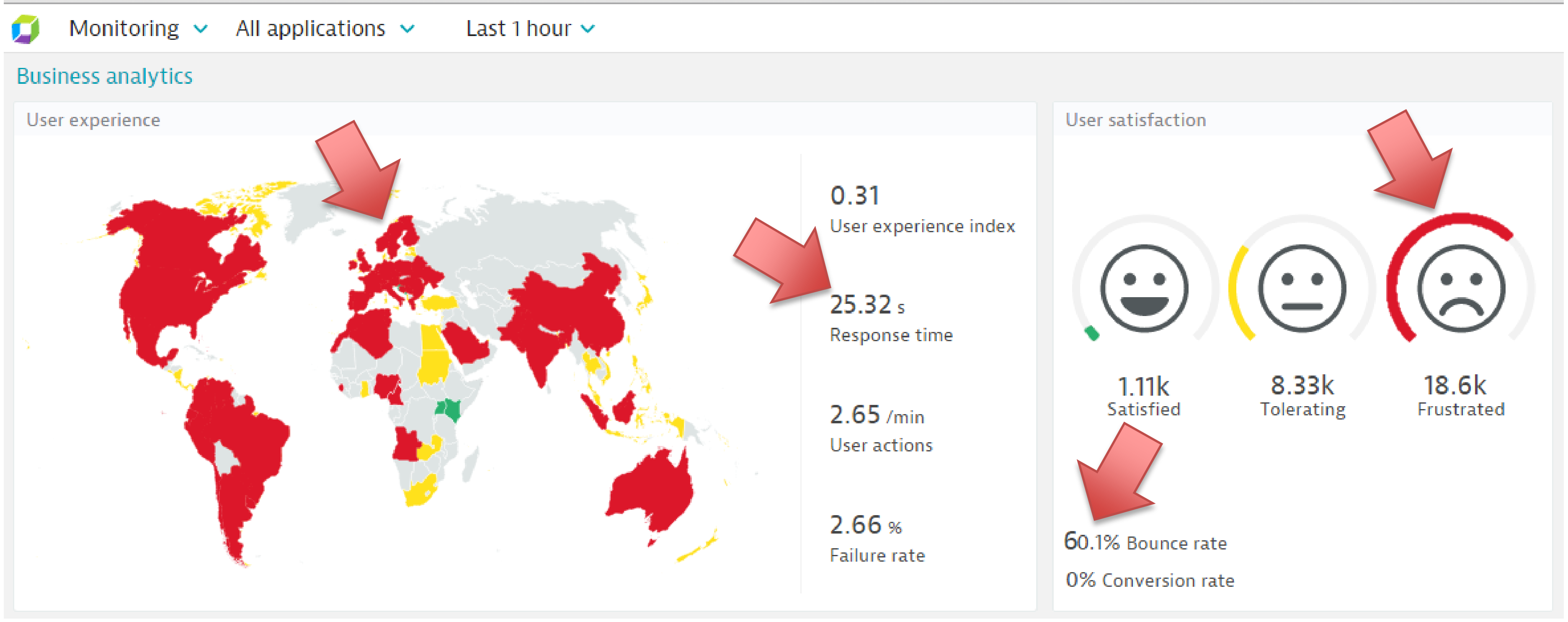
1.11k
Frustrated

20.1% Bounce rate

2% Conversion rate

@grabnerandi

Go live – 12:00 p.m.



Business analytics

User experience

0.31
User experience index

25.32 s
Response time

2.65 /min
User actions

2.66 %
Failure rate

User satisfaction

1.11k
Satisfied

8.33k
Tolerating

18.6k
Frustrated

60.1% Bounce rate

0% Conversion rate

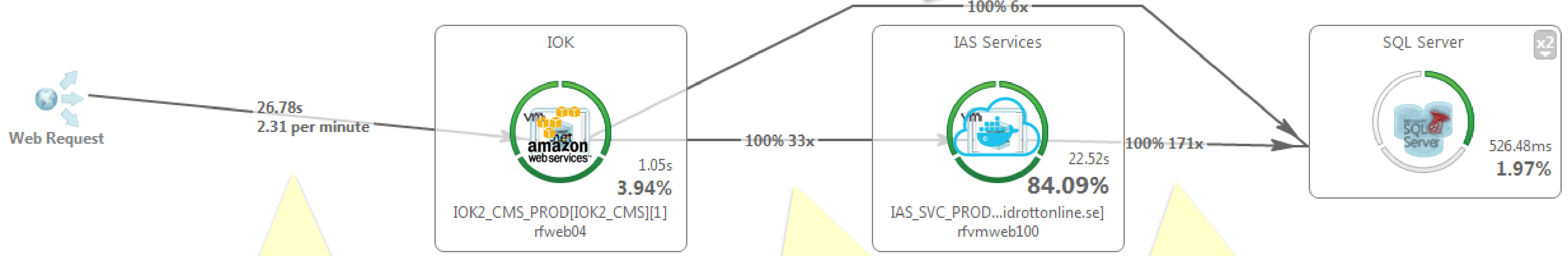
@grabnerandi

What Went Wrong?

Single search query end-to-end

Architecture Violation

Direct access to DB from frontend service



26.7s Load Time
5kB Payload

33! Service Calls
99kB - 3kB for each call!

171! Total SQL Count

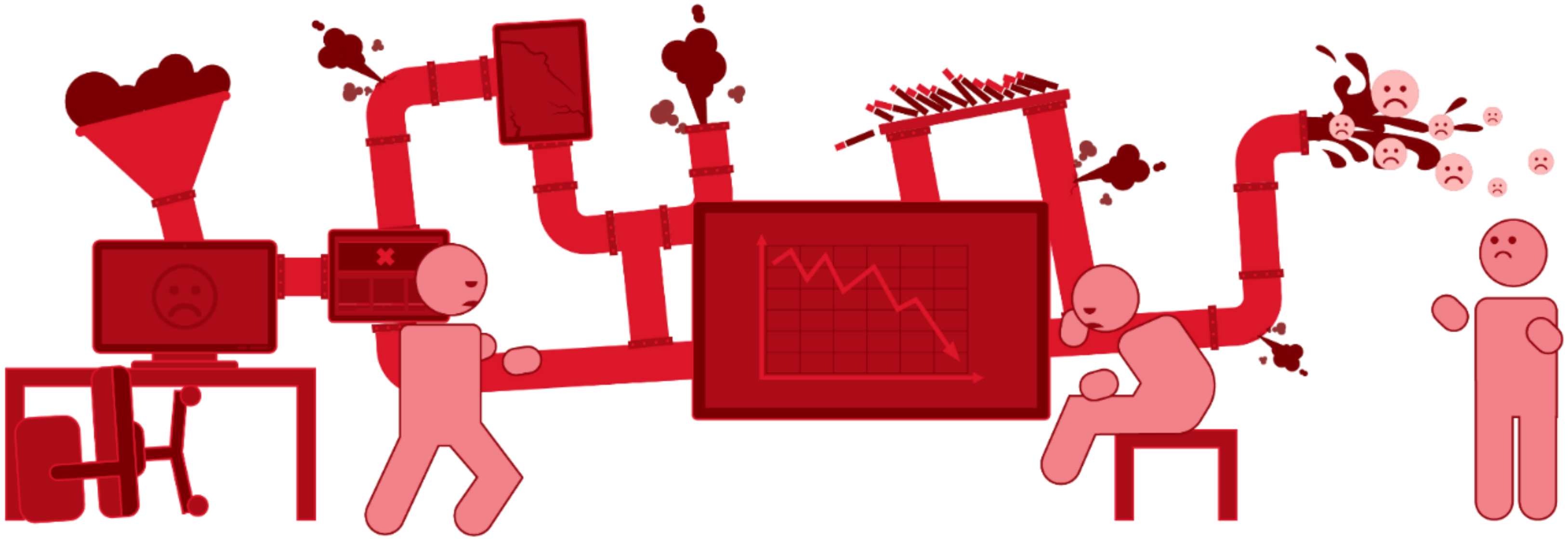
@grabnerandi

Understanding Code Complexity

- Existing 10 year old code & 3rd party
- Skills: Not everyone is a perf expert or born architect

From Monolith to Microservice

- Service usage in the End-to-End Scenarios?
- Will it scale? Or is it just a new monolith?



Understand Your End Users

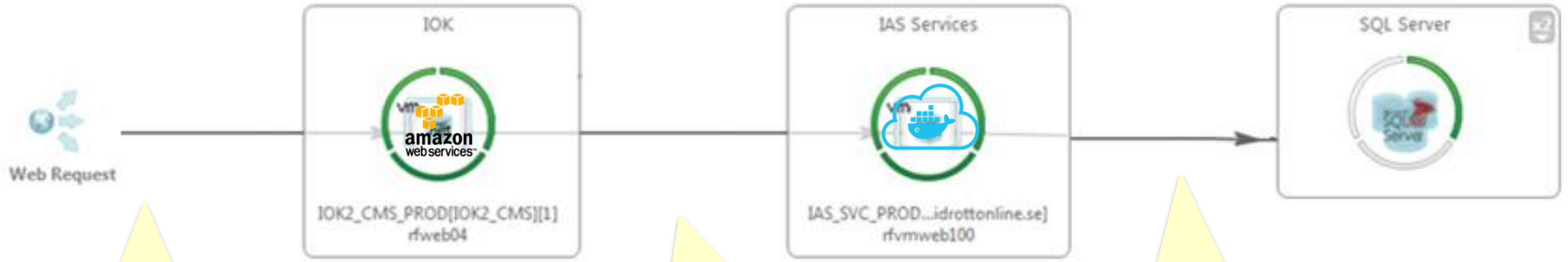
- What they like and what they DONT like!
- Its priority list & input for other teams, e.g: testing

Understand Deployment Complexity

- When moving to Cloud/Virtual: Costs, Latency …
- Old & new patterns, e.g: N+1 Query, Data

The fixed end-to-end use case

“Re-architect” vs. “Migrate” to Service-Orientation



2.5s (vs 26.7)
5kB Payload

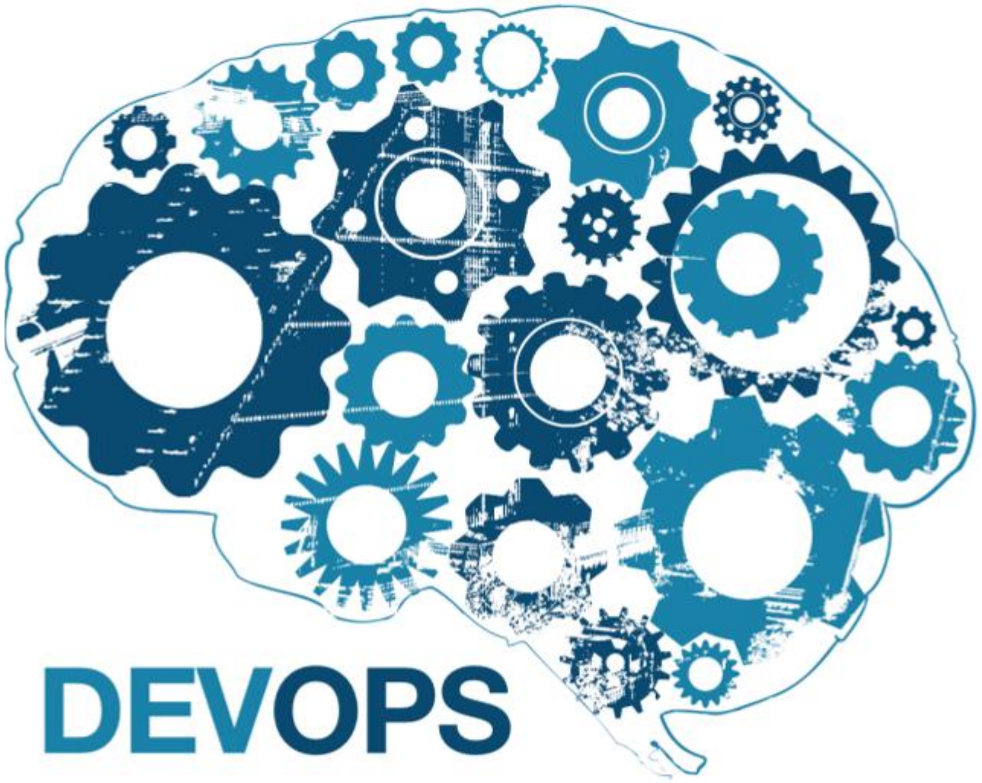
1! (vs 33!) Service Call
5kB (vs 99) Payload!

3! (vs 177)
Total SQL Count

@grabnerandi

1,3,5,7,9,11,13,15,17
Little C_T
STAY IN SCHOOL
Aa Bb Cc Dd Ee
Ff Gg Hh Ii Jj Kk
Ll Mm Nn Oo Pp
Qq Rr Ss Tt Uu
Vv Ww Xx Yy Zz
TE_CH_R
7x2=14
5x3=15
8x4=32
a²+b²=c
E=MC²
Home Work
6+0=6
15-2=13
10x3=30
12 cm
20CM
50CM
8CM
5CM

You ***measure it***! from Dev (to) Ops



@grabnerandi

Continuous Innovation and Optimization

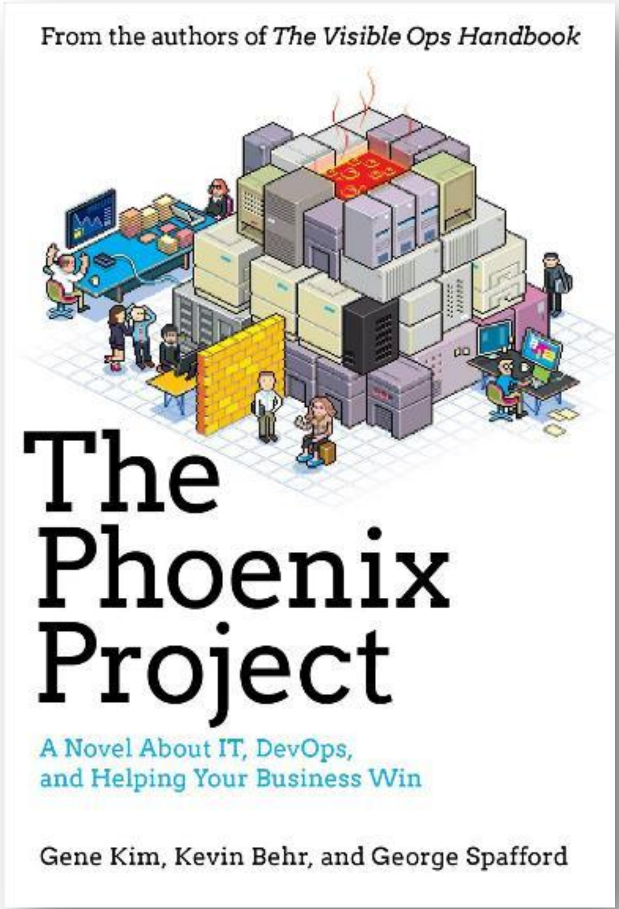
Scenario: Monolithic App with 2 Key Features

Use Case Tests and Monitors			Service & App Metrics				Ops		
Build #	Use Case	Stat	# APICalls	# SQL	Payload	CPU	#ServInst	Usage	RT
Build 17	testNewsAlert	OK	1	5	2kb	70ms	1	0.5%	7.2s
	testSearch	OK	1	35	5kb	120ms	1	63%	5.2s

Re-architecture into „Services“ + Performance Fixes

Build #	Use Case	Stat	# APICalls	# SQL	Payload	CPU	#ServInst	Usage	RT
Build 25	testNewsAlert	OK	1	4	1kb	60ms			
	testSearch	OK	34	171	104kb	550ms			
Build 26	testNewsAlert	OK	1	4	1kb	60ms	1	0.6%	3.2s
	testSearch	OK	2	3	10kb	150ms	6	75%	2.5s
Build 35	~~testNewsAlert~~	-	-	-	-	-	-	-	-
	testSearch	OK	2	3	7kb	100ms	4	80%	2.0s

Where to Start?
Where to Go?

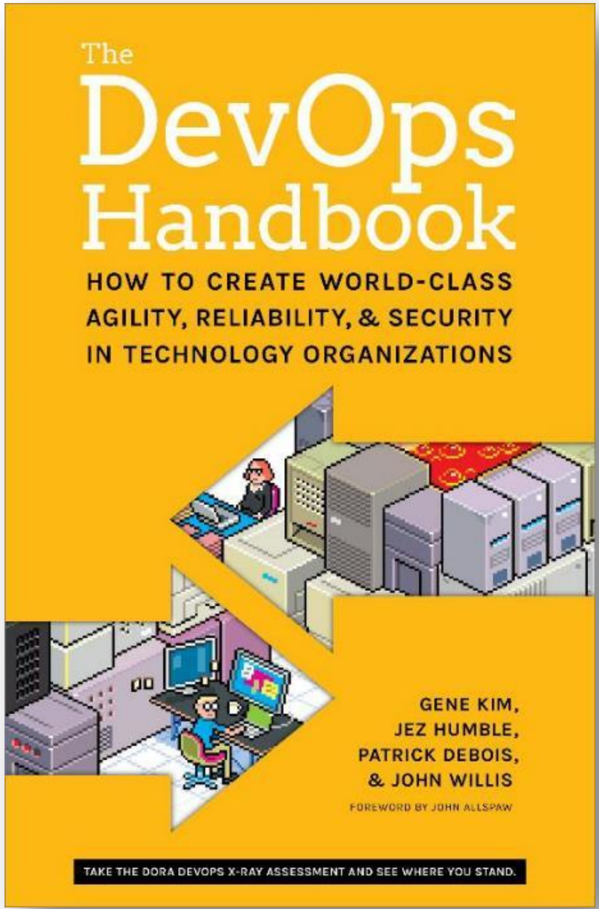


From the authors of *The Visible Ops Handbook*

The Phoenix Project

A Novel About IT, DevOps, and Helping Your Business Win

Gene Kim, Kevin Behr, and George Spafford



The DevOps Handbook

HOW TO CREATE WORLD-CLASS AGILITY, RELIABILITY, & SECURITY IN TECHNOLOGY ORGANIZATIONS

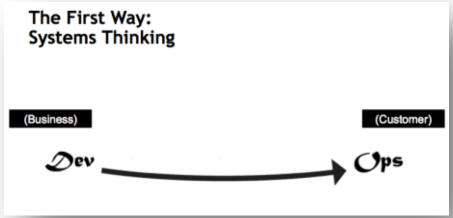
GENE KIM, JEZ HUMBLE, PATRICK DEBOIS, & JOHN WILLIS

FOREWORD BY JOHN ALLSPAW

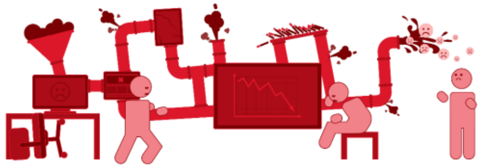
TAKE THE DORA DEVOPS X-RAY ASSESSMENT AND SEE WHERE YOU STAND.

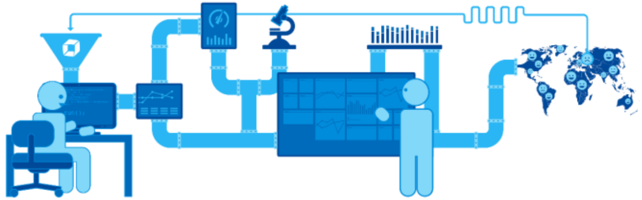
@grabnerandi

Ensure Success in The First Way



„Always seek to ***Increase Flow***“





Removing Bottlenecks

Shift-Left Quality

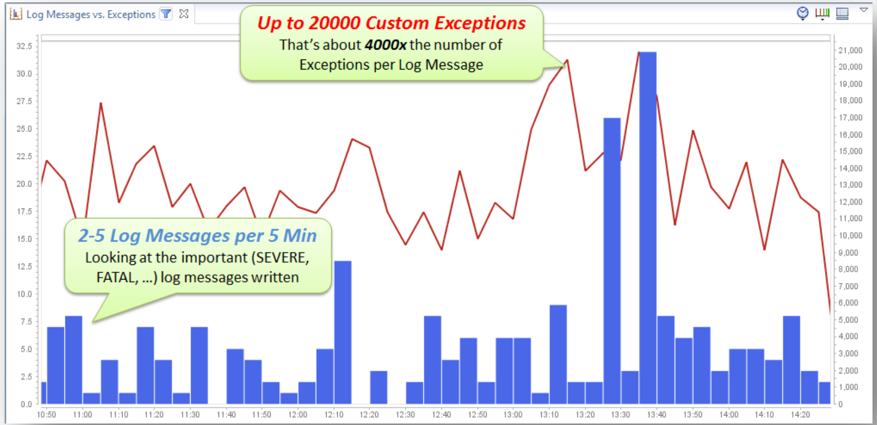
Reduce Code Complexity

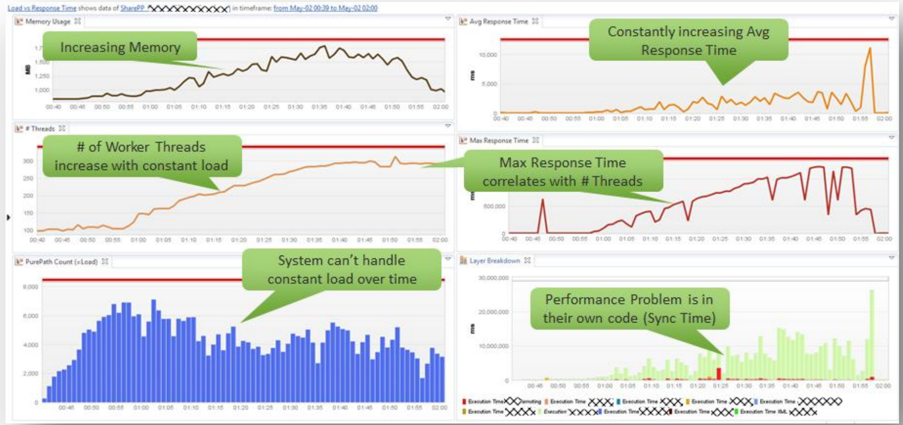
Eliminating Technical Debt

Enable Successful Cloud & Miroservices Migration

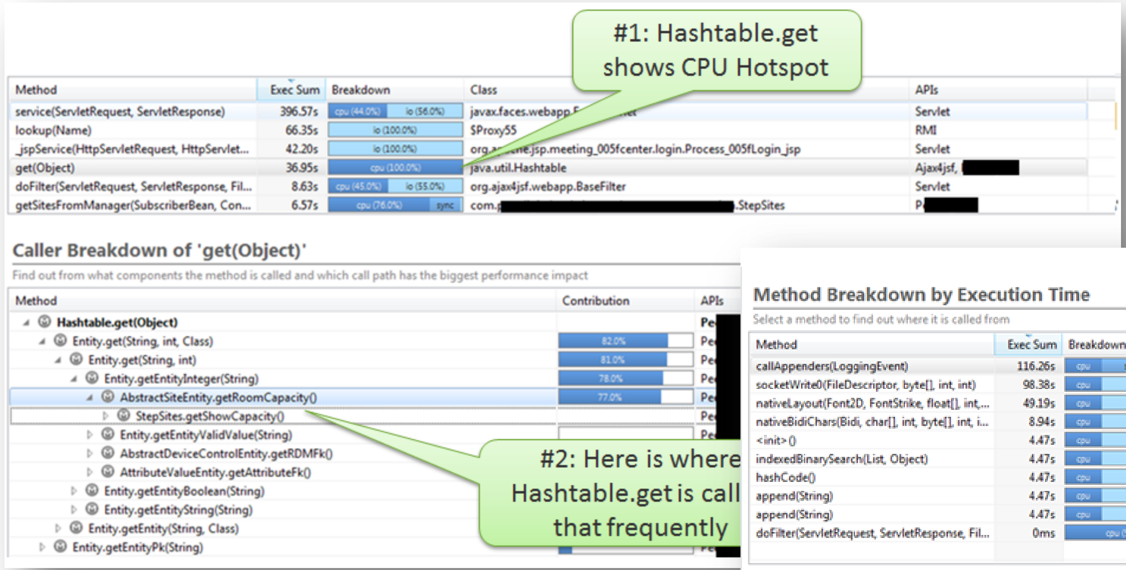
Manual Code/Architectural Bottleneck Detection

- Blog & YouTube Tutorial:
  - http://apmblog.dynatrace.com/2016/06/23/automatic-problem-detection-with-dynatrace/
  - http://bit.ly/dttutorials
- Metrics
  - # SQL, # of Same SQLs, # Threads, # Web Service/API Calls # Exceptions, # of Logs
  - # Bytes Transferred, Total Page Load, # of JavaScript/CSS/Images ...





Automatic Bottleneck *Root Cause* Information



#1: Hashtable.get shows CPU Hotspot

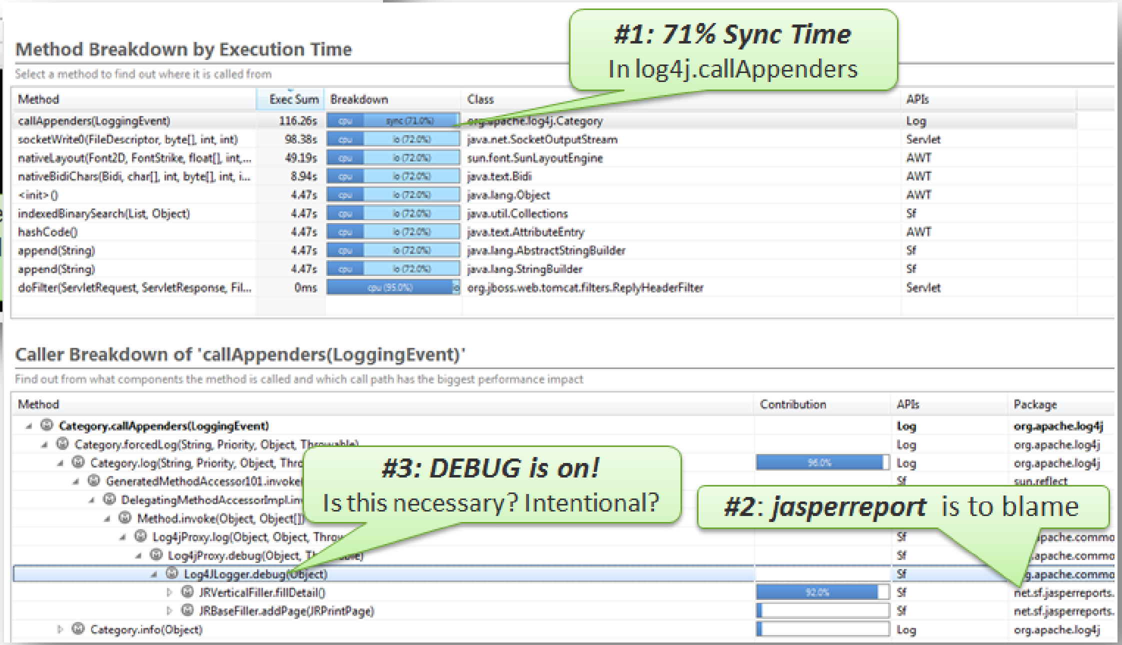
Method	Exec Sum	Breakdown	Class	APIs
service(ServletRequest, ServletResponse)	396.57s	cpu (44.0%) io (56.0%)	javax.faces.webapp.F	Servlet
lookup(Name)	66.35s	io (100.0%)	$Proxy55	RMI
_jspService(HttpServletRequest, HttpServlet...	42.20s	io (100.0%)	org.a…che.jsp.meeting_005fcenter.login.Process_005fLogin_jsp	Servlet
get(Object)	36.95s	cpu (100.0%)	java.util.Hashtable	Ajax4jsf, [redacted]
doFilter(ServletRequest, ServletResponse, Fil...	8.63s	cpu (45.0%) io (55.0%)	org.ajax4jsf.webapp.BaseFilter	Servlet
getSitesFromManager(SubscriberBean, Con...	6.57s	cpu (76.0%) sync	com.p[redacted].StepSites	P[redacted]

Caller Breakdown of 'get(Object)'

Find out from what components the method is called and which call path has the biggest performance impact

Method	Contribution	APIs
Hashtable.get(Object)		Pe [redacted]
Entity.get(String, int, Class)	82.0%	Pe
Entity.get(String, int)	81.0%	Pe
Entity.getEntityInteger(String)	78.0%	Pe
AbstractSiteEntity.getRoomCapacity()	77.0%	Pe
StepSites.getShowCapacity()		Pe
Entity.getEntityValidValue(String)		Pe
AbstractDeviceControlEntity.getRDMFk()		
AttributeValueEntity.getAttributeFk()		
Entity.getEntityBoolean(String)		
Entity.getEntityString(String)		
Entity.getEntity(String, Class)		
Entity.getEntityPk(String)		Pe

#2: Here is where Hashtable.get is call that frequently



Method Breakdown by Execution Time

Select a method to find out where it is called from

#1: 71% Sync Time
In log4j.callAppenders

Method	Exec Sum	Breakdown	Class	APIs
callAppenders(LoggingEvent)	116.26s	cpu sync (71.0%)	org.apache.log4j.Category	Log
socketWrite0(FileDescriptor, byte[], int, int)	98.38s	cpu io (72.0%)	java.net.SocketOutputStream	Servlet
nativeLayout(Font2D, FontStrike, float[], int,...	49.19s	cpu io (72.0%)	sun.font.SunLayoutEngine	AWT
nativeBidiChars(Bidi, char[], int, byte[], int, i...	8.94s	cpu io (72.0%)	java.text.Bidi	AWT
<init> ()	4.47s	cpu io (72.0%)	java.lang.Object	AWT
indexedBinarySearch(List, Object)	4.47s	cpu io (72.0%)	java.util.Collections	Sf
hashCode()	4.47s	cpu io (72.0%)	java.text.AttributeEntry	AWT
append(String)	4.47s	cpu io (72.0%)	java.lang.AbstractStringBuilder	Sf
append(String)	4.47s	cpu io (72.0%)	java.lang.StringBuilder	Sf
doFilter(ServletRequest, ServletResponse, Fil...	0ms	cpu (95.0%) io	org.jboss.web.tomcat.filters.ReplyHeaderFilter	Servlet

Caller Breakdown of 'callAppenders(LoggingEvent)'

Find out from what components the method is called and which call path has the biggest performance impact

Method	Contribution	APIs	Package
Category.callAppenders(LoggingEvent)		Log	org.apache.log4j
Category.forcedLog(String, Priority, Object, Throwable)		Log	org.apache.log4j
Category.log(String, Priority, Object, Thro	96.0%	Log	org.apache.log4j
GeneratedMethodAccessor101.invoke		Sf	sun.reflect
DelegatingMethodAccessorImpl.in			
Method.invoke(Object, Object[])			
Log4jProxy.log(Object, Object, Thro		Sf	apache.commo
Log4jProxy.debug(Object, T		Sf	.apache.commo
Log4JLogger.debug(Object)		Sf	g.apache.commo
JRVerticalFiller.fillDetail()	92.0%	Sf	net.sf.jasperreports.
JRBaseFiller.addPage(JRPrintPage)		Sf	net.sf.jasperreports.
Category.info(Object)		Log	org.apache.log4j

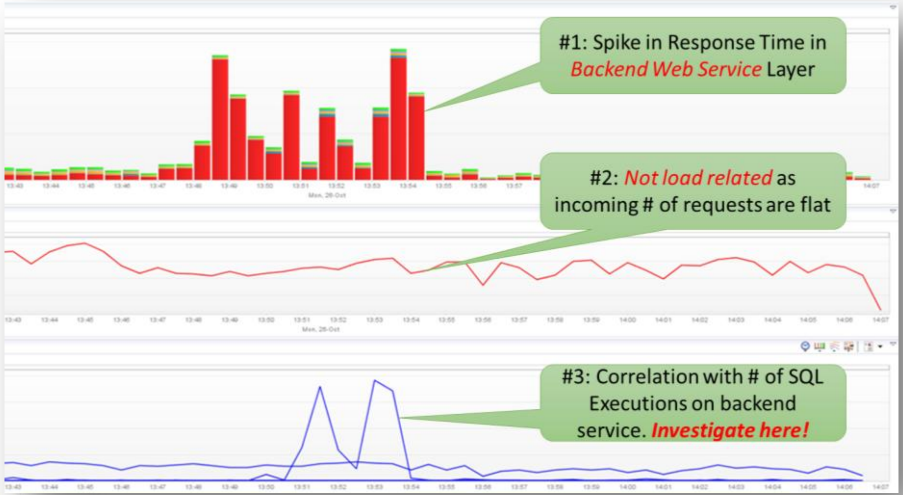
#3: DEBUG is on!
Is this necessary? Intentional?

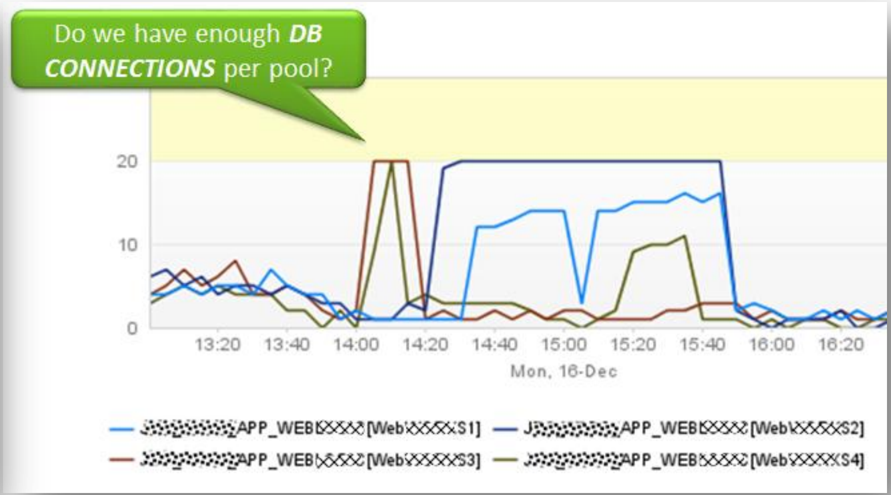
#2: ***jasperreport*** is to blame

dynatrace

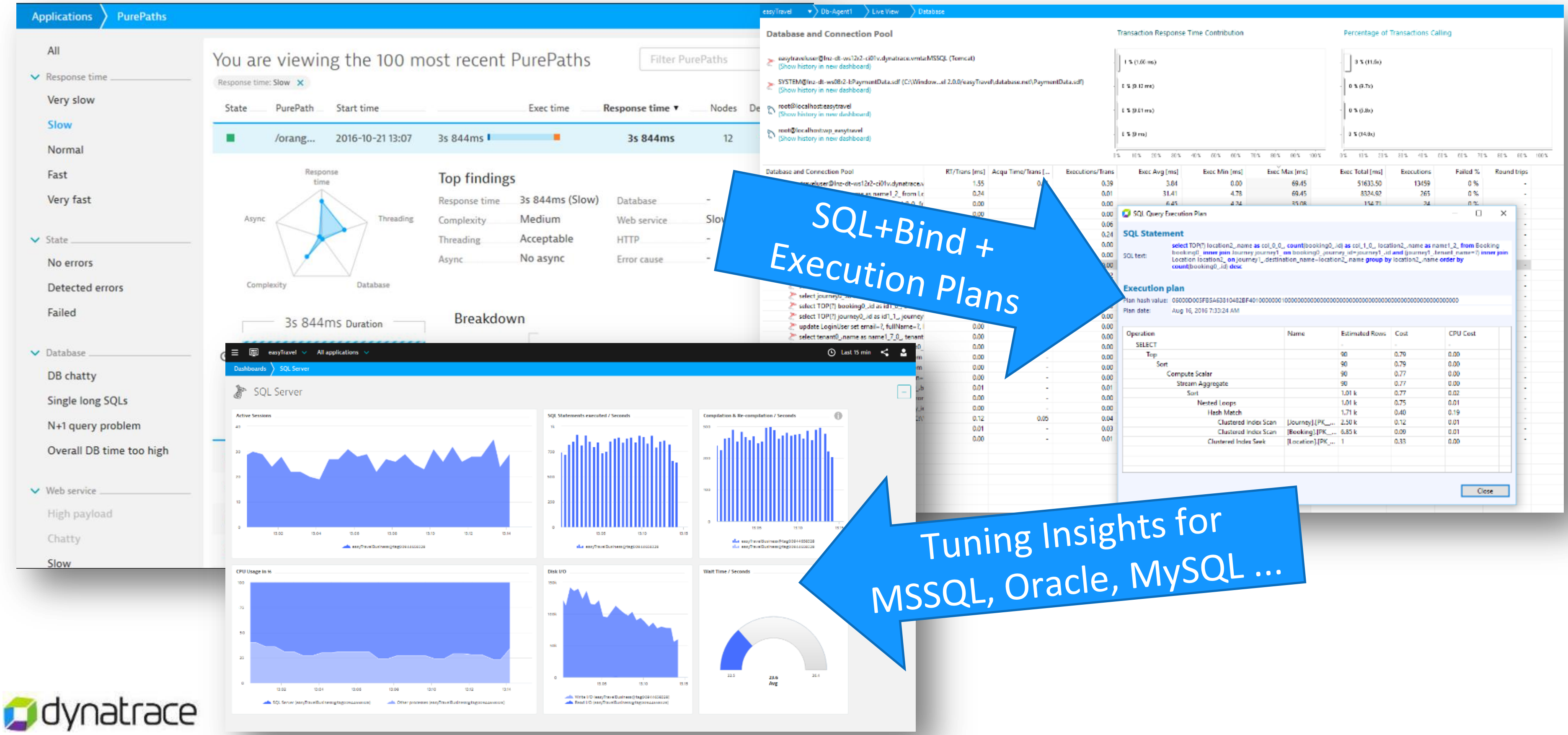
Manual Database Bottleneck Detection

- Blog & YouTube Tutorial:
  - http://apmblog.dynatrace.com/2016/02/18/diagnosing-java-hotspots/
  - http://bit.ly/dttutorials -> Database Diagnostics
- Patterns
  - N+1 Query, Unprepared SQL, Slow SQL, Database Cache, Indices, Loading Too Much Data ...





Automated Database Bottleneck Detection



SQL+Bind + Execution Plans
Tuning Insights for MSSQL, Oracle, MySQL ...

Applications > PurePaths

- All
- Response time
  - Very slow
  - Slow
  - Normal
  - Fast
  - Very fast
- State
  - No errors
  - Detected errors
  - Failed
- Database
  - DB chatty
  - Single long SQLs
  - N+1 query problem
  - Overall DB time too high
- Web service
  - High payload
  - Chatty
  - Slow

You are viewing the 100 most recent PurePaths

Response time: Slow ✕

State	PurePath	Start time	Exec time	Response time ▾	Nodes
🟩	/orang...	2016-10-21 13:07	3s 844ms	3s 844ms	12

Top findings

Response time	3s 844ms (Slow)	Database	-
Complexity	Medium	Web service	Slo
Threading	Acceptable	HTTP	-
Async	No async	Error cause	-

3s 844ms Duration

Breakdown

easyTravel > Db-Agent1 > Live View > Database

Database and Connection Pool

Transaction Response Time Contribution

Percentage of Transactions Calling

SQL Query Execution Plan

SQL Statement

SQL text: select TOP(?) location2_.name as col_0_0_, count(booking0_.id) as col_1_0_, location2_.name as name1_2_ from Booking booking0_ inner join Journey journey1_ on booking0_.journey_id=journey1_.id and (journey1_.tenant_name=?) inner join Location location2_ on journey1_.destination_name=location2_.name group by location2_.name order by count(booking0_.id) desc

Execution plan

Plan hash value: 06000D005FB5A63810482BF4010000000100000000000000000000000000000000000000000000

Plan date: Aug 16, 2016 7:33:24 AM

Operation	Name	Estimated Rows	Cost	CPU Cost
SELECT		-	-	-
Top		90	0.79	0.00
Sort		90	0.79	0.00
Compute Scalar		90	0.77	0.00
Stream Aggregate		90	0.77	0.00
Sort		1.01 k	0.77	0.02
Nested Loops		1.01 k	0.75	0.01
Hash Match		1.71 k	0.40	0.19
Clustered Index Scan	[Journey].[PK_...	2.50 k	0.12	0.01
Clustered Index Scan	[Booking].[PK_...	6.85 k	0.09	0.01
Clustered Index Seek	[Location].[PK_...	1	0.33	0.00

Close

easyTravel · All applications · Last 15 min

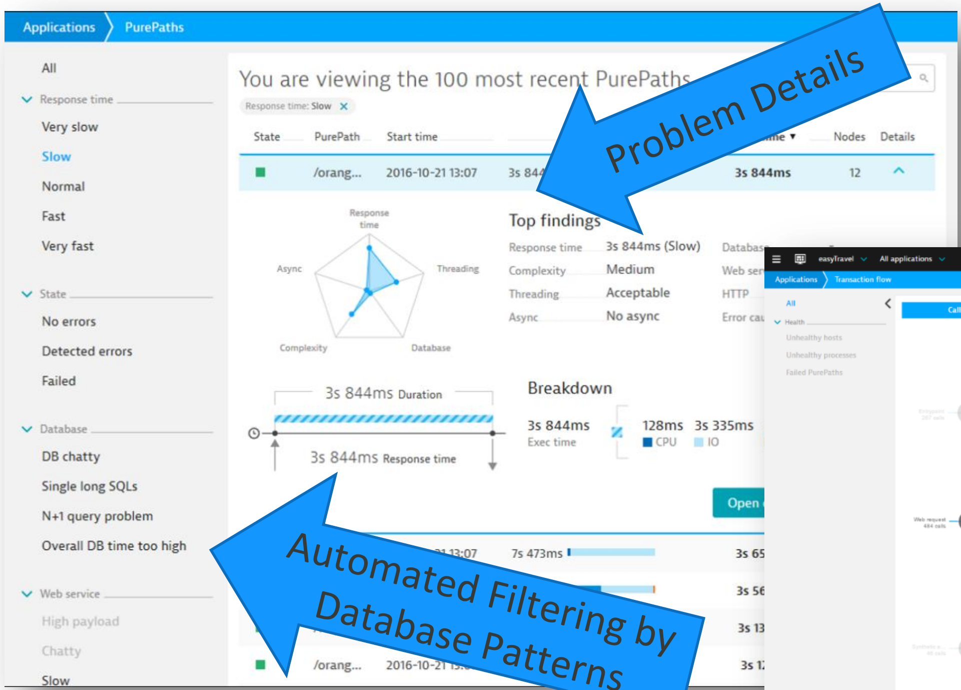
Dashboards > SQL Server

SQL Server

Active Sessions · SQL Statements executed / Seconds · Compilation & Re-compilation / Seconds · CPU Usage in % · Disk I/O · Wait Time / Seconds

dynatrace

Automated Code/Archiecture Bottleneck Detection



Applications 〉 PurePaths

All

Response time
- Very slow
- Slow
- Normal
- Fast
- Very fast

State
- No errors
- Detected errors
- Failed

Database
- DB chatty
- Single long SQLs
- N+1 query problem
- Overall DB time too high

Web service
- High payload
- Chatty
- Slow

You are viewing the 100 most recent PurePaths

Response time: Slow ✕

State	PurePath	Start time			Nodes	Details
🟩	/orang...	2016-10-21 13:07	3s 844	3s 844ms	12	^

Top findings

Response time	3s 844ms (Slow)	Databas
Complexity	Medium	Web ser
Threading	Acceptable	HTTP
Async	No async	Error cau

3s 844ms Duration

3s 844ms Response time

Breakdown

3s 844ms Exec time

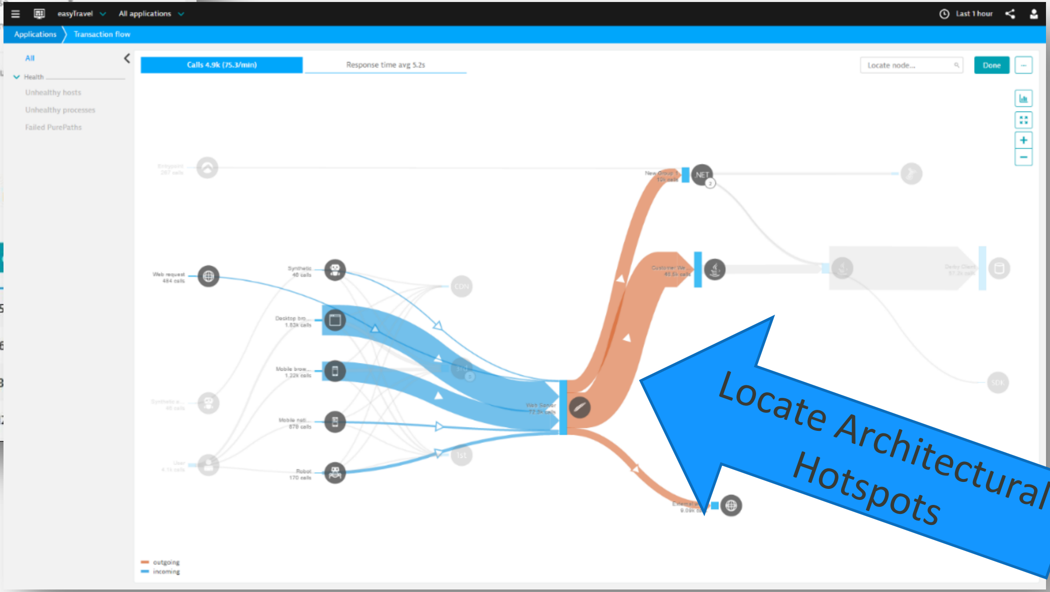
128ms CPU · 3s 335ms IO

Open

	21 13:07	7s 473ms	3s 65
			3s 56
			3s 13
🟩	/orang...	2016-10-21 13:0	3s 1

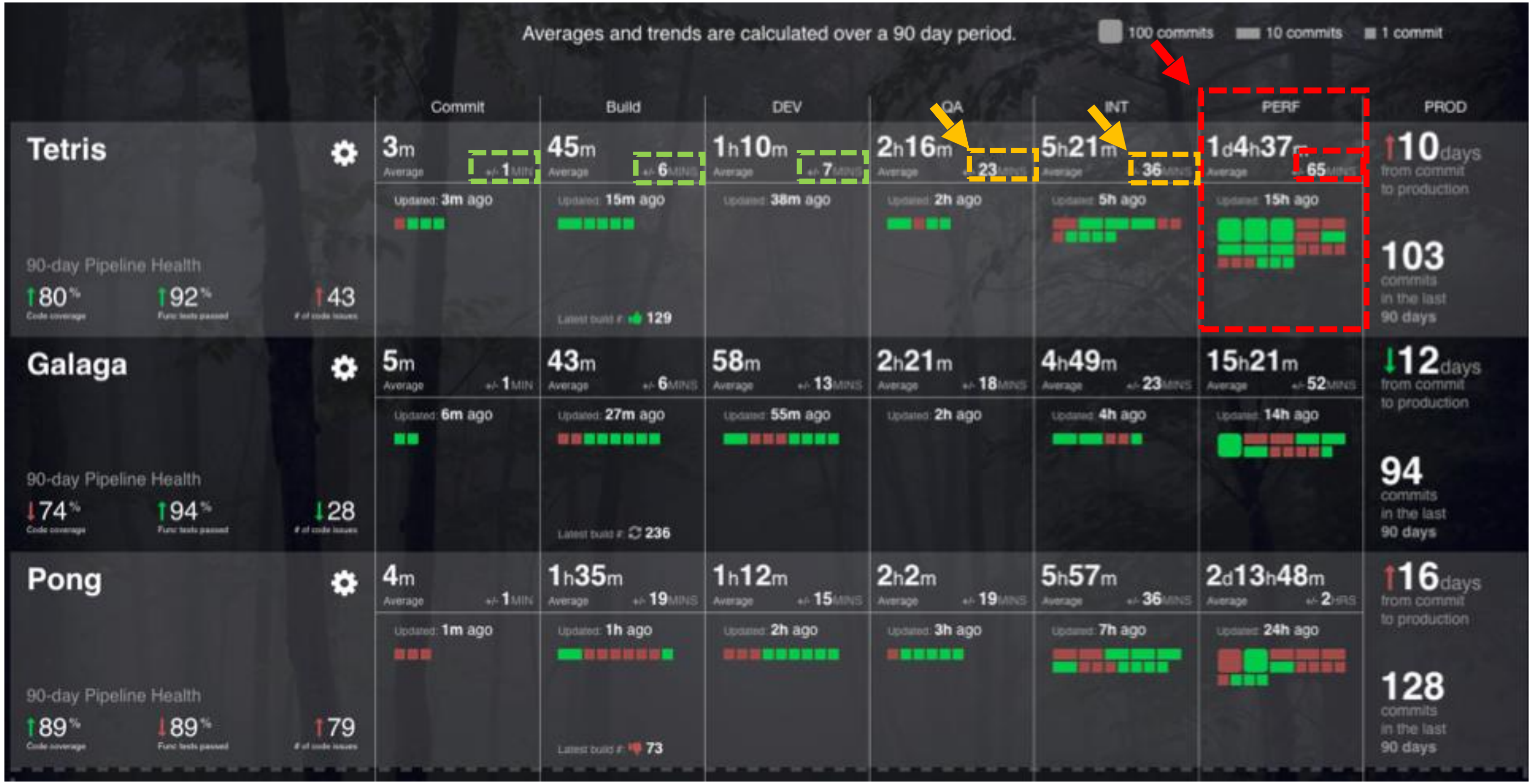
Problem Details

Automated Filtering by Database Patterns



Locate Architectural Hotspots

“To Deliver ***High Quality Working*** Software ***Faster***“



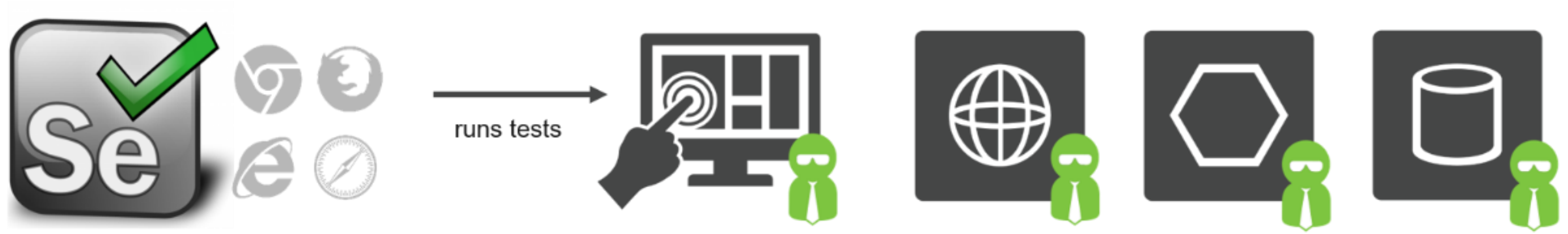
Averages and trends are calculated over a 90 day period.

100 commits · 10 commits · 1 commit

	Commit	Build	DEV	QA	INT	PERF	PROD
Tetris 90-day Pipeline Health ↑80% Code coverage · ↑92% Func tests passed · ↑43 # of code issues	3m Average +/- 1 MIN Updated: 3m ago	45m Average +/- 6 MINS Updated: 15m ago Latest build #: 129	1h10m Average +/- 7 MINS Updated: 38m ago	2h16m Average +/- 23 MINS Updated: 2h ago	5h21m Average +/- 36 MINS Updated: 5h ago	1d4h37m Average +/- 65 MINS Updated: 15h ago	↑10 days from commit to production 103 commits in the last 90 days
Galaga 90-day Pipeline Health ↓74% Code coverage · ↑94% Func tests passed · ↓28 # of code issues	5m Average +/- 1 MIN Updated: 6m ago	43m Average +/- 6 MINS Updated: 27m ago Latest build #: 236	58m Average +/- 13 MINS Updated: 55m ago	2h21m Average +/- 18 MINS Updated: 2h ago	4h49m Average +/- 23 MINS Updated: 4h ago	15h21m Average +/- 52 MINS Updated: 14h ago	↓12 days from commit to production 94 commits in the last 90 days
Pong 90-day Pipeline Health ↑89% Code coverage · ↓89% Func tests passed · ↑79 # of code issues	4m Average +/- 1 MIN Updated: 1m ago	1h35m Average +/- 19 MINS Updated: 1h ago Latest build #: 73	1h12m Average +/- 15 MINS Updated: 2h ago	2h2m Average +/- 19 MINS Updated: 3h ago	5h57m Average +/- 36 MINS Updated: 7h ago	2d13h48m Average +/- 2 HRS Updated: 24h ago	↑16 days from commit to production 128 commits in the last 90 days

„*We have to **Shift-Left Performance** to **Optimize Pipelines***“

http://apmblog.dynatrace.com/2016/10/04/scaling-continuous-delivery-shift-left-performance-to-improve-lead-time-pipeline-flow/



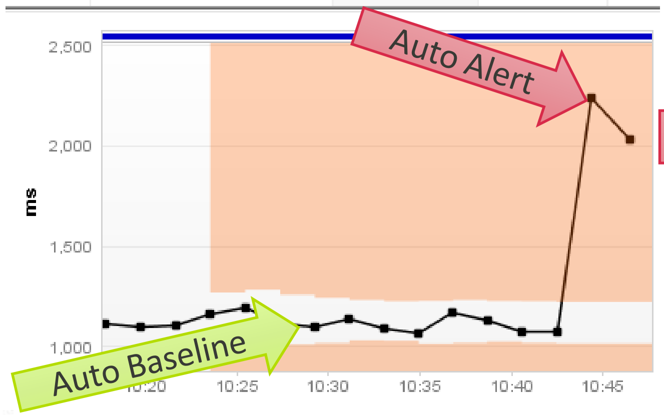
Selenium Server + Drivers

Application under test, instrumented with Dynatrace AppMon

= Functional Result (passed/failed)

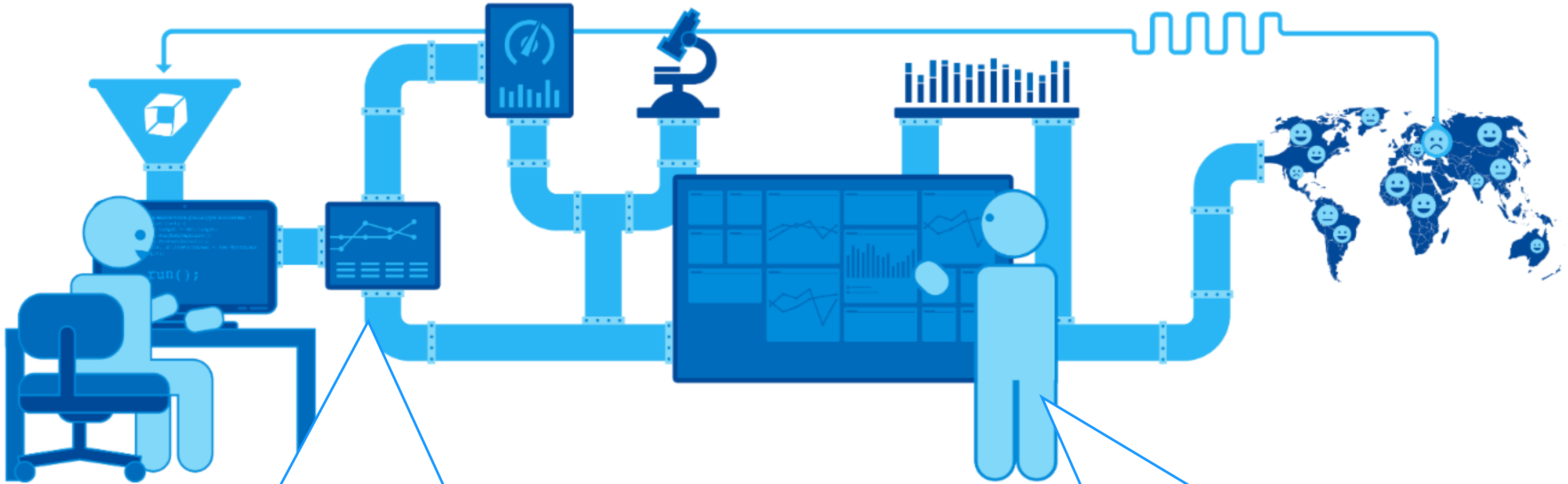
+ Web Performance Metrics (# of Images, # of JavaScript, Page Load Time, ...)

+ App Performance Metrics (# of SQL, # of Logs, # of API Calls, # of Exceptions ...)



Fail the build early!

Reduce Lead Time: Stop 80% of Performance Issues in your Integration Phase

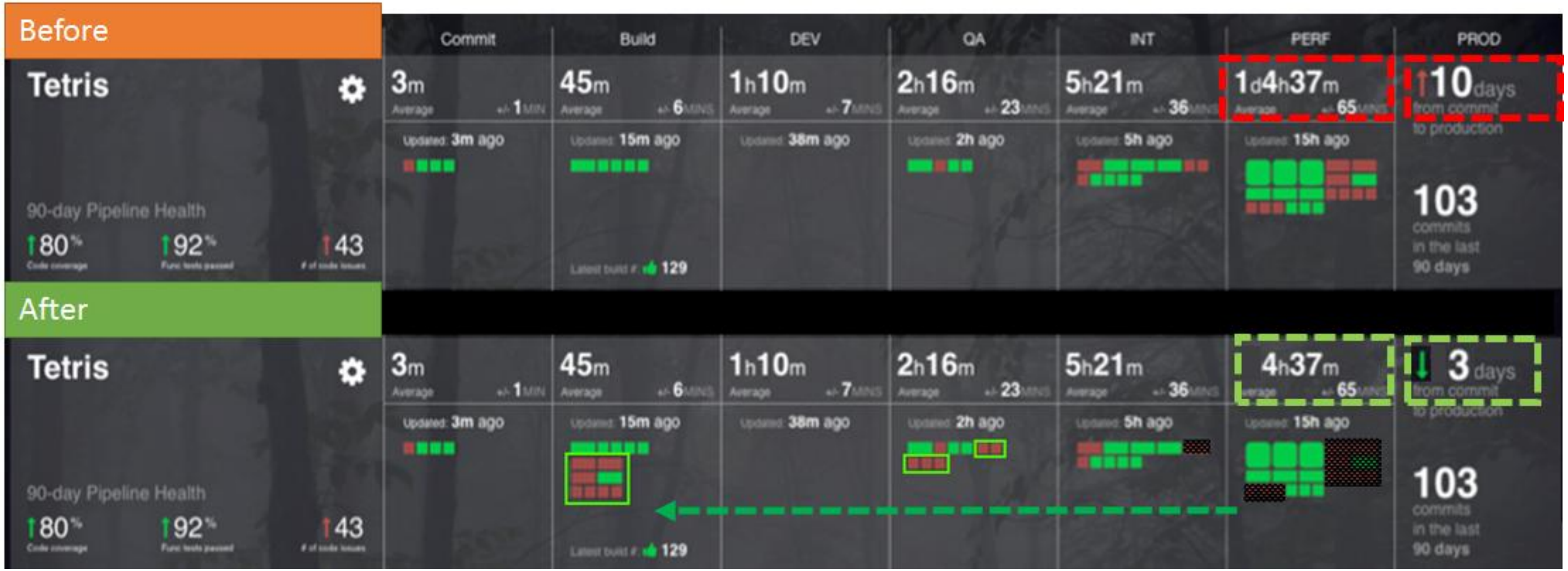


CI/CD: Test Automation (Selenium, Appium, Cucumber, Silk, ...) to ***detect functional and architectural*** (performance, scalabilty) regressions

Perf: Performance Test (JMeter, LoadRunner, Neotys, Silk, ...) to detect ***tough*** performance issues

dynatrace

Shift-Left Performance results in Reduced Lead Time powered by *Dynatrace Test Automation*



Before

Tetris

Commit	Build	DEV	QA	INT	PERF	PROD
3m Average +/- 1 MIN	45m Average +/- 6 MINS	1h10m Average +/- 7 MINS	2h16m Average +/- 23 MINS	5h21m Average +/- 36 MINS	1d4h37m Average +/- 65 MINS	10 days from commit to production
Updated: 3m ago	Updated: 15m ago	Updated: 38m ago	Updated: 2h ago	Updated: 5h ago	Updated: 15h ago	103 commits in the last 90 days
	Latest build # 129					

90-day Pipeline Health

80% Code coverage — 92% Func tests passed — 43 # of code issues

After

Tetris

Commit	Build	DEV	QA	INT	PERF	PROD
3m Average +/- 1 MIN	45m Average +/- 6 MINS	1h10m Average +/- 7 MINS	2h16m Average +/- 23 MINS	5h21m Average +/- 36 MINS	4h37m Average +/- 65 MINS	3 days from commit to production
Updated: 3m ago	Updated: 15m ago	Updated: 38m ago	Updated: 2h ago	Updated: 5h ago	Updated: 15h ago	103 commits in the last 90 days
	Latest build # 129					

90-day Pipeline Health

80% Code coverage — 92% Func tests passed — 43 # of code issues

http://apmblog.dynatrace.com/2016/10/04/scaling-continuous-delivery-shift-left-performance-to-improve-lead-time-pipeline-flow/

dynatrace

Faster Lead Times to User Value!
Results in Business Success!

Questions

Slides: slideshare.net/grabnerandi
Get Tools: bit.ly/dtpersonal
Watch: bit.ly/dttutorials
Follow Me: @grabnerandi
Read More: blog.dynatrace.com
Listen: http://bit.ly/pureperf
Mail: andreas.grabner@dynatrace.com

dynatrace

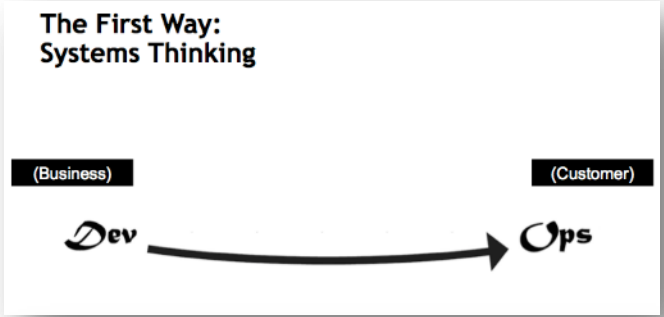
Andreas Grabner

Dynatrace Developer Advocate

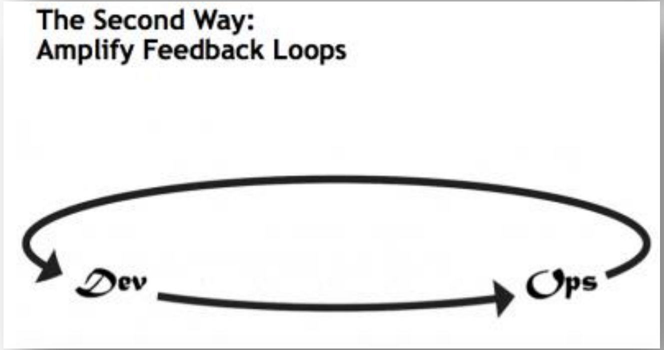
@grabnerandi

http://blog.dynatrace.com

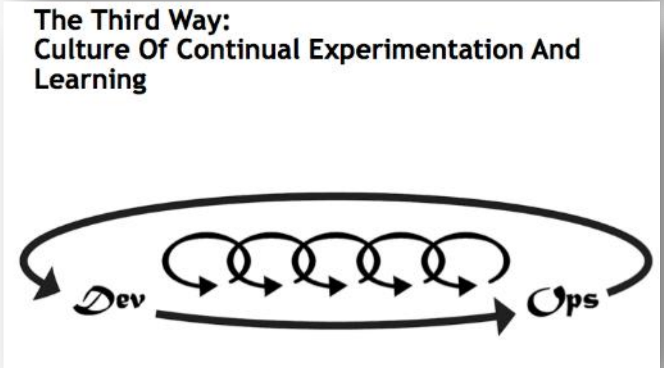
dynatrace



„Always seek to ***Increase Flow***"



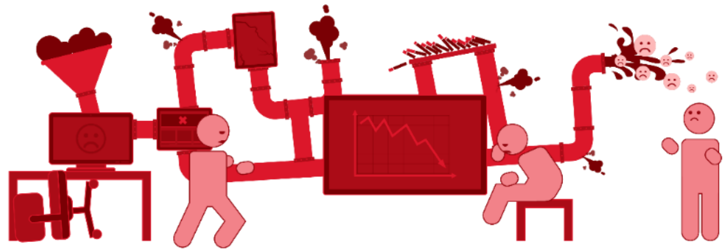
„Understand and ***Respond to Outcome***"



„Culture on ***Continual Experimentation***"

@grabnerandi

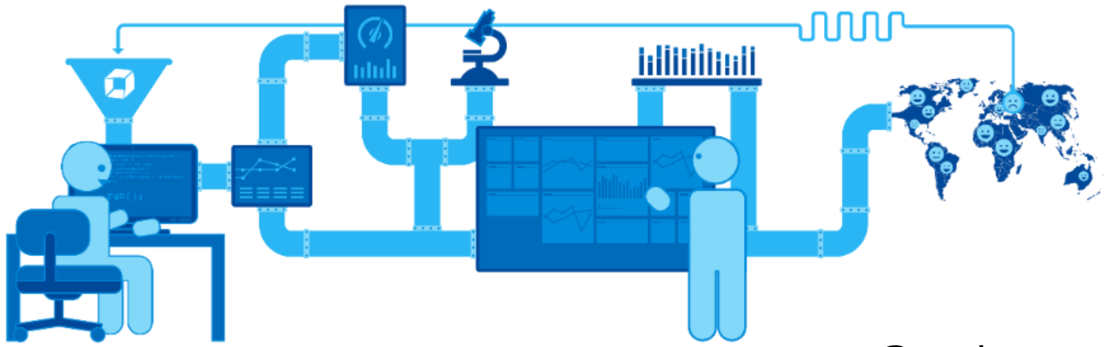
Increased *Flow of High Quality Value*



Break the Monolith
Infrastructure as Code
Migrate to Virtual/Cloud/PaaS

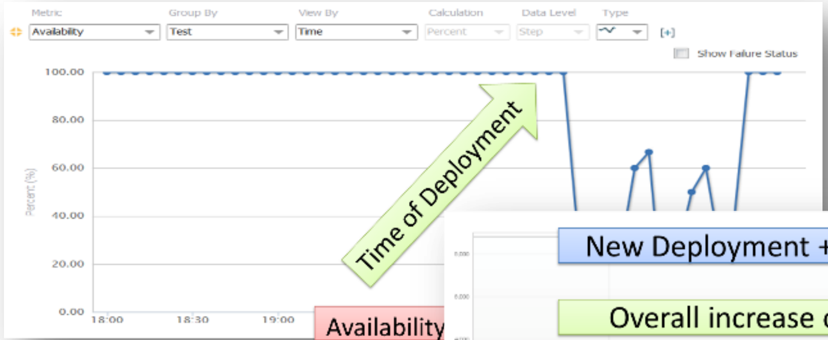
Remove Bottlenecks

Test Driven Development
Automated Deployments
Shift-Left Performance



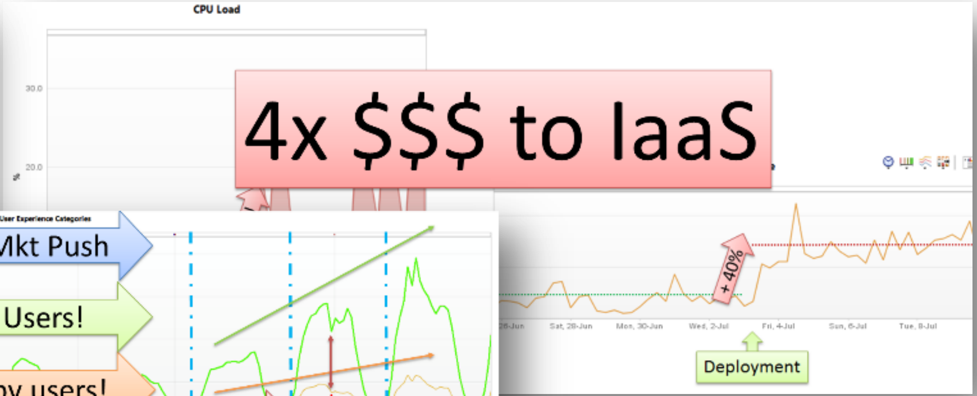
@grabnerandi

Fast *Response to Outcome: Address Deployment Impact*

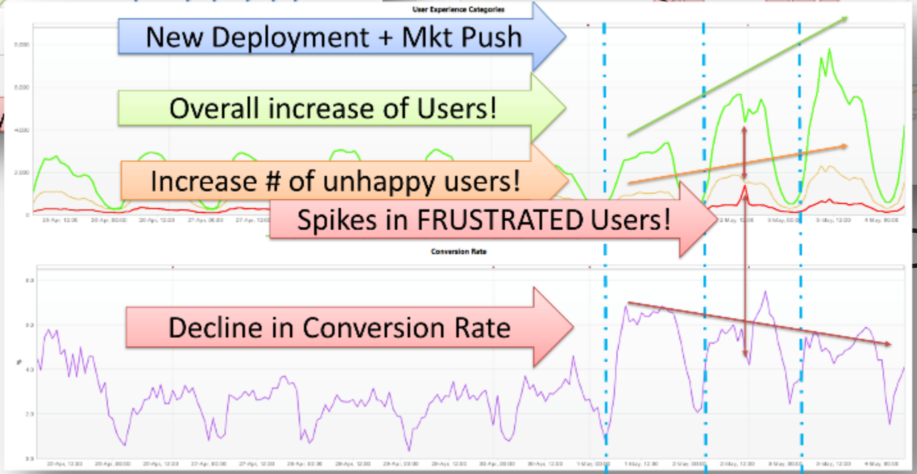


Metric: Availability
Group By: Test
View By: Time
Calculation: Percent
Data Level: Step

Availability



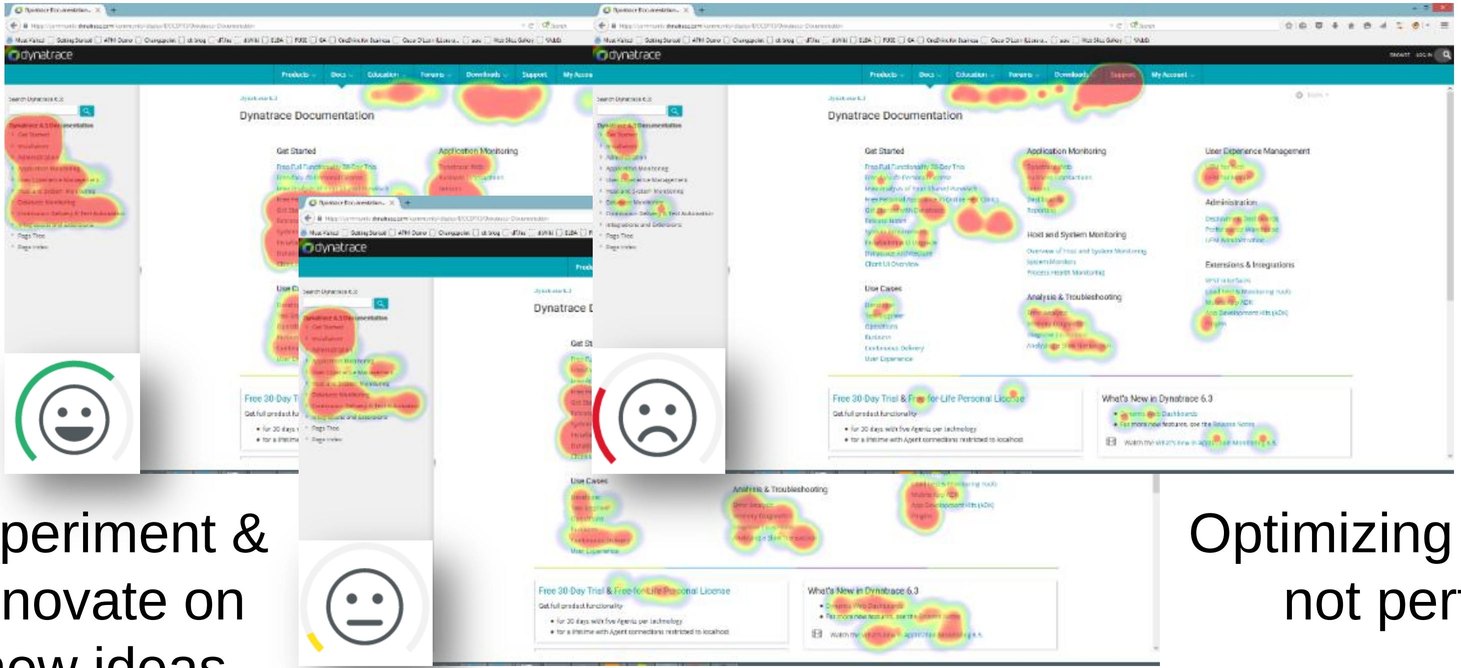
Costs and Efficiency



User Experience, Conversion Rate

@grabnerandi

Real User Feedback: Building the RIGHT thing RIGHT!


dynatrace
Dynatrace Documentation
Get Started
Application Monitoring
User Experience Management
Administration
Host and System Monitoring
Extensions & Integrations
Use Cases
Analysis & Troubleshooting
Free 30-Day Trial & Free-for-Life Personal License
What's New in Dynatrace 6.3

Removing what nobody needs

Experiment & innovate on new ideas

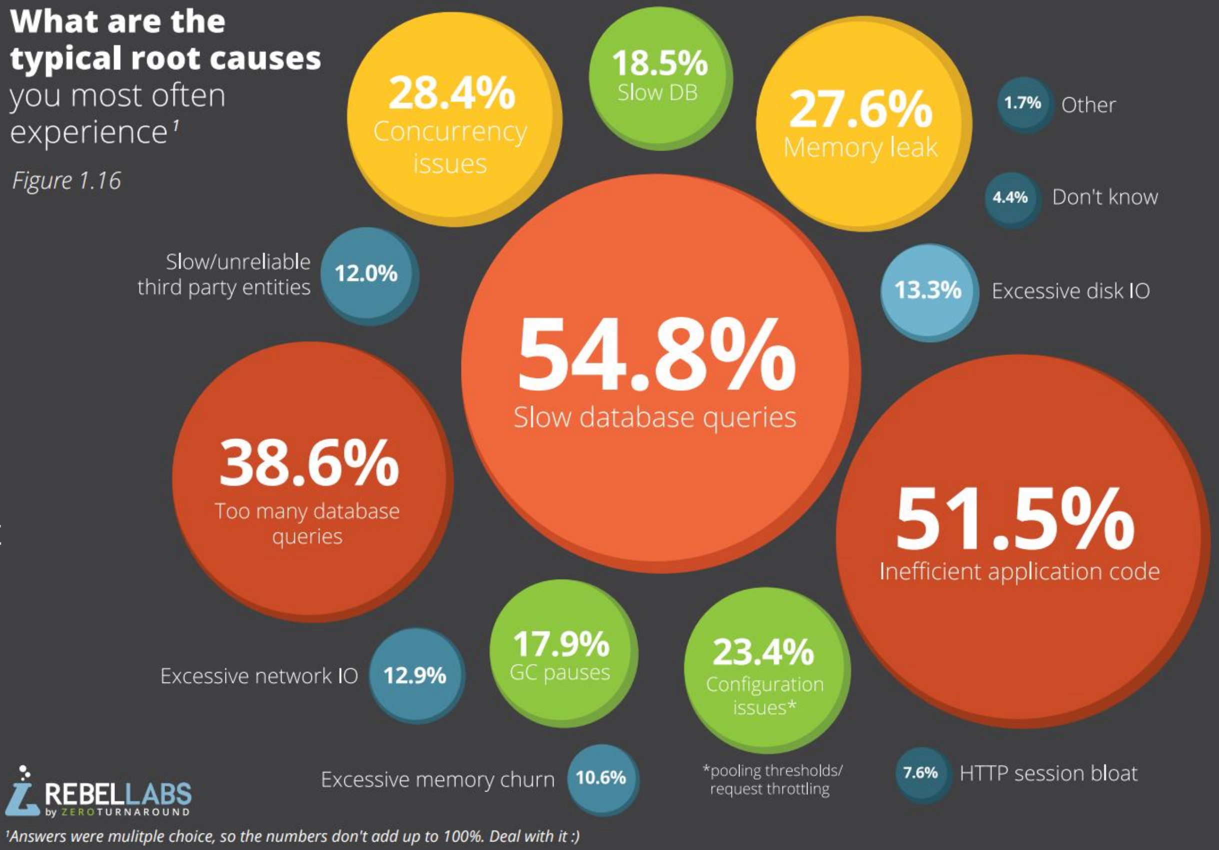
Optimizing what is not perfect

@grabnerandi

Remove Database Bottlenecks

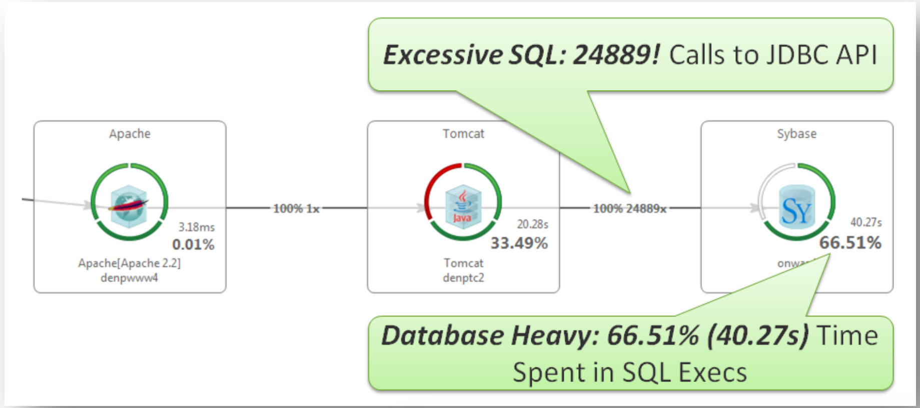
88%

cite the database as the most common challenge or issue with application performance



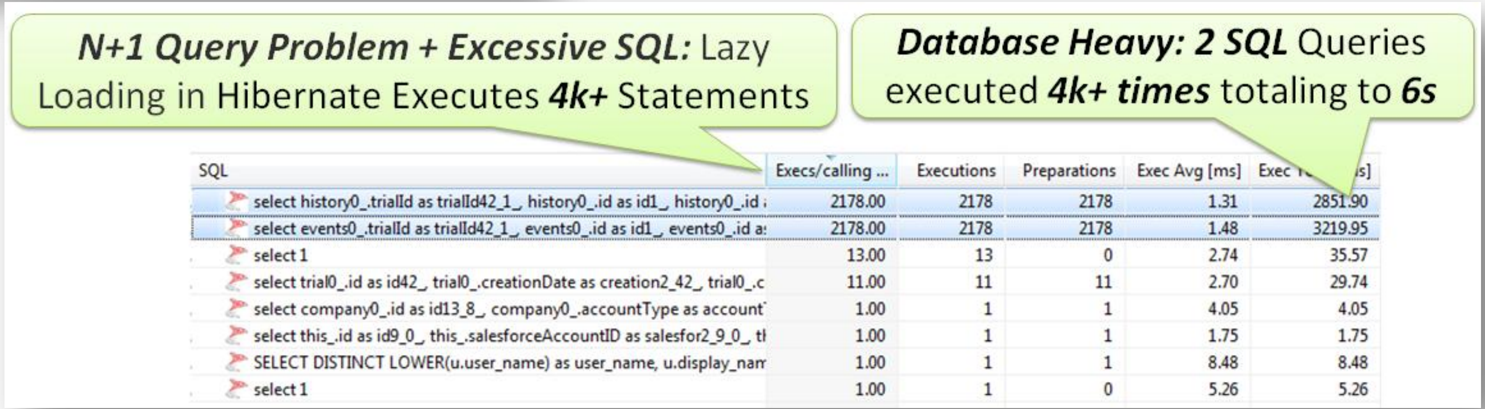
dynatrace

Automatic Bottleneck *Root Cause* Information



Excessive SQL: 24889! Calls to JDBC API

Database Heavy: 66.51% (40.27s) Time Spent in SQL Execs



N+1 Query Problem + Excessive SQL: Lazy Loading in Hibernate Executes ***4k+*** Statements

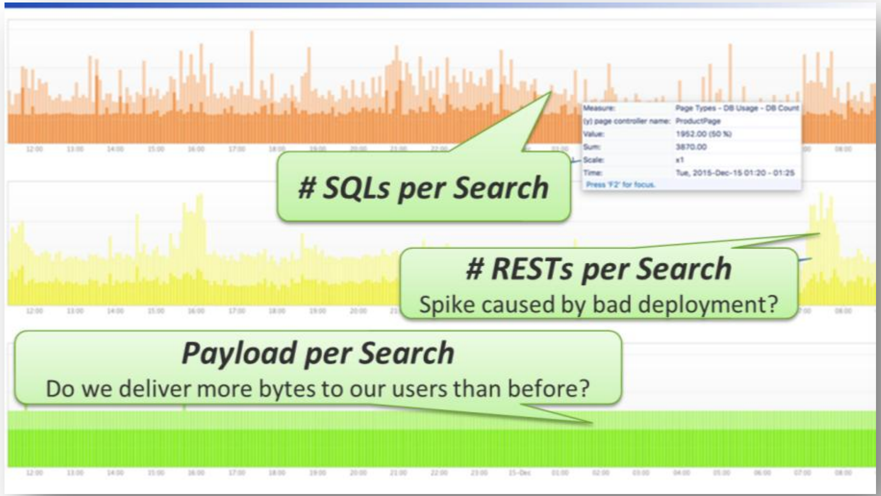
Database Heavy: 2 SQL Queries executed ***4k+ times*** totaling to ***6s***

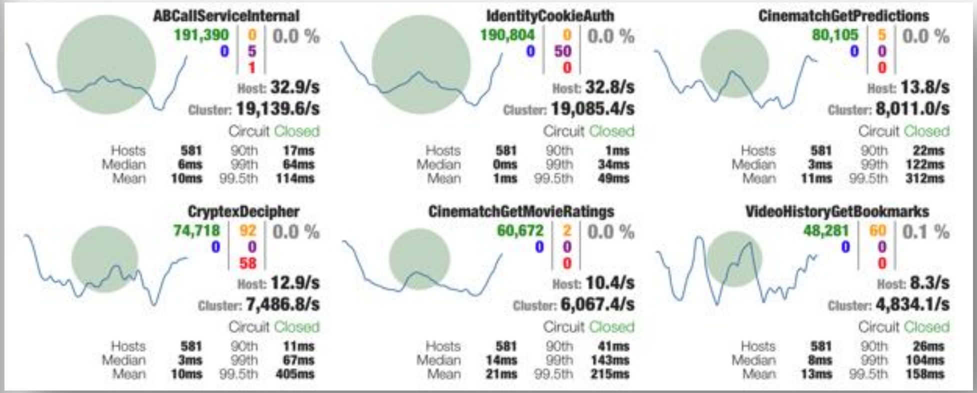
SQL	Execs/calling ...	Executions	Preparations	Exec Avg [ms]	Exec [illegible]s]
select history0_.trialId as trialId42_1_, history0_.id as id1_, history0_.id	2178.00	2178	2178	1.31	2851.90
select events0_.trialId as trialId42_1_, events0_.id as id1_, events0_.id a	2178.00	2178	2178	1.48	3219.95
select 1	13.00	13	0	2.74	35.57
select trial0_.id as id42_, trial0_.creationDate as creation2_42_, trial0_.c	11.00	11	11	2.70	29.74
select company0_.id as id13_8_, company0_.accountType as account	1.00	1	1	4.05	4.05
select this_.id as id9_0_, this_.salesforceAccountID as salesfor2_9_0_, tl	1.00	1	1	1.75	1.75
SELECT DISTINCT LOWER(u.user_name) as user_name, u.display_nam	1.00	1	1	8.48	8.48
select 1	1.00	1	0	5.26	5.26

dynatrace

Manual Service Bottleneck Detection

- Blogs:
  - http://apmblog.dynatrace.com/2016/06/08/diagnosing-common-bad-micro-service-call-patterns/
  - http://apmblog.dynatrace.com/2015/08/26/monolith-to-microservices-key-architectural-metrics-to-watch/
- Patterns
  - N+1, High Payload, Lack of Caching, Thread & Connection Pool Shortage, Excessive Async Calls





ABCallServiceInternal
191,390 | 0 0 | 0 5 1 | 0.0 %
Host: 32.9/s
Cluster: 19,139.6/s
Circuit Closed
Hosts 581 · Median 6ms · Mean 10ms · 90th 17ms · 99th 64ms · 99.5th 114ms

IdentityCookieAuth
190,804 | 0 0 | 0 50 0 | 0.0 %
Host: 32.8/s
Cluster: 19,085.4/s
Circuit Closed
Hosts 581 · Median 0ms · Mean 1ms · 90th 1ms · 99th 34ms · 99.5th 49ms

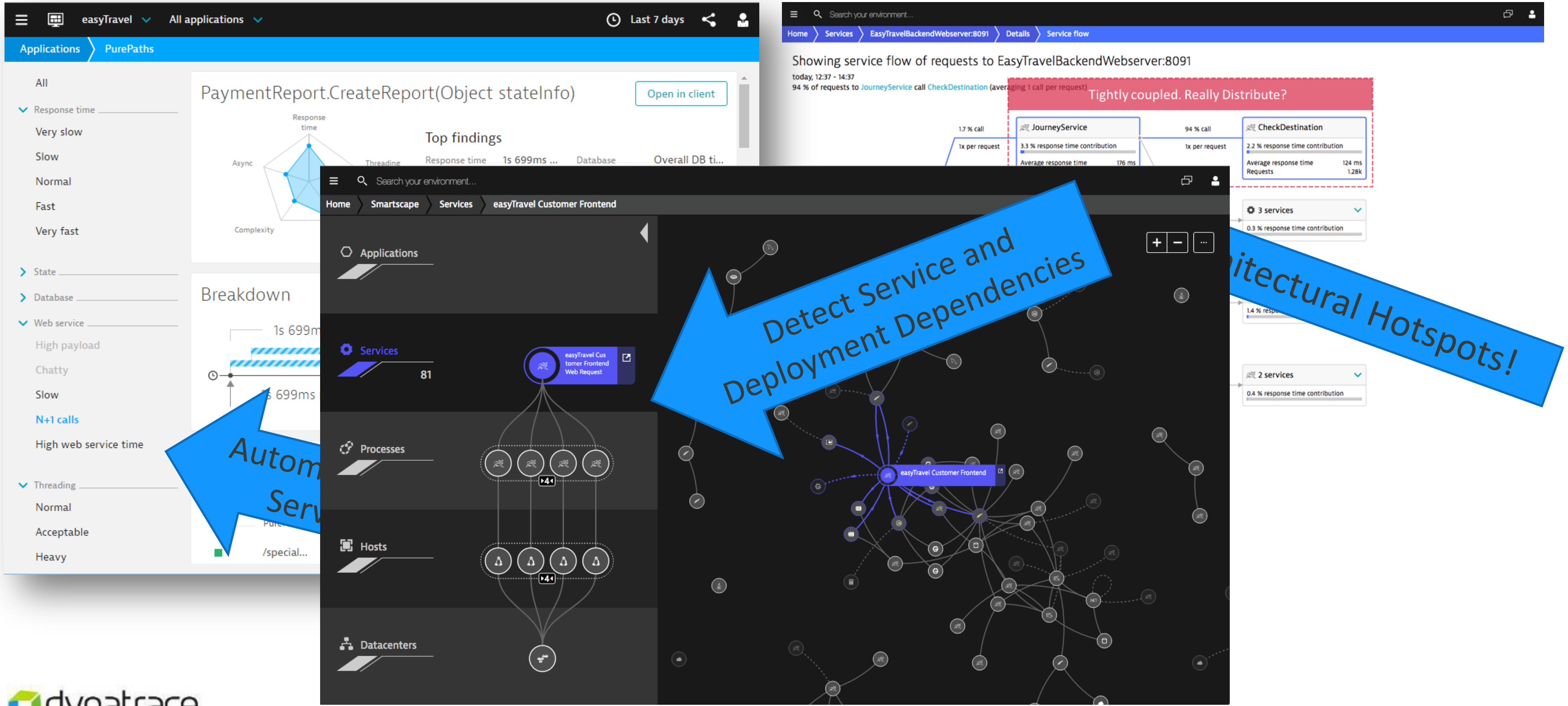
CinematchGetPredictions
80,105 | 0 | 5 0 0 | 0.0 %
Host: 13.8/s
Cluster: 8,011.0/s
Circuit Closed
Hosts 581 · Median 3ms · Mean 11ms · 90th 22ms · 99th 122ms · 99.5th 312ms

CryptexDecipher
74,718 | 0 | 92 0 58 | 0.0 %
Host: 12.9/s
Cluster: 7,486.8/s
Circuit Closed
Hosts 581 · Median 3ms · Mean 10ms · 90th 11ms · 99th 67ms · 99.5th 405ms

CinematchGetMovieRatings
60,672 | 0 | 2 0 0 | 0.0 %
Host: 10.4/s
Cluster: 6,067.4/s
Circuit Closed
Hosts 581 · Median 14ms · Mean 21ms · 90th 41ms · 99th 143ms · 99.5th 215ms

VideoHistoryGetBookmarks
48,281 | 0 | 60 0 0 | 0.1 %
Host: 8.3/s
Cluster: 4,834.1/s
Circuit Closed
Hosts 581 · Median 8ms · Mean 13ms · 90th 26ms · 99th 104ms · 99.5th 158ms

Automated Service Bottleneck Detection



PaymentReport.CreateReport(Object stateInfo)

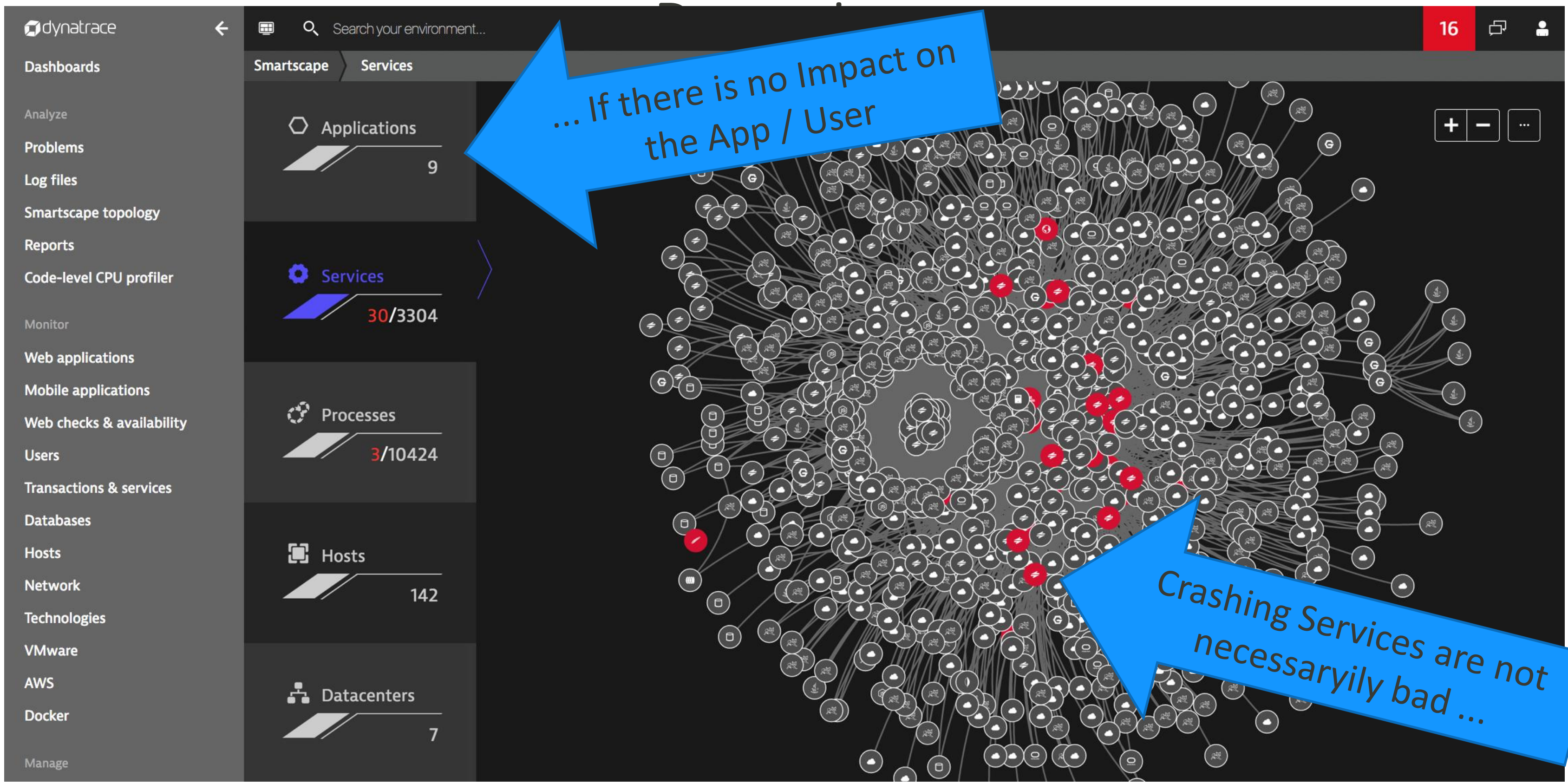
Showing service flow of requests to EasyTravelBackendWebserver:8091

today, 12:37 - 14:37

94 % of requests to JourneyService call CheckDestination (averaging 1 call per request)

dynatrace

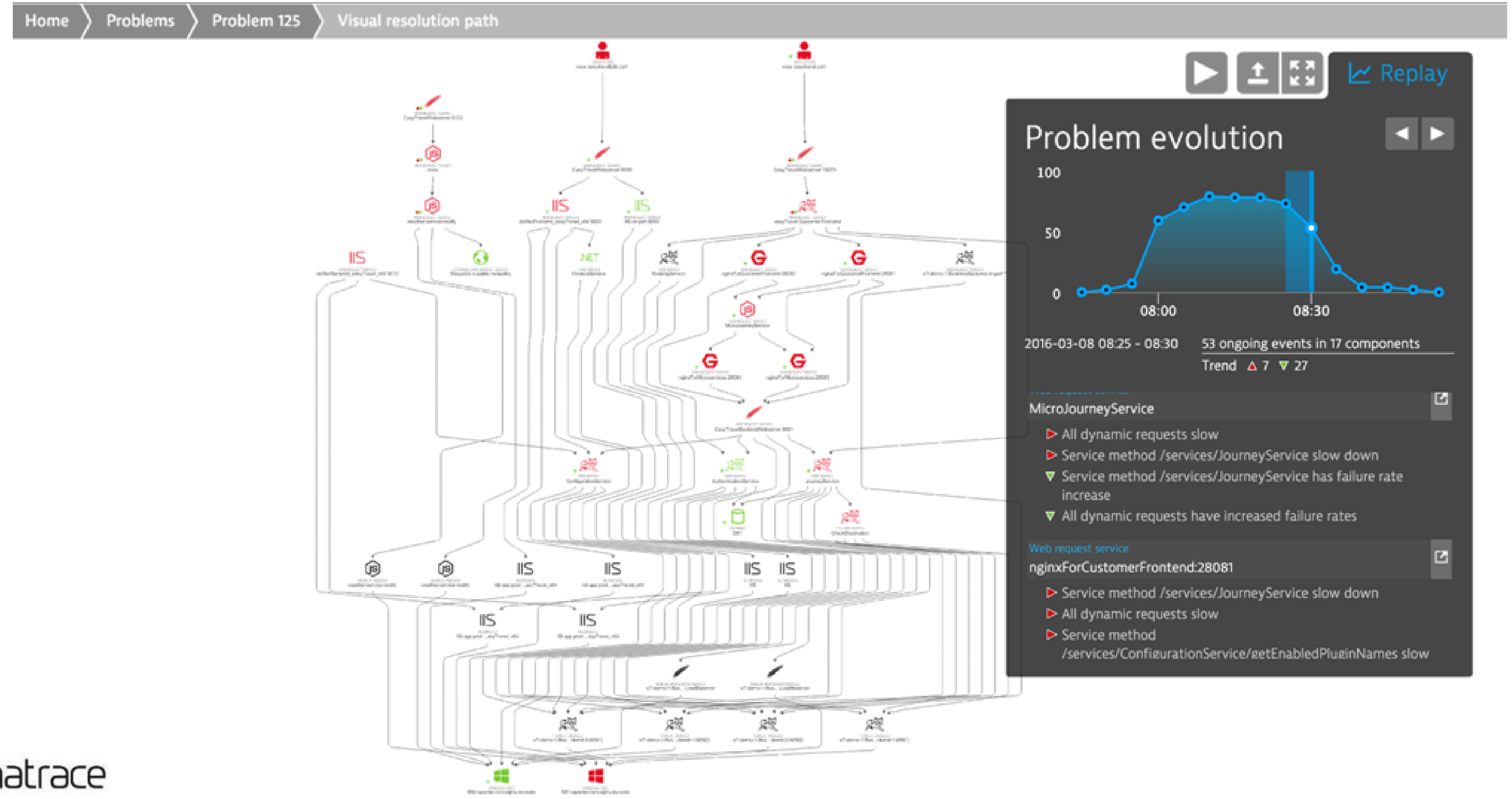
Automated Large Scale Service Monitoring and Bottleneck



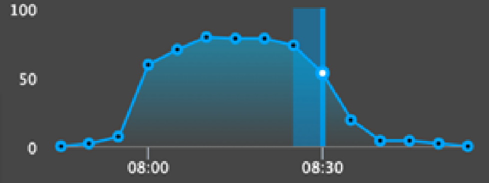
... If there is no Impact on the App / User

Crashing Services are not necessaryily bad ...

Automatic Bottleneck *Root Cause* Information



Problem evolution



2016-03-08 08:25 - 08:30

53 ongoing events in 17 components

Trend △ 7 ▽ 27

MicroJourneyService

- ▷ All dynamic requests slow
- ▷ Service method /services/JourneyService slow down
- ▽ Service method /services/JourneyService has failure rate increase
- ▽ All dynamic requests have increased failure rates

Web request service

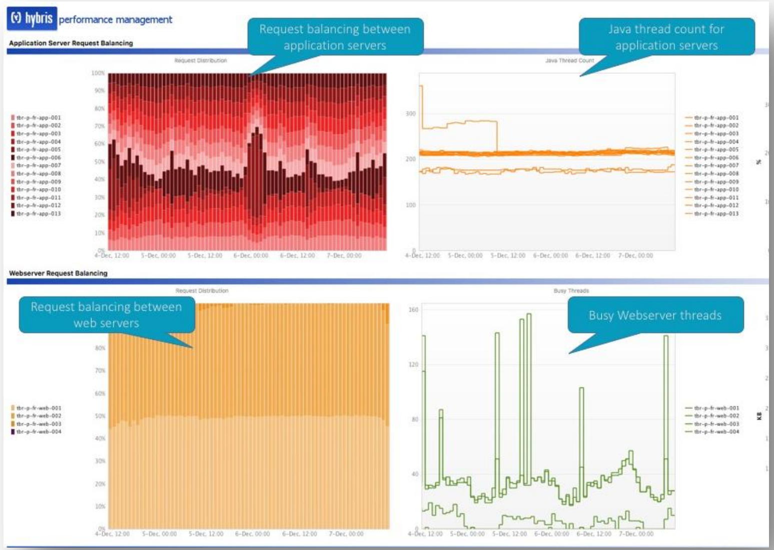
nginxForCustomerFrontend:28081

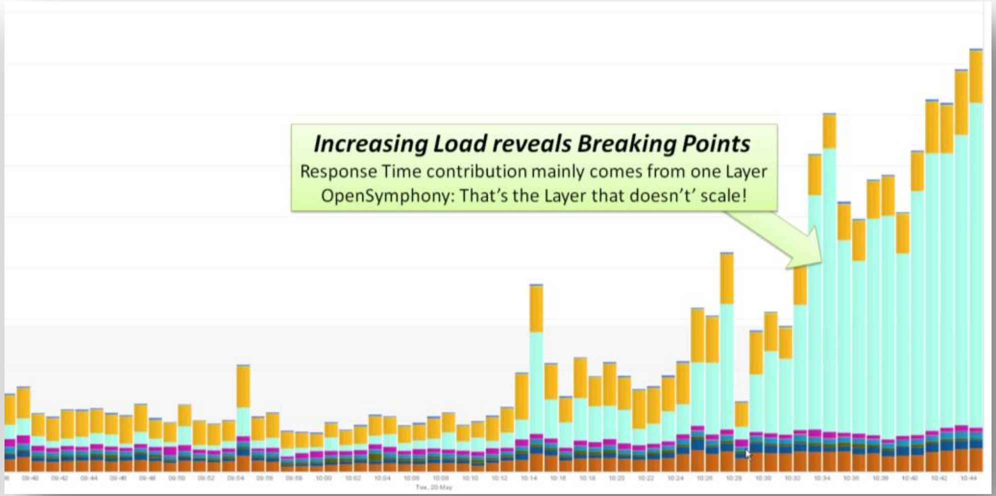
- ▷ Service method /services/JourneyService slow down
- ▷ All dynamic requests slow
- ▷ Service method /services/ConfigurationService/getEnabledPluginNames slow

dynatrace

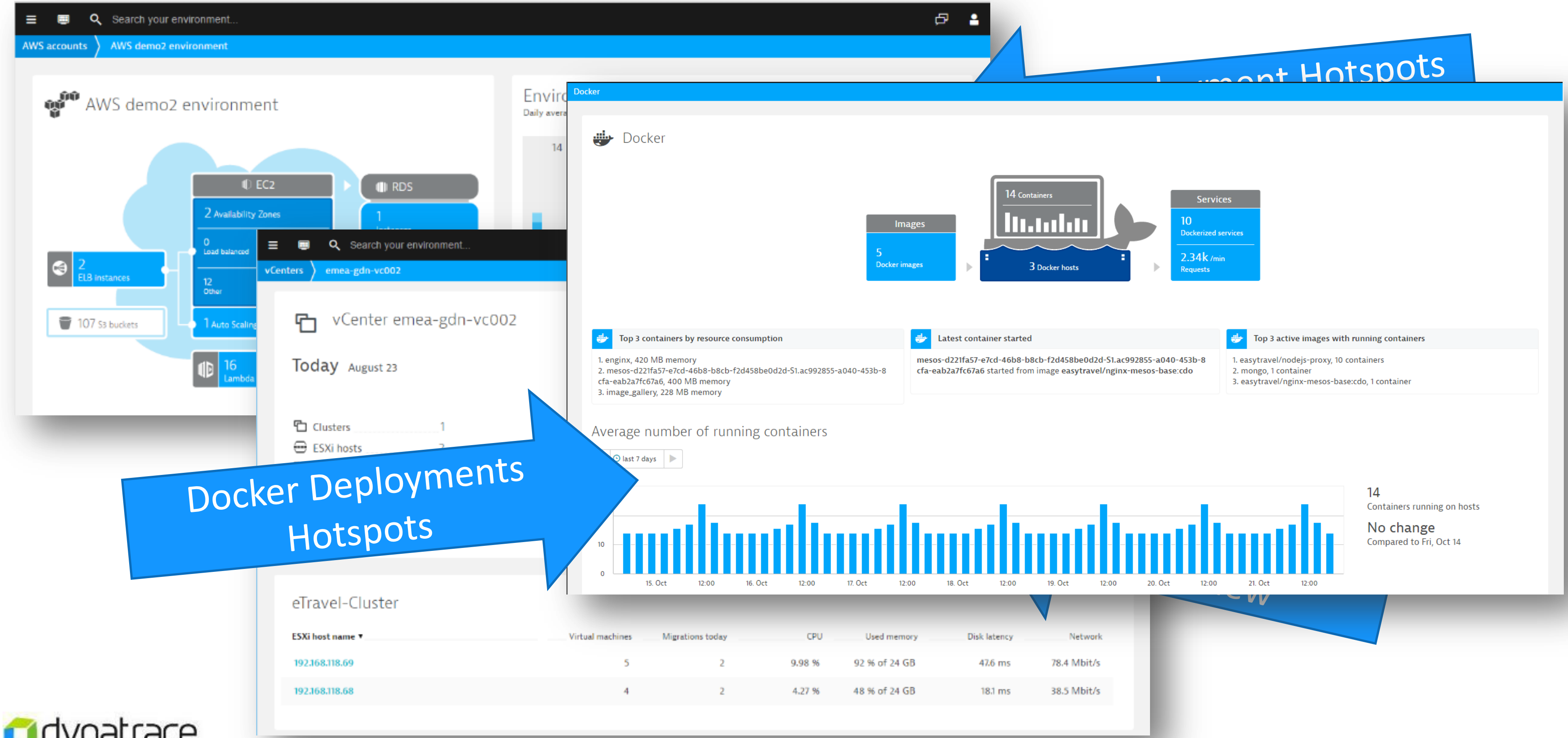
Manual Deployment Bottleneck Detection

- Blogs:
  - http://apmblog.dynatrace.com/2016/07/07/measure-frequent-successful-software-releases/
  - http://apmblog.dynatrace.com/2015/08/04/hybris-performance-review-10-system-health-checks/
- Patterns
  - Load Distribution, # HTTP 3xx/4xx/5xx, # of Exceptions, Stuck Threads, Timeouts, ...





Automated Deployment Bottleneck Detection



Top 3 containers by resource consumption

1. enginx, 420 MB memory
2. mesos-d221fa57-e7cd-46b8-b8cb-f2d458be0d2d-S1.ac992855-a040-453b-8cfa-eab2a7fc67a6, 400 MB memory
3. image_gallery, 228 MB memory

Latest container started

mesos-d221fa57-e7cd-46b8-b8cb-f2d458be0d2d-S1.ac992855-a040-453b-8cfa-eab2a7fc67a6 started from image **easytravel/nginx-mesos-base:cdo**

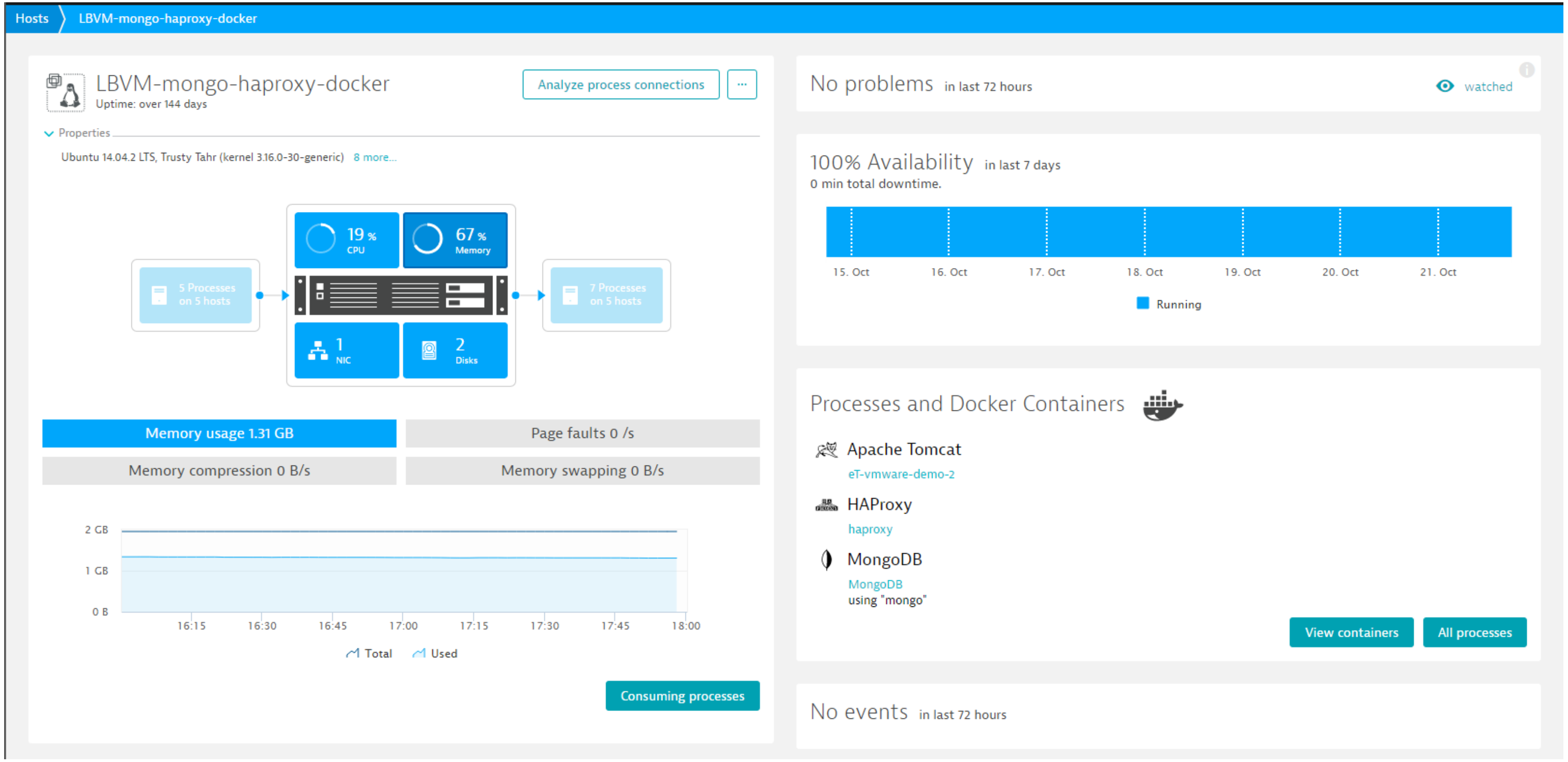
Top 3 active images with running containers

1. easytravel/nodejs-proxy, 10 containers
2. mongo, 1 container
3. easytravel/nginx-mesos-base:cdo, 1 container

Average number of running containers

ESXi host name	Virtual machines	Migrations today	CPU	Used memory	Disk latency	Network
192.168.118.69	5	2	9.98 %	92 % of 24 GB	47.6 ms	78.4 Mbit/s
192.168.118.68	4	2	4.27 %	48 % of 24 GB	18.1 ms	38.5 Mbit/s

Automatic Bottleneck *Root Cause* Information



Hosts 〉 LBVM-mongo-haproxy-docker

LBVM-mongo-haproxy-docker
Uptime: over 144 days

Analyze process connections

Properties
Ubuntu 14.04.2 LTS, Trusty Tahr (kernel 3.16.0-30-generic) 8 more...

Memory usage 1.31 GB
Page faults 0 /s
Memory compression 0 B/s
Memory swapping 0 B/s

Consuming processes

No problems in last 72 hours
watched

100% Availability in last 7 days
0 min total downtime.

Processes and Docker Containers

Apache Tomcat
eT-vmware-demo-2

HAProxy
haproxy

MongoDB
MongoDB
using "mongo"

View containers
All processes

No events in last 72 hours

dynatrace