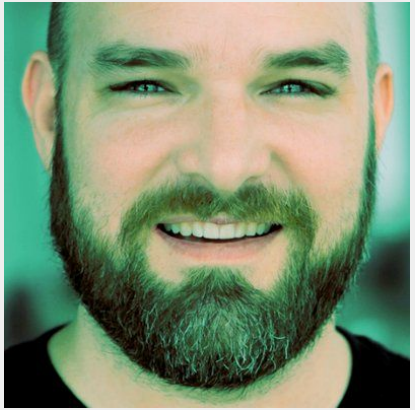
Holistic Monitoring

DevConf 2019 - Brno, Czechia

Sunday, January 27th

Who are we?



Jared Sprague
Principal Software Engineer

jsprague@redhat.com
@caramelcode

Adam Minter
Associate Project Manager

aminter@redhat.com

What we'll cover today

Problem Statement

Our Solution

Monitoring Strategy Walkthrough

How is this working for us today?

Questions!

Problem Statement

Perception & Missed Signals

Perception

Perception

The term, “Monitoring” too often coincides with a tool

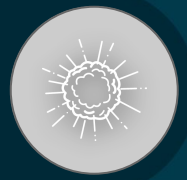
- Zabbix = “Our monitoring”
  - Nagios, Solarwinds, Prometheus, etc etc
- Mis-align that tool = monitoring

Monitoring = a wildly complex and persistent art of principals, tooling, technology, and innovation

Perception

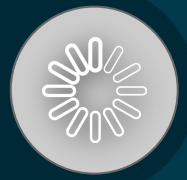
The term, “Monitoring” too often coincides with a tool

Tool Change = Really Hard



Redundant Tooling Inevitable

Technical Debt



Adoption & Usability

Missed Signals

“We didn’t catch that in our monitoring”

Noise

Aka alerts that don't matter

1 of 2
FRM:nagios.eng.rdu2@redhat.com
MSG:N: PROBLEM

Svc: Mysql Server

Hst: db02.db.eng.rdu2

State: CRITICAL

D/T: 01-25-2019
(Con't) 2 of 2
20:17:15

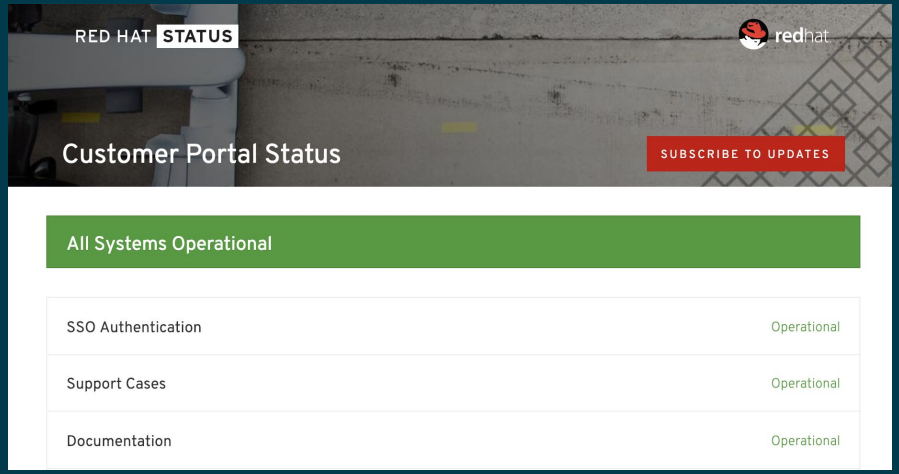
More:

SLOW_SLAVE CRITICAL: Slave IO: Yes Slave SQL: Yes Seconds Behind Master: 1000

Author:

Comment: (End)

Alerts

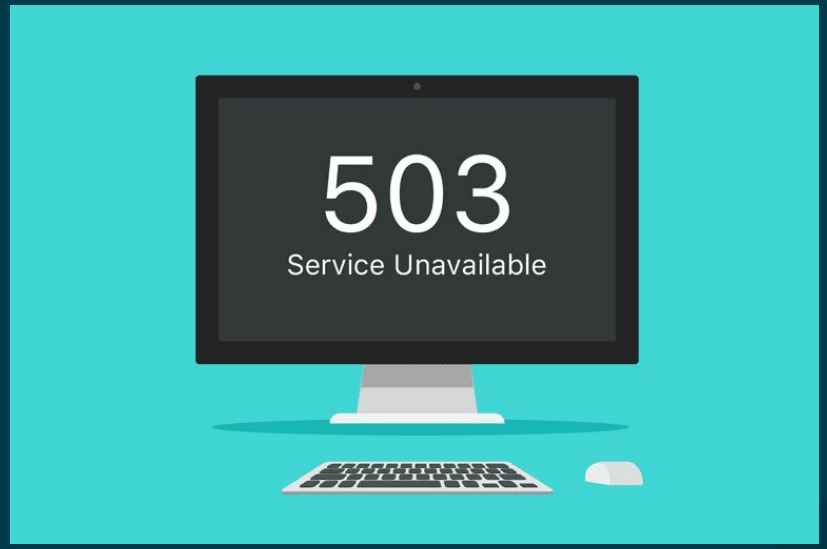


Operational State

No Signals

No alerts when it matters

Alerts



Operational State

Signal to Noise Ratio

Our Solution

Holistic Monitoring Strategy

Dashboards

- Centralized place for visual analysis of data
- Assurance that all functions can be exported and consumed here
- Provides simple digestion of the hard work done in the functions below

Logging

- A bucket for all transaction data from each function in our strategy
- Ensures we comply with data retention policies in a consistent manner
- Facilitates transparency with the data we collect from each source

Alert Orchestration

- Federation of all data sources to determine if/when someone needs to act on a problem or issue

Metrics

- Empowers engineers to trend the data collected from our sources and make proactive improvements

Application Performance

- Transaction traces and snapshots
- Stack specific performance metrics
- Inspection at the code level to determine performance

Availability

- Baseline determination if application is working
- Ping, API calls, or custom scripting
- Often reported through application performance indicators (KPIs)
- Examination from different geographical locations

Real User Monitoring

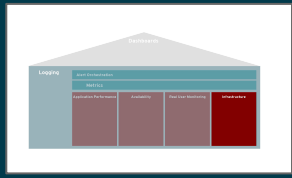
- Inspection of users interactions with a WebUI
- Sometimes Javascript that collects metrics from user's browsers
- Helps to add context to issues and scenarios afflicting a system or application

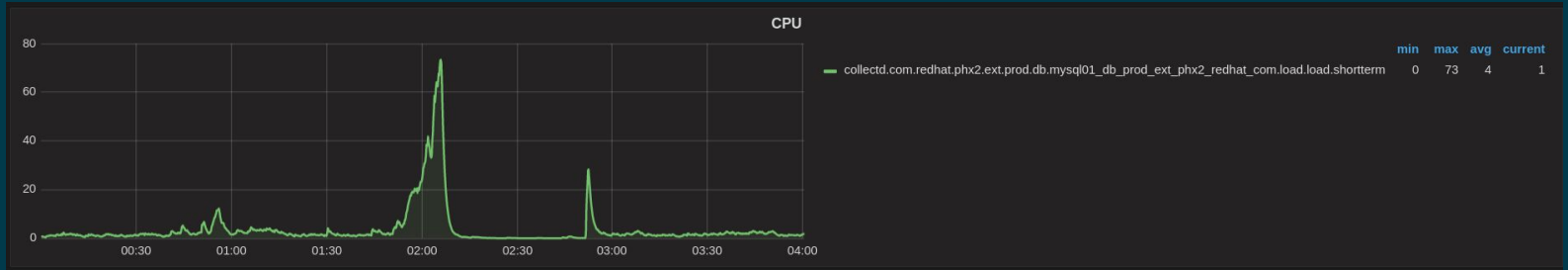
Infrastructure

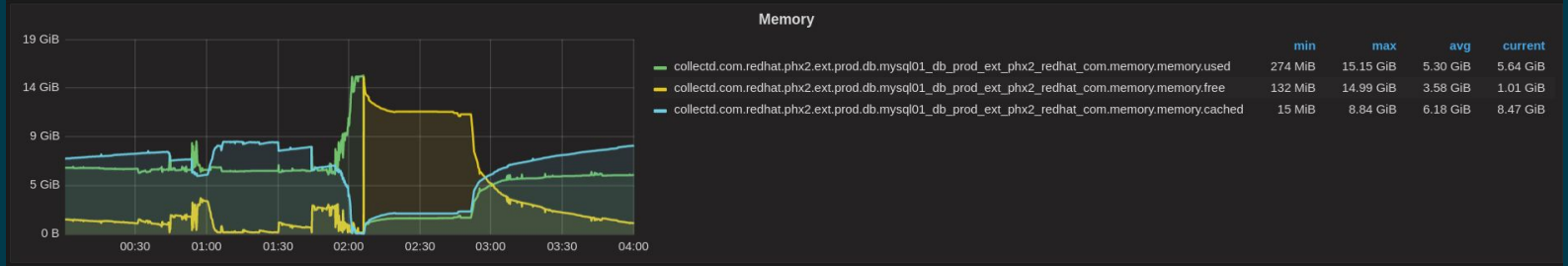
- System Resources like CPU, Memory, Disk, etc
- Network monitoring also falls into this category
- Inventory and Resource management
- The throughput of a build pipeline is one complex example

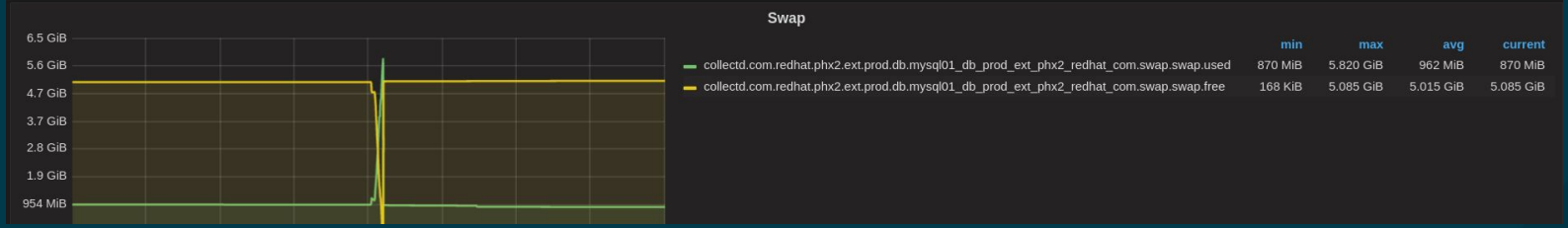
Monitoring Strategy Walkthrough

Infrastructure

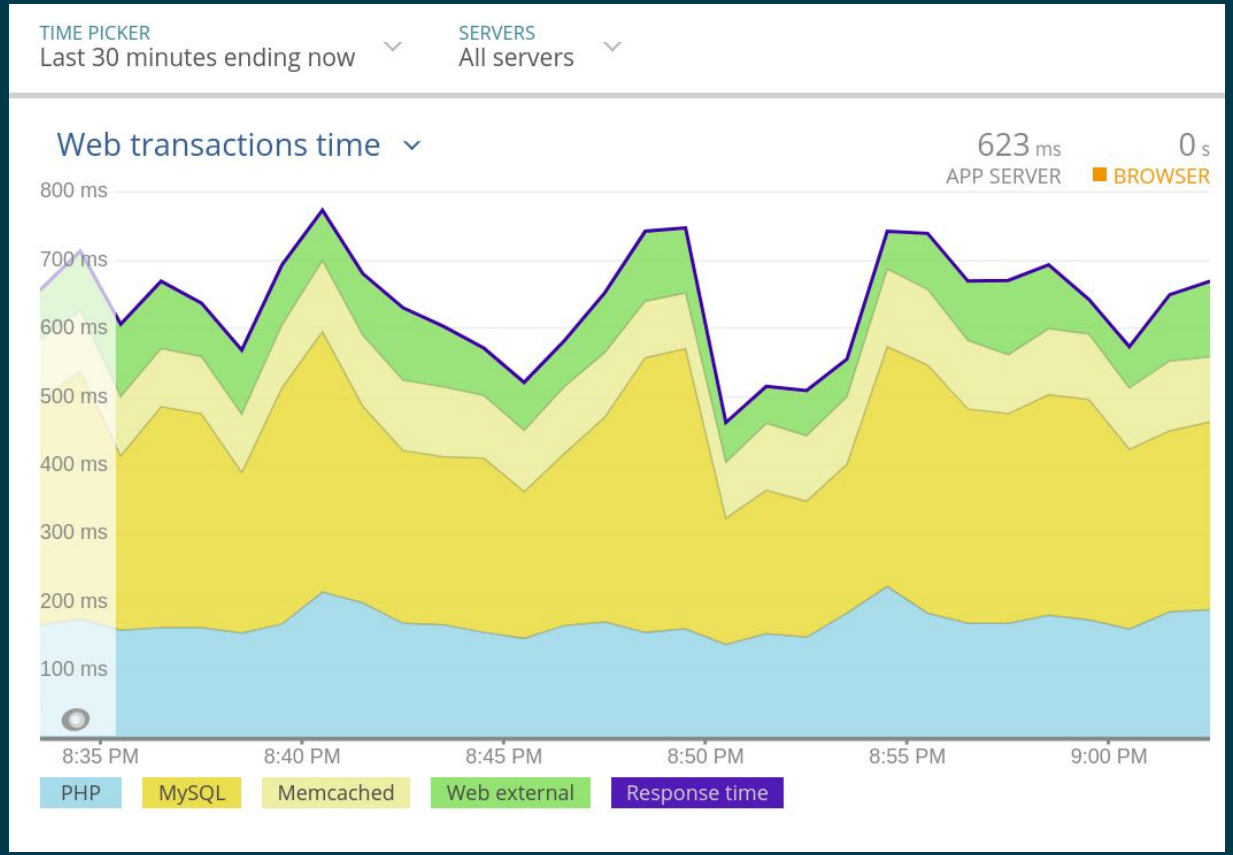




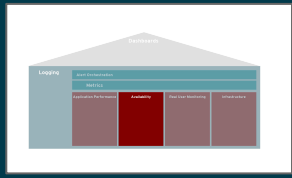




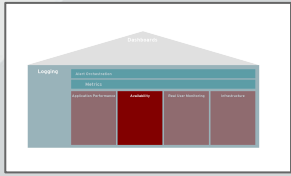
APM

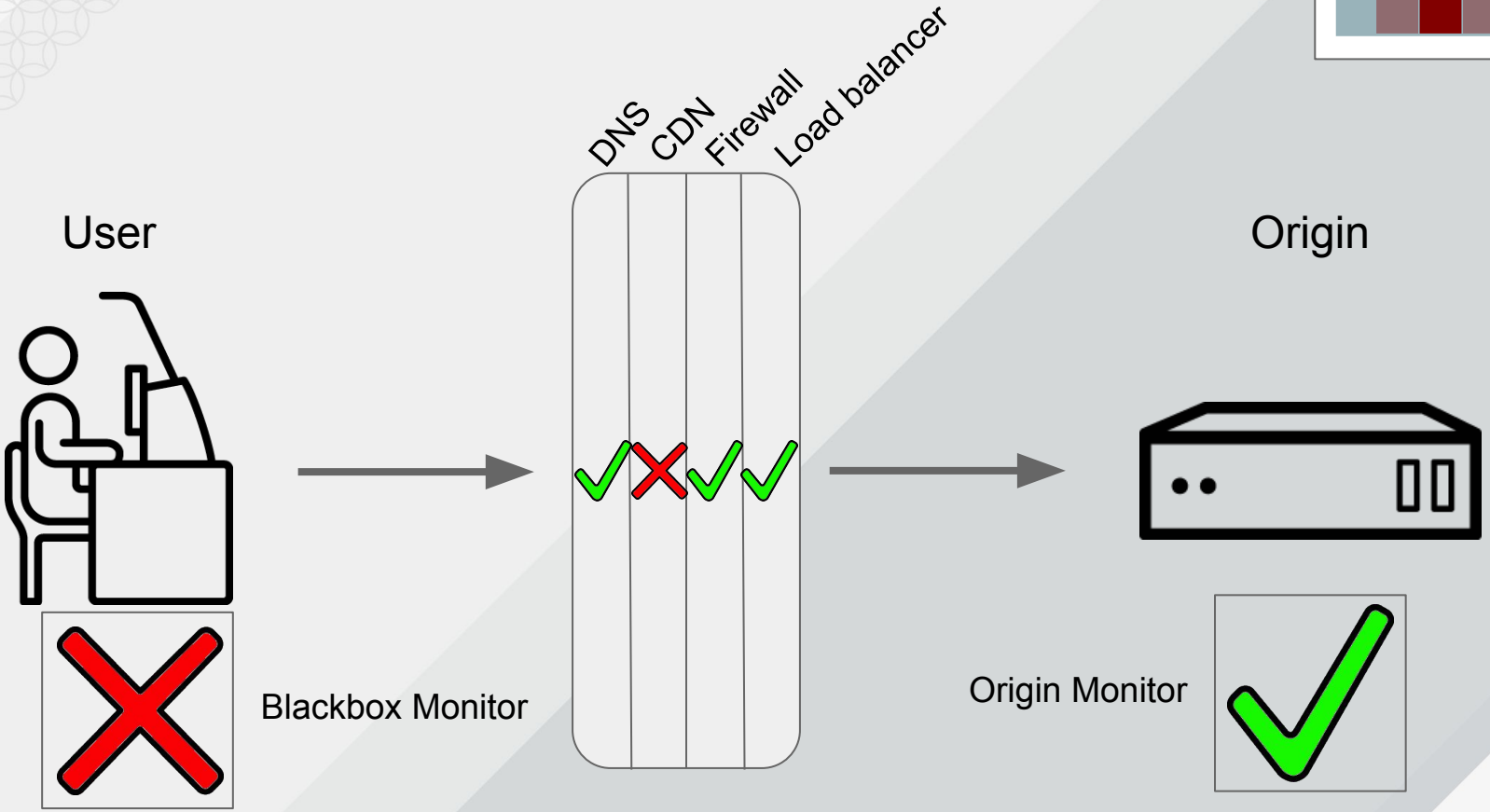

TIME PICKER
Last 30 minutes ending now
SERVERS
All servers
Web transactions time
623 ms
APP SERVER
0 s
BROWSER
800 ms
700 ms
600 ms
500 ms
400 ms
300 ms
200 ms
100 ms
8:35 PM
8:40 PM
8:45 PM
8:50 PM
8:55 PM
9:00 PM
PHP
MySQL
Memcached
Web external
Response time

Availability

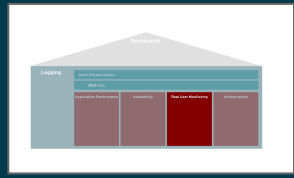


User Centric Blackbox Availability





Real User Monitoring



TIME PICKER
3 hours ending today, 2:00

Browser page load time — 4.98 s BROWSER

Release

Resolved

Alert triggered

15 sec
12.5 sec
10 sec
7.5 sec
5 sec
2.5 sec

15, PM | Sep 15, 11:30 PM | Sep 16, 12:00 AM | Sep 16, 12:30 AM | Sep 16, 1:00 AM | Sep 16, 1:30 AM

Web application | Network | DOM processing | Page rendering

Apdex — 0.91 [7.0] BROWSER

1
0.8
0.6

End User

Throughput by browser

200 ppm
100 ppm

15, PM | Sep 15, 11:30 PM | Sep 16, 12:00 AM | 16, AM | Sep 16, 1:00 AM

Chrome | Firefox | IE | Other

Recent session traces

Session traces	Started at	Page load
access.redhat.com /solutions/1355683	6:00 am	0.569s
access.redhat.com /articles/1461	5:56 am	7.814s

Page views with JS errors

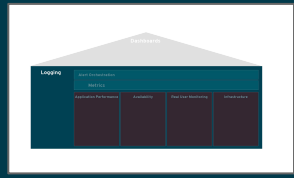
2 %
1.5 %
1 %
0.5 %

AJAX response time

7.5 sec
5 sec
2.5 sec

redhat.

Logging



Holistic Monitoring Strategy

Dashboards

- Centralized place for visual analysis of data
- Assurance that all functions can be exported and consumed here
- Provides simple digestion of the hard work done in the functions below

Logging

- A bucket for all transaction data from each function in our strategy
- Ensures we comply with data retention policies in a consistent manner
- Facilitates transparency with the data we collect from each source

Alert Orchestration

- Federation of all data sources to determine if/when someone needs to act on a problem or issue

Metrics

- Empowers engineers to trend the data collected from our sources and make proactive improvements

Application Performance

- Transaction traces and snapshots
- Stack specific performance metrics
- Inspection at the code level to determine performance

Availability

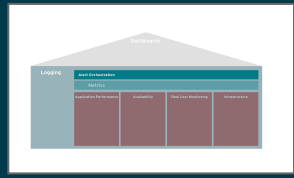
- Baseline determination if application is working
- Ping, API calls, or custom scripting
- Often reported through application performance indicators (KPIs)
- Examination from different geographical locations

Real User Monitoring

- Inspection of users interactions with a WebUI
- Sometimes Javascript that collects metrics from user's browsers
- Helps to add context to issues and scenarios afflicting a system or application

Infrastructure

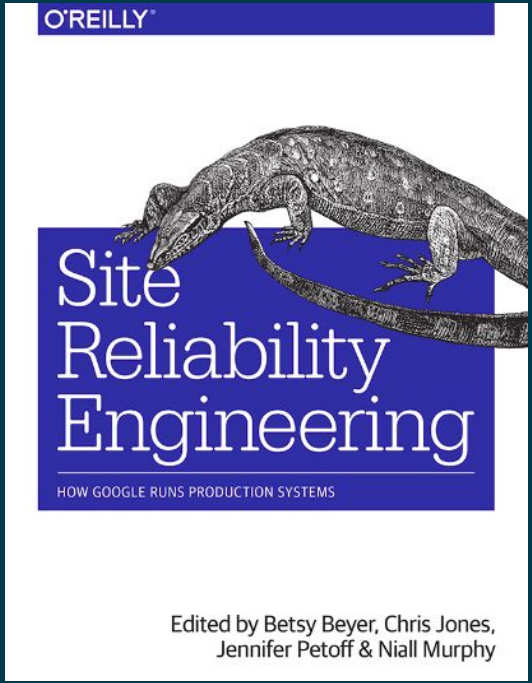
- System Resources like CPU, Memory, Disk, etc
- Network monitoring also falls into this category
- Inventory and Resource management
- The throughput of a build pipeline is one complex example



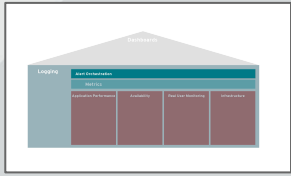
SLIs & SLOs

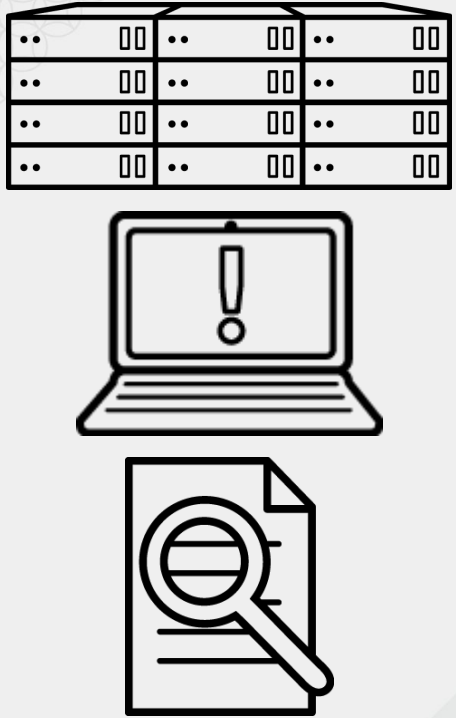
What are they? Why use them?

- Service-Level Indicator (SLI)
  - Ex: Error Rate.
- Service-Level Objective (SLO)
  - Ex: Error rate < 5% of 99% of requests over a 30min period
- Why use them?
  - Focuses alerts on business outcomes, and not arbitrary metrics
  - Improves signal to noise ratio
- Learn more in Google SRE book

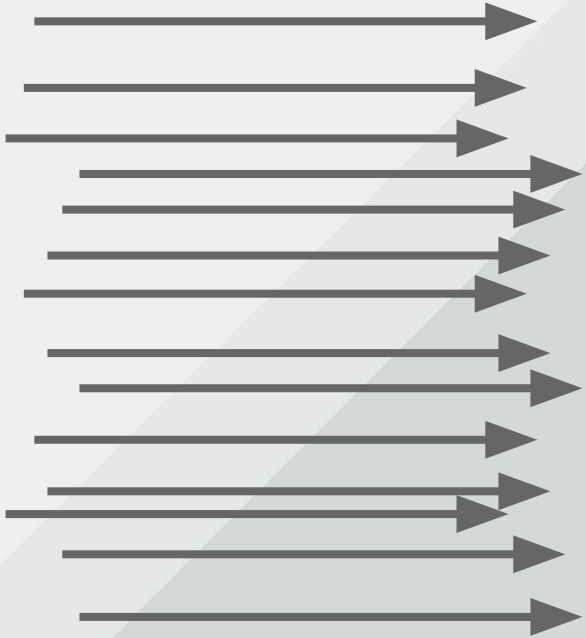


Without Alert Orchestration

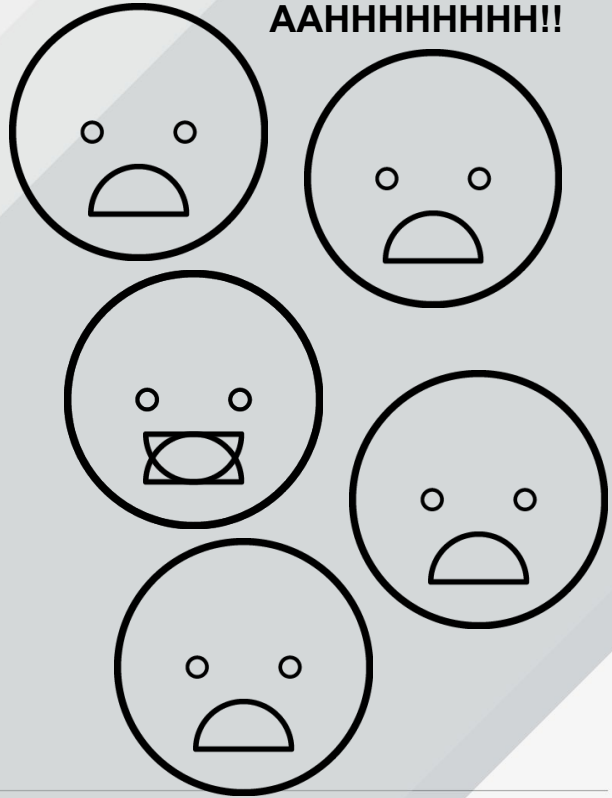




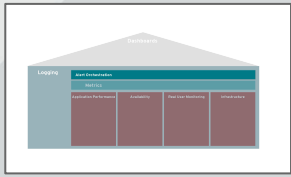
Notifications

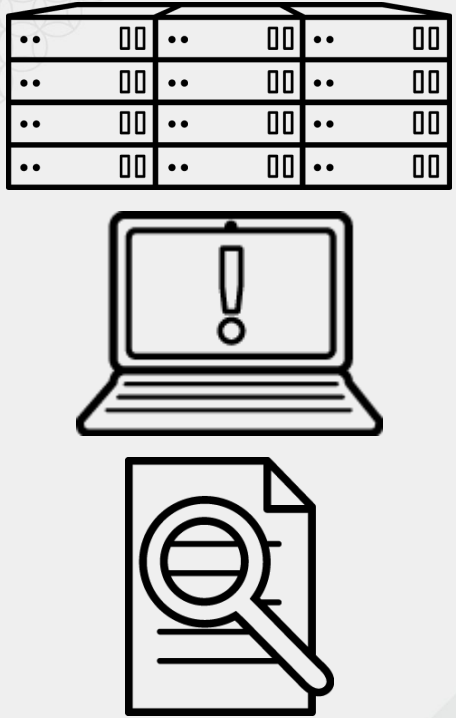


AAHHHHHHHH!!

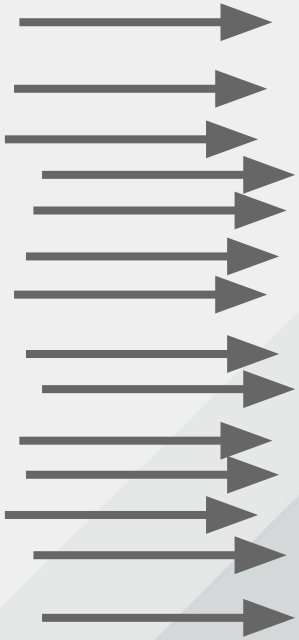


With Alert Orchestration



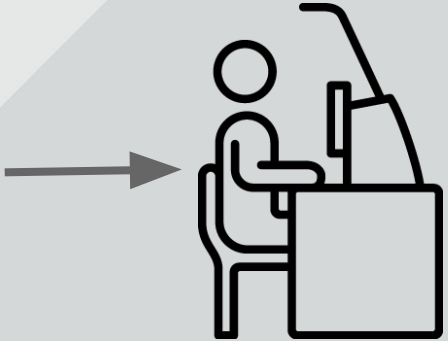


Alerts



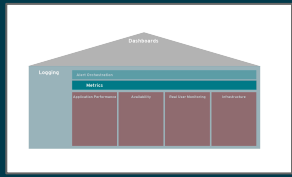
Alert Orchestrator

Notifications



On-call

Metrics and Dashboards



Monitor Specific Uptime Past 7 Days

Since 7 days ago

Availability

-99.5	MBS Builds in INIT State
-100	Estuary API Availability
-100	F2.0 Reviewrot Availability
-100	Greenwave Decision API Availability
-100	Retrieve waivers
-100	MBS unknown error in the init state
-100	MBS Builds API Availability
-100	Datagrepper API Availability
-100	WaiverDB availability healthcheck

7 Day Overall Uptime

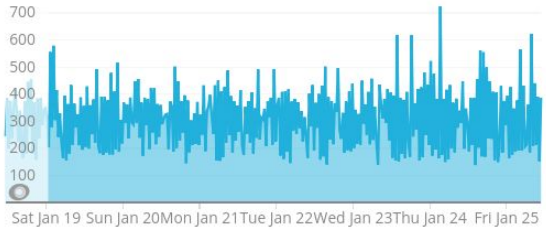
Since 7 days ago

99.94

Percentage

Greenwave Decision Latency

Since 7 days ago



30 days uptime by monitor

Since 30 days ago

Availability

-93.07	MBS Builds in INIT State
-94.23	MBS unknown error in the init state
-94.28	MBS Builds API Availability
-95.0	Greenwave Decision API Availability
-96.46	Freshmaker Events API Availability
-99.86	F2.0 Reviewrot Availability
-100	Estuary API Availability
-100	Retrieve waivers
-100	Datagrepper API Availability

30 Day Overall Uptime

Since 30 days ago

97.29

Percentage

Overall Uptime Percentage by Week

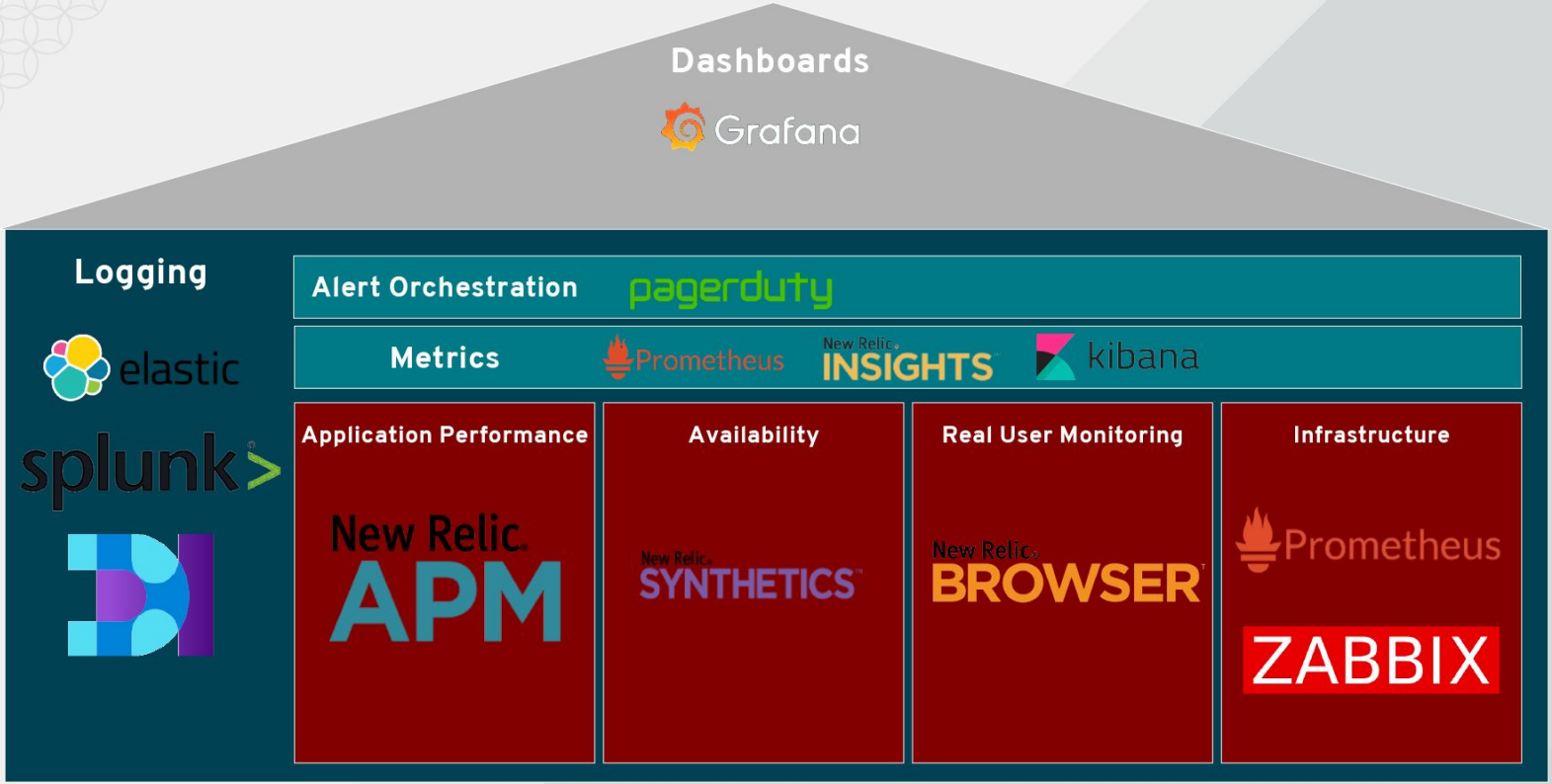
Since 1 month ago

Percentage

99.91	Week of January 21, 2019
100	Week of January 14, 2019
88.98	Week of January 7, 2019
99.45	Week of December 31, 2018
100	Week of December 24, 2018

How is this helping us today?

Holistic Monitoring Strategy (Tooling)



Open Source Alternative (Tooling)

Dashboards

Grafana

Logging

elastic

SENTRY

Alert Orchestration IRIS onCALL

Metrics Prometheus kibana

Application Performance

Prometheus

PERFORMANCE CO-PILOT

elastic

Availability

Prometheus

Se SELENIUM

Real User Monitoring

Bucky

Infrastructure

Prometheus

ZABBIX

collectd

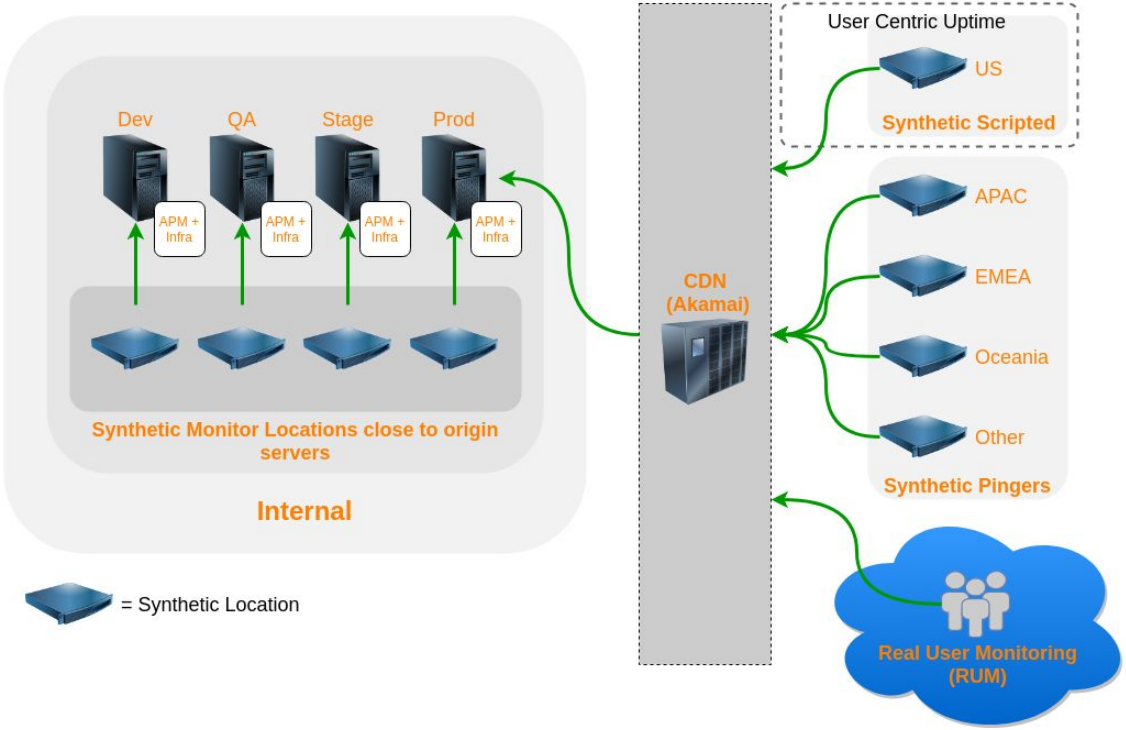
Nagios

Customer Portal Holistic Deployment

ENTERPRISE APPLICATION MONITORING
APM + Synthetics + RUM + Infrastructure

Key Features

- Exceptionally accurate Uptime reports.
- Three different monitor types both internal and external provide complete picture and assist with troubleshooting problems.
- Awareness of global performance trends and isolated outages.
- Wide variety of performance metrics stored overtime from synthetic tests + Real User experience.
- Pre-prod monitoring detects problems or improvements before release.



Holistic Postmortem

Availability

At 2:19 AM EST CP On-call members were notified about failures in many New Relic synthetic availability monitors, for Search front-end, kbase, documentation, labs, container catalog and PCM.

New Relic. APM BROWSER SYNTHETICS MOBILE PLUGINS INSIGHTS INFRASTRUCTURE Maps Alerts Tools Help CEE Prod Synthetics Pro Annu...

Monitors SLA report Private locations Permissions Monitor downtime Secure credentials Location status

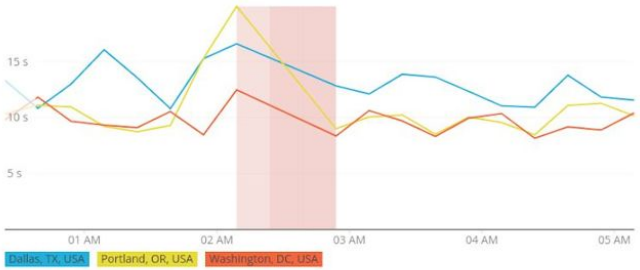
SCRIPTED BROWSER Drupal - KBase - Articl... TIME PICKER from Dec 5, 12:12 am to Dec 5, 05:23 am

MONITOR
- Overview
- Results
- Resources
- Violations

REPORTS
- SLA

SETTINGS
- General
- Script
- Alert Conditions

Load time



Availability

99.9%	99.7%	100%
LAST 30 DAYS	LAST 7 DAYS	LAST 30 MINUTES

Average load size

Slowest results

	Load time
Portland, OR, USA - Dec 5th 2018, 7:16:20	28.1 s
Dallas, TX, USA - Dec 5th 2018, 6:20:06	23.2 s
Portland, OR, USA - Dec 5th 2018, 7:07:28	21.4 s
Dallas, TX, USA - Dec 5th 2018, 9:45:04	21 s
Dallas, TX, USA - Dec 5th 2018, 7:15:27	20.7 s

Failures

Time	Location	Message
Dec 5th 2018, 7:51:08	Portland, OR, USA	TaskTimedOut: Task timed-out waiting for element to be located using: By.id("username")
Dec 5th 2018, 7:50:25	Washington, DC, USA	TaskTimedOut: Task timed-out waiting for element to be located using: By.id("username")
Dec 5th 2018, 7:50:04	Dallas, TX, USA	Error: no such element: Unable to locate element: {"method":"id","selector":"accountUserName"}

redhat.

Availability

RED HAT **STATUS**

redhat.

Customer Portal Status

SUBSCRIBE TO UPDATES

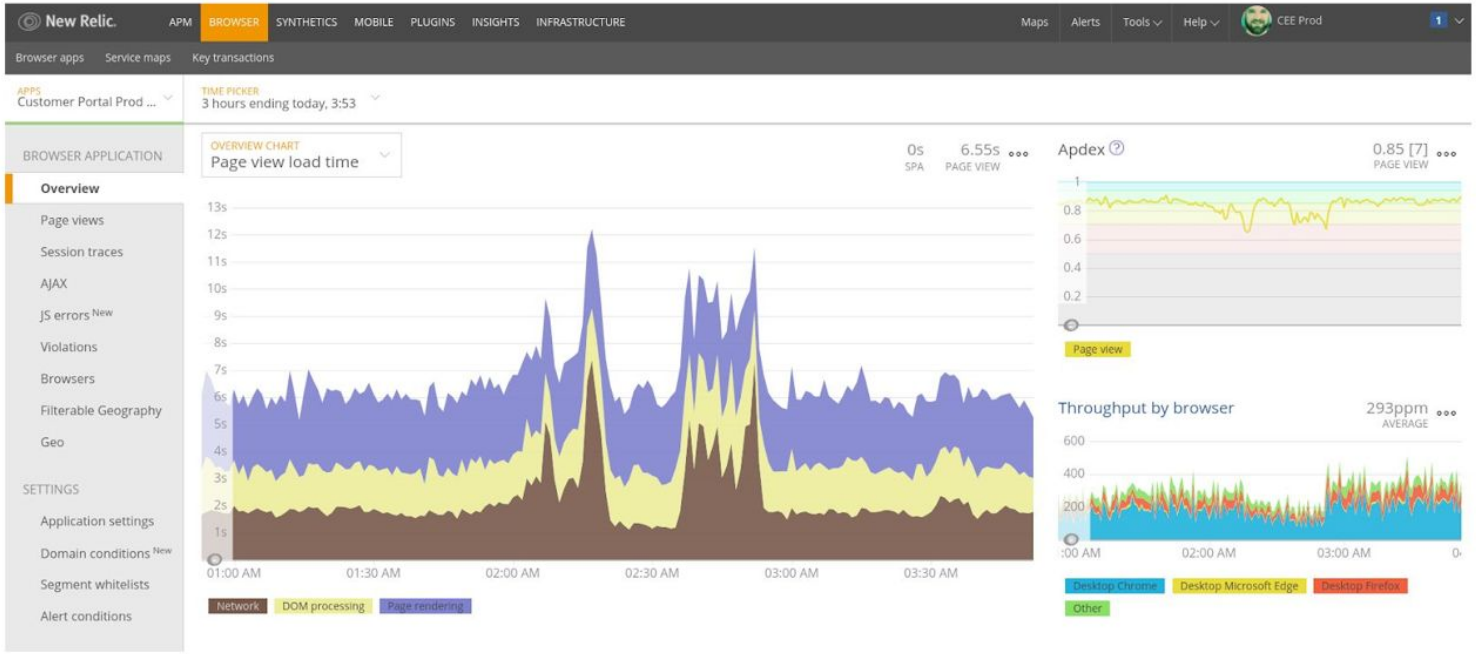
Partial System Outage

SSO Authentication	Operational
Support Cases	Operational
Documentation	Major Outage
Search	Major Outage
Downloads	Major Outage
⊞ Subscription Management	Major Outage
Knowledgebase	Major Outage

redhat.

Real User Monitoring

From RUM we can see that this also affected user experience by increasing the overall page load time for end users:



Availability

Search requests

802 ms DOM Content Loaded | 802 ms Page Loaded

Type	Request	Status	Size	Time	Timeline / Duration
HTML	access.redhat.com 10076	OK 500	-	787 ms	

https://access.redhat.com
10076 View resource

Overview | Response | Request | View Insights query | View APM transaction trace

text/html	500 OK	GET	23.46.233.122	42	07:23:15 GMT
TYPE	STATUS	METHOD	SERVER IP	CONNECTION ID	REQUEST TIME

Load time

780 ms

197 ms | 393 ms | 590 ms

1.85 ms	3.68 ms	780 ms	0.991 ms	786 ms
BLOCKED	SEND	WAIT	RECEIVE	TIME TO FIRST BYTE

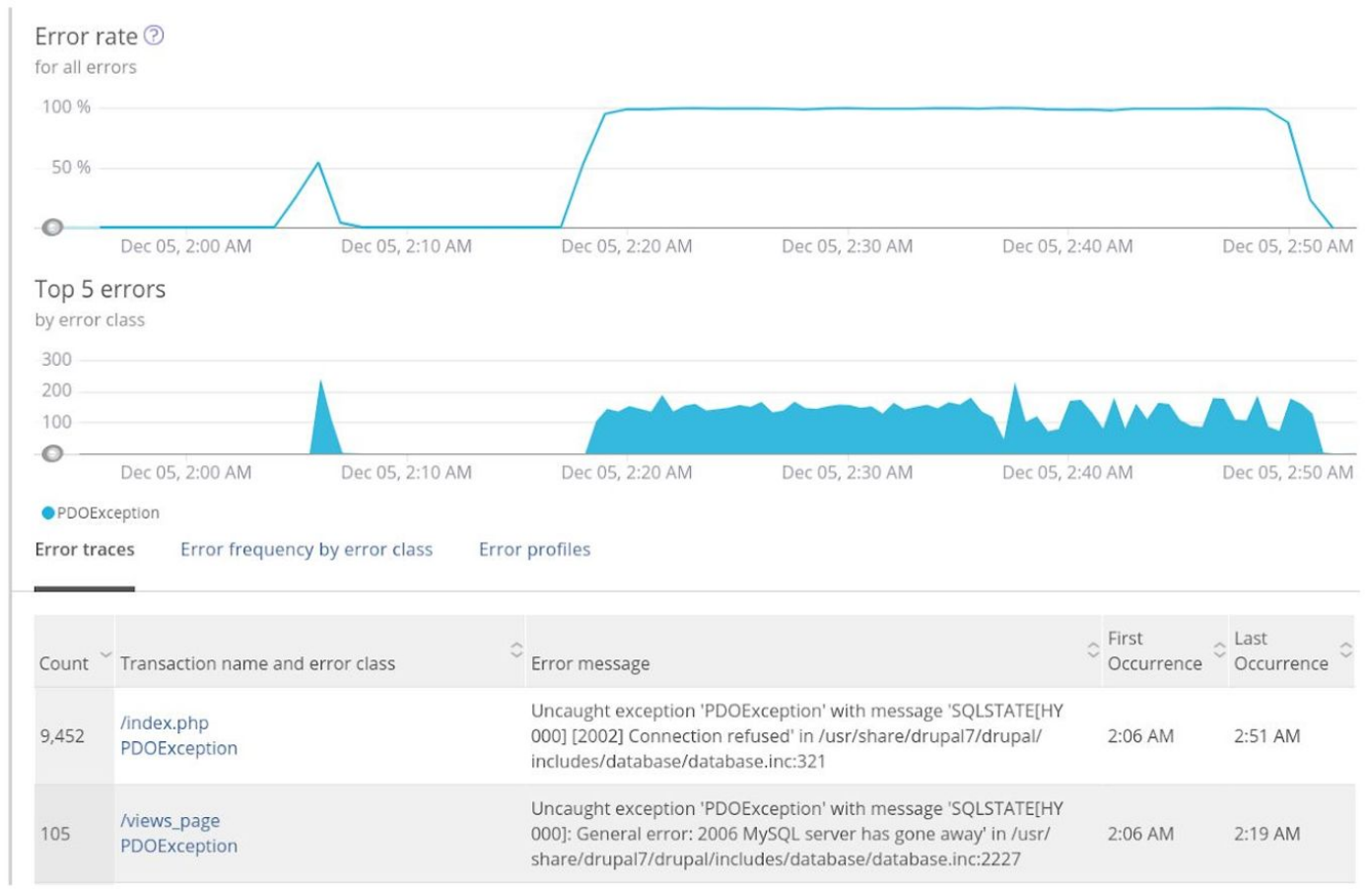
redhat.

APM

Tracing from the Synthetic check to the APM transaction trace, we see that the last call in the stack before it threw an exception was: “DatabaseConnection::__construct”

2.0	0.31%	file_scan_directory	0.584 s
31.0	4.76%	6 fast method calls	0.587 s
13.0	2.00%	file_scan_directory	0.626 s
3.0	0.46%	variable_set	0.646 s
3.0	0.46%	db_merge	0.646 s
3.0	0.46%	Database::getConnection	0.646 s
3.0	0.46%	Database::openConnection	0.646 s
3.0	0.46%	DatabaseConnection_mysql::__construct	0.646 s
3.0	0.46%	DatabaseConnection::__construct	0.646 s

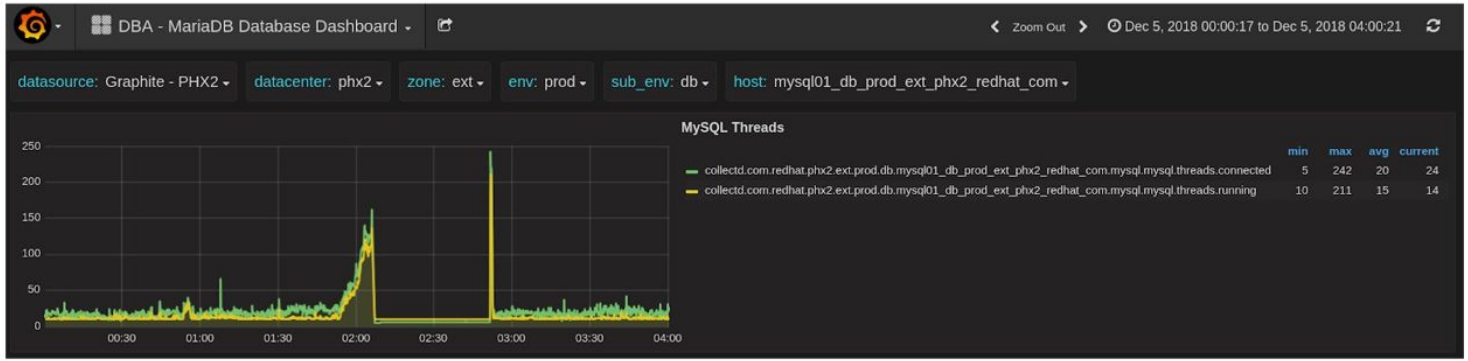
APM + Logging

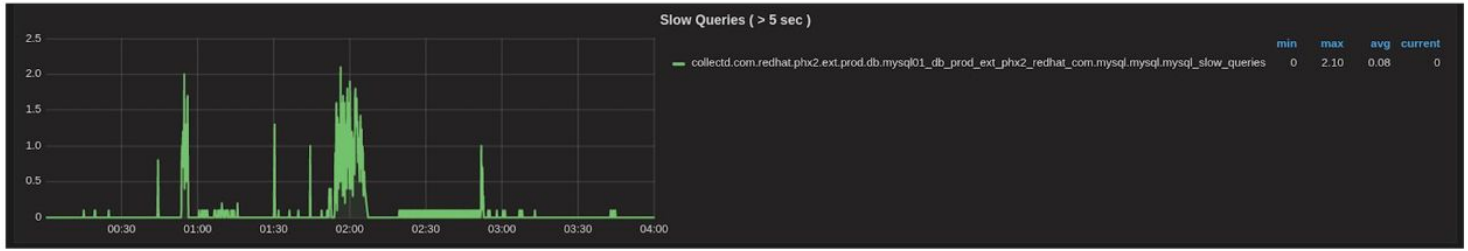


Count	Transaction name and error class	Error message	First Occurrence	Last Occurrence
9,452	/index.php PDOException	Uncaught exception 'PDOException' with message 'SQLSTATE[HY 000] [2002] Connection refused' in /usr/share/drupal7/drupal/includes/database/database.inc:321	2:06 AM	2:51 AM
105	/views_page PDOException	Uncaught exception 'PDOException' with message 'SQLSTATE[HY 000]: General error: 2006 MySQL server has gone away' in /usr/share/drupal7/drupal/includes/database/database.inc:2227	2:06 AM	2:19 AM

Infrastructure

So now we know that drupal can't reach MySQL so we check the MySQL Infrastructure monitoring which is done by collectd and grafana, and we see the spike in thread count, and the host drained of memory, swap usage spike, and finally the process is killed.





Infrastructure

Corrective Actions

Immediate Corrective Actions

Describe the immediate actions taken to stop the incident.

Restarting dead MySQL service on the mysql01.db.prod.ext.phx2.redhat.com host

Secondary Corrective Actions

- *Describe technical or process changes that are needed to prevent this issue from happening again in the future.*

Optimize slow query in drupal.

Priority	Issue	Corrective Action	Owner	Result	Completion Date	Notes
High	CPDRUPAL-4059	Optimize slow query in Drupal. Jason reports he's optimized it from 4 seconds to 70ms	Jason Smith	In progress		

Summary

- Think of monitoring by function not by tool
- Catch signals by implementing each function
- User centric blackbox availability
- Improve signal to noise ratio with SLIs SLOs
- Make Ops happy with Alert Orchestration

Happy Customers

Questions?

THANK YOU

Jared Sprague
Principal Software Engineer

jsprague@redhat.com
@caramelcode

Adam Minter
Associate Project Manager

aminter@redhat.com

redhat