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

---

Adam Minter
Missed Signals
“We didn’t catch that in our monitoring”
Noise
Aka alerts that don’t matter

Alerts

Operational State

Customer Portal Status

All Systems Operational

SSO Authentication: Operational
Support Cases: Operational
Documentation: Operational
No Signals
No alerts when it matters
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 affecting 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

CPU

Memory

Swap
APM

Web transactions time

623 ms

APP SERVER  Browser

8:35 PM  8:40 PM  8:45 PM  8:50 PM  8:55 PM  9:00 PM

100 ms  200 ms  300 ms  400 ms  500 ms  600 ms  700 ms  800 ms

PHP  MySQL  Memcached  Web external  Response time
The fallacy of origin based availability monitoring: Customers don’t use applications in the Datacenter
Real User Monitoring

Browser page load time:
- Release
- Resolved
- Alert triggered

Recent session traces:
- Started at: 6:00 am, Page load: 0.569s
- Started at: 5:56 am, Page load: 7.814s

Page views with JS errors:
- 2.1%
- 1.5%
- 1.4%
- 1.4%
- 0.5%

AJAX response time:
- 7.6 sec
- 7.5 sec
- 5 sec
- 6 sec
- 2.5 sec
- 2.5 sec

Average Apdex:
- 0.91 (7.0)

Throughput by browser:
- Chrome
- Firefox
- IE
- Safari

Alerts:
- End User
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
With Alert Orchestration

Alerts → Alert Orchestrator → Notifications

Alert Orchestrator

On-call
### Metrics and Dashboards

#### Monitor Specific Uptime Past 7 Days
Since 7 days ago

<table>
<thead>
<tr>
<th>Availability</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBS Builds in INIT State</td>
<td>99.95</td>
</tr>
<tr>
<td>Estuary API Availability</td>
<td>100</td>
</tr>
<tr>
<td>F2.0 Renewroot Availability</td>
<td>100</td>
</tr>
<tr>
<td>Greenwave Decision API Availability</td>
<td>100</td>
</tr>
<tr>
<td>Retrieve waivers</td>
<td>100</td>
</tr>
<tr>
<td>MBS unknown error in the init state</td>
<td>100</td>
</tr>
<tr>
<td>MBS Builds API Availability</td>
<td>100</td>
</tr>
<tr>
<td>Datagrepper API Availability</td>
<td>100</td>
</tr>
<tr>
<td>WaierDB availability healthcheck</td>
<td>100</td>
</tr>
</tbody>
</table>

#### 7 Day Overall Uptime
Since 7 days ago

| Percentage | 99.94 |

#### Greenwave Decision Latency
Since 7 days ago

![Greenwave Decision Latency Graph]

#### 30 days uptime by monitor
Since 30 days ago

<table>
<thead>
<tr>
<th>Availability</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBS Builds in INIT State</td>
<td>93.07</td>
</tr>
<tr>
<td>MBS unknown error in the init state</td>
<td>94.23</td>
</tr>
<tr>
<td>MBS Builds API Availability</td>
<td>94.28</td>
</tr>
<tr>
<td>Greenwave Decision API Availability</td>
<td>95.0</td>
</tr>
<tr>
<td>Freshmaker Events API Availability</td>
<td>96.46</td>
</tr>
<tr>
<td>F2.0 Renewroot Availability</td>
<td>99.86</td>
</tr>
<tr>
<td>Estuary API Availability</td>
<td>100</td>
</tr>
<tr>
<td>Retrieve waivers</td>
<td>100</td>
</tr>
<tr>
<td>Datagrepper API Availability</td>
<td>100</td>
</tr>
</tbody>
</table>

#### 30 Day Overall Uptime
Since 30 days ago

| Percentage | 97.29 |

#### Overall Uptime Percentage by Week
Since 1 month ago

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Week</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.91</td>
<td>Week of January 21, 2019</td>
</tr>
<tr>
<td>100</td>
<td>Week of January 14, 2019</td>
</tr>
<tr>
<td>88.98</td>
<td>Week of January 7, 2019</td>
</tr>
<tr>
<td>99.45</td>
<td>Week of December 31, 2018</td>
</tr>
<tr>
<td>100</td>
<td>Week of December 24, 2018</td>
</tr>
</tbody>
</table>
How is this helping us today?
Holistic Monitoring Strategy (Tooling)
Open Source Alternative (Tooling)
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.

= Synthetic Location
Holistic Postmortem
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.
## Availability

### Customer Portal Status

#### Partial System Outage

<table>
<thead>
<tr>
<th>Service</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSO Authentication</td>
<td>Operational</td>
</tr>
<tr>
<td>Support Cases</td>
<td>Operational</td>
</tr>
<tr>
<td>Documentation</td>
<td>Major Outage</td>
</tr>
<tr>
<td>Search</td>
<td>Major Outage</td>
</tr>
<tr>
<td>Downloads</td>
<td>Major Outage</td>
</tr>
<tr>
<td>Subscription Management</td>
<td>Major Outage</td>
</tr>
<tr>
<td>Knowledgebase</td>
<td>Major Outage</td>
</tr>
</tbody>
</table>
From RUM we can see that this also affected user experience by increasing the overall page load time for end users:
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”
### APM + Logging

#### Error Rate
![Error Rate Graph](Image)

#### Top 5 Errors by Error Class

<table>
<thead>
<tr>
<th>Count</th>
<th>Transaction Name and Error Class</th>
<th>Error Message</th>
<th>First Occurrence</th>
<th>Last Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>105</td>
<td><code>/views_page</code> PDOException</td>
<td>Uncaught exception 'PDOException' with message 'SQLSTATE[HY000]: General error: 2006 MySQL server has gone away' in <code>/usr/share/drupal7/drupal/includes/database/database.inc:2227</code></td>
<td>2:06 AM</td>
<td>2:19 AM</td>
</tr>
</tbody>
</table>
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.
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.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Issue</th>
<th>Corrective Action</th>
<th>Owner</th>
<th>Result</th>
<th>Completion Date</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>CPDRUPAL-4059</td>
<td>Optimize slow query in Drupal. Jason reports he's optimized it from 4 seconds to 70ms</td>
<td>Jason Smith</td>
<td>In progress</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
jsprage@redhat.com
@caramelcode

Adam Minter
Associate Project Manager
aminter@redhat.com