DevOps: An Agile Architect's Perspective

Ron Crocker
Distinguished Engineer & Architect
Safe Harbor

This presentation and the information herein (including any information that may be incorporated by reference) is provided for informational purposes only and should not be construed as an offer, commitment, promise or obligation on behalf of New Relic, Inc. (“New Relic”) to sell securities or deliver any product, material, code, functionality, or other feature. Any information provided hereby is proprietary to New Relic and may not be replicated or disclosed without New Relic's express written permission.

Such information may contain forward-looking statements within the meaning of federal securities laws. Any statement that is not a historical fact or refers to expectations, projections, future plans, objectives, estimates, goals, or other characterizations of future events is a forward-looking statement. These forward-looking statements can often be identified as such because the context of the statement will include words such as “believes,” “anticipates,” “expects” or words of similar import.

Actual results may differ materially from those expressed in these forward-looking statements, which speak only as of the date hereof, and are subject to change at any time without notice. Existing and prospective investors, customers and other third parties transacting business with New Relic are cautioned not to place undue reliance on this forward-looking information. The achievement or success of the matters covered by such forward-looking statements are based on New Relic's current assumptions, expectations, and beliefs and are subject to substantial risks, uncertainties, assumptions, and changes in circumstances that may cause the actual results, performance, or achievements to differ materially from those expressed or implied in any forward-looking statement. Further information on factors that could affect such forward-looking statements is included in the filings New Relic makes with the SEC from time to time. Copies of these documents may be obtained by visiting New Relic's Investor Relations website at ir.newrelic.com or the SEC's website at www.sec.gov.

New Relic assumes no obligation and does not intend to update these forward-looking statements, except as required by law. New Relic makes no warranties, expressed or implied, in this presentation or otherwise, with respect to the information provided.
Introduction: My view of DevOps

Project Upscale

Dev: Scaling our system

Summary
Who am I?
What is New Relic?
Every minute New Relic handles:

40M+ HTTP requests
1.8B+ new data points
1000T+ events queried
The New Relic platform includes:

- 200+ unique services
- 4PB+ SSD storage
- 50+ agile teams
DevOps is a means for moving fast
Architecture
compliments
DevOps
DevOps is a *team sport*
Product

Platform

Site Engineering
Small, autonomous full ownership teams:

- Include Software Engineers & SREs
- Own runbooks and deploys
- Own on-call rotations and process
Embedded SREs are the key to DevOps
Introduction: My view of DevOps

Project Upscale

Dev: Scaling our system

Summary
Problem:

Growing very fast
Problem: Conway's Law
Project Upscale: Re-org *with a twist*
Redesign teams from bottom up
Self-selection
(is scary)
Team Charters
Service Transfer
Team Autonomy
Full Ownership
Facilitated self-service
Idea Flow
Product Council
Swarming
time allows you to release each project as you complete it, which increases the total value of your work.

Consider a team that has two projects. In this simplified example, each project has equal value; when complete, each project will yield $ in value every month. Each project takes three months to complete.

In Scenario A (see Figure), the team works on both projects simultaneously. To avoid task-switching penalties, they switch between projects every month. They finish Project 1 after five months and Project 2 after six. At the end of the seventh month, the team has earned $$$$$$.

In Scenario B, the team works on just one project at a time. They release Project 1 at the end of the third month. It starts making money while they work on Project 2, which they complete after the sixth month, as before. Although the team's productivity didn't change—the projects still took six months—they earned more money from Project 1. By the end of the seventh month, they earned $$$$$$$$$$. That's nearly twice as much value with no additional effort.

Something this easy ought to be criminal. What's really astounding is the number of teams that work on simultaneous projects anyway.
Experience

• Unlocked team velocity
• Service deprecation process
• Idea Flow focuses teams
• Improved availability
Don't Repeat Incidents Policy

• Time-boxed, focused efforts on incident root cause
NewRelic Emergency Response Force

- Small team (6-8 people) of Incident Commanders that intervene for more severe incidents
Experience

Good judgement comes from experience;

Experience comes from bad judgement
Agenda

Introduction: My view of DevOps

Project Upscale

Dev: Scaling our system

Summary
This is here
Facts about here

~3 posts/minute/agent

~8M posts/minute overall
Little's Law

\[ L = \lambda W \]

Queue Length
Arrival rate
Service time
\[ L = \frac{\lambda W}{n} \]

\[ \lambda = 8\text{M/minute} \]
\[ = 133/\text{millisecond} \]

\[ W = 0.139 \text{ millisecond} \]

\[ L = 18.5 \text{ to } 0.84 \quad n = 22 \]
END DETOUR
This is here
Facts about there

~3 posts/minute/agent

~14 posts/minute/agent

~8M posts/minute overall

~37M posts/minute overall
\[ L = \frac{\lambda W}{n} \]

\[ \lambda = 37 \text{M/minute} \]
\[ = 617/\text{millisecond} \]

\[ W = 0.139 \text{ millisecond} \]

\[ L = 3.89 \quad n = 22 \]
This is there
This is 3.89
\[ L = \frac{\lambda W}{n} \]

\[ \lambda = 37\text{M}/\text{minute} = 617/\text{millisecond} \]

\[ W = 0.139 \text{ millisecond} \]

\[ L = 3.89 \quad 0.84 \quad n = 22 \quad n = 102 \]
L = \lambda W/n

\lambda = 37M/minute
= 617/millisecond

W = 0.139 millisecond

L = \frac{3.89}{n} \quad n = 22

= 1.90 millisecond

= 0.84 millisecond

n = 51
Vortex

- De-coupled from DB
- Load *partially* passed downstream
- Coupled to service
- Client responsible for cache updates
Data stream
Vortex

- De-coupled from service
- Load not passed on
- Service owns updates

Agents

Account
Service

Account
Stream

Cache

Preload cache from Kafka
Dirac Event Transformer

- **Account ID**
- **Agent Run ID**
- **Event Data**

- **Decorated Events**
  - **App Name**
  - **Host Name**

**Agents**

- **Data not appropriate for a Data Stream**

**Cache**

- **Load partially passed downstream**
- **Directly coupled to DB**
- **DET responsible for cache updates**
- **4 hour fleet deploys**
Decorate the request

```
Start token

Client

Token

Create token

Server

Token

Send token

Store token

Create token

Send token

Send token
```
Streaming Event Decorator

- Standalone service with no dependencies
- Fleet deploy in seconds

Agents

<table>
<thead>
<tr>
<th>Account ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent Run ID</td>
</tr>
<tr>
<td>Event Data</td>
</tr>
</tbody>
</table>

SED

<table>
<thead>
<tr>
<th>Decorated Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Name</td>
</tr>
<tr>
<td>Host Name</td>
</tr>
</tbody>
</table>

Cache

Collector

Response

- Connect
- Agent Run Token

Agents

Data not appropriate for a Data Stream

DBs

©2008–18 New Relic, Inc. All rights reserved

60
Agenda

Introduction: My view of DevOps

Project Upscale

Dev: Improvements with positive Ops side-effects

Summary
DevOps enables teams to move fast
Project Upscale enabled New Relic to move fast
Don't forget about the Dev in DevOps
Thank You