The Smallest Possible SRE Team

Zach Thomas, Genesys
240 BCE
Mechanical Advantage
“It’s a people thing.”

“Our best tools for leverage are old and humble.”
“How Google Runs Production Systems”
Bring software engineering to operations.

vs.

Bring operations to software engineering.
Conway’s Law

“Organizations which design systems...are constrained to produce designs which are copies of the communication structures of these organizations.”
Team Types

1. Stream-aligned team
2. Platform team
3. Complex subsystem team
4. Enabling team
Sidebar: DevOps
SRE Enabling Team Framework

• Provide internal consulting services
• Meet with many teams
  • SRE reviews
  • Incident reviews
  • Fire drills
  • Game days
• You’re not the boss, you’re the helping hand
• Use reliability-related data to tell you where to focus
• Scale yourself with docs, tools, and automation
You don’t scale.
(at least, not at first)
SRE Reviews

• Look at the architecture diagram through the resilience lens
• Every box is a subsystem that can be slow or down
• Every line is a potentially flaky link
• Enumerate every dependency and the failover strategy for each
• Prove that alerting works with fire drills
Fire Drills

• Focused chaos exercises on a single service
• As the teams learn how it works, they become self-sufficient
• Anatomy of a fire drill:
  1. ensure traffic is flowing
  2. choose chaos to trigger
  3. describe a hypothesis about what the behavior will be
  4. monitor the service during the chaos
  5. document the surprises and the possible remediation
Game Days

- Larger scale exercise (all hands on deck)
- Involves platform-wide effects (e.g. sever an availability zone)
- Identify weaknesses
- Practice incident response
  - Do you know who’s on call?
  - Can you find your playbooks?
  - Do you have all the access you need?
  - How fast can you restore service?
Meet Your Teams Where They Are

1. Telemetry
2. Alerting
3. On-call rotations
4. Incident response procedures
5. Post-incident reviews
6. Chaos engineering
7. Service Level Objectives
<table>
<thead>
<tr>
<th>SRE practices used in respondents' work</th>
<th>0%</th>
<th>20%</th>
<th>40%</th>
<th>60%</th>
<th>80%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply software engineering to operations work</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity planning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blameless postmortems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offer an SLA, which describes the impact of not meeting the SLO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provide users with an SLO, describing the availability or performance of a service</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop SLIs, specific metrics that inform an SLO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Prioritize with Data

• Data is like oxygen for operations
• Coarse-grained data can still be useful
• For every incident, record:
  • time to detect
  • time to recover
  • blast radius
• Use the dependency graph of your services to put them into tiers
• Look at pager data for risks of burnout
• SLOs are an advanced priority-setting technique
Interlude

brew install ponysay
ponysay --pony pinkie I love SREcon!

< I love SREcon! >
Tools for Leverage

• Resist big application and UI frameworks
• The old ways are best, e.g. the humble shell! (fish FTW)
• Our public cloud providers put astonishing capabilities at our fingertips
• With httpie and jq, any REST API becomes a command line tool
• If we need new APIs, we can add them ourselves
Go Serverless

• If my team is tiny, I can’t afford to build anything that has a maintenance overhead
• Functions as a service
• Serverless databases
• Serverless pub-sub
Erebus (chaos engine)

1. Invoke it with a service name, chaos type, and duration
2. Find all instances by tag
3. Run the chaos command with AWS Systems Manager
4. Wait for n seconds
5. Restore instances to normal
Blacklight (SRE audit)

1. Listen for configuration change events
2. Execute rule lambdas to find violations
3. Notify service owners on violations

• Examples of rules:
  • Recommended alerts missing
  • Using the deprecated availability zone
  • Old redis engine version
Oathkeeper (SLOs)

• Store SLI and SLO definitions in DynamoDB
• SLI definition includes a query to one of our metrics systems
• Evaluation triggers the query, checks against the threshold, and returns a report as JSON
Incident Reviews

John Allspaw

How Your Systems Keep Running Day After Day

DevOps Enterprise Summit 2017
Chaos Engineering

“Do not scorch the earth.”

“Evolve the chaos as the system evolves.”

“Serverless chaos emerging.”

image credit: freepik.com/macrovector
Amplify With Communication

“The pen is mightier than a bunch of meetings.”

Counting Nines

You may have heard of “nines” of availability. If we say my service has “five nines” of availability, it means it’s typically calculated on an annualized basis by subtracting the duration of all our outage incidents from the availability as the proportion of requests that succeeded. Since New Relic captures the data for each transaction (Query Language) query to count the number of failed donut requests for the last seven days:

```
SELECT count(*) as errors
FROM Transaction
WHERE httpResponseCode IN ('500', '501', '502', '503', '504')
  OR response.status IN (500, 501, 502, 503, 504)
  AND transactionType = 'Web'
  AND appName LIKE 'donut (%)'
SINCE 1 week ago
```

SRE Minute on Availability
Getting Bigger?

• Offer an “SRE Rotation” on your team
• You can afford to build UIs!
• You can build a real training program
• The topology isn’t fixed
• SREs can live in other teams
• You could even make an SRE-focused platform team
Thank you!
@dysmento