



DevOps Institute



A woman in a grey business suit and white blouse is seated in a lecture hall, smiling and raising her right hand with two fingers raised in a 'V' shape. She is surrounded by other people in business attire seated in rows of black leather chairs. The background is slightly blurred, focusing on the woman in the foreground.

SITE RELIABILITY ENGINEERING FOUNDATIONSM

Sept2021

Tell Us a Little About Yourself

- Please let us know who you are
 - Name, organization and role
 - **SRE** experience
 - Why you are attending this course
 - What you expect to learn



How reliable are your services currently?

Site Reliability Engineering Foundation Course Goals

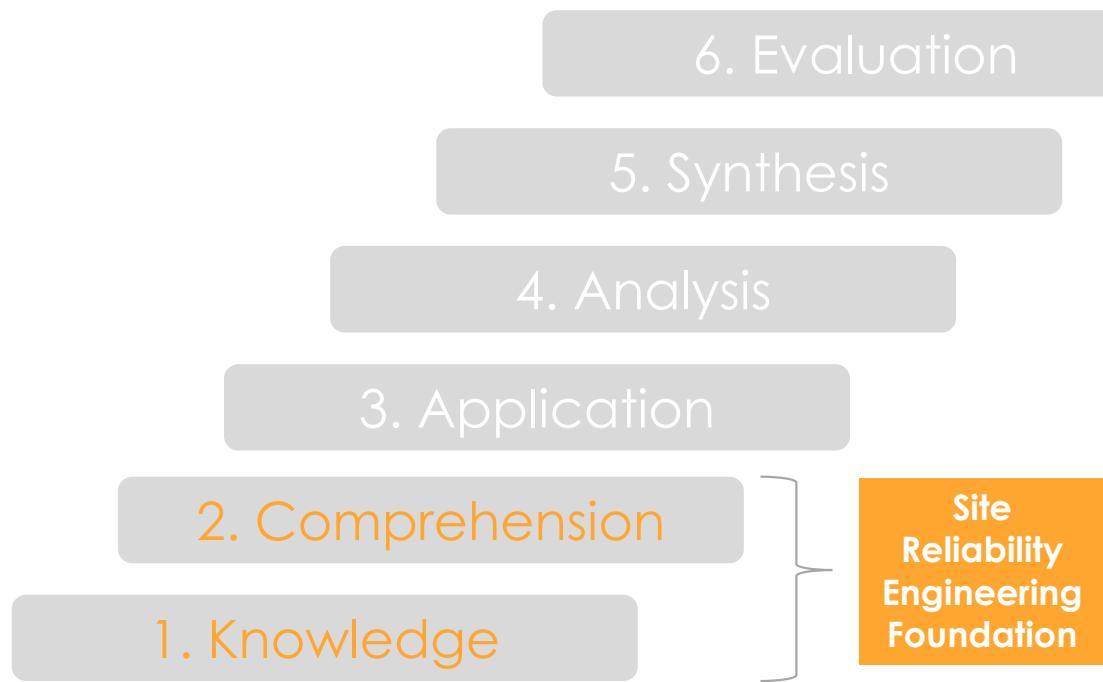
- Learn about **SRE**
- Understand its core vocabulary, principles, practices and automation
- Hear and share real life scenarios
- Have fun!



Pass the SRE Foundation Exam:

- 40 multiple choice questions
- 60 minutes
- 65% is passing
- Accredited by DevOps Institute
- Get your digital badge

About Bloom's Taxonomy



Bloom's Taxonomy is used to categorize learning objectives and, from there, assess learning achievements.

About DevOps Institute



**DevOps
INSTITUTE**

DevOps Institute is dedicated to advancing the human elements of DevOps success. As a global member association, DevOps Institute is the go-to hub connecting IT practitioners, industry thought leaders, talent acquisition, business executives and education partners to help pave the way to support digital transformation and the New IT.

DevOps Institute helps advance careers and professional development within the DevOps community through recognized certifications, research and thought leadership, events and the fastest-growing DevOps member community.

Site Reliability Engineering Foundation Course Content

Day 1	Day 2
Hello! Course & Class Welcome	Warming Up Game
Module 1 SRE Principles & Practices	Module 5 SRE Tools & Automation
Module 2 Service Level Objectives & Error Budgets	Module 6 Anti-Fragility & Learning from Failure
Module 3 Reducing Toil	Module 7 Organizational Impact of SRE
Module 4 Monitoring & Service Level Indicators	Module 8 SRE, Other Frameworks, The Future
Sample Examination Review	Examination Time

Module 1

SRE PRINCIPLES & PRACTICES

© DevOps Institute unless otherwise stated

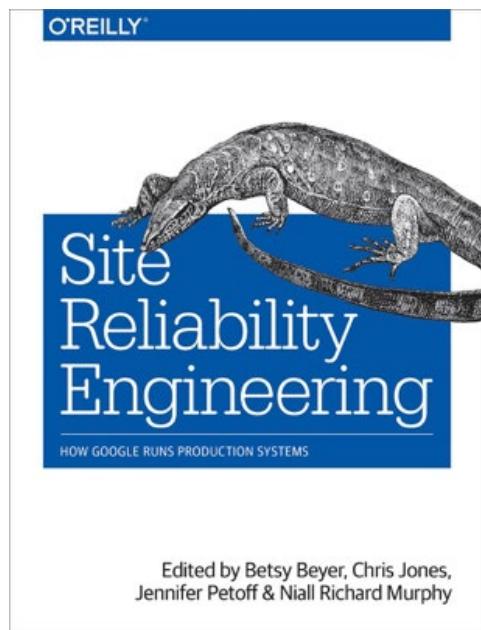
Module 1: SRE Principles & Practices

- What is Site Reliability Engineering?
- SRE & DevOps: What is the Difference?
- SRE Principles & Practices

Component	Module 1 Content
Video	DevOps & SRE (Google)
Case Story	Bloomberg
Discussion	Principles & Practices
Exercise	What do we do all day?

What is Site Reliability Engineering?

What is Site Reliability Engineering?



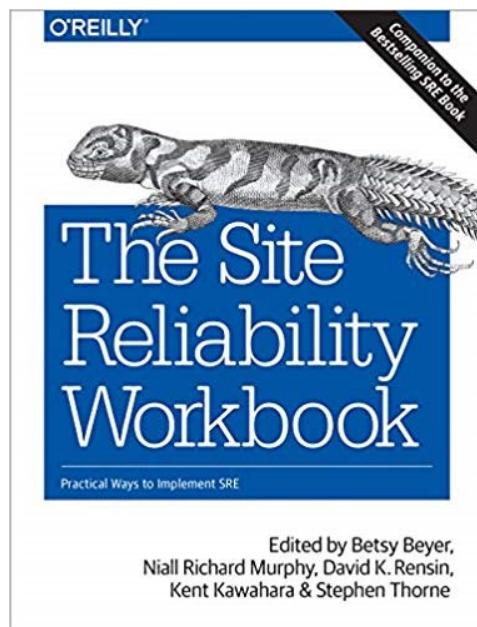
- Site Reliability Engineering (SRE) is a discipline that incorporates aspects of software engineering and applies them to infrastructure and operations problems
- Created at Google around 2003 and publicized via SRE books

“What happens when a software engineer is tasked with what used to be called operations.”

Ben Treynor, Google

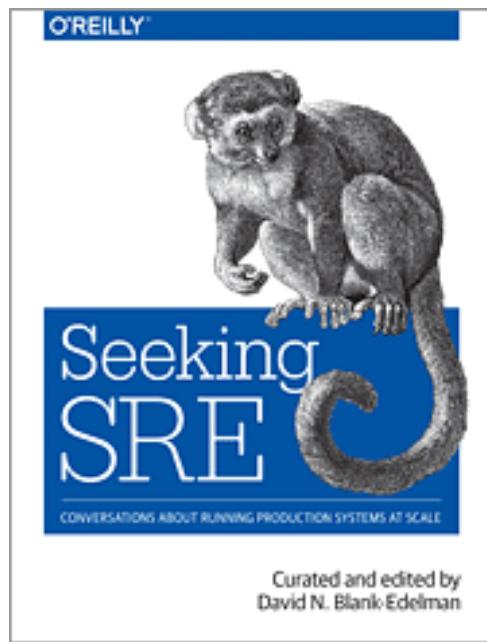


What is Site Reliability Engineering?



- The goal is to create ultra-scalable and highly reliable distributed software systems
- SRE's spend **50% of their time doing "ops" related work** such as issue resolution, on-call, and manual interventions
- SRE's spend **50% of their time on development tasks** such as new features, scaling or automation
- Monitoring, alerting and automation are a large part of SRE

What is Site Reliability Engineering?



- SRE has now spread beyond Google
- Many organizations running large-scale services are embracing SRE. Case stories in this course include:
 - Standard Chartered Bank
 - UK Dept Work & Pensions
 - Bloomberg
 - Home Depot
 - Trivago
 - Sage Group

“SRE is an engineering discipline devoted to helping an organization achieve the appropriate level of reliability.”

David N. Blank-Edelman,
Microsoft

What is Site Reliability Engineering?

The screenshot shows a website layout for "SRE WEEKLY". At the top, there's a navigation bar with links for "About SRE Weekly", "Bio", "Other Info", and "Sponsorship Information". Below the navigation is a section titled "BIO" featuring a portrait of a man with glasses and curly hair. To the right of the portrait is a bio text about Lex Neva, mentioning his interests in running large online services, Systems Engineering, troubleshooting, incident response, and his work at Fastly. It also notes his family life in Massachusetts. Below this is a note about the publication's editorial independence. On the right side of the bio section is a "Subscribe" form with fields for email address and a "Subscribe" button. Further down are sections for "RSS" (with a "Subscribe with RSS" button) and "Search Issues" (with a search bar and "Search" button).

- Scalability
- Availability
- Incident Response
- Automation

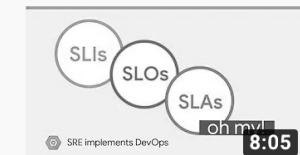


Module 1: SRE Principles & Practices

SRE & DevOps: What is the Difference?



Up next



SLIs, SLOs, SLAs, oh
SRE implements Dev...
Google Cloud Platform
66K views



Matt Parker: "The Great
Maths Mistakes" | Ta...
Talks at Google 58:41
408K views



World's Most Famous
Kevin Mitnick & Know...
Cyber Investing Summi...
493K views



VOXXED DAYS 10 Tips for failing bette...



What's the Difference Between DevOps and SRE? with Seth Vargo and Liz Fong-Jones (Google) (05:10)

15



Google Cloud Platform

Module 1: SRE Principles & Practices

SUBSCRIBE



UshersNewLook
Recommended for you

class SRE implements DevOps

► **DevOps**

is a set of practices, guidelines and culture designed to break down silos in IT development, operations, architecture, networking and security.

► **Site Reliability Engineering**

is a set of practices we've found to work, some beliefs that animate those practices, and a job role.



SRE & DevOps – What is the Difference?

DevOps (at Google) defines 5 key pillars of success:

1. Reduce organizational silos
2. Accept failure as normal
3. Implement gradual changes
4. Leverage tooling and automation
5. Measure everything



SRE is a "specific implementation of DevOps with some extensions." Google

SRE Principles & Practices

DISCUSSION

What are Principles & Practices?

#1 Operations Is a Software Problem



- The basic tenet of SRE is that doing operations well is a software problem
- SRE should therefore use software engineering approaches to solve that problem
- Software engineering as a discipline focuses on designing and building rather than operating and maintaining

Estimates suggest that anywhere between 40% and 90% of the total cost of ownership are incurred after launch

#2 Service Levels



- A Service Level Objective (SLO) is an availability target for a product or service (this is never 100%)
- In SRE services are managed to the SLO

SLOs need consequences if they are violated

#3 Toil



- Any manual, mandated operational task is bad
- If a task can be automated then it should be automated
- Tasks can provide the "wisdom of production" that will inform better system design and behavior

SREs must have time to make tomorrow better than today

#4 Automation



- Automate what is currently done manually
- Decide what to automate, and how to automate it
- Take an engineering-based approach to problems rather than just toiling at them over and over
- This should dominate what an SRE does
- Don't automate a bad process – fix the process first

SRE teams have the ability to regulate their workload

#5 Reduce the Cost of Failure



- Late problem (defect) discovery is expensive so SRE looks for ways to avoid this
- Look to improve MTTR (Mean Time to Recover/Repair)
- Smaller changes help with this
- Canary deployments

Failure is an opportunity to improve

#6 Shared Ownership



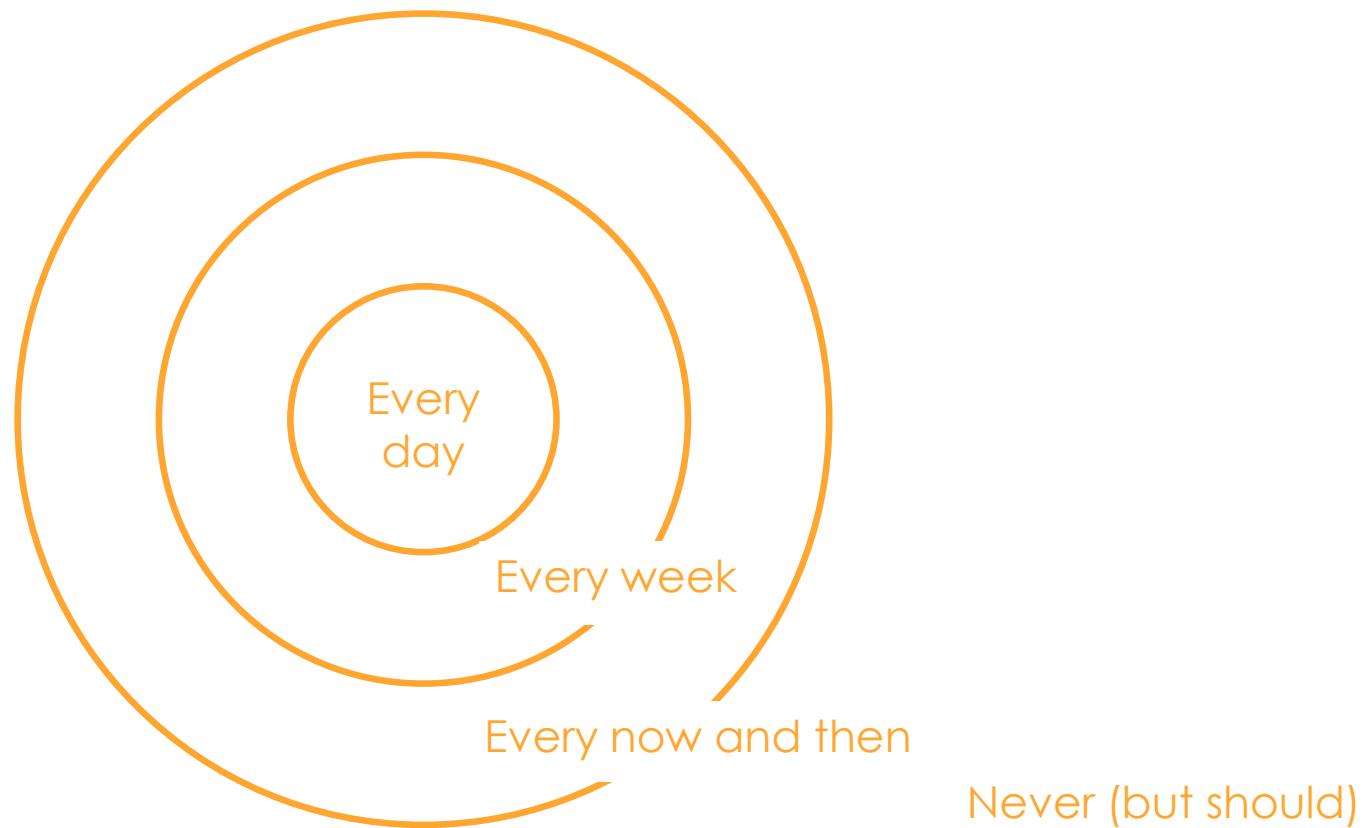
- SRE's share skill sets with product development teams
- Boundaries between "application development" and "production" (Dev & Ops) should be removed
- SRE's "shift left" and provide "wisdom of production" to development teams

Incentives across the organization are not currently aligned

EXERCISE

What Do We Do All Day?

What Do We Do All Day?



CASE STORY: Bloomberg

“Our SREs are united by a common vision of harnessing the power of automation through software development to deliver reliable, stable services to our clients. They care about how we can manage our infrastructure and applications more efficiently, and they do that through software development.”



Stig Sorensen
Manager, Production
Visibility
Bloomberg

“As an SRE, the challenge of scale is always a good one.”

Benefits

- Product stability improvements, through team collaboration
- Client cost savings, due to fewer outages
- Reduced daily grind managing servers and infrastructure, through more automation

“I LOVE my job! I mean, I really love it. I enjoyed being a full-stack developer and getting to deliver major projects for clients, but I like being an SRE far more. Before, the majority of my day was spent building features that were requested by some of our clients. Now, the work I do affects the entire platform and ALL of our clients.” Molly Struve, Lead Site Reliability Engineer at Kenna Security Inc



Module One Quiz

1	The term “site reliability engineering” was created by which organization?	a) Microsoft b) Apple c) Google d) LinkedIn
2	Which of these is not a pillar of DevOps success, as defined by Google?	a) Leverage tooling and automation b) Implement big-bang changes c) Measure Everything d) Reduce Organizational Silos
3	SLO is an acronym for:	a) System Load Objective b) Service Life Objective c) Straight Line Organization d) Service Level Objectives
4	Toil is:	a) An acronym for “Time Off In Lieu” b) Manual activities with no enduring value c) Google d) LinkedIn
5	Embracing SRE will bring value to organizations facing the challenge of what?	a) Loss of customers b) Staff Attrition c) Scope Creep d) Scale

Module One Quiz Answers

1	The term “site reliability engineering” was created by which organization?	a) Microsoft b) Apple c) Google d) LinkedIn
2	Which of these is <u>not</u> a pillar of DevOps success, as defined by Google?	a) Leverage tooling and automation b) Implement big-bang changes c) Measure Everything d) Reduce Organizational Silos
3	SLO is an acronym for:	a) System Load Objective b) Service Life Objective c) Straight Line Organization d) Service Level Objectives
4	Toil is:	a) An acronym for “Time Off In Lieu” b) Manual activities with no enduring value c) A Google product for managing SRE d) A KPI for SRE success
5	Embracing SRE will bring value to organizations facing the challenge of what?	a) Loss of customers b) Staff Attrition c) Scope Creep d) Scale

Module 2

SERVICE LEVEL OBJECTIVES & ERROR BUDGETS

© DevOps Institute unless otherwise stated

Module 2: Service Level Objectives & Error Budgets

- Service Level Objectives
- Error budgets
- Error budget policies

Component	Module 2 Content
Video	Risk & Error Budgets (Google)
Case Story	Evernote Home Depot
Discussion	Enforcing the Availability SLO
Exercise	SLO's for Your Organization

SLO's – Service Level Objectives

What is an SLO?

- An SLO (“Service Level Objective”) is a goal for how well a product or service should operate
- SLO's are tightly related to the user experience – if SLO's are being met then the user will be happy
- Setting and measuring¹ service level objectives is a key aspect of the SRE role
- The most widely tracked SLO is availability²
- Products and services could (and should) have several SLO's



SLO's are about making the user experience better

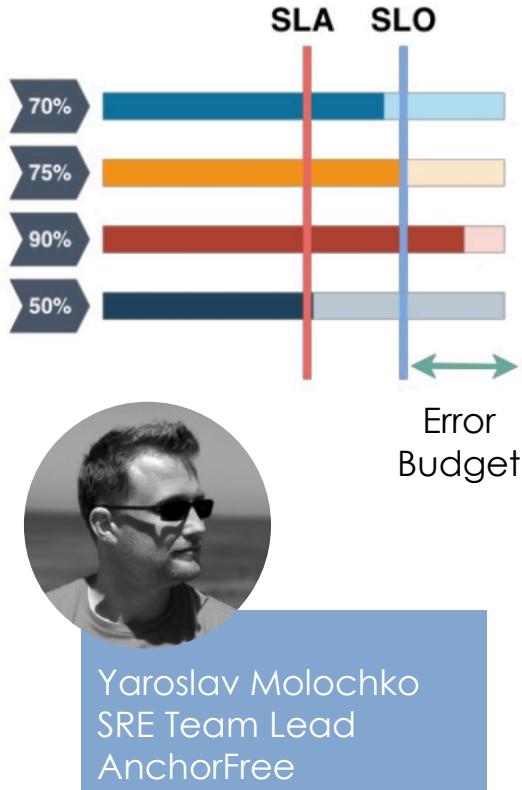
SLO's Are for Business

“Before getting into the technical details of a SLO, it is important to start the conversation from your customers' point of view: what promises are you trying to uphold?”

Ben McCormack, VP Operations,
Evernote



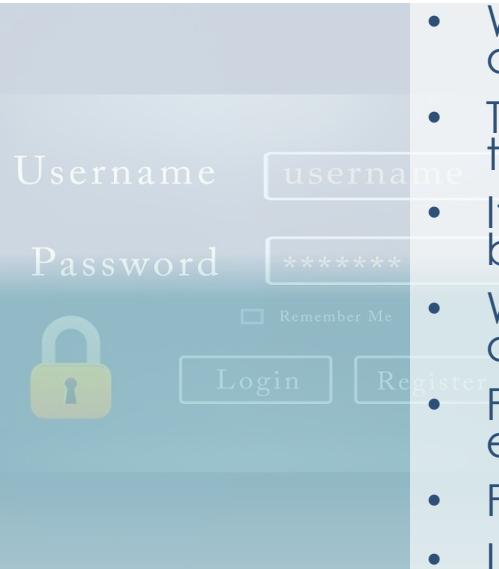
Example 1: SLO's & Error Budgets



- We decide that 99.9% of web requests (`www.....`) per month should be successful – this is a “service level objective”
- If there are 1 million web requests in a particular month, then up to 1,000 of those are allowed to fail – this is an “error budget”
- Failure to hit an SLO must have consequences – if more than 1,000 web requests fail in a month then some remediation work must take place – this is an “error budget policies”

Example 2: SLO's & Error Budgets

- Our service has an average login rate of 1,000 per hour in a rolling 31-day period (month) – or 744,000 per month ($31 * 24 * 1000$)
- We want 99% of logins each month to be successful - this is a “service level objective”
- This equates to “losing” roughly 7,440 logins a month – this is the “error budget”
 - If more 7,440 logins are lost in a month then we have breached the error budget.
 - We use a service level indicator (SLI) to tell us how many actual logins we get in a month.
 - For a particular month our actual logins were 726,560 - exceeding our error budget
 - Failure to hit an SLO must have consequences
 - In this case we instigated a business protection period preventing new releases – this is the “error budget policy”



Example 3: SLO's & Error Budgets

The screenshot shows a JIRA IT Service Desk interface. At the top, there are navigation links: JIRA, Dashboards, Projects, Issues, Service Desk, Portfolio, Create, and a search bar. Below the header, the page title is "IT Service Desk". The main content area displays a single support ticket:

- Key Information:** Ticket ID IT-680, Title "Upgrade database to next major release", Status "WAITING FOR SUPPORT" (highlighted in yellow), Priority "Major", Component "JIRA", Resolution "Unresolved".
- Details:** Type "Change", Labels "None".
- Description:** "I want to upgrade".
- Issue Links:** Relates to IT-39 Account problem (status WAITING FOR...) and IT-749 Access to DEV backend (status WAITING FOR...).
- Activity:** A section showing the timeline of the ticket.

- We decide that 75% of support tickets must complete automatically – this is a “service level objective”
- If there are 1,000 new support tickets raised each month 250 can be handled manually – this is an “error budget”
Failure to hit an SLO must have consequences – if more than 250 support tickets in a month require manual effort then some engineering work must take place – this is an “error budget policies”



“SLO’s are the most important component of SRE.”

Lyon Wong, co-founder,
Blameless - the SRE Platform

Adoption of SLO's

According to the 2019 Catchpoint SRE Survey the most popular SLO's are:

Availability	72%
Response time	47%
Latency	46%
We don't have SLOs	27%



Error Budgets

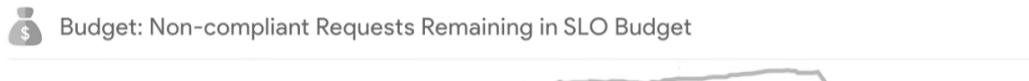
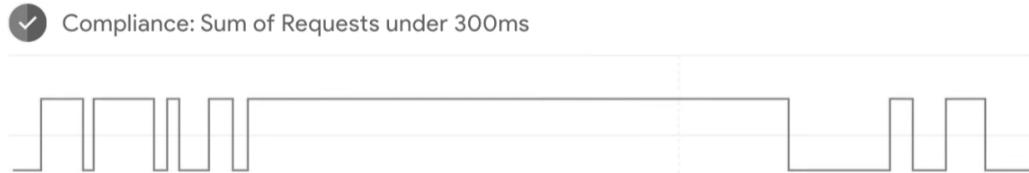
A black and white portrait of Benjamin Treynor Sloss, a man with short hair, looking slightly to his left with a thoughtful expression.

“100% is the wrong reliability target for basically everything.”

Benjamin Treynor Sloss, VP 24x7 Engineer
at Google

Module 2: SLO's & Error Budgets

43



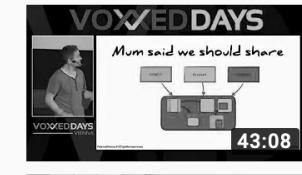
Up next



Toil and Toil Budgets implements DevOps)
Google Cloud Platform
Recommended for you



Matt Parker: "The Great Maths Mistakes" | Talks at Google
409K views



10 Tips for failing better in Microservices by David Farley
Devoxx
339K views

GOTO 2015 • Agile is

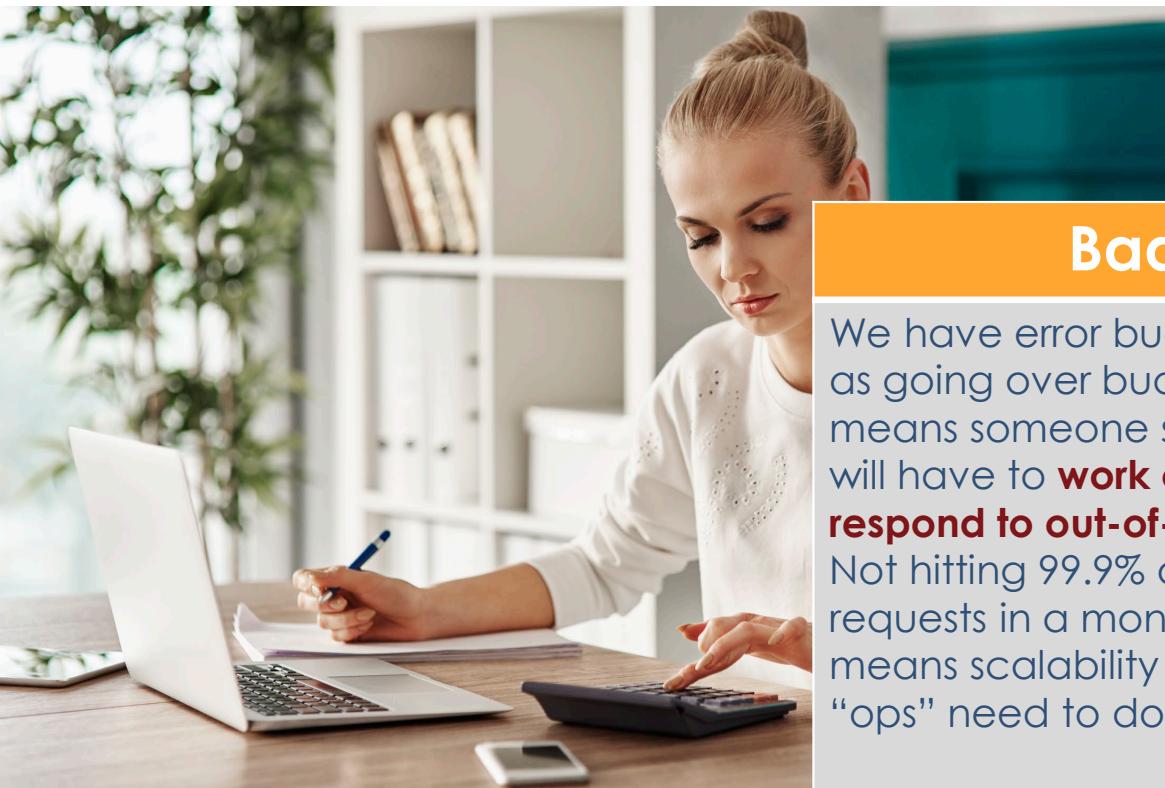


Risk and Error Budgets with Seth Vargo and Liz Fong-Jones of Google (06:17)

44



Error Budgets – Good and Bad



Bad	Good
<p>We have error budgets in SRE as going over budget usually means someone somewhere will have to work over-time or respond to out-of-hours issues. Not hitting 99.9% of HTTP requests in a month usually means scalability issues so “ops” need to do something</p>	<p>On the other hand SRE practices encourage you to strategically burn the budget to zero every month, whether it's for feature launches or architectural changes. This way you know you are running as fast as you can (velocity) without compromising availability</p>

Error Budgets – Fixed?



- But watch out – high-risk deployments or large "big-bang" changes have more likelihood of issues and therefore more chance of the error budget being blown
- This should encourage the Lean preference for small changes ("smaller batch size") to stay within the error budget.
- In some cases the error budget may need to change to accommodate complex releases but this needs to be agreed between Dev and Ops and the Business

Error Budget Policies

Consequence of Missed SLO's

Missed SLO's have noticeable impacts on business performance	
Lost Revenue	70%
Drop in Employee Productivity	57%
Lost Customers	49%
Social Media Backlash	36%



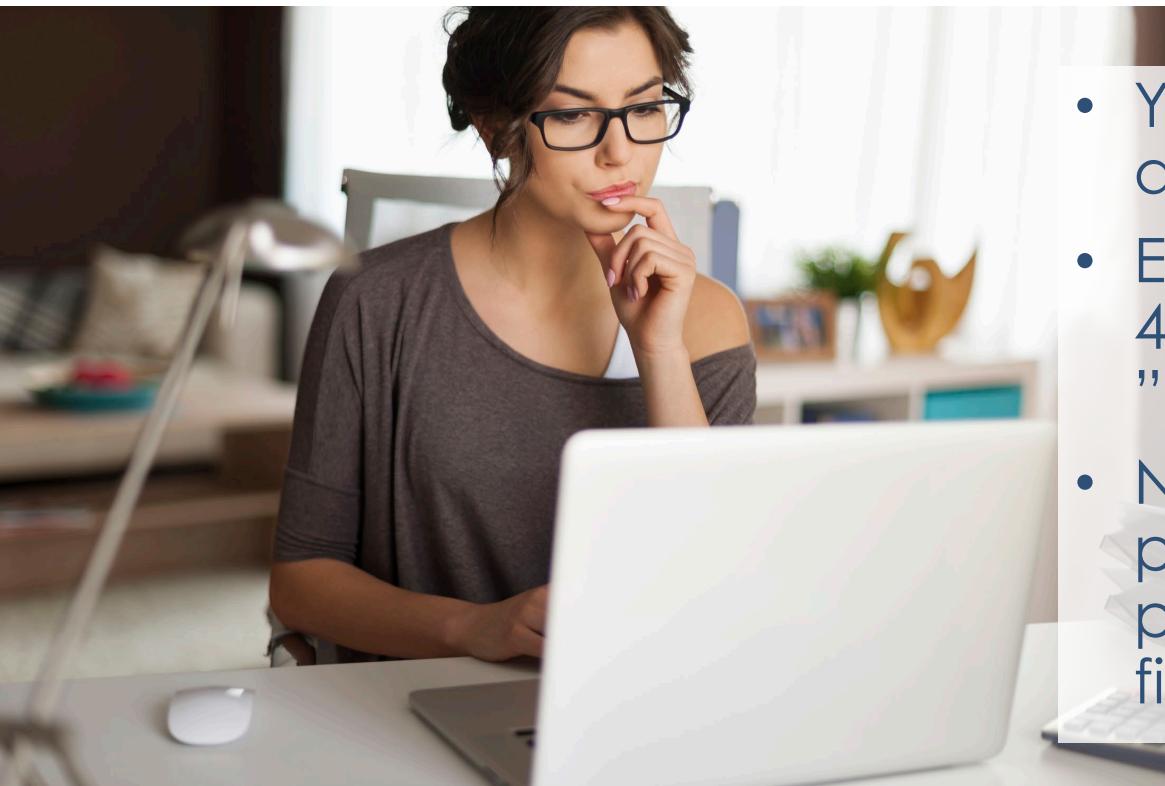
Consequences

“There will be no new feature launches allowed. Sprint planning may only pull post-mortem action items from the backlog. Software Development Team must meet with SRE Team daily to outline their improvements.”

Jennifer Petoff, Google



Another Example: Availability



- Your organization sets an availability SLO of 99.9%
- Every month this allows for 43 minutes of outages – the "error budget"
- New feature releases, patches, planned and unplanned downtime need to fit into this 43 minutes

DISCUSSION

What Error Budget Policies would you use to enforce an availability SLO?

CASE STORY: Evernote

“We wanted to ensure we initially focused on the most important and common customer need: the availability of the Evernote service for users to access and sync their content across multiple clients. Our SLO journey started from that goal.”

“We needed to ensure the move to GCP (Google Cloud Platform) did not dilute or mask our commitment to our users.”



Ben McCormack
VP Operations

Benefits

- Consistent focus on the user experience, whilst obtaining the benefits of cloud adoption
- Gain clarity around service availability and downtime
- Monitoring the right things, from a user perspective

CASE STORY: Home Depot

“Beforehand THD didn’t have a culture of SLO’s. Monitoring tools and dashboards were plentiful, but were scattered everywhere and didn’t track data over time. We weren’t always able to pinpoint the service at the root of a given outage. Often, we began troubleshooting at the user-facing service and worked backward until we found the problem, wasting countless hours. If a service required planned downtime, its dependent services were surprised. These disconnects caused confusion and disappointment between our software development and operations teams. We needed to address these disconnects by building a common culture of SLO’s.”



William Bonnell
SRE Director
The Home Depot

“We established a catchy acronym (VALET; as discussed later) to help the idea spread.”

Benefits

- Clearly understood SLO’s across the organization
- Wider involvement in setting SLO’s
- Joint responsibility model across Dev and Ops

The VALET Dimensions of SLO

	Dimension	SLO	Budget	Policy
V	Volume/traffic	Does the service handle the right volumes of data or traffic?	Budget: 99.99% of HTTP requests per month succeed with 200 OK	Address scalability issues
A	Availability	Is the service available to users when they need it?	Budget: 99.9% availability/uptime	Address downtime issues/outages, zero downtime deployments
L	Latency	Does the service deliver in a user-acceptable period of time?	Payload of 90% of HTTP responses returned in under 300ms	Address performance issues
E	Errors	Is the service delivering the capabilities being requested?	0.01% of HTTP requests return 4xx or 5xx status codes	Analyze and respond to main status codes, new functionality or infrastructure may be required
T	Tickets	Are our support services efficient?	75% of service tickets are automatically resolved	Automate more manual processes

Module 2: SLO's & Error Budgets

EXERCISE SLO's for Your Organization

Module Two Quiz

1	SLO is an acronym for:	a) Serious Local Outage b) Service Level Outcome c) Stored Local Object d) Service Level Objective
2	Which if these is <u>not</u> a recognized SLO?	a) Availability b) Latency c) Response time d) Total cost of ownership
3	Latency is:	a) Another name for velocity b) The time taken for a response to be delivered to a user c) A web request that fails d) An indicator of how well tested a service is
4	Which two items give a higher risk of an error budget being exceeded?	a) Automating the creation of users b) A big-bang release c) Rejecting all HTTP requests between 11pm and 12pm d) Optimizing the speed of your network
5	The benefit of adopting SLO's in conjunction with your users is what?	a) Less chance the user experience will be compromised b) Delivery velocity will be increased c) Development teams deliver better features d) Error budgets are easier to stay within

Module Two Quiz Answers

1	SLO is an acronym for:	a) Serious Local Outage b) Service Level Outcome c) Stored Local Object d) Service Level Objective
2	Which if these is <u>not</u> a recognized SLO?	a) Availability b) Latency c) Response time d) Total cost of ownership
3	Latency is:	a) Another name for velocity b) The time taken for a response to be delivered to a user c) A web request that fails d) An indicator of how well tested a service is
4	Which two items give a higher risk of an error budget being exceeded?	a) Automating the creation of users b) A big-bang release c) Rejecting all HTTP requests between 11pm and 12pm d) Optimizing the speed of your network
5	The benefit of adopting SLO's in conjunction with your users is what?	a) Less chance the user experience will be compromised b) Delivery velocity will be increased c) Development teams deliver better features d) Error budgets are easier to stay within

Module 3

REDUCING TOIL

© DevOps Institute unless otherwise stated

Module 3: Reducing toil

- What is toil?
- Why toil is bad
- Doing something about toil

Component	Module 3 Content
Video	Pragmatic Automation (Google Cloud)
Case Story	Accenture
Discussion	Benefits of toil reduction on individuals & teams
Exercise	Reducing Toil

Module 3: Reducing Toil

What is Toil?

What is Toil?

“Toil is the kind of work tied to running a production service that tends to be manual, repetitive, automatable, tactical, devoid of enduring value, and that scales linearly as a service grows.”

Vivek Rau, Google



Work is Toil If it is:

- **Manual**
 - Repetitive
 - Automatable
 - Tactical
 - No enduring value
 - Linear scaling
- Manual or semi-manual releases
 - Connecting to infrastructure to check something
 - Constant password resets

Toil: Finally a name for a problem we've all felt

Work is Toil If it is:

- Manual
 - **Repetitive**
 - Automatable
 - Tactical
 - No enduring value
 - Linear scaling
- Doing the same test over-and-over
 - Acknowledging the same alert every morning
 - Dealing with interrupts

Toil: Finally a name for a problem we've all felt

Work is Toil If it is:

- Manual
- Repetitive
- **Automatable**
- Tactical
- No enduring value
- Linear scaling
- Physical meetings to approve production deployments
- Manual starts/resets of equipment and components
- Creating users

Toil: Finally a name for a problem we've all felt

Work is Toil If it is:

- Manual
 - Repetitive
 - Automatable
 - **Tactical**
 - No enduring value
 - Linear scaling
- Known workarounds
 - On call responses

Toil: Finally a name for a problem we've all felt

Work is Toil If it is:

- Manual
 - Repetitive
 - Automatable
 - Tactical
 - **No enduring value**
 - Linear scaling
- Extracting some data

Toil: Finally a name for a problem we've all felt

Work is Toil If it is:

- Manual
- Repetitive
- Automatable
- Tactical
- No enduring value
- **Linear scaling**

- Manual scaling infrastructure

Toil: Finally a name for a problem we've all felt

Toil is NOT

Stuff I don't like
doing



Regular work, such as setting up that new device, developing that new alerting configuration for your service or working to remove clutter



Meetings, community events, planning sessions, HR events

Why Toil is Bad

Why Toil is Bad

Impact of High Toil	Individual	Organization
Slow Progress	Manual work and firefighting (toil) takes up the majority of time	New features do not get released quickly, missed value opportunity. Shortage of team capacity
Poor Quality	Manual work often results in mistakes, time consuming to fix, impact on reputation	Excessive costs in support of services
Career Stagnation	Career progress slows down due to working on the same things, no time for skills development. Best engineers working on low-level requests	Reputational damage, not a great place to work. Staff attrition rates increase.
Attritional	Toil is demotivating meaning people start looking elsewhere	Staff turnover results in extra costs and lost knowledge
Unending	Never ending deluge of manual tasks, no time to find solutions, more time spent managing backlog of tasks than fixing them	Toil requires engineering effort to fix, if there is no engineering time available it won't be fixed. SLA's being breached
Burnout	Personal and health problems due to overload and disruptive work patterns	Potential for litigation and negative publicity

Engineering Bankruptcy

“If you aren't careful, the level of toil in an organization can increase to a point where the organization won't have the capacity needed to stop it.”

Damon Edwards, Rundeck



Doing Something About Toil

A black and white close-up portrait of Benjamin Treynor Sloss. He is a man with short, dark hair, looking directly at the camera with a slight smile. He is wearing a dark, collared shirt. The background is a soft, out-of-focus grey.

“SRE is what happens
when you ask a software
engineer to design an
operations team.”

Benjamin Treynor Sloss, VP 24x7 Engineer at
Google

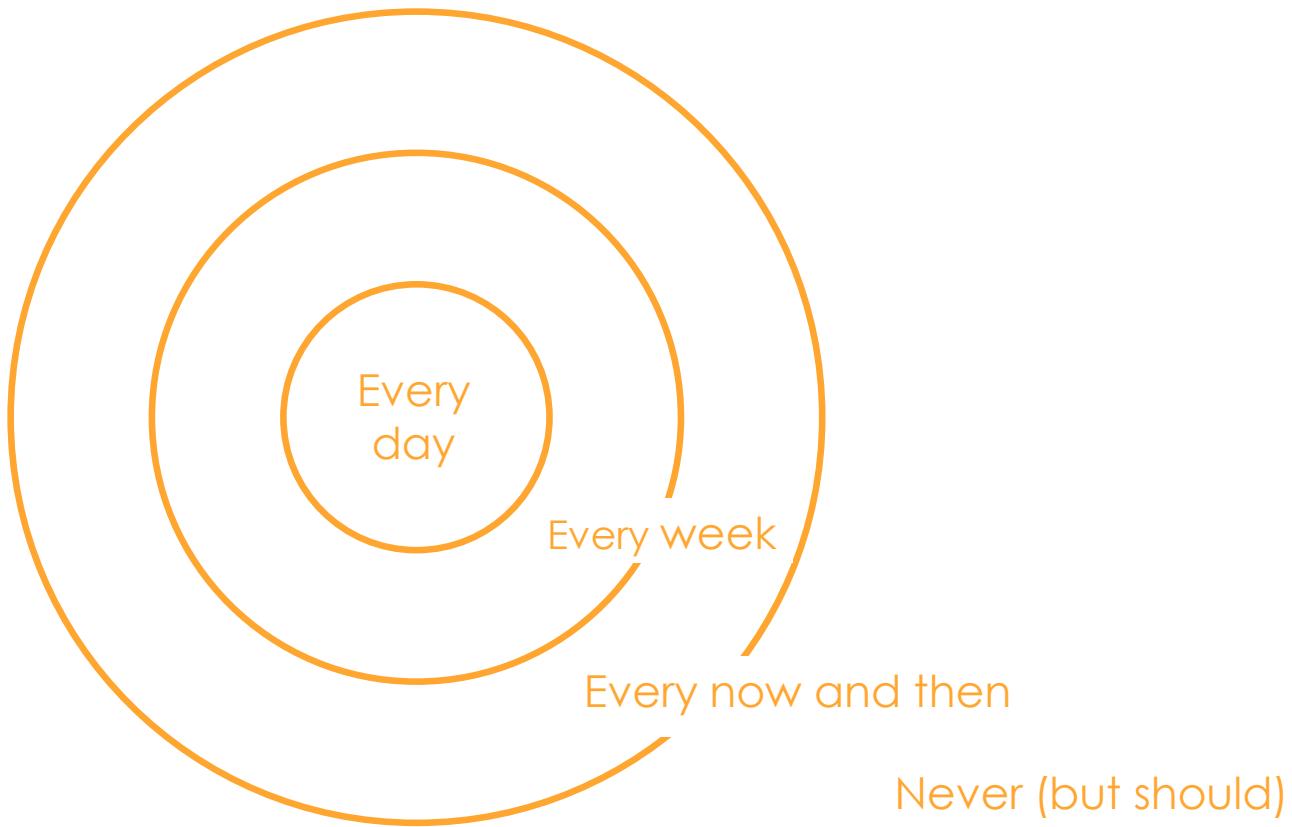
Module 3: Reducing Toil

73

What Can be Done About Toil

- Reducing toil requires engineering time
- Engineering work needed to reduce toil will typically be a choice of:
 - Creating **external automation** (i.e. scripts and automation tools outside of the service)
 - Creating **internal automation** (i.e. automation delivered as part of the service), or
 - **Enhancing the service** to not require intervention

Remember This?



EXERCISE

Reducing Toil

Making Engineering Time Available



- Google have an advertised goal of keeping operational work (i.e. toil) below 50% of an engineer's time
- At least 50% of each SRE's time should be spent on engineering project work that will either reduce future toil or add service features
- The 50% rule ensures that one teams or person does not become the “ops” (team/person)
- 50% is an average to reflect real world scenarios

Moving Towards SRE at Slack



- Slack moved from 100 AWS instances to 15,000 instances over 4 years
- Excessive toil caused by low-quality, noisy alerting
- Ops teams were so consumed by interrupt-driven toil that they were un-able to make progress on improving reliability
- Slack explicitly committed to the importance of reliability over feature velocity
- Operational ownership of services pushed back into the dev teams resulting in the teams making the code fixes necessary to stop the incident alerts

Is it Worth Automating Everything?

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE
EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

		HOW OFTEN YOU DO THE TASK					
		50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
HOW MUCH TIME YOU SHAVE OFF	1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
	5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
	30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
	1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
	5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
	30 MINUTES	6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS	
	1 HOUR	10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS	
	6 HOURS			2 MONTHS	2 WEEKS	1 DAY	
	1 DAY				8 WEEKS	5 DAYS	





"Automate yourself out of a job"

What if we're asked to automate something we have no idea how to do?

Up next



Java Project Tutorial
Login and Register Form
1BestCsharp blog
Recommended for you



Matt Parker: "The Great Maths Mistakes" | Talks at Google
Recommended for you



ITT 2016 - Kevlin Henney: Seven Ineffective Coding Practices
Istanbul Tech Talks
Recommended for you

10 Tips for failing hard



Pragmatic Automation with Max Luebbe of Google Cloud (04:45)

Module 3: Reducing Toil

[SUBSCRIBE](#)



Pragmatic Dave Trowbridge
GOTO Conference 2016

DISCUSSION

Benefits of toil reduction on individuals and teams in your organization?

CASE STORY: Accenture

"So our initial exercise was to come up with a view about what constitutes toil for the ADOP team. We implemented a 100 percent use of Jira tickets for tracking work. No gaps. We changed our Jira workflow to pop up the worklog window on every state transition. This allowed us to capture the amount of time spent on each ticket. What did we do with this information? We built automation to remove the need for toil-related work. But, crucially, because we were able to demonstrate just how much above 50 percent of our time was spent on toil, we had a clear mandate to prioritize toil payback stories and to actually work on them."



Mark Rendell
Principal Director
Accenture

"The team supporting the platform were inundated with toil to the point where they could do little else."

Benefits

- Hugely positive in protecting the team
- Reducing staff turnover
- All work made visible

Module Three Quiz

1	Which of these is an example of toil?	a) Auto scaling of cloud infrastructure b) Self service data queries c) Automated password resets d) Manual deployments
2	Engineering bankruptcy is when:	a) Fixing toil is not prioritized by the business b) Engineers work 100% on toil c) The development team have no money d) There are no engineers available to work on a task
3	Which of these approaches can be used to reduce toil?	a) Hire more engineers b) Shift ownership of toil to the development team c) Build a bigger datacenter d) Reading the Google SRE handbook
4	Which of these is <u>not</u> a way of fixing toil	a) Outsourcing operations b) Internal automation c) External automation d) Service enhancements
5	Which of these can you automate?	a) Brain storming sessions to solve a problem b) Taking time off c) Building infrastructure d) Promotion decisions

Module Three Quiz Answers

1	Which of these is an example of toil?	a) Auto scaling of cloud infrastructure b) Self service data queries c) Automated password resets d) Manual deployments
2	Engineering bankruptcy is when:	a) Fixing toil is not prioritized by the business b) Engineers work 100% on toil c) The development team have no money d) There are no engineers available to work on a task
3	Which of these approaches can be used to reduce toil?	a) Hire more engineers b) Shift ownership of toil to the development team c) Build a bigger datacenter d) Reading the Google SRE handbook
4	Which of these is <u>not</u> a way of fixing toil	a) Outsourcing operations b) Internal automation c) External automation d) Service enhancements
5	Which of these can you automate?	a) Brain storming sessions to solve a problem b) Taking time off c) Building infrastructure d) Promotion decisions

Module 4

MONITORING & SERVICE LEVEL INDICATORS

© DevOps Institute unless otherwise stated

Module 4: Monitoring & Service Level Indicators

- SLI's - Service Level Indicators
- Monitoring
- Observability

Component	Module 4 Content
Video	SLI's & Reliability Deep Dive (Microsoft)
Case Story	Trivago
Discussion	What do you monitor now?
Exercise	Set measurable objectives for your services

DISCUSSION

What Do You Monitor Now?

SLI's – Service Level Indicators

SLI's for Measurement

“SLI's are ways for engineers to communicate quantitative data about systems.”

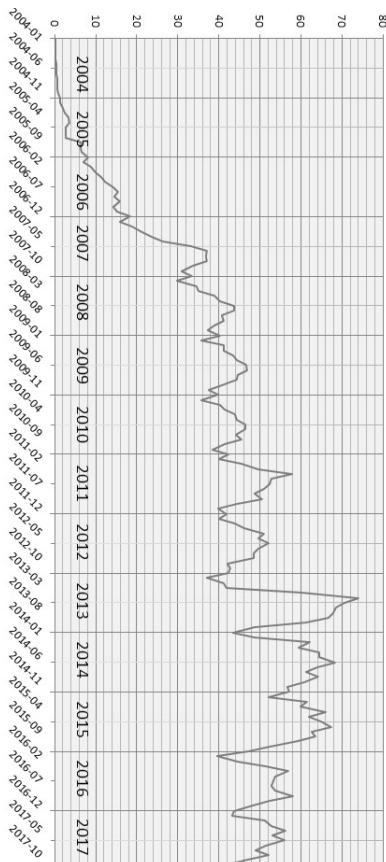
Ram Lyengar, Plumbr.io



Module 4: Monitoring & SLI's

Example

Let's Revisit an Earlier Example



- We decided that 99.9% of web requests (`www.....`) per month should be successful – this was the “service level objective”
- If there were 1 million web requests in a particular month, then up to 1,000 of those were allowed to fail – this was the “error budget”
- **In this example the “service level indicator” (SLI) is “web requests” so we need a way to track and record this data**

SLI Measurement

While many numbers can function as an SLI, it is generally recommended to treat the SLI as the ratio of two numbers: the number of good events divided by the total number of events. For our example this is:

- Number of successful (HTTP) web requests / total (HTTP) requests (success rate)

SLI Measurement

Many indicator metrics are naturally gathered on the server side, using a monitoring system such as Prometheus, or with periodic log analysis—for instance, HTTP 500 responses as a fraction of all requests.

Some service level indicators may also need client-side data collection, because not measuring behavior at the client can miss a range of problems that affect users but don't affect server-side metrics.

SLI Measurement

SLI measurement needs also to be time-bound in some way

The time horizon may vary depending on the organization and the SLO

- For web requests per month, the time horizon is clear
- SLO's such as "successful bank payments" may require a broader horizon if bank payments are only made once or twice per month

SLO's & SLI's

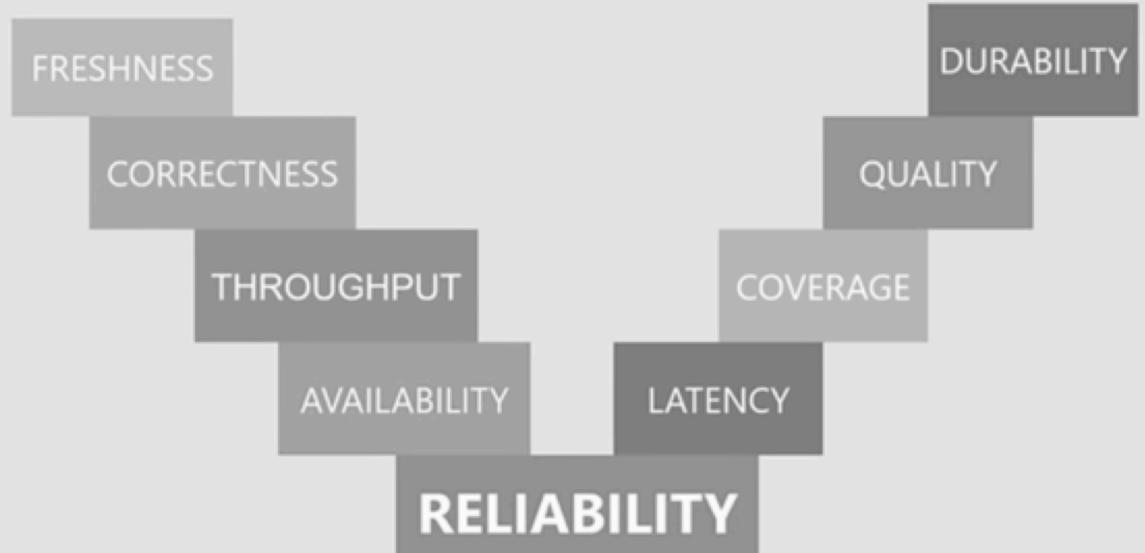
We use monitoring tools to measure SLI's constantly, aggregating across suitable time periods

Our SLO's are what we expect - monitoring our SLI's will tell us if we are meeting a SLO or not – they also tell us how much of our error budget is left (if any)



Microsoft Ignite

It has to be about the customer



SLI & Reliability Deep-Dive
with David N. Blank-Edelman (Microsoft)
(08:35)

Monitoring

Monitoring Definitions

Monitoring

System monitoring is the use of a hardware or software component to monitor the system resources and performance of a computer system



Monitoring Definitions

Telemetry

is the highly automated communications process by which measurements are made and other data collected at remote or inaccessible points and transmitted to receiving equipment for monitoring. The word is derived from Greek roots: tele = remote, and metron = measure.



Monitoring Definitions

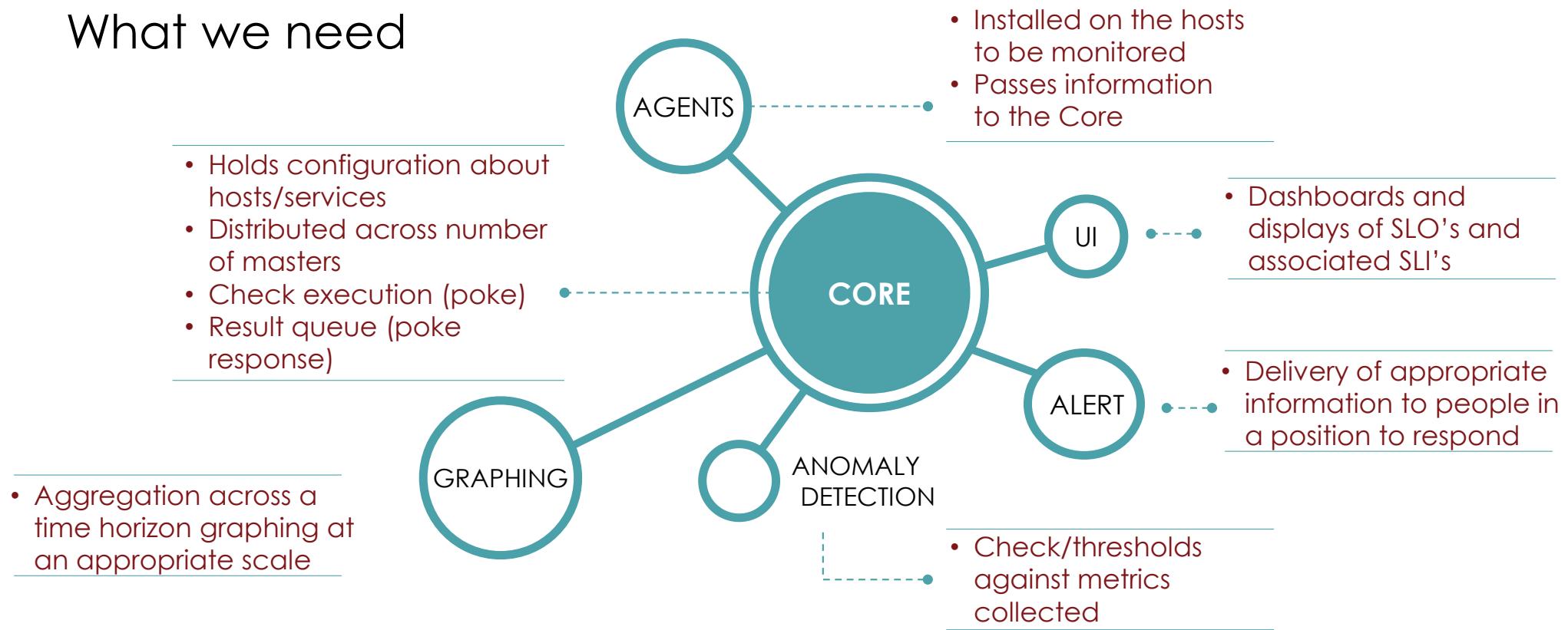
Application Performance Management (APM)

is the monitoring and management of performance and availability of software applications. APM strives to detect and diagnose application performance problems to maintain an expected level of service.



Monitoring Anatomy

What we need

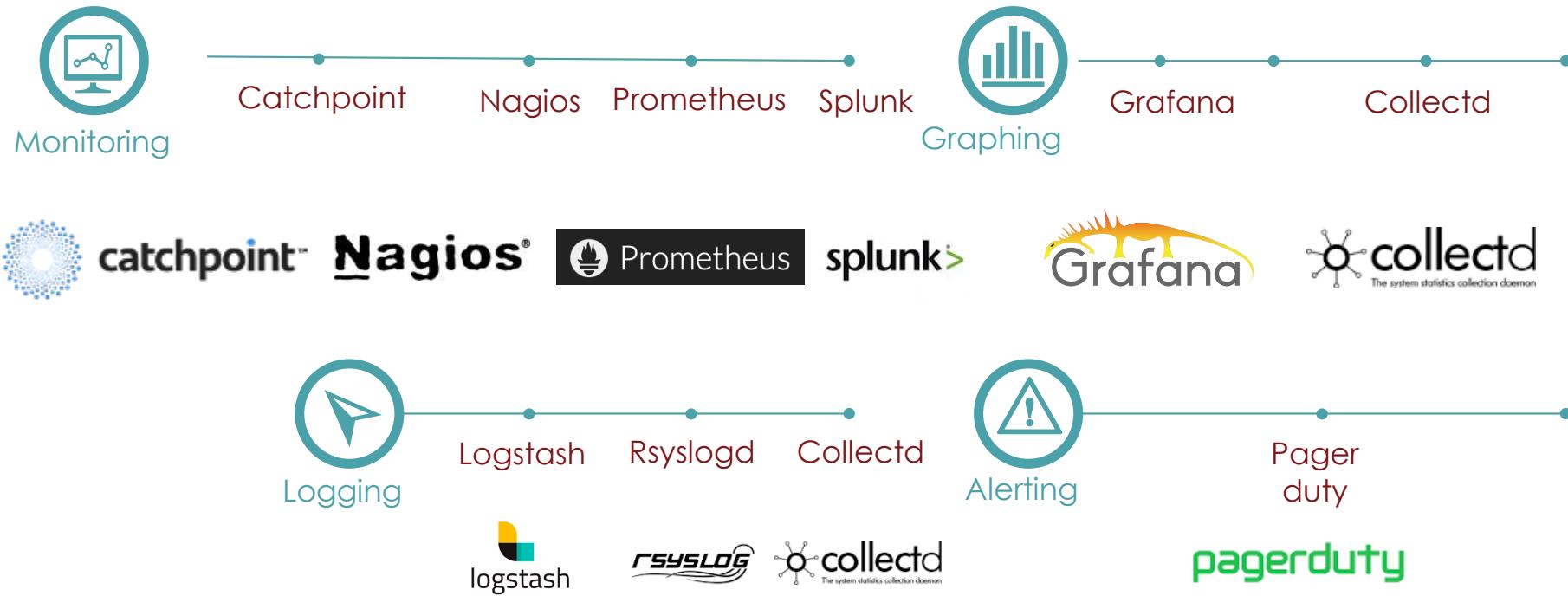


Module 4: Monitoring & SLI's

© DevOps Institute unless otherwise stated

101

SLI Supporting Tools



Monitor SLI's

“We need to make sure that monitoring is effective without drowning in a sea of non-actionable alerts. The path to success is to instrument everything, but only monitor what truly matters”

Todd Palino, Senior Staff Engineer
Site Reliability at LinkedIn



CASE STORY: Trivago

"My job is to make trivago products faster for customers accessing our services anywhere on the globe. Our SLO's focus on hotel search response times, not on specific parts of our service (or technology stack). You can only improve [SLO's] that you can measure and digital end-user perspective monitoring [using tools like Catchpoint] allow us to check that our hotel search response time indicator is delivering our SLO."



Xoan Vilas
Performance Lead
Trivago

"We have full visibility of how our services and systems are really delivering hotel search services internationally."

Benefits

- SLO's focused on the user experience, not tech
- Can spot performance issues with an ISP or CDN anywhere in the world
- Data collected by monitoring is used in diagnostics/RCA
- Feedback loop to Dev teams results in optimal solution

Module 4: Monitoring & SLI's

Observability

Monitoring & Observability

- Distributed, complex services running at scale with unpredictable users and variable throughput means there are millions of different ways that things can go wrong
- But we can't anticipate them all (monitoring myth)
- Externalizing all the outputs of a service allows us to infer the internal state of that service (observable)

Monitoring & Observability

monitor :
[mon-i-ter]

verb (used with object)

- to observe, record, or detect (an operation or condition) with instruments that have no effect upon the operation or condition.

verb (used without object)

- to serve as a monitor, detector, supervisor, etc.

Monitoring & Observability

“Monitoring is a verb; something we perform against our applications and systems to determine their state. From basic fitness tests and whether they’re up or down, to more proactive performance health checks. We monitor applications to detect problems and anomalies. “

Peter Waterhouse, CA



Monitoring & Observability

observable :

[uh b-**zur**-vuh-buh l]

noun

- capable of being or liable to be observed; noticeable; visible; discernible:
- deserving of attention; noteworthy.

Monitoring & Observability

“Observability, as a noun, is a property of a system, it's a measure of how well internal states of a system can be inferred from knowledge of its external outputs. Therefore, if our IT systems don't adequately externalize their state, then even the best monitoring can fall short”

Peter Waterhouse, CA



Why Observability is Important

Bolting on monitoring tools after the event does not scale:

- Rapid rate of service growth
- Dynamic architectures
- Container workloads
- Dependencies between services
- Customer experience matters more



Observability = Better Alerting

We need to improve our “signal” to “noise” ratio so we focus alerts on key issues

Solving
Noisy Alerts



1. Generate One Alert for One Service (Versus One Metric)
2. Use Analytics to Learn Normal Behavior
3. Improve Alerting with Multi-Criteria Alerting Policies

We need to infer what is “normal” about a service

Why Observability looks like

- Distributed tracing
- Event logging
- Internal performance data
- Application instrumentation
- Identify individual user experiences
- Fewer paging alerts - not more
- Inquisitive / what-if questions

SLO's, SLI's & Observability

- SLO's are from a user perspective and help identify what is important
 - E.g. 90% of users should complete the full payment transaction in less than one elapsed minute
- SLI's give detail on how we are currently performing
 - E.g. 98% of users in a month complete a payment transaction in less than one minute
- Observability gives use the normal state of the service
 - 38 seconds is the “normal” time it takes users to complete a payment transaction when all monitors are healthy

Be Proactive

“This rich ecosystem of introspection and instrumentation is not particularly biased towards the traditional monitoring stack's concerns of actionable alerts and outages.”

Charity Majors, CEO
Honeycomb



EXERCISE

Set measurable objectives for your service

EXERCISE

Set measurable objectives for your service

1. Map your user journeys

EXERCISE

Set measurable objectives for your service
2. Prioritize the most important user journey

EXERCISE

Set measurable objectives for your service

3. Define what "good" means to users

EXERCISE

Set measurable objectives for your service

4. Map out high-level system components

EXERCISE

Set measurable objectives for your service

5. Define the SLIs

EXERCISE

Set measurable objectives for your service

6. Make the service “observable”

Module Four Quiz

1	What does SLI stand for?	a) Service Life Indicator b) Service Level Integration c) Service Level Indicator d) Service Light Indicator
2	SLI's are best represented as:	a) A ratio of two numbers b) A fixed list of allowable values c) The format HH:MM d) Hexadecimal
3	What approach would you use to measure the internal performance of a service	a) Canary Testing b) Application Performance Management c) Ping d) Selenium
4	Which of these is a popular monitoring tools?	a) Bladerunner b) Hal c) Yoda d) Prometheus
5	Which of these is <u>not</u> a technical element of observability?	a) Number of failed logins b) Distributed tracing c) Event logging d) Internal performance data

Module Four Quiz Answers

1	What does SLI stand for?	a) Service Life Indicator b) Service Level Integration c) Service Level Indicator d) Service Light Indicator
2	SLI's are best represented as:	a) A ratio of two numbers b) A fixed list of allowable values c) The format HH:MM d) Hexadecimal
3	What approach would you use to measure the internal performance of a service	a) Canary Testing b) Application Performance Management c) Ping d) Selenium
4	Which of these is a popular monitoring tools?	a) Bladerunner b) Hal c) Yoda d) Prometheus
5	Which of these is <u>not</u> a technical element of observability?	a) Number of failed logins b) Distributed tracing c) Event logging d) Internal performance data

A photograph of a woman in a grey business suit and white blouse sitting in a lecture hall. She is smiling and has her right hand raised with her index finger pointing upwards, as if asking a question or volunteering. Behind her, several other people in business attire are seated in rows of black leather chairs, looking towards the front of the room. The background is slightly blurred.

SITE RELIABILITY ENGINEERING FOUNDATION

Site Reliability Engineering Foundation Course Content

Day 1		Day 2	
Hello! Course & Class Welcome		Warming Up Game	
Module 1	SRE Principles & Practices	Module 5	SRE Tools & Automation
Module 2	Service Level Objectives & Error Budgets	Module 6	Anti-Fragility & Learning from Failure
Module 3	Reducing Toil	Module 7	Organizational Impact of SRE
Module 4	Monitoring & Service Level Indicators	Module 8	SRE, Other Frameworks, The Future
Sample Examination Review		Examination Time	

Module 5

SRE TOOLS & AUTOMATION

© DevOps Institute unless otherwise stated

Module 5: SRE Tools & Automation

- Automation Defined
- Automation Focus
- Hierarchy of Automation Types
- Secure Automation
- Automation Tools

Component	Module 5 Content
Video	Ironies of Automation (Microsoft)
Case Story	Standard Chartered
Discussion	Automation “Greatest Hits”
Exercise	How Much Automation Do You Have?

Automation Defined

Automation Gives Us

- Consistency – a machine will be more consistent than a human
- A platform upon which to build, re-use, extend
- Faster action, faster fixes
- Time savings

“For SRE, automation is a force multiplier, not a panacea.”

– Niall Murphy, Google SRE

Automation Requires:

- A problem to be solved:
 - Eliminating toil
 - Improving SLO's
- Appropriate tooling
- Engineering effort
- Measurable outcomes



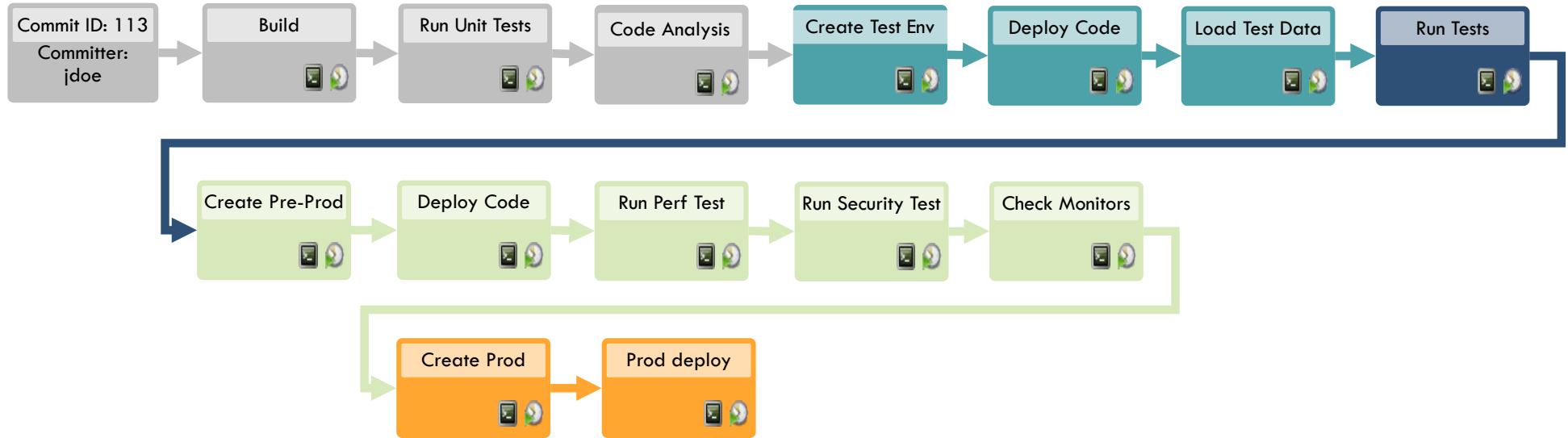
“For SRE, automation is a force multiplier, not a panacea.” – Niall Murphy, Google SRE

DISCUSSION

Automation “Greatest Hits”

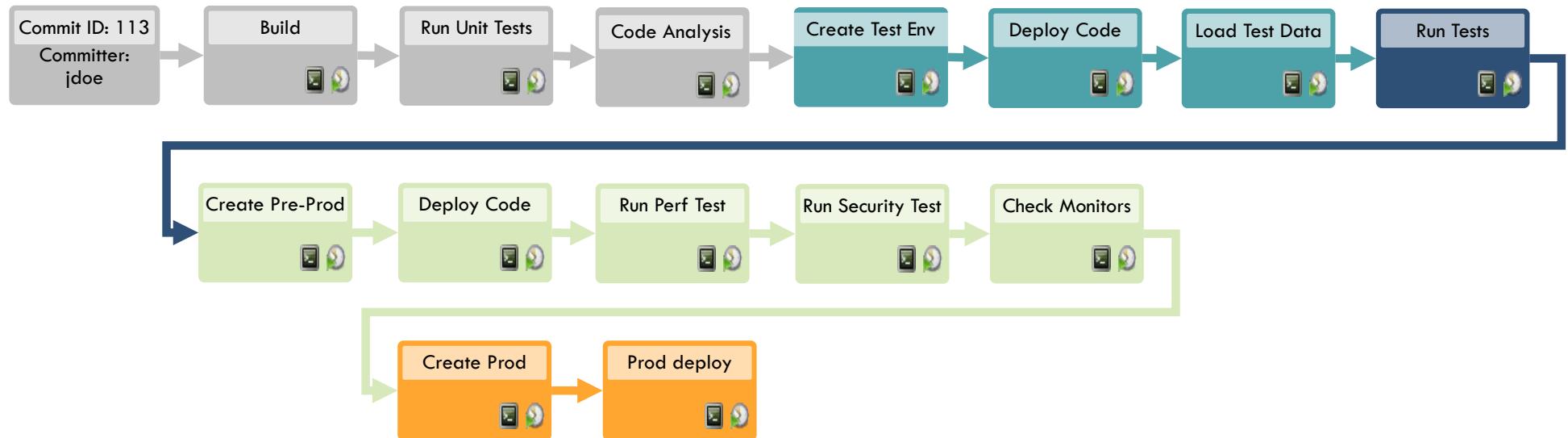
Automation Focus

Automation Focus: Typical DevOps Delivery Pipeline

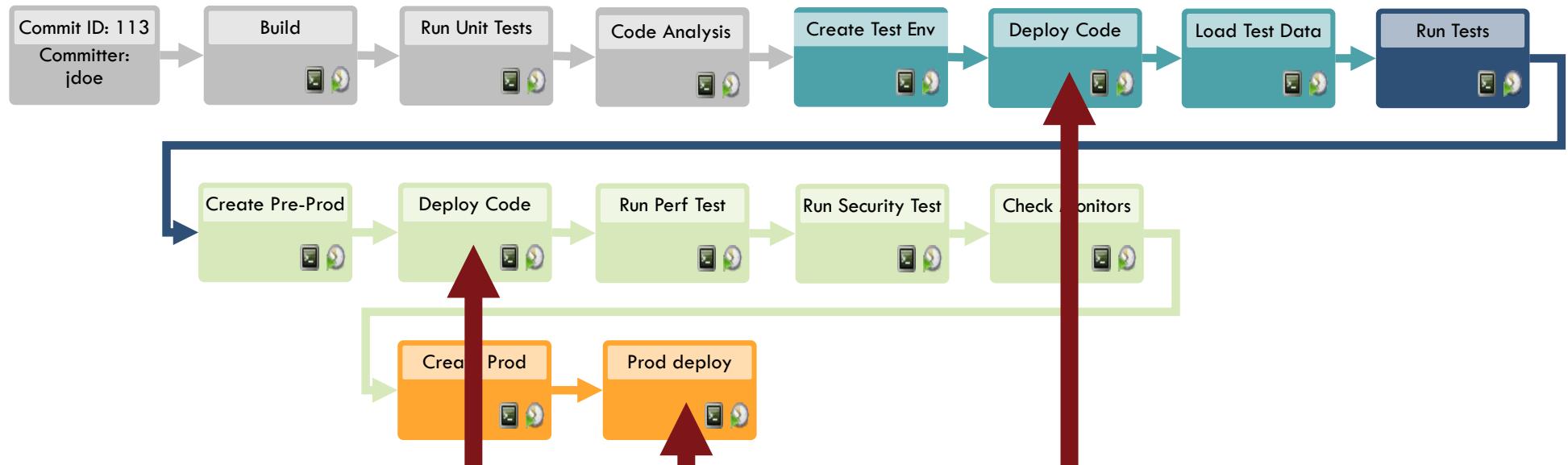


Automation Focus

A lot of automation effort is “Dev” led (left-to-right), DevOps with a “big Dev and small Ops”.

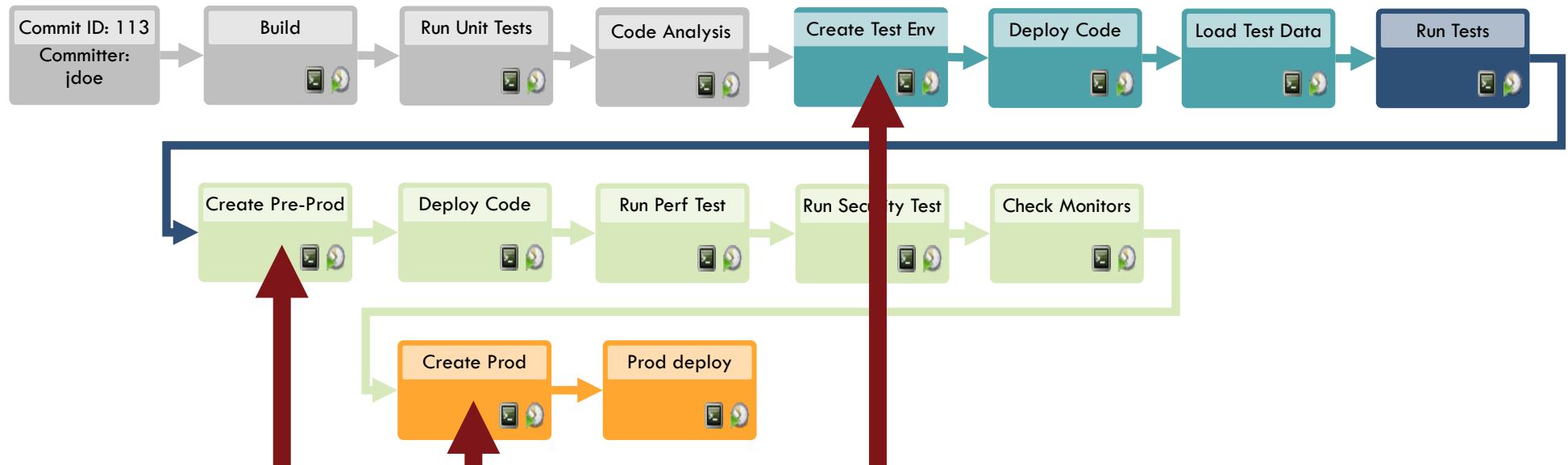


Automation Focus

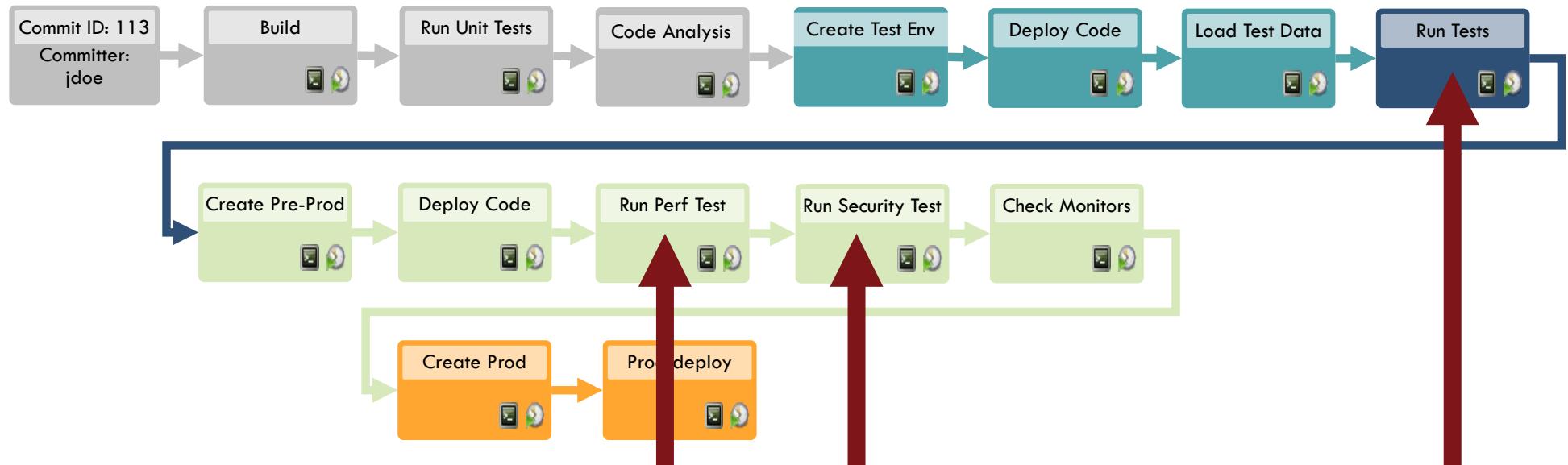


Features are being pushed to those supporting production in ever increasing numbers

Automation Focus

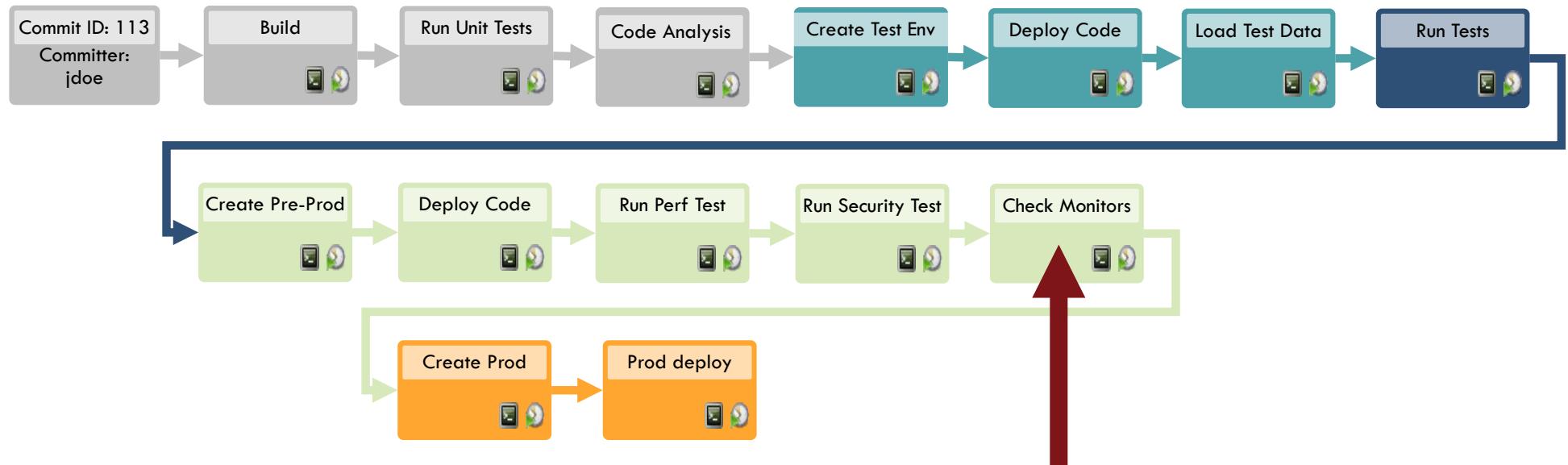


Automation Focus



Testing steps introduce false confidence as production is always unique

Automation Focus

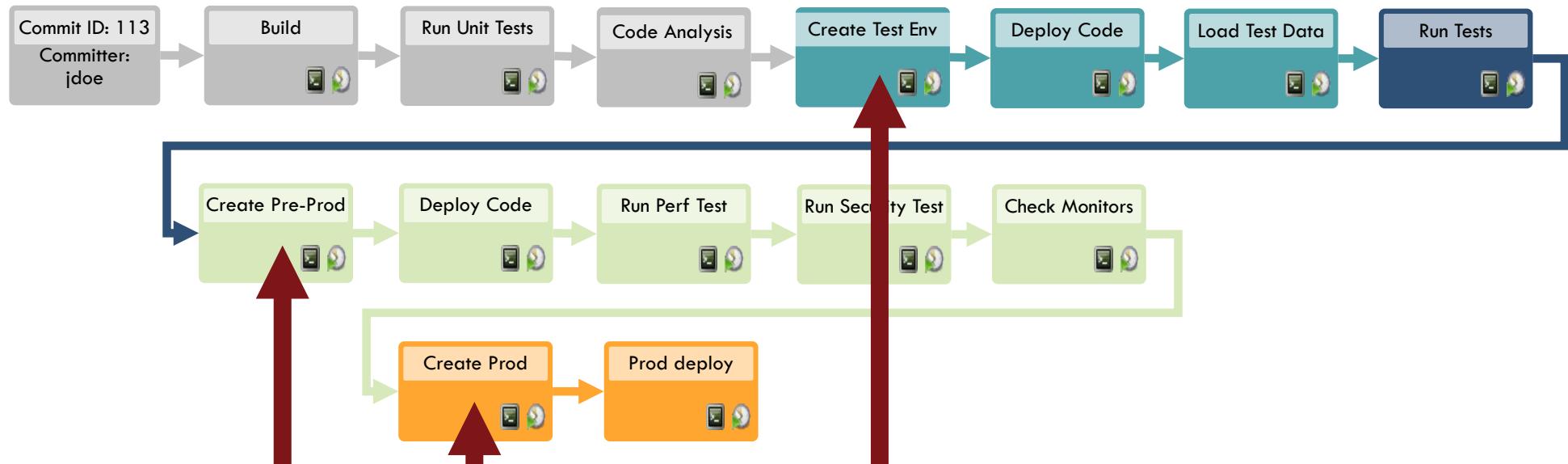


Monitoring and alerting is focused on things that are known to go wrong

A Change in Focus: SRE-Led Service Automation

Automation Focus: In SRE-Led Service Automation

Automation effort is “Ops” led (“shifting left”), to ensure reliability engineering priorities



Environments must be provisioned as Infrastructure- (and Configuration-) as-Code

SRE-Led Service Automation

All code can be rebuilt from a code repository e.g. GitLab, Azure DevOps, Bitbucket.

1. Environments provisioned using Infrastructure/Config as Code
2. Automated functional and non-functional tests in production
3. Versioned (& signed) artefacts to deploy system components
4. Instrumentation in place to make the service externally viewable
5. Future growth envelope outlined
6. Clear anti-fragility strategy

	Components	Example Tools
1. Environments provisioned using Infrastructure/Config as Code	Infrastructure as Code (IaC)	Servers, networks, storage
2. Automated functional and non-functional tests in production	Configuration as Code (CaC)	Software, dependencies, containers



SRE-Led Service Automation

All code can be rebuilt from a code repository e.g. GitHub, GitLab, Bitbucket.

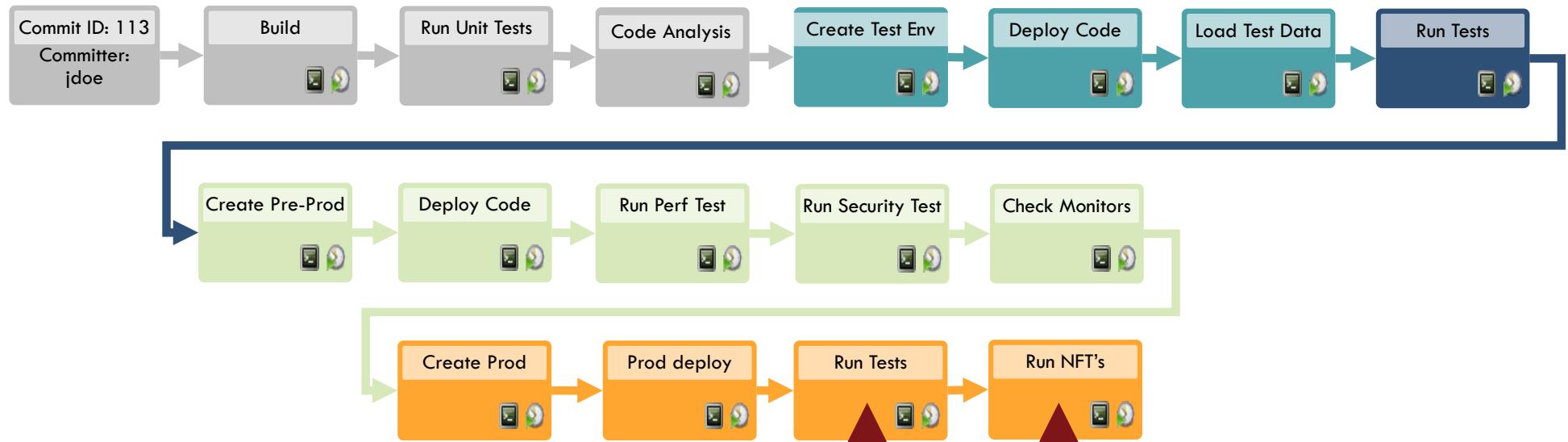
	Environments provisioned using Infrastructure/Config as Code	Infrastructure as Code (IaC)	Consistent, repeatable, production ready environments	Example Tools
1.	Environments provisioned using Infrastructure/Config as Code	Infrastructure as Code (IaC)	Consistent, repeatable, production ready environments	Terraform, AWS CloudFormation, Azure Resource Manager
2.	Automated functional and non-functional tests in production			Puppet, Chef, Ansible, Saltstack, Docker, GCP Deployment Manager
3.	Versioned (& signed) artefacts to deploy system components		Software, dependencies, containers	
4.	Instrumentation in place to view the service externally via API			
5.	Future growth envelope			
6.	Clear anti-fragility			

© DevOps Institute unless otherwise stated

Module 5: SRE Tools & Automation

143

SRE-Led Service Automation



Automated functional and non-functional tests in production

SRE-Led Service Automation

Environment progression includes prod – Dev, Test, Pre-Prod, Prod (inc. “hidden live”)

1. Environments provisioned using Infrastructure/Config as Code
2. **Automated functional and non-functional tests in production**
3. Versioned (& signed) artefacts to deploy system components
4. Instrumentation in place to make the service externally viewable
5. Future growth envelope outlined
6. Clear anti-fragility strategy

	Tests	Example Tools
Extend build pipeline	Automated functional	Selenium, Cucumber, Jasmine, Mocha, Zephyr, Mockito
Extend test pipeline	Automated non-functional	JMeter, Sonatype Nexus Lifecycle, SoapUI, WhiteSource, Veracode, Nagios



Test history is recorded in the pipeline logs

SRE-Led Service Automation

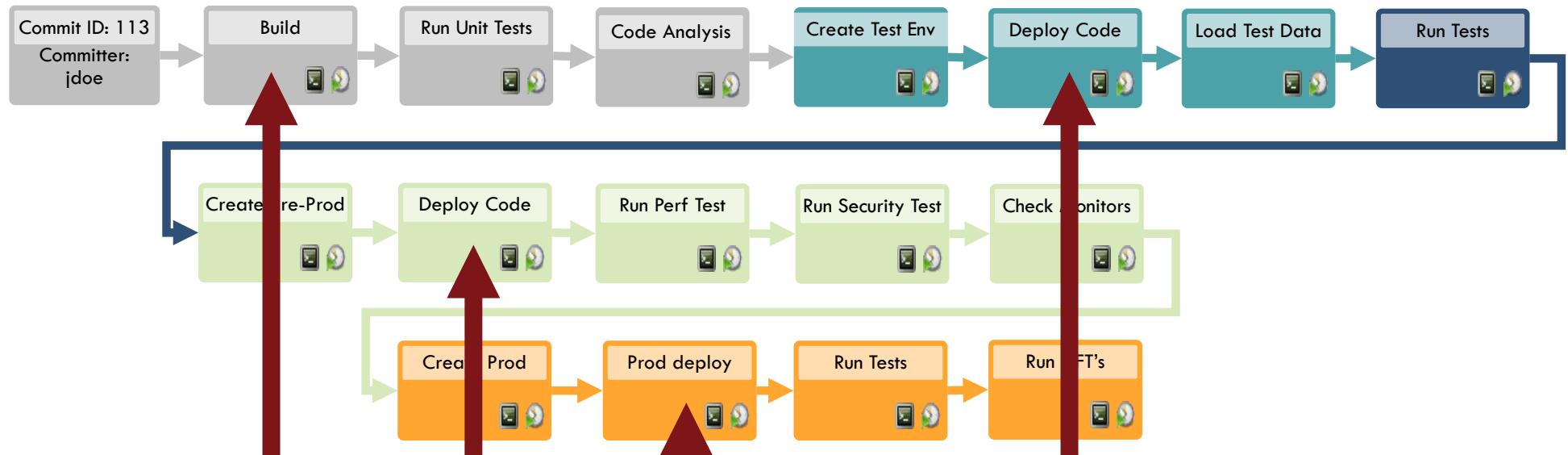
Environment progression includes prod – Dev, Test, P (inc. “hidden live”)

1. Environments provisioned using Infrastructure/Config as Code
2. **Automated functional and non-functional tests in production**
3. Versioned (& signed) artefacts + deploy system components
4. Instrumentation in place to the service externally view
5. Future growth envelope
6. Clear anti-fragility

Example Tools
Selenium, Cucumber, Jasmine, Mocha, Zephyr, Mockito
JMeter, Sonatype Nexus Lifecycle, SoapUI, WhiteSource, Veracode, Nagios
     
  

Test history is recorded in the pipeline logs

SRE-Led Service Automation



Versioned (& signed) artefacts to deploy system components

SRE-Led Service Automation

All service components, libraries and dependencies (or containers) stored in an artifact repository

1. Environments provisioned using Infrastructure/Config as Code
2. Automated functional and non-functional tests in production
3. **Versioned (& signed) artifacts to deploy system components**
4. Instrumentation in place to make the service externally viewable
5. Future growth envelope outlined
6. Clear anti-fragility strategy

Artifacts	How	Example Tools
Digitalized	With semantic versioning x.y.z	Nexus, Artifactory
Digitalized	For security and auditability	Nexus, Artifactory



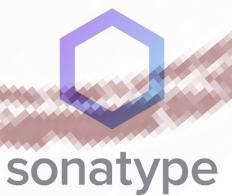
SRE-Led Service Automation

All service components, libraries and dependencies (e.g. code, configuration, databases) are stored in a artifact repository

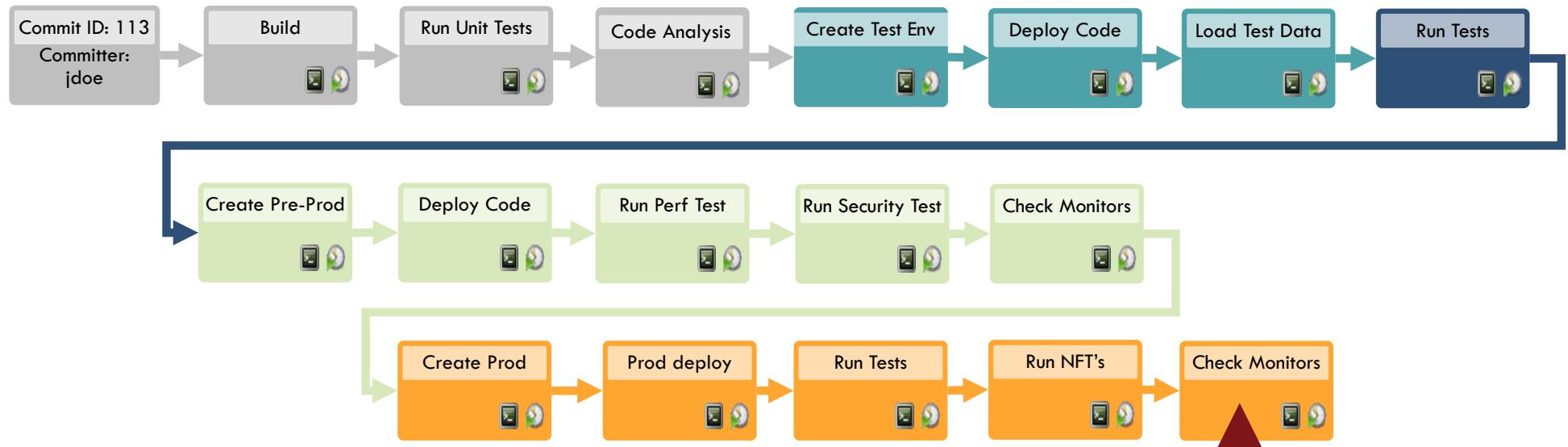
	Artifacts	Dependencies	Example Tools
1. Environments provisioned using Infrastructure/Config as Code	Digital assets (e.g. JAR, WAR, ZIP)	With semantic versioning (e.g. x.y.z)	Nexus, Artifactory
2. Automated functional and non-functional tests in production	Automated build artifacts	For security and auditability	Nexus, Artifactory
3. Versioned (& signed) artifacts deployed system components	Digitally signed		
4. Instrumentation in place to monitor the service externally			
5. Future growth enabled by reuse			
6. Clear anti-fragility strategies			

Promotion of change automated (not manual)

- Reduction in dependency errors
- Easier to determine security vulnerabilities



SRE-Led Service Automation



Instrumentation in place to make the service externally observable

SRE-Led Service Automation

Alignment with SLAs, SLOs, SLIs and telemetry everywhere

	Consider	What	Example Tools
1. Environments provisioned using Infrastructure/Config as Code	Service Level Indicators	Are understood and published	OpsGenie
2. Automated functional and non-functional tests in production	Instrumentation	Provides additional data and analytics	Nagios, Dynatrace, AppDynamics, Prometheus
3. Versioned (& signed) artifacts to deploy system components	Log files	Aggregated and ready for access	Splunk, LogStash
4. Instrumentation in place to make the service externally viewable			
5. Future growth envelope outlined			
6. Clear anti-fragility strategy			

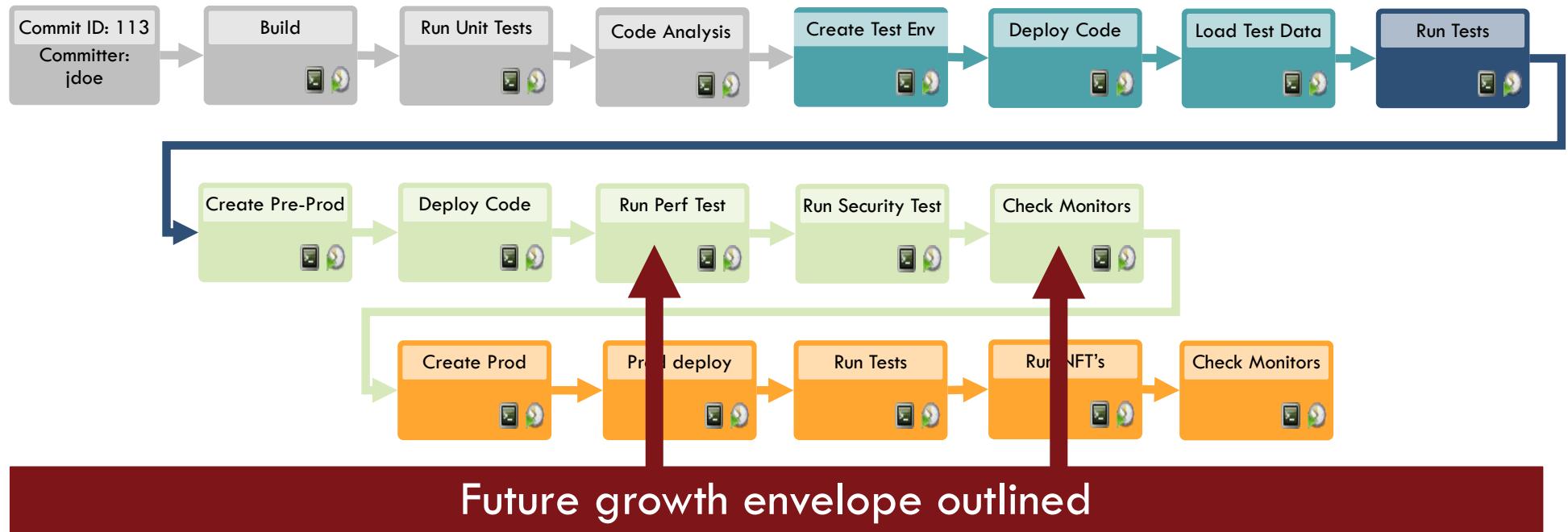


SRE-Led Service Automation

Alignment with SLAs, SLOs, SLIs and TCO		Consider	Example Tools
1. Environments provisioned using Infrastructure/Config as Code	Service Indicators	understood and published	OpsGenie
2. Automated functional and non-functional tests in production	Assists with protective monitoring	Provides additional data and analytics	Nagios, Dynatrace, AppDynamics, Prometheus
3. Versioned (& signed) artifacts deployed system components	Reduces mean time to fix by granting developers read only access to logs	Aggregated and ready for access	Splunk, Logstash
4. Instrumentation in place for the service externally	Security and audit events centralised		
5. Future growth environments	Log files		
6. Clear anti-fragility			



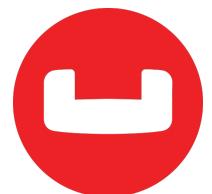
SRE-Led Service Automation



SRE-Led Service Automation

Current and future scale estimates

	Consider	What	Example Tools
1. Environments provisioned using Infrastructure/Config as Code	Autoscaling	In place	Amazon Cloud Auto Scaling, Kubernetes Pod Scaling
2. Automated functional and non-functional tests in production	Administrative activities	Automated	Custom built tooling Cloud API's
3. Versioned (& signed) artifacts to deploy system components	Databases	Scalable	Amazon Cloud RDS, NoSQL-type databases like MongoDB, Couchbase
5. Future growth envelope outlined			
6. Clear anti-fragility strategy			



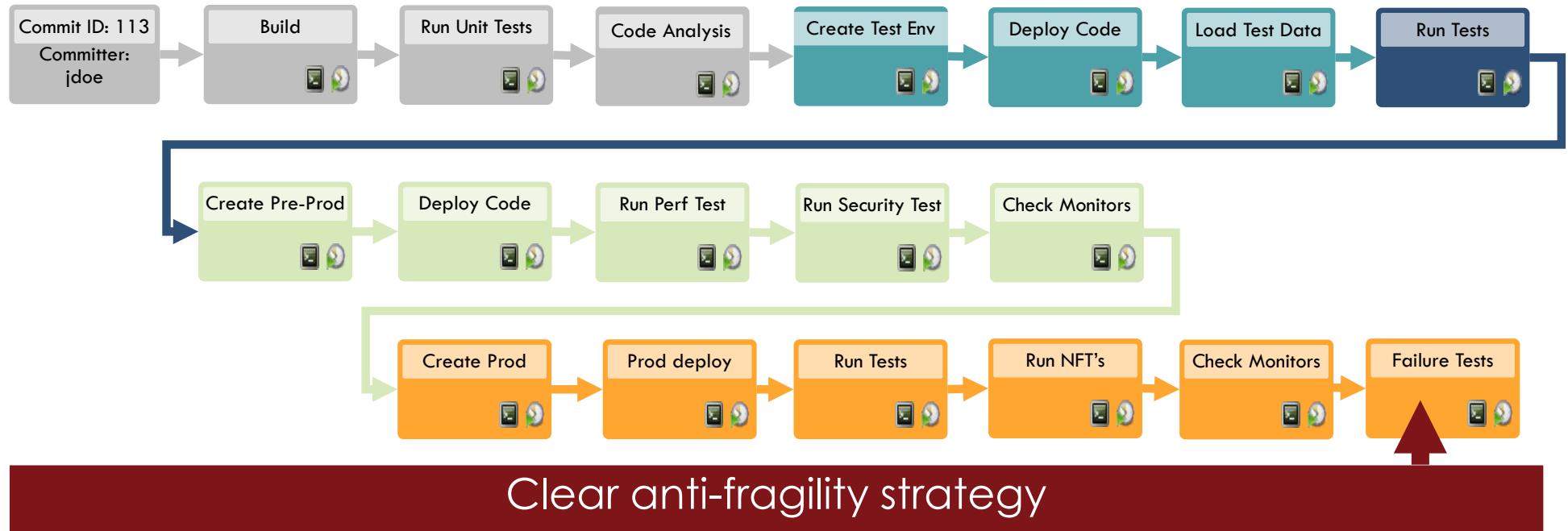
SRE-Led Service Automation

Current and future service delivery		Considerations	Example Tools
What	How		
1. Environments provisioned using Infrastructure/Config as Code	In place	Automated	Amazon Cloud Auto Scaling, Kubernetes Pod Scaling
2. Automated functional and non-functional tests in production	Automated	Scalable	Custom built tooling, Cloud API's
3. Versioned (& signed) artifacts deployed system components	Databases	Scalable	Amazon Cloud RDS, NoSQL-type databases like MongoDB, Couchbase
4. Instrumentation in place to monitor the service externally	Scalable	Scalable	
5. Future growth enabled by automation	Scalable	Scalable	
6. Clear anti-fragility strategy	Scalable	Scalable	

Toil is minimized upfront

- Reduced service Total Cost of Ownership (TCO)
- Reduced rework
- Reduced administrative overhead
- Scalability

SRE-Led Service Automation



SRE-Led Service Automation

Current and future scale estimates

1. Environments provisioned using Infrastructure/Config as Code
2. Automated functional and non-functional tests in production
3. Versioned (& signed) artifacts to deploy system components
4. Instrumentation in place to make the service externally viewable
5. Future growth envelope outlined
6. **Clear anti-fragility strategy**

Consider	What	Example Tools
Disaster Recovery (DR)	Tests complete	Fire drills
Chaos engineering	Practiced	Chaos Monkey
On call mechanisms	In place	PagerDuty, VictorOps, Squadcast



SRE-Led Service Automation

Current and future scale enabled by automation

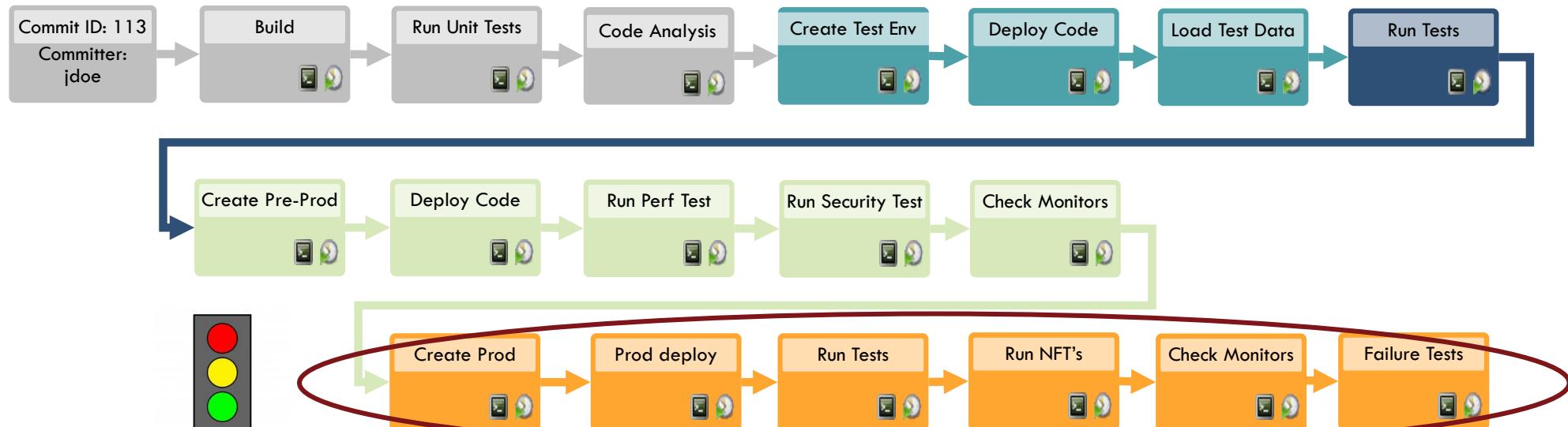
1. Environments provisioned using Infrastructure/Config as Code
2. Automated functional and non-functional tests in production
3. Versioned (& signed) artifacts to deploy system components
4. Instrumentation in place to monitor the service externally visible
5. Future growth envelope identified
6. **Clear anti-fragility**

Consider	Example Tools
Disaster Recovery (DR)	Fire drills
Chaos Engineering	Chaos Monkey
Practiced	PagerDuty, VictorOps, Squadcast
In place	

- Availability and integrity risks to system are appropriately mitigated
- Mitigations evidenced and tested reducing risk



SRE-Led Service Automation



Overall there is more focus on prod – DevOps gains “wisdom of production”, SRE can say “No”

SRE Automation Is Not Just Service Automation



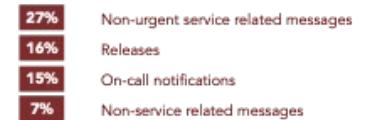
“The team supporting the platform were inundated with toil to the point where they could do little else.”

Mark Rendell,
Accenture

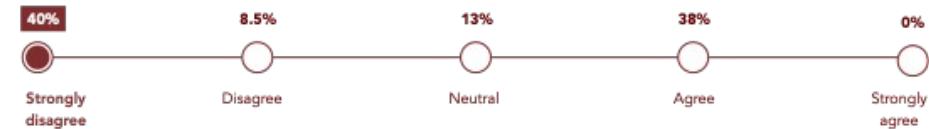
© DevOps Institute unless otherwise stated

What is your top source of toil?

30%
30% of respondents
said maintenance tasks



We have used automation to reduce toil



Module 5: SRE Tools & Automation

160

Hierarchy of Automation Types

Hierarchy of Automation Types

Systems that require no intervention

Internally maintained system-specific automation

Externally maintained generic automation

Externally maintained system-specific automation

None

The database notices problems, and automatically fails over without human intervention.

The database ships with its own failover script that is used if there is a problem

The SRE adds database support to a "generic failover" script that everyone uses if there is a problem

An SRE has a failover script in his or her home directory that is used if there is a problem

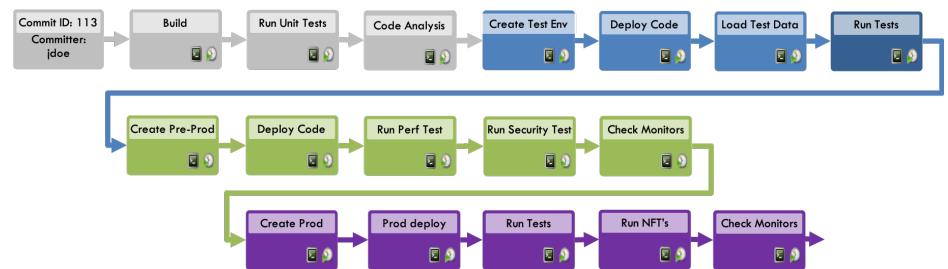
Database master is failed over manually if there is a problem

Secure Automation

Secure Automation

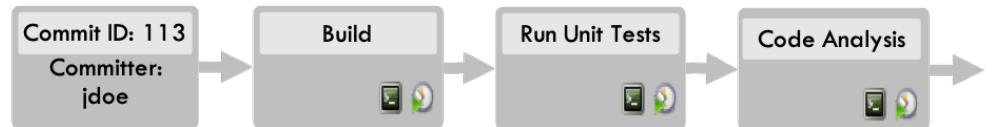
Automation removes the chance of human error (or willful sabotage) and provides security opportunities

- We can secure automated steps in the pipeline – we cannot provenly secure manual steps
- Artifacts generated and used by the pipeline can be validated and checked for compliance
- DevSecOps introduced security into the build-test-deploy lifecycle
- SRE places extra emphasis on security of production



Secure Build

- Application, infrastructure and configuration code changes run through code analysis tools that check for security issues
- Digitally signing build artifacts avoids possibility of “fake” code
- Secure coding practices widely published and embraced
- Secure code repositories with access control
- Coding done in the open ('open source') with community feedback

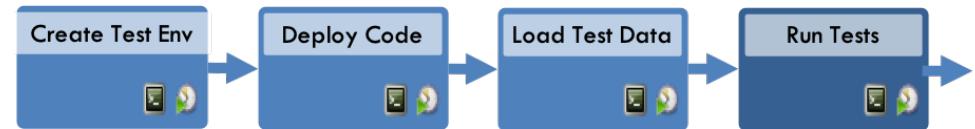


“Having proper configuration management does play a huge role in compliance.”

Will Gregorian,
Director of Security Technical Operations
Omada Health

Secure Test

- When infrastructure or configuration is changed then test environments do not mutate, they are "immutable" and are-created, guaranteeing compliance with code repository
- We deploy the artifacts securely built by our engineers to our test environments
- Test data and test scenarios built to test security



"The only system which is truly secure is one which is switched off and unplugged. Even then, I wouldn't stake my life on it."

Eugene Howard Spafford,
Professor of Computer Science – Computer
Security
Purdue University

Secure Staging

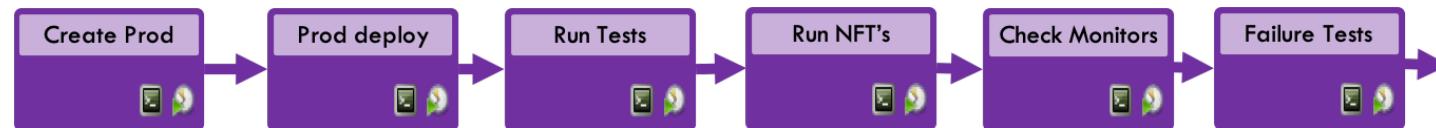
- Staging environments are also immutable
- Same artifact deployed to staging/pre-prod environment
- Production-like data introduced into staging introduces data security considerations e.g. GDPR, PCI
- Dedicated security scanning will try and uncover security vulnerabilities
- Dependencies and integration to other services may introduce vulnerabilities or proxy security requirements



"I'm more and more convinced that staging environments are like mocks - at best a pale imitation of the genuine article and potentially the worst form of confirmation bias."

Cindy Sridharan
Author – Scaling Microservices
O'Reilly

Secure Production



- Production environments are also immutable
- Same artifact deployed to production environment
- Production data requires data security compliance e.g. GDPR, PCI, SOX
- Dedicated security scanning will try and uncover security vulnerabilities
- Regulatory compliance needs to be evidenced
- Failure testing can help with audit compliance

"Having proper configuration management does play a huge role in compliance."

Will Gregorian,
Director of Security Technical
Operations
Omada Health

Automation Tools

Automation Tools

- Any discussion around tools quickly turns into a "favorite tech" talk
- Tools are constantly changing
- Organizations will have bias towards certain types of tools, from "open source" to "big IT" vendors
- Autonomy to use the most appropriate tools to do a job is often best delegated to those with the most knowledge of the job
- Engineers will be more productive with the tools they are familiar

You wouldn't hire a plumber and give them a joiners tools, would you?

CASE STORY: Standard Chartered

"We established some fundamental SRE principles that are being applied across the bank's global footprint, which include 'everything as code', 'everything via API's', 'one-pipeline', 'self-test and self-heal' as well as 'zero-touch automate and orchestrate'. We really now seeing the benefits of the organisation adopting these principles."



Shaun Norris
Global Head, Cloud
Standard Chartered

"We gave the SRE's an 'Andon Cord' so they can stop the line at any time."

Benefits

- 28 person-years of effort saved
- 13,000+ manual reviews of production operations activities avoided
- Time to Repair reduced by 25 minutes on average
- 200 self-inflicted operations incidents avoided

How Much Automation Do You Have?

	Manage	Plan	Create	Verify	Package	Secure	Release	Configure	Monitor	Defend
Audit Management	Issue Tracking	Source Code Management	Continuous Integration (CI)	Package Registry	SAST	Continuous Delivery (CD)	Auto DevOps	Metrics	RASP	
Authentication and Authorization	Kanban Boards	Code Review	Code Quality	Container Registry	DAST	Release Orchestration	Kubernetes Configuration	Logging	WAF	
DevOps Score	Time Tracking	Design Management	Performance Testing	Dependency Proxy	Secret Detection	Pages	ChatOps	Tracing	Threat Detection	
Value Stream Management	Agile Portfolio Management	Wiki	Usability Testing		Dependency Scanning	Review apps	Runbooks	Cluster Monitoring	UEBA	
	Service Desk	Web IDE			Container Scanning	Incremental Rollout	Serverless	Error Tracking	Vulnerability Management	
		Snippets			License Compliance	Feature Flags	Infrastructure as Code	Incident Management	DLP	
					Vulnerability Database				Storage Security	
	Requirements Management			Helm Chart Registry		Release Governance	Chaos Engineering	Synthetic Monitoring		
	Quality Management		System Testing	Dependency Firewall	IAST	Secrets Management	Cluster Cost Optimization	Status Page	Container Network Security	
					Fuzzing					

EXERCISE

How much automation do you have?

Manage (1)

- Audit Management
 - The use of automated tools to ensure products and services are auditable, including keeping audit logs of build, test and deploy activities, auditing configurations and users, as well as log files from production operations.
- Authentication & Authorization
 - Mechanisms for ensuring appropriate access to products, services and tools. For example user and password management and two-factor authentication. Cloud providers use their own tools such as AWS IAM (Identity and Access Management) alongside cloud users tools.

Manage (2)

- DevOps Score
 - A metric showing DevOps adoption across an organization and the corresponding impact on delivery velocity.
- Value Stream Management
 - The ability to visualize the flow of value delivery through the DevOps lifecycle. Gitlab CI and the Jenkins extension (from Cloud Bees) DevOptics can provide this visualization.

Plan (1)

- Issue Tracking
 - Tools like Jira, Trello, CA's Agile Central and VersionOne can be used to capture incidents or backlogs of work
- Kanban Boards
 - On the back of issue tracking, the same tools can represent delivery flow through Scrum and Kanban workflow boards
- Time Tracking
 - Similarly, issue tracking tools also allow for time to be tracked, either against individual issues or other work or project types.
- Agile Portfolio Management
 - Involves evaluating in-flight projects and proposed future initiatives to shape and govern the ongoing investment in projects and discretionary work. CA's Agile Central and VersionOne are examples.

Plan (2)

- Service Desk
 - Service Now is a well used platform for managing the lifecycle of services as well as internal and external stakeholder engagement.
- Requirements Management
 - Tools than handle requirements definition, traceability, hierarchies & dependency. Often also handles code requirements and test cases for requirements.
- Quality Management
 - Tools that handle test case planning, test execution, defect tracking (often into backlogs), severity and priority analysis. CA's Agile Central

Create (1)

- Source Code Management
 - Tools to securely store source code and make it available in a scalable, multi-user environment. Git and SVN are popular examples.
- Code Review
 - The ability to perform peer code-reviews to check quality can be enforced through tools like Gerrit, TFS (Team Foundation Service), Crucible and Gitlab.
- Wiki
 - Knowledge sharing can be enabled by using tools like Confluence which create a rich Wiki of content

Create (2)

- Web IDE
 - Tools that have a web client integrated development environment. Enables developer productivity without having to use a local development tool. Gitlab
- Snippets
 - Stored and shared code snippets to allow collaboration around specific pieces of code. Also allows code snippets to be used in other code-bases. BitBucket and GitLab allow this.

Verify (1)

- Continuous Integration
 - “Refers to integrating, building, and testing code within the development environment” – Martin Fowler, Chief Scientist, ThoughtWorks.
- Code Quality
 - Also referred to as code analysis, Sonar and Checkmarks are examples of tools that automatically check the seven main dimensions of code quality – comments, architecture, duplication, unit test coverage, complexity, potential defects, language rules.

Verify (2)

- Performance Testing
 - Performance testing is the process of determining the speed, responsiveness and stability of a computer, network, software program or device under a workload. Tools like Gatling can performance test services.
- Usability Testing
 - Usability testing is a way to see how easy to use something is by testing it with real users. Tools can be used to track how a user works with a service e.g. with scroll recording, eye checking, mouse tracking. Crazy Egg, Optimizely are examples.

Package (1)

- Package Registry
 - A repository for software packages, artifacts and their corresponding metadata. Can store files produced by an organization itself or for third party binaries. Artifactory and Nexus are amongst the most popular.
- Container Registry
 - Secure and private registry for Container images. Typically allowing for easy upload and download of images from the build tools. Docker Hub, Artifactory, Nexus,
- Dependency Proxy
 - For many organizations, it is desirable to have a local proxy for frequently used upstream images/packages. In the case of CI/CD, the proxy is responsible for receiving a request and returning the upstream image from a registry, acting as a pull-through cache.

Package (2)

- Helm Chart Registry
 - Helm charts are what describe related Kubernetes resources. Artifactory and Codefresh support a registry for maintaining master records of Helm Charts.
- Dependency Firewall
 - Many projects depend on packages that may come from unknown or unverified providers, introducing potential security vulnerabilities. There are tools to scan dependencies but that is after they are downloaded. These tools prevent those vulnerabilities from being downloaded to begin with.

183

Secure (1)

- SAST
 - Static Application Security Testing test applications from the “inside out” by looking at source code, byte code or binaries. Gitlab, CA’s Veracode,
- DAST
 - Dynamic Application Security Testing tests applications from the “outside in” to detect security vulnerabilities. Gitlab, CA’s Veracode
- IAST
 - Interactive Application Security Testing combines SAST and DAST approaches but involves application tests changing in “real time” based on information fed back from SAST and DAST, creating new test cases on the fly. Synopsis, Acunetix, Parasoft and Quotium are solutions evolving in this direction.
- Secret Detection
 - Secret Detection aims to prevent that sensitive information, like passwords, authentication tokens, and private keys are unintentionally leaked as part of the repository content.

Secure (2)

- Dependency Scanning
 - Used to automatically find security vulnerabilities in your dependencies while you are developing and testing your applications. Synopsis, Gemnasium, Retire.js and bundler-audit are popular tools in this area.
- Container Scanning
 - When building a Container image for your application, tools can run a security scan to ensure it does not have any known vulnerability in the environment where your code is shipped. Blackduck, Synopsis, Synk, Claire and klar are examples.
- License Compliance
 - Tools, such as Blackduck and Synopsis, that check that licenses of your dependencies are compatible with your application, and approve or blacklist them.

Secure (3)

- Vulnerability Database
 - Is aimed at collecting, maintaining, and disseminating information about discovered computer security vulnerabilities. This is then checked as part of the delivery pipeline.
- Fuzzing
 - Fuzzing or fuzz testing is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a service and watching for the results.

Release (1)

- Continuous Delivery
 - “Is a software development discipline where you build software in such a way that the software can be released to production at any time” – Martin Fowler, Chief Scientist, ThoughtWorks
- Release Orchestration
 - Typically a deployment pipeline, used to detect any changes that will lead to problems in production. Orchestrating other tools will identify performance, security, or usability issues. Tools like Jenkins and Gitlab CI can “orchestrate” releases.
- Pages
 - For creating supporting web pages automatically as part of a CI/CD pipeline.
- Review Apps
 - Allow code to be committed and launched in real time – environments are spun up to allow developers to review their application. Gitlab CI has this capability.
- Incremental Rollout
 - Incremental rollout means deploying many small, gradual changes to a service instead of a few large changes. Users are incrementally moved across to the new version of the service until eventually all users are moved across. Sometimes referred to by colored environments e.g. Blue/green deployment.

Release (2)

- Canary Deployments
 - Similar to incremental rollout, it is where a small portion of the user base is updated to a new version first. This subset, the canaries, then serve as the proverbial “canary in the coal mine”. If something goes wrong then a release is rolled back and only a small subset of the users are impacted.
- Feature Flags
 - Sometimes called feature toggles, a technique that allows system behavior to change without changing the underlying code, through the use of “flags” to decide which behavior is invoked. A programming practice primarily there are tools such as Launch Darkly which can help with flag management and invocation.
- Release Governance
 - Release Governance is all about the controls and automation (security, compliance, or otherwise) that ensure your releases are managed in an auditable and trackable way, in order to meet the need of the business to understand what is changing. Gitlab CI.
- Secrets Management
 - Secrets management refers to the tools and methods for managing digital authentication credentials (secrets), including passwords, keys, APIs, and tokens for use in applications, services, privileged accounts and other sensitive parts of the IT ecosystem

Configure (1)

- Auto DevOps
 - Auto DevOps brings DevOps best practices to your project by automatically configuring software development lifecycles. It automatically detects, builds, tests, deploys, and monitors applications. Gitlab and AWS Code Pipelines are strong examples.
- ChatOps
 - The ability to execute common DevOps actions directly from chat (build, deploy, test, incident management, rollback ,etc) with the output sent back to a channel.
- Runbooks
 - A collection of procedures necessary for the smooth operation of a service. Previously manual in nature they are now usually automated with tools like Ansible.
- Serverless
 - A code execution paradigm where no underlying infrastructure or dependencies are needed, moreover a piece of code is executed by a service provider (typically cloud) who takes over the creation of the execution environment. Lambda functions in AWS and Azure Functions are examples.

189

Configure (2)

- Auto DevOps
 - Auto DevOps brings DevOps best practices to your project by automatically configuring software development lifecycles. It automatically detects, builds, tests, deploys, and monitors applications. Gitlab and AWS Code Pipelines are strong examples.
- ChatOps
 - The ability to execute common DevOps actions directly from chat (build, deploy, test, incident management, rollback ,etc) with the output sent back to a channel.
- Runbooks
 - A collection of procedures necessary for the smooth operation of a service. Previously manual in nature they are now usually automated with tools like Ansible.
- Serverless
 - A code execution paradigm where no underlying infrastructure or dependencies are needed, moreover a piece of code is executed by a service provider (typically cloud) who takes over the creation of the execution environment. Lambda functions in AWS and Azure Functions are examples.

Monitor (1)

- Metrics
 - Tools that collect and display performance metrics for deployed apps, such as Prometheus.
- Logging
 - The capture, aggregation and storage of all logs associated with system performance including, but not limited to, process calls, events, user data, responses, error and status codes. Logstash and Nagios are popular examples.
- Tracing
 - Tracing provides insight into the performance and health of a deployed application, tracking each function or microservice which handles a given request.
- Cluster Monitoring
 - Tools that let you know the health of your deployment environments running in clusters such as Kubernetes.

Monitor (2)

- Error Tracking
 - Tools to easily discover and show the errors that application may be generating, along with the associated data.
- Incident Management
 - Involves capturing the who, what, when of service incidents and the onward use of this data in ensuring service level objectives are being met.
- Synthetic Monitoring
 - The ability to monitor service behavior by creating scripts to simulate the action or path taken by a customer/end-user and the associated outcome.
- Status Page
 - Self-explanatory, service pages that easily communicate the status of services to customers and users.

192

Defend (1)

- RASP
 - Runtime Application Self Protection (RASP) – tools that actively monitor and block threats in the production environment before they can exploit vulnerabilities.
- WAF
 - Web Application Firewall – tools that examine traffic being sent to an application and can block anything that looks malicious.
- Threat Detection
 - Refers to the ability to detect, report, and support the ability to respond to attacks. Intrusion detection systems and denial-of-service systems allow for some level of threat detection and prevention.
- UEBA
 - User and Entity Behavior Analytics (UEBA) is a machine learning technique to analyze normal and “abnormal” user behavior with the aim of preventing the latter.

Defend (2)

- **Vulnerability Management**
 - Is about ensuring that assets and applications are scanned for vulnerabilities and then the subsequent processes to record, manage, and mitigate those vulnerabilities.
- **DLP**
 - Data Loss Protection – tools that prevent files and content from being removed from within a service environment or organization.
- **Storage Security**
 - a specialty area of security that is concerned with securing data storage systems and ecosystems and the data that resides on these systems.
- **Container Network Security**
 - Used to prove that any app that can be run on a container cluster with any other app can be confident that there is no unintended use of the other app or any unintended network traffic between them.

194

Ironies of Automation

A Comedy in Three Parts



Ironies of Automation: A Comedy in Three Parts with Tanner Lund (Microsoft) (18:32)

195

© DevOps Institute unless otherwise stated

Module 5: SRE Tools & Automation

Module Five Quiz

1	Chaos monkey is used for:	a) Automated incident response b) The development of games c) The accuracy of Google searches d) Anti-fragility testing
2	Which two pattern ensure services are consistent across all environments	a) Configuration as code b) Cloud a code c) Security as code d) Infrastructure as code
3	What does automation <u>not</u> give us?	a) Time savings b) Consistency c) Re-use d) Better meetings
4	What are two benefits of using signed artifacts?	a) Prevent viruses being spread b) Avoids the possibility of fake code being deployed c) Compliance verification d) Enhances user access security
5	What do we need in place to make a service externally observable?	a) Instrumentation b) Container scanning c) Secrets management d) Code analysis

Module Five Quiz Answers

1	Chaos monkey is used for:	a) Automated incident response b) The development of games c) The accuracy of Google searches d) Anti-fragility testing
2	Which two pattern ensure services are consistent across all environments	a) Configuration as code b) Cloud a code c) Security as code d) Infrastructure as code
3	What does automation <u>not</u> give us?	a) Time savings b) Consistency c) Re-use d) Better meetings
4	What are two benefits of using signed artifacts?	a) Prevent viruses being spread b) Avoids the possibility of fake code being deployed c) Compliance verification d) Enhances user access security
5	What do we need in place to make a service externally observable?	a) Instrumentation b) Container scanning c) Secrets management d) Code analysis

Module 6

ANTI-FRAGILITY AND LEARNING FROM FAILURE

© DevOps Institute unless otherwise stated

Module 6: Antifragility & Learning from Failure

- Why learn from failure
- Benefits of anti-fragility
- Shifting the organizational balance

Component	Module 6 Content
Video	Introducing network failures (Indeed.com)
Case Story	Netflix Simian Army
Discussion	Failure is bad – for organizations and individuals
Exercise	What's the difference between outages?

Module 6: Antifragility & Learning from Failure

DISCUSSION Failure is Bad – For Who?

Changing the Reputation of Failure

“There is no such thing as failure. There are only results. It's time to stop beating yourself up and start realizing that everything you do is a success or a learning experience.”

Tony Robbins, author & life coach



Module 6: Antifragility & Learning from Failure

Why Learn from Failure

Why Learning from Failure is Important

- Ask why, 5 times:

- We need to be better at breaking things
- We need to understand how things work together
- So when things break, we know how to fix them
- So we can prevent things from breaking
- To reduce downtime and to ensure we make money*

Statistics

Risk
versus
Benefits

New features
versus
Anti-fragility

Module 6: Antifragility & Learning from Failure

POWER OF
EMAIL
INTERNET
WORLD
NO
SEARCH
203

Firefighting

How often do you end up firefighting (as we call it) in the workplace?

Firefighting real fires usually requires a lot of training:

- 6 – 28 weeks of basic training
- Specialist training (woodland) another 6 - 12 weeks
- Specialist training (city) another 6 – 12 weeks
- Plan for the unexpected in training (what if scenarios?)



Do we do this in tech?

Or Alternatively

Solutions that might fix the problem without breaking anything



Essential

Hoping This Works

O RLY?

@ThePracticalDev

How to actually learn any new programming concept



Essential

Changing Stuff and Seeing What Happens

O RLY?

@ThePracticalDev

The internet will make those bad words go away



Essential

Googling the Error Message

O RLY?

The Practical Developer
@ThePracticalDev

Benefits of Antifragility

Failure Happens, Use It To Your Advantage

“Failure happens, there is no way around it so stop pointing fingers. Embracing failure will help improve MTTD and MTTR metrics. Proactively addressing failure leads to more robust systems.”

Jennifer Petoff, Global Program Manager for SRE Education, Google



Improving Metrics through Antifragility

- MTTD – Mean Time to Detect (Failure/Incidents)
- MTTR – Mean Time to Recover/Repair (Components)
- MTRS – Mean Time to Recover (Service)
- Service Level Objective (SLO)
- Recovery Point Objective (RPO)



Are you measuring these metrics?

Improving Metrics through Antifragility

- MTTD – Mean Time to Detect (Failure/Incidents)
- MTTR – Mean Time to Recover/Repair(Components)
- MTRS – Mean Time to Recover (Service)
- Service Level Objective (SLO)
- Recovery Point Objective (RPO)
- *By introducing failure we optimize our monitoring making it more likely we will detect real incidents*

Improving Metrics through Antifragility

- MTTD – Mean Time to Detect (Failure/Incidents)
- MTTR – Mean Time to Recover/Repair (Components)
- MTRS – Mean Time to Recover (Service)
- Service Level Objective (SLO)
- Recovery Point Objective (RPO)
- *Simulating component failure allows us to create automation to try and auto-recover*
- *We can also build in more resilience to prevent failure of single components*

Improving Metrics through Antifragility

- MTTD – Mean Time to Detect (Failure/Incidents)
- MTTR – Mean Time to Recover/Repair (Components)
- MTRS – Mean Time to Recover (Service)
- Service Level Objective (SLO)
- Recovery Point Objective (RPO)
- Chaos engineering approaches identify key interfaces & dependencies across services pinpointing areas where more resilience may be required

Improving Metrics through Antifragility

- MTTD – Mean Time to Detect (Failure/Incidents)
- MTTR – Mean Time to Recover/Repair (Components)
- MTRS – Mean Time to Recover (Service)
- Service Level Objective (SLO)
- Recovery Point Objective (RPO)
- A *fire drill* where e.g. a database is taken down may result in an SLO being broken
- Caching data in the case of a database outage instead could mean the SLO is met

Improving Metrics through Antifragility

- MTTD – Mean Time to Detect (Failure/Incidents)
- MTTR – Mean Time to Recover/Repair (Components)
- MTRS – Mean Time to Recover (Service)
- Service Level Objective (SLO)
- Recovery Point Objective (RPO)
- *Introducing failure to e.g. a messaging queue may indicate excessive data loss, outside the RPO*
- *More frequent back ups of the queue data may be needed to meet the RPO*

Module 6: Antifragility & Learning from Failure

Shifting the Organizational Balance

Creating a Culture of Learning from Failure

“You’re either a learning organization or you’re losing to somebody who is...”

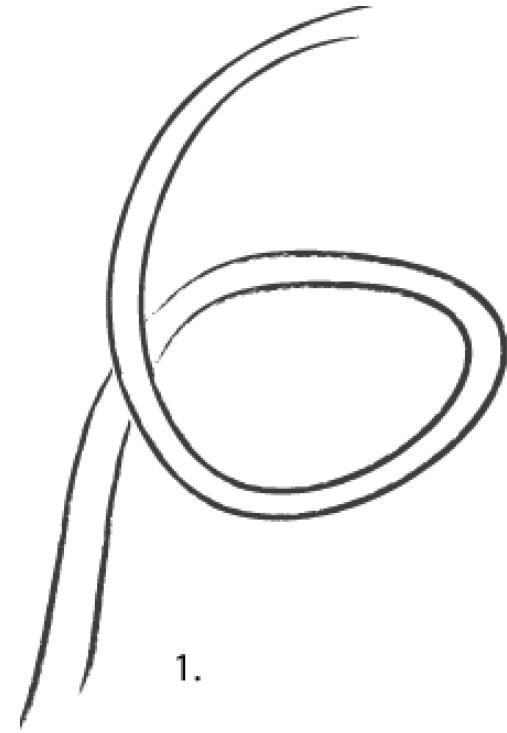
Andrew Shafer quoted in ‘Beyond the Phoenix Project’



Andrew Clay-Shafer is a foundational voice in the DevOps movement

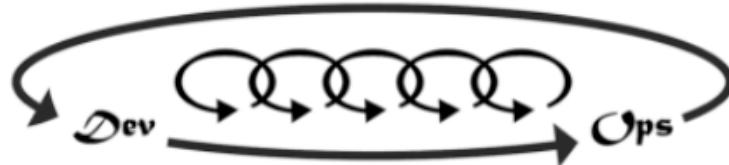
Shifting the balance

- **1st Play – enable the 3rd of the “three ways”**
- 2nd Play – Benchmark v “Westrum Model”
- 3rd Play – Introduce “Fire Drills”
- 4th Play - “Chaos engineering” next steps



“The only real mistake is the one from which we learn nothing.” - Henry Ford

The Third Way: Continual Experimentation and Learning



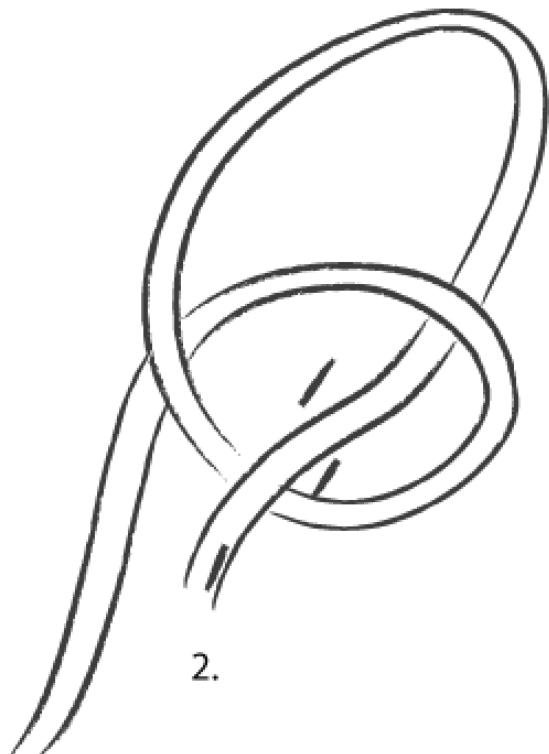
The Third Way encourages a culture that fosters two things:

1. Continual experimentation, taking risks and **learning from failure**
2. Understanding that repetition and practice is the prerequisite to mastery.

- Allocate time for the improvement of daily work
- Create rituals that reward the team for taking risks
- Introduce faults into the system to increase resilience
- Plan time for safe experimentation and innovation (hackathons)

Shifting the balance

- 1st Play – enable the 3rd of the “three ways”
- **2nd Play – Benchmark v “Westrum Model”**
- 3rd Play – Introduce “Fire Drills”
- 4th Play - “Chaos engineering” next steps



"We make mistakes, and we get back up ... " Jack Dorsey, Twitter

Culture and the Flow of Information

Pathological (Power-oriented)	Bureaucratic (Rule-oriented)	Generative (Performance-oriented)
Information is hidden	Information may be ignored	Information is actively sought
Messengers are ‘shot’	Messengers are isolated	Messengers are trained
Responsibilities are shirked	Responsibility is compartmentalized	Responsibilities are shared
Bridging is discouraged	Bridging is allowed but discouraged	Bridging is rewarded
Failure is covered up	Organization is just and merciful	Failure causes enquiry
Novelty is crushed	Novelty creates problems	Novelty is implemented

Source: Westrum, *A Typology of Organizational Cultures*

High-trust organizations encourage good information flow, cross-functional collaboration, shared responsibilities, learning from failures and new ideas.

Shifting the balance

- 1st Play – enable the 3rd of the “three ways”
- 2nd Play – Benchmark v “Westrum Model”
- **3rd Play – Introduce “fire drills”**
- 4th Play – “Chaos Engineering” next steps



"It's fine to celebrate success, but it is more important to heed the lessons of failure." Bill Gates

Introduce Fire Drills

- Fire drills build on the concepts of business continuity planning (BCP) and disaster recovery (DR) which have been around for decades
- Need to ensure a business can continue to operate during unforeseen events or failures, such as natural disasters or emergencies
- This is quite often an audit requirement
- For a lot of organisations this is an annual data centre failover test

Fire drills are a good first step towards chaos engineering

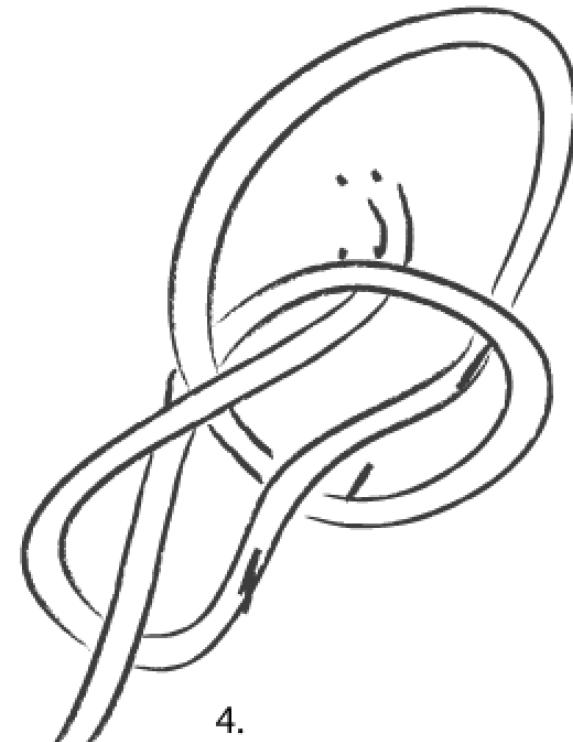
Introduce Fire Drills

“Fire Drills” can go beyond technology:

1. Loss of facility (datacentre) or region (cloud)
2. Loss of technology (e.g. database)
3. Loss of resources (e.g. key person)
4. Loss of critical third-party vendors

Shifting the balance

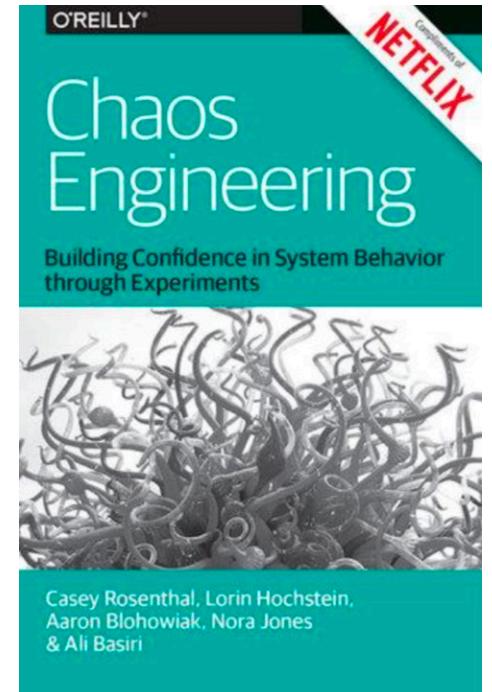
- 1st Play – enable the 3rd of the “three ways”
- 2nd Play – Benchmark v “Westrum Model”
- 3rd Play – Introduce “Fire Drills”
- **4th Play - Chaos engineering next steps**



“You shouldn’t be afraid of failure” Niklas Zennström, Skype

What is Chaos Engineering

- Chaos engineering is the discipline of experimenting on a software system in production in order to build confidence in the system's capability to withstand turbulent and unexpected conditions
- Most famous example is the Simian Army from Netflix



<https://www.oreilly.com/library/view/chaos-engineering/9781491988459/>

CASE STORY: Netflix

"The Simian Army is a suite of tools developed by Netflix to test the reliability, security, or resiliency of its Amazon Web Services* infrastructure and includes Chaos Monkey, which disables production instances at random, Latency Gorilla, which simulates network delays and Chaos Gorilla, which brings down whole Amazon datacenters (AZ's), amongst others."



Greg Orzell
Software Engineer
Netflix

© DevOps Institute unless otherwise stated

"So next time an instance fails at 3 am on a Sunday, we won't even notice."

Benefits

- Minimize impact on customer experience and hence revenue
- Automated "self healing" improves staff morale due to less "on call" incidents

Chaos Engineering Next Steps

1. Segregate the system into key components
2. Test the system without key components being available
3. Break the system (in non-prod environments first)
4. Introduce failure of key components in prod
5. Introduce Database failure in prod
6. Introduce a total system failure in prod

Chaos Engineering Next Steps

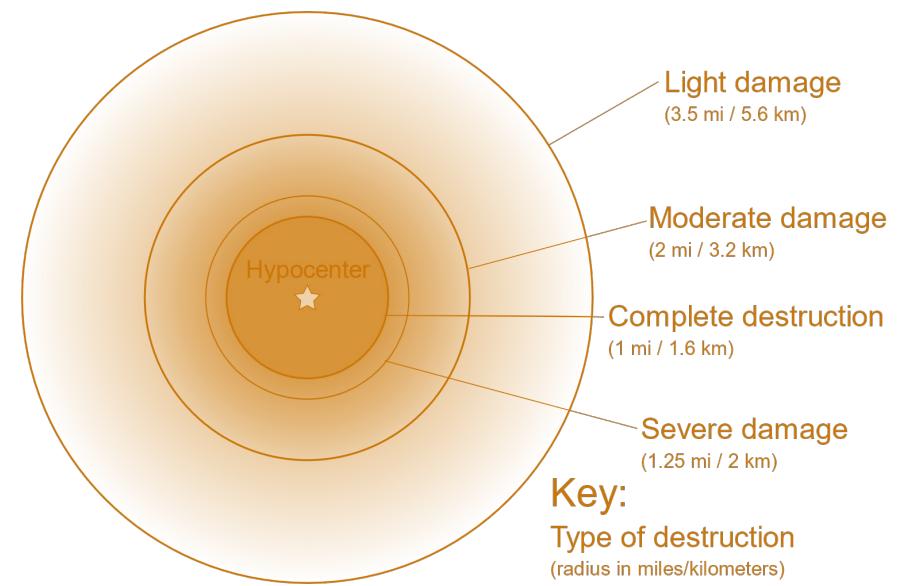
1. Look at “holistic” logging (e.g. what keeps full service up?)
2. Identify dependencies
3. Improve by error handling and recovery (manual-to-automated)
4. Learn from “real” failures

Chaos Engineering Next Steps

Chaos engineering also helps us to “minimize the blast radius”

Any outage should effect as little of the ecosystem as possible

Failure testing shows the current span of this radius



Chaos Engineering Next Steps

What automated recovery looks like:

1. Create immutable infrastructure as code
2. Cover all system state and functional code with automated tests
3. Deploy holistic logging and monitoring systems – make services “observable”
4. Implement smart alerting, triggers and prescriptive analytics
5. Create the self healing infrastructure/applications where **most appropriate**
6. Test! Then Test again!

Do not allow SSH / RDP in production (unless there is a real new problem)

Automation Can Help

Tools like Kubernetes and AWS Auto-Scaling can:

1. Detect impaired instances of servers or containers and destroy/replace
2. Maintain infrastructure at a (code) defined level
3. Automatically scale infrastructure/applications up and down based on demand
4. Execute maintenance commands “on the fly” e.g. to re-index databases when queries are running slow
5. Integrate with monitoring services

Look to leverage the capabilities of tools and platforms before building

I am putting sloths on the map

Destructive testing in a complex operational environment

Preetha Appan
Software Engineer

indeed



‘Sloth, a Tool for Inducing Network Failures’
with Preetha Appan Indeed.com (04:45)



231

Module 6: Antifragility & Learning from Failure

EXERCISE

What's the difference between outages?

Module Six Quiz

1	Who created the Chaos Monkey service?	a) Amazon Prime b) BBC iplayer c) Netflix d) Disney
2	Which metric is most concerned with early failure detection?	a) MTTR b) MTTD c) MTTA d) MTTF
3	Which approach can help understand the impact of a key person dependence?	a) Squad health check b) Value stream map c) Chaos Monkey d) Fire Drill
4	If service failures are usually covered up by an organization then that organization is what?	a) Hedonistic b) Pessimistic c) Neurologic d) Pathologic
5	If a service recovery point objective (RPO) is ten minutes then how frequently must the service be backed up	a) At less than ten minute intervals b) Every ten minutes c) At least daily d) Backups do not need to be taken

Module Six Quiz Answers

1	Who created the Chaos Monkey service?	a) Amazon Prime b) BBC iplayer c) Netflix d) Disney
2	Which metric is most concerned with early failure detection?	a) MTTR b) MTTD c) MTTA d) MTTF
3	Which approach can help understand the impact of a key person dependence?	a) Squad health check b) Value stream map c) Chaos Monkey d) Fire Drill
4	If service failures are usually covered up by an organization then that organization is what?	a) Hedonistic b) Pessimistic c) Neurologic d) Pathologic
5	If a service recovery point objective (RPO) is ten minutes then how frequently must the service be backed up	a) At less than ten minute intervals b) Every ten minutes c) At least daily d) Backups do not need to be taken

Module 7

ORGANIZATIONAL IMPACT OF SRE

© DevOps Institute unless otherwise stated

Module 7: Organizational impact of SRE

- Why organizations embrace SRE
- Patterns for SRE adoption
- SRE Job Description
- Sustainable Incident Response
- Blameless post mortems
- SRE & Scale

Component	Module 7 Content
Video	A history of SRE (Uber)
Case Story	Sage Group & DWP
Discussion	Why do you want to adopt SRE? Who in your organization currently provides SRE?
Exercise	Your organizational plan for SRE

Why Organizations Embrace SRE

Increased Service Resilience

Courtesy of downdetector.co.uk

Outage top 10 week 31

Week 30		
+1	1	 YAHOO! MAIL
	2	 Virgin Media
-2	3	 BT
+1	4	 Sky
5		 Outlook
6		 TalkTalk
7		 Instagram
8		 Xbox Live
9		 EE
10		 Vodafone

- As service usage grows then the volume of users grows
- Downtime is more widely published through social media channels
- Brand reputation can be compromised

<https://downdetector.co.uk/top10/>

Minimize Loss of Revenue

According to Gartner....

- The average cost of service downtime is \$5,600 per minute. Because there are so many differences in how businesses operate, downtime, at the low end, can be as much as \$140,000 per hour, \$300,000 per hour on average, and as much as \$540,000 per hour at the higher end



<https://www.the20.com/blog/the-cost-of-it-downtime/>

Because Its Cool

How often do you hear this:

- “I was just at a conference and...”
- “I read this online and...”
- “Such and such an organization are doing....”



Do not change your current approach without a good reason

Module 7: Organizational impact of SRE

DISCUSSION 1

Why do you want to adopt SRE?

Patterns for SRE Adoption

DISCUSSION 2

Who in your organization currently provides SRE services or capabilities?

Typical SRE Adoption Steps

- **Consulting**
 - Embedded
 - Platform
 - Slice & Dice
 - Full SRE
- Specialist advice provided by experts
- No “hands-on” delivery involvement
- Ownership of SRE with the service/delivery teams – consultants not “on call”

These steps are not necessarily sequential.....

Typical SRE Adoption Steps

- Consulting
- **Embedded**
- Platform
- Slice & Dice
- Full SRE
- SRE experts embedded in service/delivery teams
- Co-working on SRE activities
- Some movement towards "shared responsibility"
- SRE's take initial "on call" before responsibilities become shared

These steps are not necessarily sequential.....

Typical SRE Adoption Steps

- Consulting
- Embedded
- **Platform**
- Slice & Dice
- Full SRE
- SRE “own” the deployment platform and tooling
- Guardians of production environments (and maybe others) – provide “on call”
- One size fits all approach
- Little or no shared responsibility

These steps are not necessarily sequential.....

Typical SRE Adoption Steps

- Consulting
 - Embedded
 - Platform
 - **Slice & Dice**
 - Full SRE
- *SRE own sections of the service, typically application and infrastructure*
 - *Some shared responsibility although "on-call" responsibility is also sliced*

These steps are not necessarily sequential.....

Typical SRE Adoption Steps

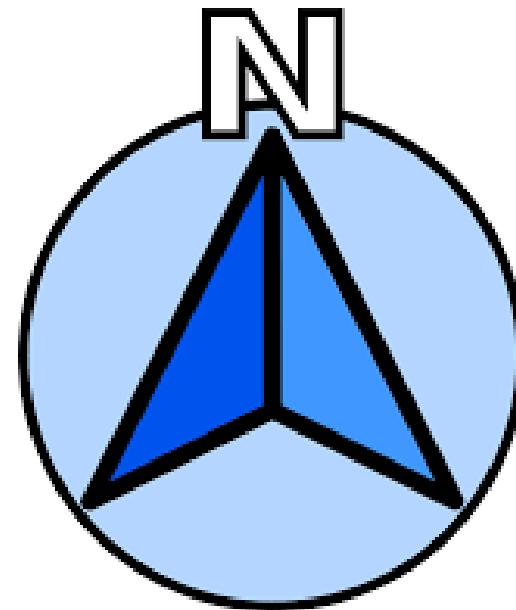
- Consulting
- Embedded
- Platform
- Slice & Dice
- **Full SRE**
- *Full organization embraces SRE*
- *Share responsibility and shared “on call”*
- *Reliability a first class citizen*

These steps are not necessarily sequential.....

True North Vision

Set realistic expectations of SRE:

- Impossible to achieve 100% uptime – what appropriate SLO's would you set?
- Can you afford complete end to end automation?
- Is automatic recovery and restore even possible?



It may not all be achievable, the aim is to be as close as possible to a “true north” vision

Rick Boone

Site Reliability Engineer, Uber



A history of SRE at Uber, with Rick Boone
(06:24)

250

© DevOps Institute unless otherwise stated

Module 7: Organizational impact of SRE

Module 7: Organizational impact of SRE

SRE Job Description

Typical Responsibilities

Software Engineer, Site Reliability Engineering

Share Save

Google Mountain View, CA, USA

Apply

- Engage in and improve the whole lifecycle of services—from inception and design, through deployment, operation and refinement.
- Support services before they go live through activities such as system design consulting, developing software platforms and frameworks, capacity planning and launch reviews.
- Maintain services once they are live by measuring and monitoring availability, latency and overall system health.
- Scale systems sustainably through mechanisms like automation, and evolve systems by pushing for changes that improve reliability and velocity.
- Practice sustainable incident response and blameless post-mortems.

Sustainable Incident Response

Incident Response: On Call

Being on-call is a critical duty that operations and engineering teams must undertake:

- To support services during working and non-working hours
- To respond to issues (outages, incidents, etc.)
- To ensure SLO's are being met



Incident response must be "sustainable" as per the Google Job Description

On Call By Numbers (1)

How much down time is allowed for service issues?

- SLO's introduce a constraint on the amount of time available
- A service with a “three-nines” availability SLO requires all issues in a month to be fixed inside 43 minutes
- This time includes issue identification, alerting, messaging, triage and fix

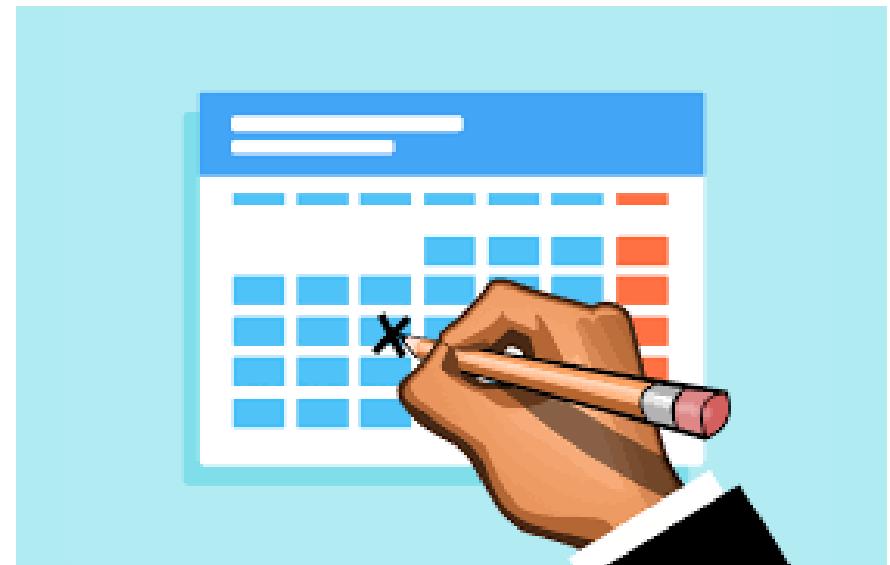
	per year	per quarter	per month	per week	per day	per hour
90%	36.5 days	9 days	3 days	16.8 hours	2.4 hours	6 minutes
95%	18.25 days	4.5 days	1.5 days	8.4 hours	1.2 hours	3 minutes
99%	3.65 days	21.6 hours	7.2 hours	1.68 hours	14.4 minutes	36 seconds
99.5%	1.83 days	10.8 hours	3.6 hours	50.4 minutes	7.20 minutes	18 seconds
99.9%	8.76 hours	2.16 hours	43.2 minutes	10.1 minutes	1.44 minutes	3.6 seconds
99.95%	4.38 hours	1.08 hours	21.6 minutes	5.04 minutes	43.2 seconds	1.8 seconds
99.99%	52.6 minutes	12.96 minutes	4.32 minutes	60.5 seconds	8.64 seconds	0.36 seconds

You can see why appropriate SLO's for services are so important

On Call By Numbers (2)

How often are SRE's on call?

- As well as the 50% toil limit Google also advocate a 25% on-call rule
- To avoid single point of failure at least two SRE's are on-call
- Providing 24/7 support will require eight SRE's
- SRE's are on call one week every month



Spreading on call across multiple sites can help but beware of under-load

Checklist For Effective On Call

- Allocated individuals – giving organizational clarity around who is “on call”
- Suitable devices – providing a mechanism for receiving all relevant information
- Alert delivery systems – capturing and delivering the right information and any background context
- Documented procedures – minimizing the risk of on call response activities
- Blameless postmortems – Making sure that the same issue does not repeat

Responses to on call issues should be rational, focused and deliberate – SRE's need to feel “safe”

Using Automation to Replace On-Call

- Being on call sucks – there is no avoiding that.
- However, we can look to use automation to replace the on-call burden
 - Self healing services allow us to move aware from traditional operator repair service
 - Automation avoids the issue of impaired judgement for called-out operators
 - Tools like Kubernetes, AWS auto-scaling groups provide self-healing capabilities out of the box
 - Self-healing actions can be reviewed in blameless post-mortems

Responses to on call issues should be rational, focused and deliberate – SRE's need to feel “safe”

Module 7: Organizational impact of SRE

Blameless Post Mortems

“So, failure happens. This is a foregone conclusion when working with complex systems.”

John Allspaw, (former) CTO, Etsy



Module 7: Organizational impact of SRE

260

Reasons for a Blameless Post Mortem

- User-visible downtime or degradation beyond a certain threshold (e.g. SLO)
- Data loss of any kind
- On-call engineer intervention (release rollback, rerouting of traffic, etc.)
- A resolution time above some threshold
- A monitoring failure (which usually implies manual incident discovery)

Define criteria before an incident occurs so that everyone knows when a postmortem is necessary.

Blameless Post Mortems

Those involved in the failure can give a detailed account of....

- What actions they took at what time,
- What effects they observed,
- Expectations they had,
- Assumptions they had made,
- Their understanding of timeline of events as they occurred.



Without fear of **punishment** or **retribution**.

Culture and the Flow of Information

Pathological (Power-oriented)	Bureaucratic (Rule-oriented)	Generative (Performance-oriented)
Information is hidden	Information may be ignored	Information is actively sought
Messengers are 'shot'	Messengers are isolated	Messengers are trained
Responsibilities are shirked	Responsibility is compartmentalized	Responsibilities are shared
Bridging is discouraged	Bridging is allowed but discouraged	Bridging is rewarded
Failure is covered up	Organization is just and merciful	Failure causes enquiry
Novelty is crushed	Novelty creates problems	Novelty is implemented

Source: Westrum, *A Typology of Organizational Cultures*

High-trust organizations encourage good information flow, cross-functional collaboration, shared responsibilities, learning from failures and new ideas.

Creating a Safe Environment

“Regardless of what we discover, we understand and truly believe that everyone did the best job they could, given what they knew at the time, their skills and abilities, the resources available, and the situation at hand”

Norm Kerth, Project Retrospectives: A Handbook for Team Review



Creating a Safe Environment

Need to balance safety and accountability

- We give engineers the requisite authority to improve safety by allowing them to give detailed accounts of their contributions to failures.
- We encourage people who do make mistakes to be the experts on educating the rest of the organization how not to make them in the future.

“To be effective we need more people to access production, rather than less”, Damon Edwards, Rundeck

Changing Failure Reaction

“In order to understand how failures happen, we first have to understand our reactions to failure.” John Allspaw, Etsy

- Assume the single cause is incompetence and scream at engineers to make them “pay attention!” or “be more careful!”
- Look at how the incident actually happened, treat the engineers involved with respect, and learn from the event.

Which side would you prefer?

Post Mortem Outputs

- Details of the incident or failure, summary, impact, trigger, detection, resolution, participants
- A list of follow up actions to mitigate future chances of this incident, or similar ones, happening again
- Lessons learned from the incident
- A time-line of what happened
- Any supporting information

Incident management automation can help streamline the creation of these outputs

CASE STORY: Sage Group

"When an incident occurs a post is made in a Slack channel called "incidents". This triggers an automated form to capture more details about the incident, all of which is fed into our service management tool, Service Now. A dedicated Slack channel is also created automatically adding interested parties – the channel also posts regular reminders to keep our customers/stakeholders updated. For high impact incidents a Zoom "bridge" is created so participants can collaborate in real time about the incident. Finally the incident cannot be closed without a post mortem taking place."

"We want to automate the process around the incident as much as we can so our engineers can focus fully on fixing the incident."



Jon Noble
SRE, Sage Group

Benefits

- "Chat Ops" approach ensures everyone is kept up-to-date
- Less admin time dealing with incidents
- Workflow ensures all incidents are followed up and learning shared.

Module 7: Organizational impact of SRE

SRE & Scale

SRE – A Reminder

- Google didn't invent "SRE" to write some cool books or to add another acronym to the IT world
- They did it to solve a problem: how do you handle problems in massively distributed systems operating at mind-blowing scale?

Key Success Factors for SRE Adoption

- Exec support and buy-in
- Funding
- Good relationships across the delivery spectrum – engineers, testing, infrastructure, etc.
- An organization that is growing....



An organization that is not growing (or scaling) will not face the kinds of challenges that SRE is trying solve

An organization that is growing....

Are you witnessing growth across any of these domains?

- Platform growth (large volumes of users, irregular data flows, legacy-to-modern architectures)
- Scope growth (new products/services)
- Ticket growth (volume of incidents/outages/requests/toil)



"The challenge of scale is always a good one" Stig Sorensen, Bloomberg

Use Engineering Approaches to Scale

"Through engineering solutions SRE allows organizations to scale their services at a much greater rate than the scale of their organisation"

Jennifer Petoff, Google



SRE Approaches to Platform Growth

There are a range of automation capabilities to safely handle platform growth

- Automation techniques such as auto scaling, containerization, clustering
- Flexible platforms such as public/private cloud
- Non-structure databases, NoSQL, MongoDB
- “As-a-service” capabilities around build, deploy, test, monitoring



SRE Approaches to Scope Growth

- SRE ownership of common tools and platforms ("platform SRE") which other development use
- SRE expertise "shifting left" into development teams ("embedded SRE")
- Automating toil makes more SRE time available for development



"Team size should not scale directly with service growth" Betsey Beyer, Google

SRE Approaches to Ticket Growth

- Toil reduction approaches such as automated ticket responses and "self service" features
- Prioritizing toil reduction activities
- DRY – don't repeat yourself – solutions to eliminate repetition of problems



"Build the automation to remove the need for toil-related tickets" Mark Rendell, Accenture

Module 7: Organizational impact of SRE

EXERCISE

Your organizational plan for SRE

CASE STORY: Department for Work & Pensions

“A major focus of SREs is the aspiration to never see the same issue twice, often using automation as a resolution. We spend a large amount of time on reducing human labor, sharing knowledge among teams, and creating a blameless environment”



Sean Lukel
Chief SRE, DWP

“We’re not competing with Google or Spotify but our goals for reliability are the same.”

Benefits

- Appropriate SLO's by matching business criticality with target availability
- Reduced number of high impact outages
- Proven scalability of digital services provided to 20+ million users across the UK

Module Seven Quiz

1	When is it best to hold a blameless post mortem?	a) After the daily stand-up b) Every Monday morning c) After every incident d) When an incident matches pre-set criteria
2	What can be done to ensure staff are not being “burnt out” providing on-call support	a) Hire more SRE's b) Pay SRE's more c) Set a 25% on call limit for SRE's d) Make some SRE's 100% on call
3	A geographically spread team are involved in fixing a live issue – what approach can help out?	a) DevOps b) Align teams in one country with a service c) Fly everyone to the same location d) Chat Ops
4	What outputs should be produced after every post mortem?	a) Disciplinary procedures against the on-call engineer b) A new product backlog c) Some follow up actions to mitigate future incidents d) A flexing of the error budget
5	What is a key factor behind SRE success?	a) A growing organization b) A shrinking organization c) An organizational restructure d) Embracing agile principles

Module Seven Quiz

1	When is it best to hold a blameless post mortem?	a) After the daily stand-up b) Every Monday morning c) After every incident d) When an incident matches pre-set criteria
2	What can be done to ensure staff are not being “burnt out” providing on-call support	a) Hire more SRE's b) Pay SRE's more c) Set a 25% on call limit for SRE's d) Make some SRE's 100% on call
3	A geographically spread team are involved in fixing a live issue – what approach can help out?	a) DevOps b) Align teams in one country with a service c) Fly everyone to the same location d) Chat Ops
4	What outputs should be produced after every post mortem?	a) Disciplinary procedures against the on-call engineer b) A new product backlog c) Some follow up actions to mitigate future incidents d) A flexing of the error budget
5	What is a key factor behind SRE success?	a) A growing organization b) A shrinking organization c) An organizational restructure d) Embracing agile principles

Module 8

SRE, OTHER FRAMEWORKS, TRENDS

© DevOps Institute unless otherwise stated

Module 8: SRE, Other Frameworks, Trends

- SRE & Other Frameworks
- SRE Evolution

Component	Module 8 Content
Video	A Look at ITIL4 & SRE (DevOps Institute)
Case Story	Victor Ops
Discussion	How does agile inform what we do in SRE? Where do you see the future of SRE heading?
Exercise	Sketch board your understanding of SRE

SRE & Other Frameworks

SRE & Other Frameworks

Sometimes the plethora of competing frameworks can seem like a maze



- Agile
- Build pipelines
- Continuous delivery
- DevOps
- Etc....

and of course ITSM

If you have services running in production, then you may benefit from SRE adoption

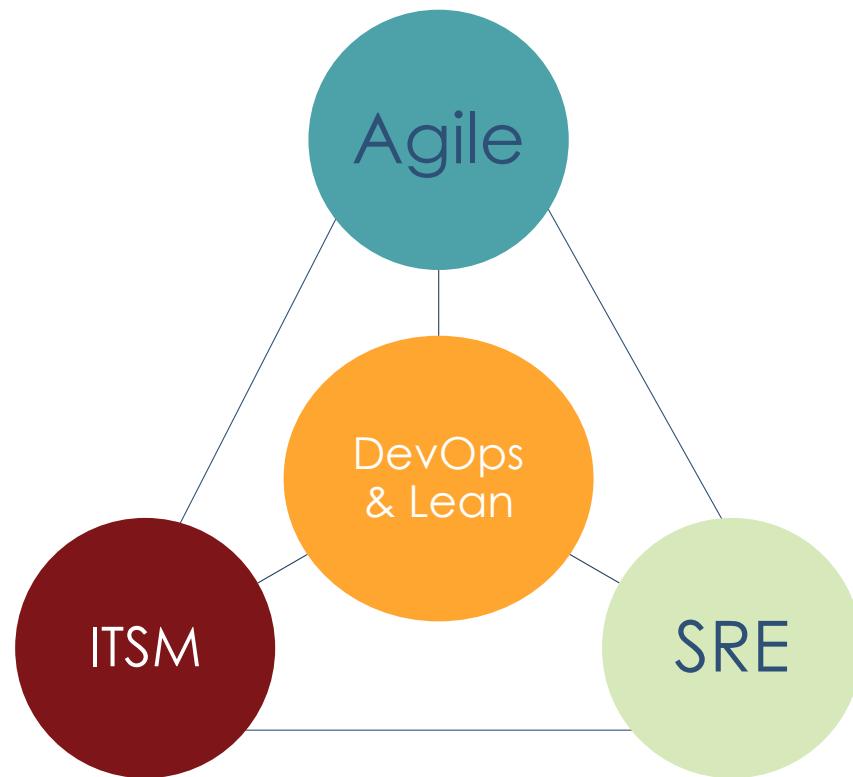
SRE & Other Frameworks

Organizations are looking at their whole value stream and optimizing their approach

Framework opportunities:

- Agile unifies the business with delivery
- DevOps advocates mechanisms like Continuous Delivery & Continuous Deployment for driving velocity and flow
- SRE provides business wide focus on stability & reliability
- ITSM builds organizational learning across the value stream

SRE Does Not Stand Alone

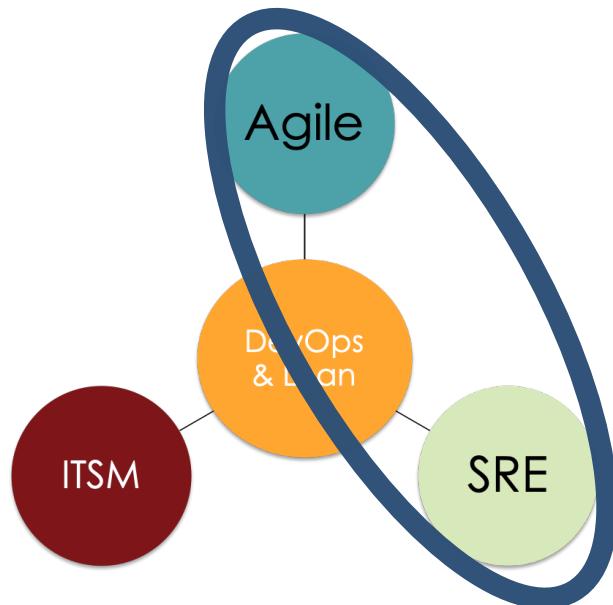


DISCUSSION

How does agile inform what we do in SRE?

SRE Does Not Stand Alone

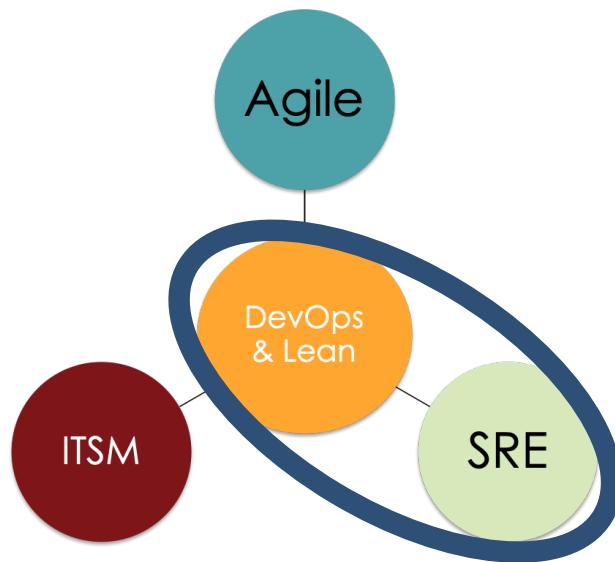
SRE teams can operate in an agile way – using frameworks like Scrum and Kanban



- Backlogs of toil make work visible – and automation can be prioritized
- Ceremonies ensure co-ordination, visibility and prioritization
- Definition of done more production focused
- Value delivered through “working software” (agile principle 1)

SRE Does Not Stand Alone

SRE and DevOps/Lean are complimentary approaches



- Organizational silos are further broken down
- Pipelines of delivery go further
- DevOps metrics and measures are further improved
- Automation more widespread and consistent

CASE STORY: VictorOps

“Like DevOps, there is no one-size fits all approach to Site Reliability Engineering. What works for a company like Google or Facebook doesn’t make sense for us. Effective SRE isn’t simply responding to incidents quickly when they happen, but in building infrastructure, proper testing, and improving the availability of your systems..”



Dan Jones
CTO

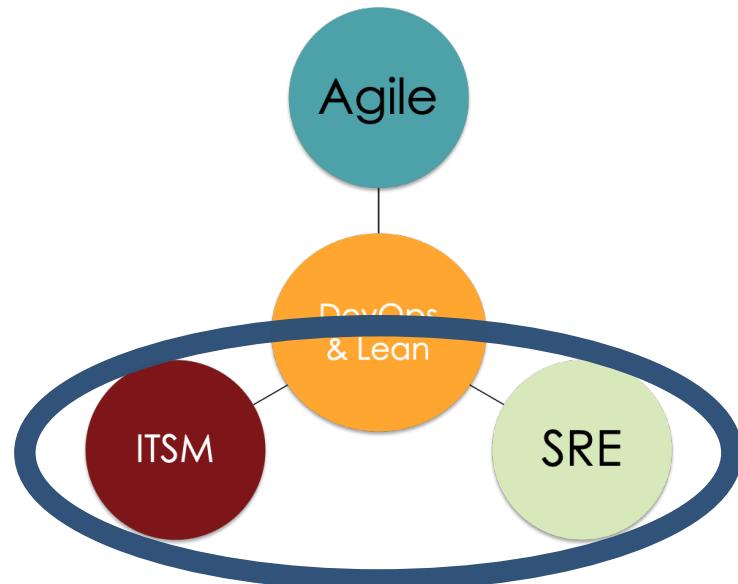
“Reliability is our most important feature”

Benefits

- SRE a key part of a DevOps culture
- Extended DevOps automation to include SRE to improve reliability
- Avoided unnecessary call-outs through understanding a service “normal”
- Reduce the number of “unknowns”

SRE Does Not Stand Alone

SRE can help with ITSM compliance activities through automation & engineering



- Like SRE, ITIL processes are underpinned by automation particularly during transition and operation processes as part of continuous testing and delivery
- In SRE failure is a learning opportunity, continuous learning is embedded in ITSM
- ITIL Provides guidance and structure to processes such as Change, Configuration, Release, Incident and Problem Management –areas that SRE are involved in

AXELOS®, ITIL® and IT Infrastructure Library® are registered trade marks of AXELOS Limited.

ITSM Process Models Support SRE

- Predefined procedures
 - Steps to be taken
 - Chronological order and dependencies
 - Responsibilities
 - Timescales and thresholds
 - Escalation procedures
- Define steps for handling specific types of transaction
- Ensure a defined path or timeline is followed
- Can be automated

Examples

- Change models
- Release models
- Test models
- Incident models
- Problem models
- Request models

AXELOS®, ITIL® and IT Infrastructure Library® are registered trade marks of AXELOS Limited.
Based on ITIL Text - ST 4.2.4.5

ITSM Process Models Support SRE

- Predefined procedures
 - Standard tasks

SRE helps compliance with ITSM by using engineering approaches to remove the human factor from the equation. Automation leaves a documented and auditable trail, this not only increases confidence that ITSM rules aren't being ignored, but it also increases the deploy and release velocity by removing the time needed to make human decisions.

process
models

process
models

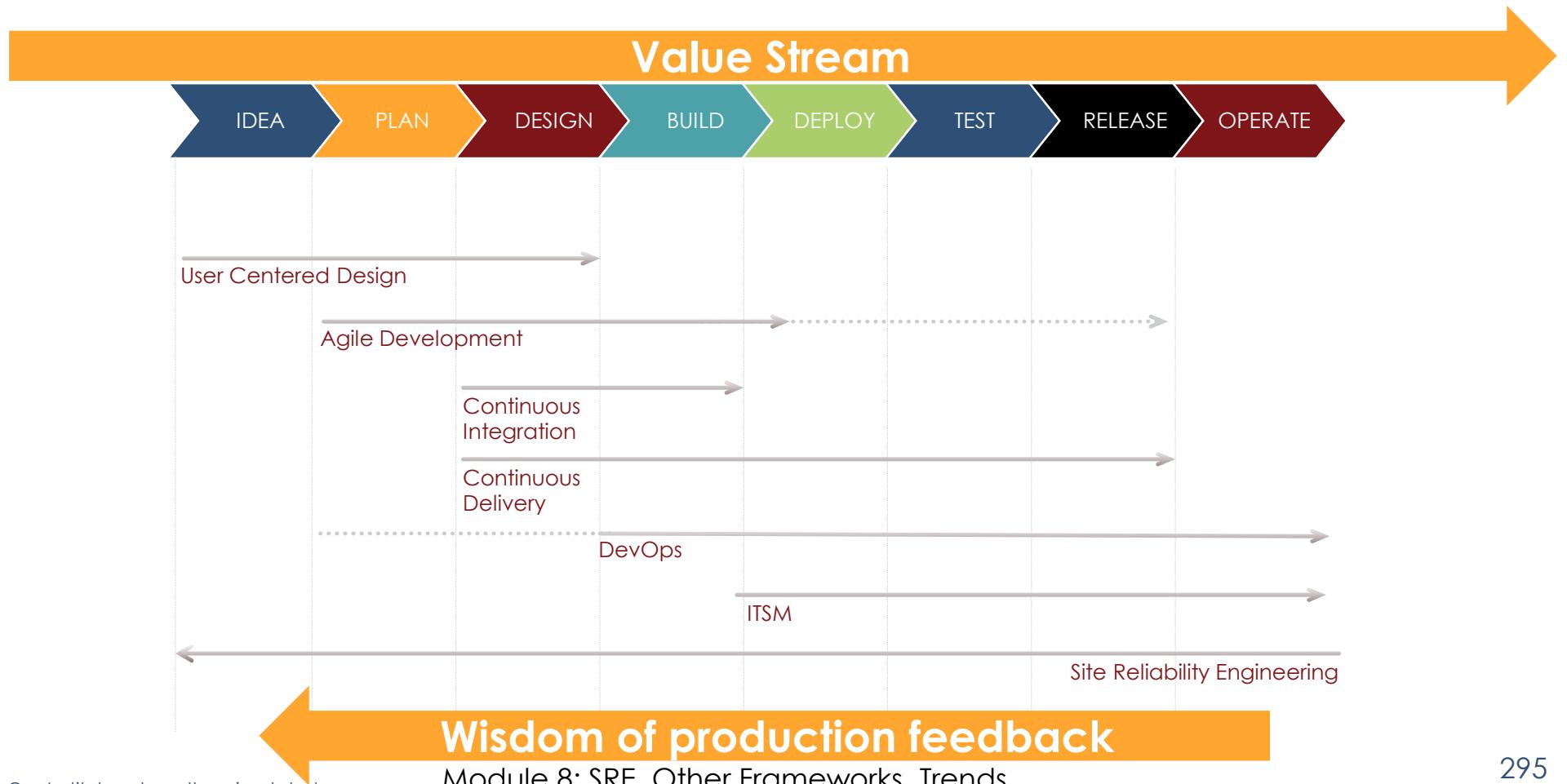
Infrastructure Library® are
trademarks of AXELOS Limited.
Based on ITIL Text - ST 4.2.4.5

SRE Does Not Stand Alone

- SRE emphasizes the development of systems and software that increase the reliability and performance of applications and services.
- DevOps integrates various teams and processes across the development and delivery of software
- ITIL 4 emphasizes service quality and consistency and aims for improved stakeholders' satisfaction through ensuring value from the perspective of the stakeholders.

Stop the Arguments: ITIL 4 and SRE and DevOps All Are Transformation Aids

SRE is part of a “system of systems” for delivery



© DevOps Institute unless otherwise stated

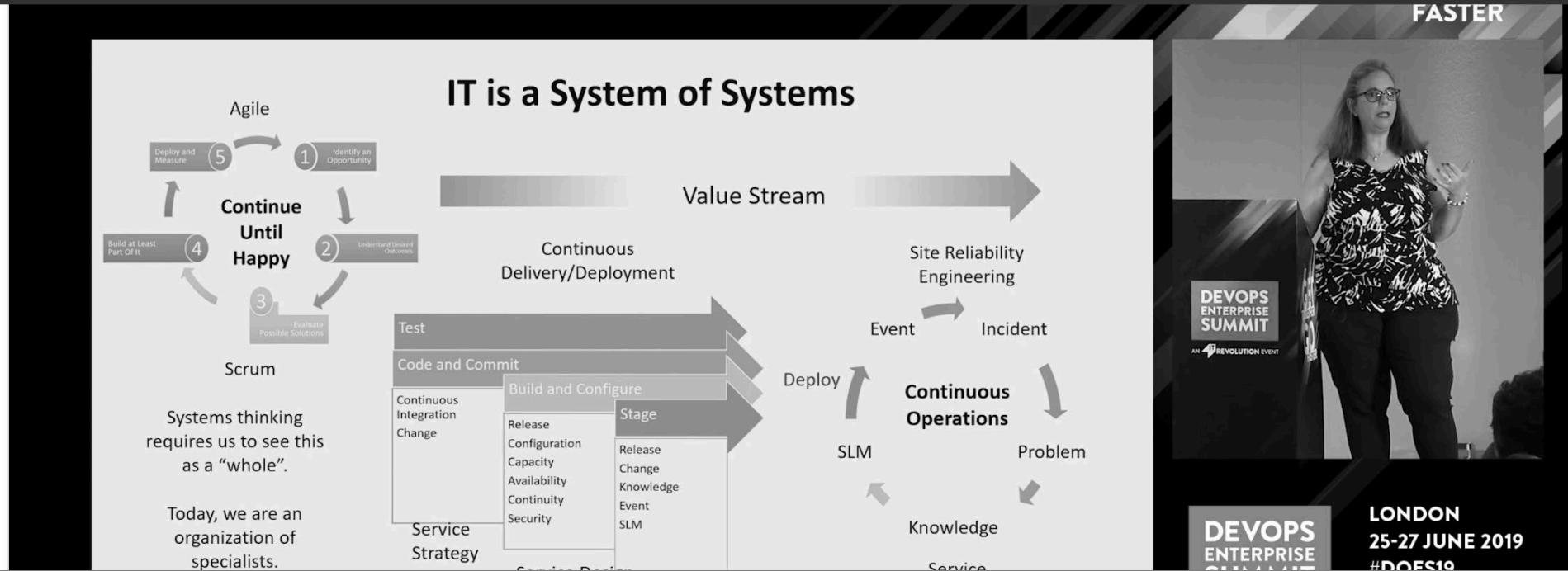
Wisdom of production feedback

Module 8: SRE, Other Frameworks, Trends

295



FASTER



A Look at ITIL4 & SRE, with Jayne Groll
(11:25)

296

Module 8: SRE, Other Frameworks, Trends

SRE Evolution

DISCUSSION

What future challenges do you see coming that effect SRE?

Trends in Site Reliability Engineering

“The five new trends that I see emerging are: failure as the new normal, automation as a service, cloud is king, observe & learn and the evolution of the network engineer (NRE).”

Michael Kehoe, LinkedIn



A Network Reliability Engineer (NRE)

- Applies an engineering approach to measure and automate the reliability of networks
- Codifies software-defined networks (SDN's) and applies SDLC principles to build, test and deploy network changes
- Uses chaos engineering to test the reliability of networks
- Monitors network service-level indicators and creates according automated or manual responses

“The evolution of the network engineer is a major trend in reliability.”

Michael Kehoe, LinkedIn

A Database Reliability Engineer (DBRE)

- Uses software and tooling to automate manual tasks
- Applies chaos engineering to the database to confirm failover and restore responses
- Moves to managed database services e.g. RDS
- Provides innovative solutions to organizational data challenges

“The infrastructure-as-code revolution is also affecting database administration.”

Charity Majors, Facebook

A Customer Reliability Engineer (CRE)

- Applies an engineering mindset to customer support
- Creates a shared responsibility across supplier and customer
- Uses SRE practices to improve customer applications
- Leverages automation (e.g. automated incident management, chat-bots) to reduce toil

“The optimal SLO threshold keeps most users happy while minimizing engineering costs.”

Myk Taylor, Google Cloud

A Heritage Reliability Engineer (HRE)

- Establishes SLO's focused on legacy processing e.g. batch time
- Re-platforms onto modern architectures (mainframe-to-intel - and even cloud)
- Eliminates toil by investing time in engineering
- Expands telemetry to improve observability
- Runs failure tests to ensure critical services remain available

“Even legacy applications have the opportunity to benefit from the new approach.”

Craig Pearson, DWP

Observe & Learn

“I believe that in the next ~3 years, all three of those categories — APM, monitoring/metrics, logs, and possibly others — are likely to cease to exist. There will only be one category: observability. And it will contain all the insights you need to understand any state your system can get itself into.”

Charity Majors, CEO Honeycomb

© DevOps Institute unless otherwise stated

Module 8: SRE, Other Frameworks, Trends



304

EXERCISE

“Sketch board” your understanding of SRE

Module Eight Quiz

1	Which is <u>not</u> a new SRE trend that Michael Kehoe of LinkedIn sees emerging?	a) Failure as the new normal b) Cloud is queen c) Automation as a service d) Observe and learn
2	What does the term NRE stand for?	a) Normal Reliability Engineering b) Network Routing Expert c) Network Reliability Engineer d) Nano Reliability Engineer
3	Which of these is a delivery framework impacted by SRE?	a) Microsoft Project b) Agile Development c) Kano Model d) Java Spring
4	Which of these is an example of how SRE can improve agile development?	a) Definition of done becomes production focused b) Definition of ready includes Chaos Monkey c) Daily stand-up's get longer d) Scrum becomes more predictable
5	Which of these ITSM models can add more structure to SRE activities?	a) Project Management Model b) Incident Model c) Coding Model d) Organization Model

Module Eight Quiz Answers

1	Which is <u>not</u> a new SRE trend that Michael Kehoe of LinkedIn sees emerging?	a) Failure as the new normal b) Cloud is queen c) Automation as a service d) Observe and learn
2	What does the term NRE stand for?	a) Normal Reliability Engineering b) Network Routing Expert c) Network Reliability Engineer d) Nano Reliability Engineer
3	Which of these is a delivery framework impacted by SRE?	a) Microsoft Project b) Agile Development c) Kano Model d) Java Spring
4	Which of these is an example of how SRE can improve agile development?	a) Definition of done becomes production focused b) Definition of ready includes Chaos Monkey c) Daily stand-up's get longer d) Scrum becomes more predictable
5	Which of these ITSM models can add more structure to SRE activities?	a) Project Management Model b) Incident Model c) Coding Model d) Organization Model

Summary

- Site Reliability Engineering:
 - Applies an engineering focus to operations
 - Requires service level objectives to be business led
 - Uses automation to remove toil and improve reliability
 - Gives observable services through effective monitoring
 - Embraces anti-fragility to improve reliability
 - Can co-exist with other frameworks

Become Part of the DevOps Community



Join The DevOps Member Association

Advancing the Human Elements of DevOps



© DevOps Institute unless otherwise stated

309

DevOps Institute Learning Tracks & Community



Why DevOps Institute Certifications Matter?
DevOps Institute offers eight certifications based on role-based competencies that are highly desirable when it comes to hiring and retaining talent in today's most competitive organizations.

Why Get Certified?

- Gain knowledge and understanding of DevOps principles and practices
- Develop your skill set within a key area of DevOps
- Learn how to apply key DevOps principles
- Prove your subject matter expertise
- Enhance your professional credibility
- Stand out from other job applicants

Certifications

DevOps Foundation®
SRE Foundation (SREF)™
Continuous Testing Foundation (CTF)™
DevOps Leader (DOL)™
Continuous Delivery Ecosystem Foundation (CDEF)™
DevSecOps Foundation (DSOF)™
Certified Agile Service Manager (CASM)®
Certified Agile Process Owner (CAPO)®



DevOps Institute advances the human elements of DevOps equipping them with Skills, Knowledge, Ideas and Learning (the SKIL Framework).

DevOpsInstitute.com

© DevOps Institute unless otherwise stated