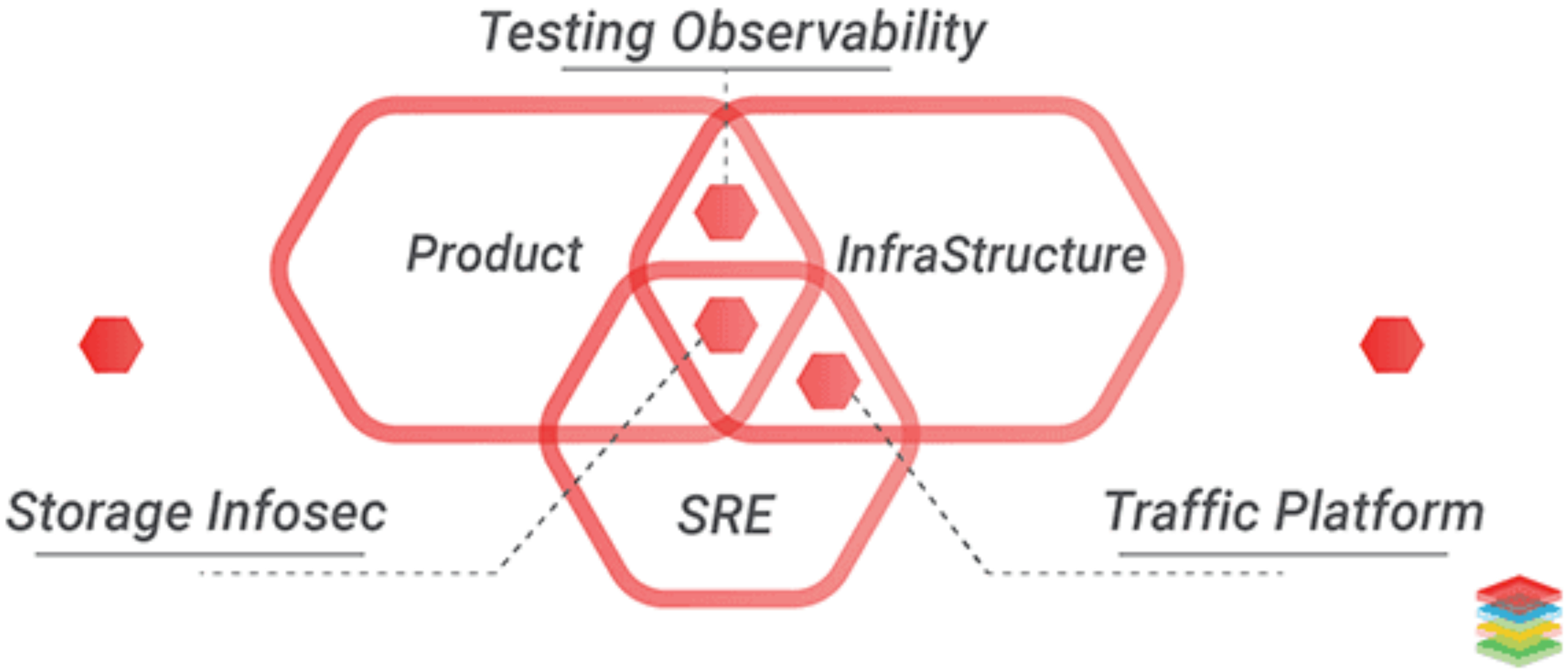
Outline

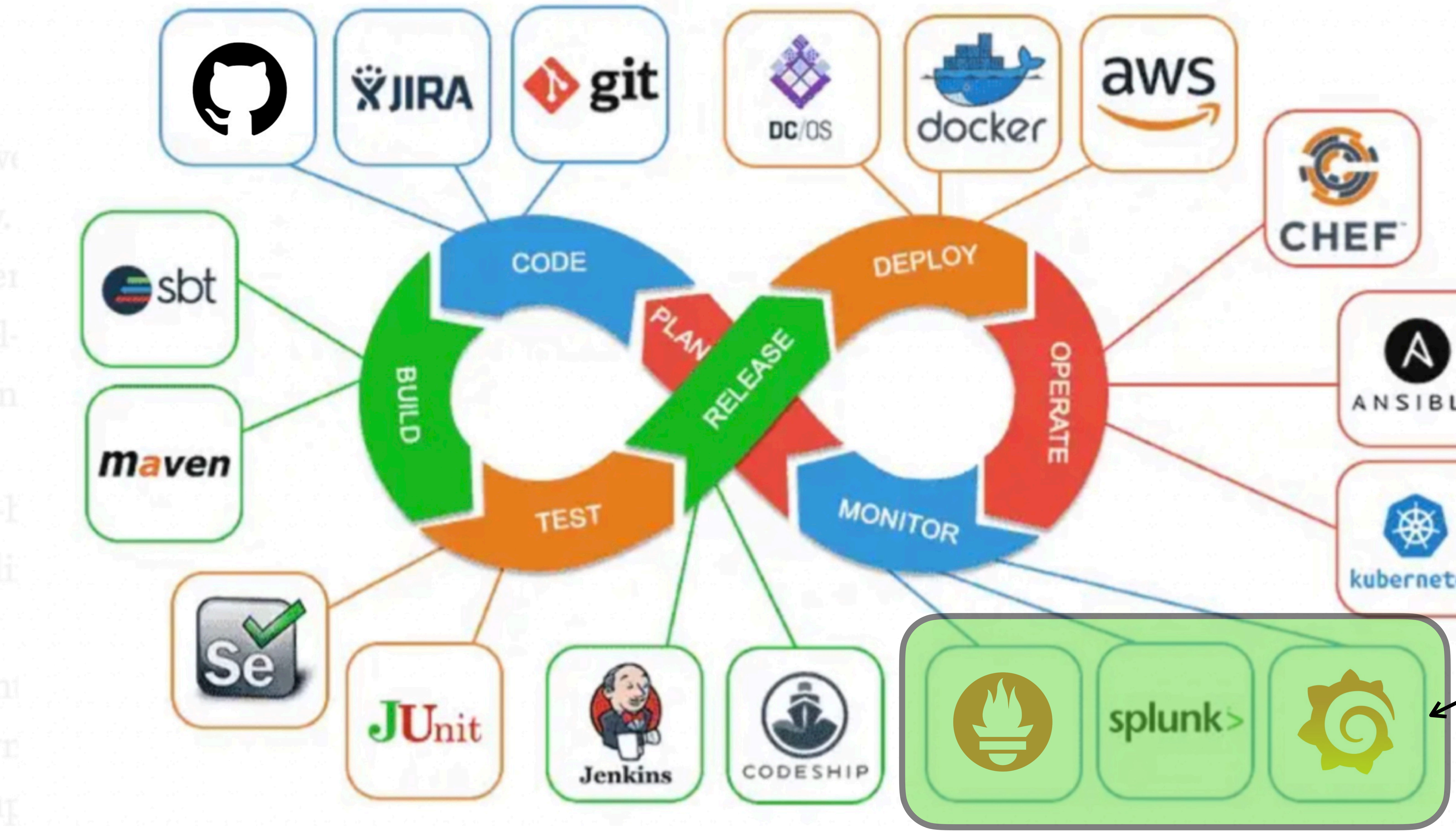
Planned topics for this lesson:

- What is Site Reliability Engineering (SRE)?
  - RELIABILITY, PERFORMANCE + SCALABILITY
- SLIs, SLOs and SLAs
  - WHAT ARE THESE?
- Monitoring and Alerting Tools
  - PROMETHEUS



Site Reliability Engineering (SRE)

An Overview

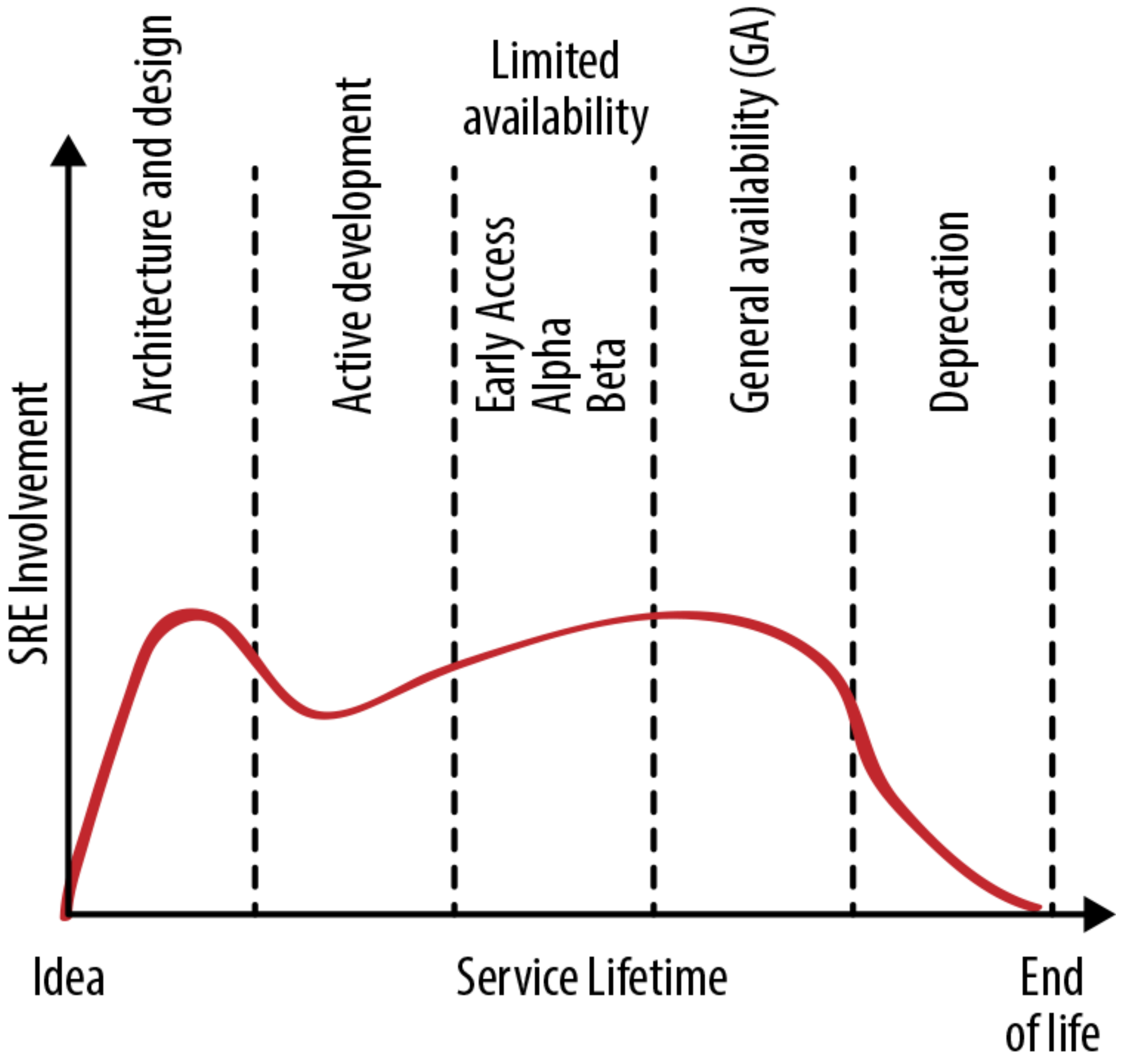


THIS WEEK : SRE

Site Reliability Engineering (SRE)

An Overview

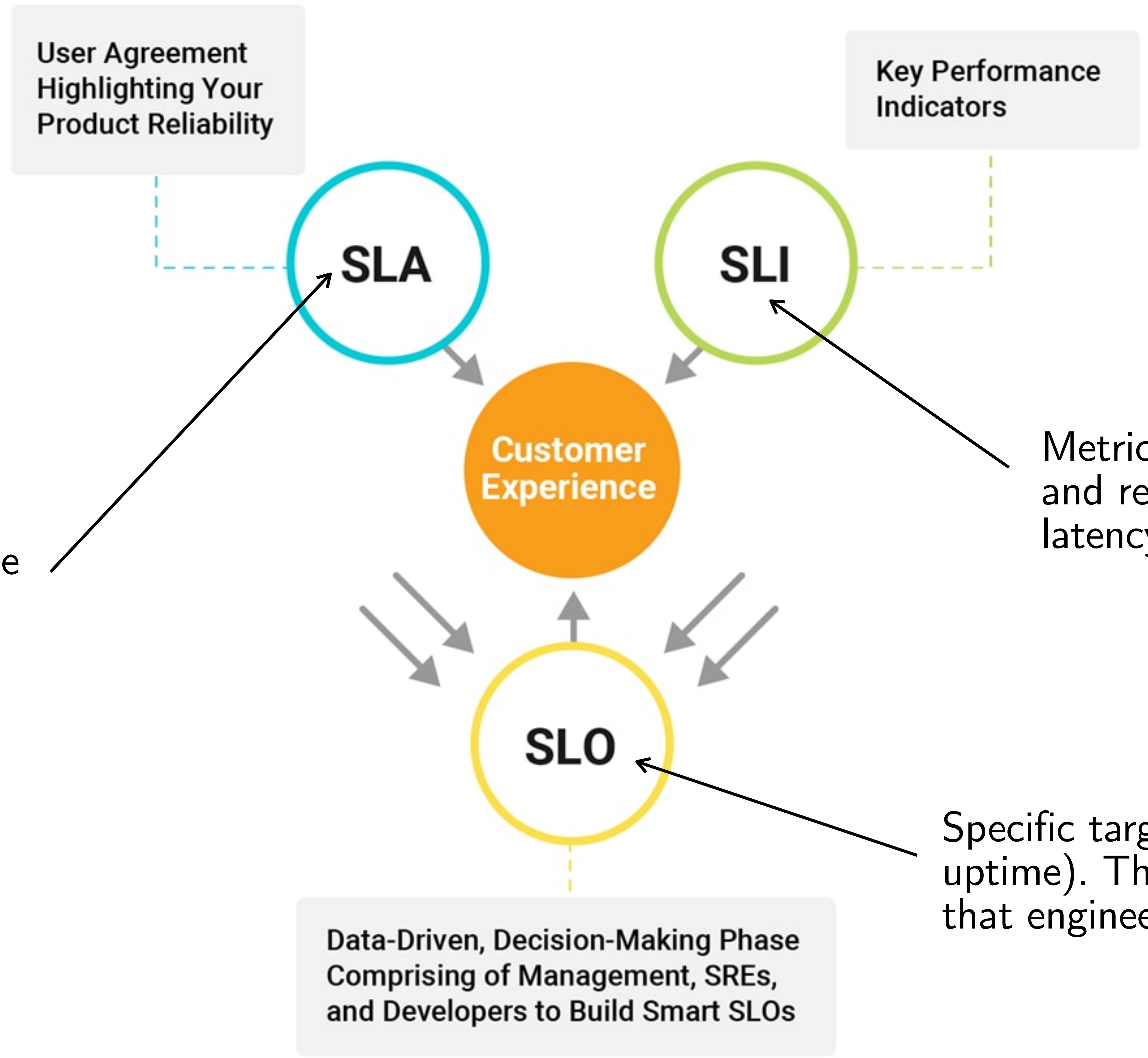
- What is SRE?
  SRE is an engineering discipline that combines software development and IT operations to ensure the reliability, performance, and scalability of large-scale systems.
  - It applies software engineering practices to solve operations problems.
- Goal of SRE:
  Improve system reliability through automation, scalability, and performance optimisation, while balancing new feature delivery with system stability.
- Why SRE Matters in DevOps:
  SRE teams work closely with development teams, helping bridge the gap between releasing new features and maintaining uptime and service quality.
- SRE essentially takes operations to the next level by using software engineering principles to reduce toil and improve system reliability



Site Reliability Engineering (SRE)

Key Concepts

- SLIs, SLOs, and SLAs form the foundation of how SRE teams ensure system reliability.
- The focus is on setting realistic performance goals and measuring whether those goals are met.


User Agreement Highlighting Your Product Reliability
Key Performance Indicators
SLA
SLI
Customer Experience
SLO
Data-Driven, Decision-Making Phase Comprising of Management, SREs, and Developers to Build Smart SLOs

Formal contracts between service providers and users that define penalties if SLOs are not met.

Metrics that measure the performance and reliability of a service (e.g., request latency, availability).

Specific targets for SLIs (e.g., 99.95% uptime). These are performance goals that engineers strive to meet.

Site Reliability Engineering (SRE)

Examples Calculation - SLI

Let's track *Uptime* as the SLI for a web service.

- **Total Time in a Month** = 30 days × 24 hours = 720 hours.
- **Downtime** = 1.44 hours due to server issues in a month.

SLI Calculation:

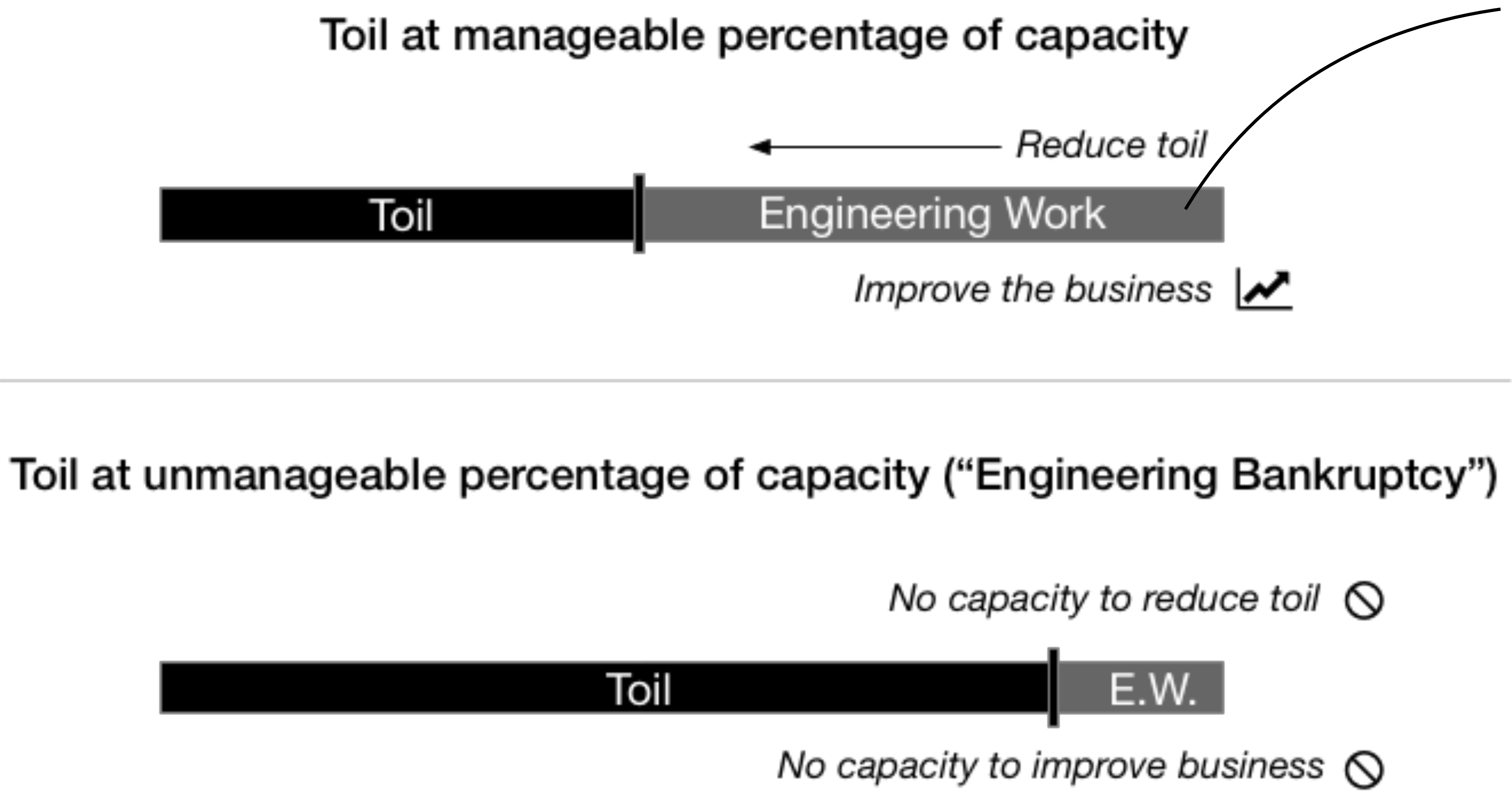
$$SLI = \left(\frac{\text{Uptime}}{\text{Total Time}}\right) \times 100 = \left(\frac{720 - 1.44}{720}\right) \times 100 = 99.8\,\%$$

The SLO is set to 99.9% uptime over a month, meaning the system should not exceed 0.72 hours (43 minutes) of downtime.

- Since the actual downtime was 1.44 hours (exceeding 0.72 hours), the service fails to meet the SLO.

SRE Reduces "Toil"

Key Concepts



Toil at manageable percentage of capacity

Reduce toil

Toil | Engineering Work

Improve the business

Toil at unmanageable percentage of capacity ("Engineering Bankruptcy")

No capacity to reduce toil ⊘

Toil | E.W.

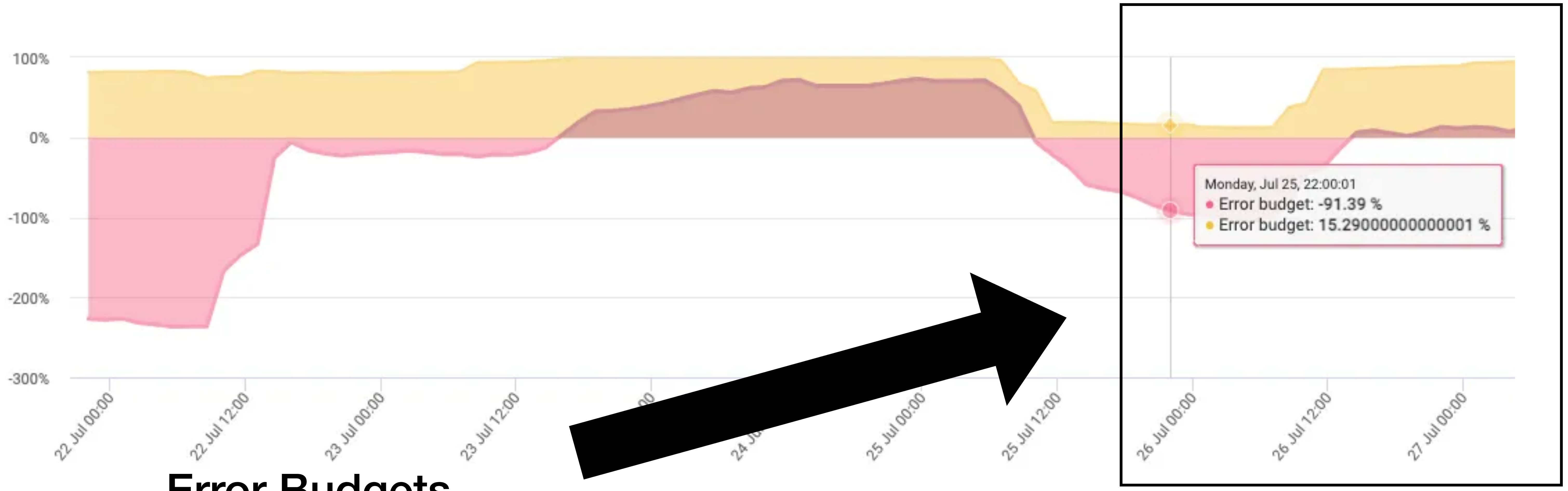
No capacity to improve business ⊘

REDUCING TOIL FREES UP ENGINEERS TO FOCUS ON INNOVATION AND SYSTEM IMPROVEMENTS, LEADING TO HIGHER RELIABILITY AND FASTER INCIDENT RESPONSE

What is Toil?

- Toil refers to repetitive, manual, and automatable work that is typically performed by operations teams to maintain systems.
- SRE's goal is to minimise toil by automating routine tasks.
- As systems grow in complexity and scale, toil increases proportionally unless it is actively reduced through automation or process improvements

Error Budget and Risk Management



Error budgets allow for calculated risks in deploying new features.

If the system's error budget is depleted, focus shifts to stability until reliability is restored.

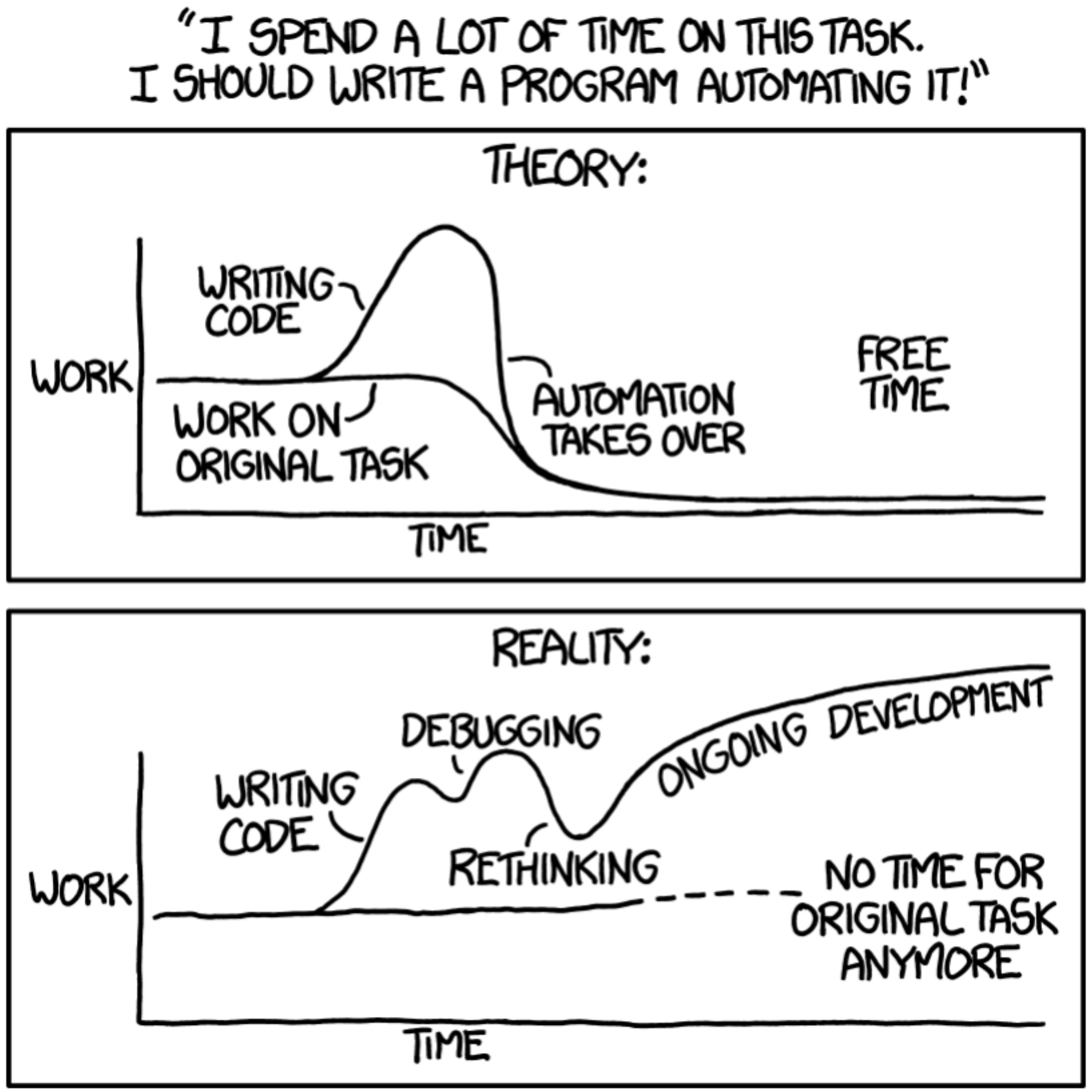
Error Budgets

- An SRE innovation that defines how much downtime or failure is acceptable within a given time frame.
- It's based on SLOs.
- Also - the maximum amount of time that a technical system can fail without contractual consequences.

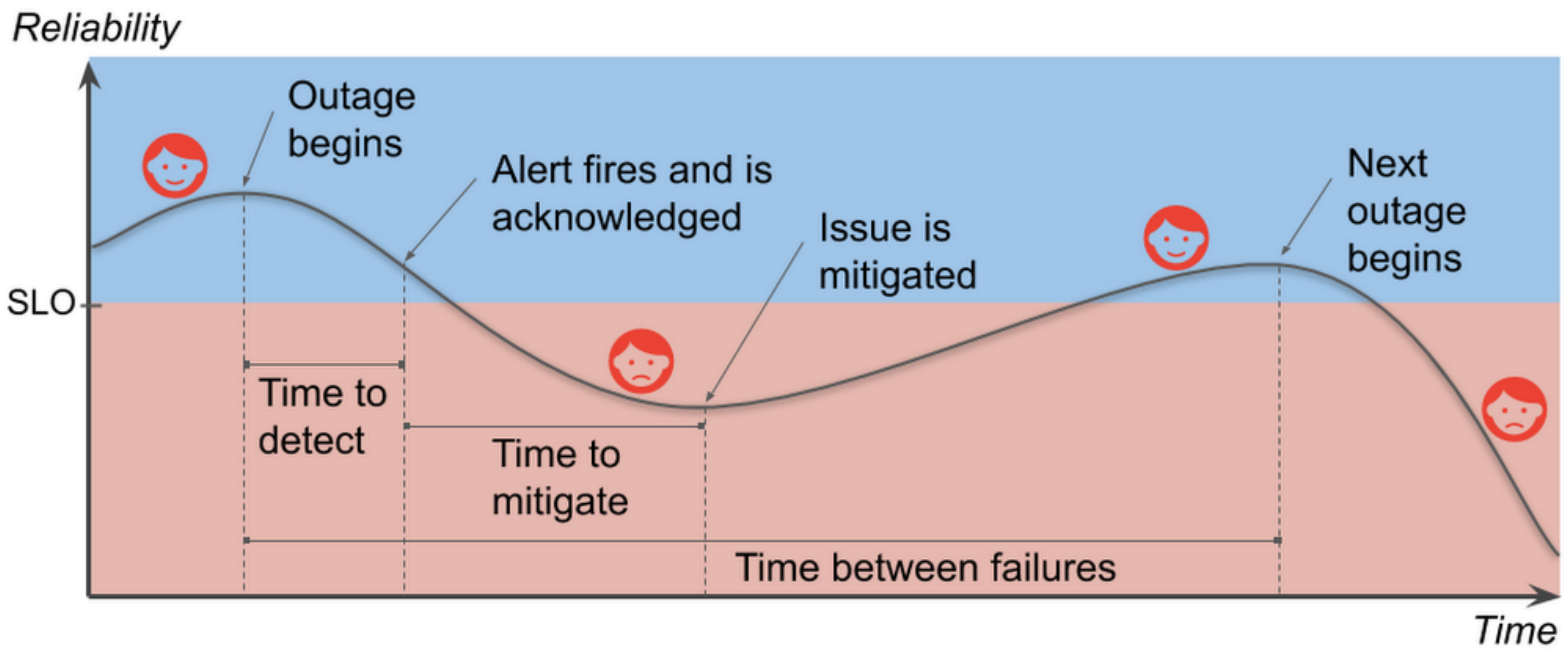
Error budgets help align development and operations teams by providing a shared understanding of risk tolerance, ensuring that uptime doesn't come at the cost of innovation

- If a system has an SLO of 99.9% uptime per month, the remaining 0.1% downtime is the error budget. Teams can deploy updates, perform experiments, or make risky changes.
- But if unplanned downtime uses up this budget, further risky changes are paused.

Error Budget and Risk Management



} BALANCING INNOVATION AND RELIABILITY



Monitoring and Observability

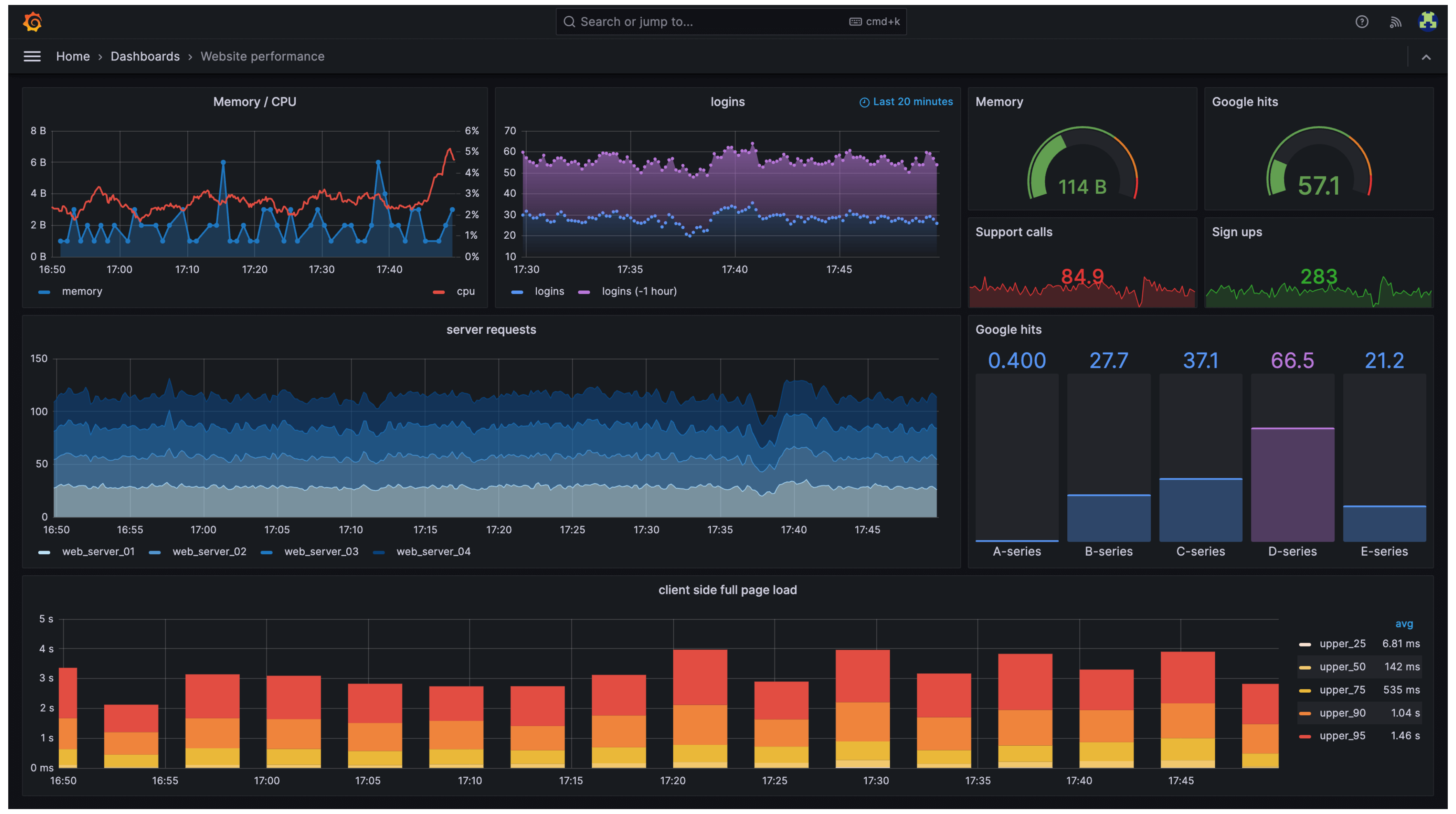
How to Monitor and Observe?

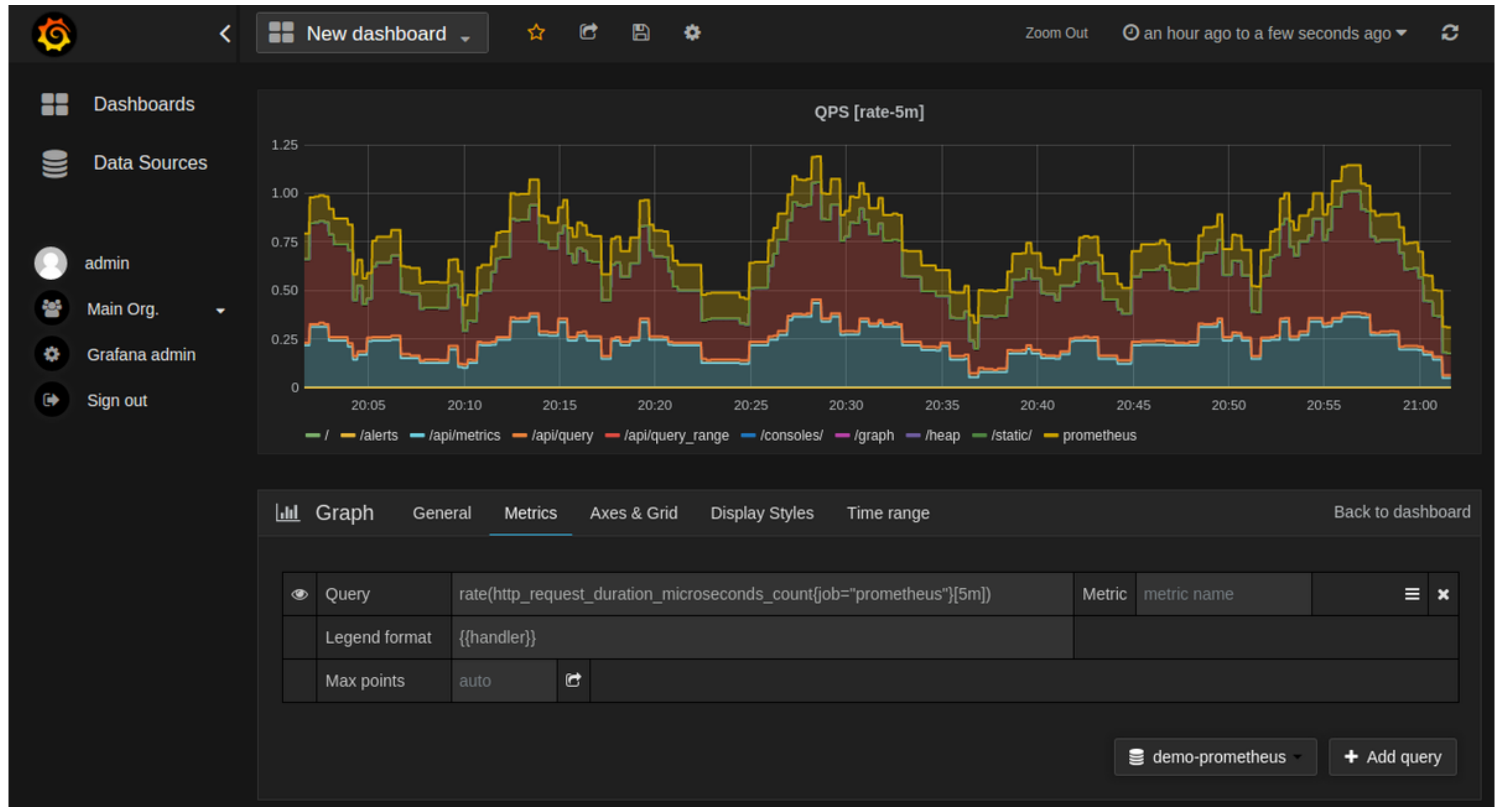
Monitoring — Refers to the process of continuously tracking system performance metrics such as latency, uptime, error rates, and throughput.

Monitoring is proactive and helps detect known issues and failures.

- E.g., CPU usage, memory consumption, and network latency.
- Monitoring is focused on alerting when predefined thresholds are crossed (e.g., high memory usage).

A widely used open-source monitoring tool that scrapes and stores time series data and can trigger alerts





Query: rate(http_request_duration_microseconds_count{job="prometheus"}[5m])
Metric: metric name
Legend format: {{handler}}
Max points: auto
demo-prometheus

Monitoring and Observability

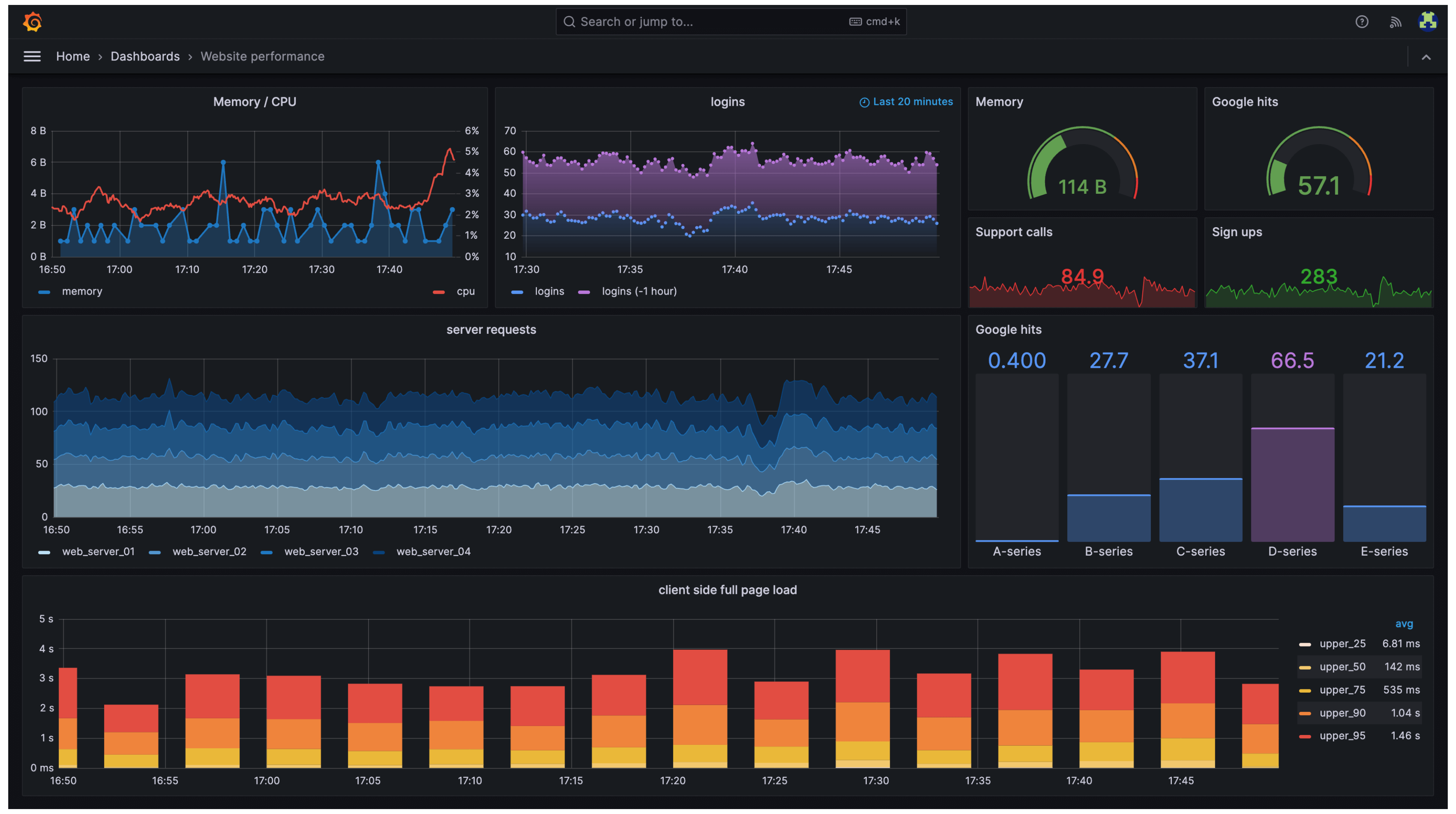
How to Monitor and Observe?

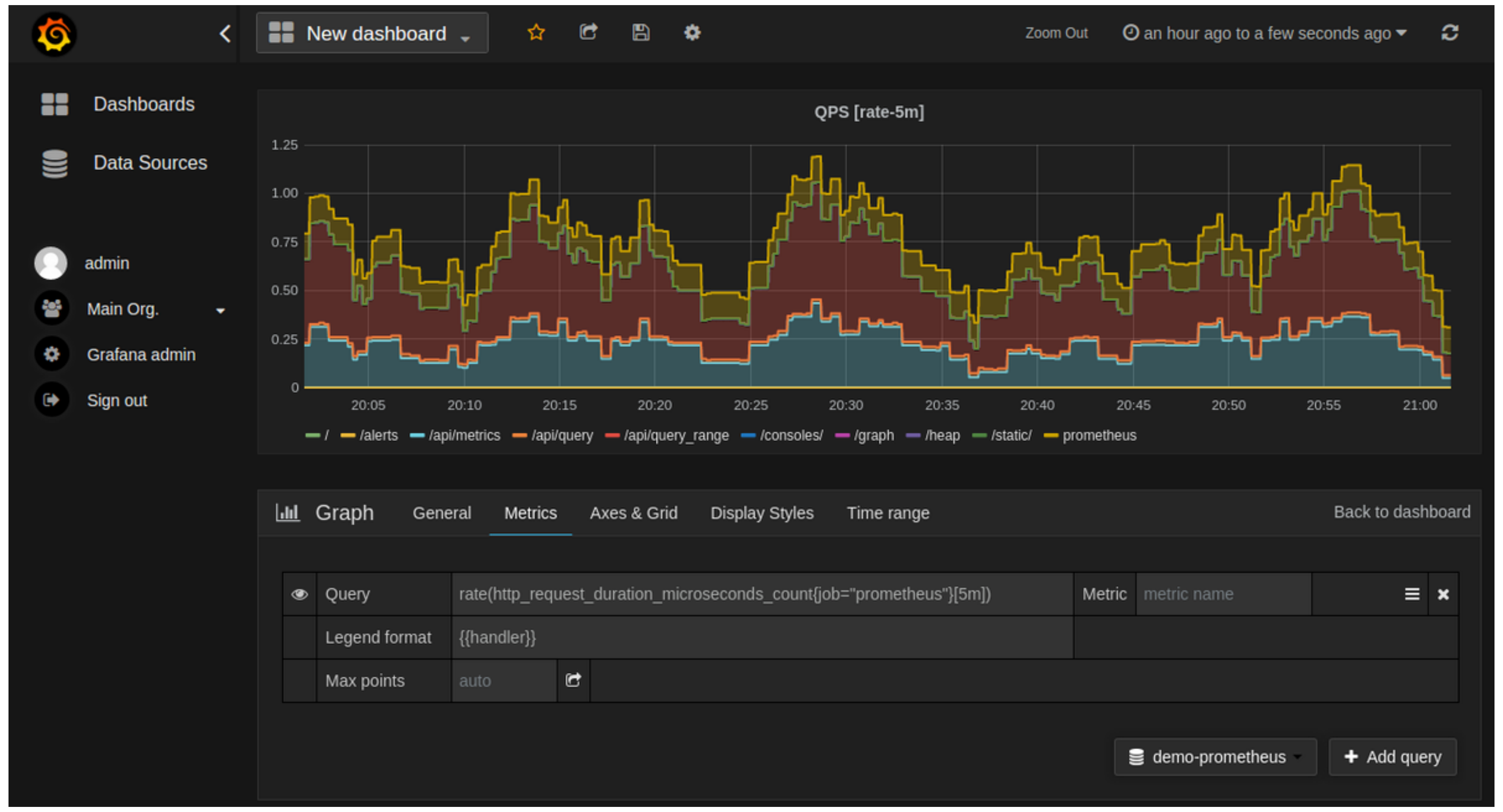
Observability — Goes beyond monitoring by offering deep insights into the internal state of a system based on its outputs (logs, metrics, traces).

Observability aims to help SRE teams understand why something is failing, not just that it's failing.

- Key Elements:
  - Logs: Textual records of system events (e.g., error logs).
  - Metrics: Numerical data over time (e.g., request counts).
  - Traces: A record of the journey of a request across various services (e.g., distributed tracing).
- Observability is focused on diagnosing complex issues in real-time.
- **High observability reduces Mean Time to Recovery (MTTR)**

An open-source visualisation tool that works with Prometheus to create real-time dashboards of system metrics





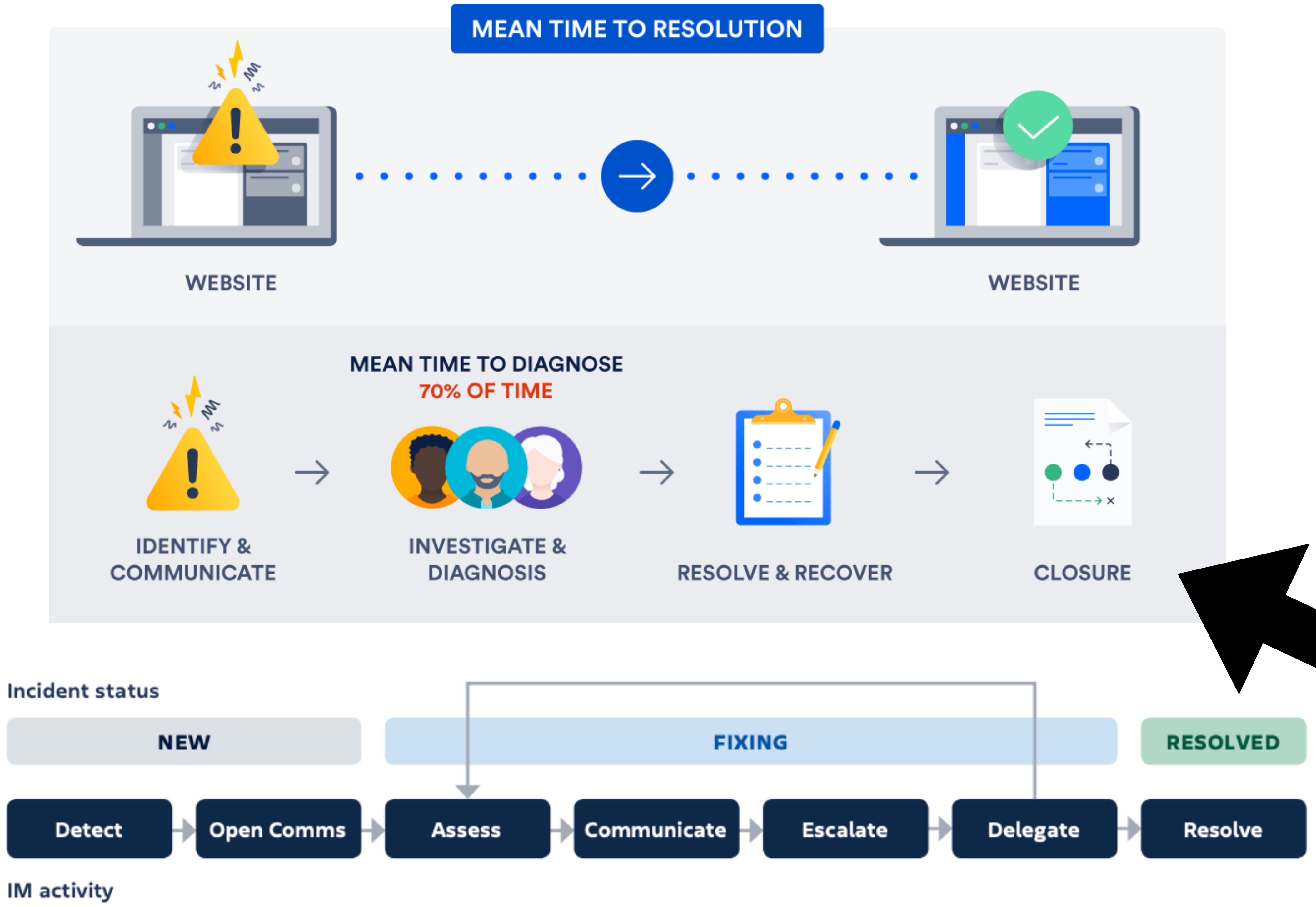
Query rate(http_request_duration_microseconds_count{job="prometheus"}[5m]) Metric metric name

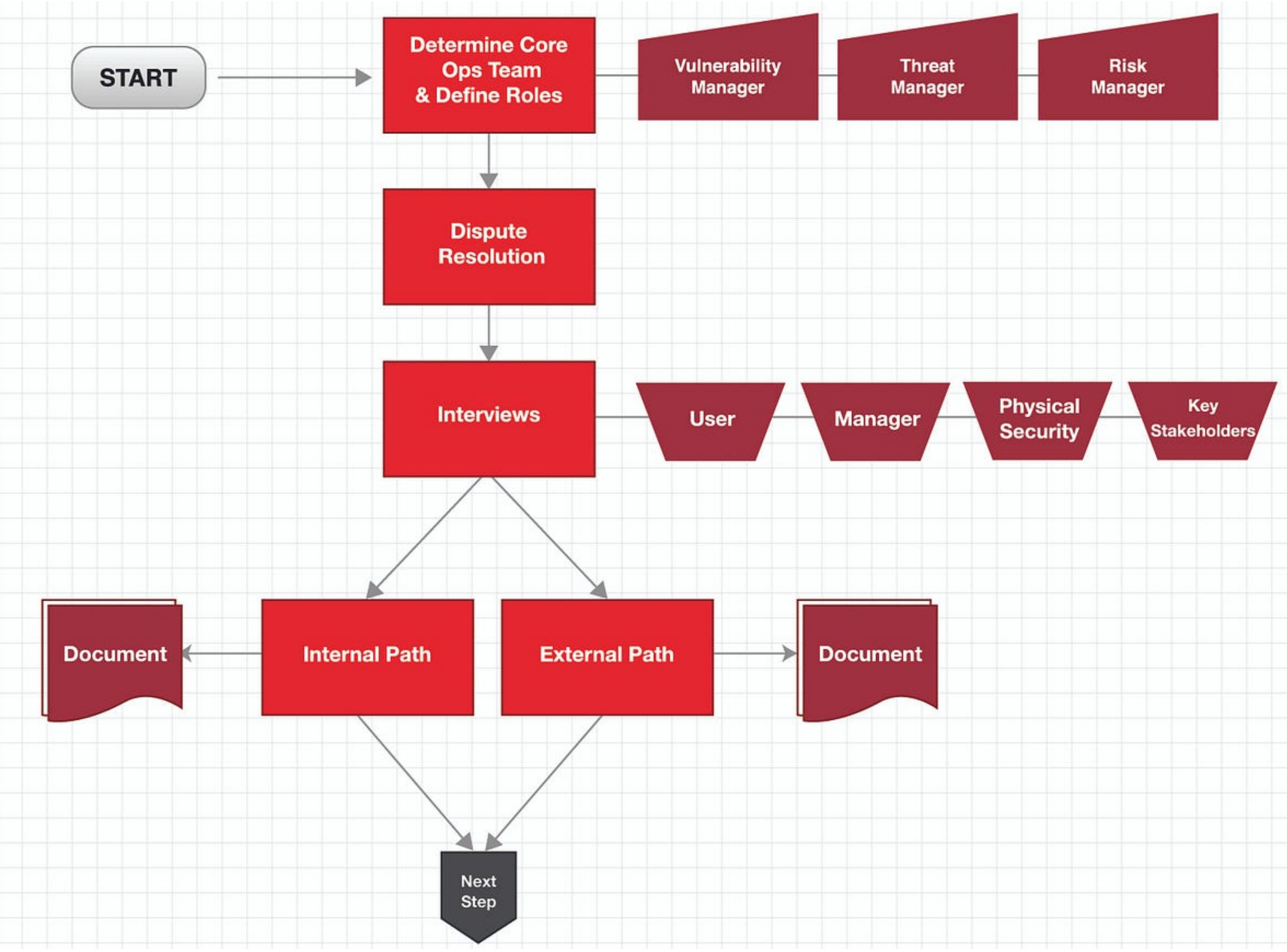
Legend format {{handler}}

Max points auto

Incident Responses

Incident Management - Playbook



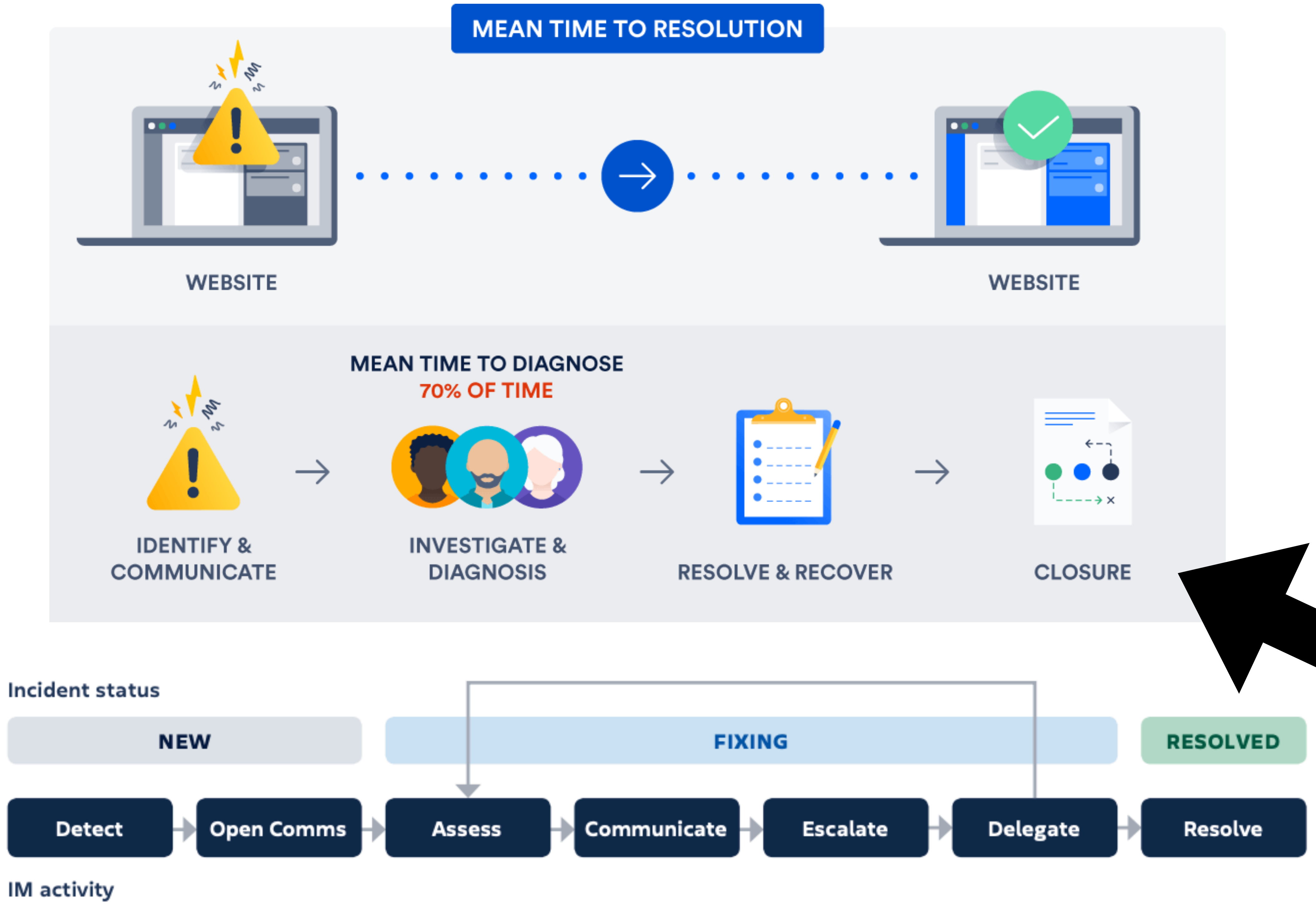


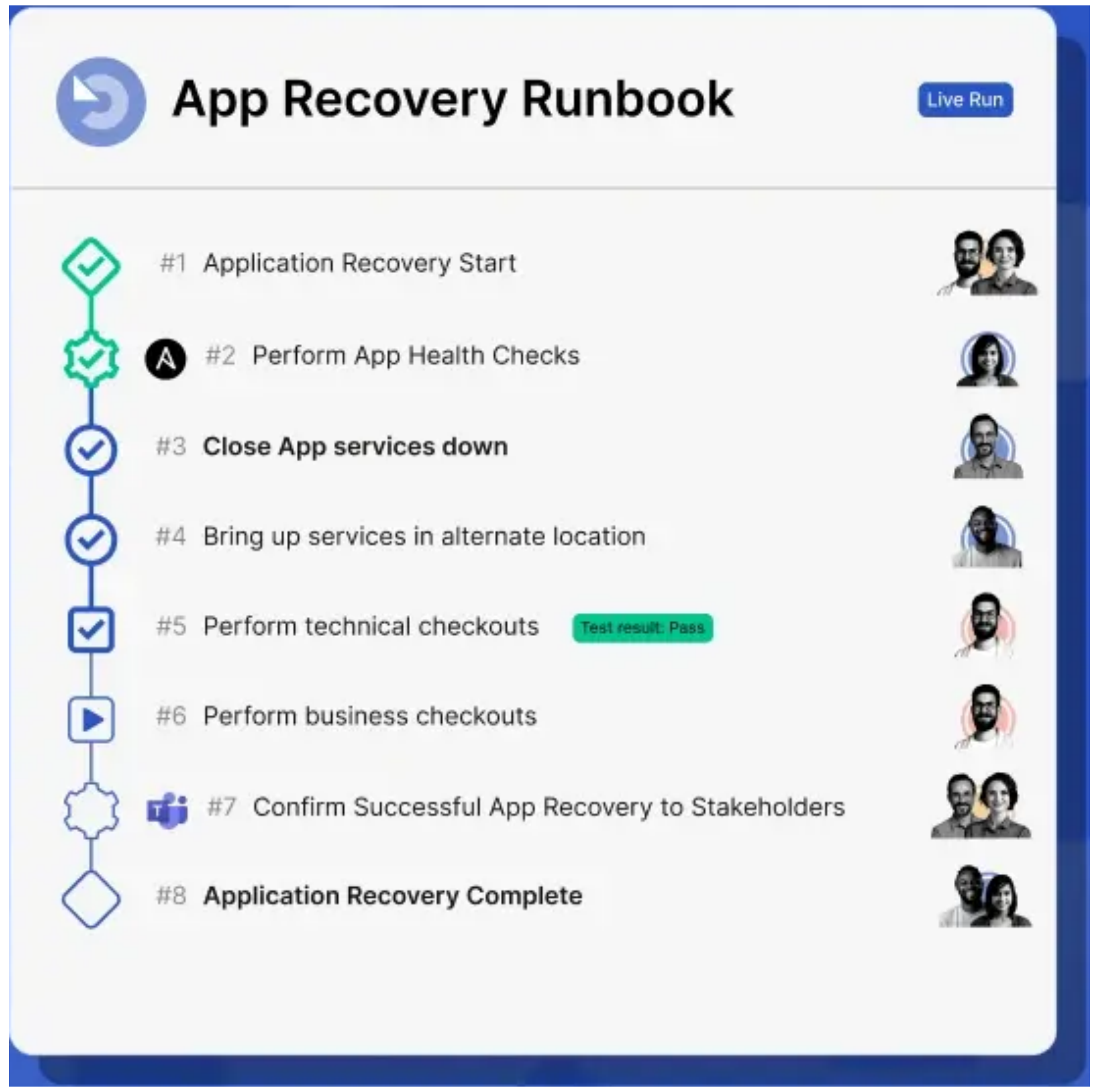
Site Reliability Engineering (SRE) teams create **playbooks** and **runbooks** to define specific steps for responding to incidents.

- **Playbooks —**It provides **organisational goals, escalation paths, communication strategies**, and general guidelines to be followed during different types of incidents. It may contain references to runbooks but is not limited to technical troubleshooting steps.
- **Runbooks —** Documentation that guides the on-call engineer through responding to alerts, monitoring system health, and deciding when to escalate an issue.

Incident Responses

Incident Management - Runbook





App Recovery Runbook

Live Run

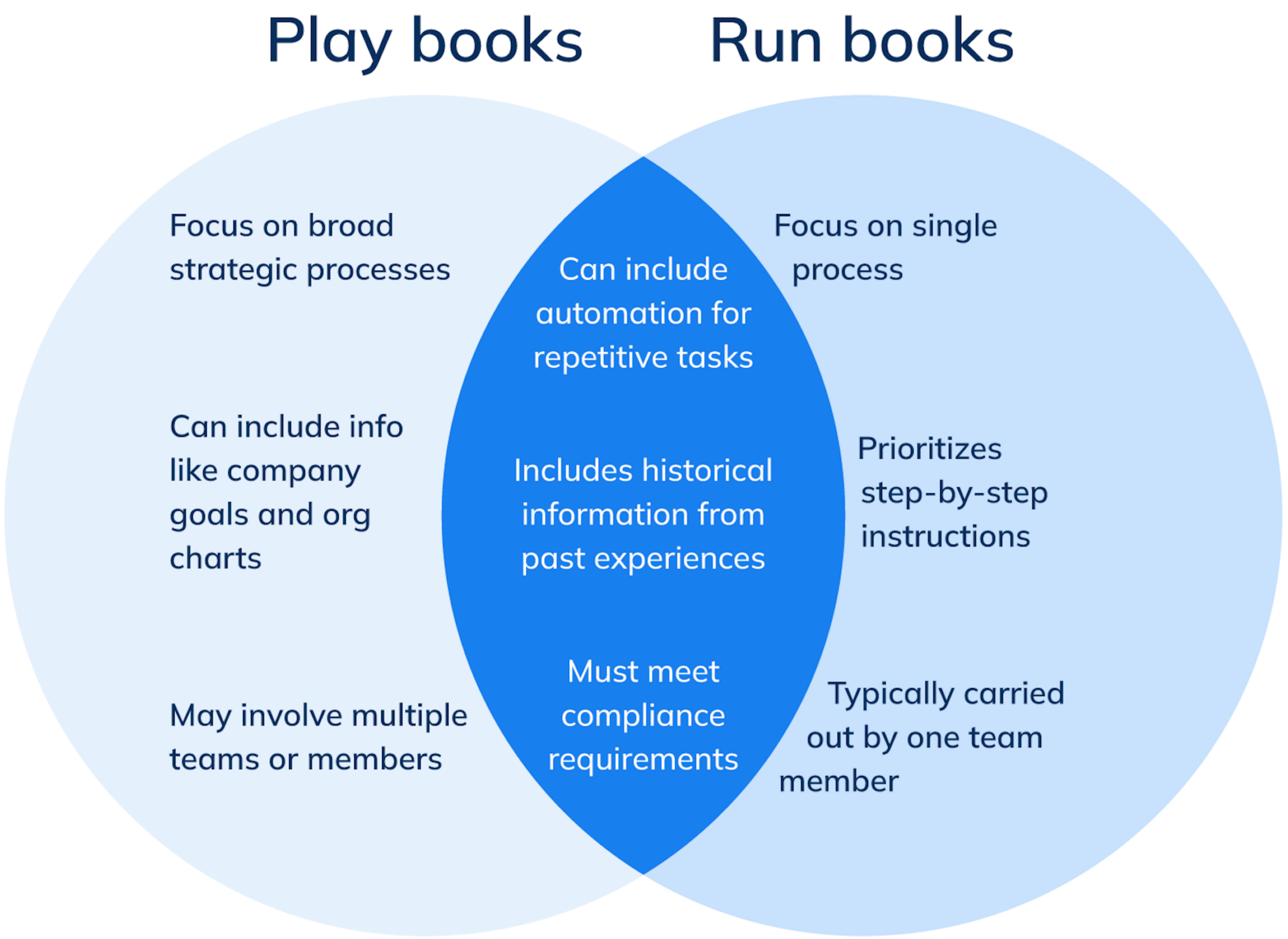
- #1 Application Recovery Start
- #2 Perform App Health Checks
- #3 **Close App services down**
- #4 Bring up services in alternate location
- #5 Perform technical checkouts Test result: Pass
- #6 Perform business checkouts
- #7 Confirm Successful App Recovery to Stakeholders
- #8 **Application Recovery Complete**

Site Reliability Engineering (SRE) teams create **playbooks** and **runbooks** to define specific steps for responding to incidents.

- **Playbooks** —It provides **organisational goals, escalation paths, communication strategies**, and general guidelines to be followed during different types of incidents. It may contain references to runbooks but is not limited to technical troubleshooting steps.
- **Runbooks** — Documentation that guides the on-call engineer through responding to alerts, monitoring system health, and deciding when to escalate an issue.

Incident Responses

Incident Management - Playbook vs Runbook

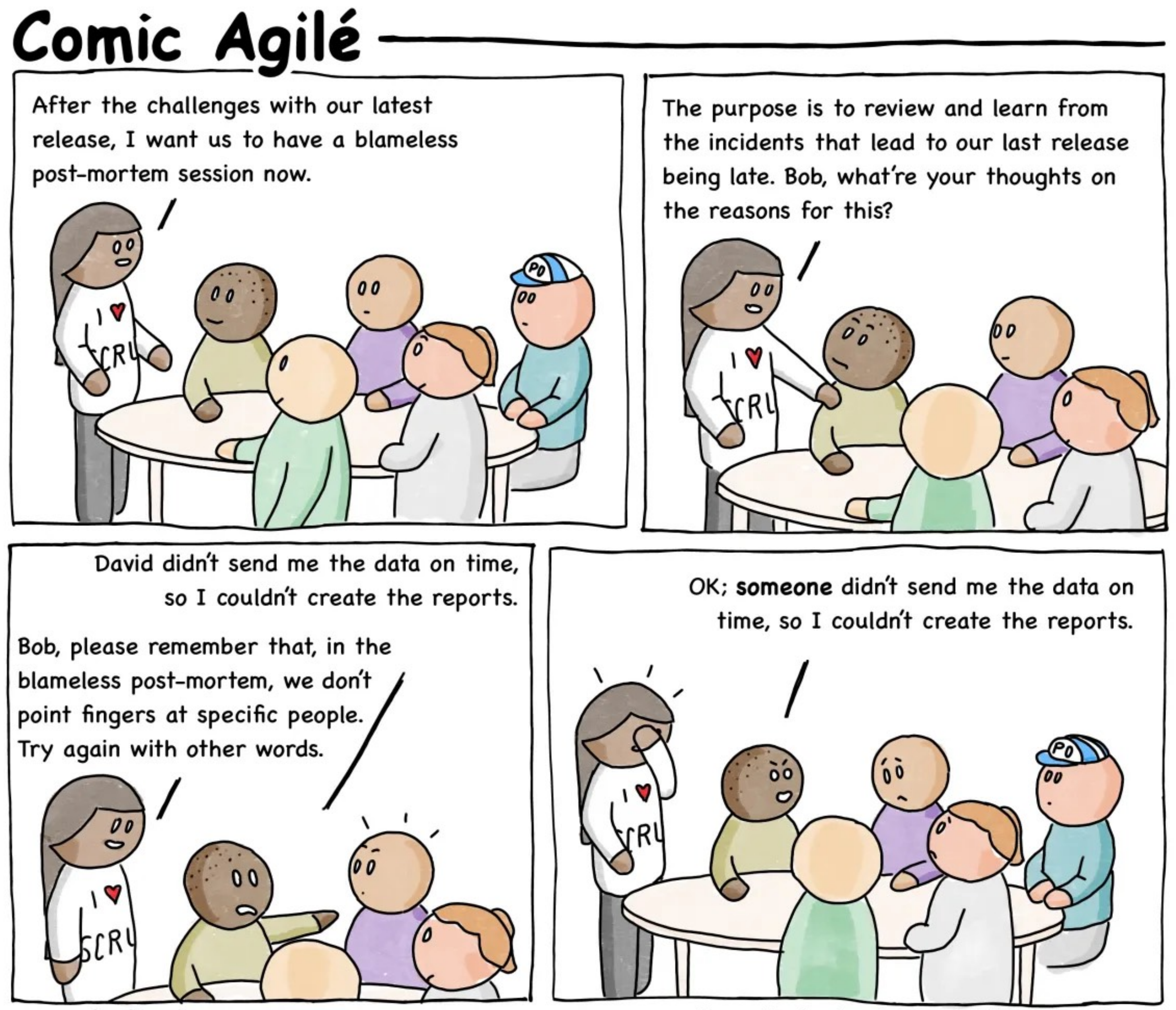


Play books
Run books
Focus on broad strategic processes
Can include info like company goals and org charts
May involve multiple teams or members
Can include automation for repetitive tasks
Includes historical information from past experiences
Must meet compliance requirements
Focus on single process
Prioritizes step-by-step instructions
Typically carried out by one team member

Blameless Postmortems

After Incident

- After resolving an incident, SRE teams conduct blameless postmortems.
- The focus is on root-cause analysis and improving processes, not assigning blame.
  - This encourages engineers to be transparent about mistakes without fear of retribution.
  - This open culture promotes learning and ensures incidents aren't repeated due to fear of sharing vital information.
  - Key Elements of a Postmortem:
    - *What Happened?* Timeline of the incident from detection to resolution.
    - *Why Did It Happen?* Root-cause analysis using tools like the 5 Whys technique.
    - *What Can We Do Better?* Suggested system changes, automation improvements, and enhanced monitoring to prevent future incidents.


Comic Agilé
After the challenges with our latest release, I want us to have a blameless post-mortem session now.
The purpose is to review and learn from the incidents that lead to our last release being late. Bob, what're your thoughts on the reasons for this?
David didn't send me the data on time, so I couldn't create the reports.
Bob, please remember that, in the blameless post-mortem, we don't point fingers at specific people. Try again with other words.
OK; **someone** didn't send me the data on time, so I couldn't create the reports.
I ♥ SCRUM
PO

www.comicagile.net

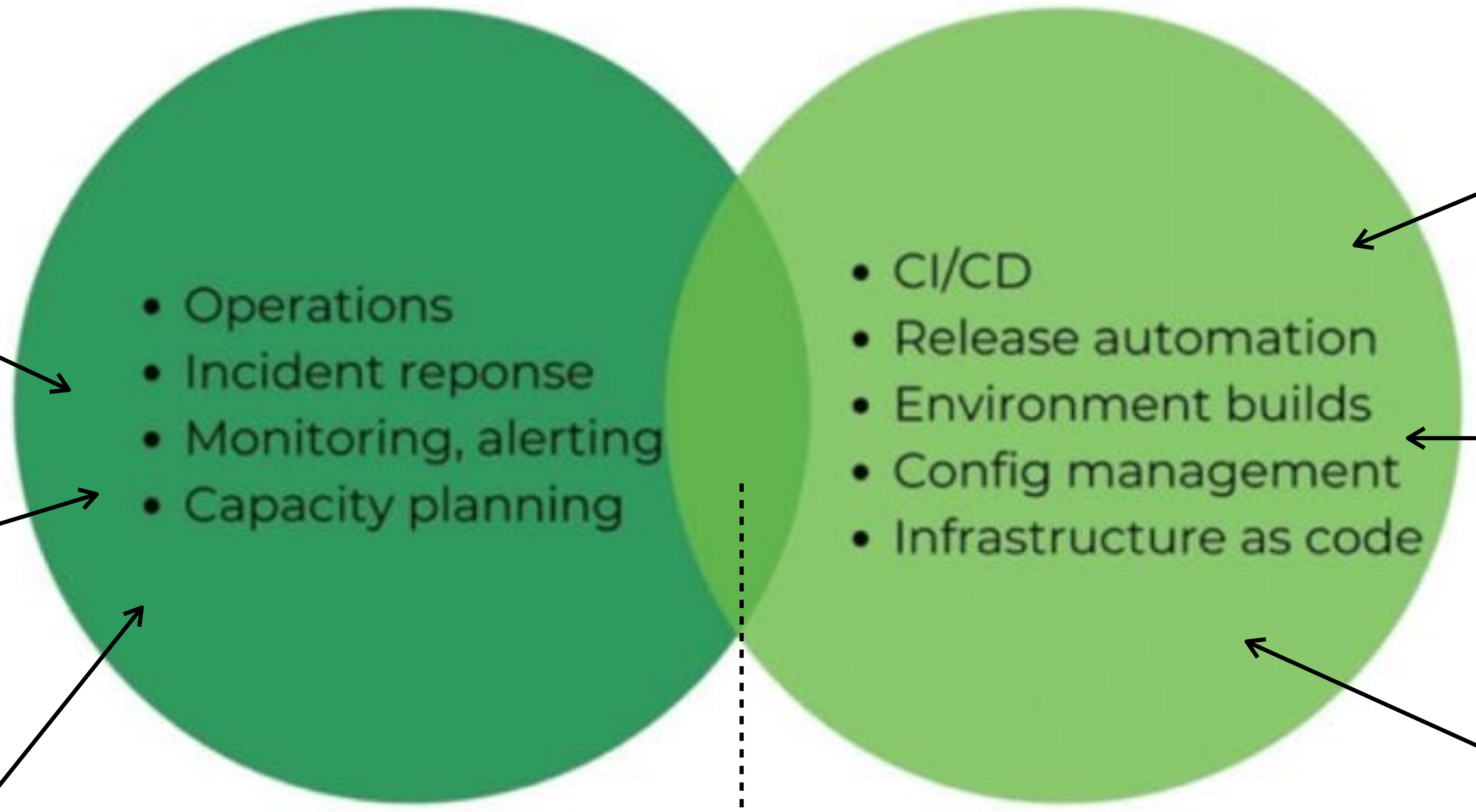
Created by Luxshan Ratnaravi & Mikkel Noe-Nygaard

SRE vs DevOps

Adds a layer of **rigorous engineering principles** to handle the complexities of large-scale systems.

It introduces specific concepts such as **error budgets**, **toil reduction**, and **service-level objectives (SLOs)** as measurable targets for reliability

More specialised role within operations that applies software engineering principles to optimize and automate operations tasks.



Build a collaborative environment between development and operations teams, fostering a **culture of shared responsibility** for the software lifecycle

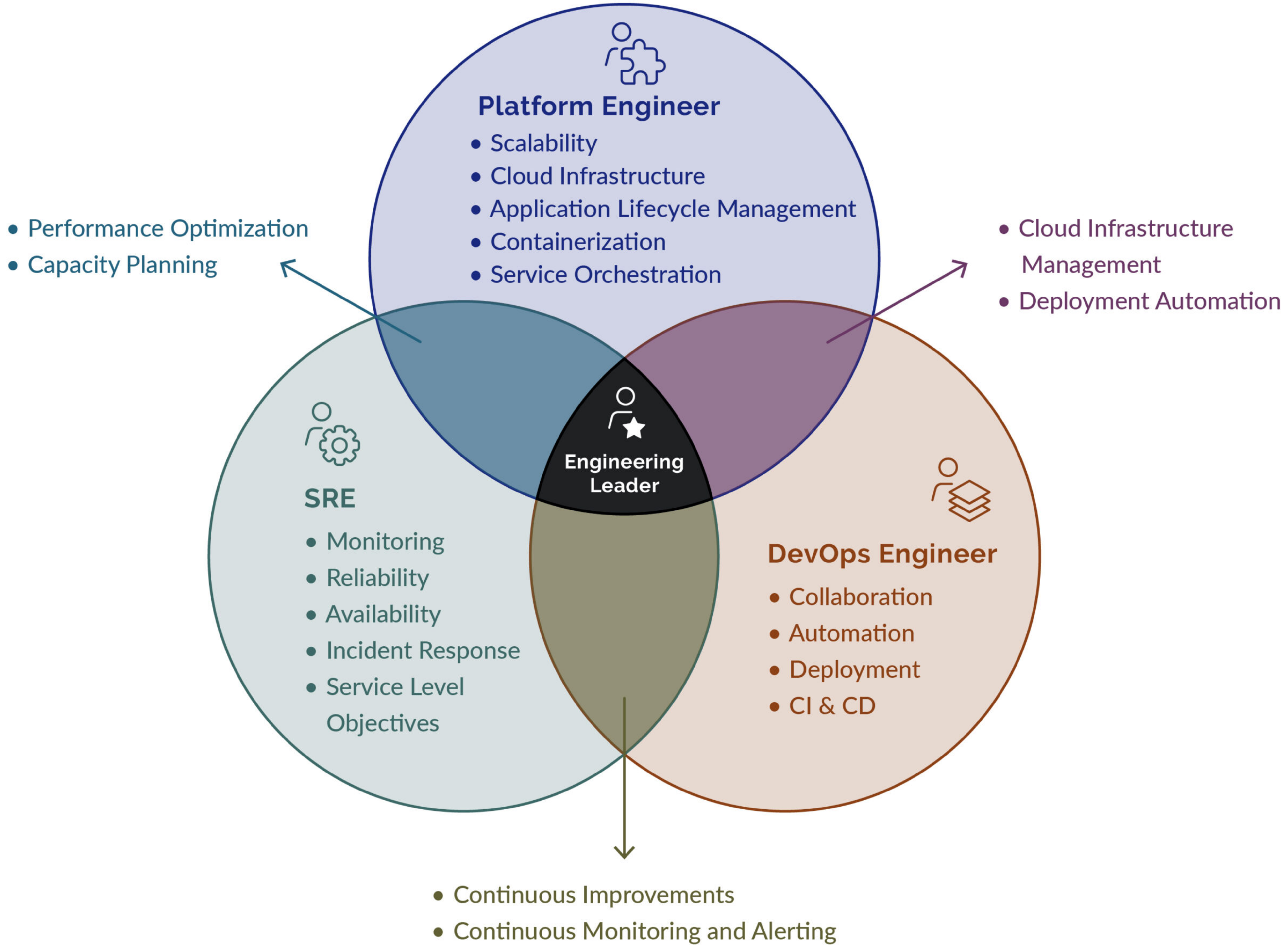
Prioritises agility and adaptability.

Emphasises practices like **CI/CD**, **Infrastructure as Code (IaC)**, and **continuous feedback loops** to streamline software delivery and operations.

- Improve collaboration between development and operations teams, accelerating delivery, and ensuring system reliability.
- The focus on **breaking down silos** and improving communication between traditionally separate teams is a central tenet of both approaches.

SRE vs DevOps vs Platform Eng

There's more …



Platform Engineer
- Scalability
- Cloud Infrastructure
- Application Lifecycle Management
- Containerization
- Service Orchestration

- Performance Optimization
- Capacity Planning

- Cloud Infrastructure Management
- Deployment Automation

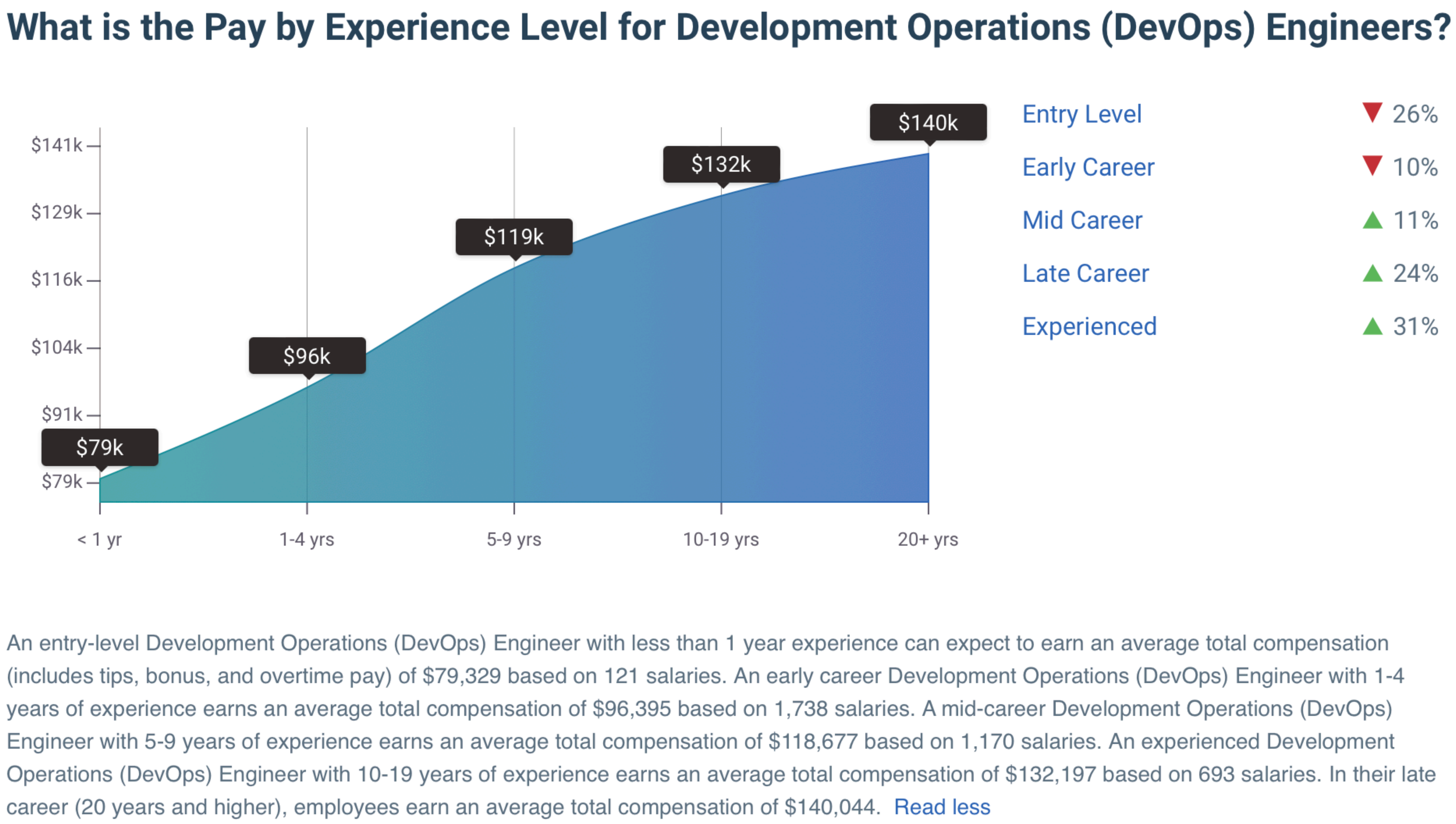
Engineering Leader

SRE
- Monitoring
- Reliability
- Availability
- Incident Response
- Service Level Objectives

DevOps Engineer
- Collaboration
- Automation
- Deployment
- CI & CD

- Continuous Improvements
- Continuous Monitoring and Alerting

Which is best?

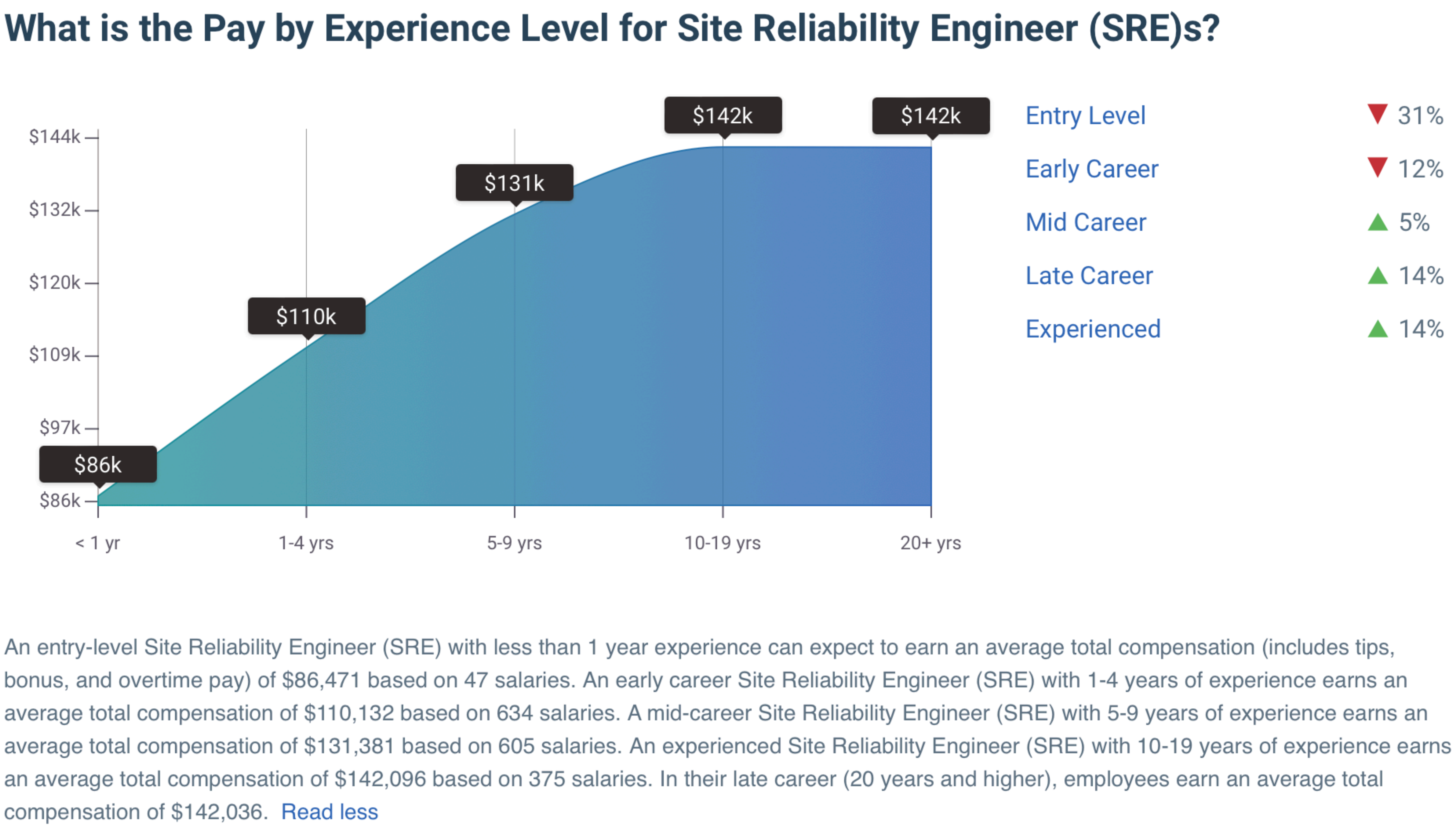


What is the Pay by Experience Level for Development Operations (DevOps) Engineers?

Entry Level	▼ 26%
Early Career	▼ 10%
Mid Career	▲ 11%
Late Career	▲ 24%
Experienced	▲ 31%

An entry-level Development Operations (DevOps) Engineer with less than 1 year experience can expect to earn an average total compensation (includes tips, bonus, and overtime pay) of $79,329 based on 121 salaries. An early career Development Operations (DevOps) Engineer with 1-4 years of experience earns an average total compensation of $96,395 based on 1,738 salaries. A mid-career Development Operations (DevOps) Engineer with 5-9 years of experience earns an average total compensation of $118,677 based on 1,170 salaries. An experienced Development Operations (DevOps) Engineer with 10-19 years of experience earns an average total compensation of $132,197 based on 693 salaries. In their late career (20 years and higher), employees earn an average total compensation of $140,044. Read less

Which is best?

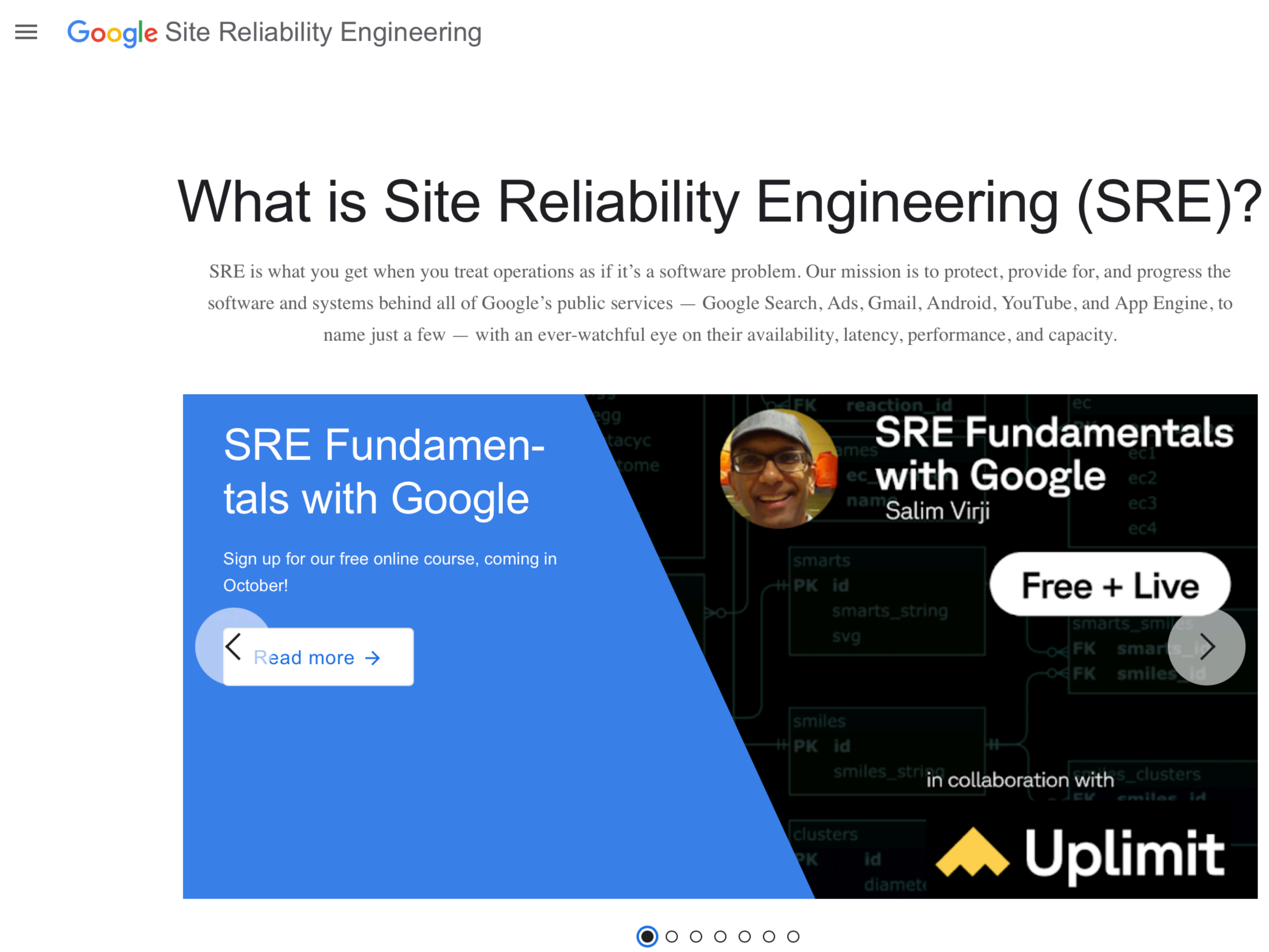


What is the Pay by Experience Level for Site Reliability Engineer (SRE)s?

Entry Level	▼ 31%
Early Career	▼ 12%
Mid Career	▲ 5%
Late Career	▲ 14%
Experienced	▲ 14%

An entry-level Site Reliability Engineer (SRE) with less than 1 year experience can expect to earn an average total compensation (includes tips, bonus, and overtime pay) of $86,471 based on 47 salaries. An early career Site Reliability Engineer (SRE) with 1-4 years of experience earns an average total compensation of $110,132 based on 634 salaries. A mid-career Site Reliability Engineer (SRE) with 5-9 years of experience earns an average total compensation of $131,381 based on 605 salaries. An experienced Site Reliability Engineer (SRE) with 10-19 years of experience earns an average total compensation of $142,096 based on 375 salaries. In their late career (20 years and higher), employees earn an average total compensation of $142,036. Read less

Google - SRE



Google Site Reliability Engineering

What is Site Reliability Engineering (SRE)?

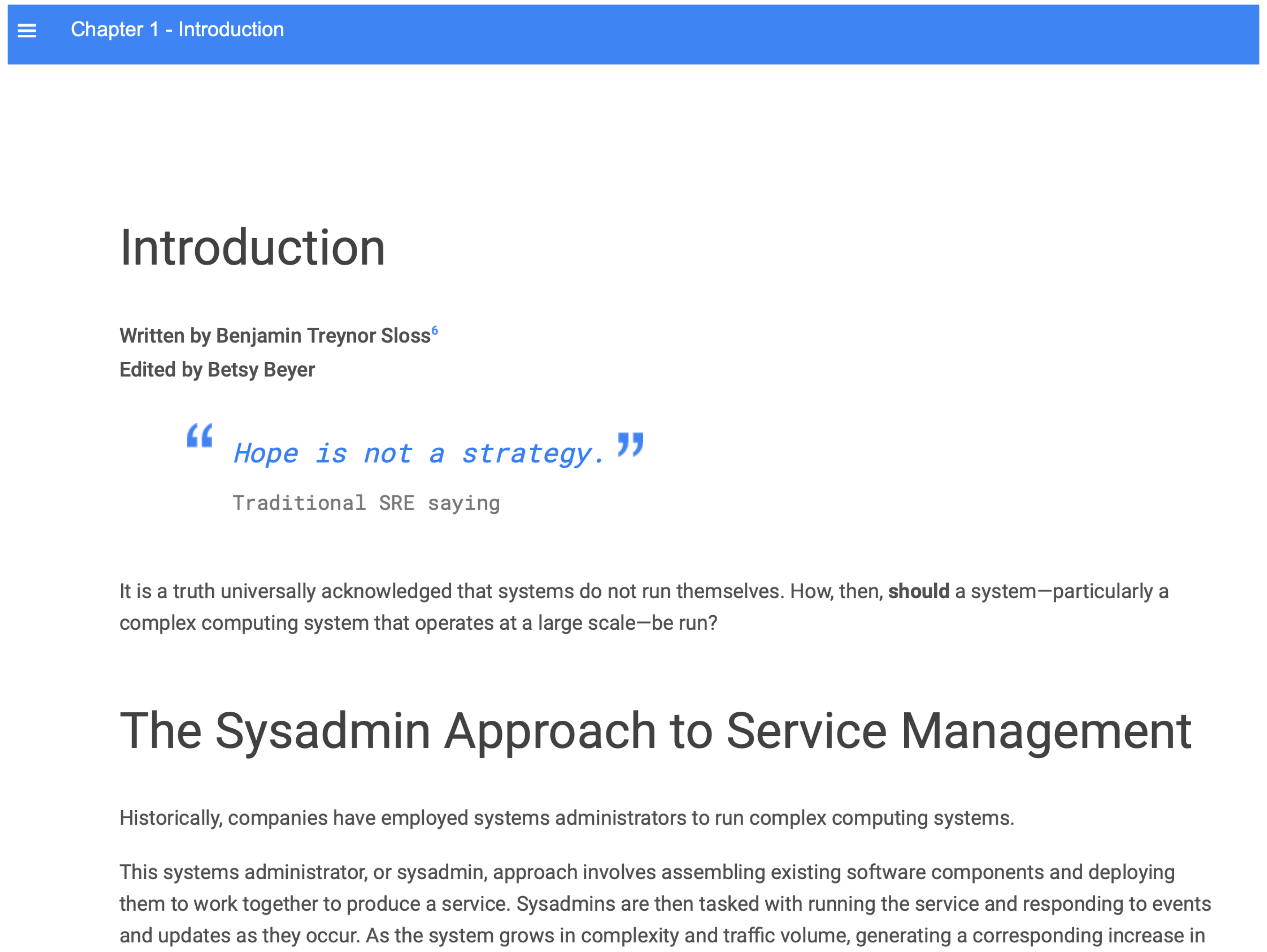
SRE is what you get when you treat operations as if it's a software problem. Our mission is to protect, provide for, and progress the software and systems behind all of Google's public services — Google Search, Ads, Gmail, Android, YouTube, and App Engine, to name just a few — with an ever-watchful eye on their availability, latency, performance, and capacity.

SRE Fundamentals with Google

Sign up for our free online course, coming in October!

Read more →

https://sre.google/



Chapter 1 - Introduction

Introduction

Written by Benjamin Treynor Sloss[6]
Edited by Betsy Beyer

> *Hope is not a strategy.*
>
> Traditional SRE saying

It is a truth universally acknowledged that systems do not run themselves. How, then, **should** a system—particularly a complex computing system that operates at a large scale—be run?

The Sysadmin Approach to Service Management

Historically, companies have employed systems administrators to run complex computing systems.

This systems administrator, or sysadmin, approach involves assembling existing software components and deploying them to work together to produce a service. Sysadmins are then tasked with running the service and responding to events and updates as they occur. As the system grows in complexity and traffic volume, generating a corresponding increase in

https://sre.google/sre-book/introduction/