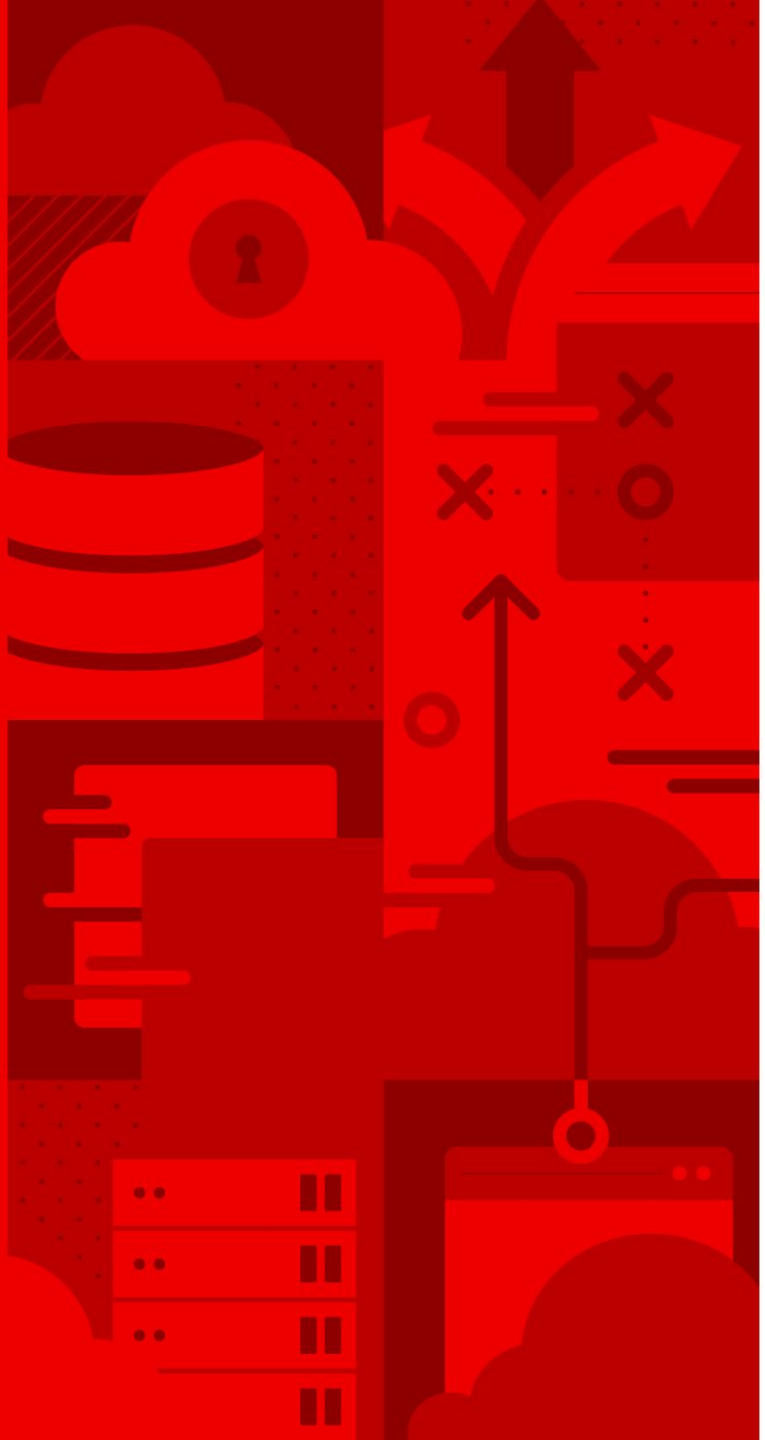


Cloud Native Operating Models

Andrew Clay Shafer
@littleidea

Red Hat

Andrew Clay Shafer

Old

- Manage servers
- Minimize incidents
- Long lived infra
- Manual checklists
- Support bau

New

- Manage services
- Minimize mttr
- Ephemeral infra
- API enforced
- Enable innovation

What is DevOps?
What is SRE?

"Optimizing the human experience and performance of operating software...
with software...
and humans"

Andrew Clay Shafer
Vice President, Global Transformation Office, Red Hat

"What happens when a software **engineer** is tasked with what used to be called operations"

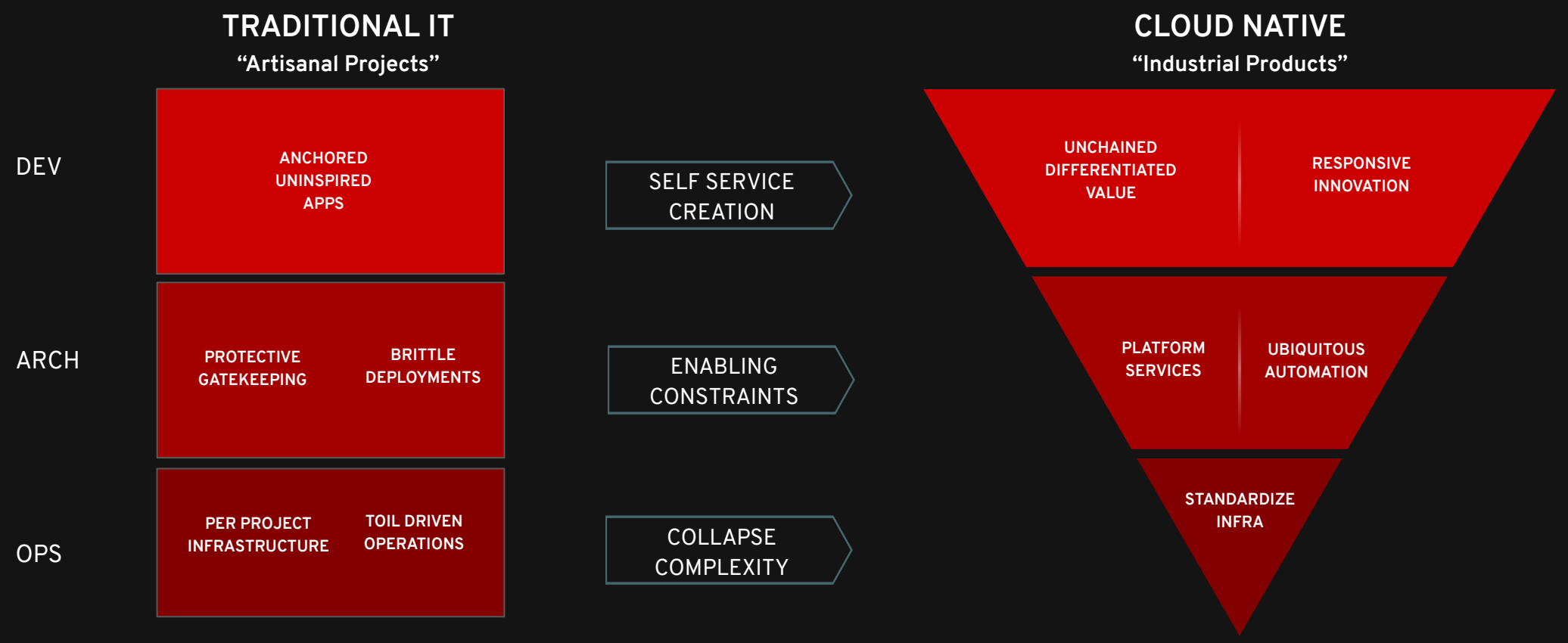
Benjamin Treynor Sloss
Vice President, Engineering, Google

But why???

The new models evolved due to pressure to deliver adaptable services at scale.

- Platform Patterns
- You Build It, You Run It
- SRE in Theory and Practice

The Cloud Native Organization



Netflix, Amazon, Google, and every ‘cloud native’ company built a platform

Because they had to…

Netflix Lessons

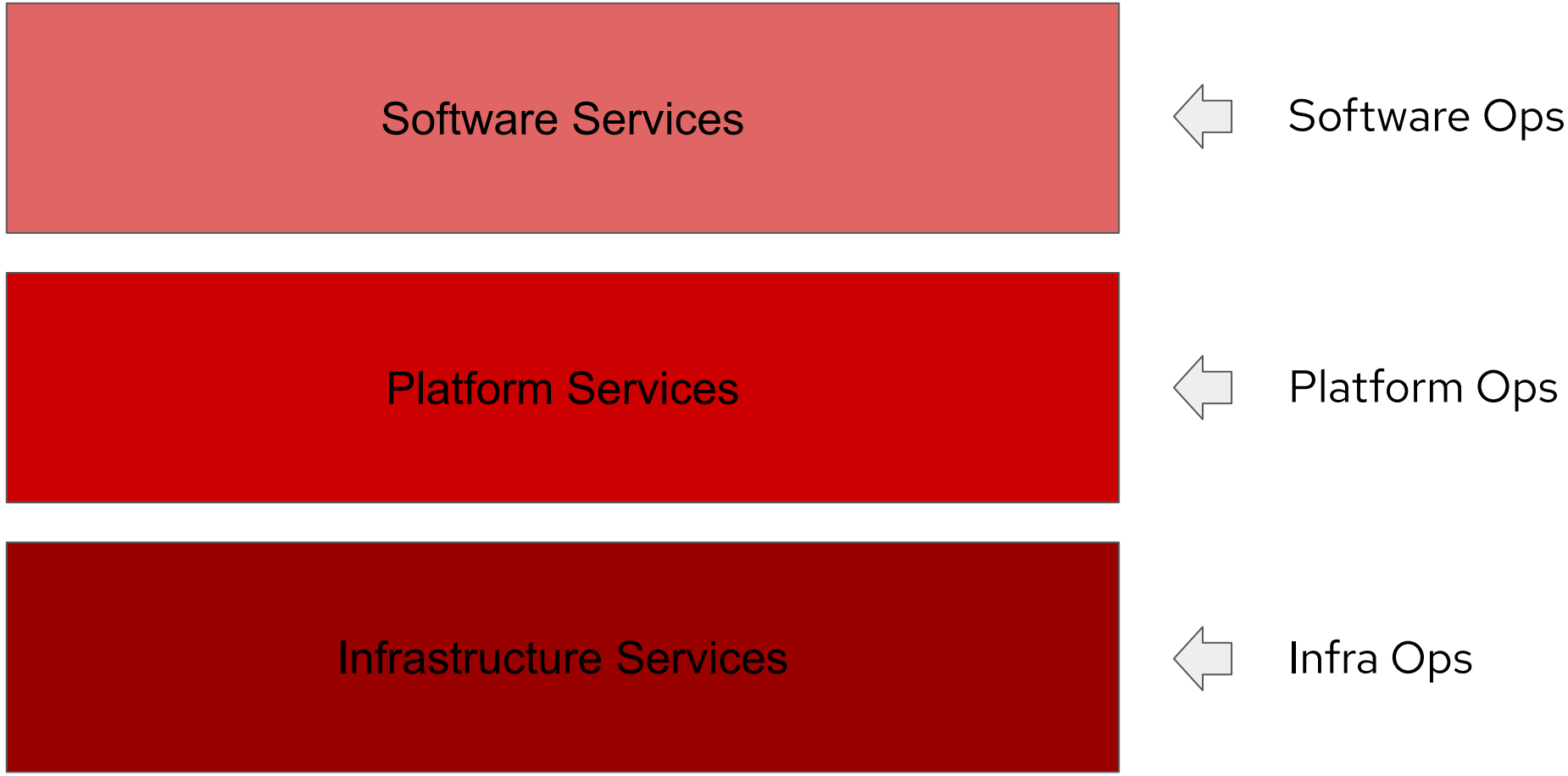
- Remove friction from product development
- High trust, low process, no hand-offs between teams
- Don’t do your own undifferentiated heavy lifting
- Use simple patterns automated by tooling
- Self Service cloud makes impossible things instant

‘Cloud’ evolved from lessons learned building and operating these internal services

Software Services ⇦ Needs to be Operated

Platform Services ⇦ Needs to be Operated

Infrastructure Services ⇦ Needs to be Operated


Software Services
Software Ops
Platform Services
Platform Ops
Infrastructure Services
Infra Ops

What is ‘Operations’?

Metrics

Alerting

Monitoring

Automation

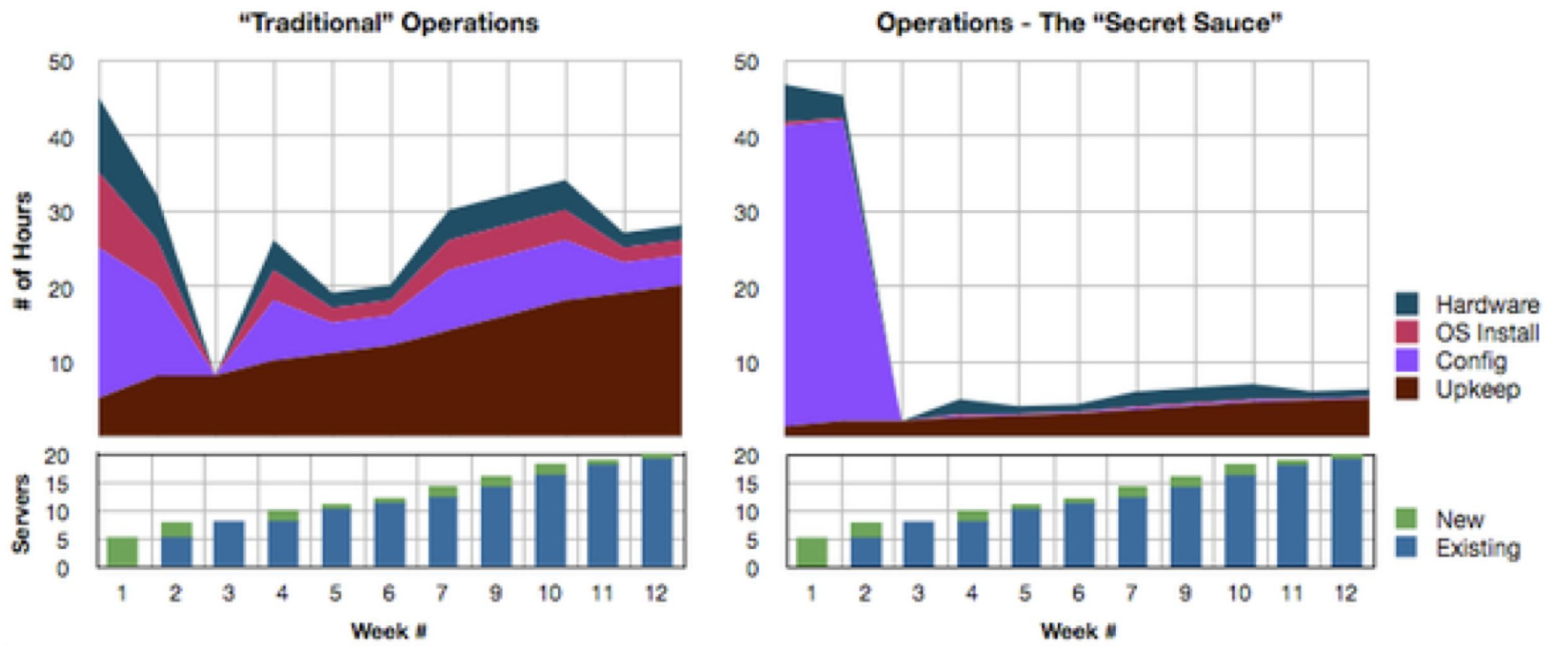
Troubleshooting

Incident Management

Post-Incident Analysis

Capacity Planning

Operations is the 'Secret Sauce'



(http://radar.oreilly.com/archives/2007/10/operations-advantage.html)

“The traditional model is that you take your software to the wall that separates development and operations, and throw it over and then forget about it. Not at Amazon. You build it, you run it. This brings developers into contact with the day-to-day operation of their software. It also brings them into day-to-day contact with the customer. This customer feedback loop is essential for improving the quality of the service.”

–Werner Vogels, CTO Amazon

2006

Three years before devops is a word.

devops - calms

- culture
- automation
- lean
- metrics
- sharing

Damon Edwards and John Willis, plus Jez Humble

Software Services

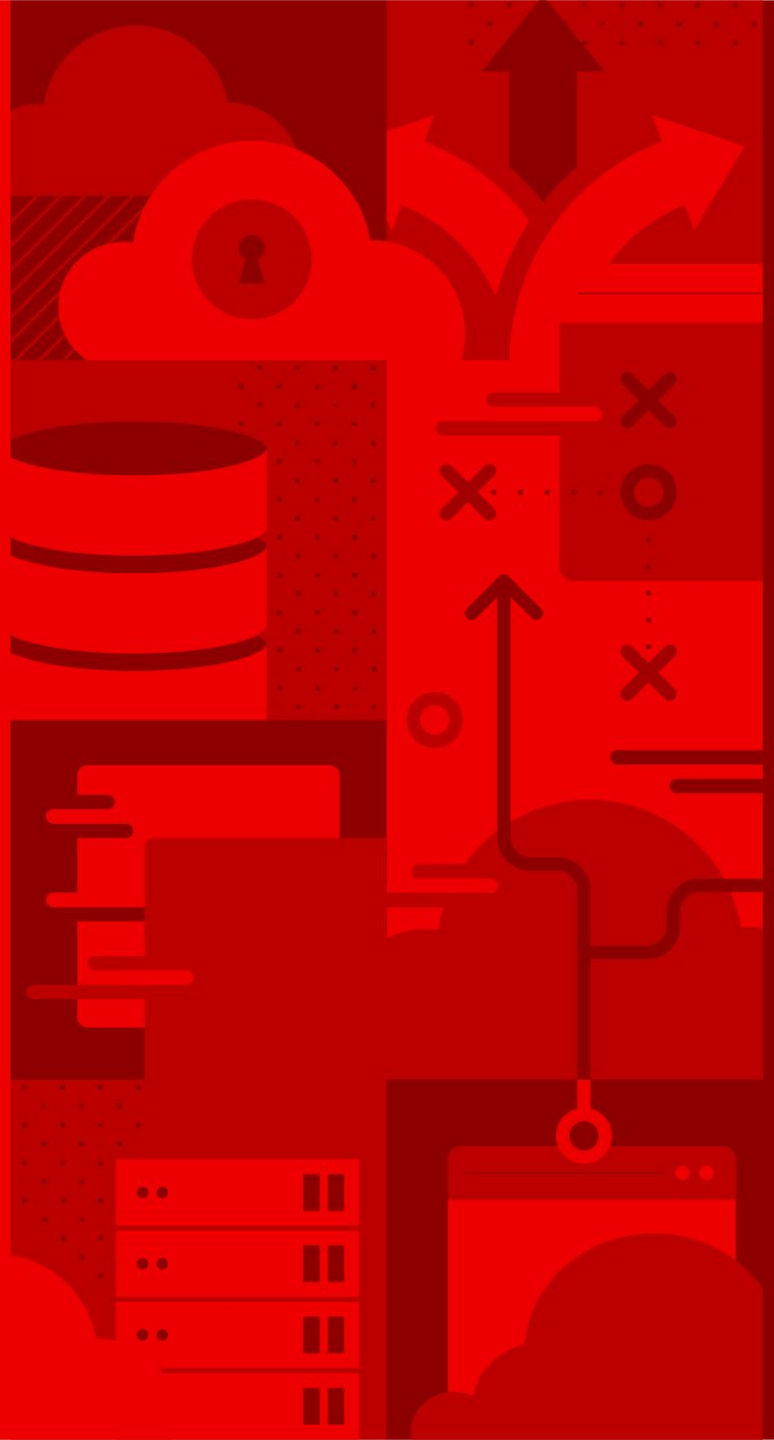
Werner meant run ‘this’

Platform Services

Not this

Infrastructure Services

Not this



Enter Google SRE

2016, after 10 years

Google’s devops implementation

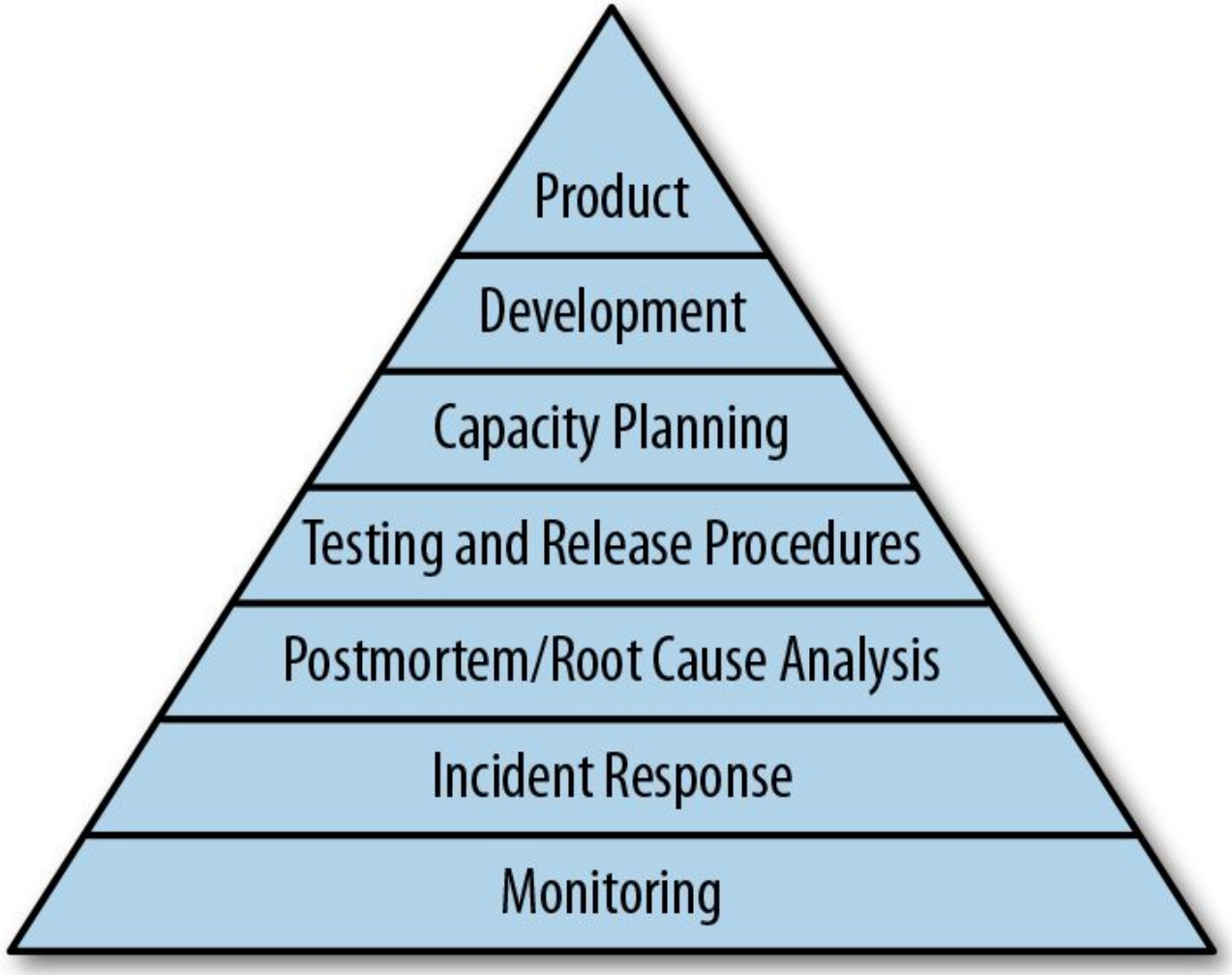
SRE - calms

- ✓ culture
- ✓ automation
- ✓ lean
- ✓ metrics
- ✓ sharing

good devops copy

great devops steal

Foundational Monitoring



Dickerson's "Hierarchy of Reliability"

Almost every task run under Borg contains a built-in HTTP server that publishes information about the health of the task and thousands of performance metrics

Service Level Terminology

Describe the metrics that matter, the values we want, and how we will react

Indicators

Defined measurement of an aspect of a service.

Objectives

Target value (or range of values) as measured by an SLI

Agreements

Explicit or implicit contract with users or customers, with consequences of meeting or missing objectives

Categories of services

Generalization of SLIs based on service type

User-facing systems
Availability, latency, throughput

Storage systems
Latency, availability, durability

Data systems
Throughput, end-to-end latency

Common indicators
Correctness

The Four Golden Signals

This is not comprehensive, and can be controversial, but this is a good place to start.

These are four areas to focus on for user-facing systems

Latency

How long does it take to service a request? Should split between successful request and latency and failed request latency as separate conditions.

Traffic

How much demand is being placed on the system? Requests/second, for example. When traffic drops outside of predicted norms, can illustrate other issues.

Errors

What is the rate of failed requests? Error codes may not be sufficient to capture this.

Saturation

How overloaded is the system? Systems may begin to degrade prior to 100% utilization. May also reflect predicted saturation that is impending.

Defining objectives

SLOs should be specific on how they are measured and what is an acceptable condition

99% (averaged over 1 minute) of Get RPC calls will complete in less than 100 ms (measured across all backend servers)

99% of Get RPC calls will complete in less than 100 ms

Choosing your targets

SLO targets are not purely an SRE decision - there are product and business implications.

Setting SLOs is a team sport.

Don't use current performance as an objective

Simply adopting current values without investigation and consideration may create an unsupportable setup

Keep it simple

Avoid overly complex aggregations of SLIs

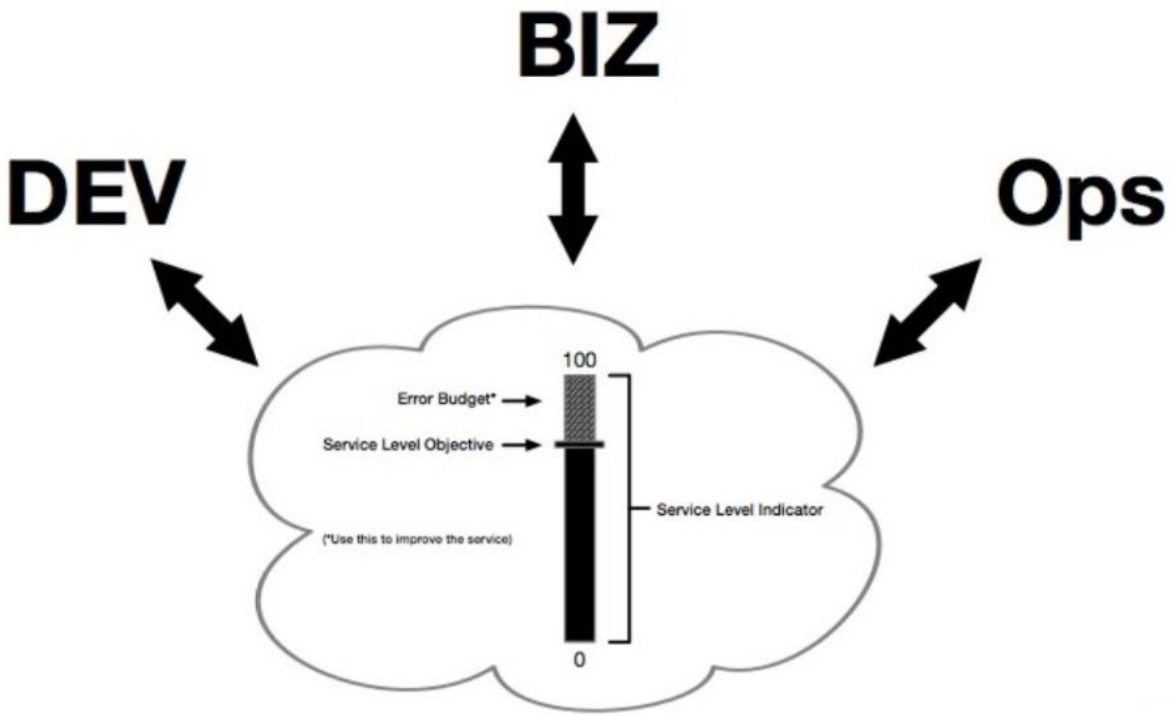
Have as few SLOs as possible

Choose "just enough" for good coverage of the attributes of your system. Be able to defend the SLOs you choose.

Progress over perfection

SLOs are constantly being refined and adjusted as system and user behavior is better understood. Avoid "analysis paralysis"

- SLO is 3 way contract



100% reliability is *unrealistic*

Error Budgets

An acceptable level of unreliability

It's a budget. It can be allocated.

What are the consequences?

What happens when we exhaust or overspend our error budget?

Freeze feature releases

Until the error budget is recovered, no new features can be released

Prioritize post mortem items

Action items from post-incident reviews are set higher in the team's work stream

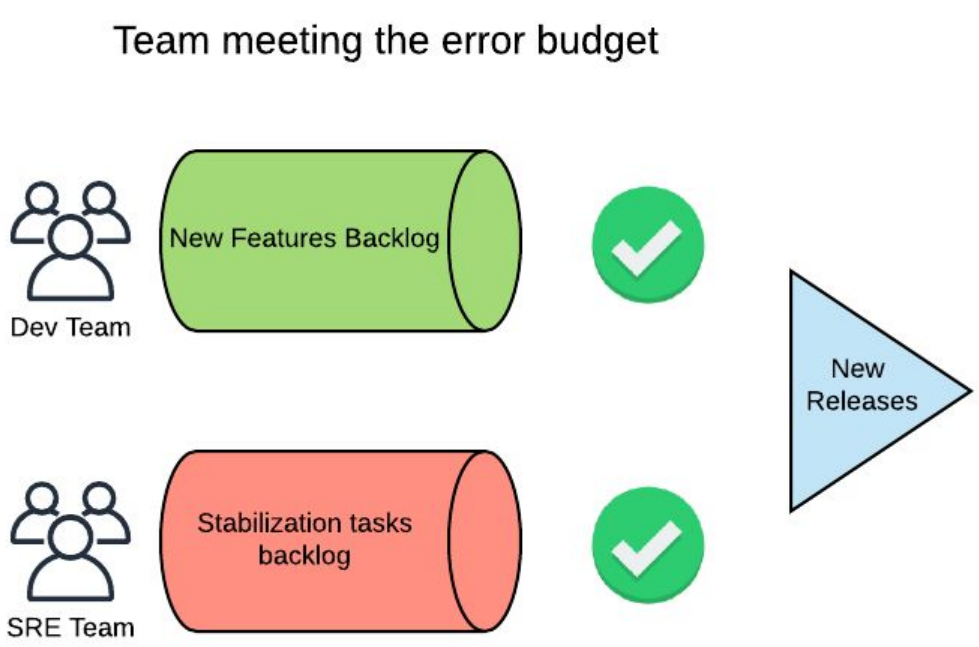
Improve monitoring and observability

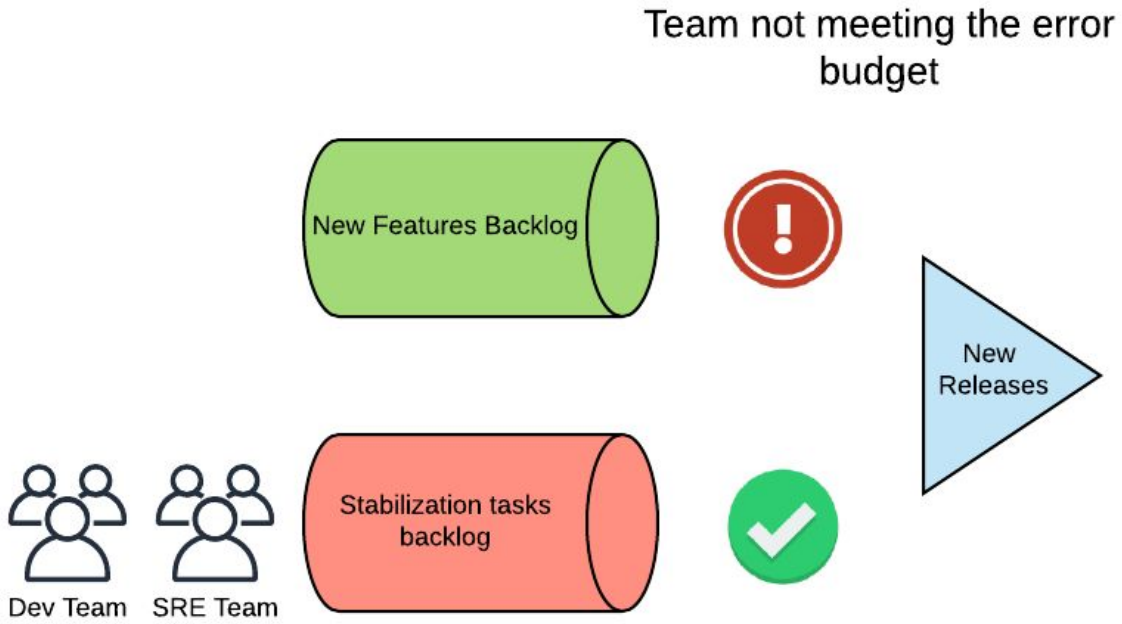
It may be required to add additional monitors or alerts on SLOs to enable more proactive response

Error Budget and Release Management

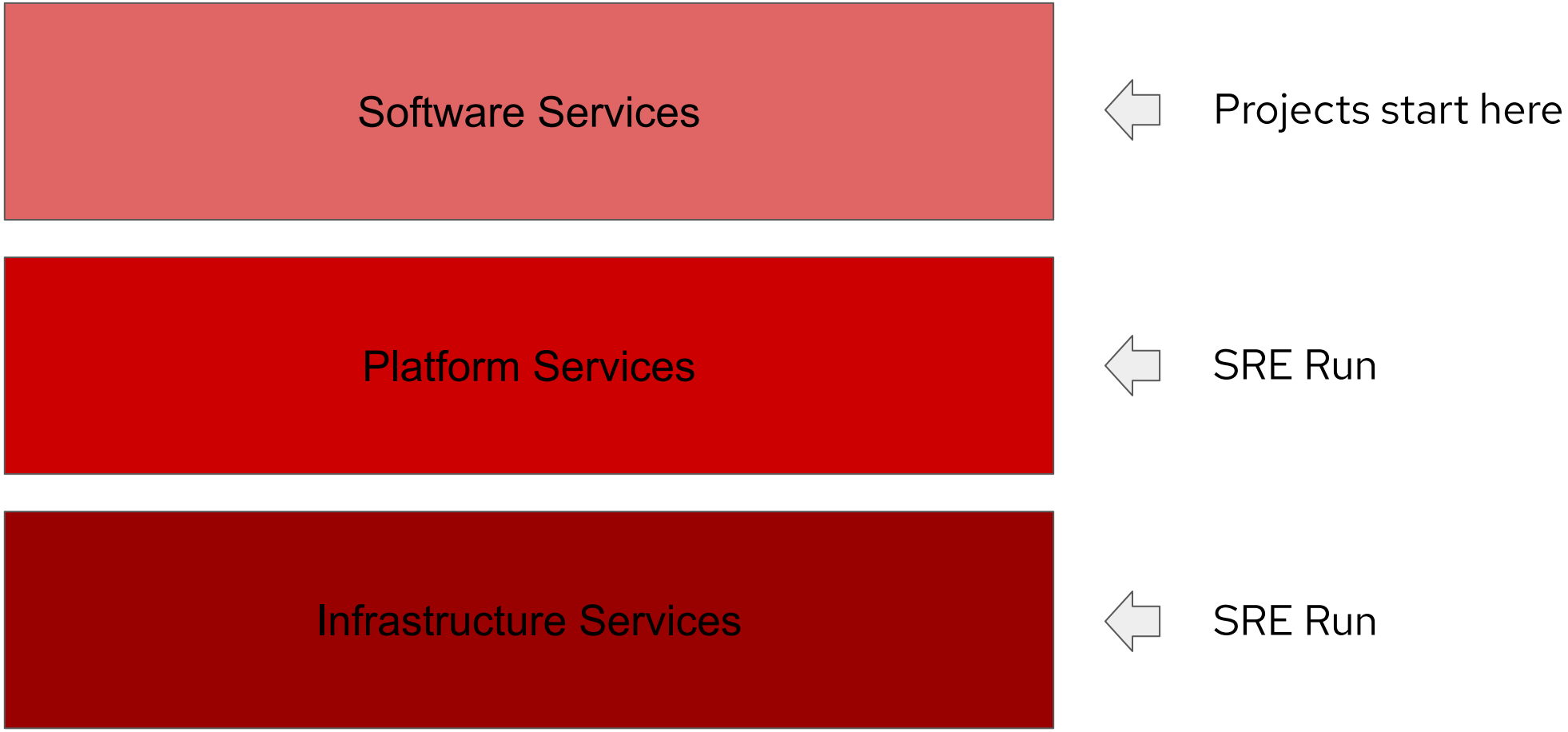
Rule: If a team met the error budget they can release new features, if not they have to work on stabilization.

Brilliant self-regulating mechanism to fix the age-old debate between devs and ops on the speed of change.

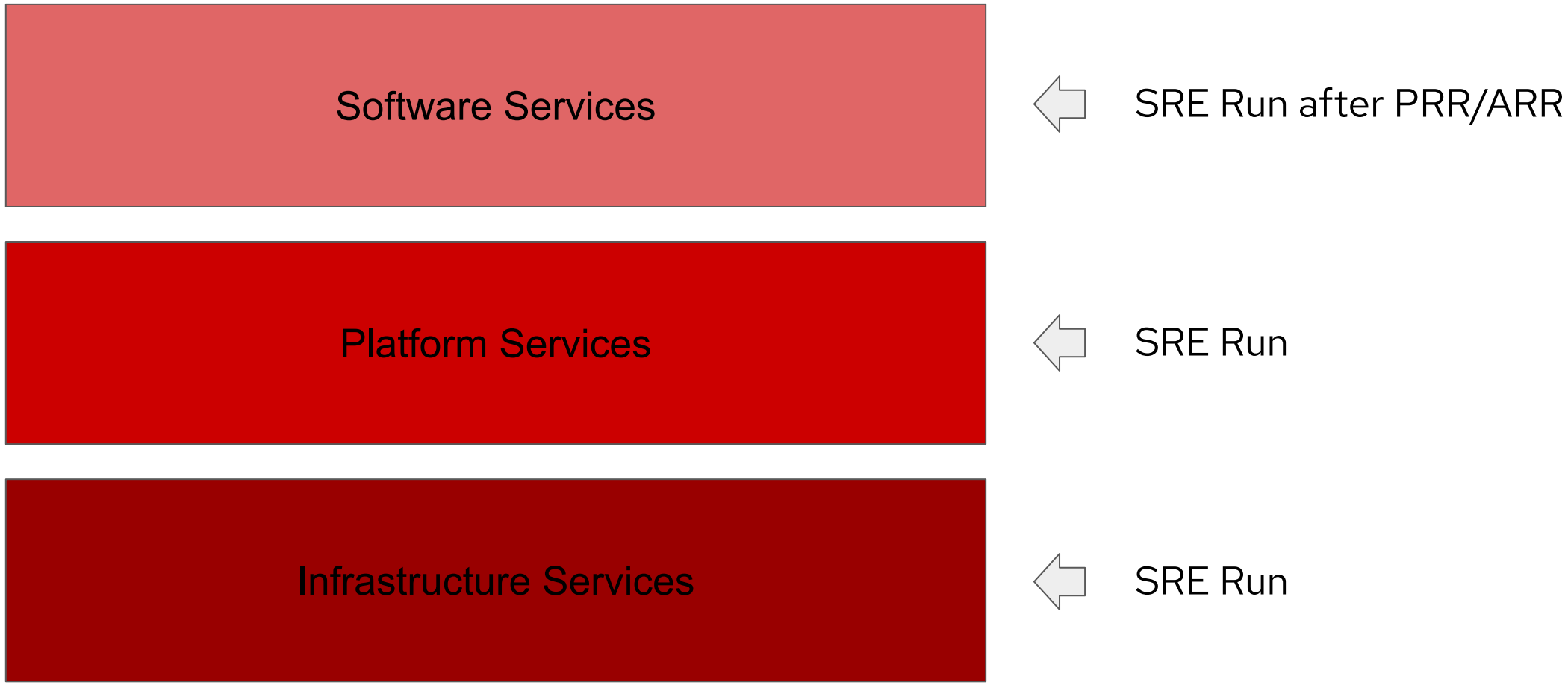




Not all projects get SRE at Google



Projects Earn SRE



SRE is not just there to take toil away from SWE

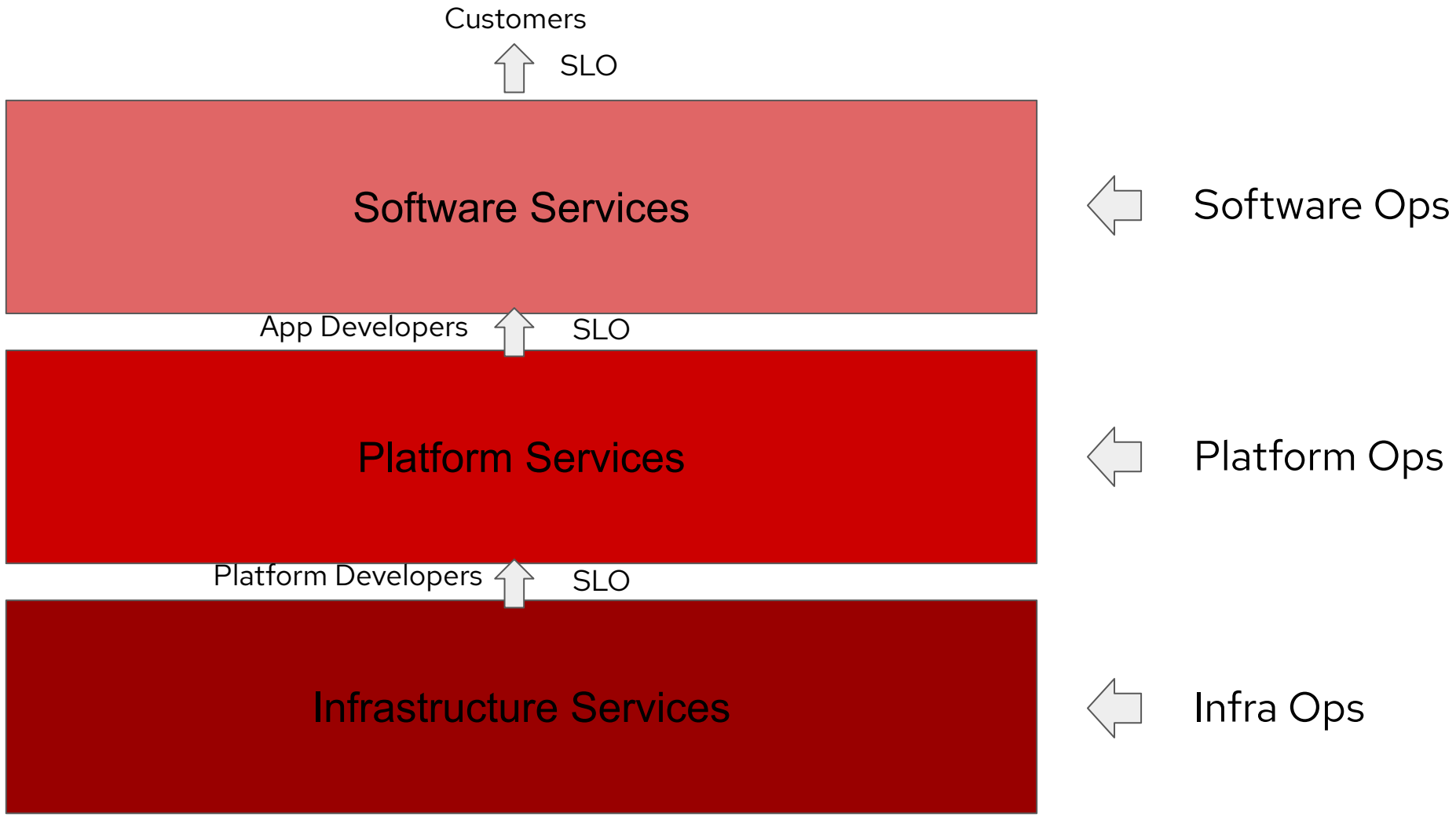
but to drive toil out of the system

SRE are effectively architects

(and product managers)

SRE builds framework modules to implement canonical solutions for the concerned production area. As a result, development teams can focus on the business logic, because the framework already takes care of correct infrastructure use.

Be deliberate and explicit
who, what, when, where, why, how



Adoption is a Continuum

Traditional Ops

- Toil is not measured
- Engagement is via ticketing systems
- Metrics/measurement not tied to SLOs

Getting Started

- Some initial SLOs defined
- Blameless postmortems introduced
- Incident response process exists

Collaboration Begins

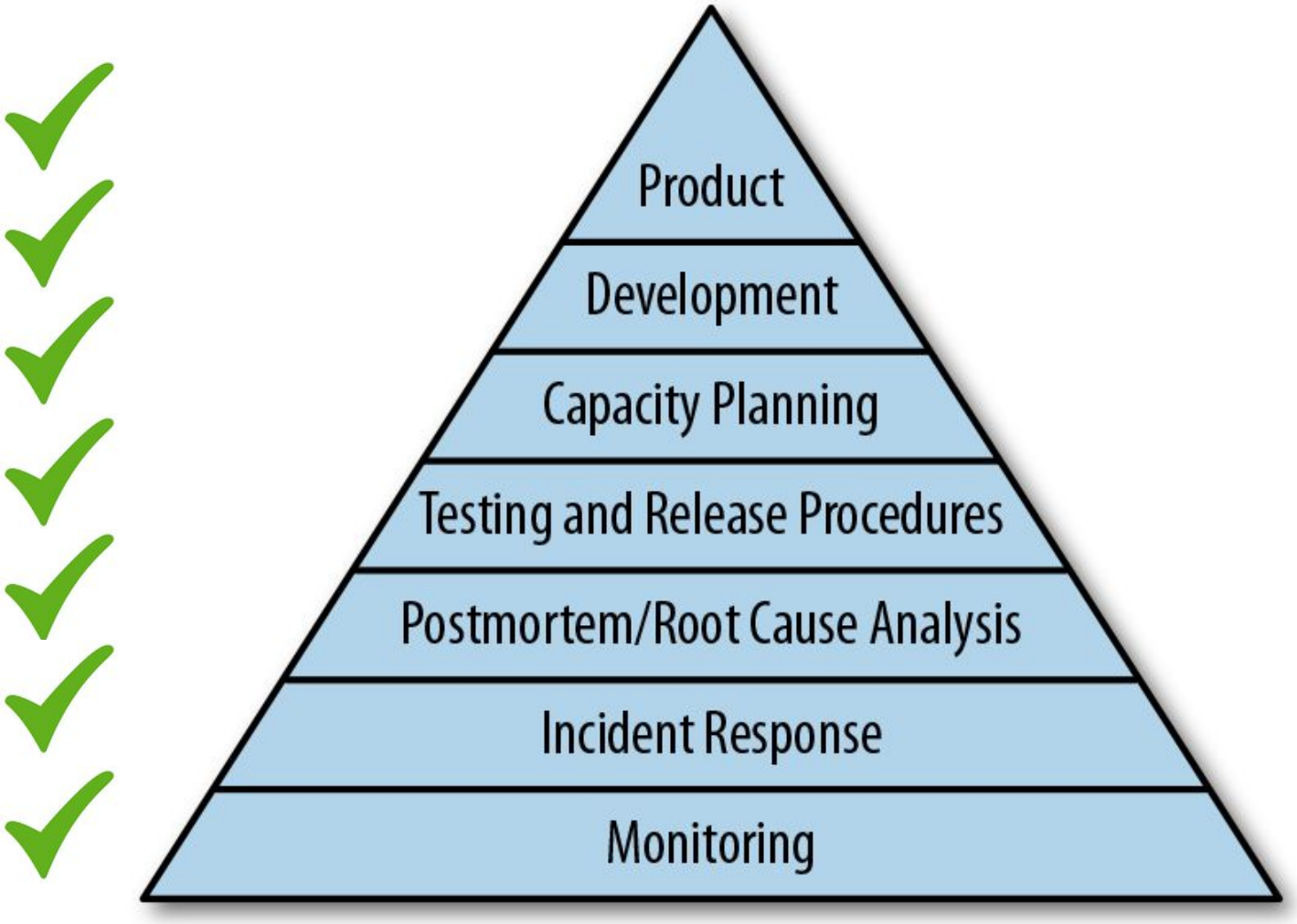
- Release process is documented and automated, including appropriate rollbacks and canaries
- SRE team plans and executes project work
- SRE team charter has been reviewed by appropriate leadership
- Project planning and execution is done jointly by developers and operations

Understanding More

- Periodic review of SRE project work and impact with business leaders
- Periodic review of SLIs and SLOs with business leaders
- Escalation policy tied to SLO violations (error budgets, etc)
- Toil is measured

Advanced

- A reasonable goal for amount of toil is set and achieved (generally no more than 50%)
- Most service alerts are based on SLOs
- SRE team can identify major positive impact on a business impact beyond firefighting/operational concerns



Dickerson's "Hierarchy of Reliability"

What is DevOps?
What is SRE?

Can you put code in prod?
And keep it running?

Who cares?
What works?
What works for you?

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat