

is this you?



SRE in Enterprise

Site Reliability Engineering

<https://g.co/cloud/ent-sre>



DISCLAIMER:

These are our own personal opinions,
not the opinions of our employer

The book is official though

Agenda

Learn about the challenges of adopting site reliability engineering (SRE) in enterprises, and how we recommend cloud customers go about this journey

- Adoption of SRE best practices by cloud customers through evaluating their **existing** environment and architecture
- Identify how SRE guiding **principles** fit into a cloud customers existing organization (e.g. how to embrace risk)
- Adapt SRE **practices** for cloud customers existing team structure and knowledge
- **Nurture** a successful SRE initiative outside of Google

Intros



@JamesBrookbank

linkedin.com/in/jamesbrookbank



James Brookbank is a cloud solutions architect. Solution architects help make cloud easier for Google's customers by solving complex technical problems and providing expert architectural guidance. Before joining Google, James worked at a number of large enterprises with a focus on IT infrastructure and financial services.



@stevemcghee

linkedin.com/in/stevemcghee



Steve was an SRE at Google for about 10 years in Android, YouTube and Cloud. He then joined a company to help them move onto the Cloud.

Now he's back at Google as a Reliability Advocate, helping more companies do that.

The tl;dr

What are we seeing with SRE in the Enterprise?

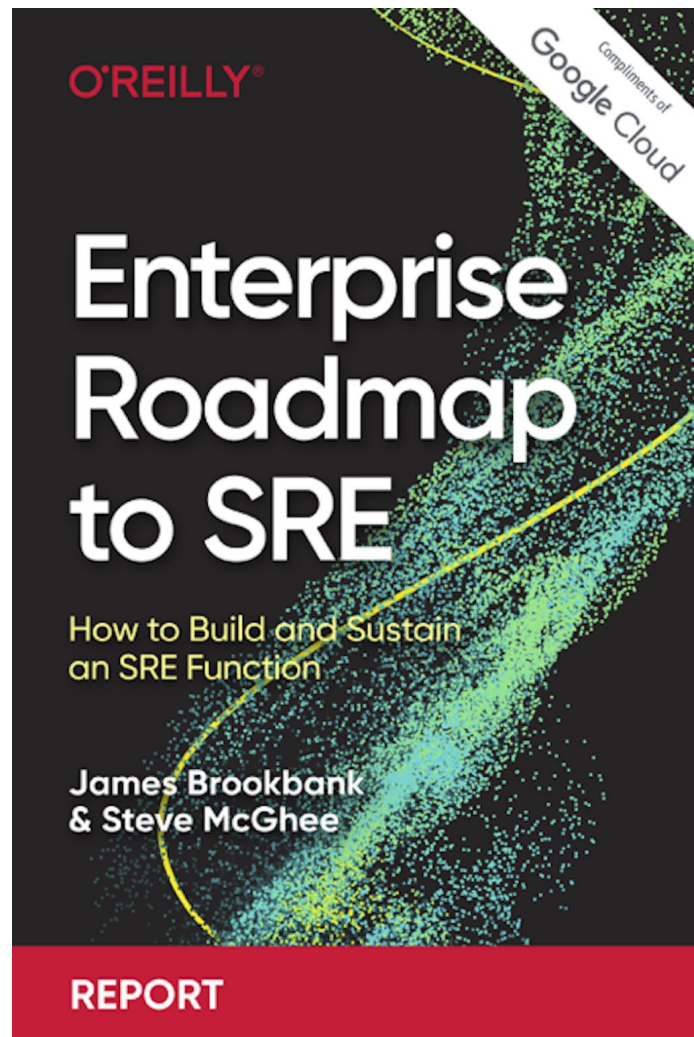
"It's only SRE if it comes from the Mountain View region in California, otherwise it's just sparkling operations"

- **Enthusiasm > Successful Adoption** of SRE
- Reliability **isn't the most important thing** for everything
- SRE is often seen as **expensive** or **difficult** to achieve (usually both)
- Not everyone wants *the Google SRE way*
 - but they usually still want something that is **better than today**



Enterprise Roadmap to SRE

Copies are available!
~~Here!~~ Soon! ~~Now!~~ Lunch!

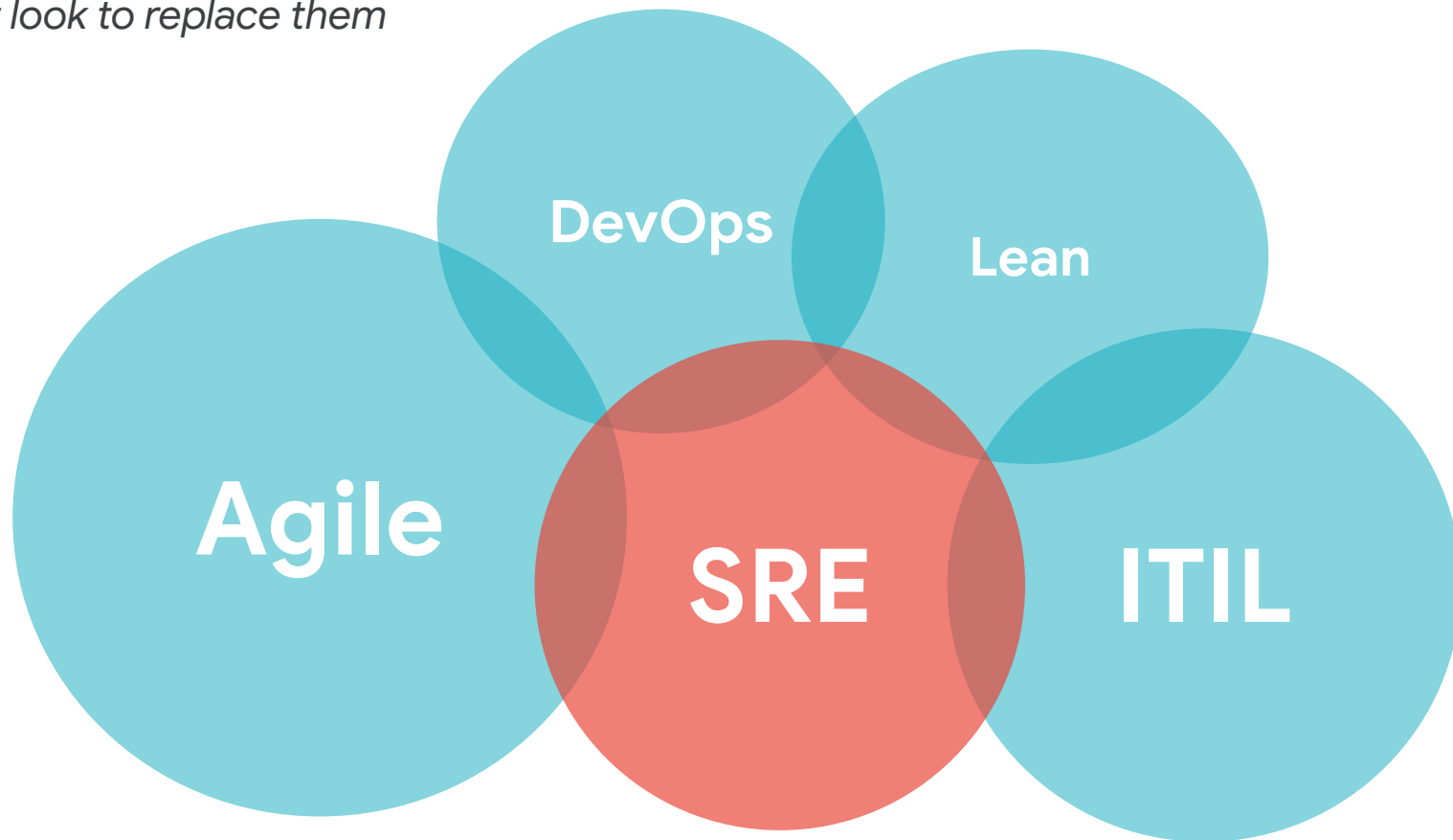


Section 1

Getting started with Enterprise SRE

SRE overlaps with many other things

It doesn't look to replace them



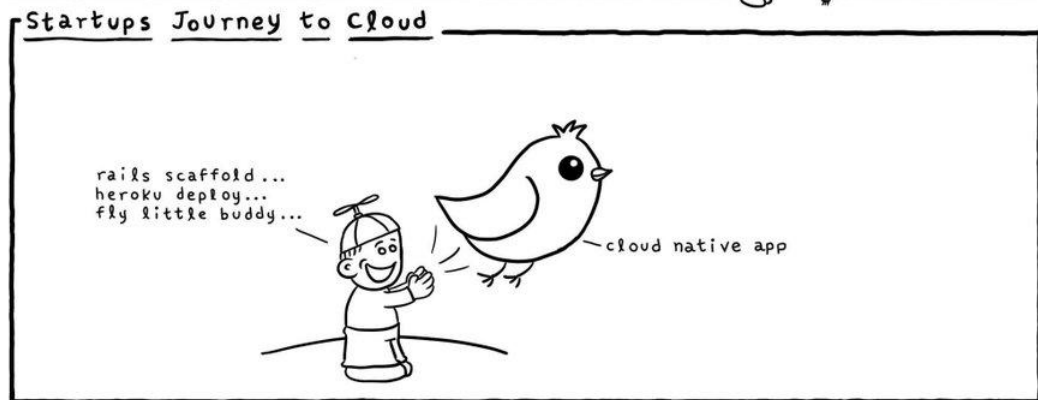
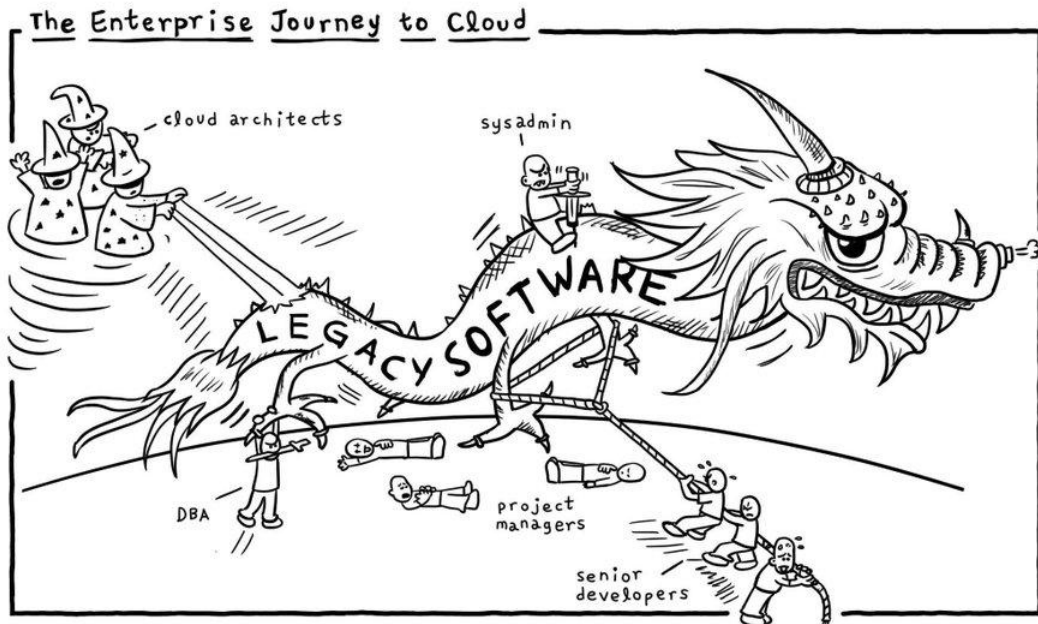
Sticking points

CABs
NOCs
...etc.

these individual practices aren't faulty on their own.

it's the **centralization** and **top-down organization** that doesn't work @ scale.

Getting Started with Cloud



DevOps!



Capabilities

Technical

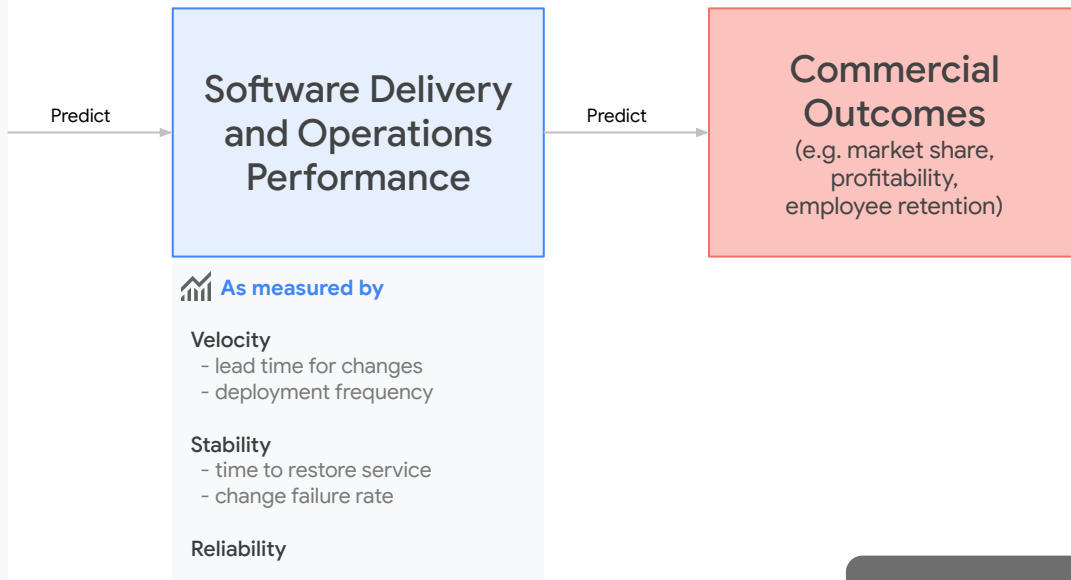
- Trunk-based development
- Cloud infrastructure
- Shifting left on security
- ...

Process

- Work in small batches
- Streamlined change approval
- Visibility of work in value stream
- ...

Cultural

- Generative, trust-based
- Learning culture
- Transformational leadership
- ...



g.co/devops

Reliability is a **force multiplier**

Teams that excel at reliability engineering are

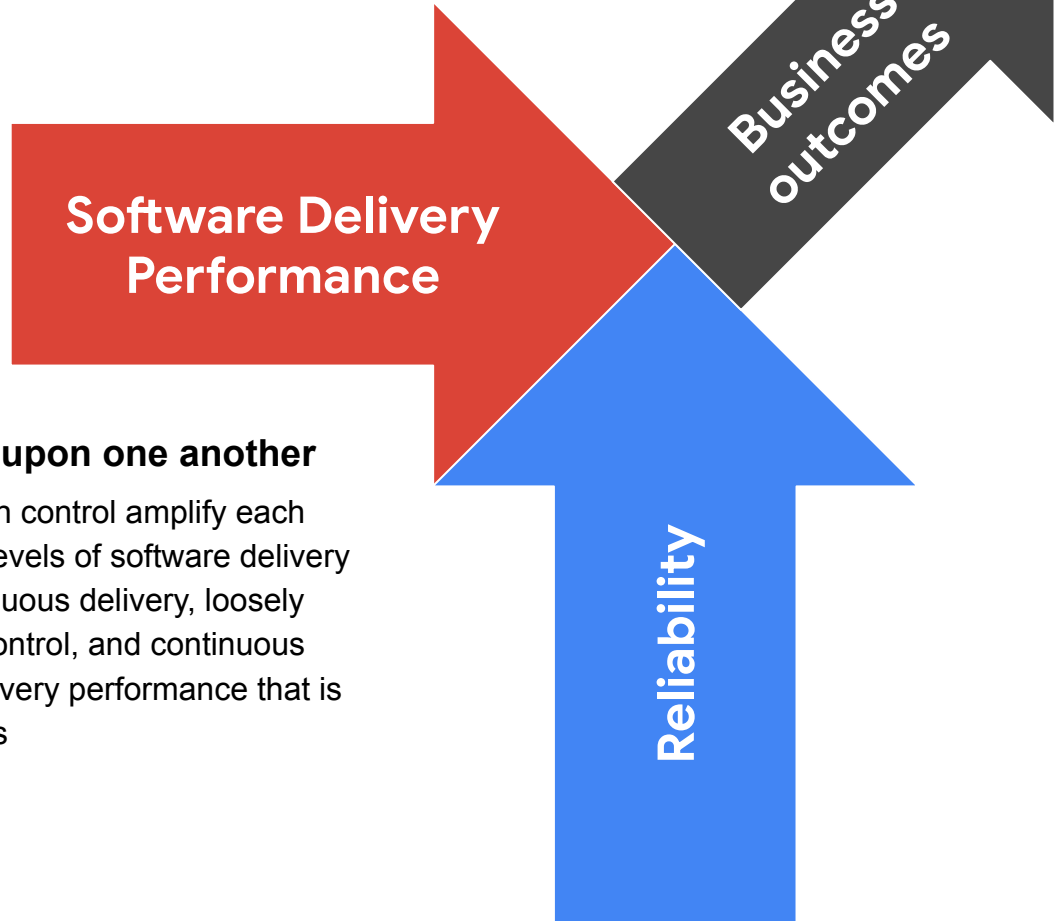
1.8x

more likely to meet or
exceed organizational goals

2022 State of DevOps

Technical capabilities build upon one another

Continuous delivery and version control amplify each other's ability to promote high levels of software delivery performance. Combining continuous delivery, loosely coupled architecture, version control, and continuous integration fosters software delivery performance that is greater than the sum of its parts



We think SRE is **emergent** from culture

Pathological (power oriented)	Bureaucratic (rule oriented)	Generative (performance oriented)
Low cooperation	Modest cooperation	High cooperation
Messengers shot	Messengers neglected	Messengers trained
Responsibilities shirked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure leads to scapegoating	Failure leads to justice	Failure leads to enquiry
Novelty crushed	Novelty leads to problems	Novelty implemented

Lessons from DevOps

What works? What doesn't?

- **Training centers**
 - ~10% of training should be classroom based
 - Most training should be mentoring or learning by doing e.g. Dojos
- **Centers of excellence**
 - Centers of *enablement* use hands on coaches
 - Learning by doing instead of best practices from the ivory tower
- **Big Bang**
 - Continuous improvement is better than delayed perfection
 - In complex areas we can't predict the future



Communities of practice



Bottom-up or grassroots



Training centers



Centers of excellence



A big bang approach

Section 2

Why the SRE approach to Reliability?

What is driving the evolution of SRE?

EVOLUTION OF OPERATIONS

OPS



- PRIMORDIAL, PROTOZOIC
- BORN IN THE SWAMPS OF PERL
- OPERATES IN A SINGLE-CELL SILO
- SURPRISINGLY RESILIENT

DEVOPS



- A CROSS-FUNCTIONAL MARVEL
- VASTLY INCREASED AGILITY
- SECRETLY JUST A BUNCH OF SINGLE CELLS THAT HAVE LEARNED NOT TO KILL EACH OTHER

DEVSECOPS



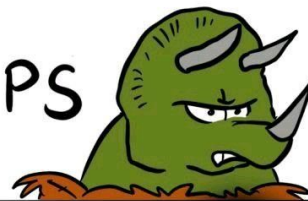
- MORE ADVANCED, MORE PARANOID
- SECURITY IS AUTOMATED RIGHT INTO ITS DNA
- KNOWS THAT SHARED RESPONSIBILITY IS THE ONLY ESCAPE FROM FOSSILIZATION

DEVSECMLOPS



- WHAT EVEN IS THIS?
- IS IT A FISH WITH FEET?
- WE SHOULD PROBABLY LEAVE IT ALONE FOR A FEW MILLION YEARS AND SEE WHAT HAPPENS

TRICERATOPS



- DOES NOT CARE ABOUT YOUR ORG STRUCTURE
- VULNERABLE ONLY TO DIRECT METEOR STRIKES
- WHAT WERE WE TALKING ABOUT, AGAIN?

@acloudguru

Q: Can you build **99.99%** services on **99.9%** infra?



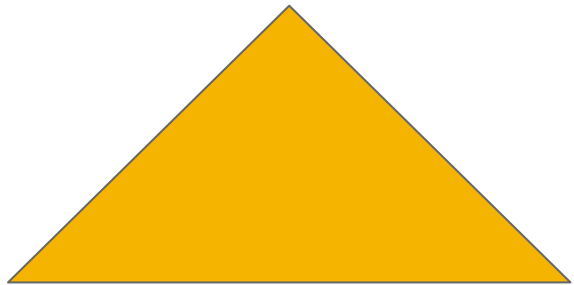
Q: Can you build **99.99%** services on **99.9%** infra?

Yes.

You can build
more reliable things
on top of
less reliable things

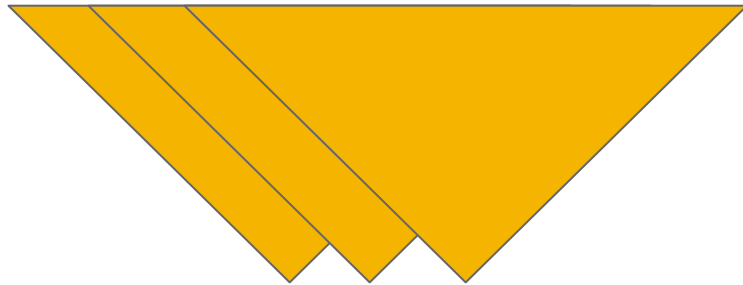
a simple example: RAID. see: *The SRE I Aspire to Be*, @aknin SREconEMEA 2019

Why the SRE approach to Reliability?



Component reliability:

- Inherit reliability from the base
- Lower levels *must* be more reliable
- "scale up"



Scalable reliability:

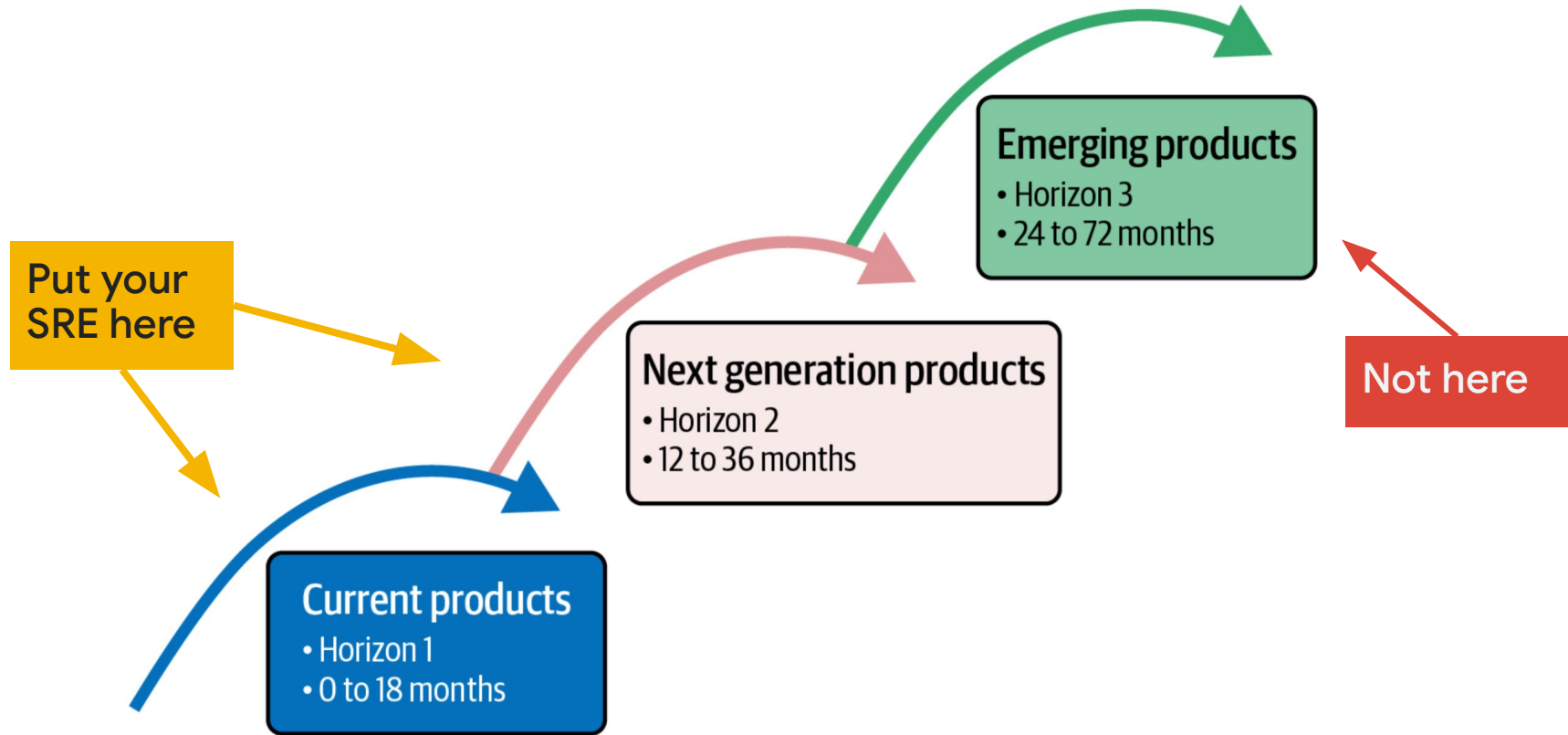
- Cost-effective base at scale
- Software *must improve* availability
- "scale out"

Why the SRE approach to Reliability (R9y)?

- ✓ R9y as product differentiator
- ✓ Critical risk mitigation
- ✓ Hyperscale services

However! – **not every service** needs SRE

Why the SRE approach to Reliability?




Why the SRE approach to Reliability?

Cost reduction...?

Yes! ... But also no.

SRE is a **strategic investment** (\$↑)
in long-term operational efficiency (\$↓)

Cost optimization is **global**, not **local**.



Section 3

SRE Principles

SRE Principles



SRE Principles

Start small

- Build practices incrementally
- More advanced capabilities need to have foundational ones first
- Prevent **organization destroying mistakes**



SRE Principles

Invest in people

- Staffing and retention
- Hiring feels easy but growing is more sustainable
- **Don't fire everyone in ops who can't code**
- Value existing employees – they know the business!

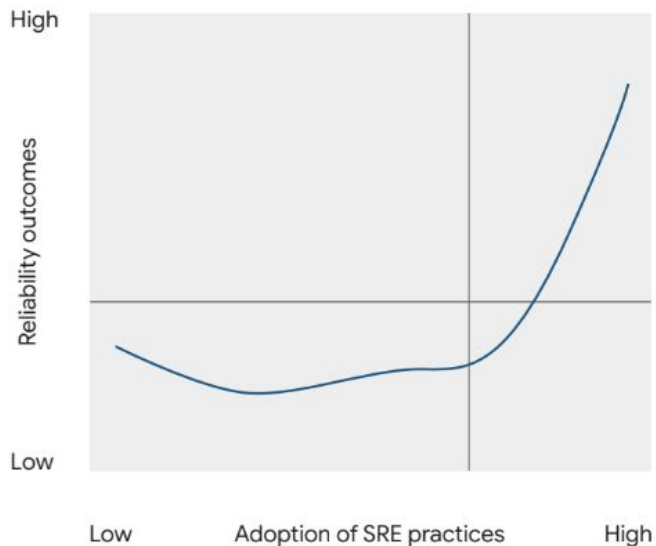
SRE Principles

Embrace risk

- Create a **safe to fail** environment
- **You can't only take risks that will succeed**
- Demonstrating active leadership is important
- Treat failures as **unplanned learning opportunities**

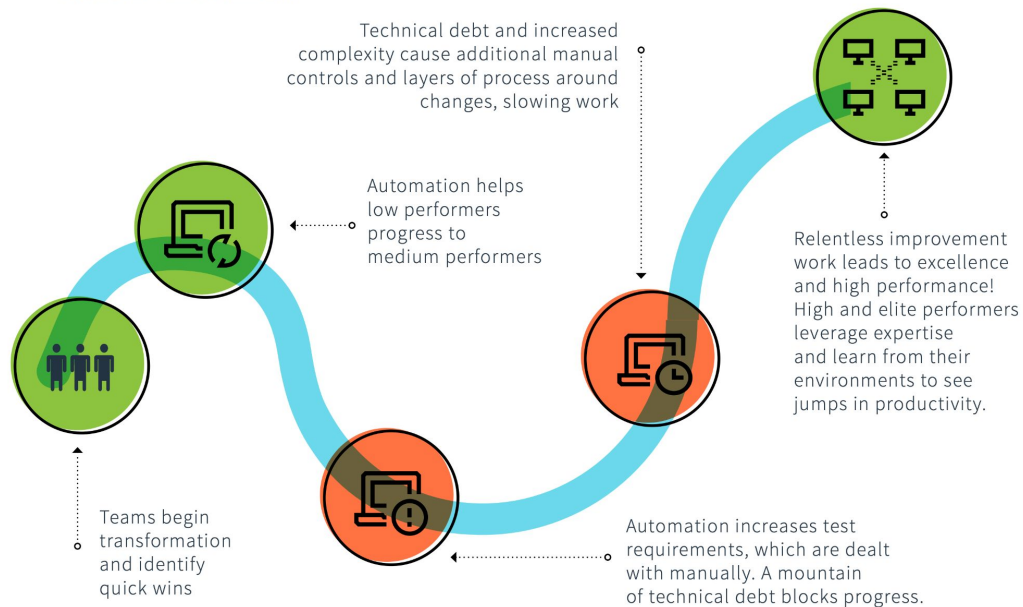
SRE & DevOps agree (SoDR 2022)

Give it time



Teams that persist beyond initial steps of SRE adoption see increasing improvement in reliability outcomes

J-CURVE OF TRANSFORMATION





Section 4

SRE Practices

SRE Practices

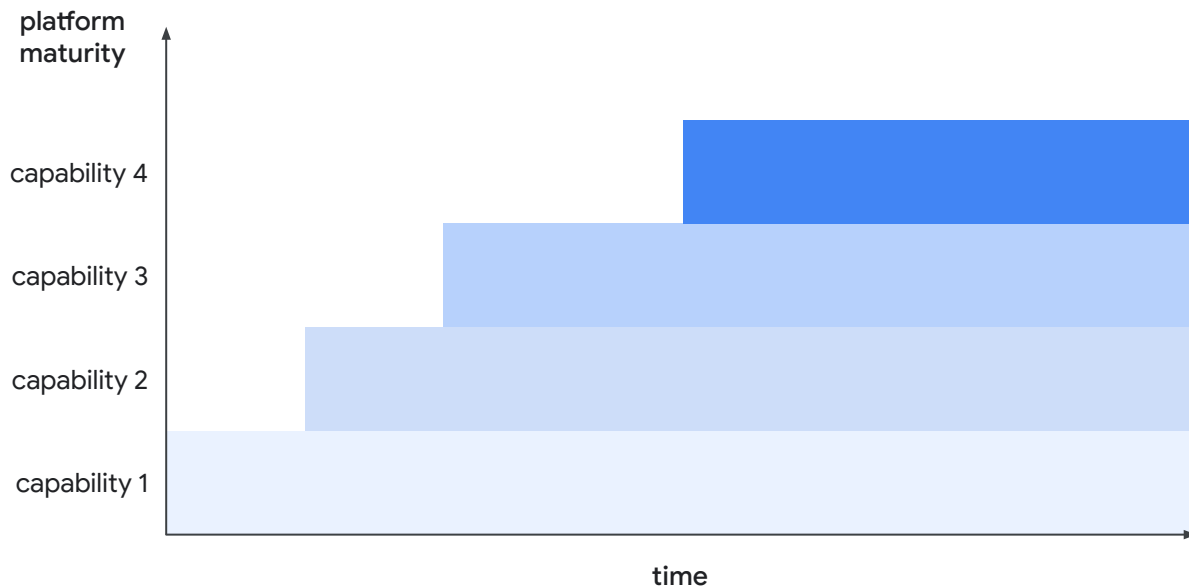
- Avoid SRE as [dev support](#) / "Developer IT"
 - "hey prod is broken" :(
 - "my laptop is broken" :(:(:(
 - "the printer is broken" :(:(:(:(:(
- Target **mid term planning** (6 months to 2 years)
- Getting started? Incident Response, Postmortems and review, repeat.
- **Cause-based** alerting vs **symptom-based**
- Use feedback loops to make this **intentional** (e.g. [PDCA](#) / [OODA](#))

SRE Practices

- Build a **platform** of **capabilities**!
- Capabilities get built, purchased, added over time
- Services are introduced to the platform, **as it makes sense**.
- Antipattern: **Pick the toughest thing first** since that will fix all the problems

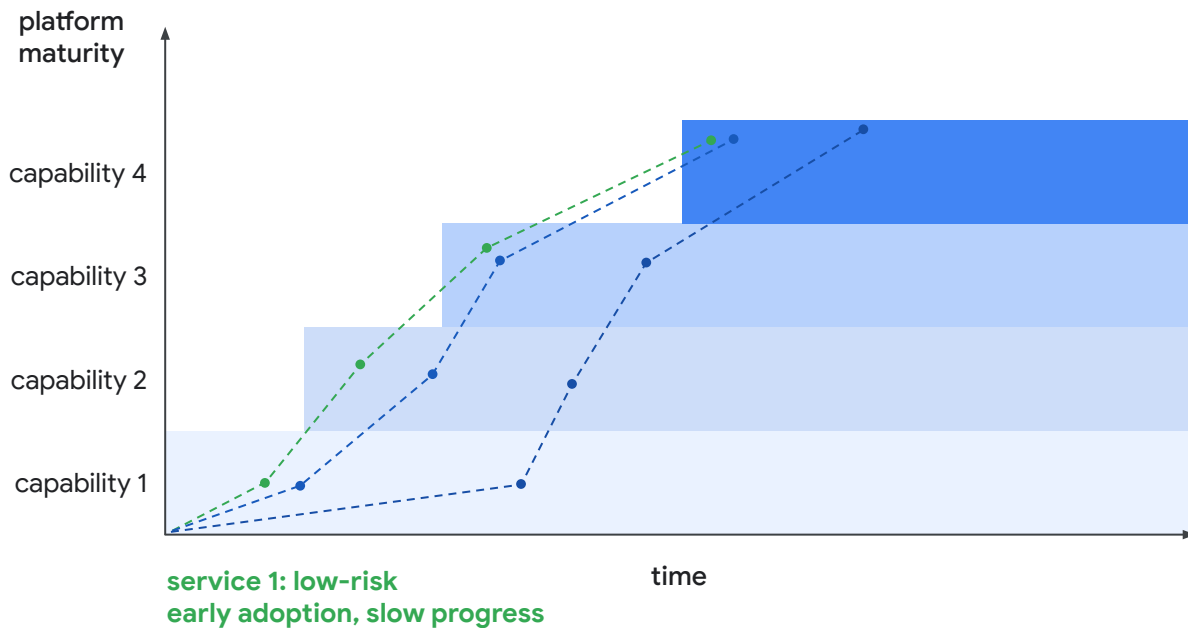
Building Platforms

Add capabilities over time: CI/CD, rollbacks, multi-cluster



Building Platforms

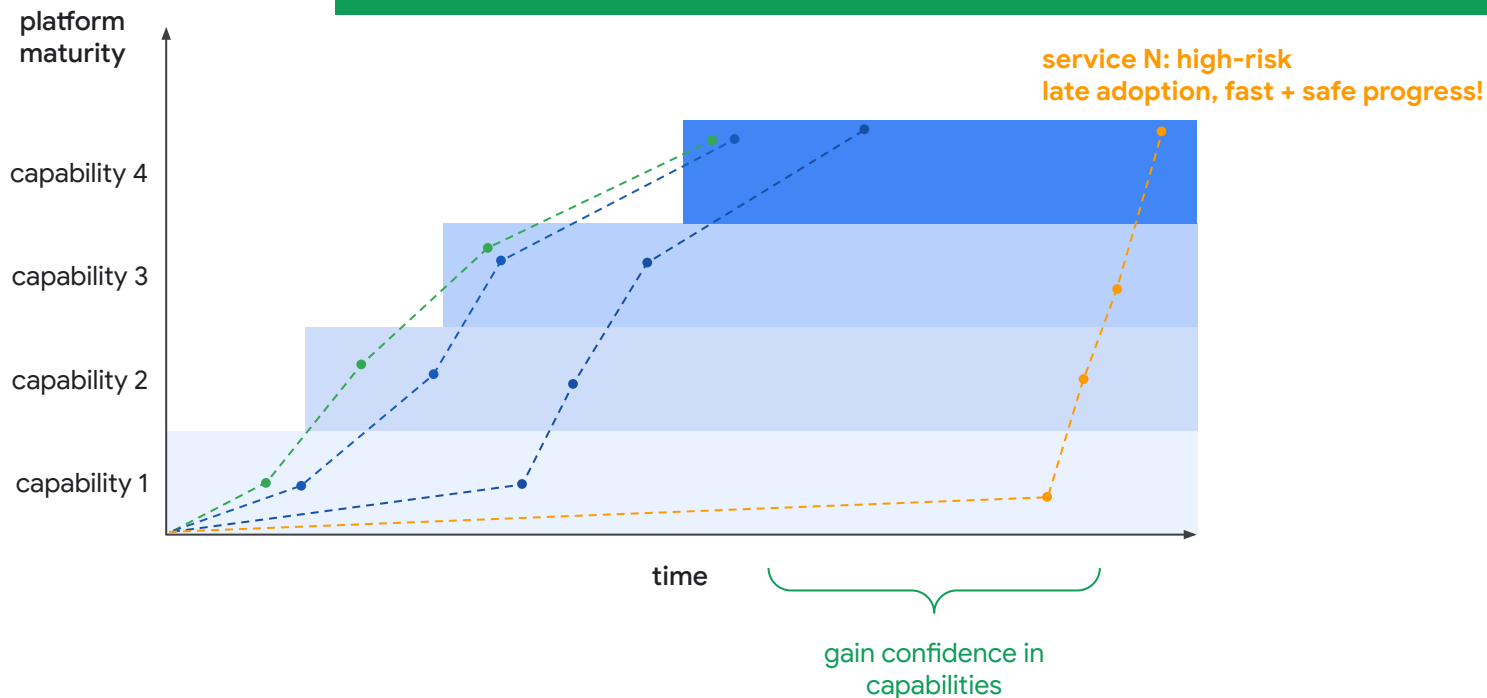
Introduce services as their platform requirements are met



Building Platforms

Gain confidence in capabilities of platform over time.

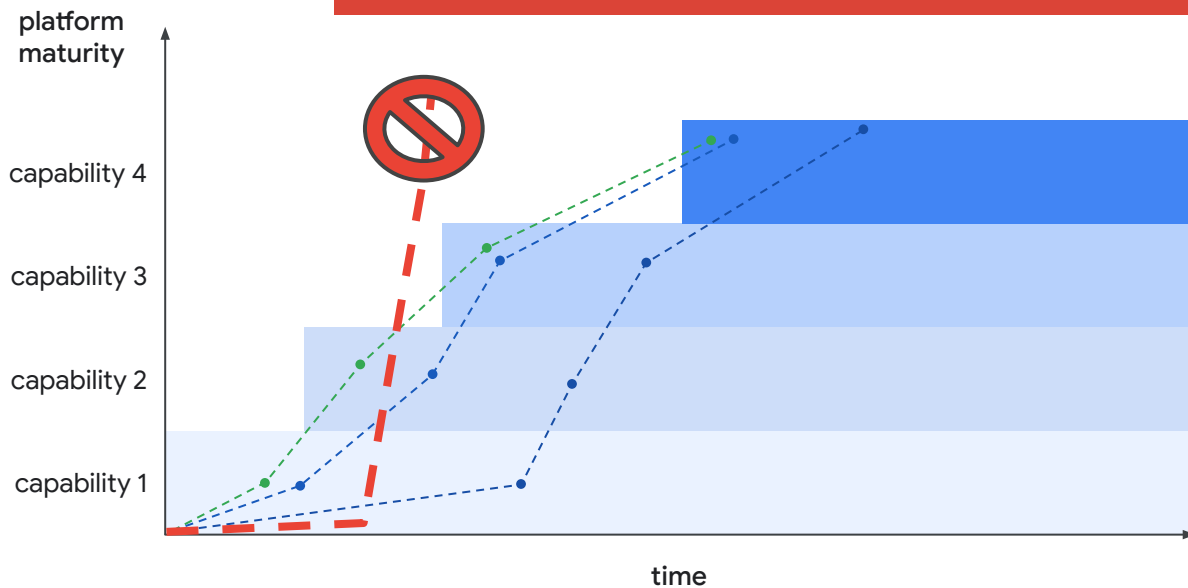
Deploy high-value services *later*, granting those capabilities all at once.



Building Platforms

Don't adopt high-value services too early!

Not enough resilience in platform, not enough confidence in capabilities.



SRE Practices

- Consider a **Chief Reliability Officer** (CRO)
 - Senior person with a [seat at the table](#) for strategic reliability decisions
 - If your enterprise has an equivalent of Google's **Ben Treynor Sloss** then this will be a key indicator of success
 - **Sponsorship matters!**
- Compare with a **Chief Information Security Officer** (CISO)
 - "Security is everyone's responsibility"
 - Enterprises also have CISOs **to nurture and champion** those efforts
- Just like Security, Reliability needs to be seen as an **investment** and **not a cost center**
- Maybe not *necessary*, but we've seen **sponsor abandonment** ⇒ team failure

SRE Practices

Is it working?

Often knowing whether you're making progress **won't be in a dashboard**, you'll need to use **proxy metrics** to evaluate:

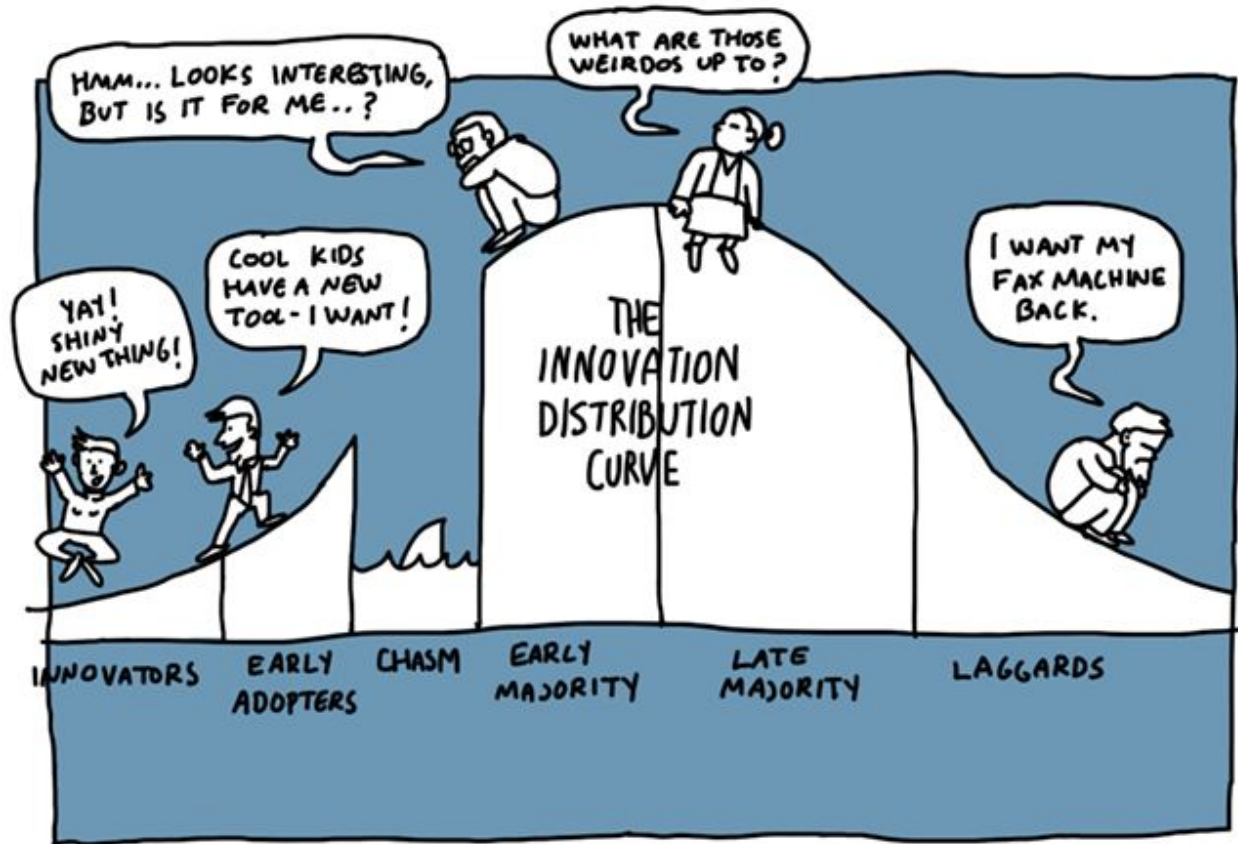
- Can you **enforce consequences** when an error budget is exhausted?
- Is individual **heroism** still being praised?
- Are you **correlating funding with outages**?
- Is **success celebrated** or treated as table stakes?



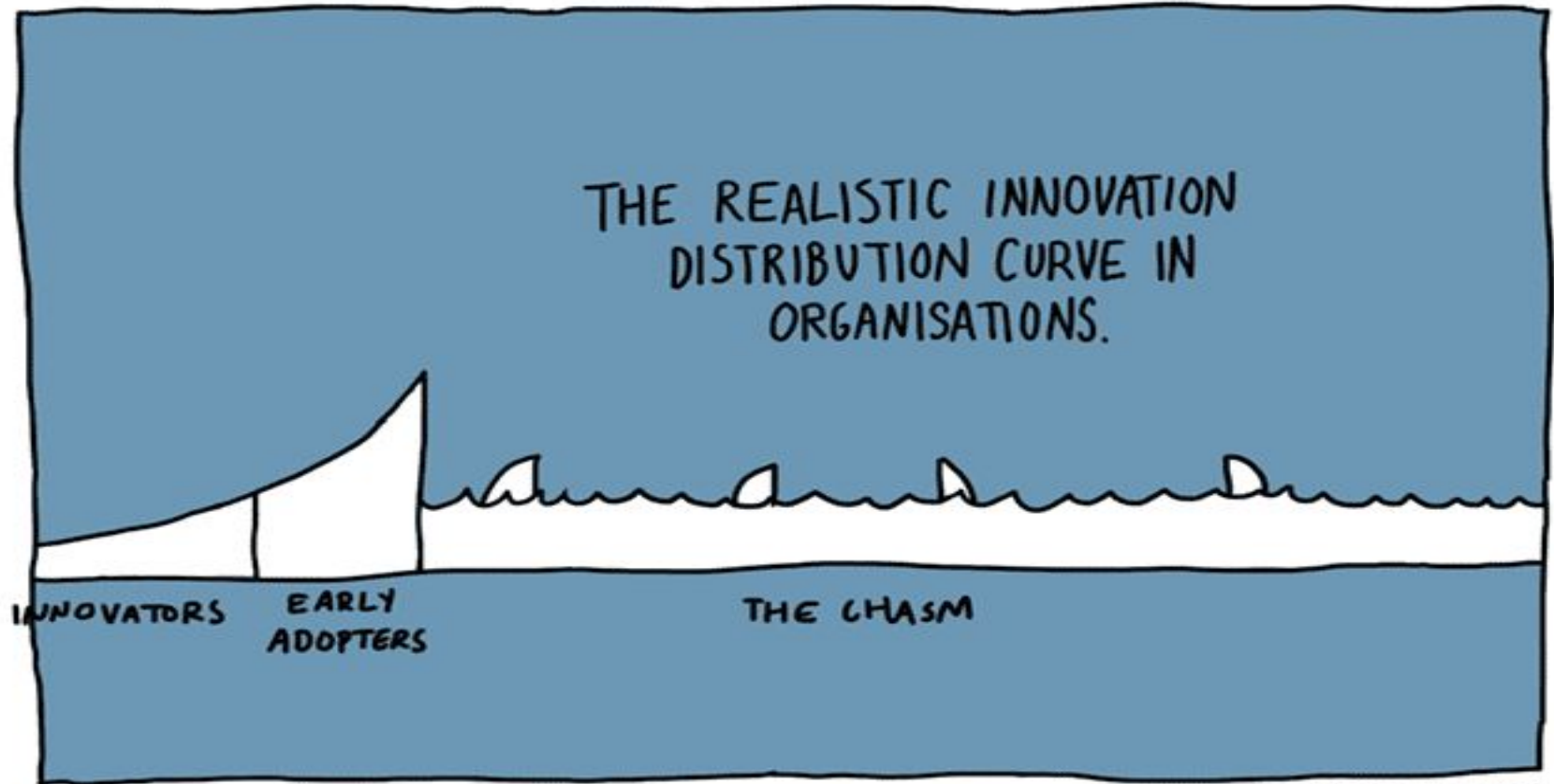
Section 5

Actively Nurturing Success

Actively Nurturing Success



Actively Nurturing Success



"Culture eats strategy for breakfast"

Google did the research on what makes the magic happen.

It's **not just**: free food and ping pong

Those don't **cause** the right culture.
They come from the right culture.

Follow the re:Work model to adapt your culture



Actively Nurturing Success

Hints and tips!

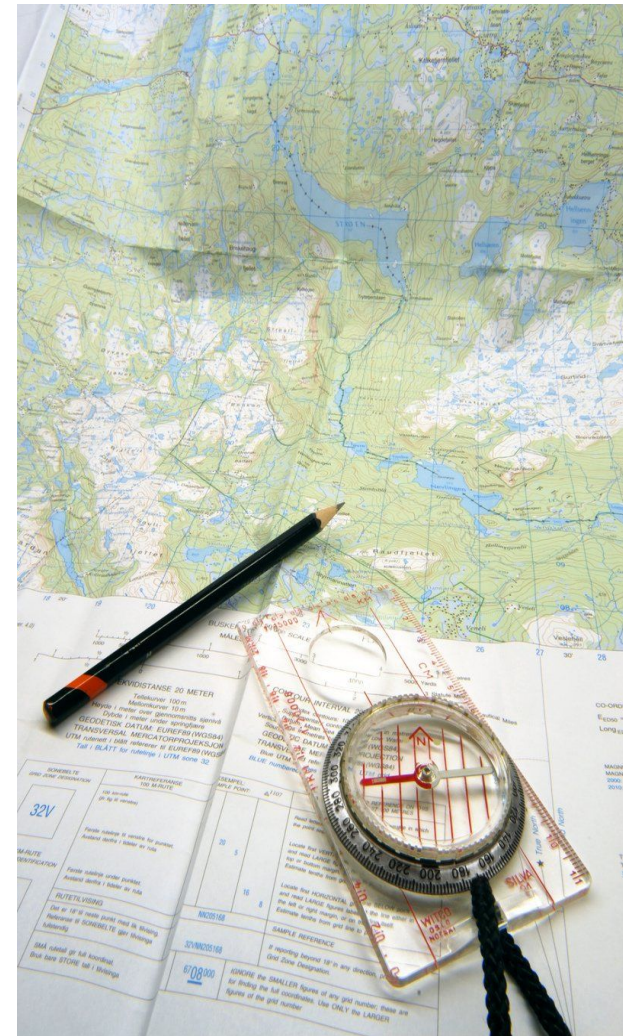
- Strive for **sublinear scaling**
- Building and retaining sustainable teams (grow your teams organically)
- SRE is dynamic and **evolves over time**
- High reliability levels take (much) longer than you think
- Understand the dedicated org model **isn't supposed to be a silo**
- Communities need water and sunlight to thrive
- Promotion/training/**compensation** needs to match other roles (especially dev)

Bonus – What's next?

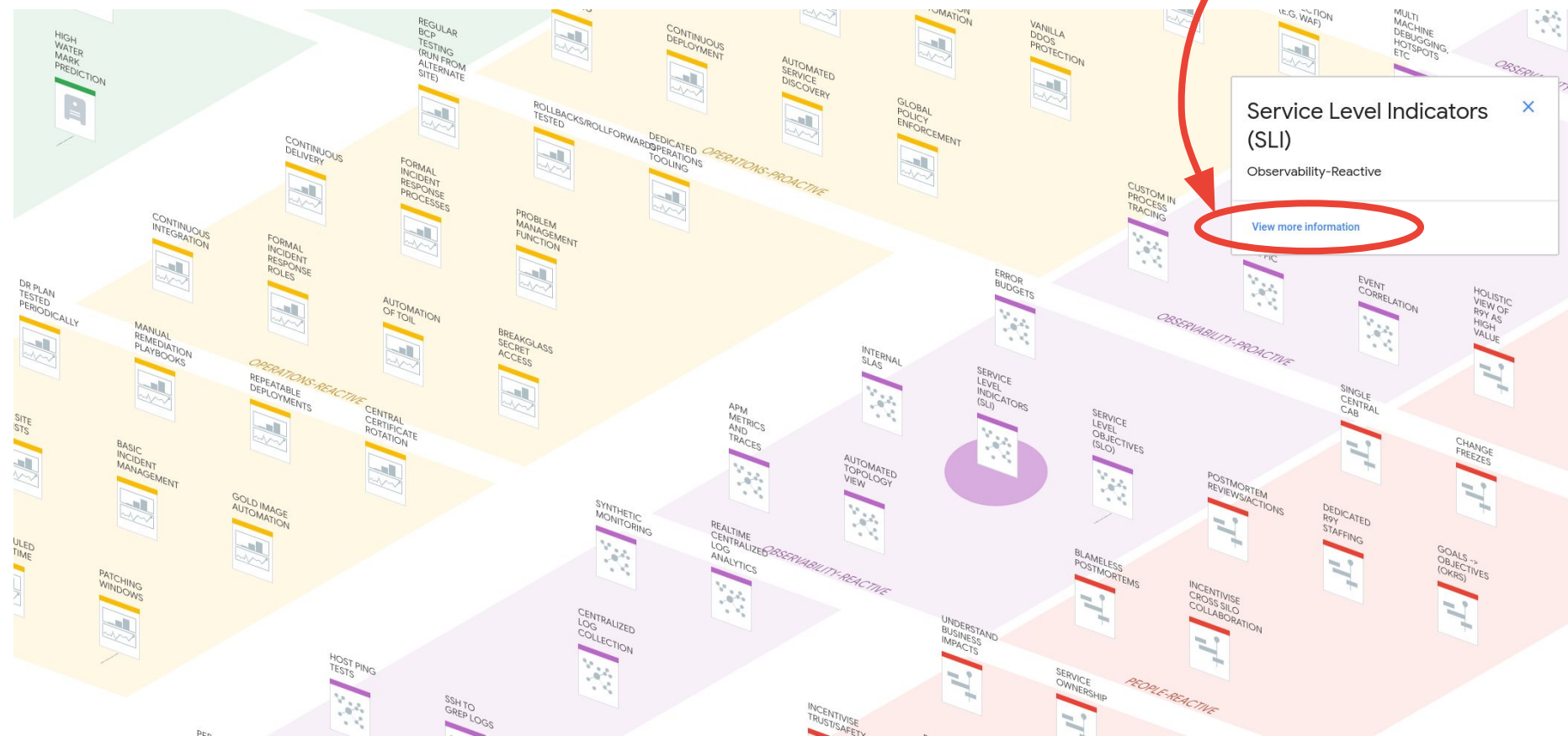
Reliability Mapping

- An SME-constructed map of **reliability capabilities**
- Divided into **Eras** (demarcated by availability nines)
- And **Streams/Personas** e.g. Dev, Infra, Observability

This is in preview!



Zoomed Reliability Map - <https://r9y.dev/>



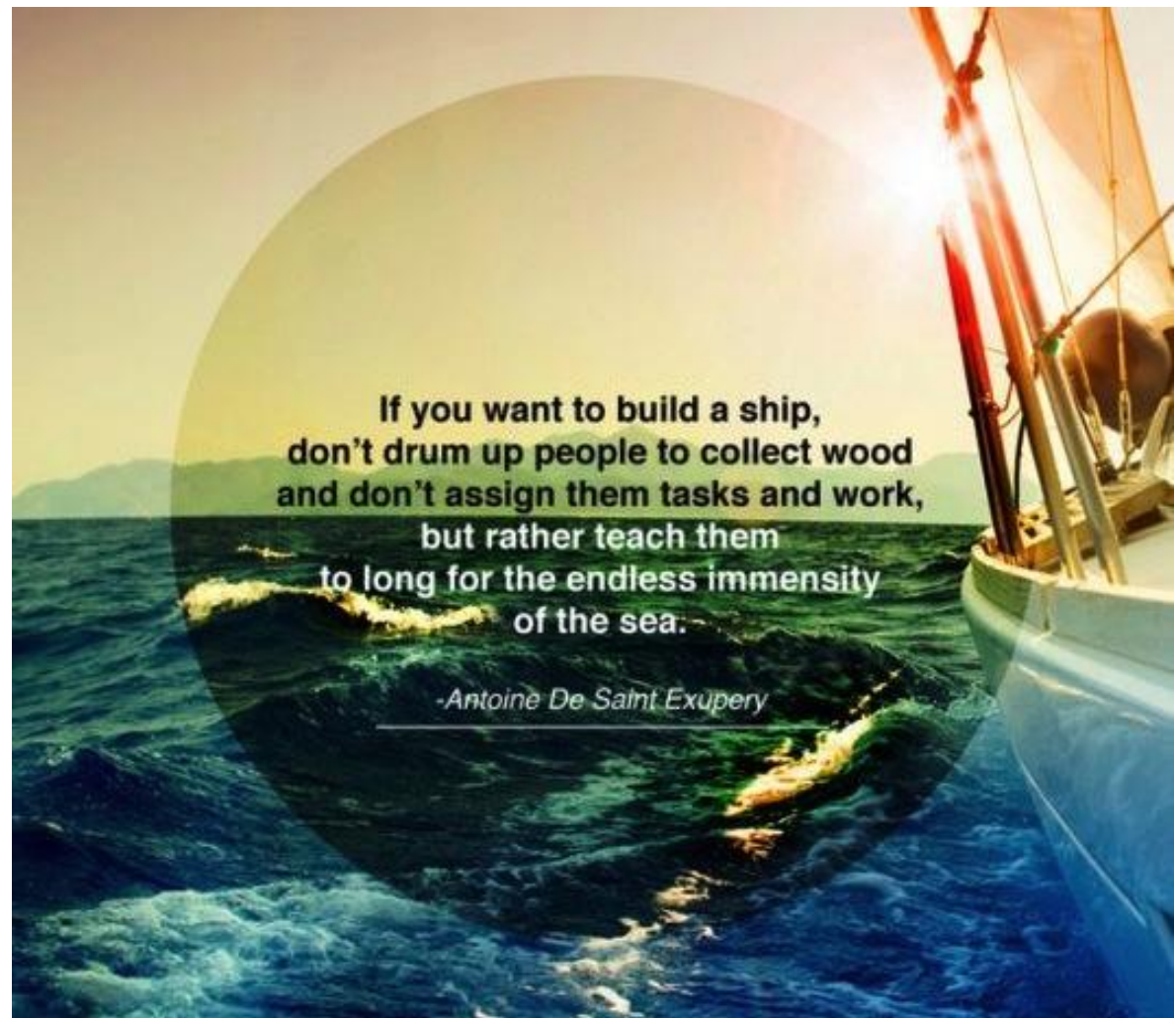
Big Picture Reliability Map - <https://r9y.dev/>





Conclusion

Conclusion

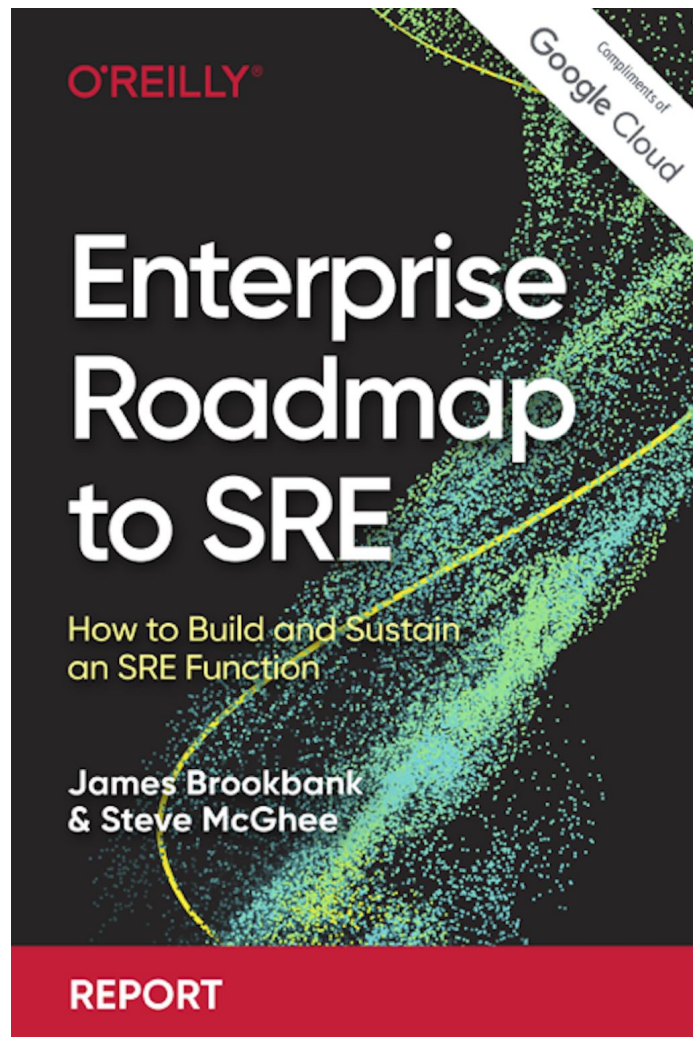


**If you want to build a ship,
don't drum up people to collect wood
and don't assign them tasks and work,
but rather teach them
to long for the endless immensity
of the sea.**

-Antoine De Saint Exupery

Enterprise Roadmap to SRE

Copies are available!
~~Here!~~ Soon! ~~Now!~~ Lunch!



Fin

