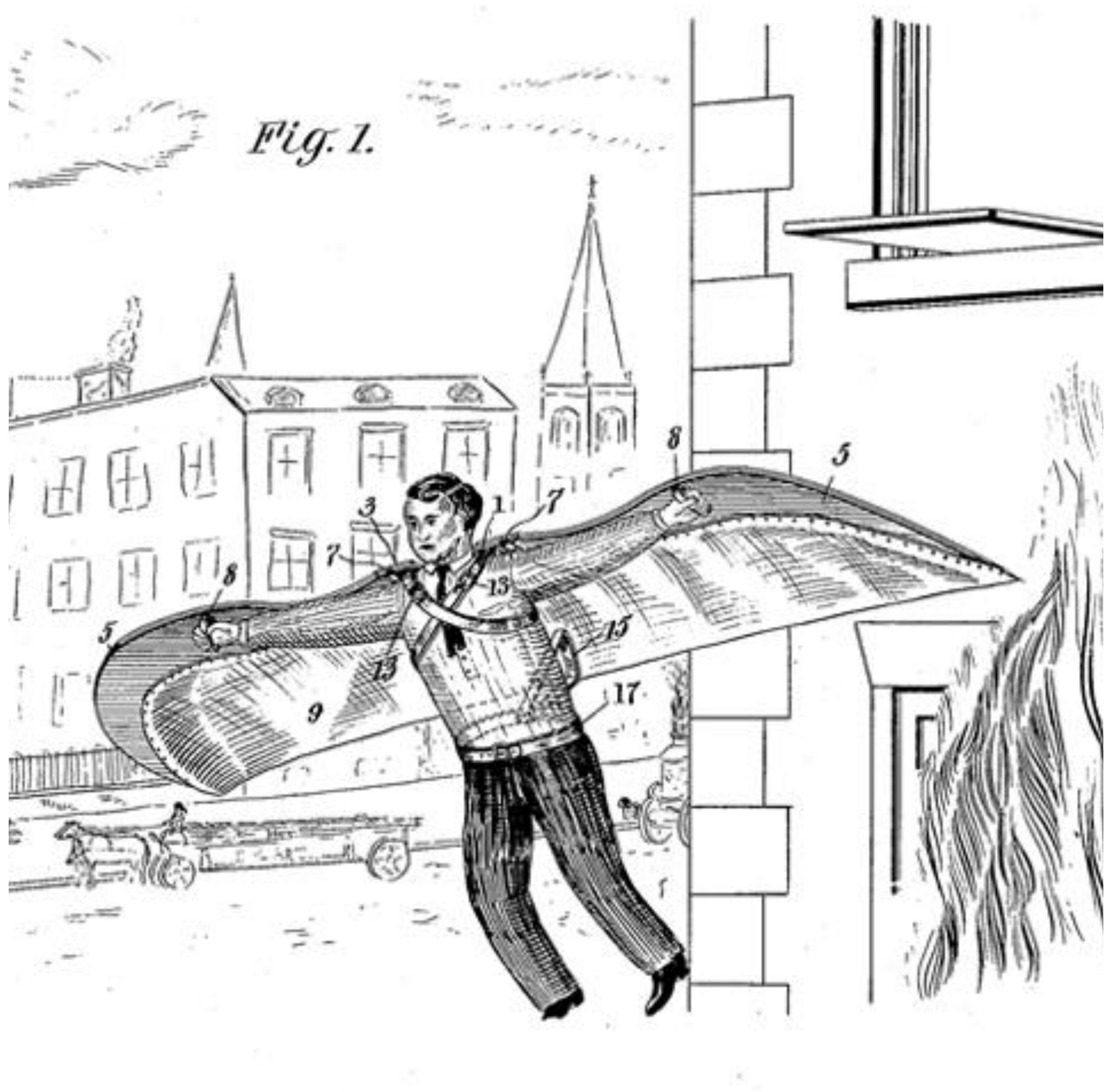
Site Reliability Engineering

How **KPN** Might Run Production Systems

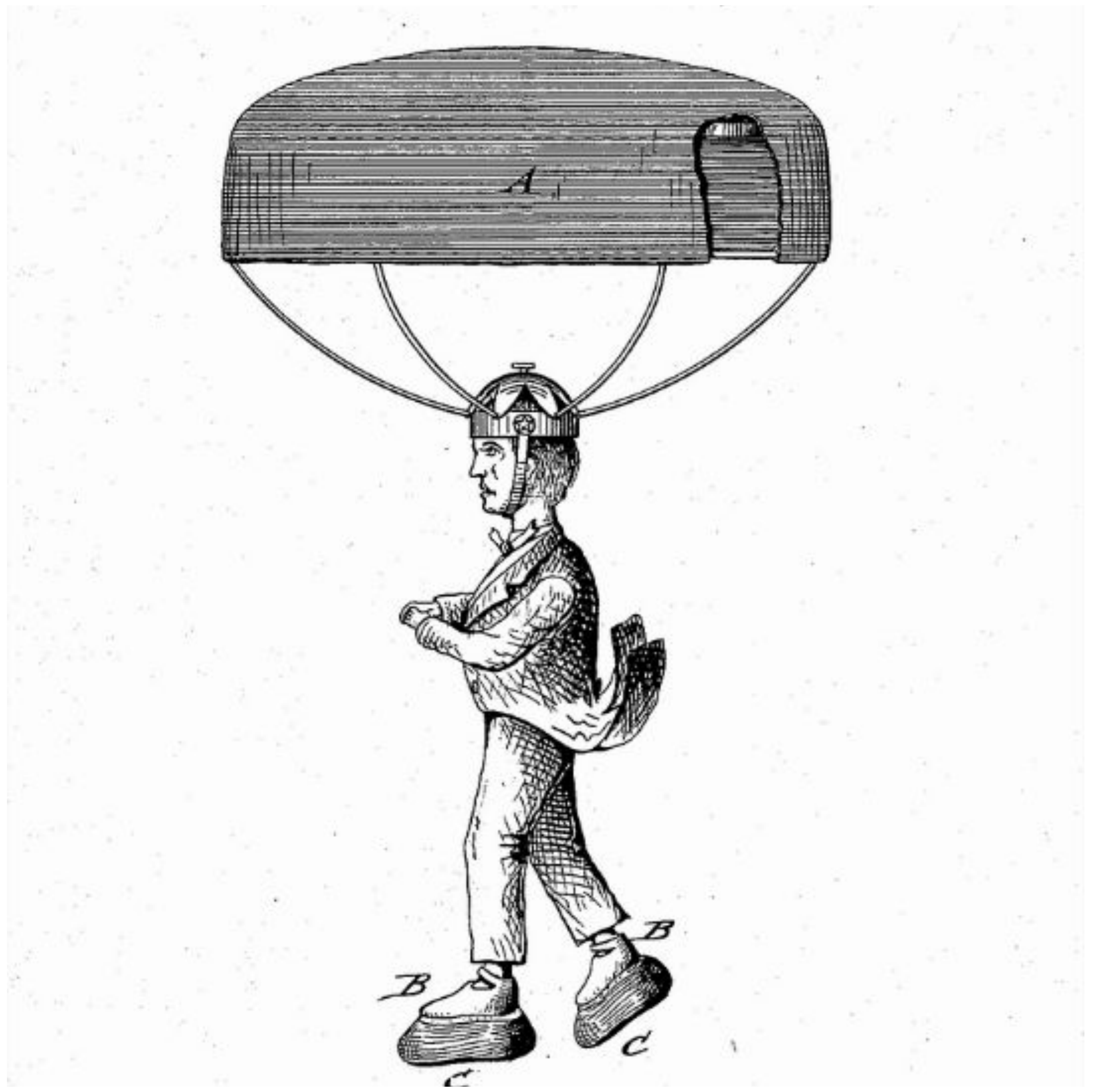
Picture by Matt Howard at Unsplash

agenda

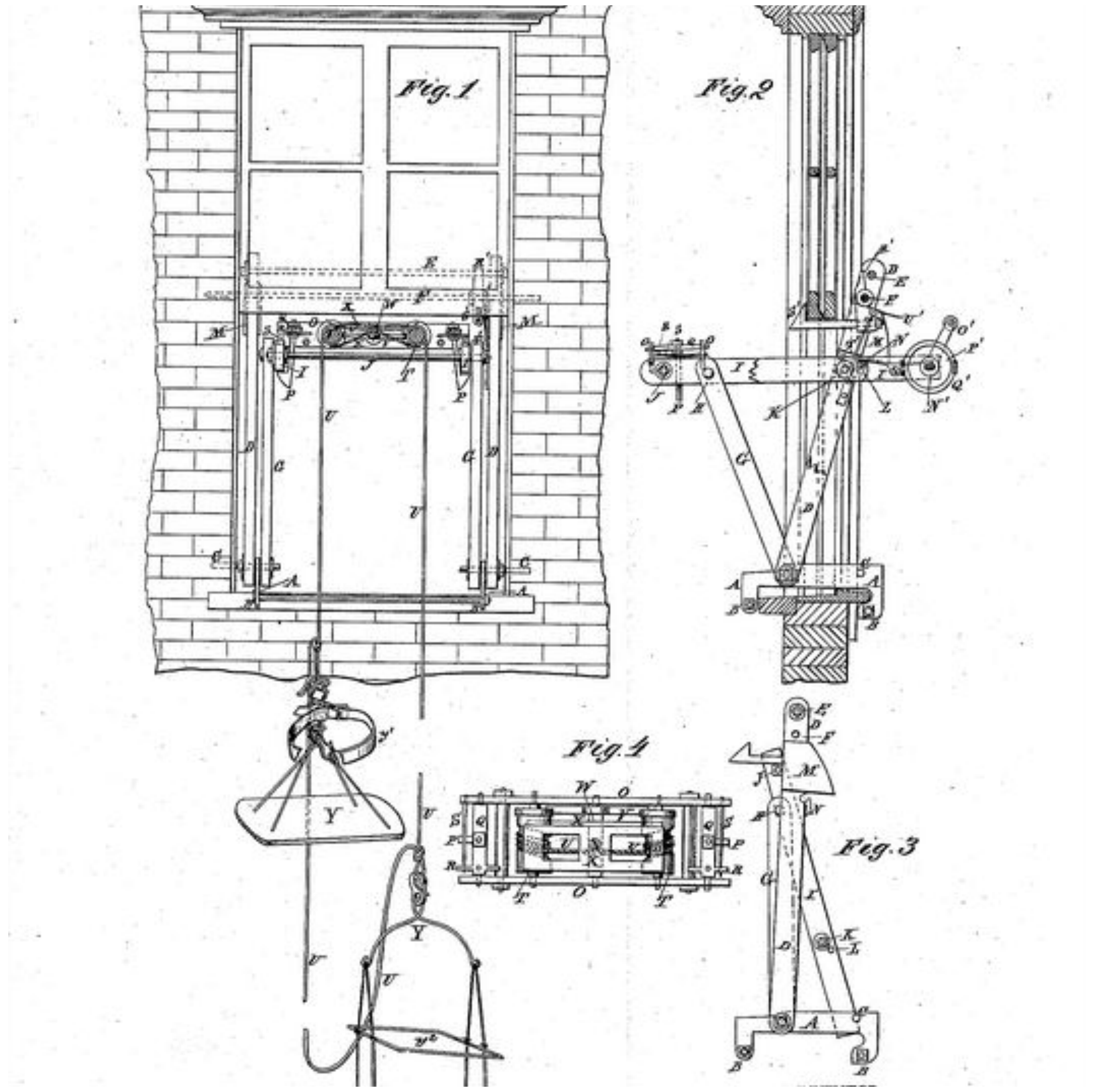
introduction



principles



practices

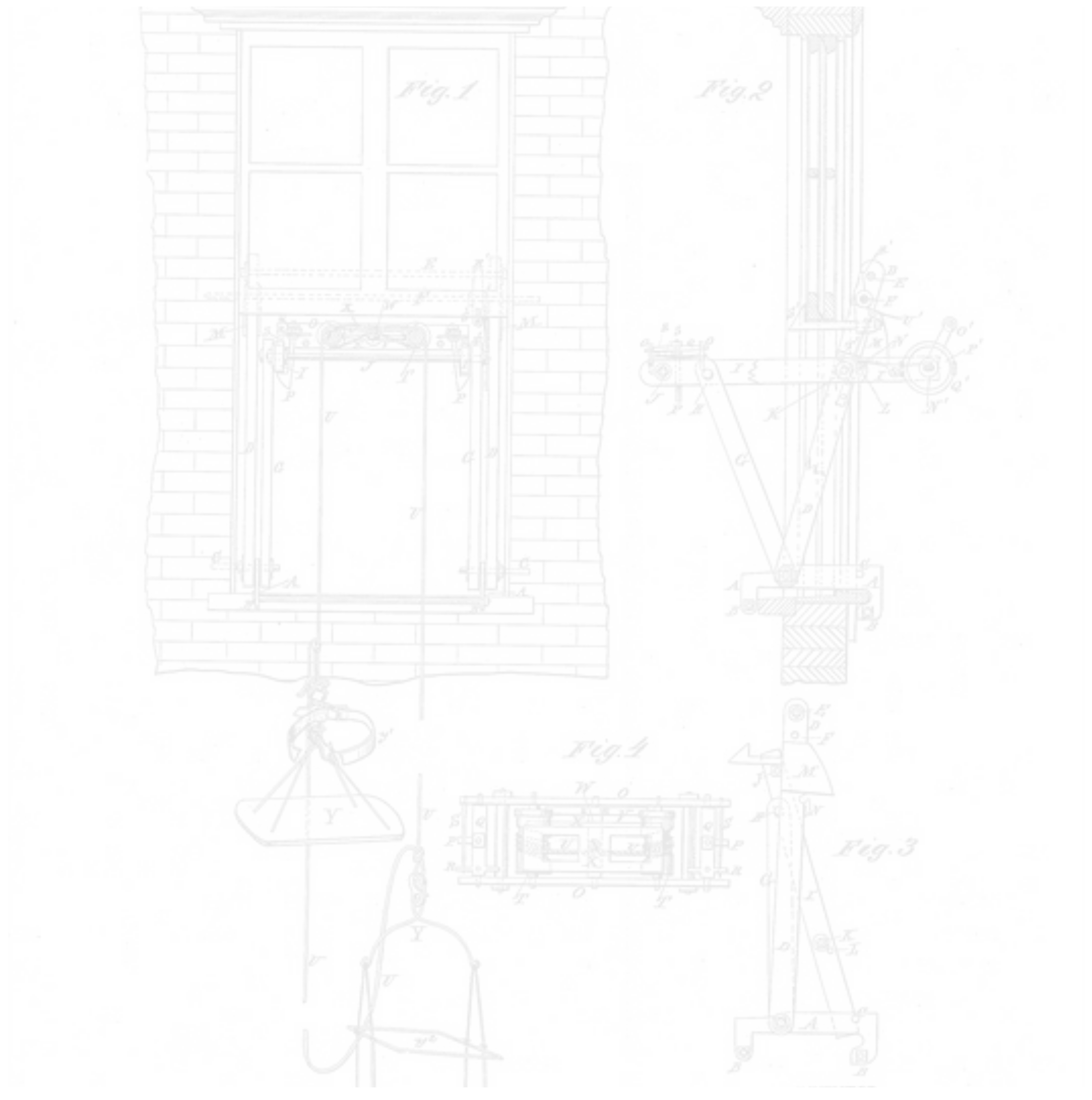


agenda

introduction

principles

practices



DevOps

The main goal of DevOps is to **break down the silos in IT** (development, operations, networking and security) through a loose set of practices, guidelines and culture.

The key points of DevOps are easy to remember by the acronym **CALMS** credited to Jez Humble, co-author of "*The DevOps Handbook*". The five pillars are:

01 **culture**

A culture of shared responsibility and collaboration.

02 **automation**

Seek ways to automate tasks to eliminate repetitive manual work and create reliable systems.

03 **lean**

Focus on producing value to the end user and work in small batch sizes or in other words be agile.

04 **measurement**

Measure everything to provide visibility into the system and to show your improvement.

05 **sharing**

Open information sharing, collaboration and communication to break the friction between development and operations due to a lack of common ground.

How SRE Relates to DevOps

The term (and job role) Site Reliability Engineering (SRE) originated from Google, where it inherited much of the culture within Google before the term become widely known across the industry.

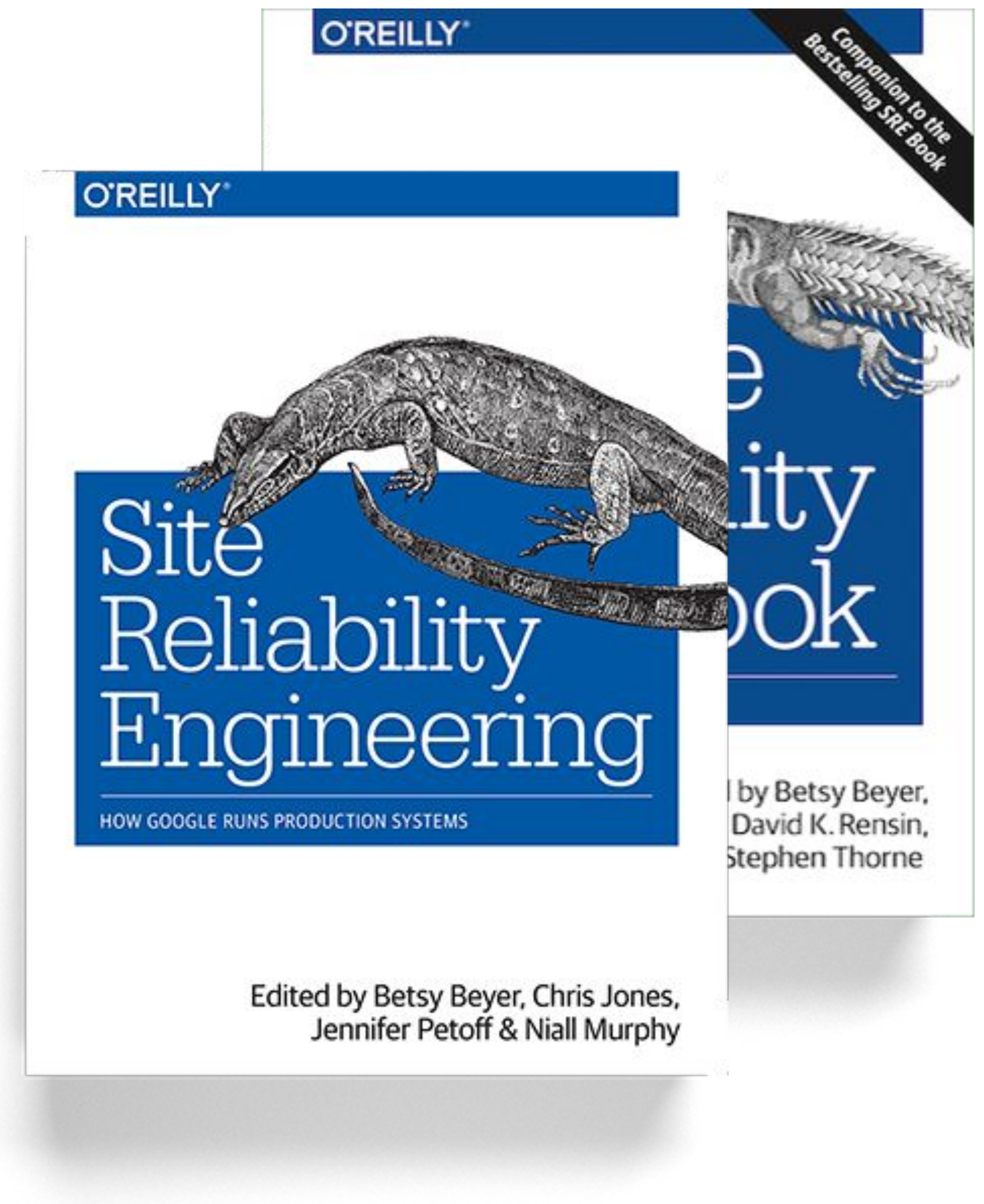
The core principles of DevOps are consistent with many of the SRE principles. They both require discussion, management support, and buy-in support from the engineers doing the work to make it work.

*"class **SRE** implements interface **DevOps**"*

The Site Reliability Workbook

"Fundamentally, it's what happens when you ask a software engineer to design an operations function."

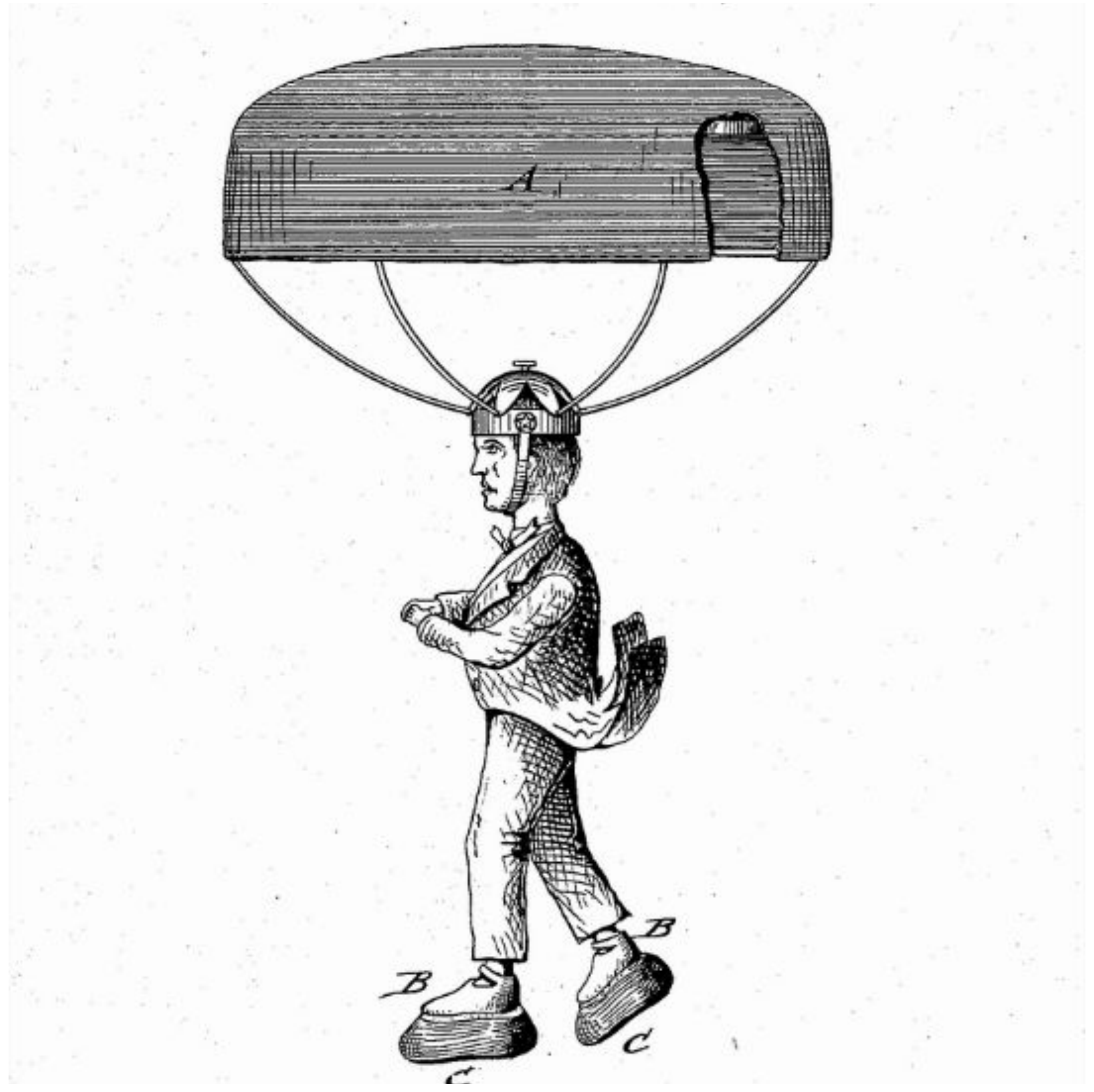
Ben Treynor, VP of Engineering at Google


O'REILLY
Site Reliability Engineering
HOW GOOGLE RUNS PRODUCTION SYSTEMS
Edited by Betsy Beyer, Chris Jones, Jennifer Petoff & Niall Murphy
O'REILLY
Companion to the Bestselling SRE Book
by Betsy Beyer, David K. Rensin, Stephen Thorne

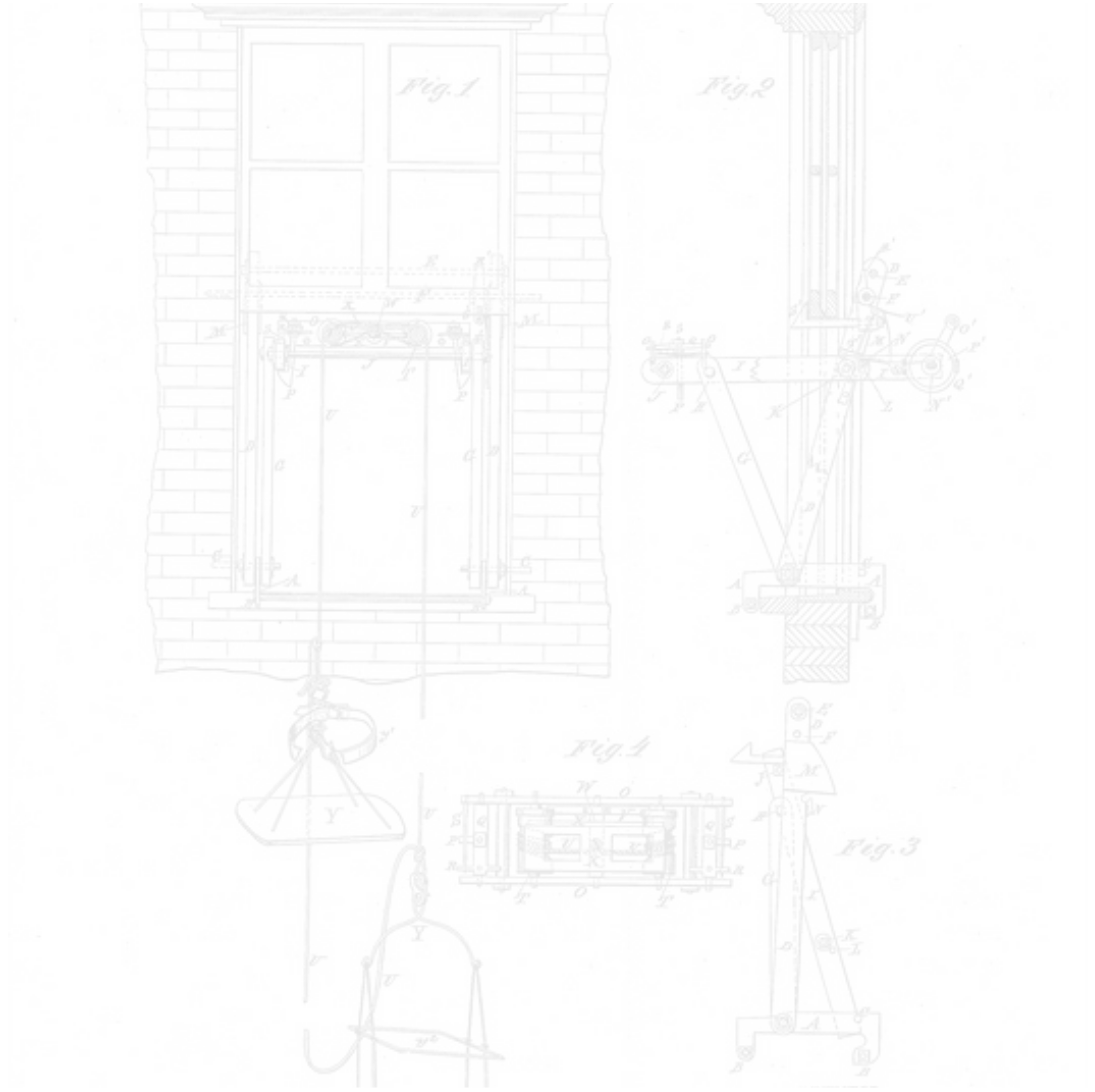
agenda

introduction

principles



practices



Embracing Risk

Extreme reliability comes at a cost:

- Cost does not increase linearly as reliability increments.
- Limits how fast new features a team can afford to offer.

Example

A user on a **99% reliable smartphone** can't tell the difference between **99.99%** and **99.999%** service reliability!

Error Budgets

Forming your error budget:

1. Define an Service Level Object (**SLO**).
2. **Measure uptime**.
3. Difference between SLO and uptime is the “**budget**”.
4. If the budget is above the SLO, new releases can be pushed.

Service Level Objectives

• Service Level Indicator	**SLI**
• Service Level Objective	**SLO**
• Service Level Agreement	**SLA**

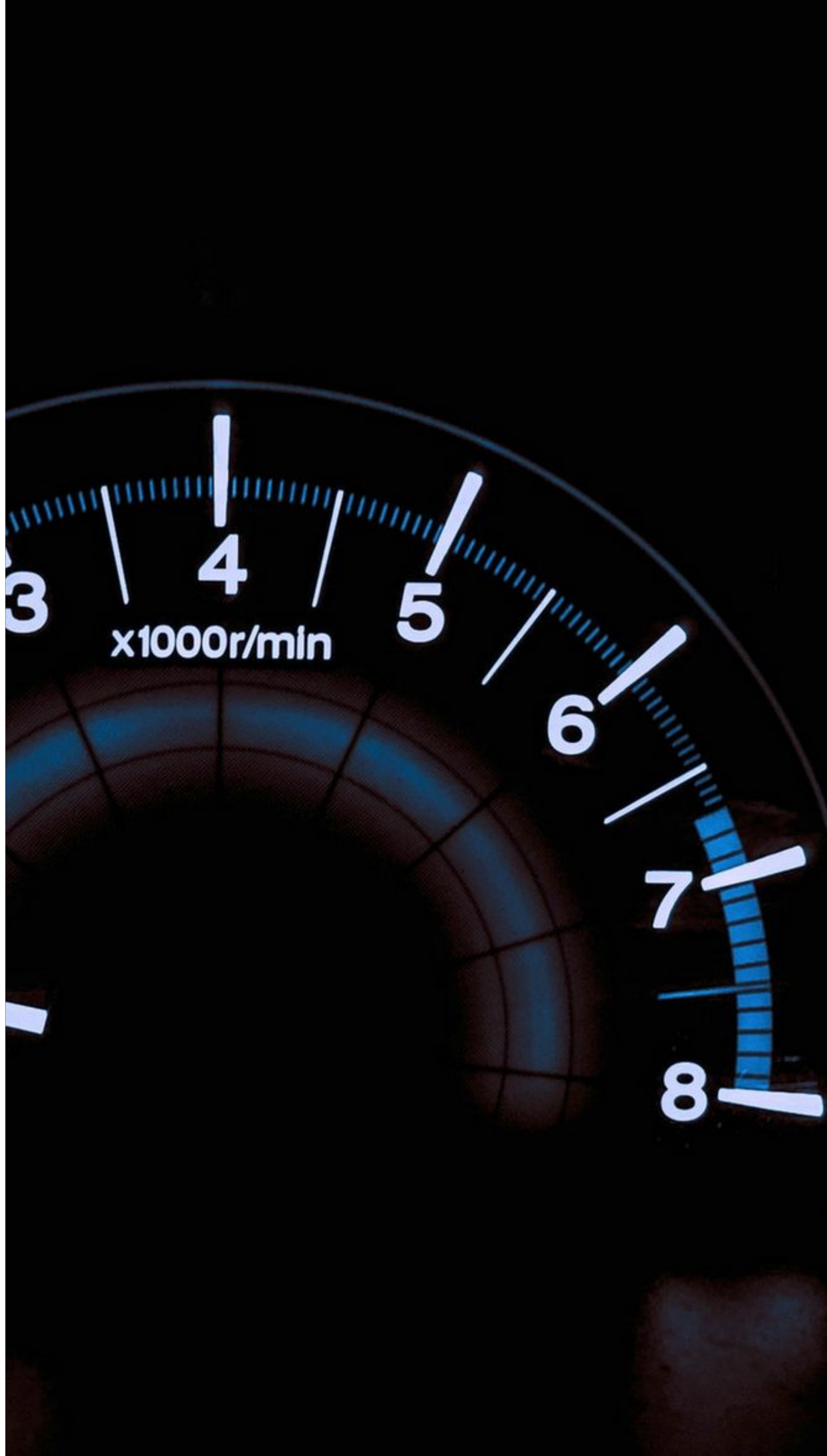
SLOs set expectations:

- Keep a safety margin.
- Don’t overachieve!

Tip

Easy way to tell the difference between SLO and an SLA is to ask “W*hat happens if the SLOs aren’t met?*”:

If there is no explicit consequence, then your are most certainly looking at an SLO.



Eliminating Toil

Toil is the **manual**, **repetitive**, **automatable** work tied to running a production service. As the service growns, the amount of toil grows.

Less toil means more time to spend on improving reliability, performance or utilization.

Spend time on long-term engineering project work instead of toil.

Note

Being on-call is part of toil!

Monitoring

Whitebox & blackbox monitoring

The four golden signals:

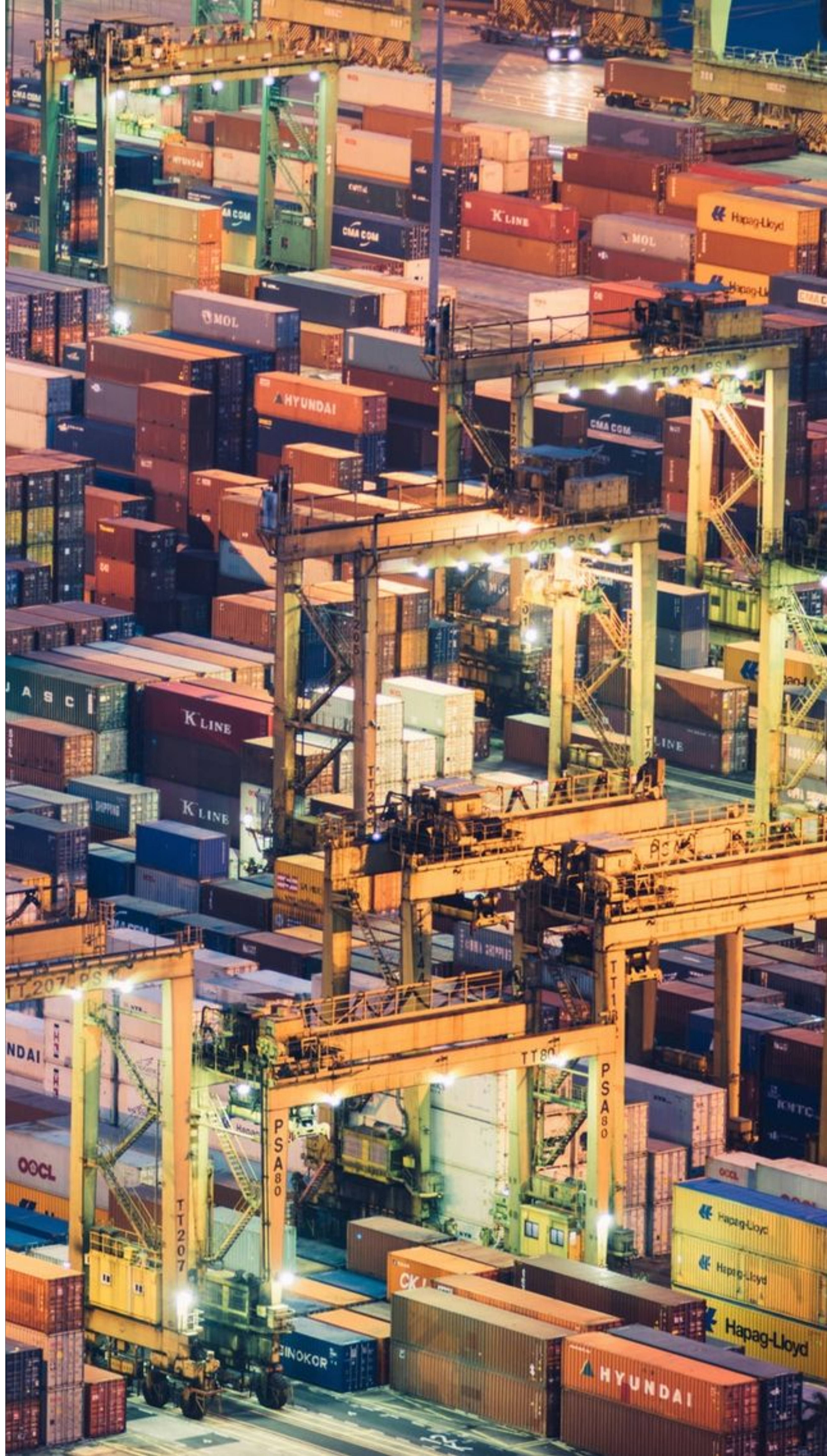
1. Latency
2. Traffic
3. Errors
4. Saturation

Automation

As an SRE you try to automate this year's job away.

The value of automation:

- Consistency
- Platform: engineers both deeply understand the existing processes and can later automate novel processes more quickly
- Faster repairs
- Faster action
- Time saving



Release Engineering

This to consider when looking at release engineering:

- Self-service model
- High velocity
- Hermetic builds: insensitive to the libraries and other software installed on the build machine
- Enforcement of policies and procedures
- Start releasing engineering at the beginning

Summary

Build → Test → Package → Deploy → REPEAT!

Simplicity

Software simplicity is a prerequisite for reliability

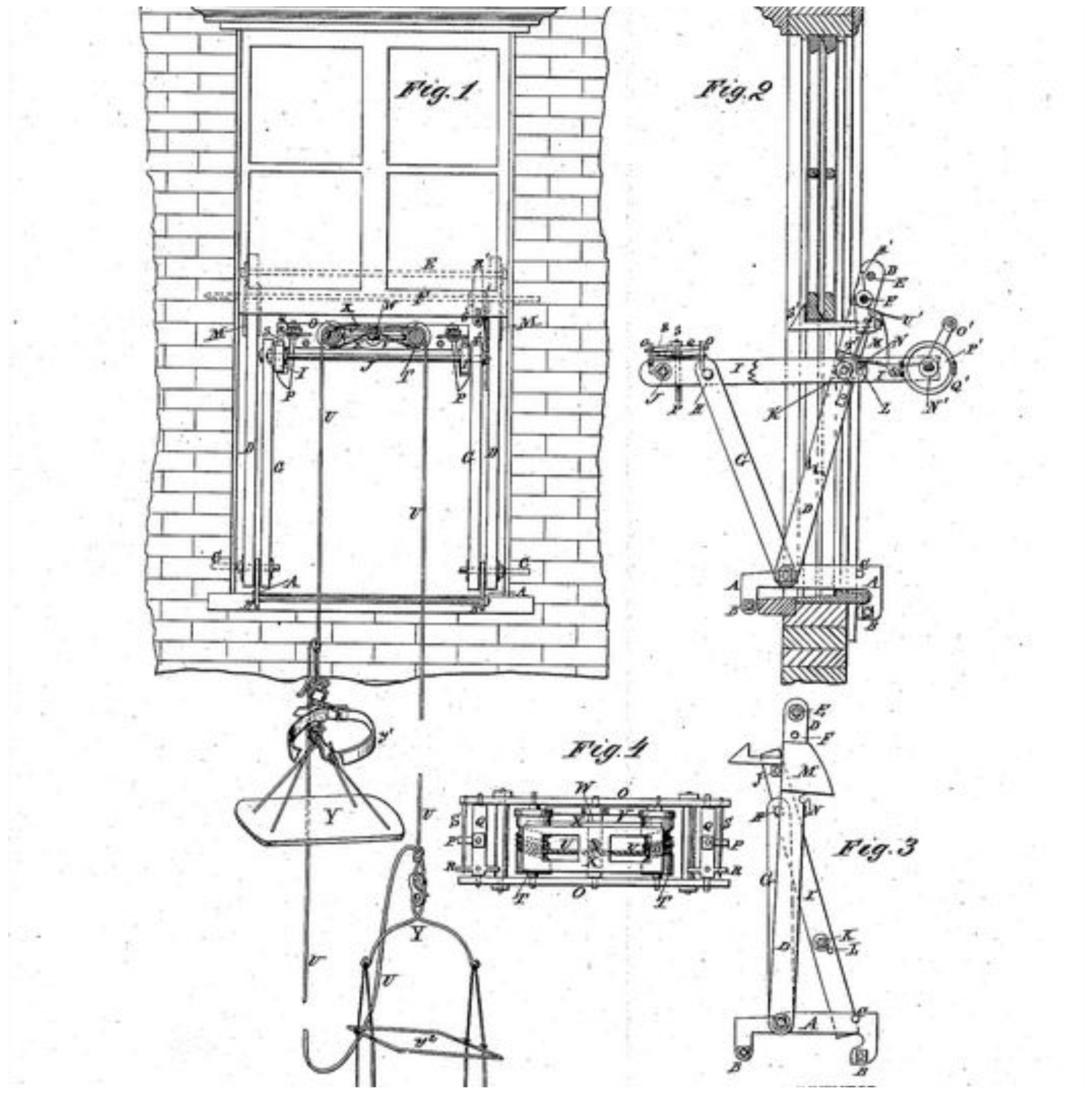
- Software systems are inherently dynamic and unstable
- System stability versus agility
- The virtue of boring: software should not be spontaneous and interesting!
- I won't give up my code!
- The "negative lines of code" metric
- Minimal APIs
- Modularity
- Release simplicity

agenda

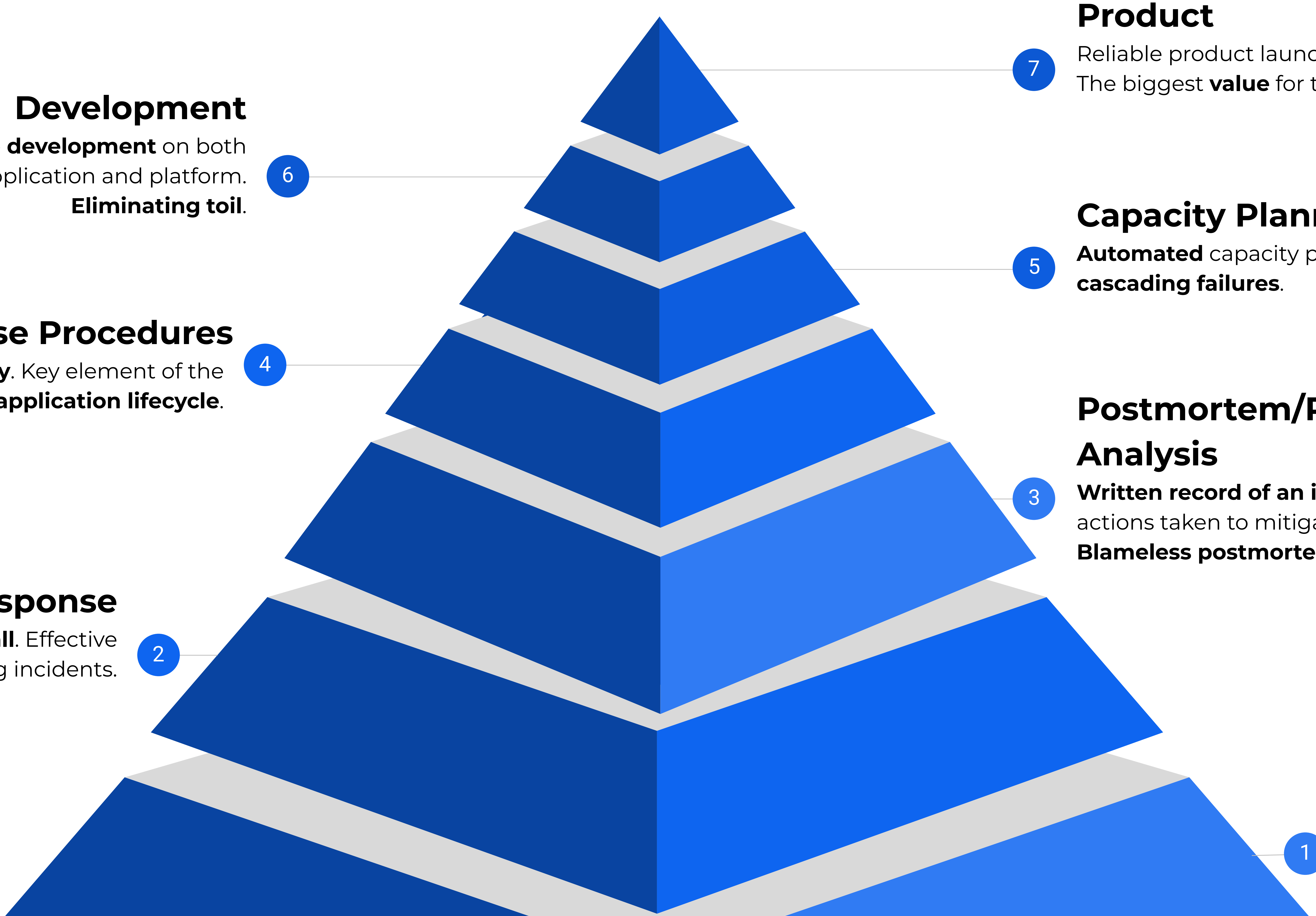
introduction

principles

practices



Service Reliability Hierarchy



Product

7 Reliable product launches..
The biggest **value** for the **business**.

Development

6 **Software development** on both the application and platform. **Eliminating toil**.

Capacity Planning

5 **Automated** capacity planning. Addressing **cascading failures**.

Testing & Release Procedures

4 Testing for **reliability**. Key element of the **application lifecycle**.

Postmortem/Root Cause Analysis

3 **Written record of an incident**, the impact and actions taken to mitigate and resolve the issue..
Blameless postmortem culture

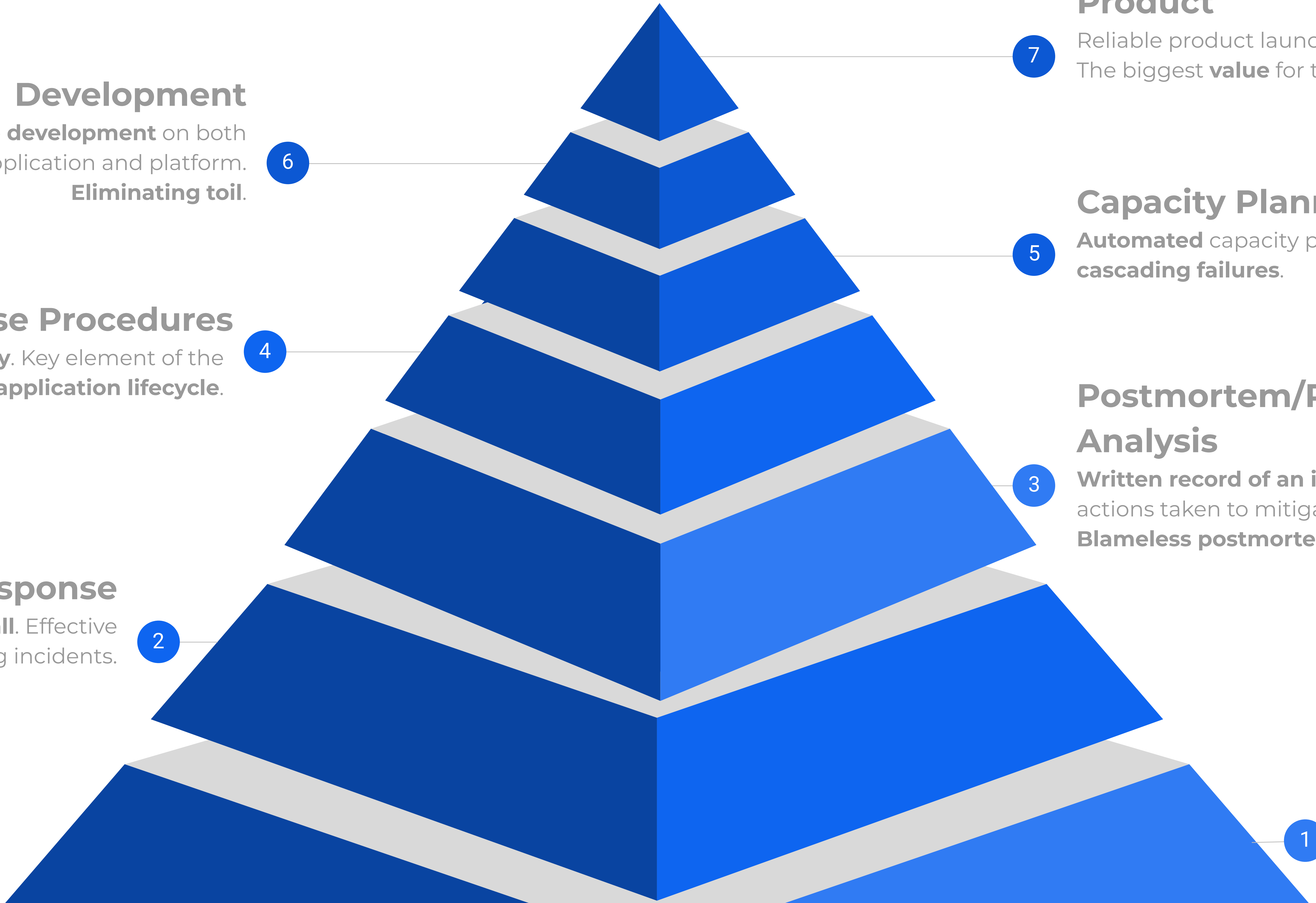
Incident Response

2 **Embracing risk**. Being **on-call**. Effective **troubleshooting**. Managing incidents.

Monitoring

1 **SLI, SLO** & **SLA**.
The four golden signals.

Service Reliability Hierarchy



Product
7 Reliable product launches..
The biggest **value** for the **business**.

Development
6 **Software development** on both the application and platform. **Eliminating toil**.

Capacity Planning
5 **Automated** capacity planning. Addressing **cascading failures**.

Testing & Release Procedures
4 Testing for **reliability**. Key element of the **application lifecycle**.

Postmortem/Root Cause Analysis
3 **Written record of an incident**, the impact and actions taken to mitigate and resolve the issue..
Blameless postmortem culture

Incident Response
2 **Embracing risk**. Being **on-call**. Effective **troubleshooting**. Managing incidents.

Monitoring
1 **SLI, SLO** & **SLA**.
The four golden signals.

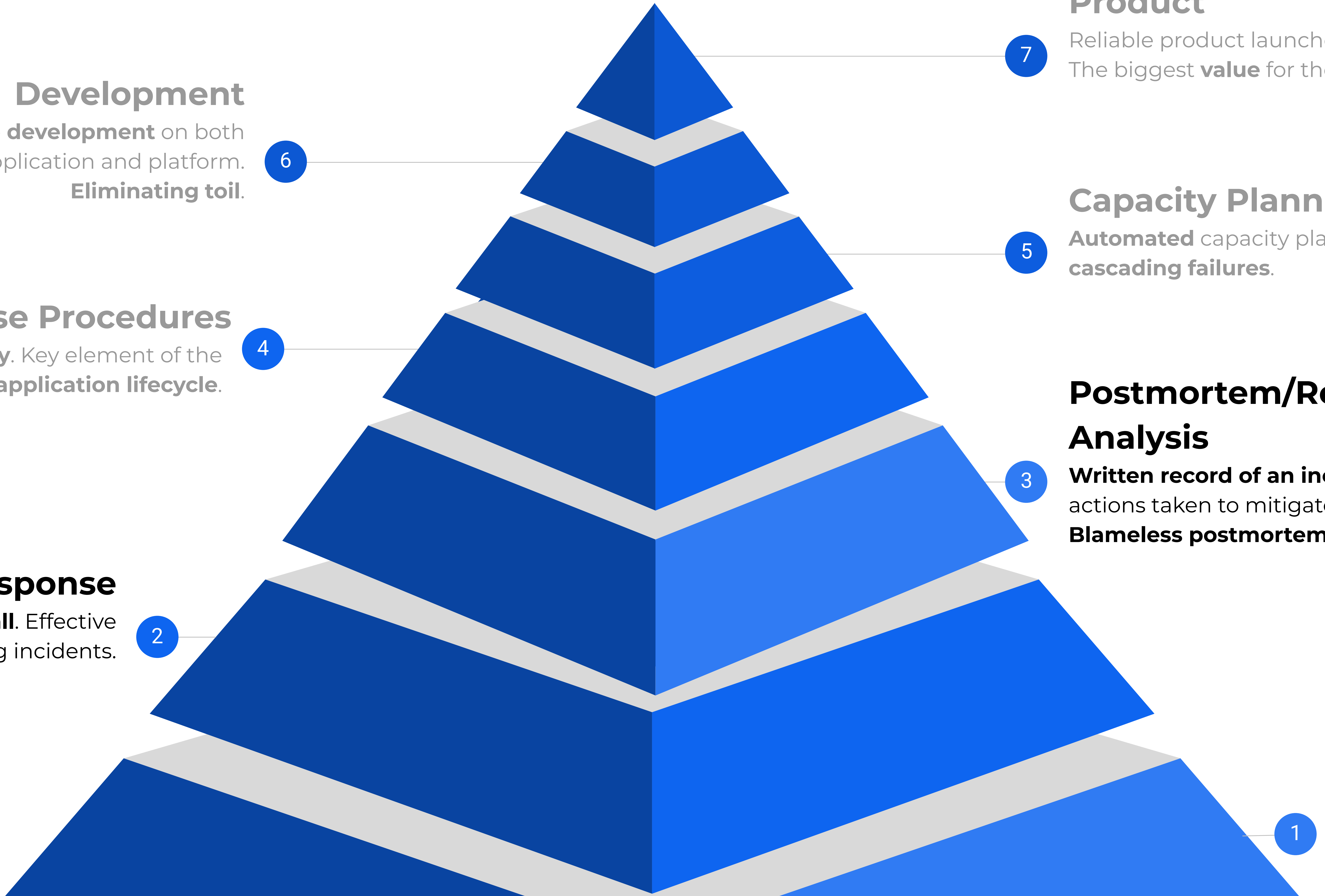
Monitoring

Whitebox & blackbox monitoring

The four golden signals:

1. Latency
2. Traffic
3. Errors
4. Saturation

Service Reliability Hierarchy



Product
7 Reliable product launches..
The biggest **value** for the **business**.

Development
6 **Software development** on both the application and platform. **Eliminating toil**.

Capacity Planning
5 **Automated** capacity planning. Addressing **cascading failures**.

Testing & Release Procedures
4 Testing for **reliability**. Key element of the **application lifecycle**.

Postmortem/Root Cause Analysis
3 **Written record of an incident**, the impact and actions taken to mitigate and resolve the issue..
Blameless postmortem culture

Incident Response
2 **Embracing risk**. Being **on-call**. Effective **troubleshooting**. Managing incidents.

Monitoring
1 **SLI, SLO** & **SLA**.
The four golden signals.

Being On-Call

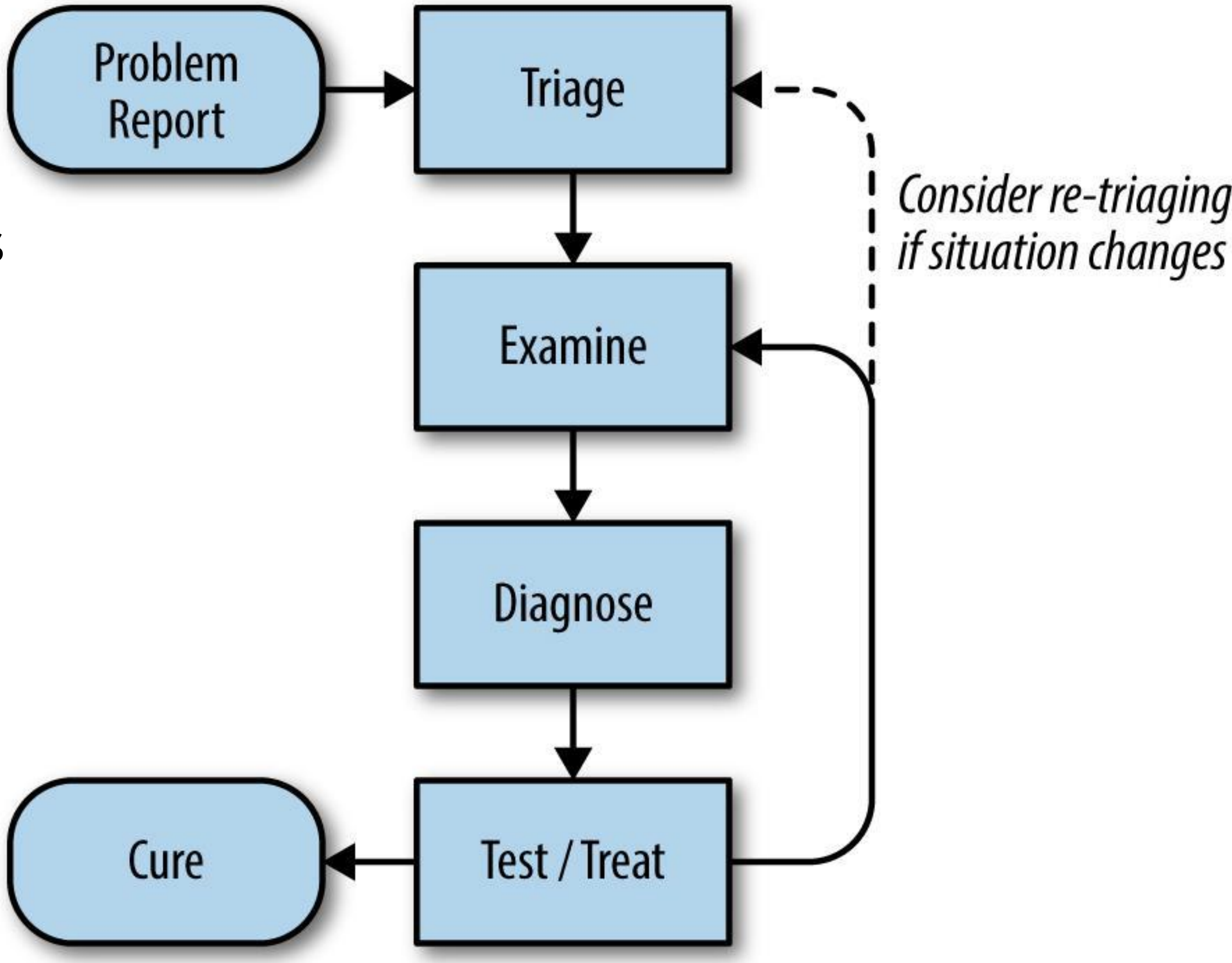
Life of an on-call engineer:

- 99,99% availability is less than 5 minutes of allowed downtime per month;
- Feeling safe;
- Balanced on-call;
- Compensation for being on-call.

Effective Troubleshooting

Your first response in a major outage may be to start troubleshooting and try to find a root cause as quickly as possible…

Ignore that instinct!



Emergency Response

“Things break; that’s life! At the very worst, half of the Internet is down. So take a deep breath…and carry on.”

What to do when systems break:

- Don’t panic
- Pull in others if necessary
- Learn from the past

Managing Incidents

- Separation of responsibilities:
  - *Incident Command*
  - *Operational Work*
  - *Communications*
- A recognized command post
- Live incident state document
- Clear, live handoff

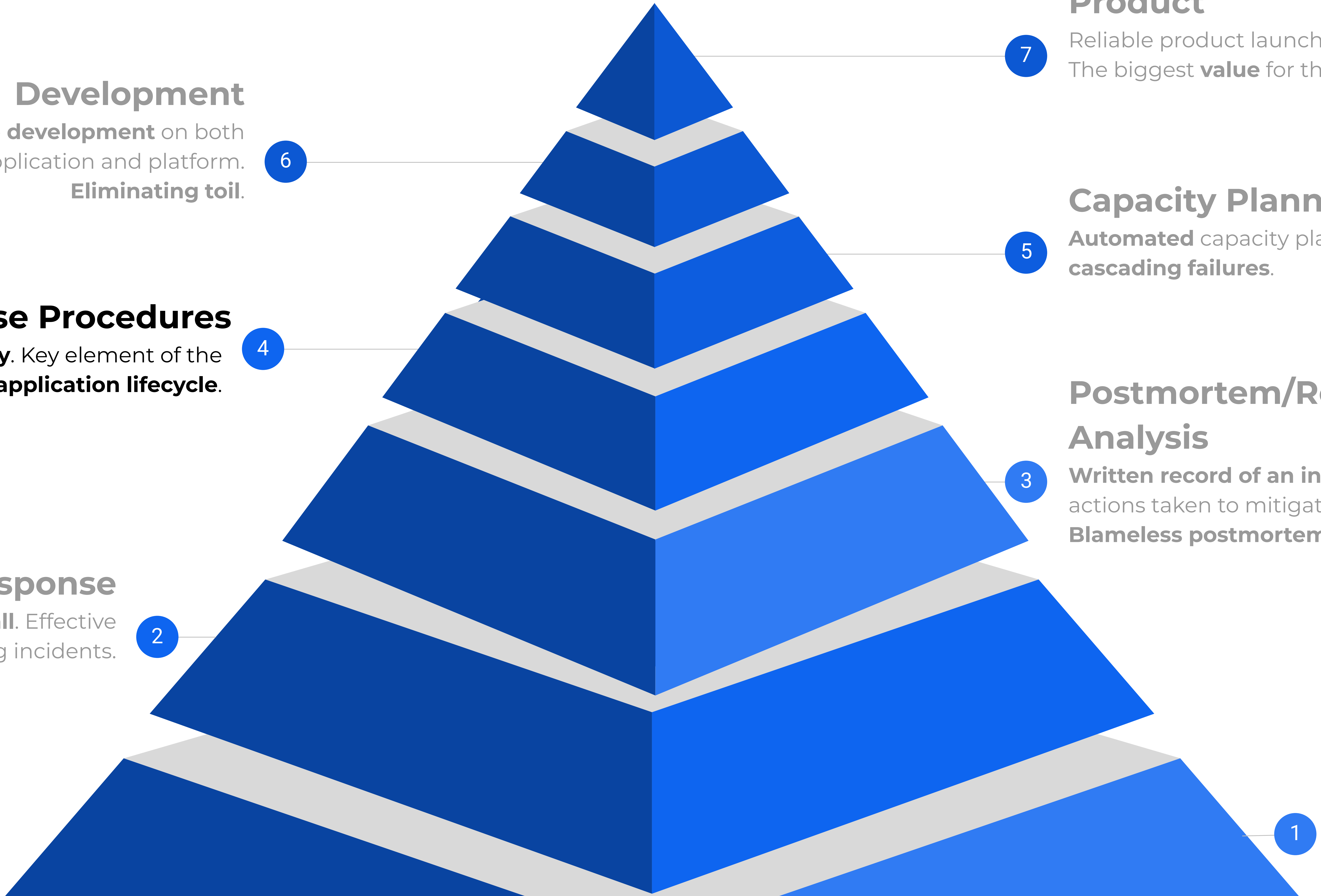
Tracking Outages

A single event may, and often will, trigger multiple alerts!

Tracking outages:

- Aggregation
- Tagging
- Analysis

Service Reliability Hierarchy



Product

Reliable product launches..
The biggest **value** for the **business**.

Development

Software development on both the application and platform. **Eliminating toil**.

Capacity Planning

Automated capacity planning. Addressing **cascading failures**.

Testing & Release Procedures

Testing for **reliability**. Key element of the **application lifecycle**.

Postmortem/Root Cause Analysis

Written record of an incident, the impact and actions taken to mitigate and resolve the issue..
Blameless postmortem culture

Incident Response

Embracing risk. Being **on-call**. Effective **troubleshooting**. Managing incidents.

Monitoring

SLI, SLO & **SLA**.
The four golden signals.

Testing for Reliability

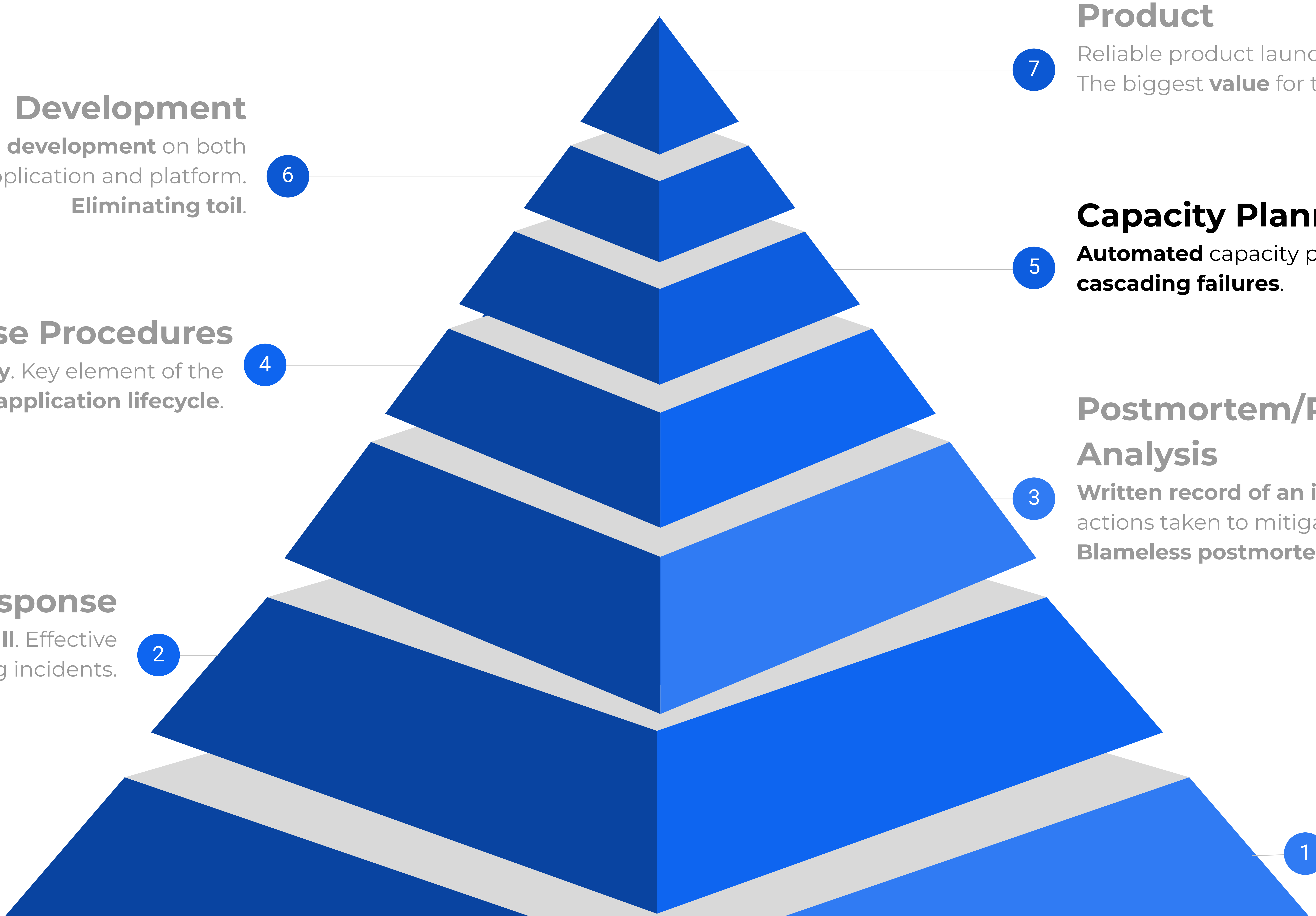
Types of software testing:

- Unit tests
- Integration tests
- System tests

Types of production tests:

- Configuration test
- Stress test
- Canary test

Service Reliability Hierarchy



Product

Reliable product launches..
The biggest **value** for the **business**.

Development

Software development on both the application and platform. **Eliminating toil**.

Capacity Planning

Automated capacity planning. Addressing **cascading failures**.

Testing & Release Procedures

Testing for **reliability**. Key element of the **application lifecycle**.

Postmortem/Root Cause Analysis

Written record of an incident, the impact and actions taken to mitigate and resolve the issue.. **Blameless postmortem** culture

Incident Response

Embracing risk. Being **on-call**. Effective **troubleshooting**. Managing incidents.

Monitoring

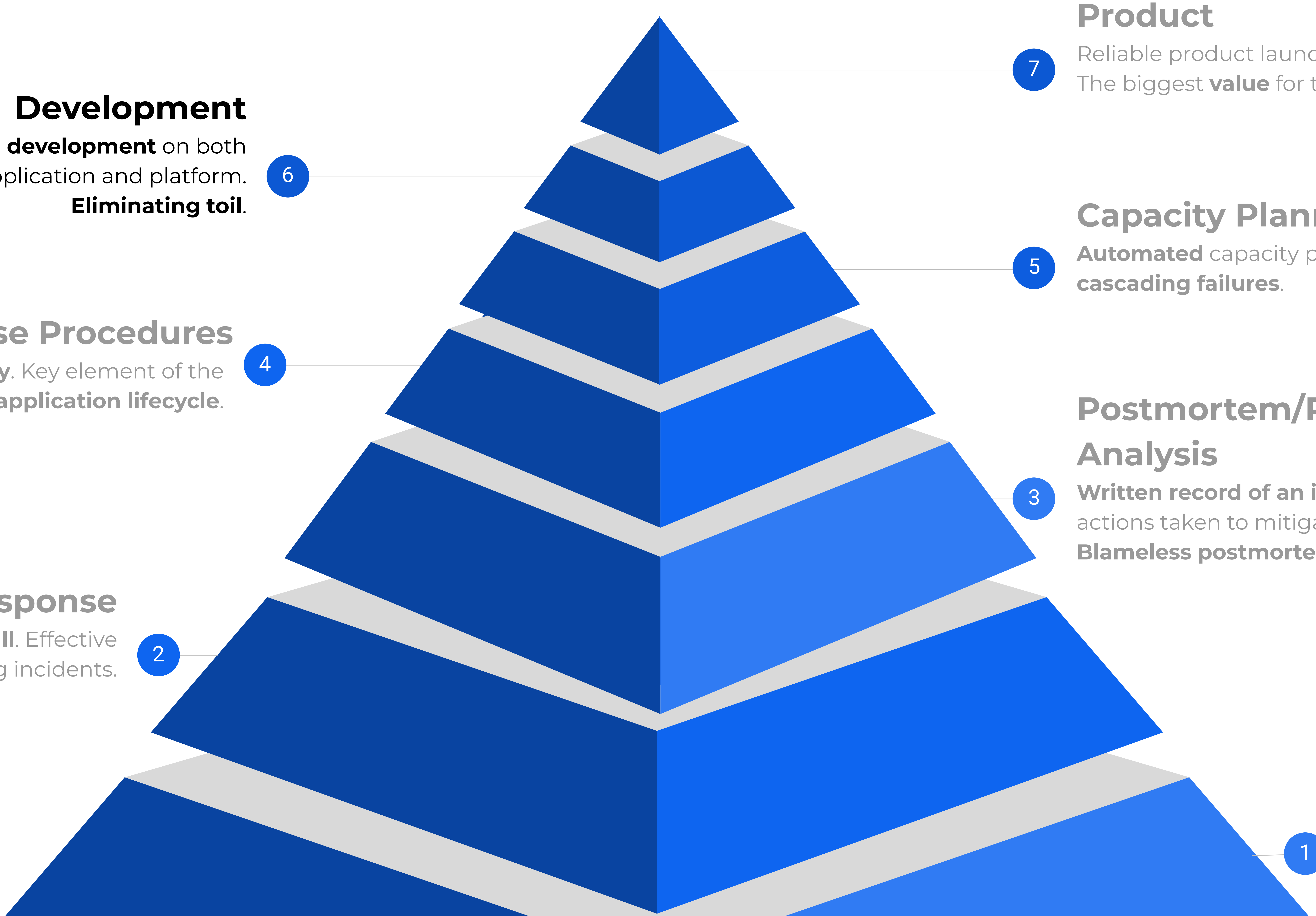
SLI, SLO & **SLA**.
The four golden signals.

Addressing Cascading Failures

Immediate steps to address cascading failures:

- Increase resources
- Stop health check failure/deaths
- Restart servers
- Drop traffic
- Enter degraded mode
- Eliminate batch load
- Eliminate bad traffic

Service Reliability Hierarchy



Product
Reliable product launches..
The biggest **value** for the **business**.

Development
Software development on both the application and platform.
Eliminating toil.

Capacity Planning
Automated capacity planning. Addressing **cascading failures**.

Testing & Release Procedures
Testing for **reliability**. Key element of the **application lifecycle**.

Postmortem/Root Cause Analysis
Written record of an incident, the impact and actions taken to mitigate and resolve the issue..
Blameless postmortem culture

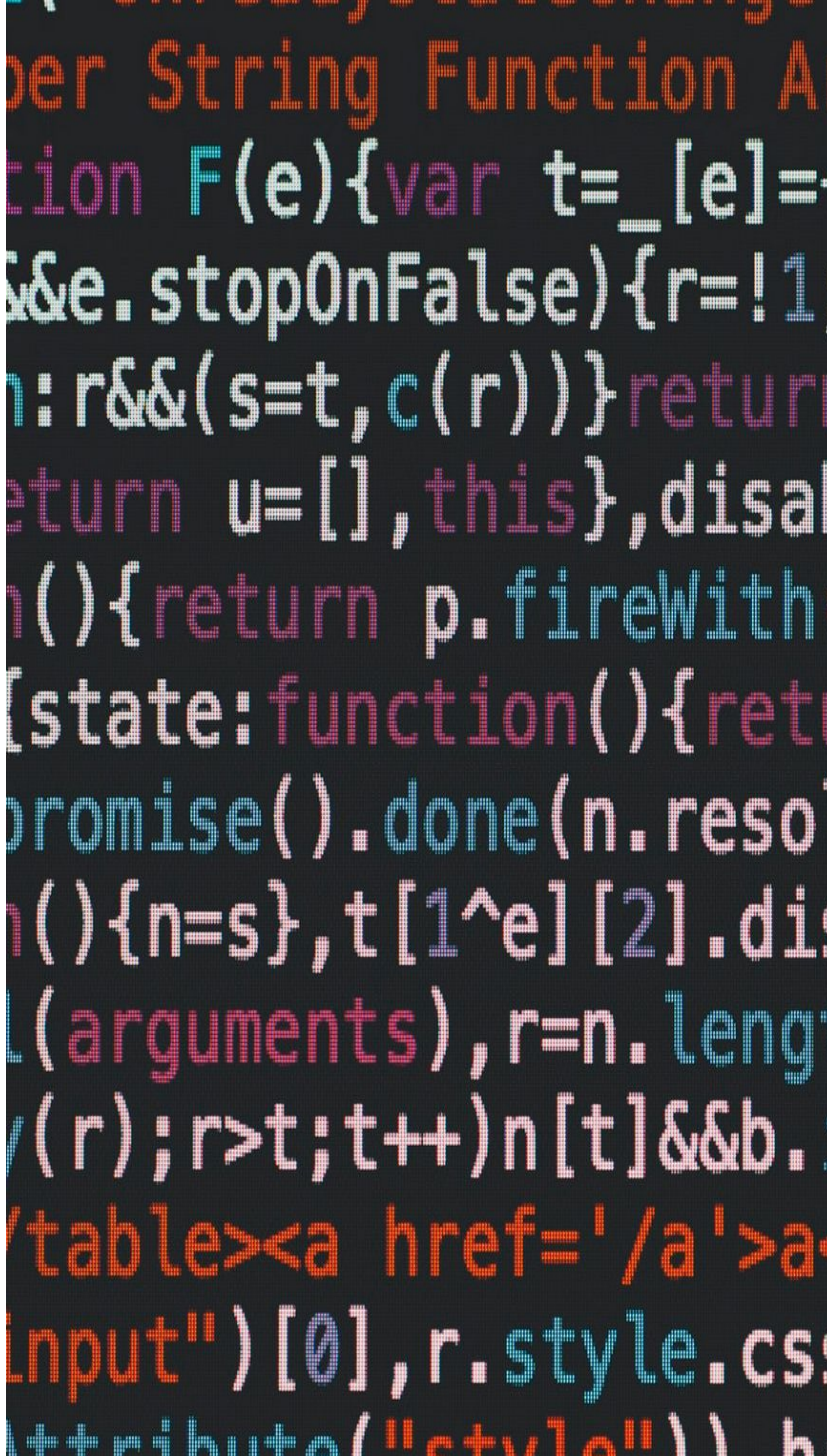
Incident Response
Embracing risk. Being **on-call**. Effective **troubleshooting**. Managing incidents.

Monitoring
SLI, SLO & **SLA**.
The four golden signals.

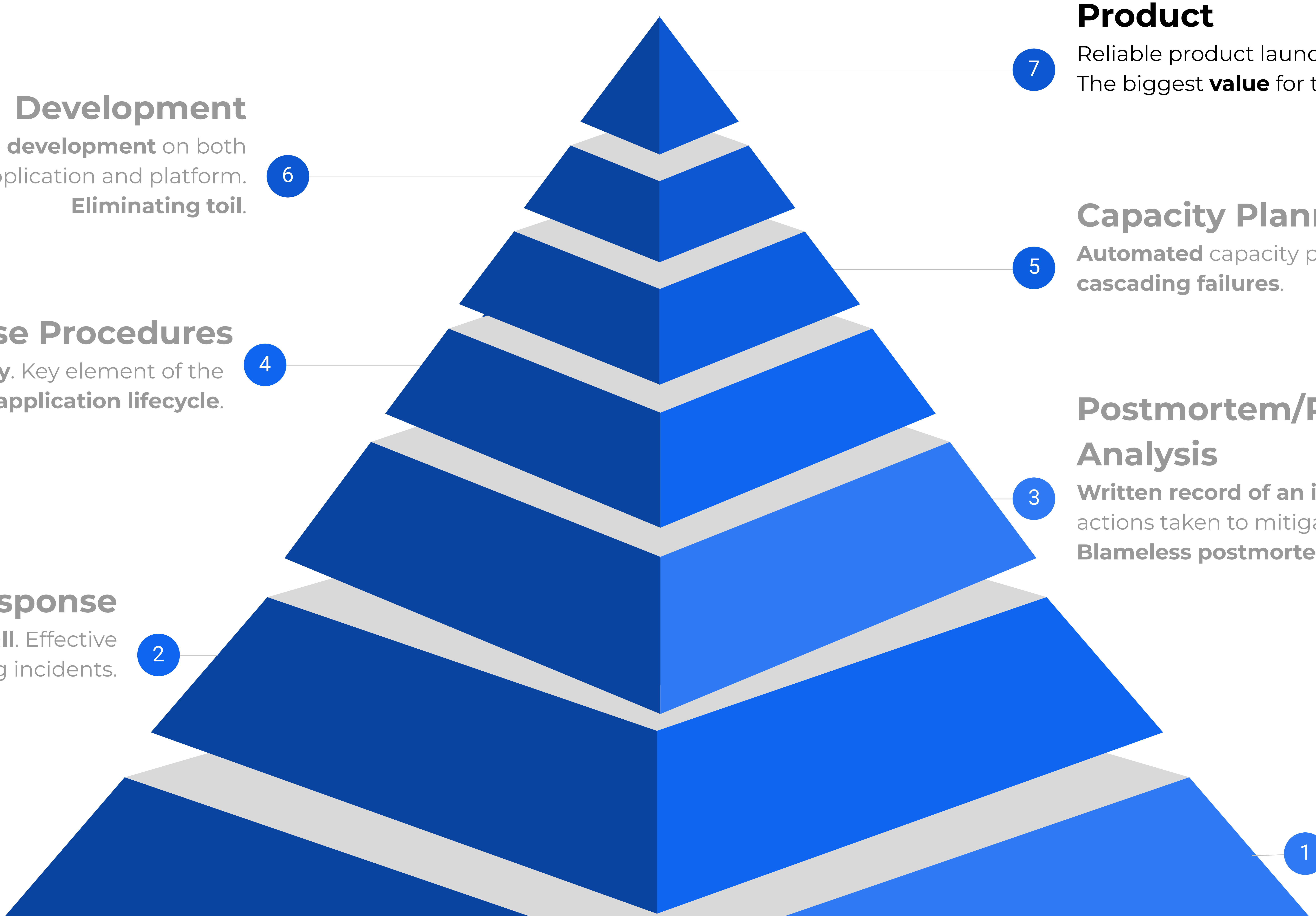
Testing for Reliability

Develop software to streamline inefficient processes of automate common tasks:

SRE teams doesn’t have to scale linearly!



Service Reliability Hierarchy



Product

7 Reliable product launches..
The biggest **value** for the **business**.

Development

6 **Software development** on both the application and platform. **Eliminating toil**.

Capacity Planning

5 **Automated** capacity planning. Addressing **cascading failures**.

Testing & Release Procedures

4 Testing for **reliability**. Key element of the **application lifecycle**.

Postmortem/Root Cause Analysis

3 **Written record of an incident**, the impact and actions taken to mitigate and resolve the issue.. **Blameless postmortem** culture

Incident Response

2 **Embracing risk**. Being **on-call**. Effective **troubleshooting**. Managing incidents.

Monitoring

1 **SLI, SLO** & **SLA**.
The four golden signals.

Testing for Reliability

- Launch Coordination Engineering
- The launch checklist:
  - *Capacity planning*
  - *Rollout planning*
  - *[...]*
- Gradual and staged rollouts:
  - *Canary in a coal mine*

questions

introduction

principles

practices