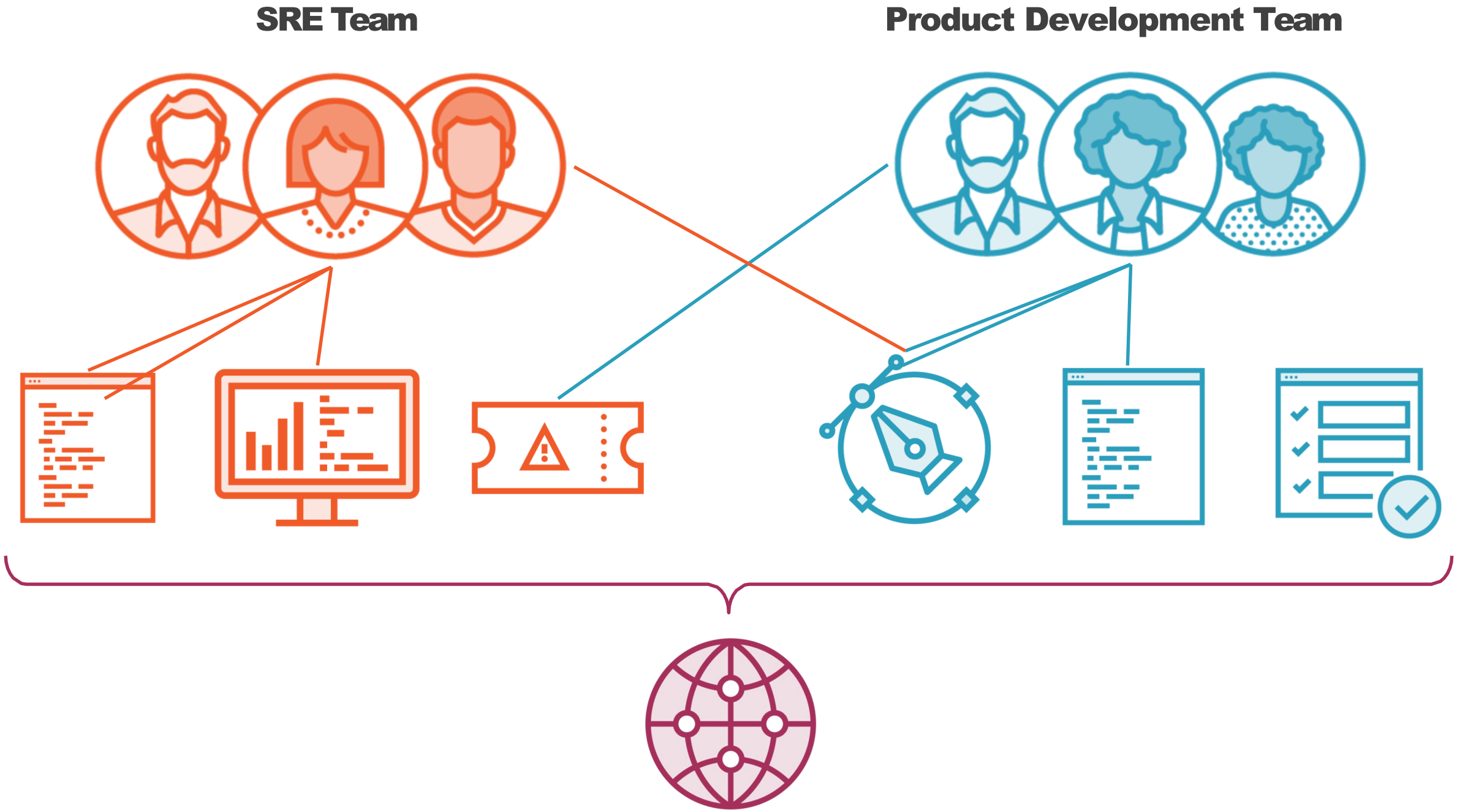
Site Reliability Engineering (SRE): The Big Picture

INTRODUCING SITE RELIABILITY ENGINEERING

What is SRE?

An approach to operations which uses software as the primary tool for managing systems


SRE Team
Product Development Team

Functions of Site Reliability Engineering

Eliminating Toil

Automating manual, repetitive work

Managing Risk

Agreed service levels with explicit tolerance

Handling Failure

Incident management and post-mortems

How Google runs production systems

- Engineering approach to ops, 2003
- *Site Reliability Engineering*, 2016
- *The Site Reliability Workbook*, 2018

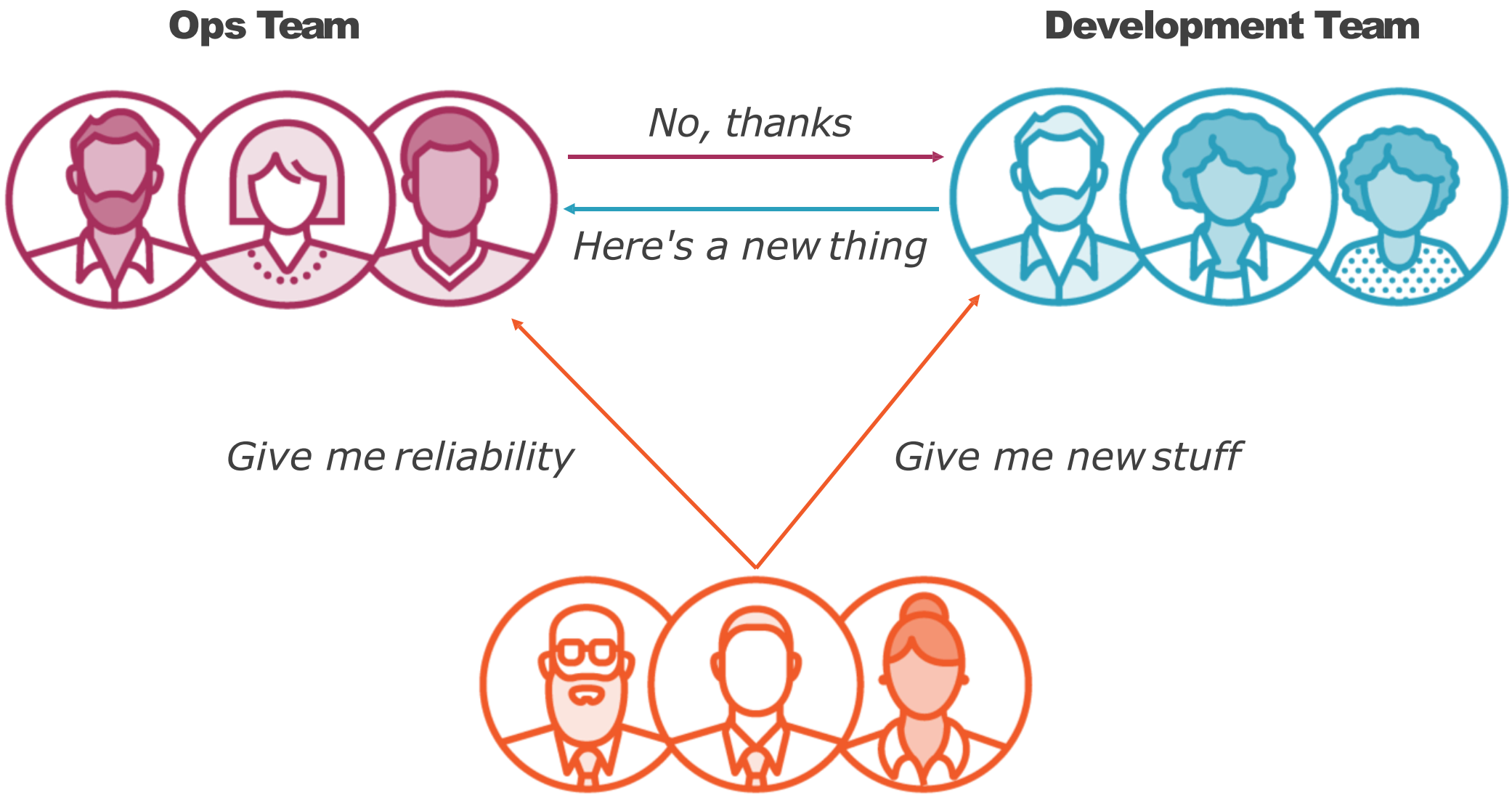
Industry-recognized role

- @SREcon
- Google, Netflix, Amazon

Viable transition from "ops"

- Solves dev vs. ops problems
- Without the upheavel of DevOps

Comparing Traditional Ops and SRE


Ops Team
Development Team
No, thanks
Here's a new thing
Give me reliability
Give me new stuff
"The Business"

- Different goals
- Different skillsets
- Different tools
- No common ground

Development Team

Ops Team

Design

Development

Testing

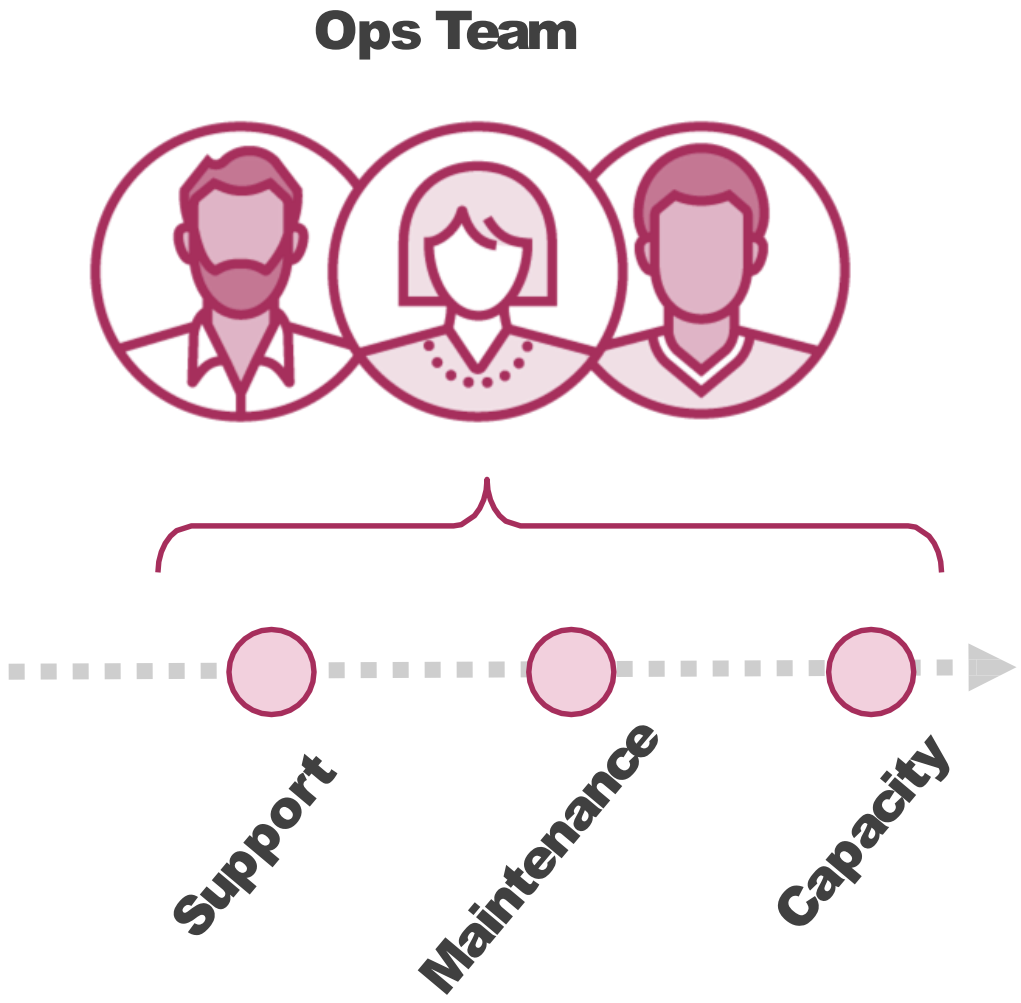
Acceptance

Delivery

Support

Maintenance

Capacity



Black box delivery

- No design input
- Automation options limited
- Infrastructure-level monitoring

Limited agency

- Post-acceptance
- Go or no-go decision
- Conflict guaranteed

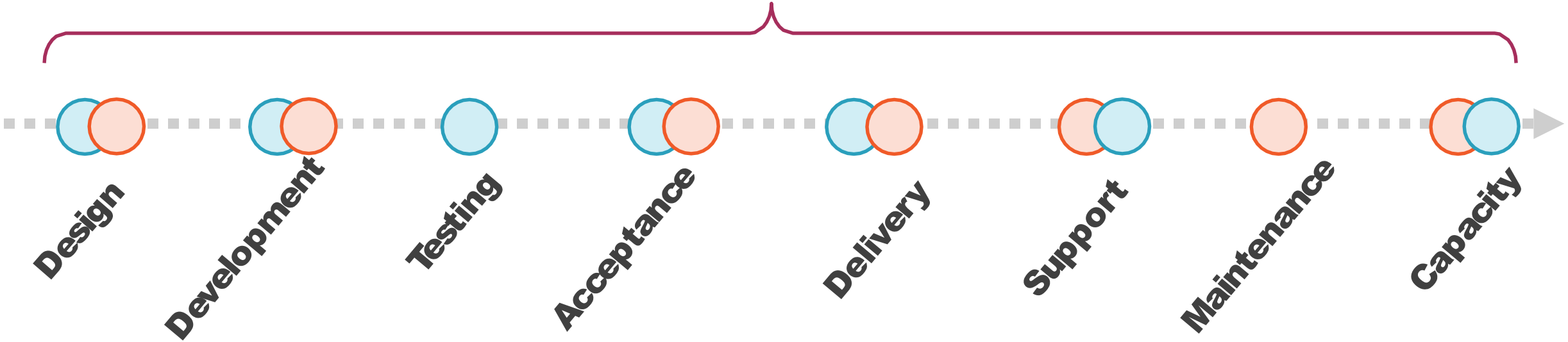
After-the-event

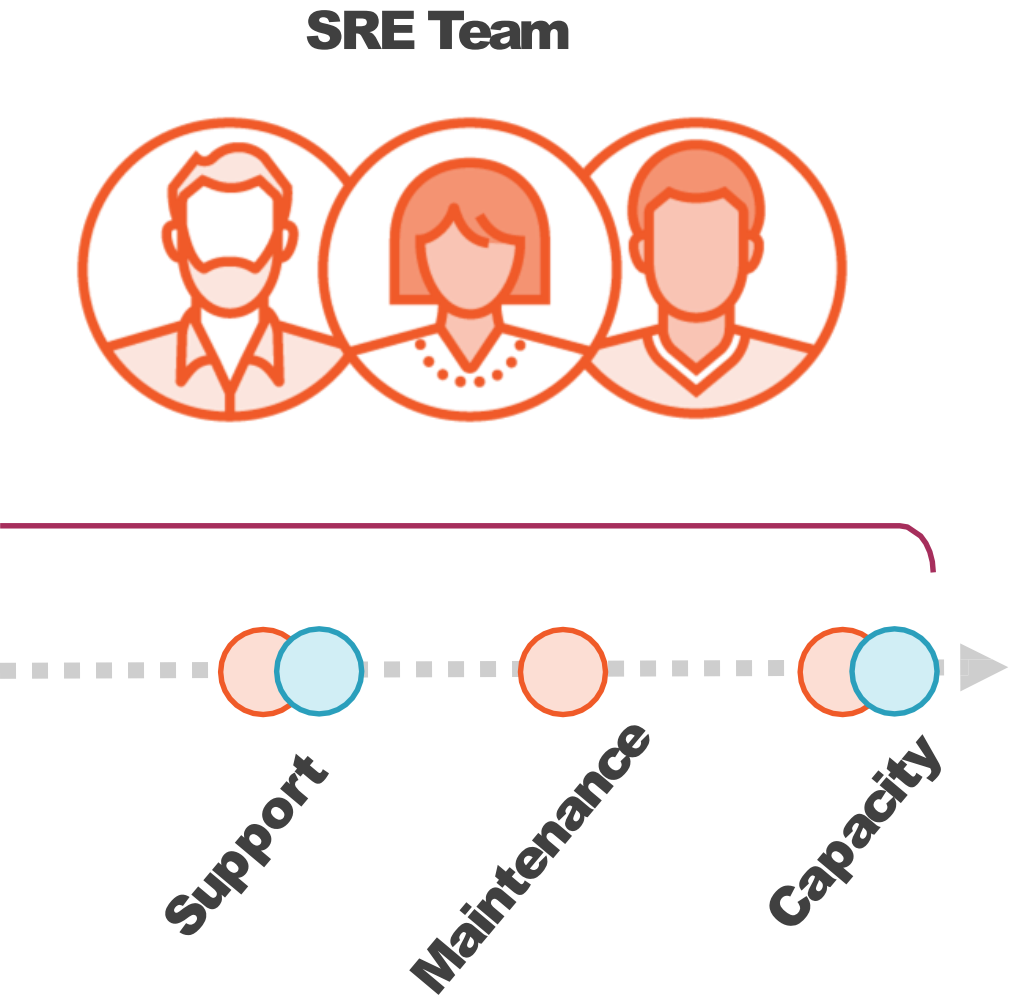
- Mandated service levels
- Reactive support

- Shared goal
- Overlapping skillsets
- Consistent tools
- Common basis

Development Team

SRE Team





Agreed delivery

- Ongoing design input
- Extended automation options
- User-focused monitoring

Full agency

- Dev team support
- Stop deployments
- Hand back the pager

After-the-event

- Agreed service levels
- Limited support time

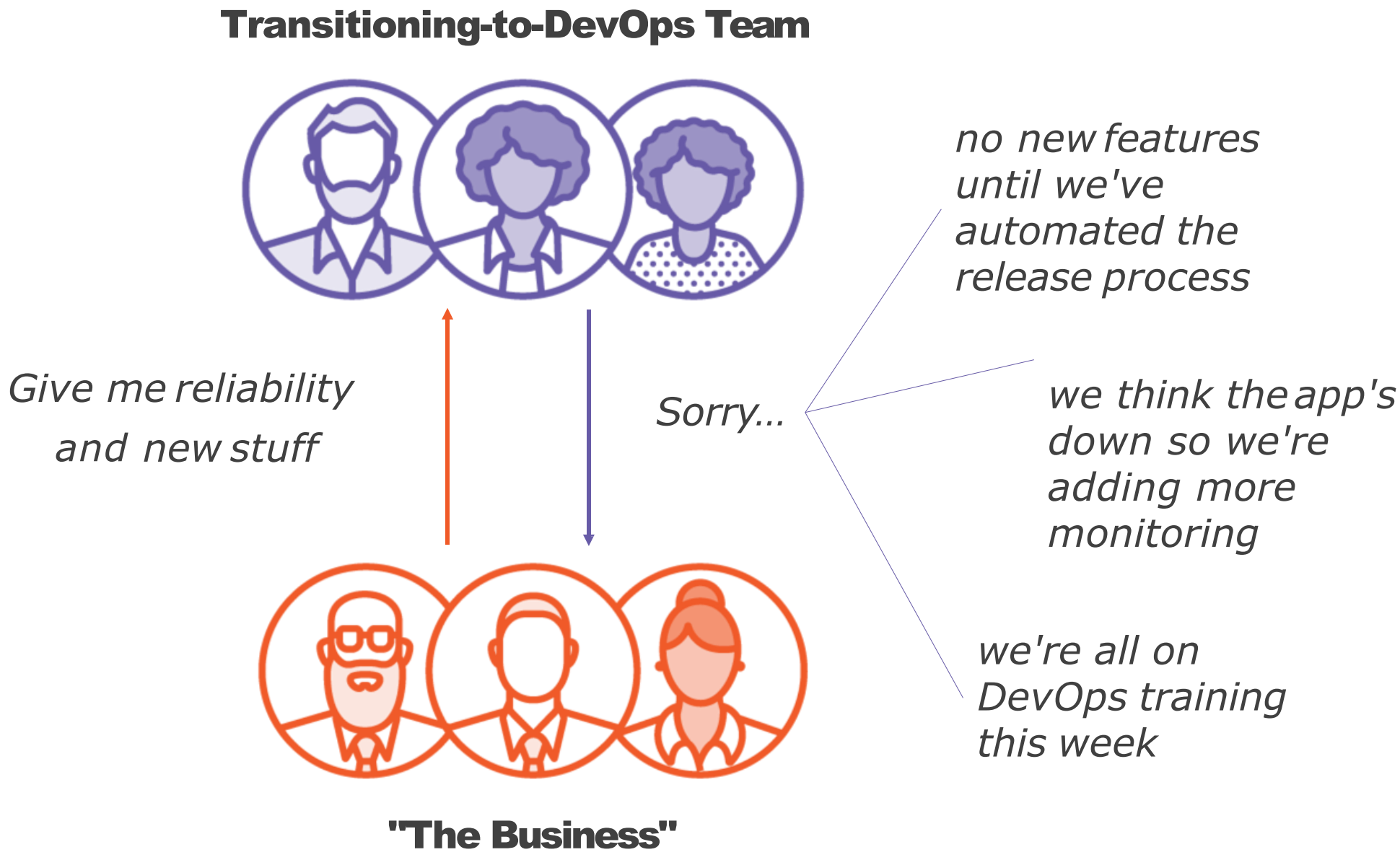
Comparing DevOps and SRE

DevOps Team

Give me reliability and new stuff

OK

"The Business"


Transitioning-to-DevOps Team
Give me reliability and new stuff
Sorry...
no new features until we've automated the release process
we think the app's down so we're adding more monitoring
we're all on DevOps training this week
"The Business"

- Shared goal
- Overlapping skillsets
- Consistent tools
- Common basis

DevOps Team

Design

Development

Testing

Acceptance

Delivery

Support

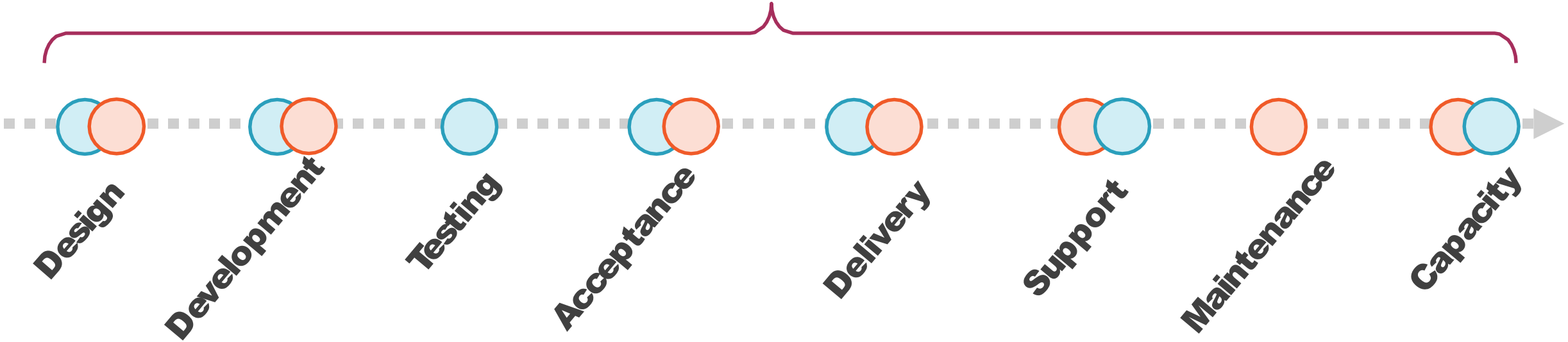
Maintenance

Capacity

- Shared goal
- Overlapping skillsets
- Consistent tools
- Common basis

Development Team

SRE Team



SRE Team

DevOps Team

Much in common

- Focus on quality &velocity
- Automation & tooling
- Removing silos

But DevOps...

- Has a broader remit
- Significant cultural change

And SRE...

- Is more prescriptive
- Feasible internal migration

Development Team

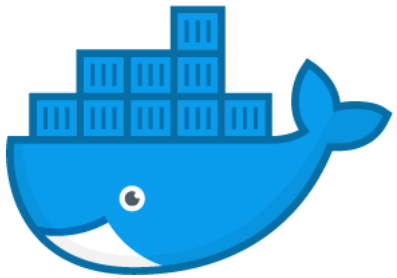
Ops Team

DevOps Team

- Significant investment
- Acceptance of change
- Needs outside help
- Uneven rollout

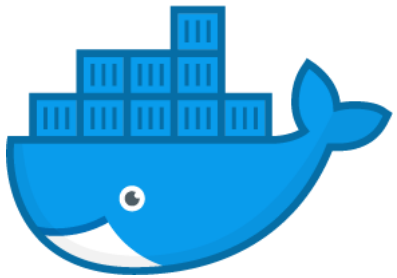

Development Team
Ops Team
Development Team
SRE Team

DevOps Team



Development Team

SRE Team



Exploring the Key Tenets of SRE

Eliminating Toil

Working to Service Levels

Managing Failure

Log in weekly; delete old log files

New release; logs fill disk daily

Script to delete old log files; log in and run daily

Scale up to 50 servers

CRON job to run script daily

Work with dev team to reduce logs

Service level maintained

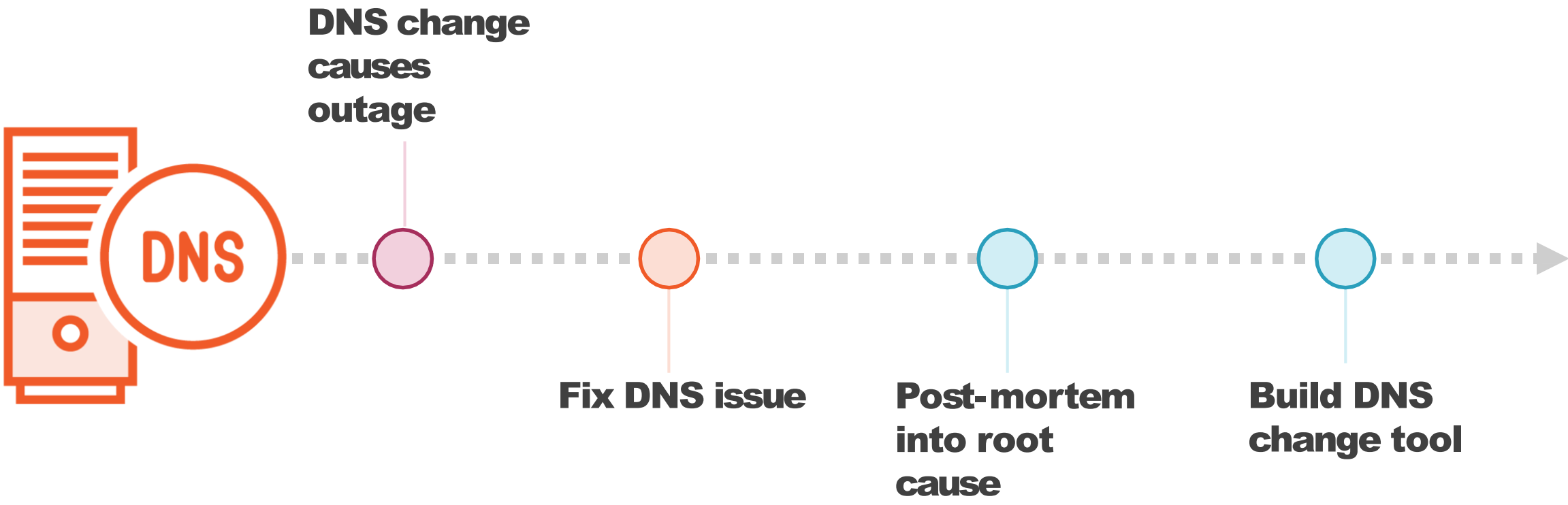
New release; bugs cause latency

Increase capacity; report issues to dev team

New release; config issue causes outage

Fix config and restore service

Block new releases for agreed period


DNS change causes outage
DNS
Fix DNS issue
Post-mortem into root cause
Build DNS change tool

Understanding Why SRE Works

vs.

Conflict between dev and ops

- Made explicit
- Contracted service level
- Business buy-in

Integration between dev and ops

- Shared responsibility
- Shared workload
- Common toolkit

Prescriptive practices

- On-call teams and targets
- %age SRE time on toil
- Service level framework

IT-led migration

- Within existing org chart
- Maintain product knowledge
- Start small

SRE empowerment

- Improving systems
- Formal structure
- Tools to meet responsibilities

SRE as an attractive role

- Reduce mundane work
- Expand skills
- Career development

Miles Bryant
@milesbxf

We run an on-call rota @monzo that is so popular we have a waiting list of people looking to join it. If you'd like to hear more, I'm really looking forward to speaking about it at @SREcon 20 Americas West 👉

- Monzo, UK startup bank

“It's what happens when you ask a software engineer to design an operations function.”

Ben Treynor Sloss, VP Engineering, Founder of Google SRE

Summary

SRE is an approach to IT operations

- Maintains a dev/ops split
- But adds engineering to "ops"

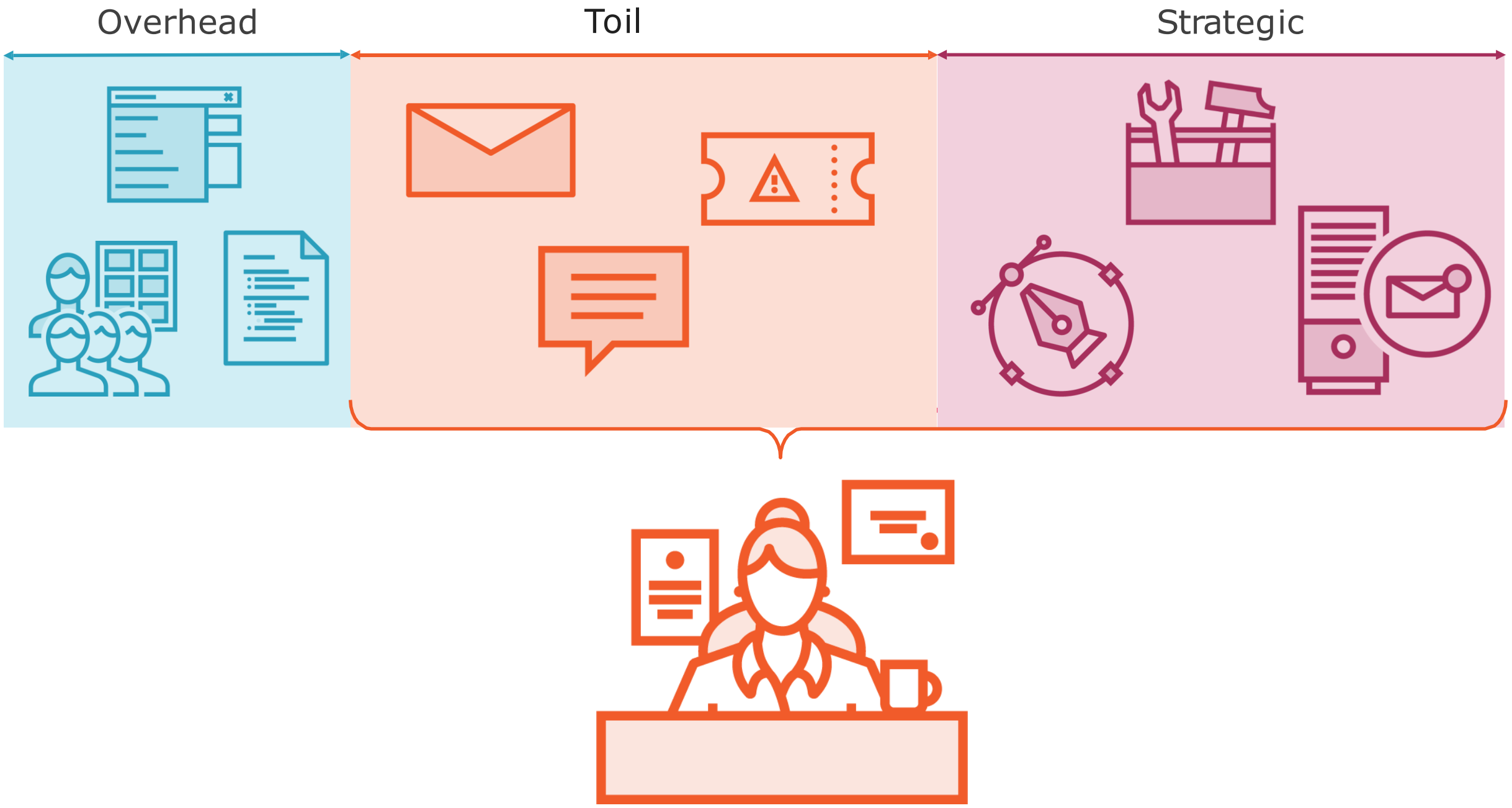
Shares goals with DevOps

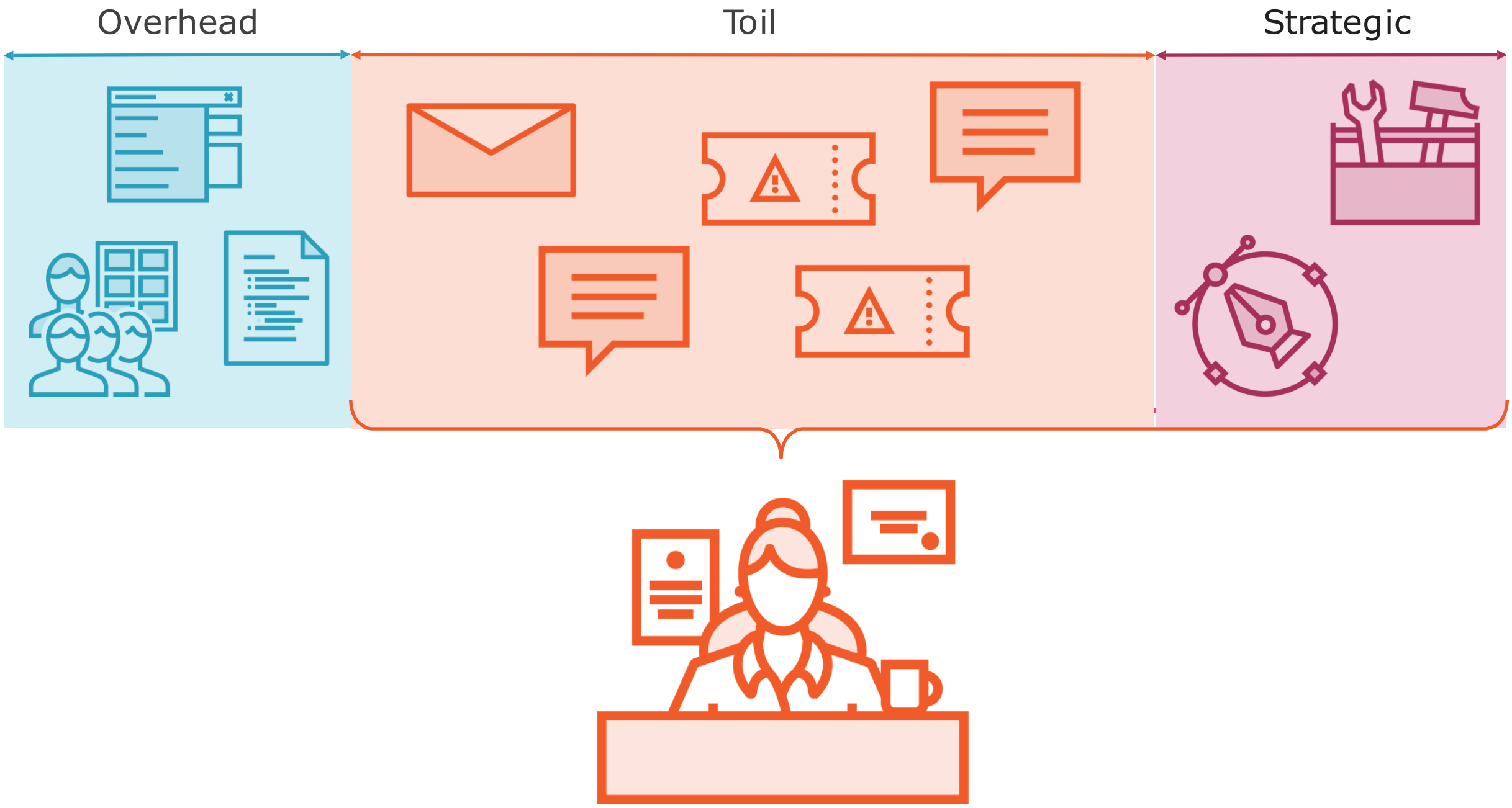
- Breaking down silos
- Increasing quality and velocity

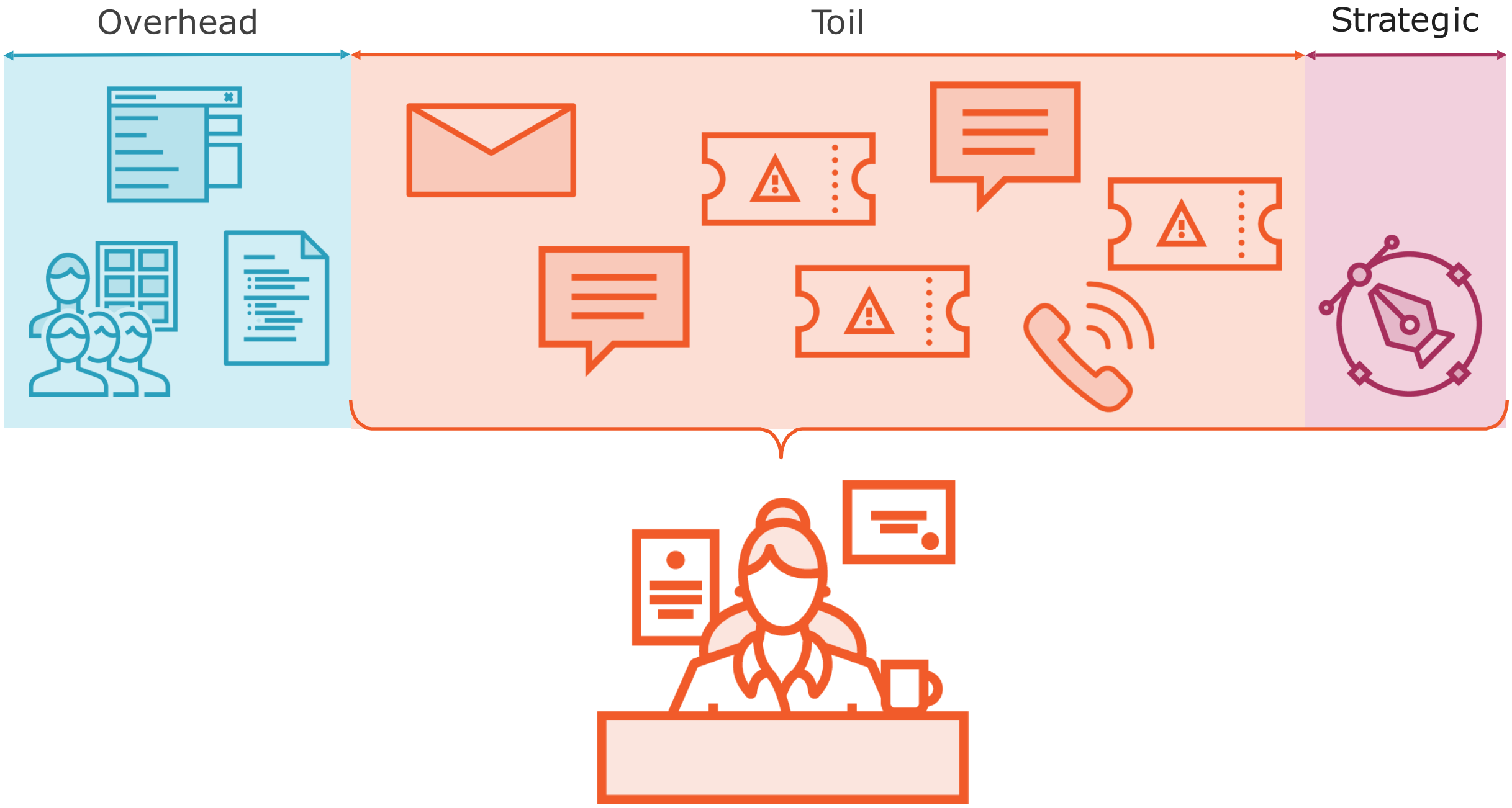
Growing adoption

- '00s job roles
- Sympathetic migration

Automation and Eliminating Toil

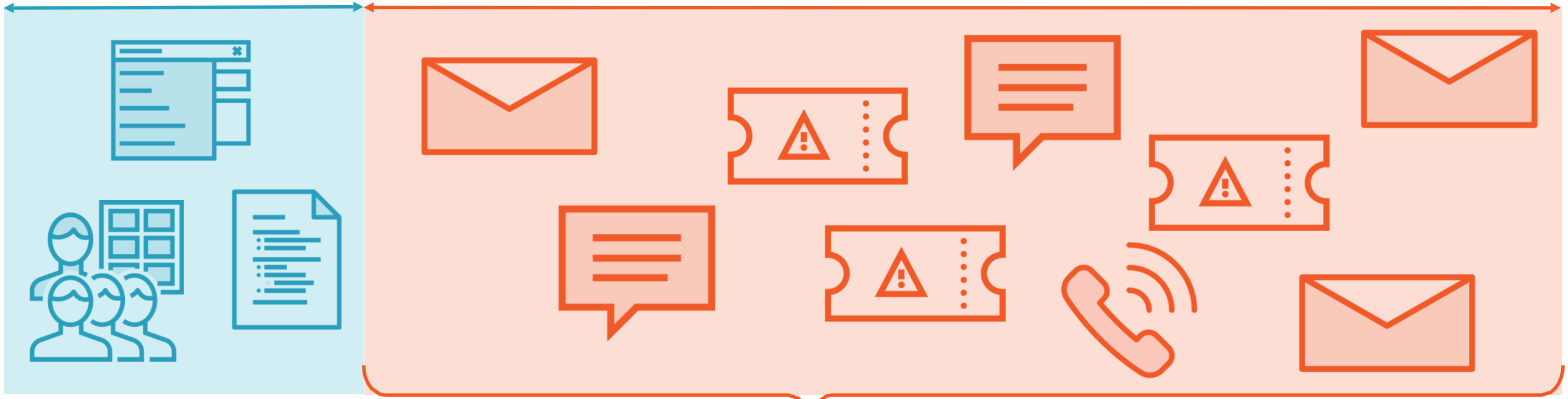

Overhead
Toil
Strategic

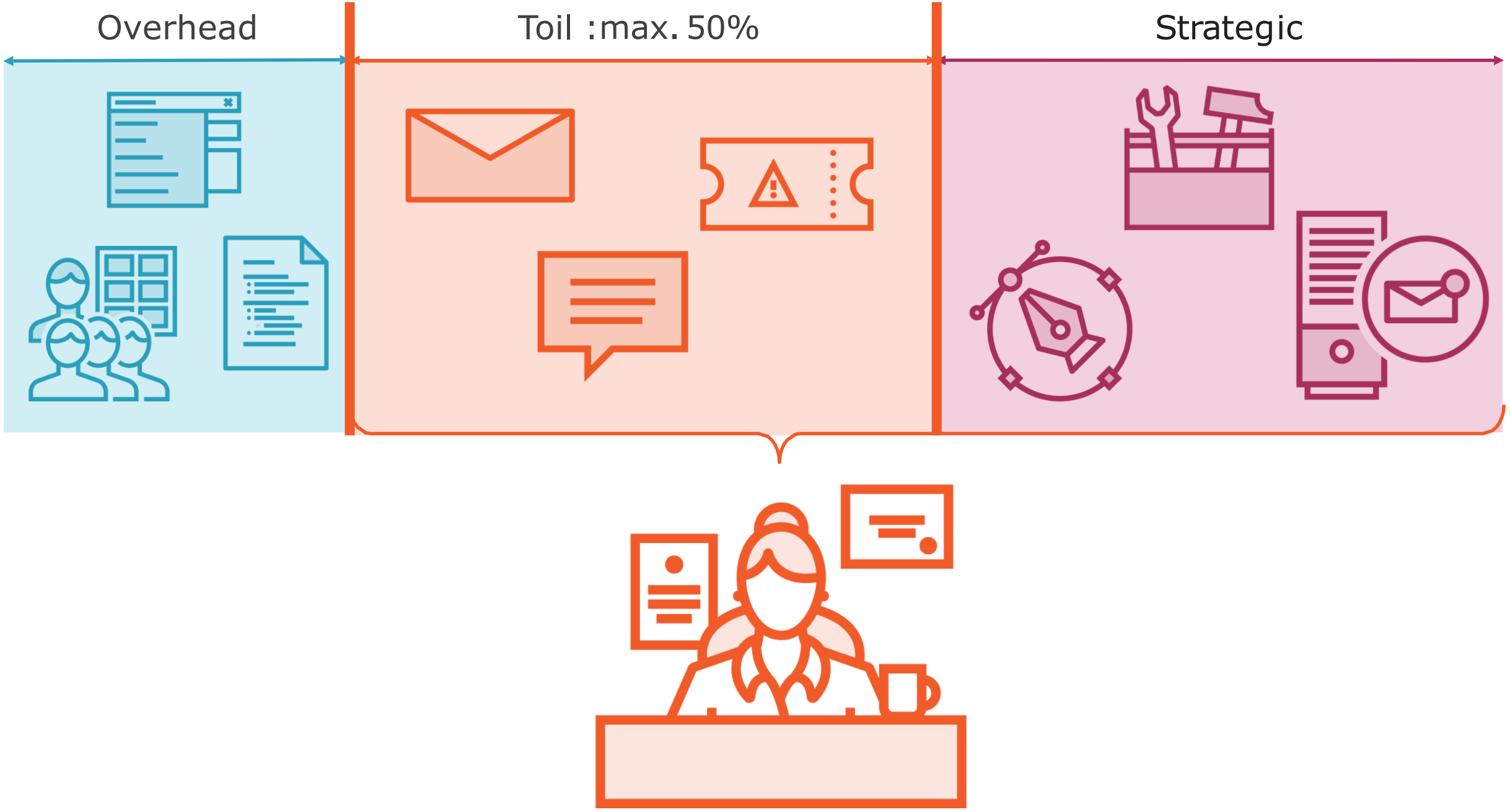

Overhead
Toil
Strategic


Overhead
Toil
Strategic

Overhead

Toil

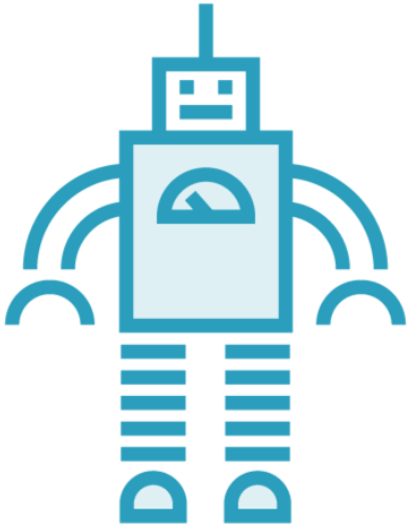



Overhead
Toil :max. 50%
Strategic

What is Toil?

Labor Intensive

Repetitive



Automatable

App not responding

Raise ticket

Log in to server

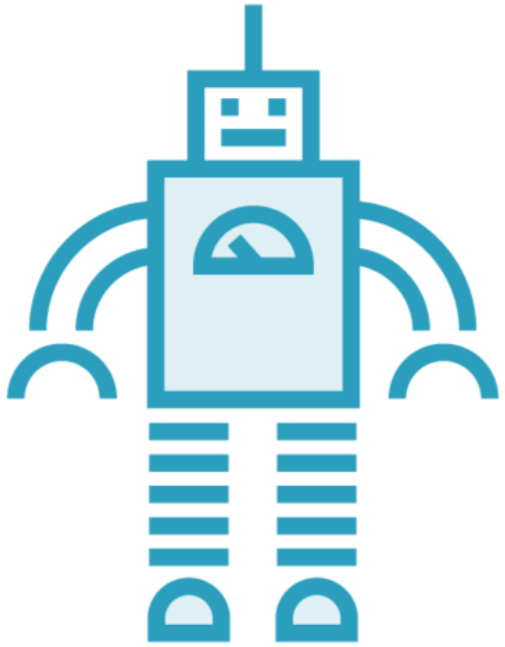
Restart process

- Manual
- Repetitive
- Automatable

- Reactive
- Low-value
- Scales linearly

Is it Toil?

"I can do this!"



"Me too! Bloop."

Engineering in SRE

Strategic

High Value

Human

Engineering in SRE

Systems Engineering

Configuration

Tuning

Software Engineering

Reliability

Scalability

- Runtime change
- Use platform capabilites
- Doesn't fix the issue

App not responding

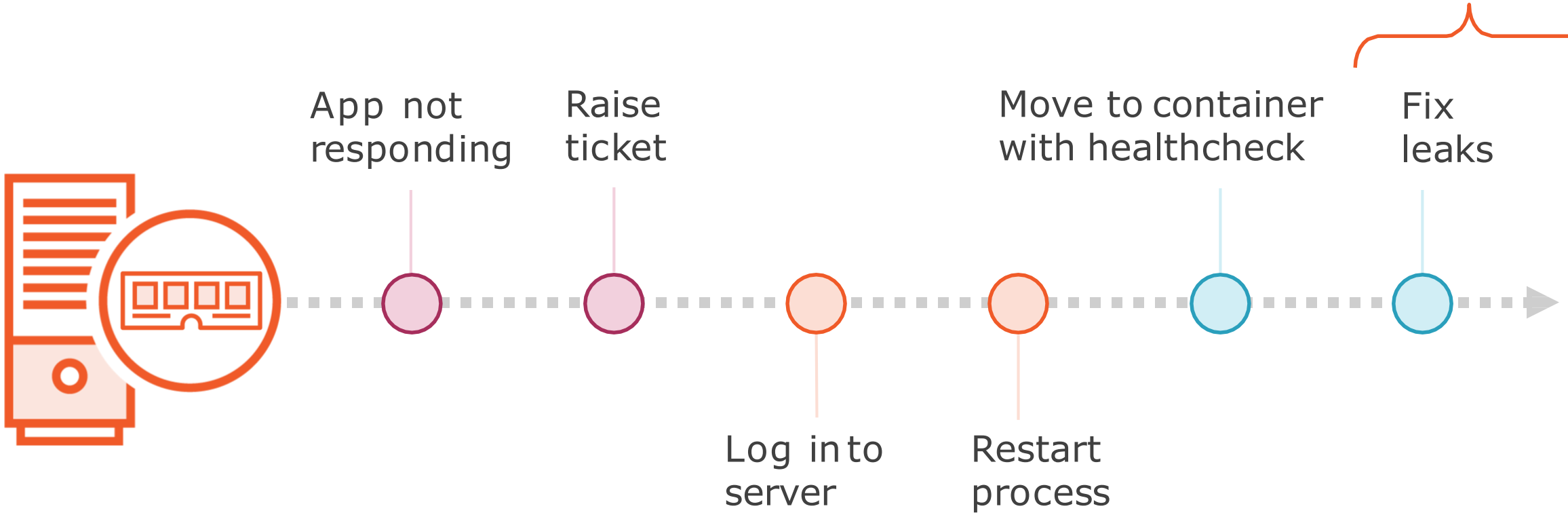
Raise ticket

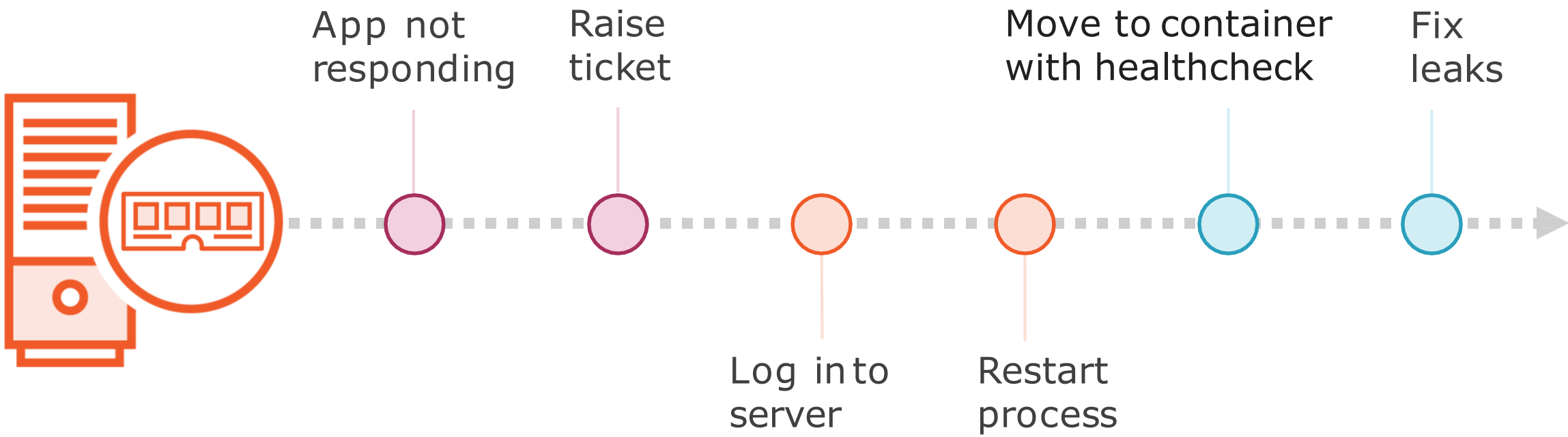
Log in to server

Restart process

Move to container with healthcheck

- Permanent fix
- More work
- Introduces risk




App not responding
Raise ticket
Log in to server
Restart process
Move to container with healthcheck
Fix leaks

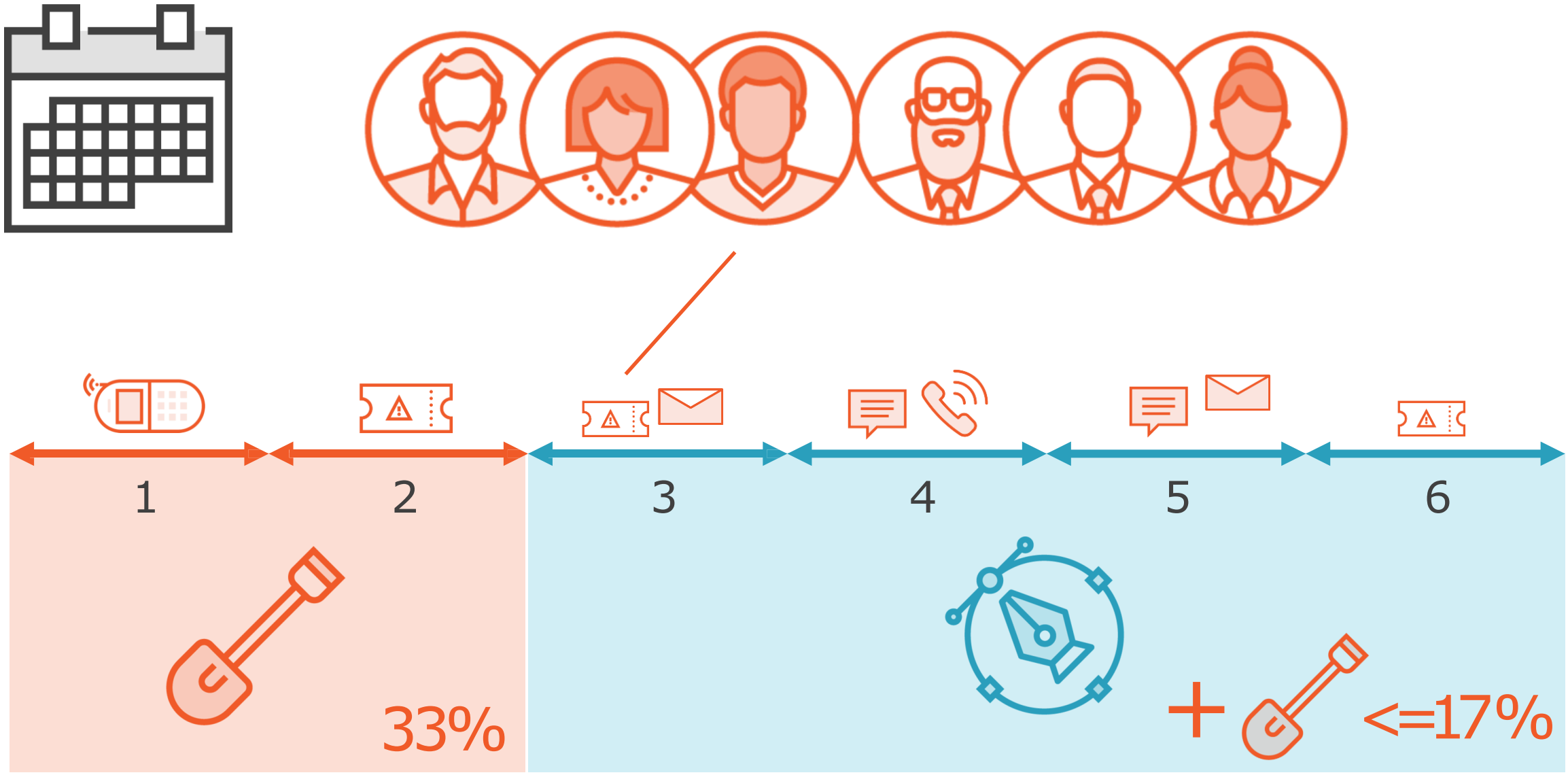
Restricting Toil to 50%

Stated limit

- Google use 50%
- Publicly communicated
- Commitment to the team

Toil guarantee

- Part of the job spec
- Management support
- Clear difference from ops


1
2
3
4
5
6
33%
+
<=17%

Toil can be attractive

- Quick fixes
- Sense of achievement
- But it's short-term

And it's bad for morale

- Team's energy dips
- Strategic projects stall
- People move on

Identifying and Measuring Toil

Data-driven analysis

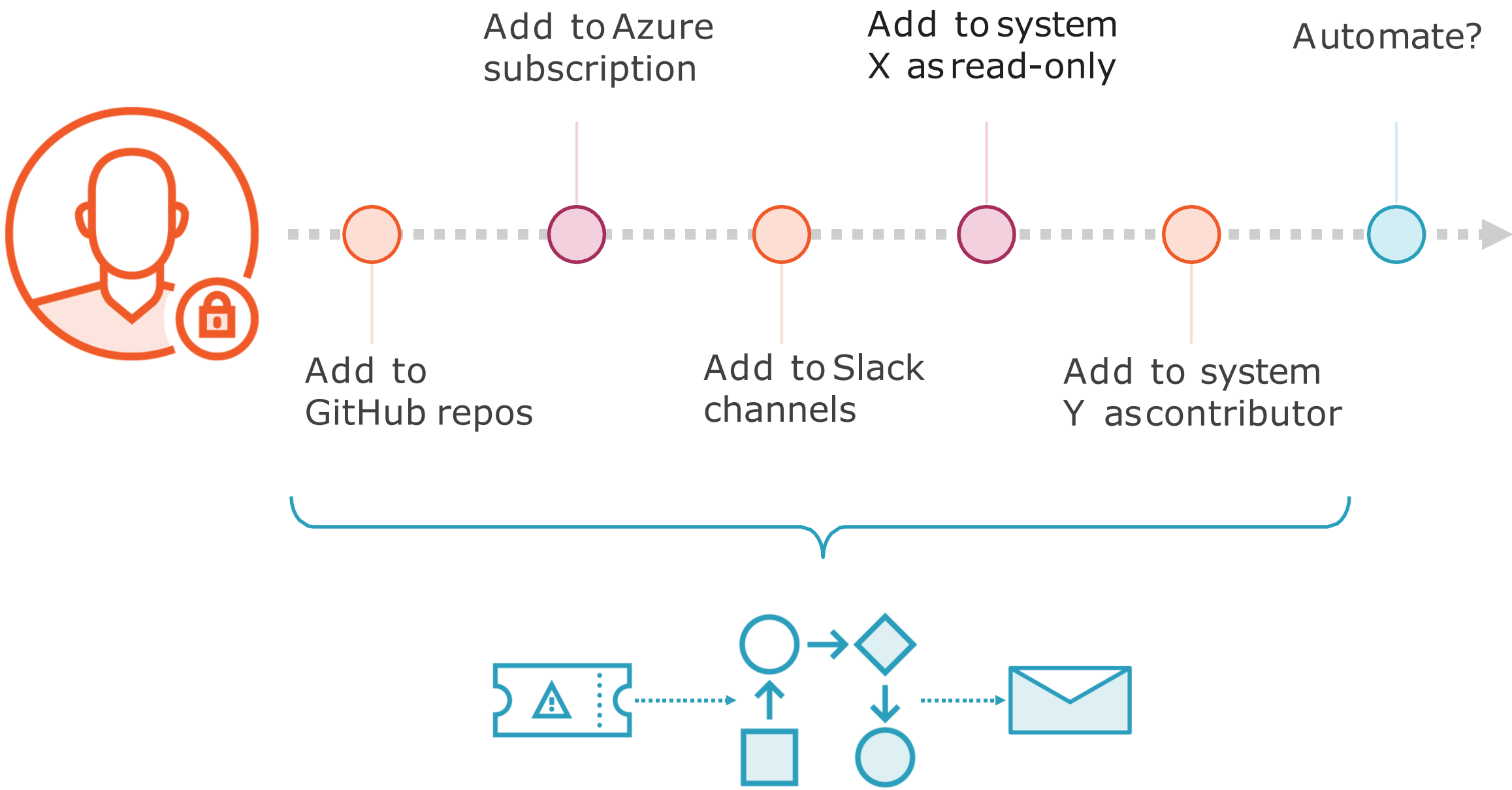
- Identify toil
- Quantify ongoing costs

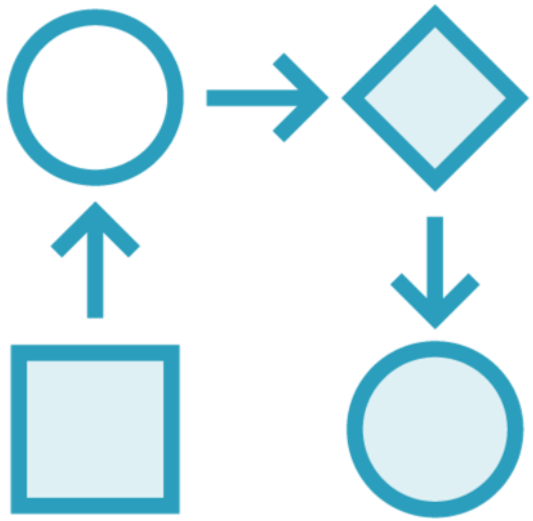
Toil-reduction backlog

- Prioritize projects
- SREs or whole team progress

Cost-benefit analysis

- Objective measures of toil
- Effort or elapsed time


Add to Azure subscription
Add to system X as read-only
Automate?
Add to GitHub repos
Add to Slack channels
Add to system Y as contributor



Automation projects

- Software engineering
- Support, maintenance, updates

Build costs vs. toil cost

- Time to value
- Maintenance and support
- Sharing with other teams

Other costs: $

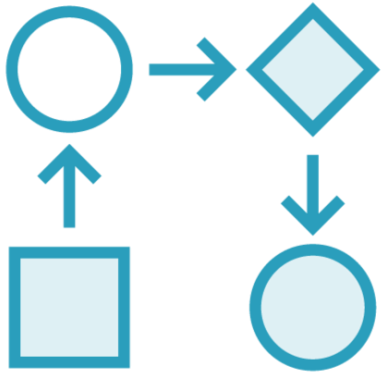
- Server outages
- Unused capacity

Engineering Away Toil

Identifying Automation Candidates

Analysis

Documentation



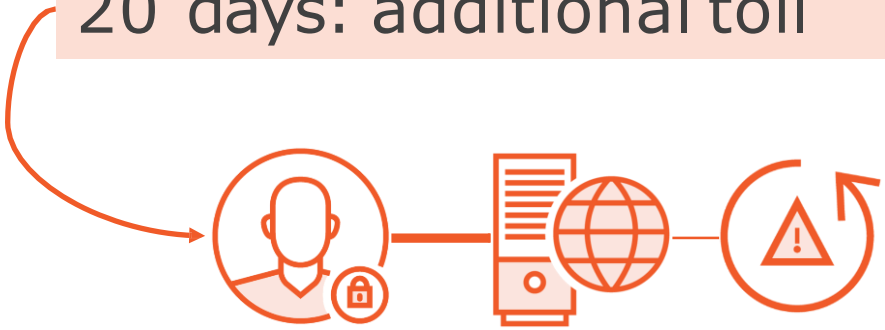
Repeatable Process

20 working days x6 SREs =120 days

50%

40 days: on-call
20 days: additional toil

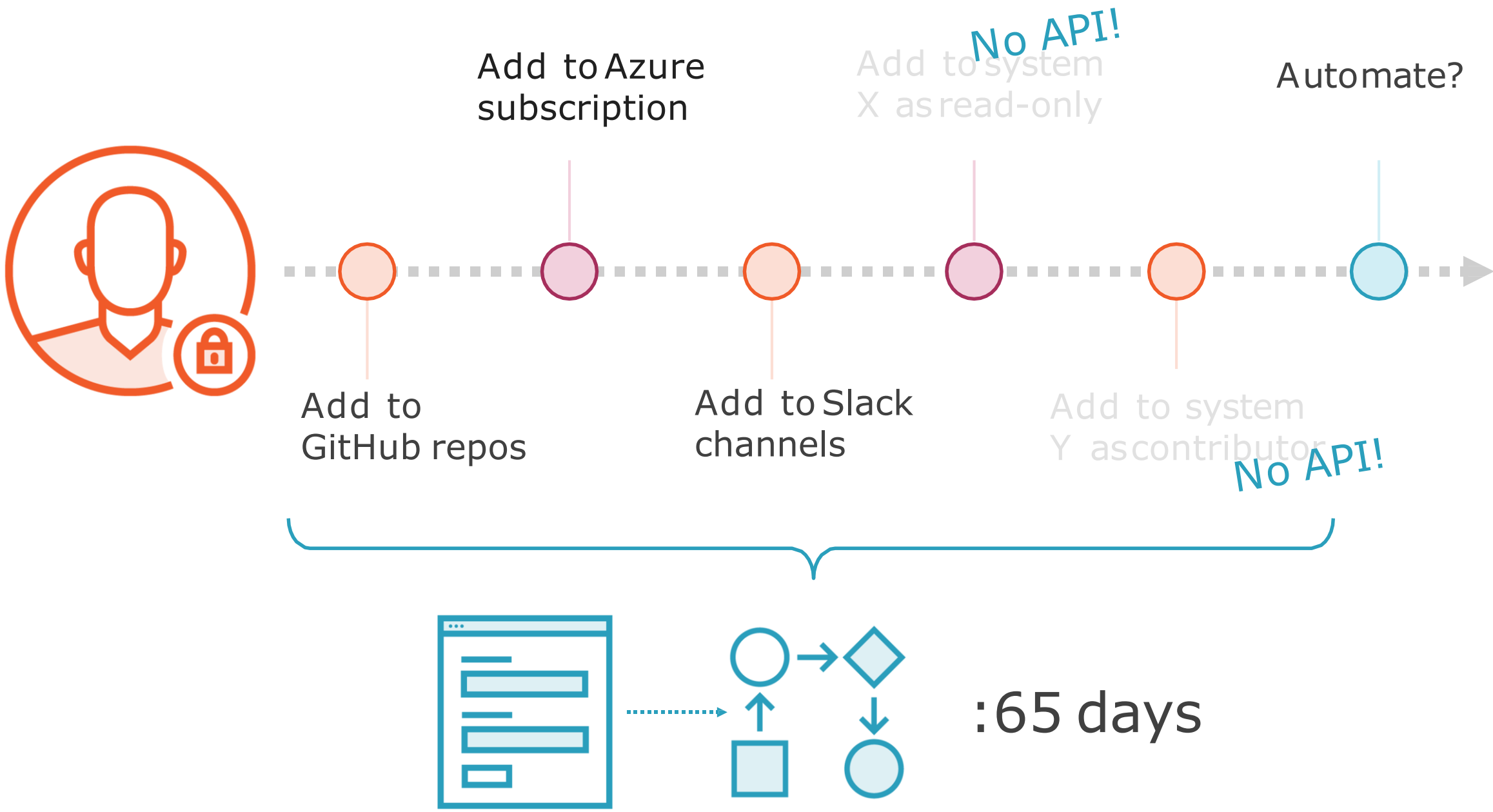
60 days: engineering budget

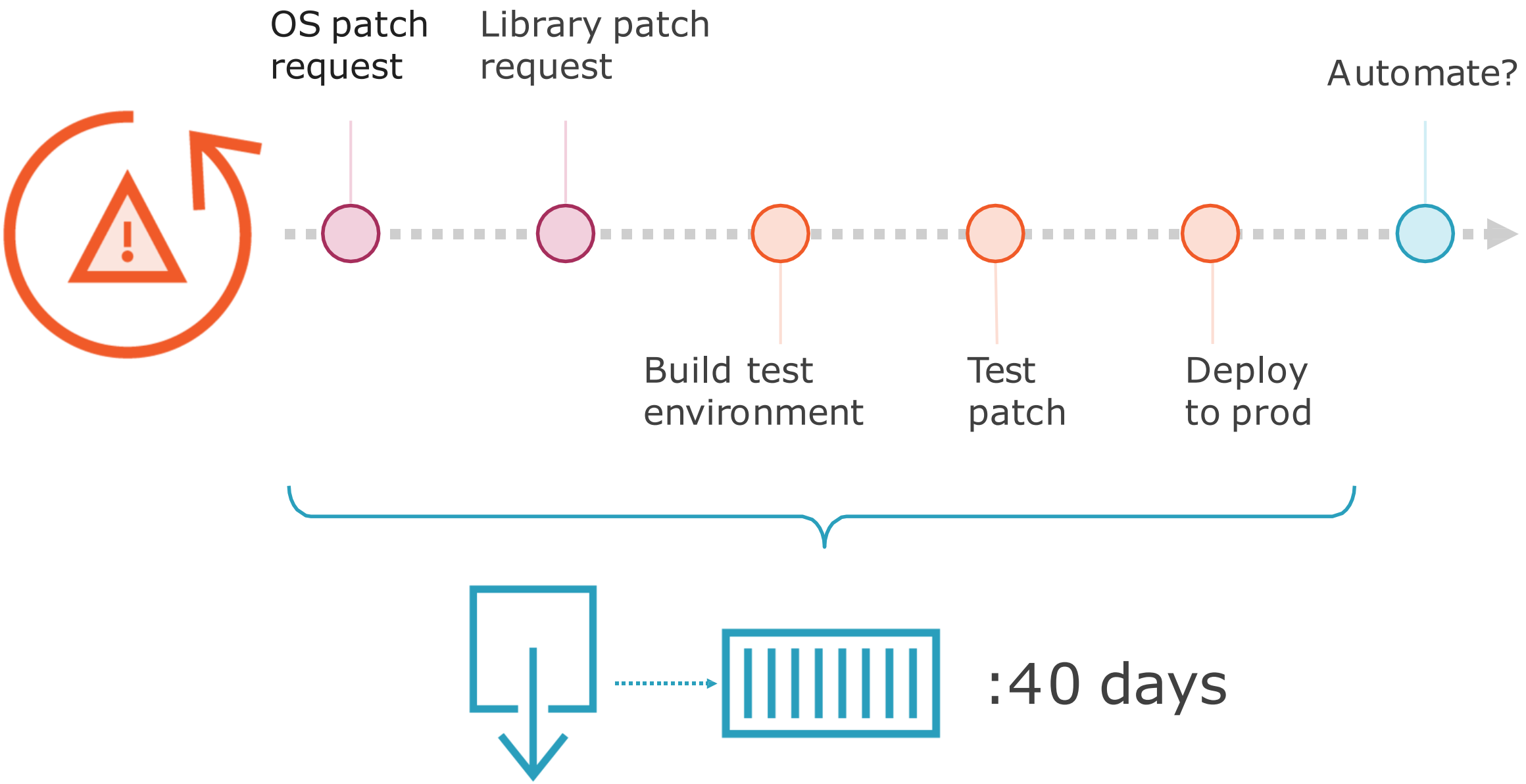


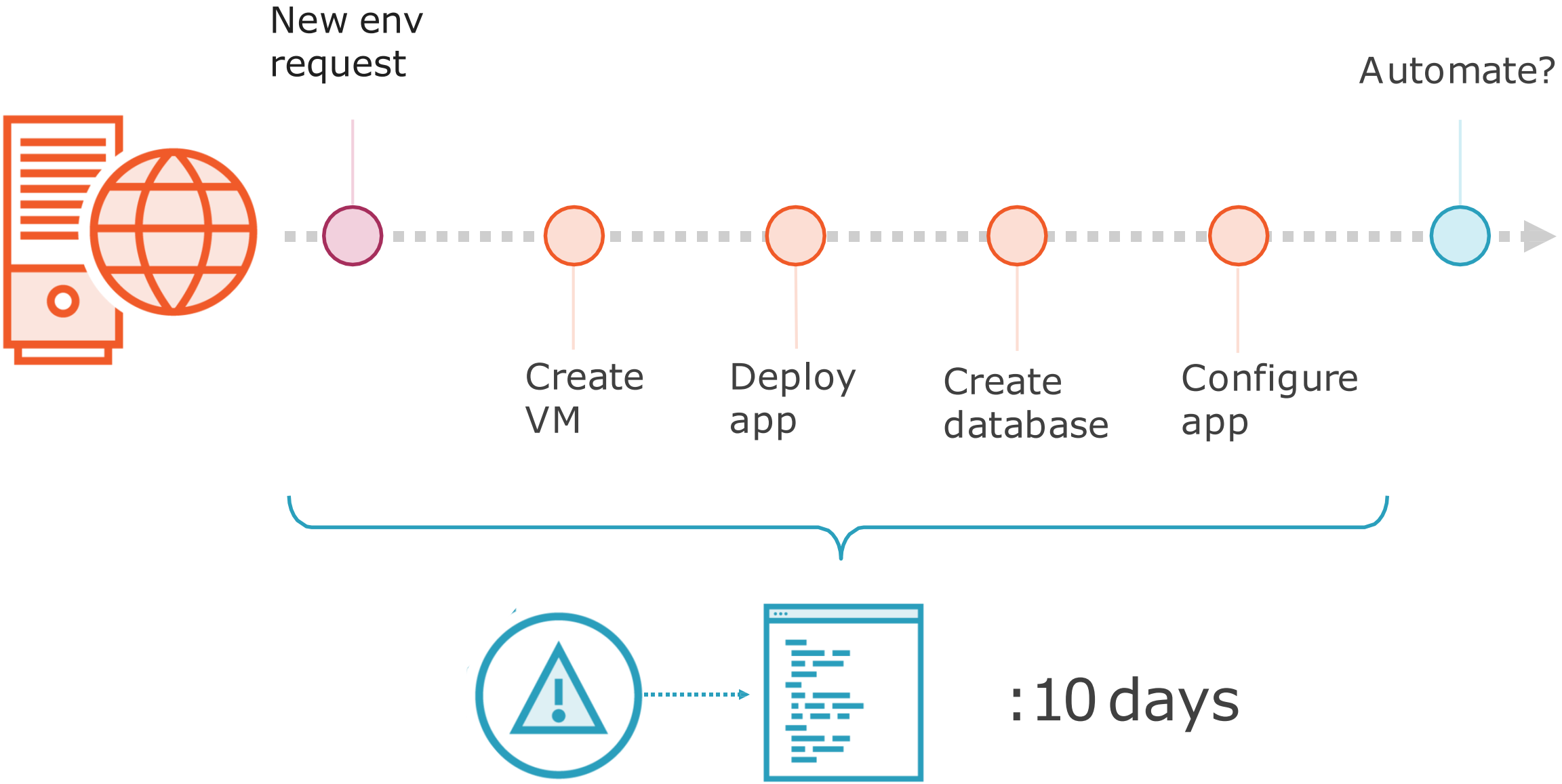
:12 days

Top Toil Targets

		Effort	*Frequency*	*Days /month*
#1		On-boarding	1day	x5	5
#2		Patching	2 days	x2	4
#3		Provisioning	1day	x3	3


Add to Azure subscription
No API!
Add to system X as read-only
Automate?
Add to GitHub repos
Add to Slack channels
Add to system Y as contributor
No API!
:65 days


OS patch request
Library patch request
Automate?
Build test environment
Test patch
Deploy to prod
:40 days


New env request
Automate?
Create VM
Deploy app
Create database
Configure app
: 10 days

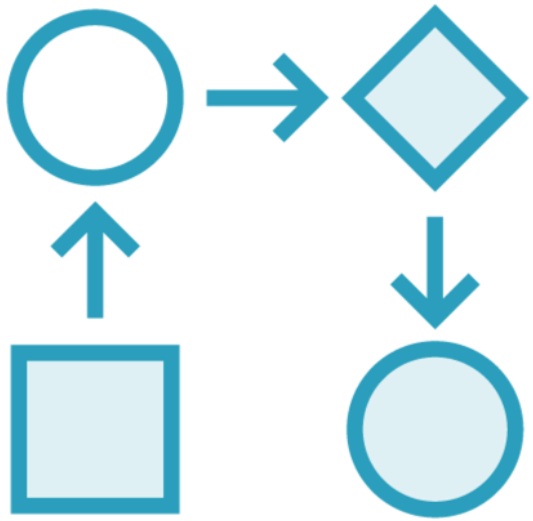
Prioritising Toil-Reducing Projects

Top Toil Projects

		Toil Days /month	*Project Estimate*
#1 	On-boarding	5	65
#2 	Patching	4	40
#3 	Provisioning	3	10

Top Toil Projects

		Toil Days /month	*Project Estimate*
#1	On-boarding	5	65
#2	Provisioning	3	10
#3	Patching	4	40



Toil-reduction side effects

- Productivity increase
- System reliability and availability
- Toolset standardization
- System simplification
- Culture of automation

#1?

Add to Azure subscription

Add to system X as read-only

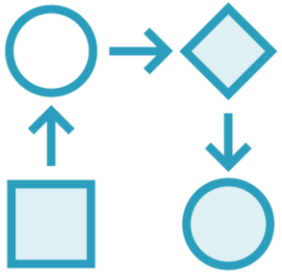
Automate?

Add to GitHub repos

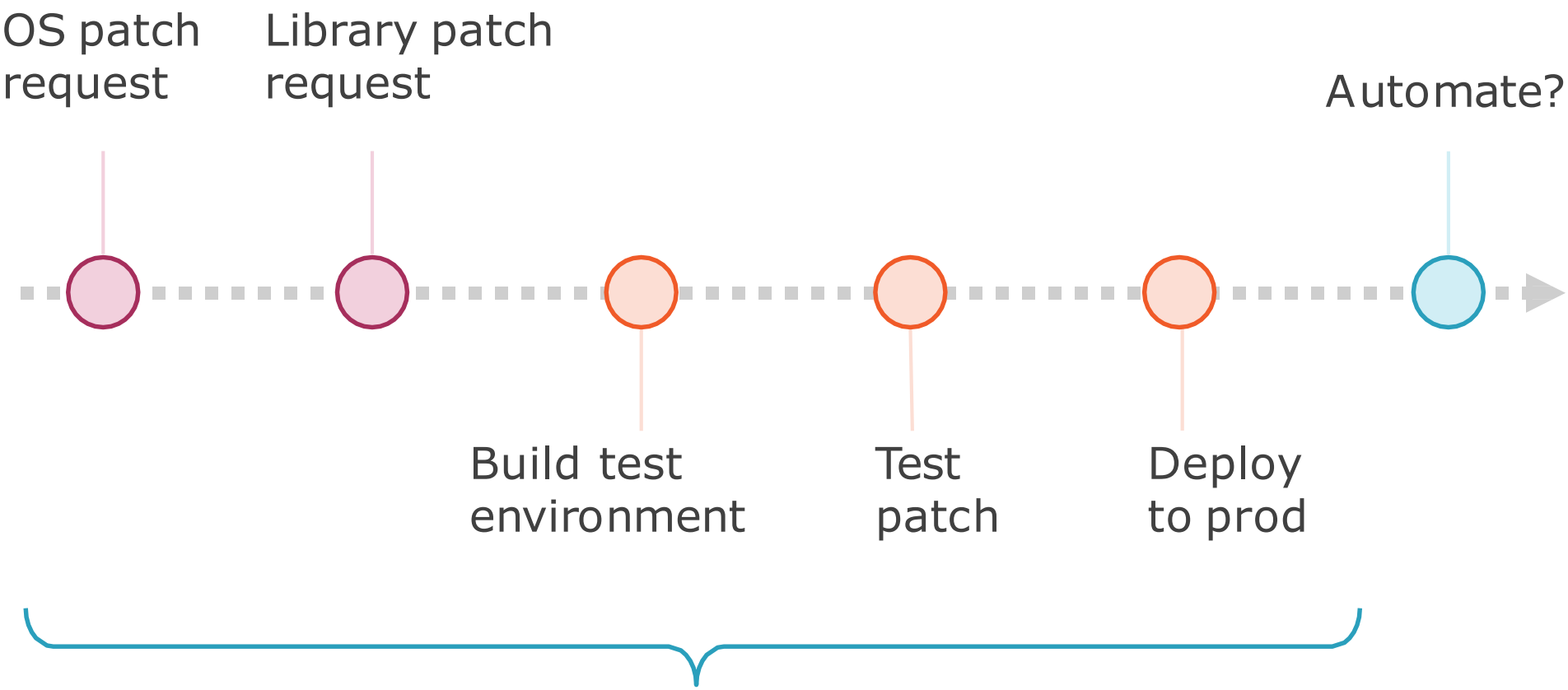
Add to Slack channels

Add to system Y as contributor

- Substantial software engineering
- Limited re-use
- No other side effects

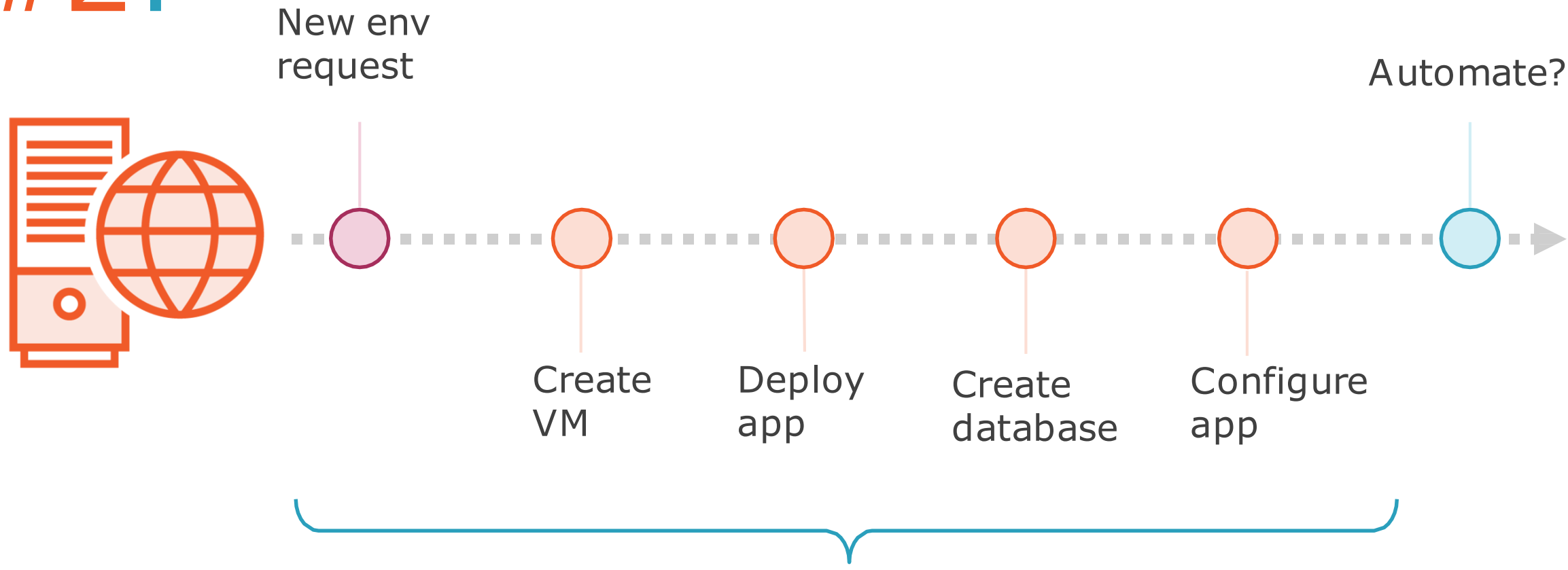


#3?



- Moderate systems engineering
- Platform standardization
- Capacity & health benefits

#2?



- Standard container deployment
- Self-service
- Reusable approach

Top Toil Projects

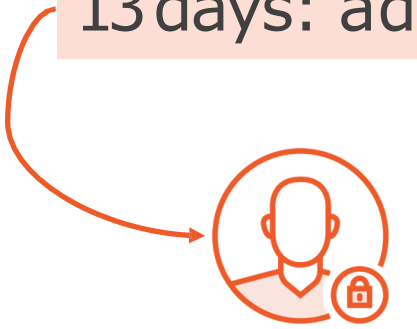
		Toil Days /month	*Project Estimate*
#1	Patching	4	40
#2	Provisioning	3	5
#3	On-boarding	5	65

20 working days x6 SREs =120 days

44%

40 days: on-call
13 days: additional toil

67 days: engineering budget



:5 days

Top Toil Projects

#1

	Toil Days /month	*Project Estimate*
On-boarding	5	65

Dealing with the Remaining Toil

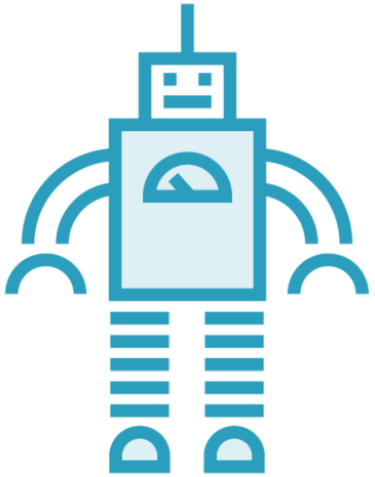
Impact Reduction Techniques

Identify & Measure

Batch Up

Ignore

Impact Reduction Techniques



Service Facade

Self-Service

Uniformity

Summary

Understanding toil

- Repetitive, low-value work
- Reduces high-value work time

Limiting toil

- Upper limit guarantee (50%)
- Constantly work to reduce

Eliminating toil

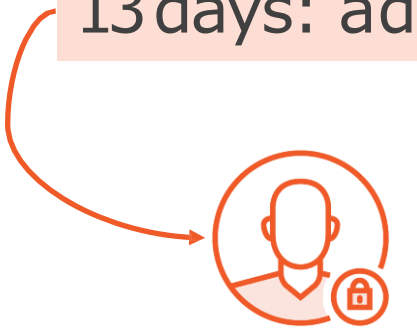
- Identify & measure
- Prioritized toil-reduction projects
- Adopt toil-reduction techniques

20 working days x6 SREs =120 days

44%

40 days: on-call
13 days: additional toil

67 days: engineering budget



:5 days

20 working days x6 SREs =120 days

33%

40 days: on-call
0 days: additional toil

80 days: engineering budget

Up Next:
Service Levels, Monitoring and Alerting

Service Levels, Monitoring, and Alerting

"The Business"

Ops Team

Why not 100%?

99% availability! Agreed?

What happens at 98%?

IT Management

Service Level Objectives

Success rate — 99.9% of requests have 2xx response

Response time — 90% of requests within 0.5 seconds

Response time — 99% of requests within 2 seconds

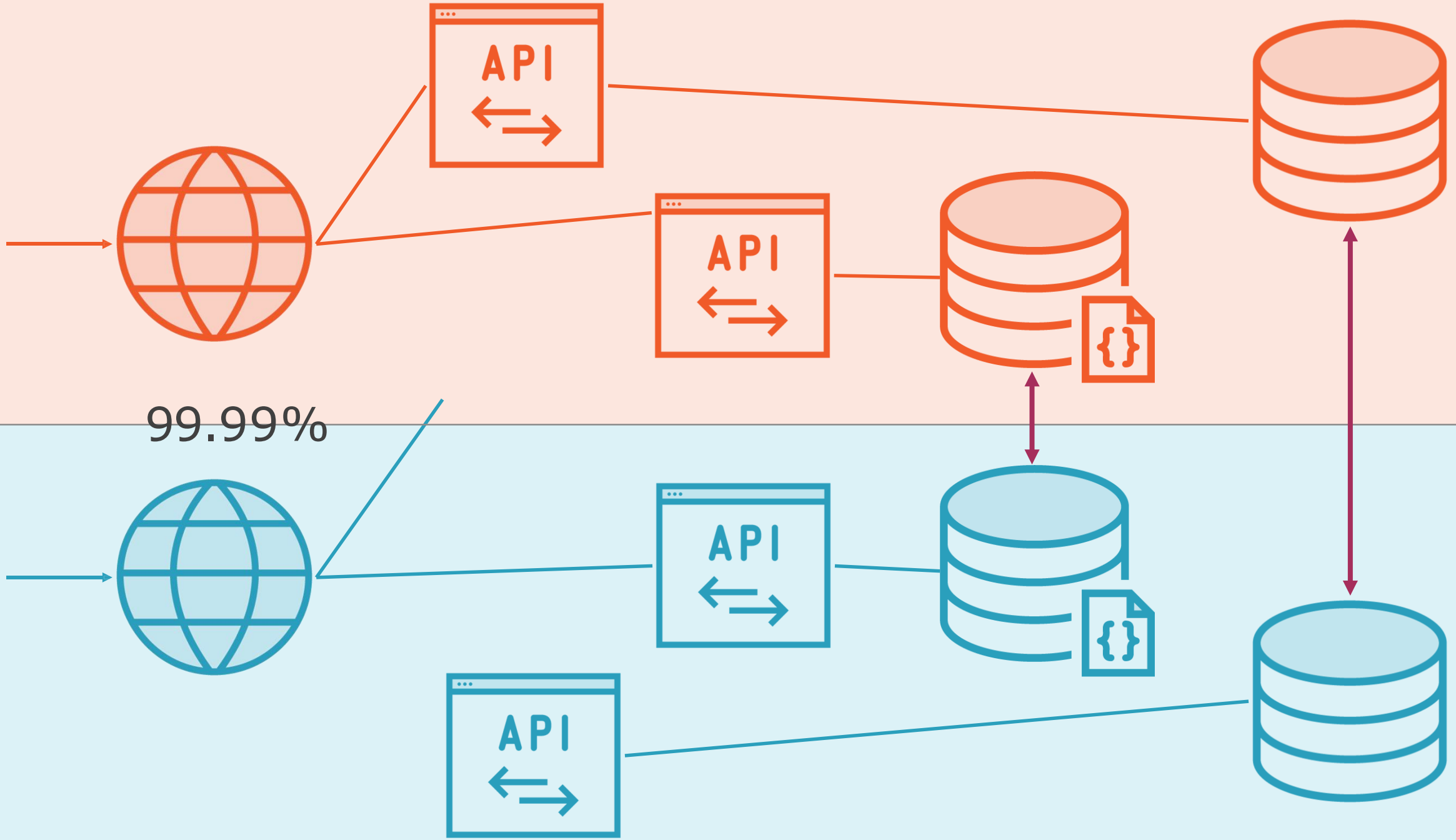
Service Level Objectives

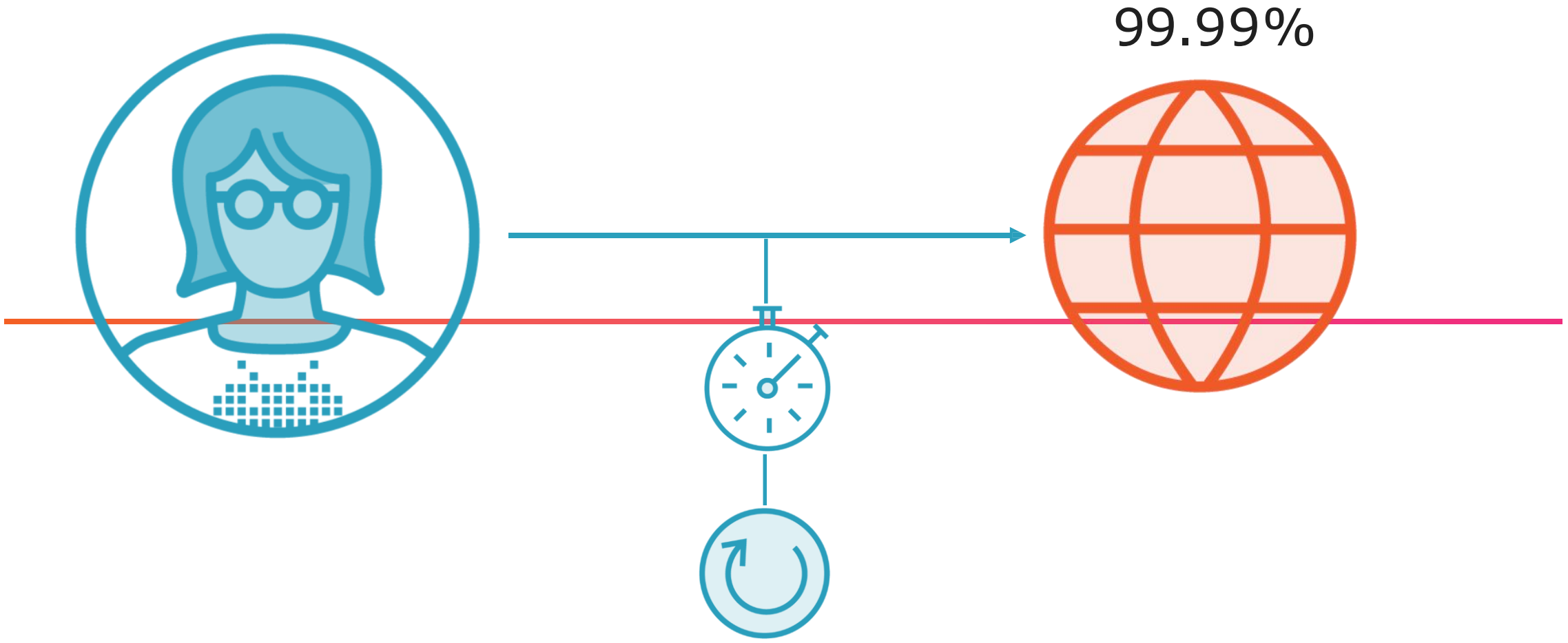
Service Level Objectives

Response time — 99% of requests within 2seconds

28 days = 40,320 minutes

SLO	*Time before breach*
99%	403 minutes (~7hours)
99.9%	40 minutes
99.99%	4 minutes
99.999%	0.4 minutes (~24seconds)


API
API
{}
99.99%
API
{}
API


99.99%

Service Level Objectives

Response time — 99.9% of requests within 5 seconds

28 days

Normal operation

99.9% within 5 seconds

40,280 minutes

- Change window
- Product updates
- Configuration

Error budget

0.1% outside of 5 seconds

40 minutes

Service Level Objectives

Response time

97% of requests within 5 seconds

28 days

Normal operation

97% within 5 seconds

39,110 minutes

Error budget

3% outside of 5 seconds

1,210 minutes

Service Level Objectives

Response time — 99.99% of requests within 5 seconds

28 days

Normal operation

99.99% within 5 seconds

40,316 minutes

Error budget

0.01% outside of 5 seconds

4 minutes

Error budget policy

- Formal document
- Agreed between product, dev and SRE

Enacted on SLA breach

- Prioritize reliability fixes
- Feature freeze - only reliability fixes
- Change freeze - only security patches

Contracted balance

- Change velocity
- Product reliability

Defining Service Level Indicators and Service Level Objectives

Service Level Measurement

Success rate
"Availability"

SLO: 99.9% of requests succeed
SLI: web server status codes

Response time
"Latency"

SLO: 90% of requests within 0.5 sec
SLI: web server response times

Service Level Indicators

Availability

SLI

5,000 requests
4,800 successful

96% successful

Latency

5,000 requests
~~Average 1.5s~~
4,000 within 0.5s
600 within 2s
300 within 5s

80% within 0.5s
92% within 2s
95% within 5s

Mapping SLIs to SLOs

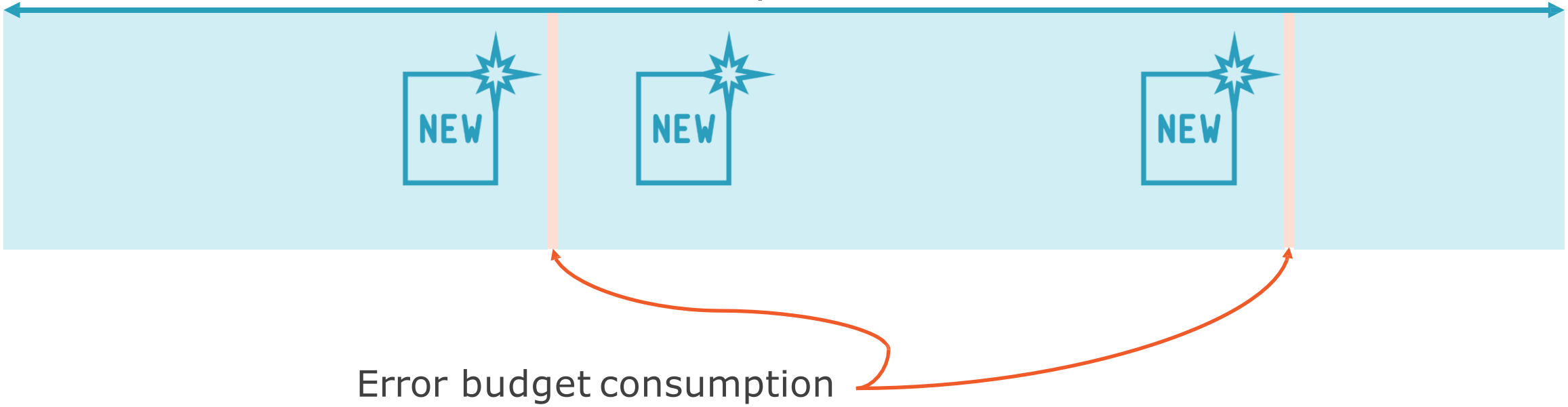
	Current SLI	*Target SLO*
Availability	5,000 requests 4,800 successful	96% successful	99% successful

Latency	5,000 requests 4,000 within 0.5s 600 within 2s 300 within 5s	80% within 0.5s 92% within 2s 95% within 5s	90% within 0.8s 95% within 2s

Service Level Period

Response time　99.9% of requests within 5seconds

14 days



Error budget consumption

Service Level Period

Response time　99.9% of requests within 5seconds

84 days

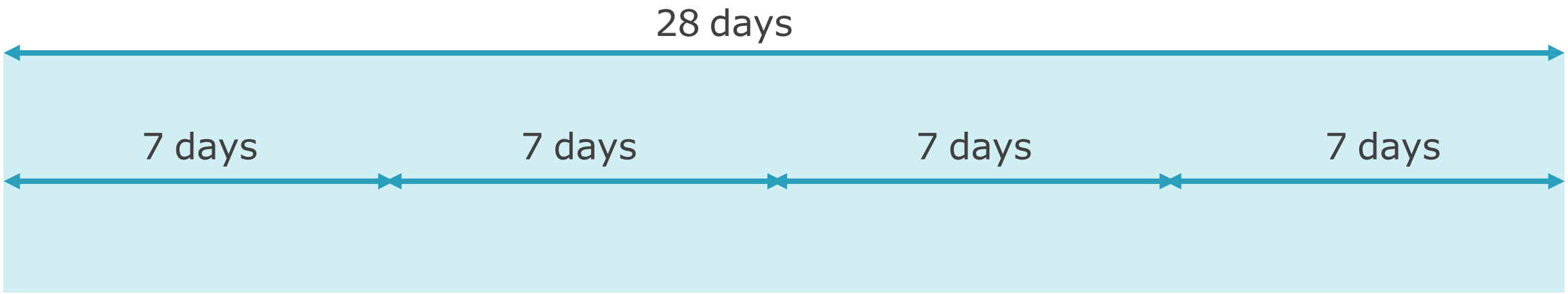
NEW

NEW

Error budget consumption

Service Level Period

Response time 99.9% of requests within 5 seconds



Monitoring Service Level Indicators

Four Golden Signals

Latency

Job processing time

Response generation time

Traffic

Length of message queue

Requests per second

Errors

Request failures

Response correctness

Saturation

CPU & memory utilization

Network bandwidth

Implementing SLIs

Server Logs

Response duration

No network time

Synthetic Testing

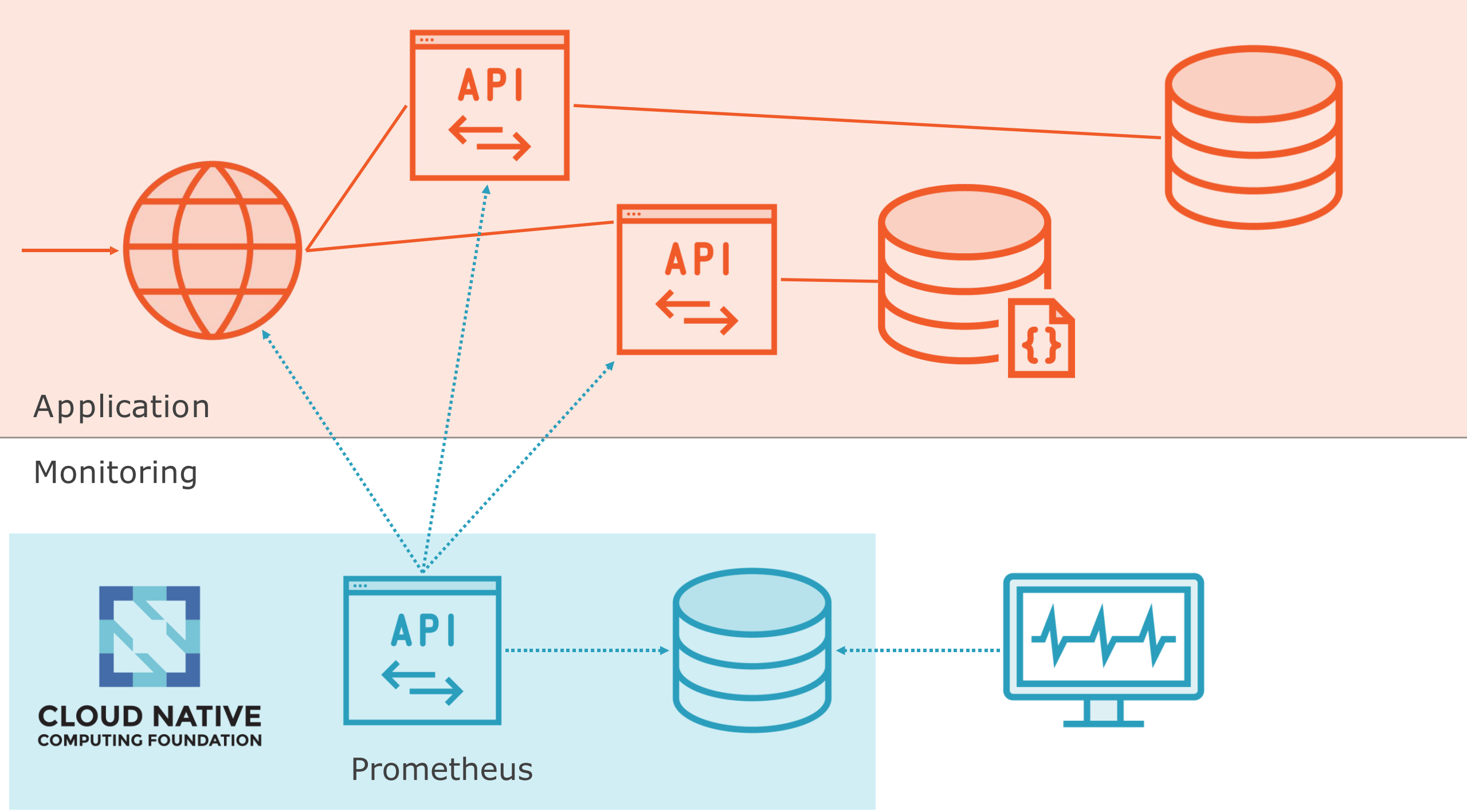
External services

Pingdom/StatusCake

JavaScript Monitoring

Network load time

Browser rendering


API
API
{}
Application
Monitoring
CLOUD NATIVE
COMPUTING FOUNDATION
API
Prometheus

Metric	Labels	Value
http_requests_count	{statusCode="200"}	4600
http_requests_count	{statusCode="401"}	360
http_requests_count	{statusCode="500"}	18
http_requests_count	{statusCode="503"}	149

Total HTTP requests: 5,127

Total failures (5xx): 167

Metric	Labels	Value
latency_seconds_bucket	{le="0.25"}	200
latency_seconds_bucket	{le="0.50"}	4300
latency_seconds_bucket	{le="1.00"}	4700
latency_seconds_bucket	{le="5.00"}	5000

50th percentile = *PromQL query*

90th percentile = *PromQL query*

Dashboards

- Visualize SLIs
- Four golden signals

Textual information

- Release versions
- Configuration timestamps

Analysis head-start

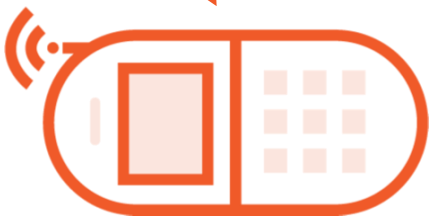
- SLO under threat
- Memory saturation high
- Since latest release?

Alerting on Service Level Objectives

Monitoring

Alerting

- Timeliness
- Accuracy
- Signal-to-noise



Precision

- Only trigger on significant events

Recall

- Trigger on all significant events

Detection time

- Time to trigger the alert

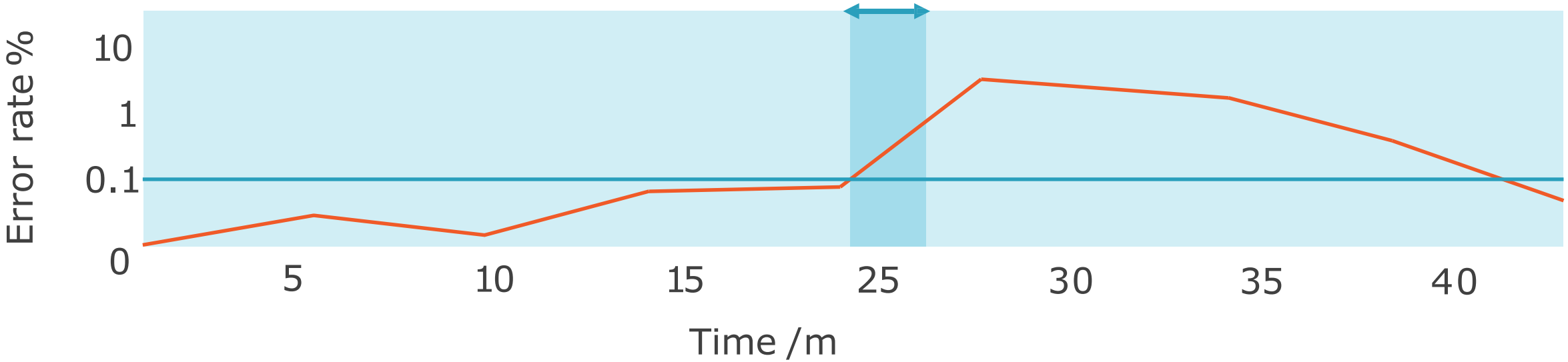
Reset time

- Time to stop alerting

Availability SLO: 99.9% success rate

Alerting on SLO Threshold

1%over 1minute

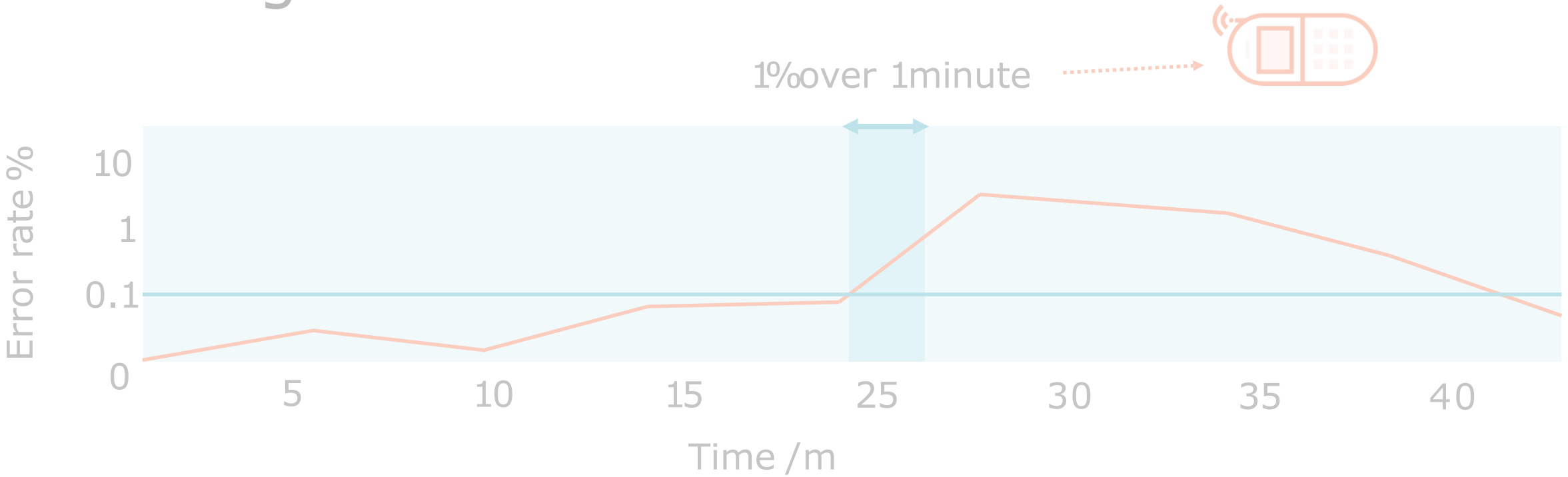


Availability

SLO: 99.9% success rate

Period	28 days
Expected load	1M requests
SLO window	10,000 errors

Alerting on SLO Threshold

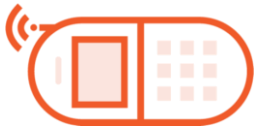


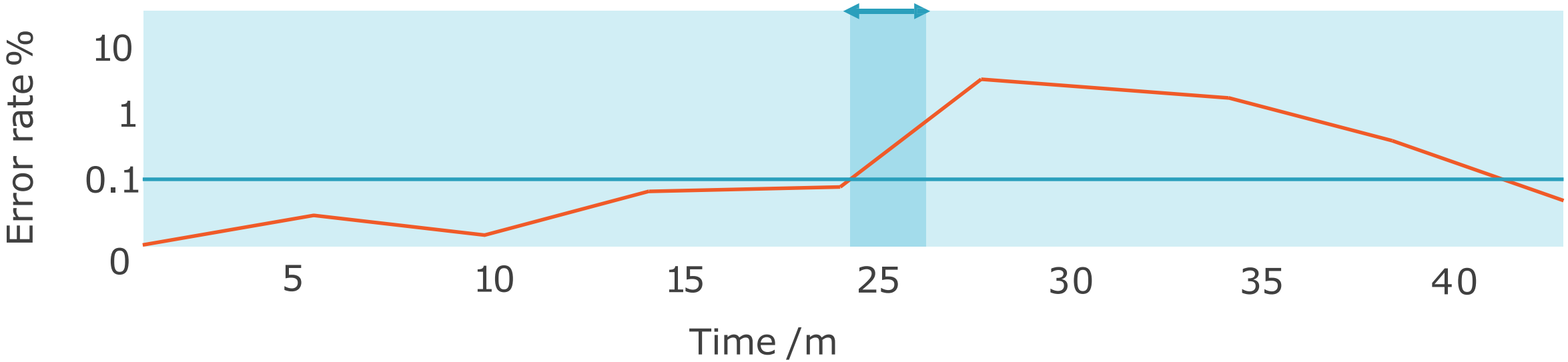
Availability SLO: 99.9% success rate

Period	28 days
Expected load	1M requests
SLO window	10,000 errors

Alerting on SLO Threshold

1% over 1minute
=1% of 25 req/s
=2.5 requests

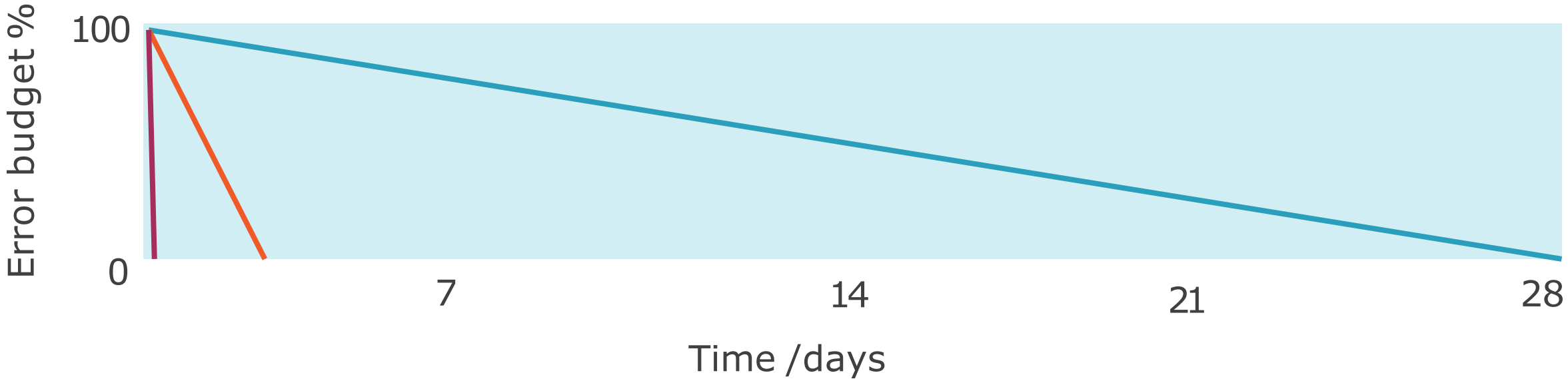




Alerting on Error Budget Burn

Availability SLO: 99.9% success rate

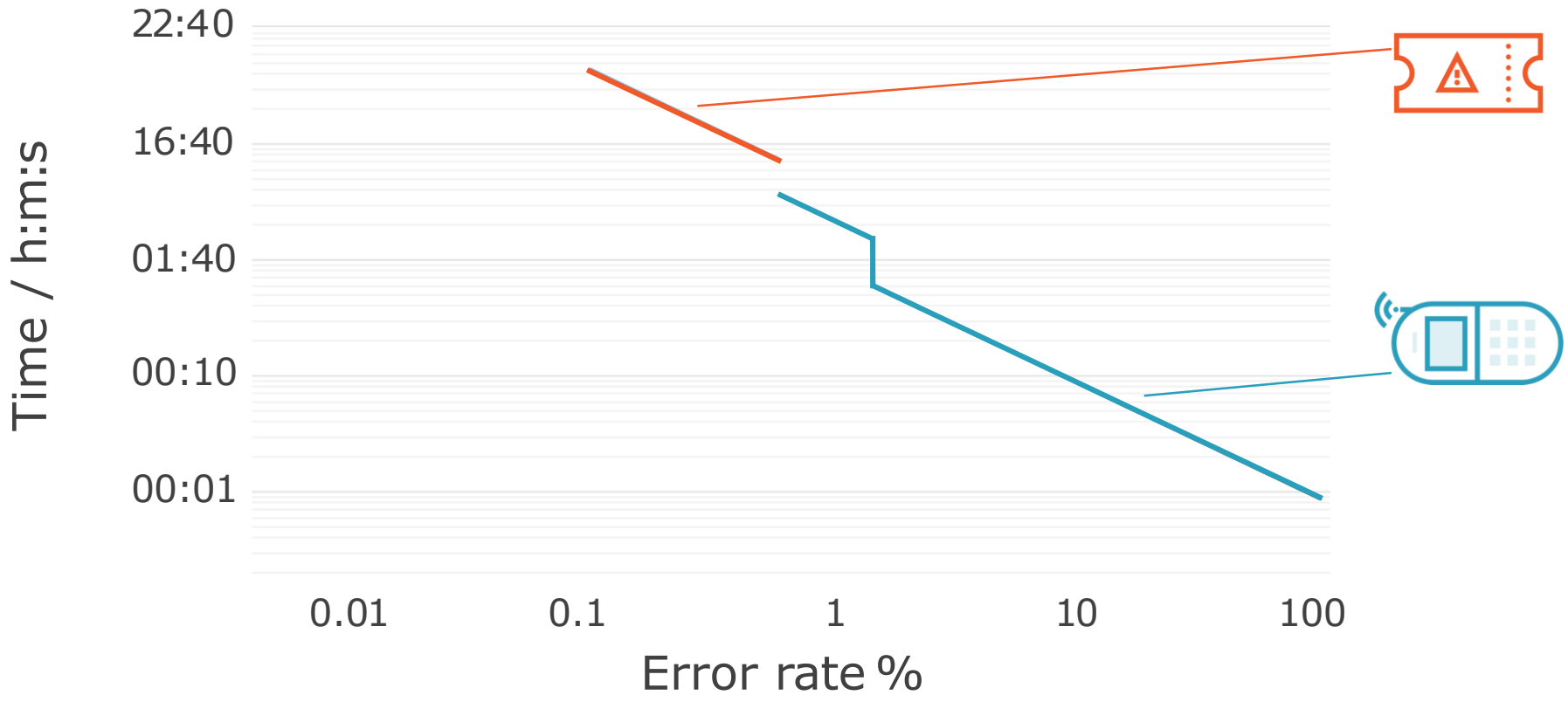
Error rate	*Burn rate*
0.1%	1
1%	10
100%	1,000



Alerting on Error Budget Burn

Availability

SLO: 99.9% success rate



Multiple Burn Rate Calculator

Source code

SLO
99.9

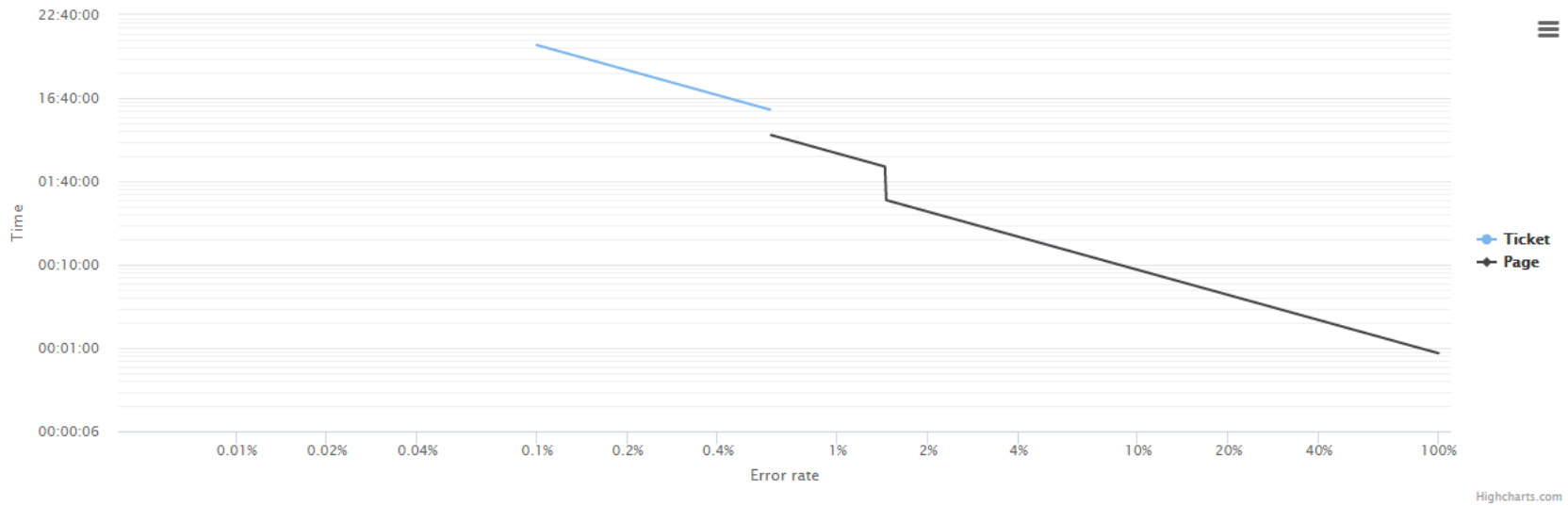
SLO Window (days)
30

Error Budget Consumption thresholds

Time	Budget consumption (%)	Burn rate
1h	2	14.4
6h	5	6
1d	10	3
3d	10	1

https://is.gd/1oILuV

Detection time



Source:
is.gd/mJwaoa

```
expr: (job:slo_errors_per_request:ratio_rate1h{job="x"} > (14.4*0.001)
      and
       job:slo_errors_per_request:ratio_rate5m{job="x"} > (14.4*0.001))
    or
      (job:slo_errors_per_request:ratio_rate6h{job="x"} > (6*0.001)
      and
       job:slo_errors_per_request:ratio_rate30m{job="x"} > (6*0.001))
severity: page
```

```
expr: (job:slo_errors_per_request:ratio_rate24h{job="x"} > (3*0.001)
      and
       job:slo_errors_per_request:ratio_rate2h{job="x"} > (3*0.001))
    or
      (job:slo_errors_per_request:ratio_rate3d{job="x"} > 0.001
      and
       job:slo_errors_per_request:ratio_rate6h{job="x"} > 0.001)
severity: ticket
```

“The SLO error rate is 0.1%. Owners get paged if their backend has returned more than 1.44% 5xx responses over the last 1h and over the last 5m. They also get paged if it has returned more than 0.6% 5xx responses over the last 6d and over the last 30m.”

Björn Rabenstein, SoundCloud https://is.gd/EhtIPT

Summary

Monitoring with service levels

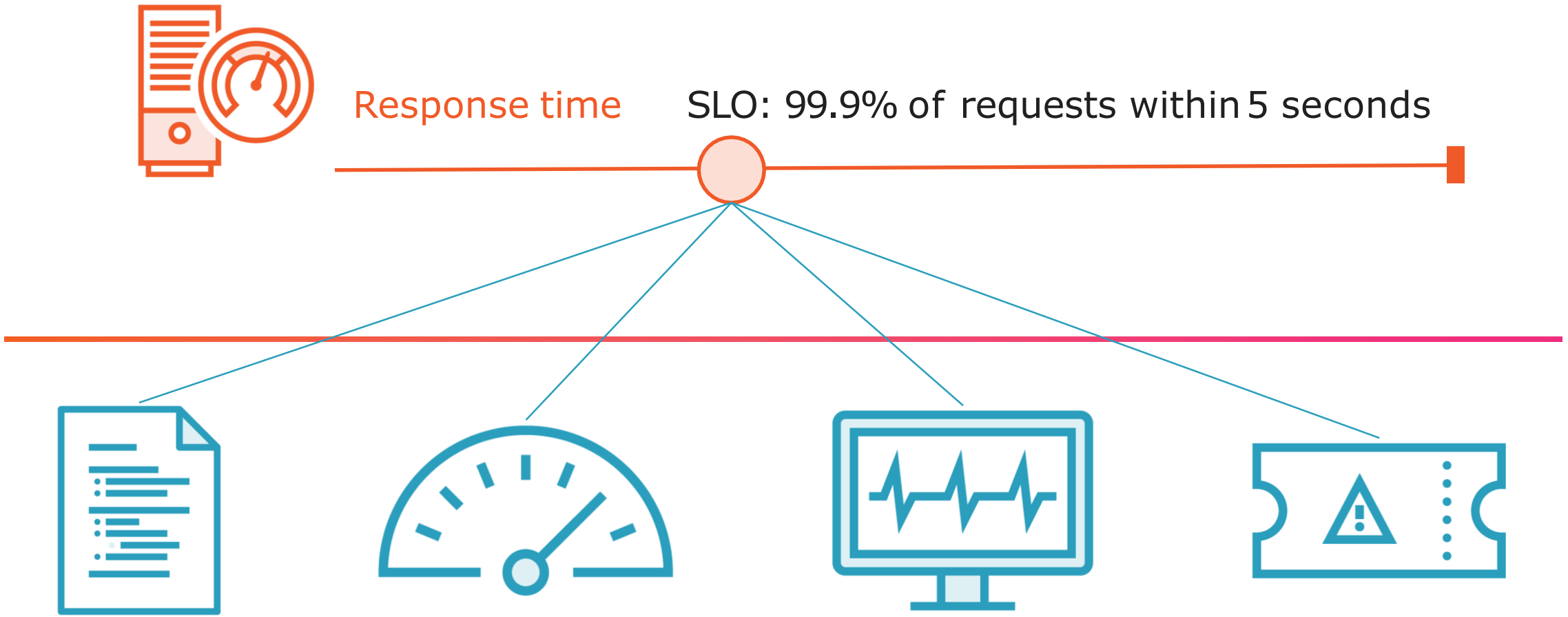
- Service Level Objectives
- Service Level Indicators

Error budget policy

- Agreed between business, dev & SRE
- Enacted if SLOs are not met
- Feature or full change freeze

Alerting

- Parameters for timeliness & accuracy
- Error budget burn rate
- Multiple time windows


Response time
SLO: 99.9% of requests within 5 seconds

SLO Review

Accuracy

User Experience

Toil

SLO Review

SLO Met	User Satisfaction	SRE Toil	Action
✓			Relax SLO
✗			Relax SLO
✗			Product design

Enhanced SLOs

Success rate

99.9% of requests have 2xx response

99.99% of checkouts have 2xx response

Response time

95% of requests within 2 seconds

95% of homepage within 0.7 seconds

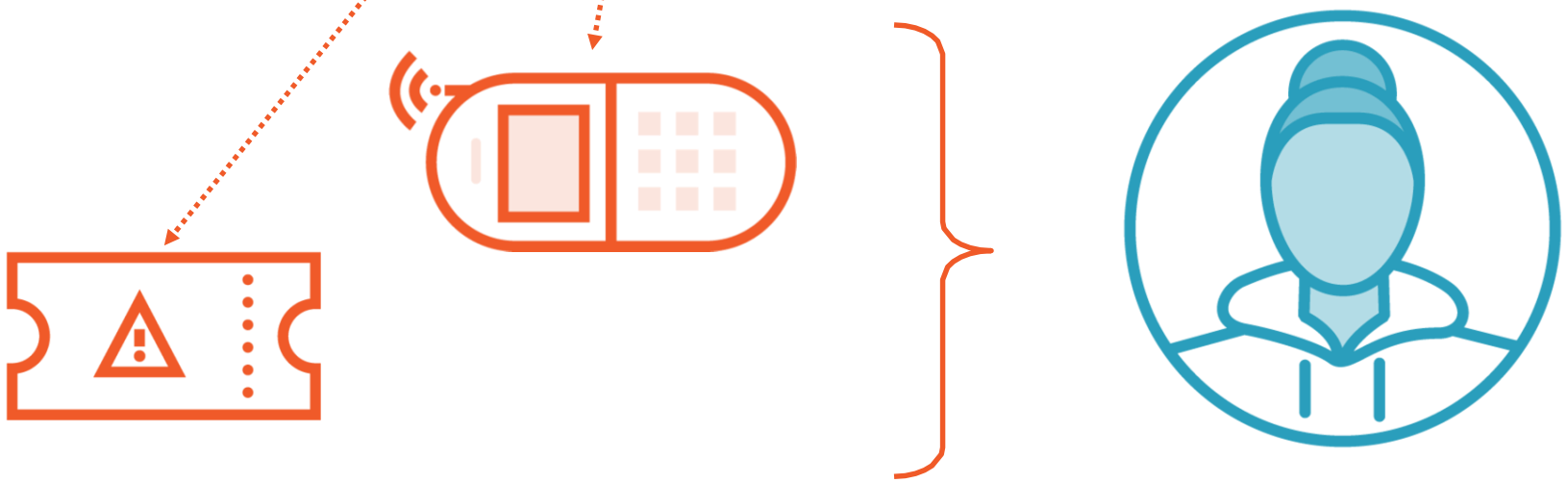
95% of search within 1.5 seconds

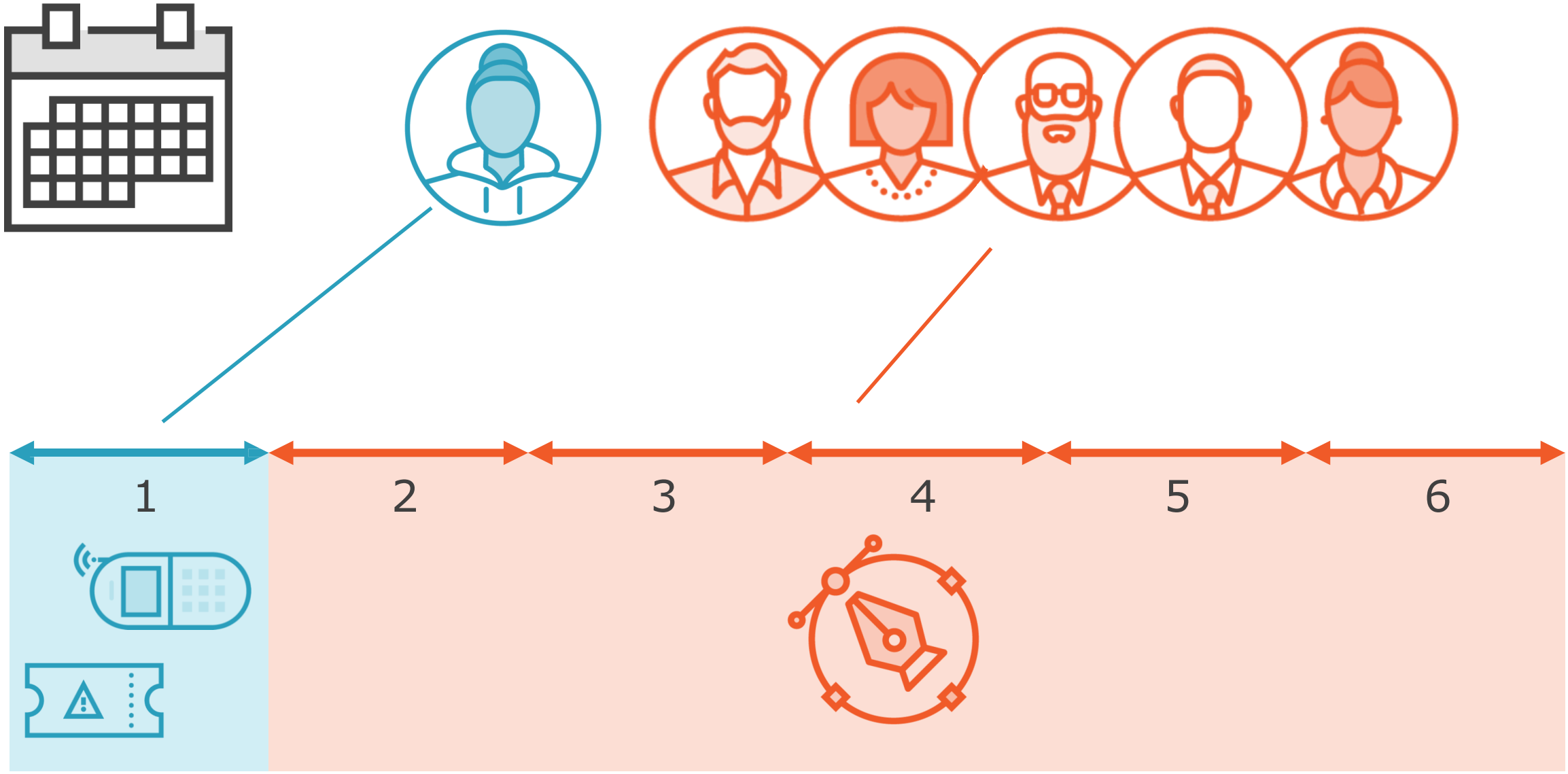
Up Next
Incident Management: On-call and Postmortems

Incident Management: On-call and Postmortems

Monitoring

Alerting




1
2
3
4
5
6

On-call

- Alerted on SLOs
- Acknowledge alert
- Begin investigation

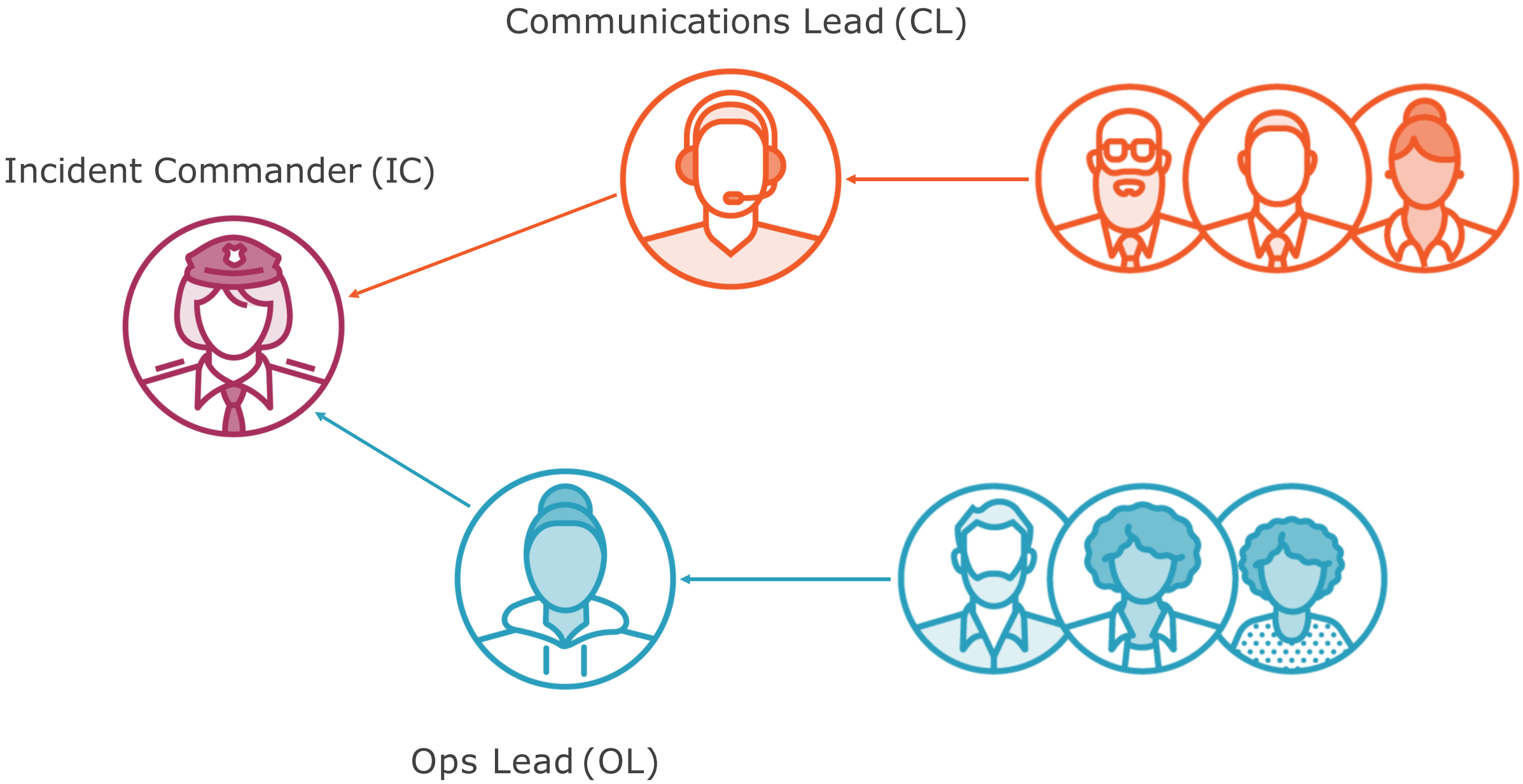
Timely response

- Business critical - within 5 minutes
- Urgent - within 30 minutes

Major commitment

- No social engagements
- Sleep deprivation
- Stress

Control, Co-ordinate & Communicate


Communications Lead (CL)
Incident Commander (IC)
Ops Lead (OL)

Is it an Incident?

Standard Response

Playbook

No incident

SRE Discretion

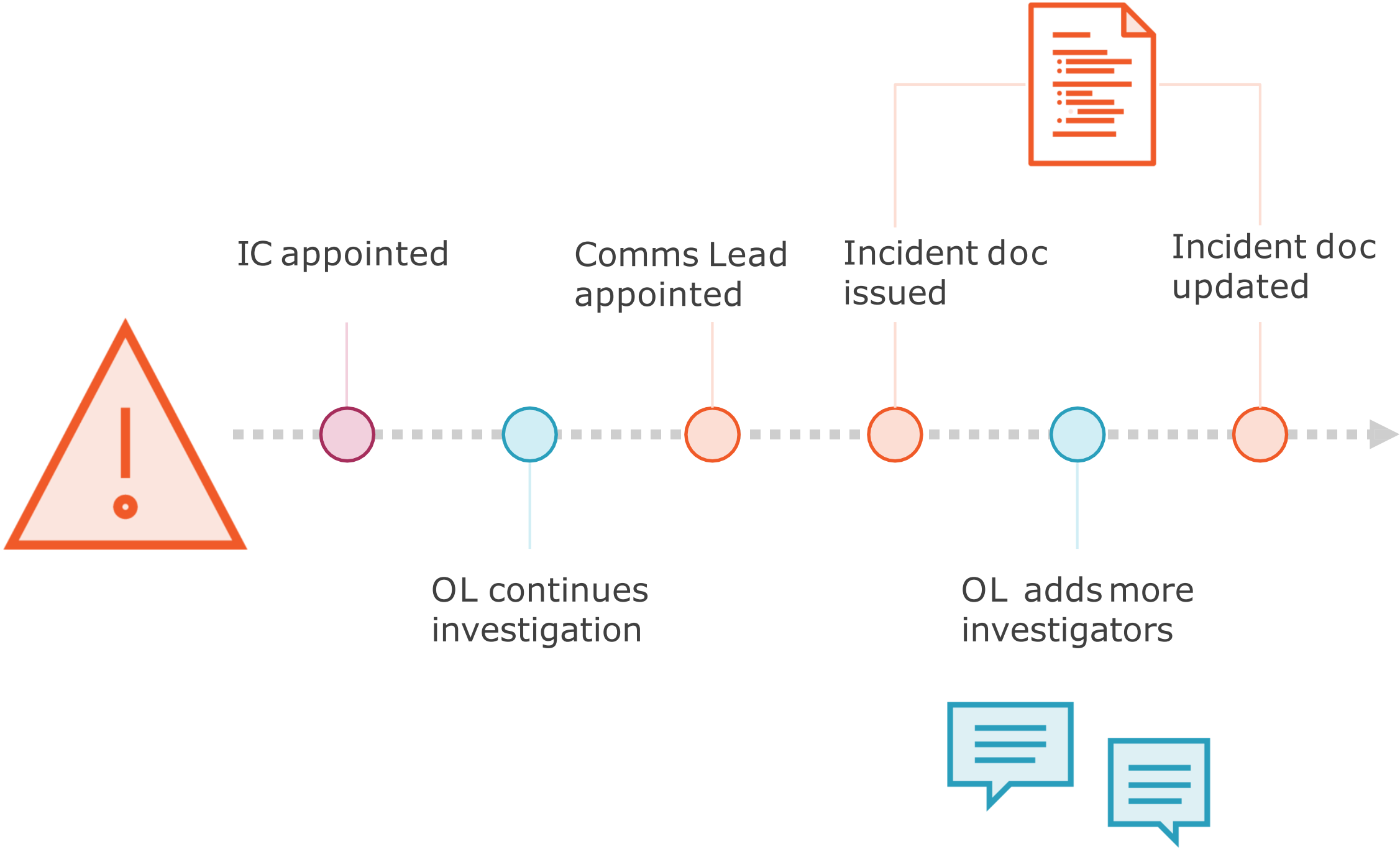
Past experience

Issue complexity

Set Criteria

Investigation time

Impact


IC appointed
Comms Lead appointed
Incident doc issued
Incident doc updated
OL continues investigation
OL adds more investigators

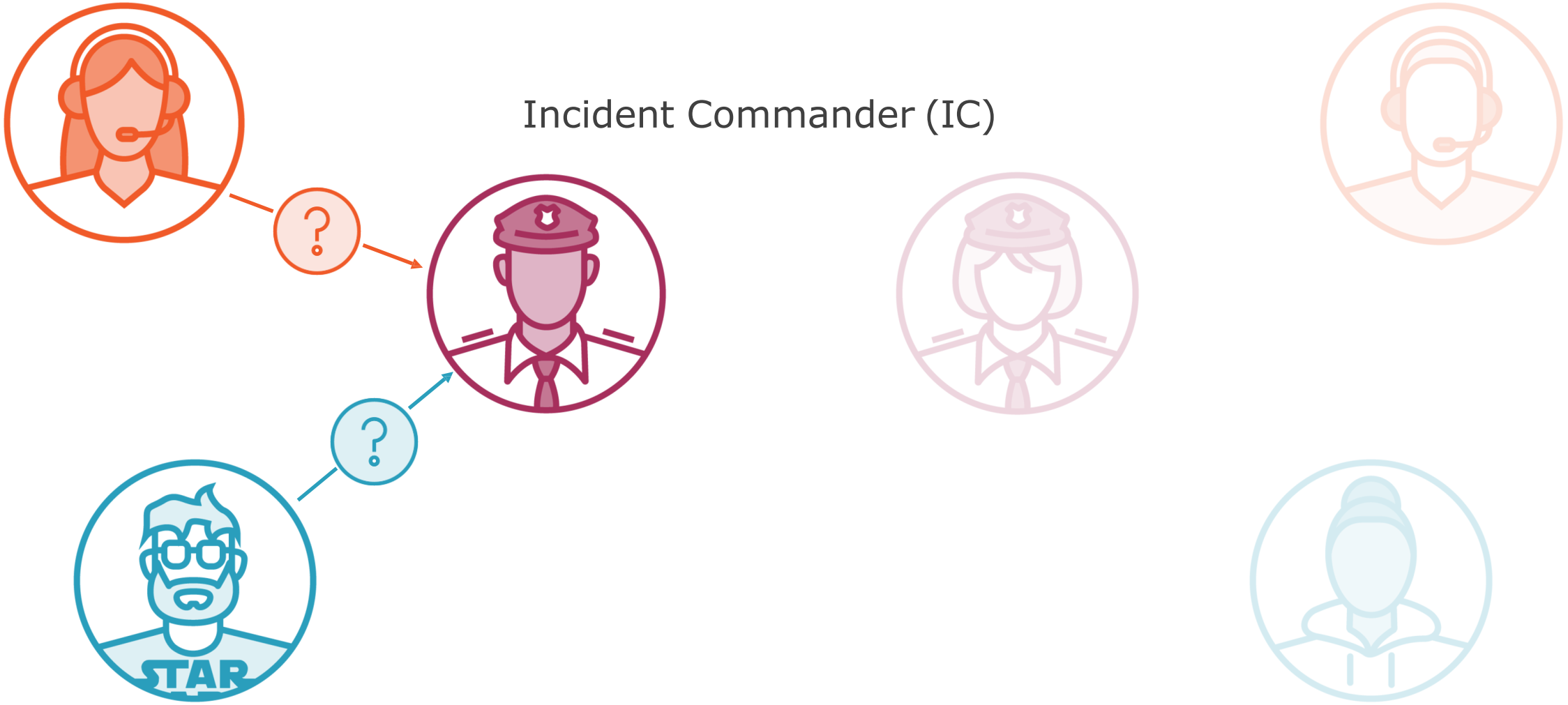
Incident Commander (IC)

Comms Lead (CL)

I confirm I am now the IC

Ops Lead (OL)

Comms Lead (CL)



Incident Commander (IC)

Ops Lead (OL)

SRE

SRE

SRE

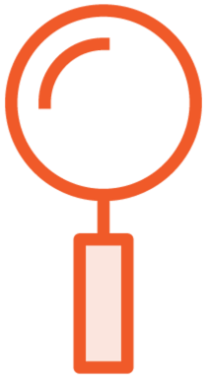
Incident Document

Postmortem

Working on Incidents Effectively

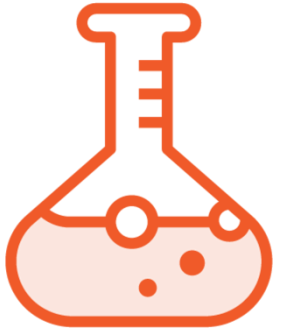
Incident Model

Triage

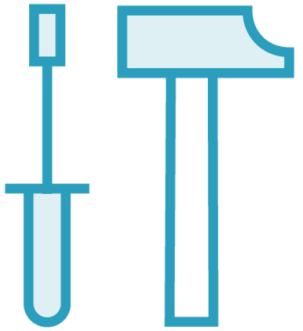


Examine

Diagnose



Test



Cure

Incident Model

- Automated alert
- SLO breach
- Metric details

- Problem report
- Manually recorded
- Expected, actual & repro

Triage

- Get back to "good enough"
- ASAP

Remediation tactics

- Add compute power
- Re-route traffic
- Downgrade service

Output

- Stable system

Examine

- Understand the problem
- Identify the trigger

Investigation tools

- Metrics and dashboards
- Centralized logging
- Service graphs
- Distributed Tracing

Examine

- Understand the problem
- Identify the trigger

Investigation tools

- Metrics and dashboards
- Centralized logging
- Service graphs
- Distributed Tracing

Output

- Know the problem and trigger

Diagnose

- Find the possible cause

Analysis tools

- Vertical path through system
- *What* is it doing?
- *Why* isn't it doing what it should?
- *Where* are resources going?
- *When* did it start?

Output

- Shortlist of potential causes

Test

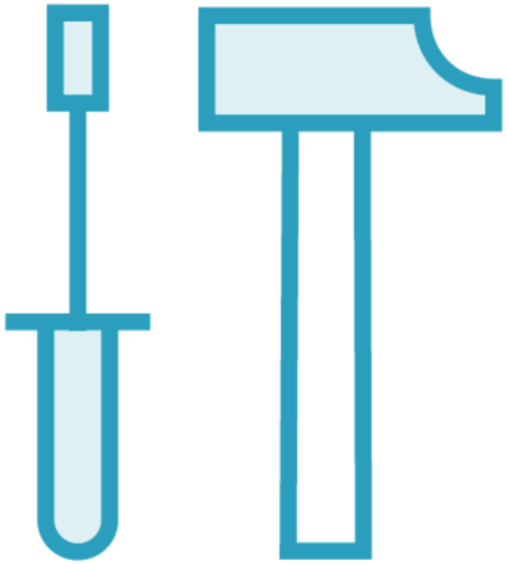
- Identify the probable cause

Testing tactics

- Manually recreate the workflow
- cURL Web apps and APIs
- Connect and verify permissions
- Trace database calls & network routes
- Document everything

Output

- Confidence in the actual cause



Cure

- Fix the problem
- Document the solution

Output

- Fully working system OR
- Mitigated system with known fix OR
- Mitigation with monitoring requirements

Producing and Publishing Postmortems

Postmortem goals

- Document the incident & resolution
- Identify root cause & fix

Formal documentation

- Drafted by SREs
- Review & publish process

Continuous improvement

- Blame-free
- Neutral & constructive

Date:
Authors:
Reviewers:
Incident Commander:

Executive Summary

Problem Summary

Action Items

Lessons Learned

Timeline

Date:
Authors:
Reviewers:
Incident Commander:

Executive Summary

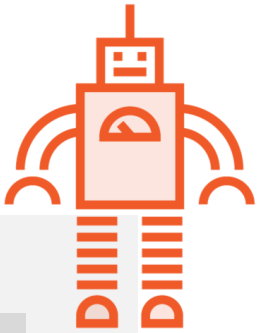
Problem Summary

Action Items

-
-
-

Timeline

Lessons Learned



Postmortems are expensive

- Writing & reviewing time

Promote use

- "Postmortem of the month"
- Reading clubs
- Group review sessions

For major incidents

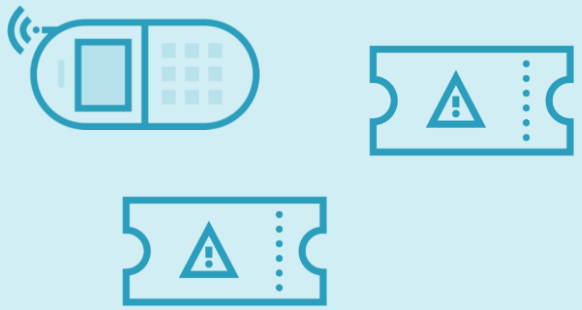
- Customer-facing
- Data issues
- Multiple impacts
- Lengthy resolution

Avoiding Operational Overload

- No project work
- Pages and tickets
- On-call only

Primary

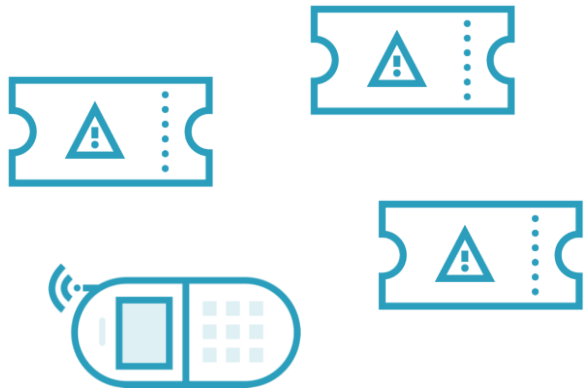
Secondary
9-5



Primary

Secondary
9-5

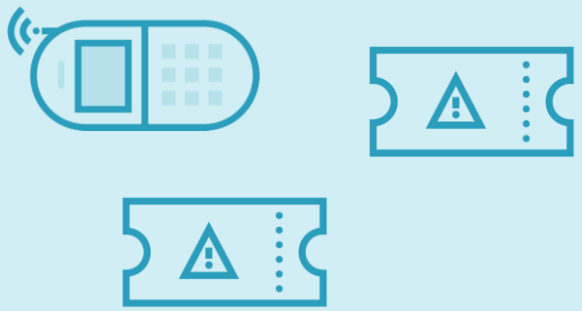
Tickets
9-5



Primary 1
06:00-17:59

Primary 2
18:00-05:59

Secondary
9-5



Incident resolution

- Assume 6 hours
- Two per on-call shift

No pages or tickets?

- Documentation, tidy-up work
- Easily dropped

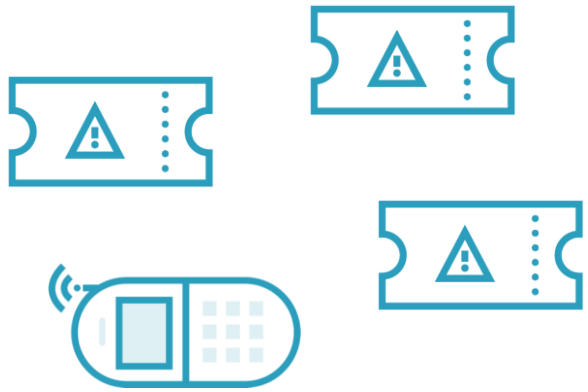
Consistently few incidents

- Does the project need SRE?
- Move to product team management

Primary

Secondary
9-5

Tickets
9-5

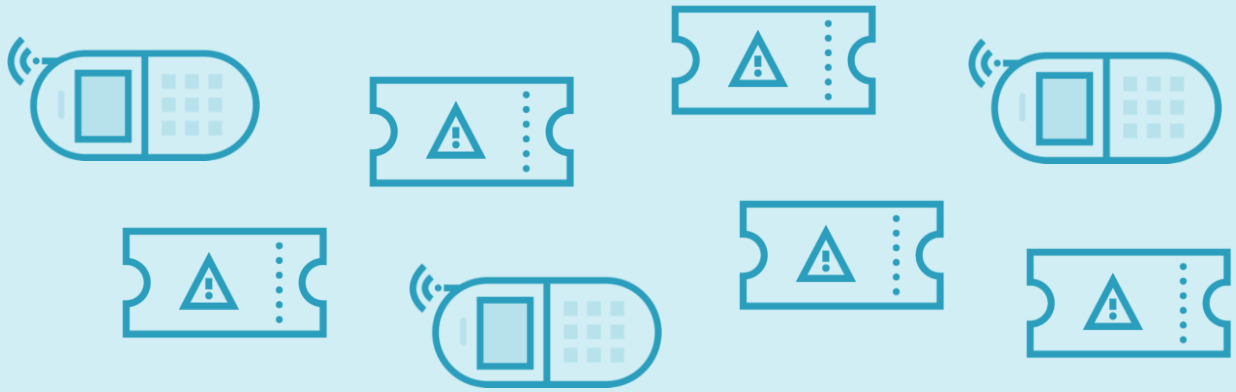


Primary

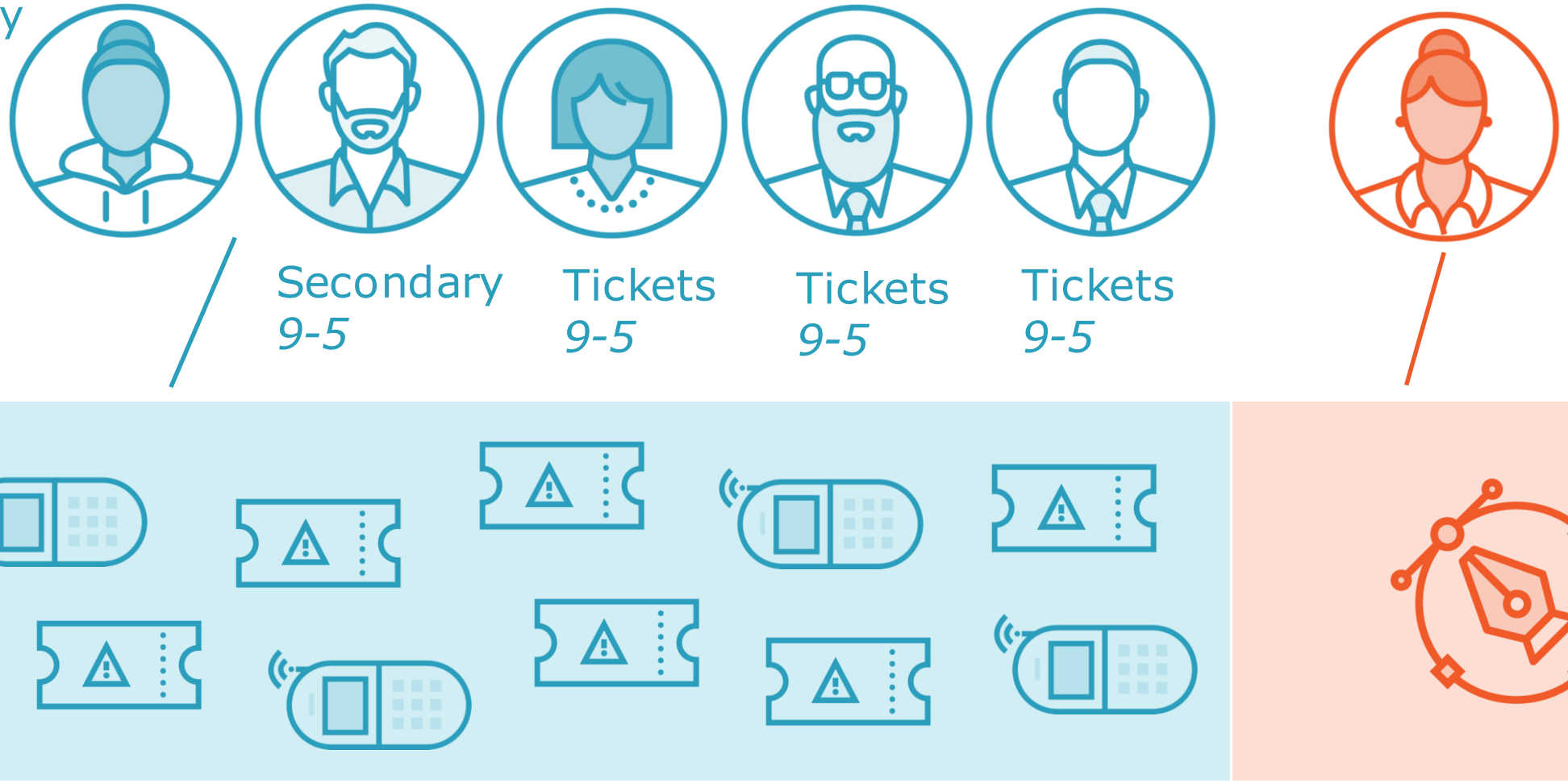
Secondary
9-5

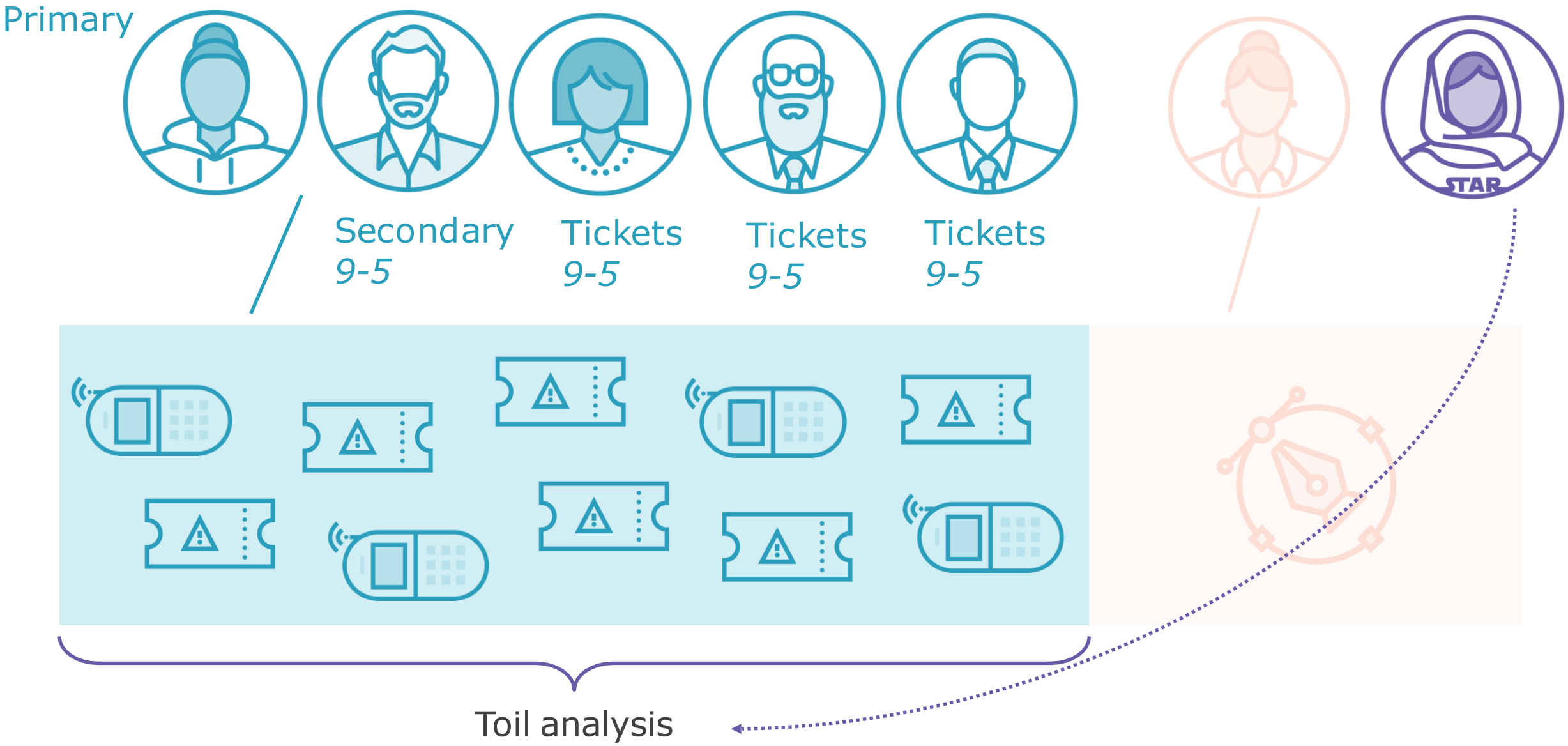
Tickets
9-5

Tickets
9-5



Primary




Primary
Secondary
9-5
Tickets
9-5
Tickets
9-5
Tickets
9-5
STAR
Toil analysis

Ops mode!

- Stick to SRE principles
- Reduce SLOs
- Change freeze

SRE is a balance

- Respect the time of the team
- Respect customer expectations

Summary

Incident management

- Incident Commander
- Ops Lead focuses on investigation
- Comms Lead informs stakeholders

Investigation model

- Triage, examine, diagnose, test, cure
- Tactics and output

Postmortems

- Blameless analysis
- Aiming for continuous improvement

Operational overload

- Managing on-call and workloads

More in SRE

Guidance

- Production readiness reviews
- Capacity planning
- Designing for simplicity

Practical advice

- Load balancing
- Cascading failures
- Testing for reliability

Follow the learning path!

- Here on Pluralsight

We're Done!

So...

- Please leave a rating
- Follow `@EltonStoneman` on Twitter
- Check out `blog.sixeyed.com`
- Watch my other courses ☺