Scaling towards a thousand micro services

DIEGO BERRUETA | ENGINEERING PRINCIPAL

About me

Born in Oviedo

Graduated from Uniovi

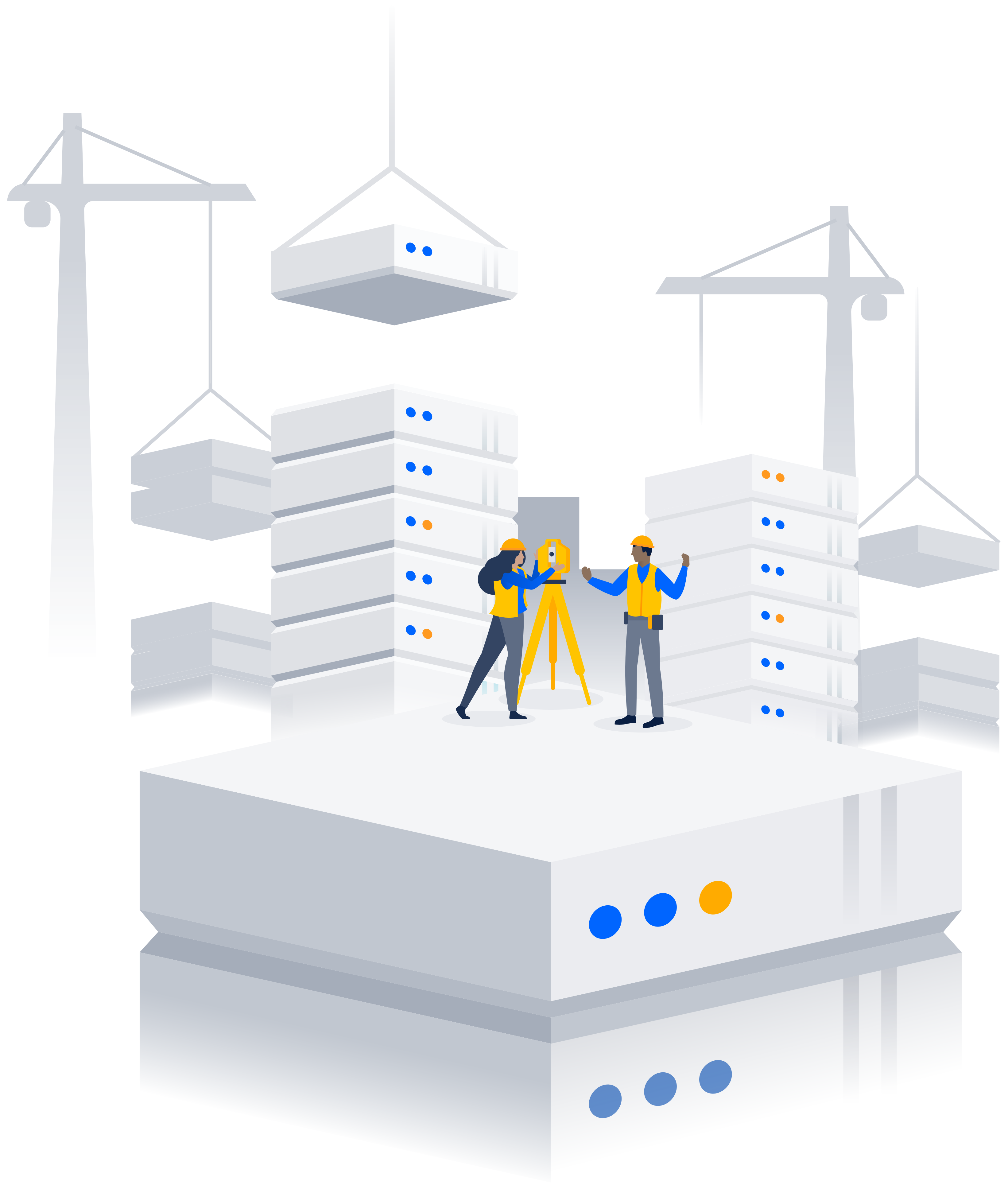
Moved to Sydney in 2012

Hiking addict,
space nerd

4

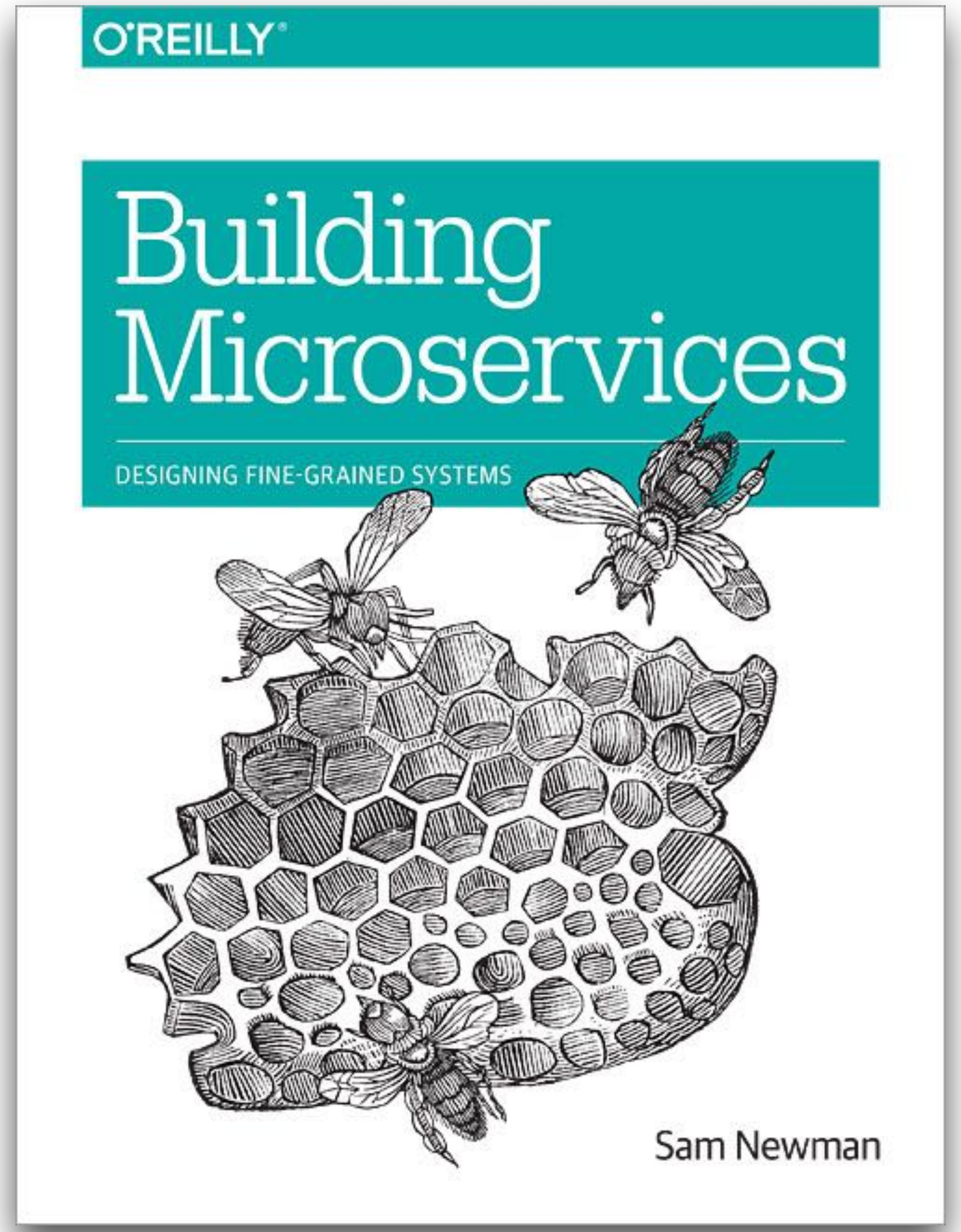
challenges

FIRST CHALLENGE



One does not simply...

Read this book



O'REILLY

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS

Sam Newman

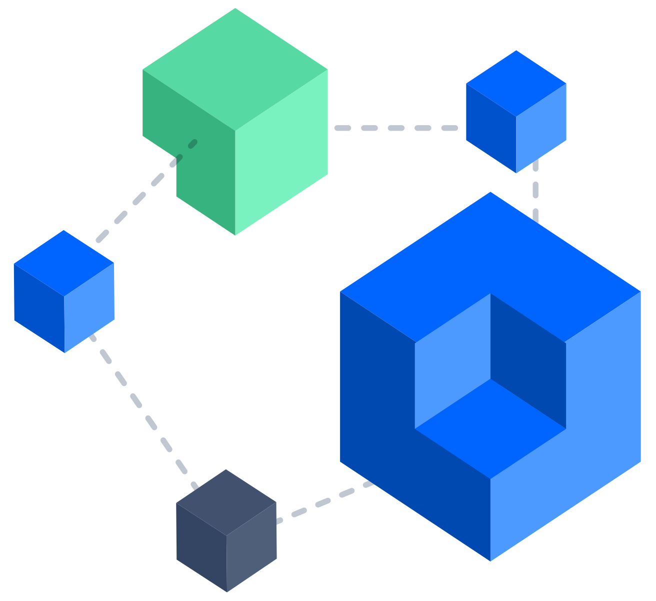
CHALLENGE

Enable every engineer to easily develop and deploy secure services

OUR SOLUTION

A secure platform with uniform processes

Micros: our internal PaaS



Built on top of AWS

Standardise service architecture and promote best practices (12 factor)

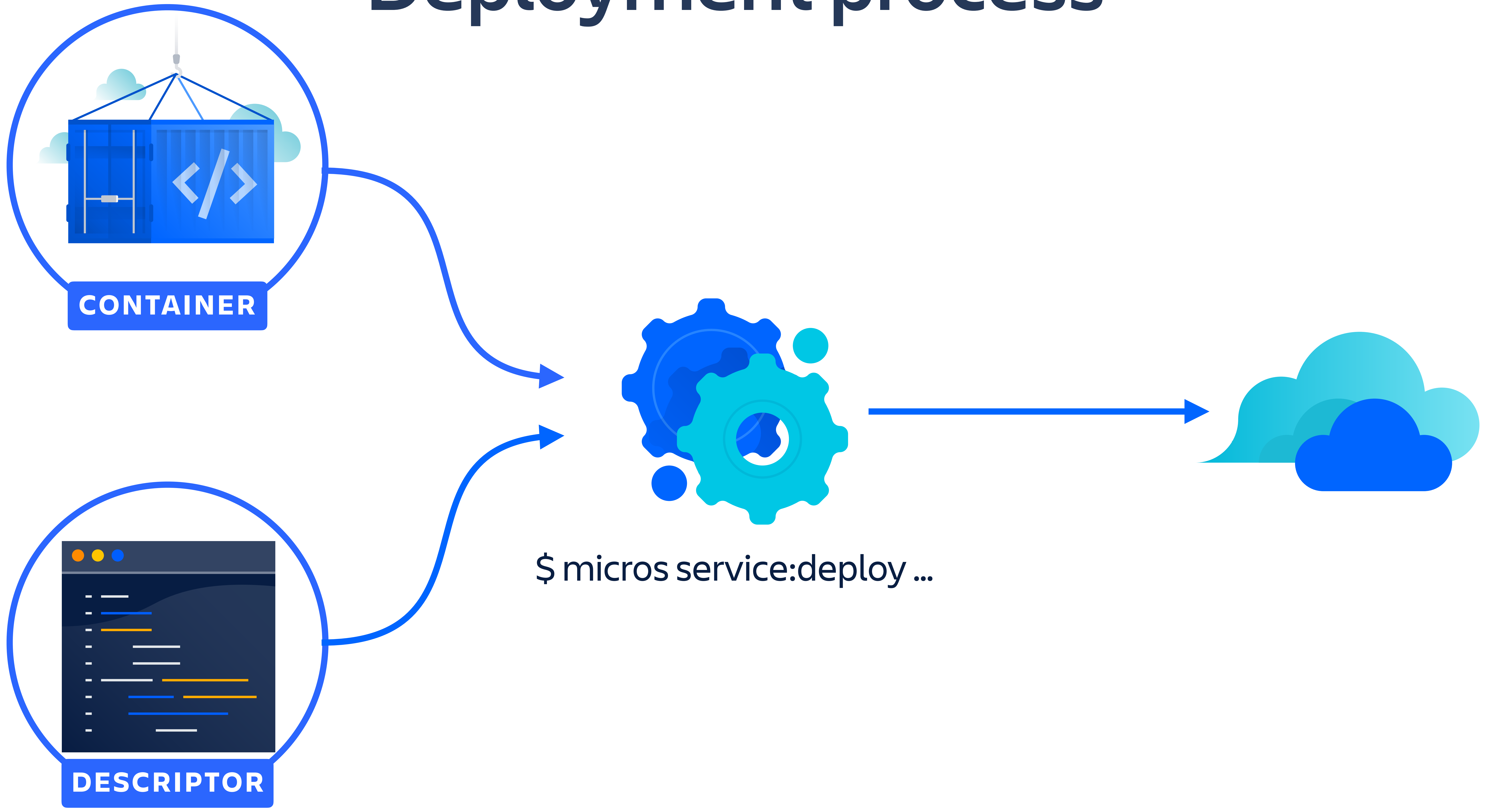
Clear contract

Similar packaging and operations

Uniform deployments

Familiar processes across all services and environments

Deployment process



```
$ micros service:deploy ...
```

Learn more about Micros

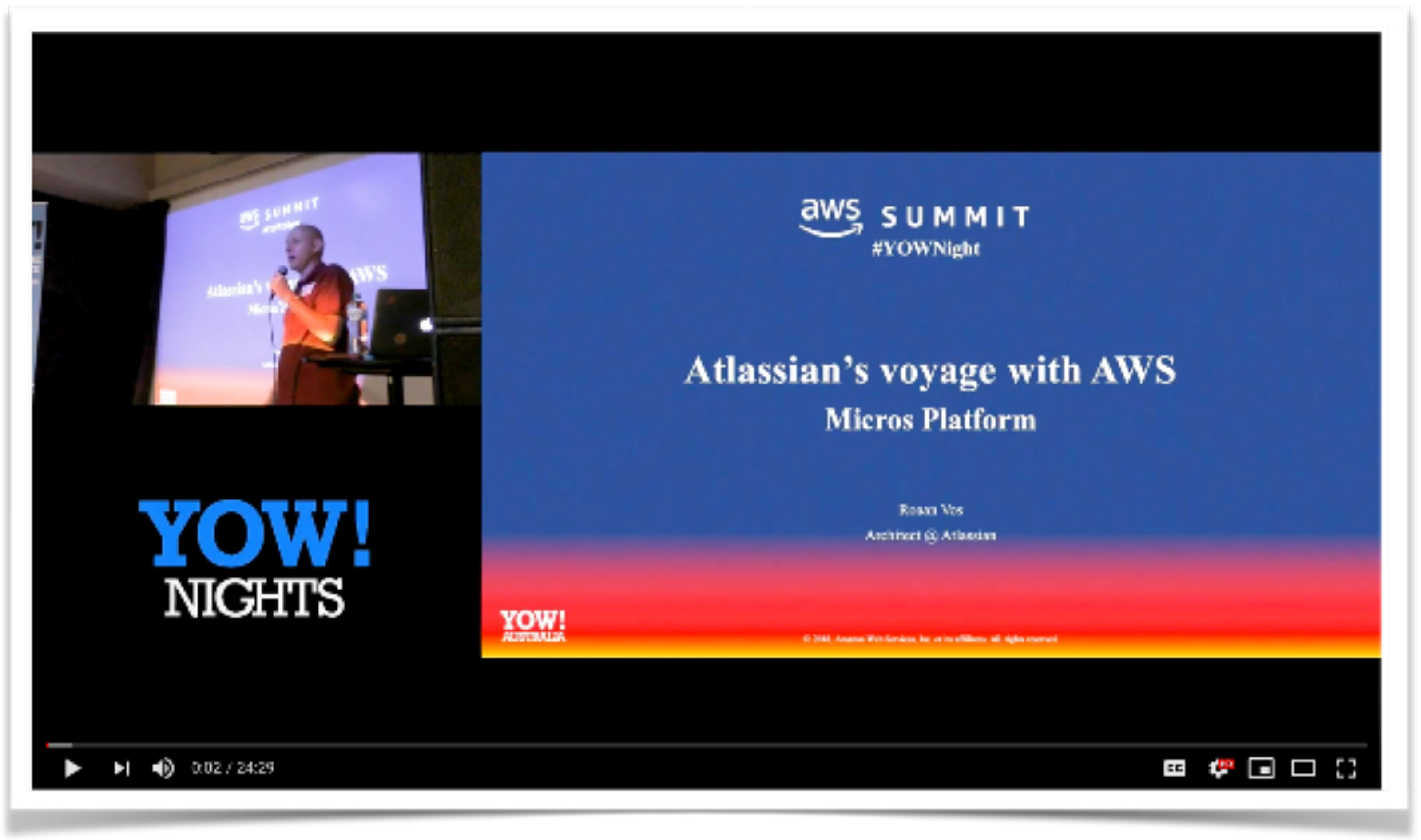
AtlasCamp
developers.connect()

Art of PaaS:
Lessons learned running "Micros",
a platform for hundreds of microservices

ROBIN FERNANDES · MICROS TEAM LEAD · ATLASSIAN · @REWBS

Robin Fernandes

Art of PaaS
(2016)

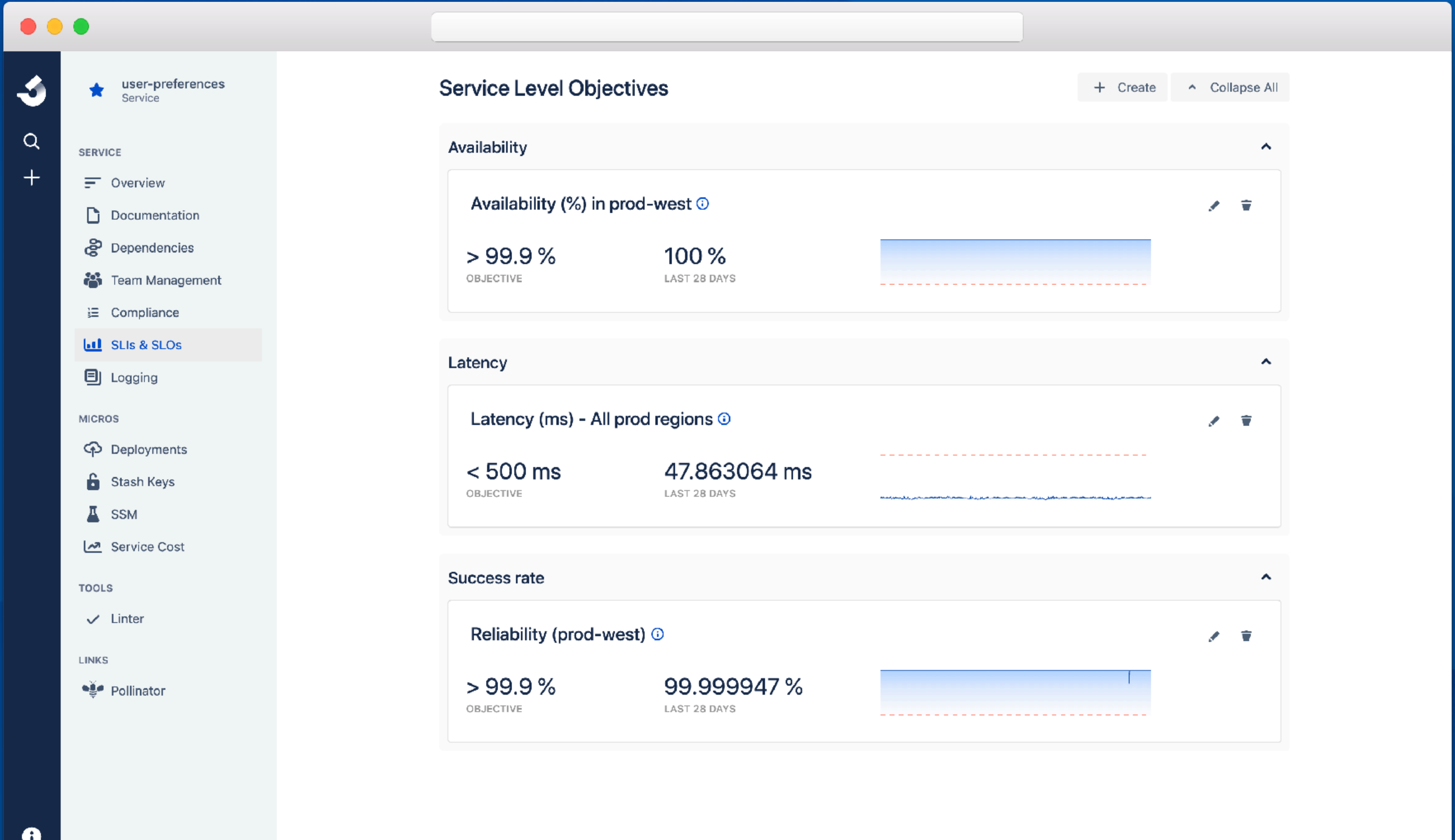

aws SUMMIT
#YOWNight

Atlassian's voyage with AWS
Micros Platform

Roaan Vos
Architect @ Atlassian

Roaan Vos

Atlassian's voyage with AWS
(2018)


user-preferences
Service

SERVICE
Overview
Documentation
Dependencies
Team Management
Compliance
SLIs & SLOs
Logging

MICROS
Deployments
Stash Keys
SSM
Service Cost

TOOLS
Linter

LINKS
Pollinator

Service Level Objectives

+ Create   Collapse All

Availability

Availability (%) in prod-west

> 99.9 %
OBJECTIVE

100 %
LAST 28 DAYS

Latency

Latency (ms) - All prod regions

< 500 ms
OBJECTIVE

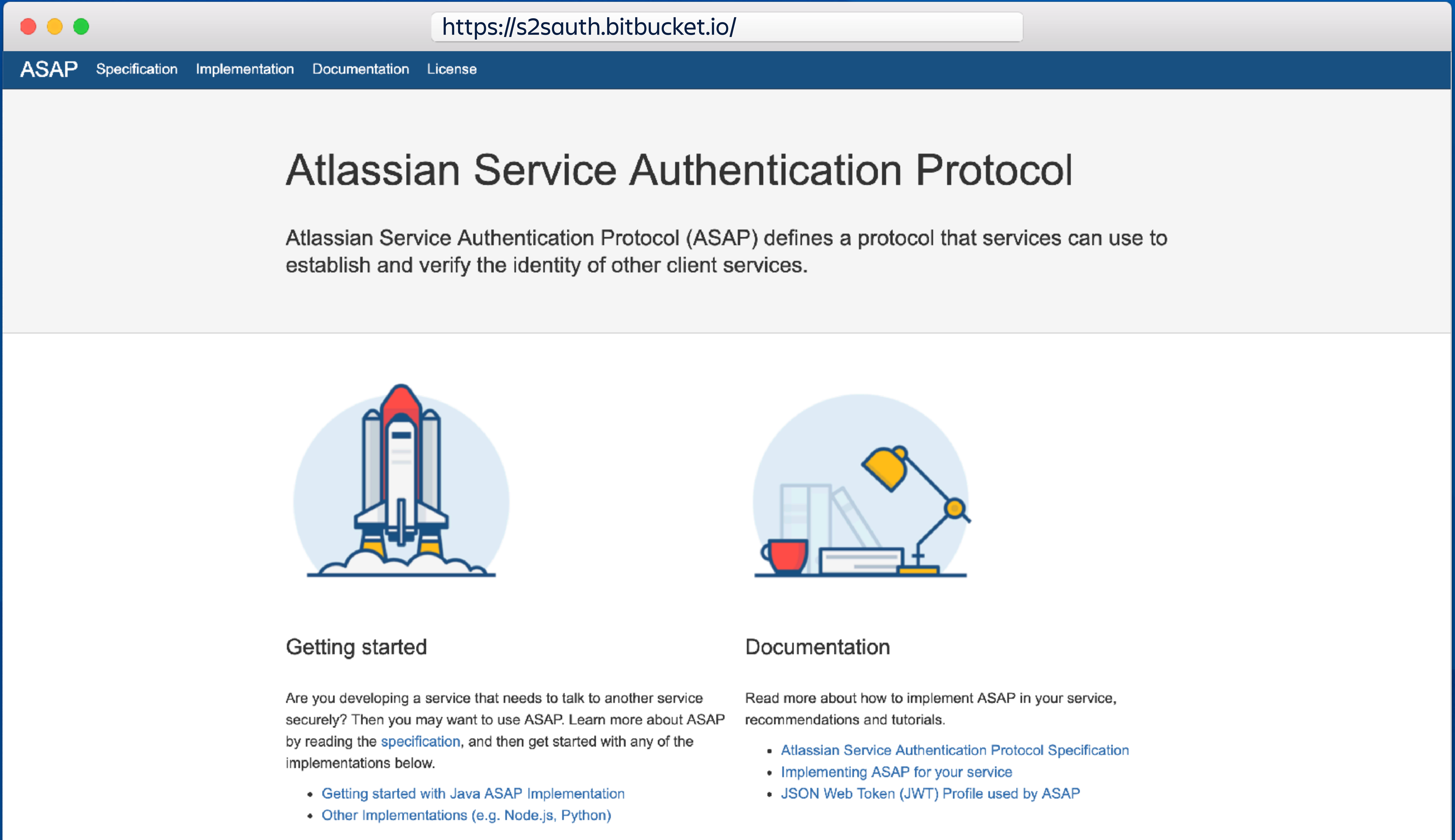
47.863064 ms
LAST 28 DAYS

Success rate

Reliability (prod-west)

> 99.9 %
OBJECTIVE

99.999947 %
LAST 28 DAYS



https://s2sauth.bitbucket.io/

Atlassian Service Authentication Protocol

Atlassian Service Authentication Protocol (ASAP) defines a protocol that services can use to establish and verify the identity of other client services.

Getting started

Are you developing a service that needs to talk to another service securely? Then you may want to use ASAP. Learn more about ASAP by reading the specification, and then get started with any of the implementations below.

- Getting started with Java ASAP Implementation
- Other Implementations (e.g. Node.js, Python)

Documentation

Read more about how to implement ASAP in your service, recommendations and tutorials.

- Atlassian Service Authentication Protocol Specification
- Implementing ASAP for your service
- JSON Web Token (JWT) Profile used by ASAP

Value unlocked

Security
Service-to-service authentication, security scans, secret store...

Resilience
Chaos engineering, failover tests and automated backups

Compliance
Change traceability and strict permissions

Visibility
Cost allocation and optimisation, bird's eye view of the platform

SECOND CHALLENGE

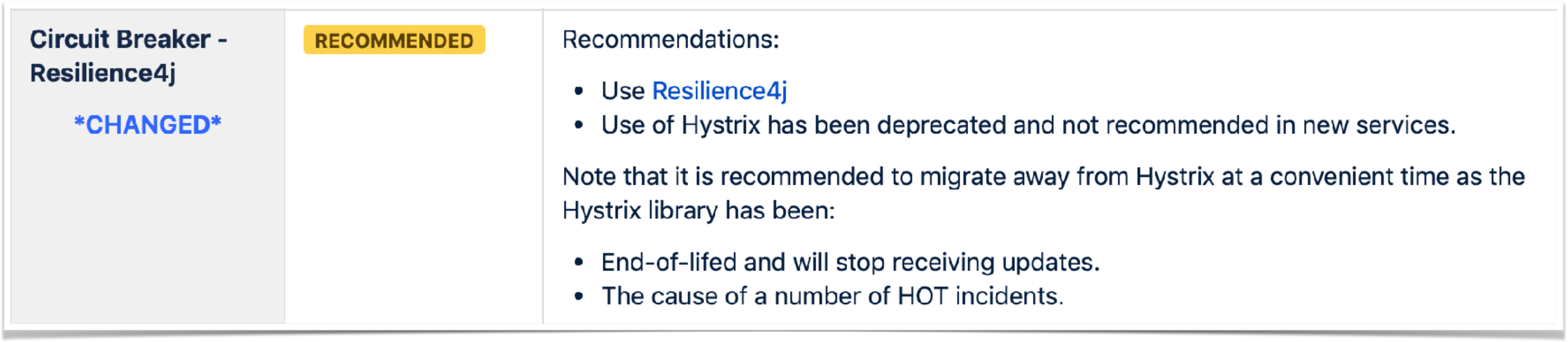
CHALLENGE

Reuse knowledge and enable experimentation

OUR SOLUTION

Converge on a handful of tech stacks

Tech stack guidelines



Circuit Breaker - Resilience4j *CHANGED*	RECOMMENDED	Recommendations: • Use Resilience4j • Use of Hystrix has been deprecated and not recommended in new services. Note that it is recommended to migrate away from Hystrix at a convenient time as the Hystrix library has been: • End-of-lifed and will stop receiving updates. • The cause of a number of HOT incidents.

(Extract from the Java tech stack)

Value unlocked

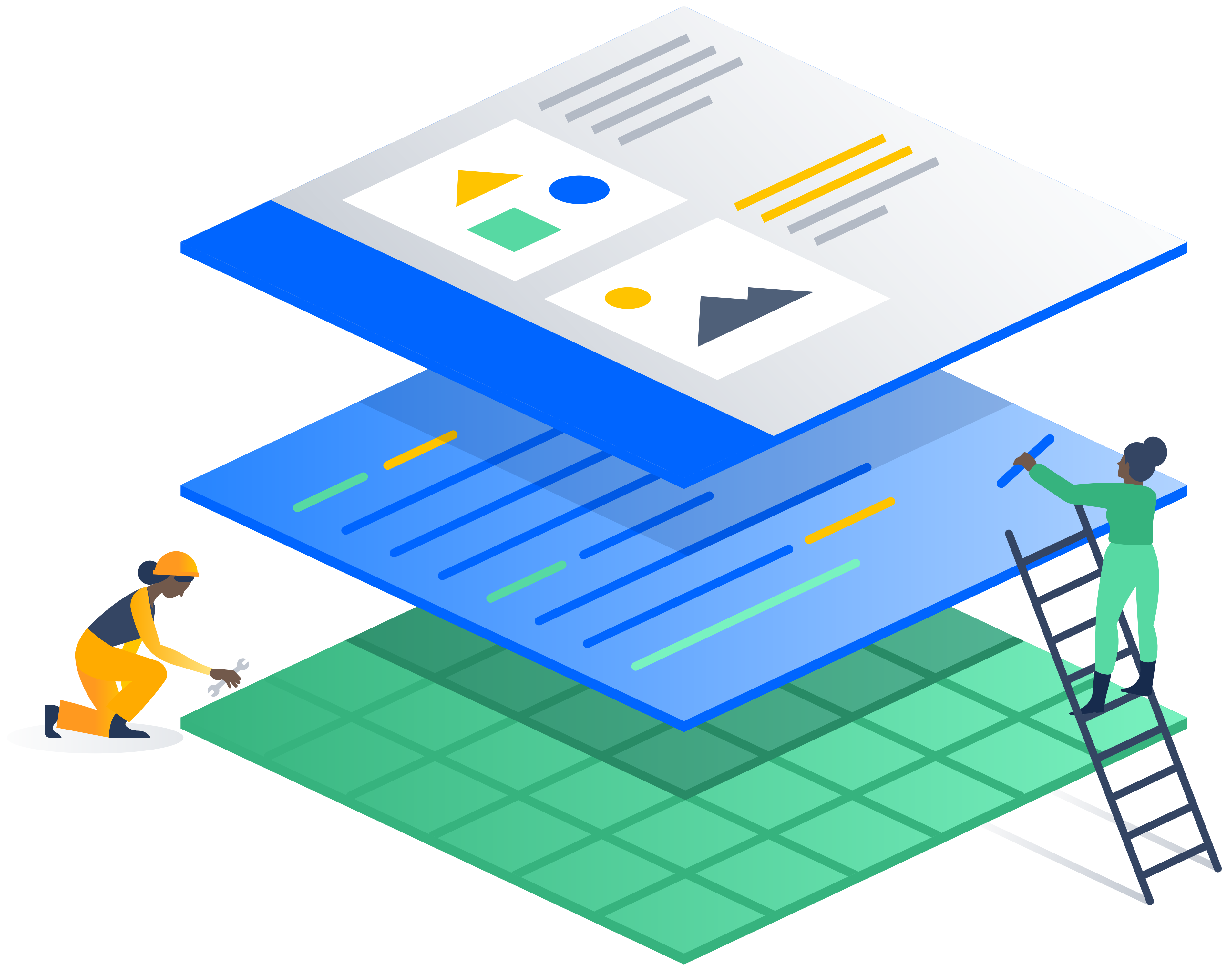
Economies of scale
Shared libraries and tools are used in hundreds of services

Training
Internal brownbags, intranet blog posts, peer support

Collaboration
Engineers can understand and contribute to other teams' services

Experimentation
Recommendations are regularly updated (example: Kotlin)

THIRD CHALLENGE

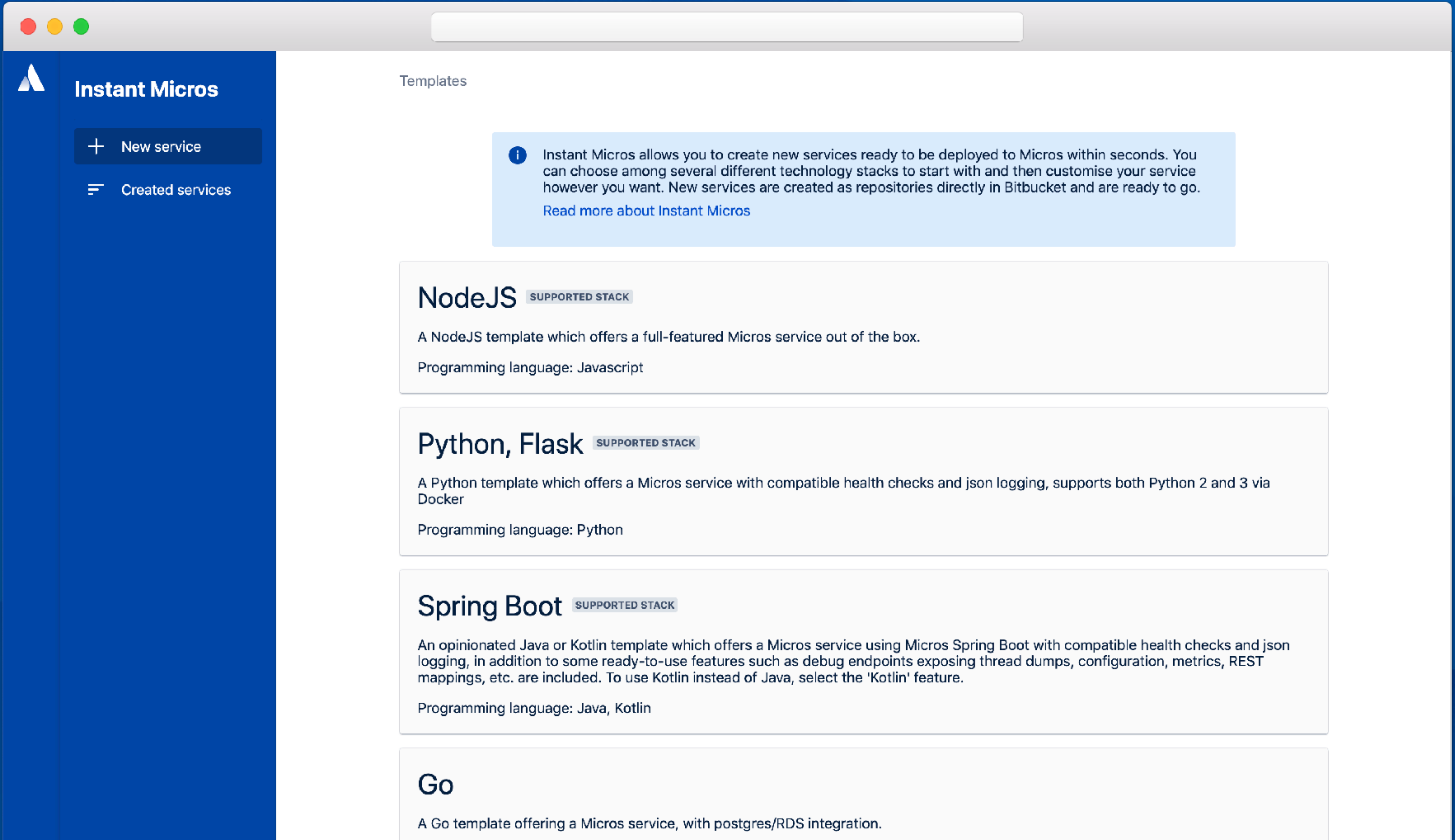


CHALLENGE

Copy-pasting code does not scale

OUR SOLUTION

Cookie-cutter approach to service creation and maintenance



Instant Micros allows you to create new services ready to be deployed to Micros within seconds. You can choose among several different technology stacks to start with and then customise your service however you want. New services are created as repositories directly in Bitbucket and are ready to go.

Read more about Instant Micros

NodeJS SUPPORTED STACK

A NodeJS template which offers a full-featured Micros service out of the box.

Programming language: Javascript

Python, Flask SUPPORTED STACK

A Python template which offers a Micros service with compatible health checks and json logging, supports both Python 2 and 3 via Docker

Programming language: Python

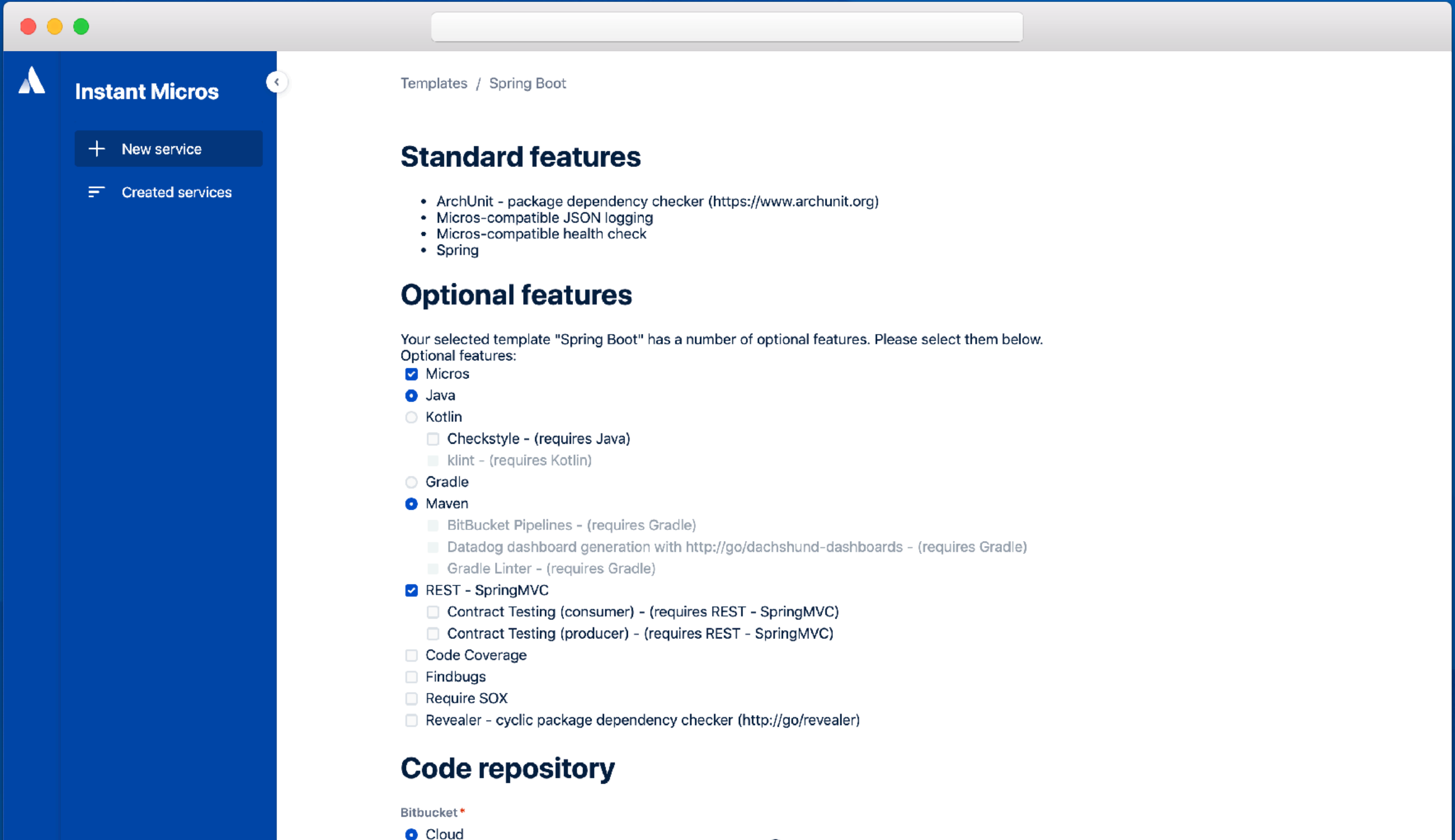
Spring Boot SUPPORTED STACK

An opinionated Java or Kotlin template which offers a Micros service using Micros Spring Boot with compatible health checks and json logging, in addition to some ready-to-use features such as debug endpoints exposing thread dumps, configuration, metrics, REST mappings, etc. are included. To use Kotlin instead of Java, select the 'Kotlin' feature.

Programming language: Java, Kotlin

Go

A Go template offering a Micros service, with postgres/RDS integration.



Templates / Spring Boot

Standard features

- ArchUnit - package dependency checker (https://www.archunit.org)
- Micros-compatible JSON logging
- Micros-compatible health check
- Spring

Optional features

Your selected template "Spring Boot" has a number of optional features. Please select them below.
Optional features:

[✓ Micros]
[✓ Java]
Kotlin
  Checkstyle - (requires Java)
  klint - (requires Kotlin)
Gradle
[✓ Maven]
  BitBucket Pipelines - (requires Gradle)
  Datadog dashboard generation with http://go/dachshund-dashboards - (requires Gradle)
  Gradle Linter - (requires Gradle)
[✓ REST - SpringMVC]
  Contract Testing (consumer) - (requires REST - SpringMVC)
  Contract Testing (producer) - (requires REST - SpringMVC)
Code Coverage
Findbugs
Require SOX
Revealer - cyclic package dependency checker (http://go/revealer)

Code repository

Bitbucket*
[✓ Cloud]

Value unlocked

Quick prototyping

Create and deploy a new service in minutes

Scalable maintenance

Fix it once for everyone by eliminating code duplication

Frictionless decomposition

Avoid temptation to add more code to the monolith

Reuse best practices

From resilience to code organisation

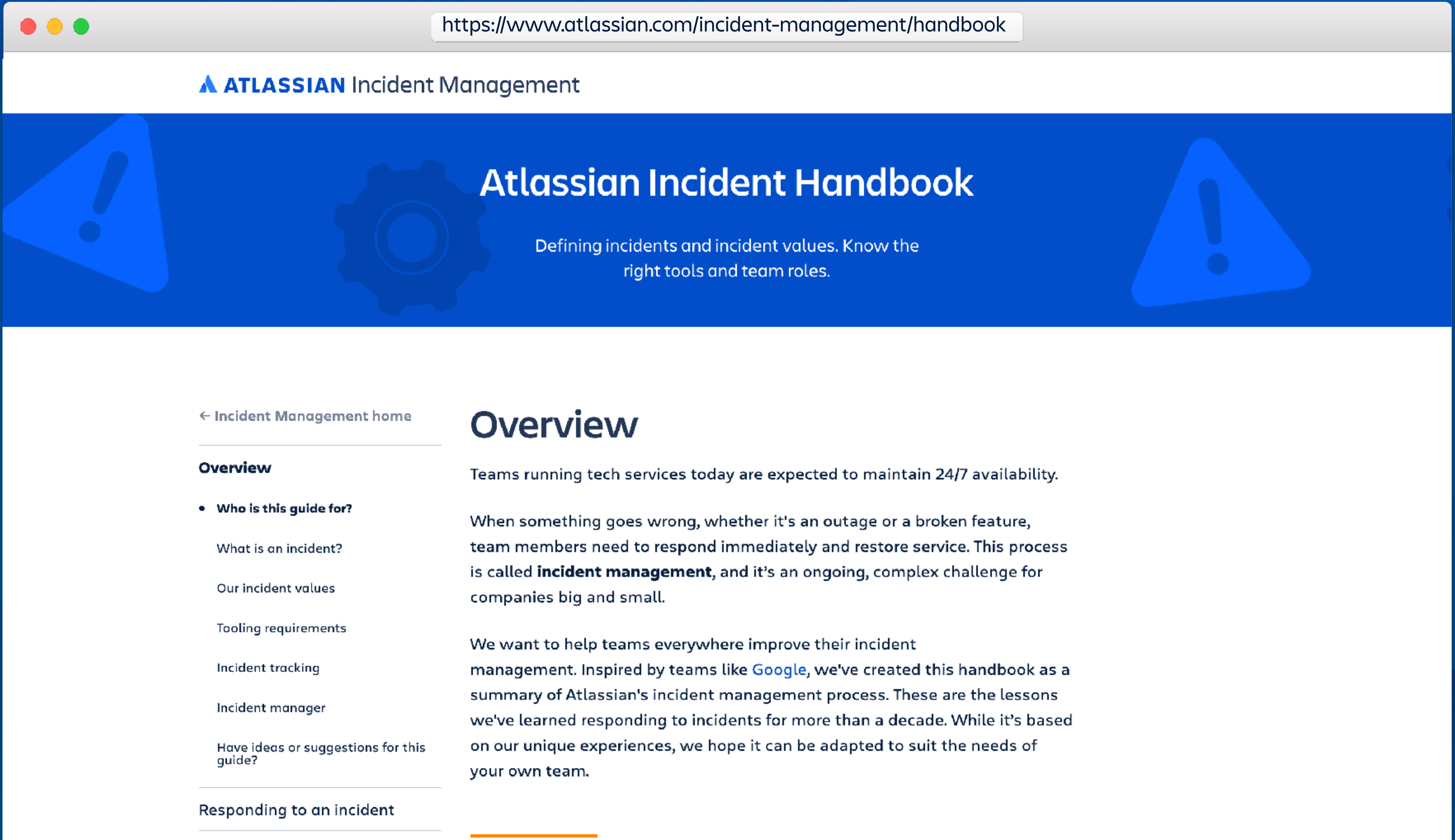
FOURTH CHALLENGE

CHALLENGE

A sustainable balance between speed and reliability

OUR SOLUTION

End-to-end service ownership and continuous improvement



https://www.atlassian.com/incident-management/handbook

ATLASSIAN Incident Management

Atlassian Incident Handbook

Defining incidents and incident values. Know the right tools and team roles.

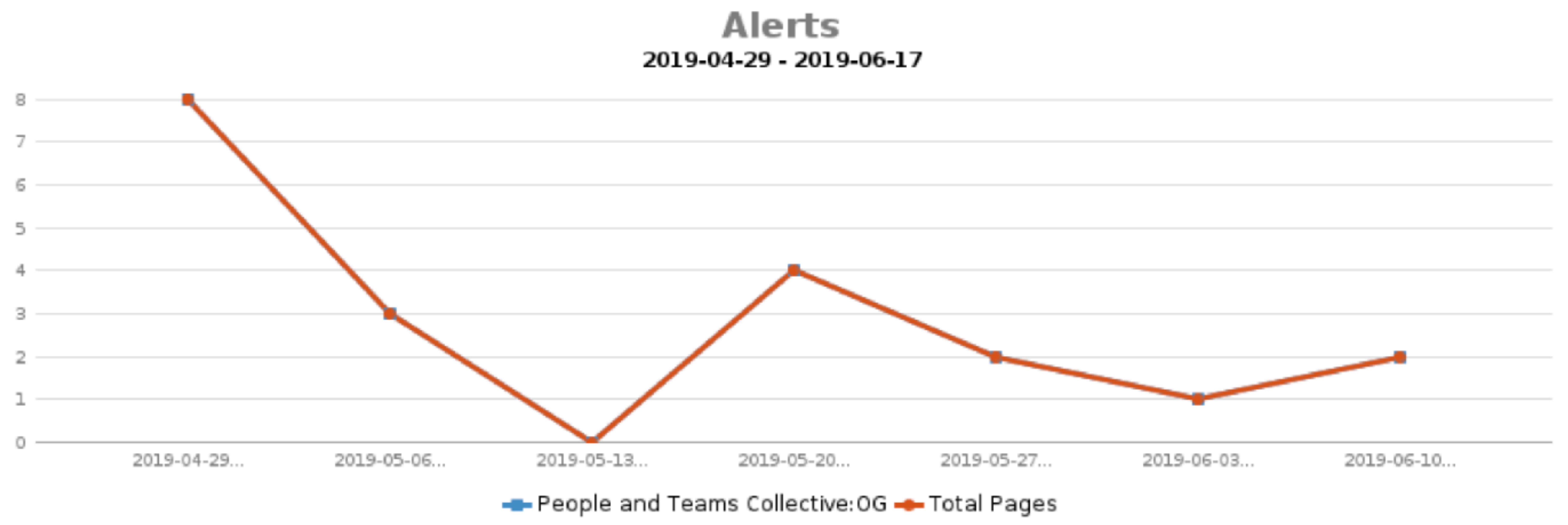
Overview

Teams running tech services today are expected to maintain 24/7 availability.

When something goes wrong, whether it's an outage or a broken feature, team members need to respond immediately and restore service. This process is called **incident management**, and it's an ongoing, complex challenge for companies big and small.

We want to help teams everywhere improve their incident management. Inspired by teams like Google, we've created this handbook as a summary of Atlassian's incident management process. These are the lessons we've learned responding to incidents for more than a decade. While it's based on our unique experiences, we hope it can be adapted to suit the needs of your own team.

Continuous improvement



Value unlocked

Transparency
Teams set their objectives and openly track their success

Close feedback loop
Teams are motivated and empowered to continuously learn and improve

Trust
Blameless incident investigations find and address root cause

Scalable model
Decentralised operations scale horizontally and interests are aligned

Recap

1st challenge:
A secure platform

2nd challenge:
Knowledge reuse

3rd challenge:
Speed at scale

4th challenge:
Operational excellence

Thank you

DIEGO BERRUETA