

From Cloud-Native Apps to Cloud-Native Platforms

Abby Bangser

Architecting for producers will optimise platforms to meet growing expectations



Truths about system optimisation

Truths about system optimisation

1

A bottleneck defines
the scale limit for a system.

Truths about system optimisation

1

A bottleneck defines the scale limit for a system.

2

Any optimising outside the bottleneck is wasted effort.

Truths about system optimisation

1

A bottleneck defines the scale limit for a system.

2

Any optimising outside the bottleneck is wasted effort.

3

All systems have a bottleneck, it just moves as optimisations are made.



**The biggest bottleneck in
platforms is growing capabilities**



BICYCLE DUTCH

[Home](#)

[About](#)

[Theme index](#)

[Map](#)

[YouTube](#)

[In other languages](#) ▾



Amsterdam commute by bicycle





Home

About

Th

Fat tire bikes are faster and getting more popular



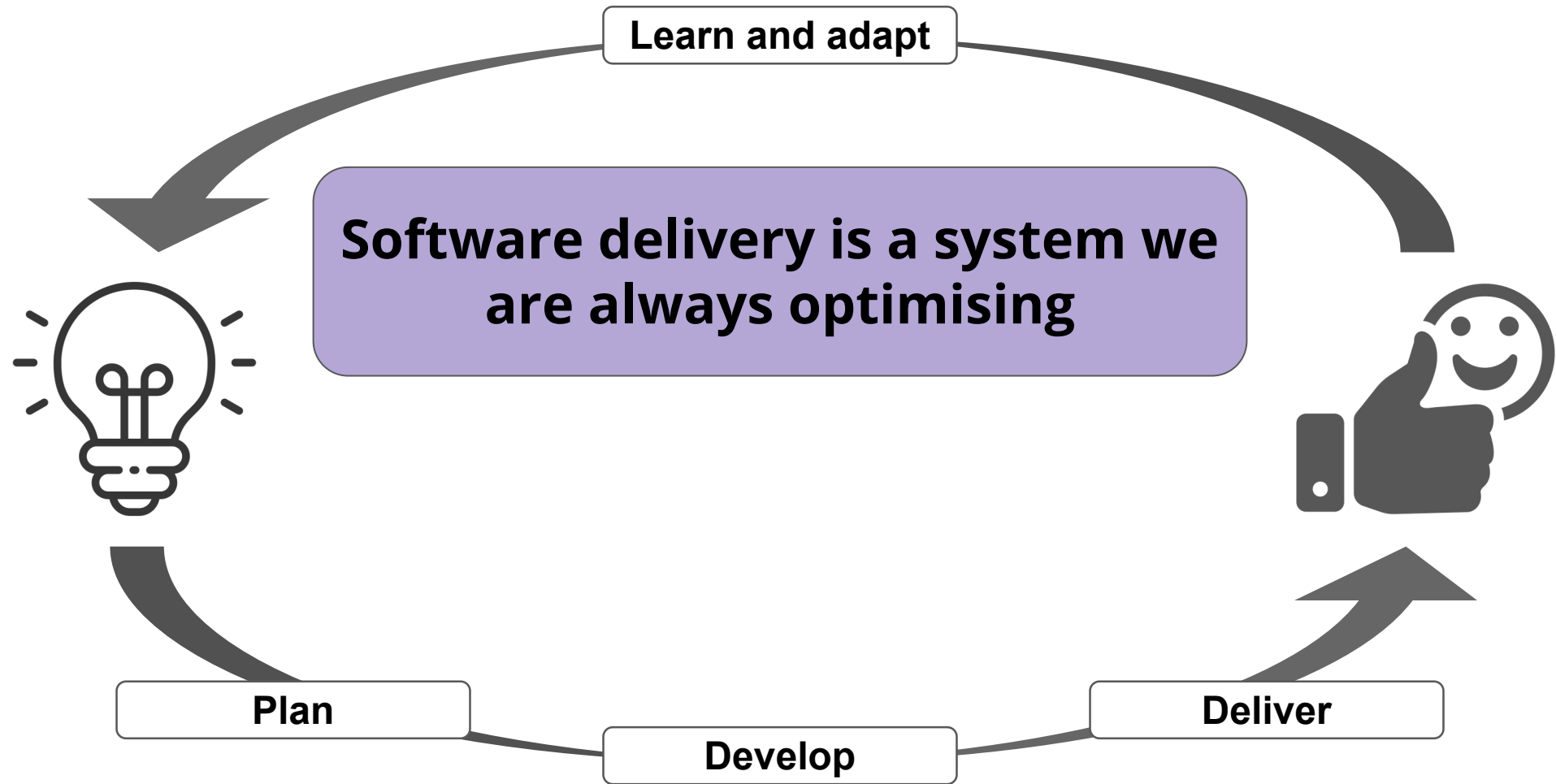
Amsterdam commute by bicycle

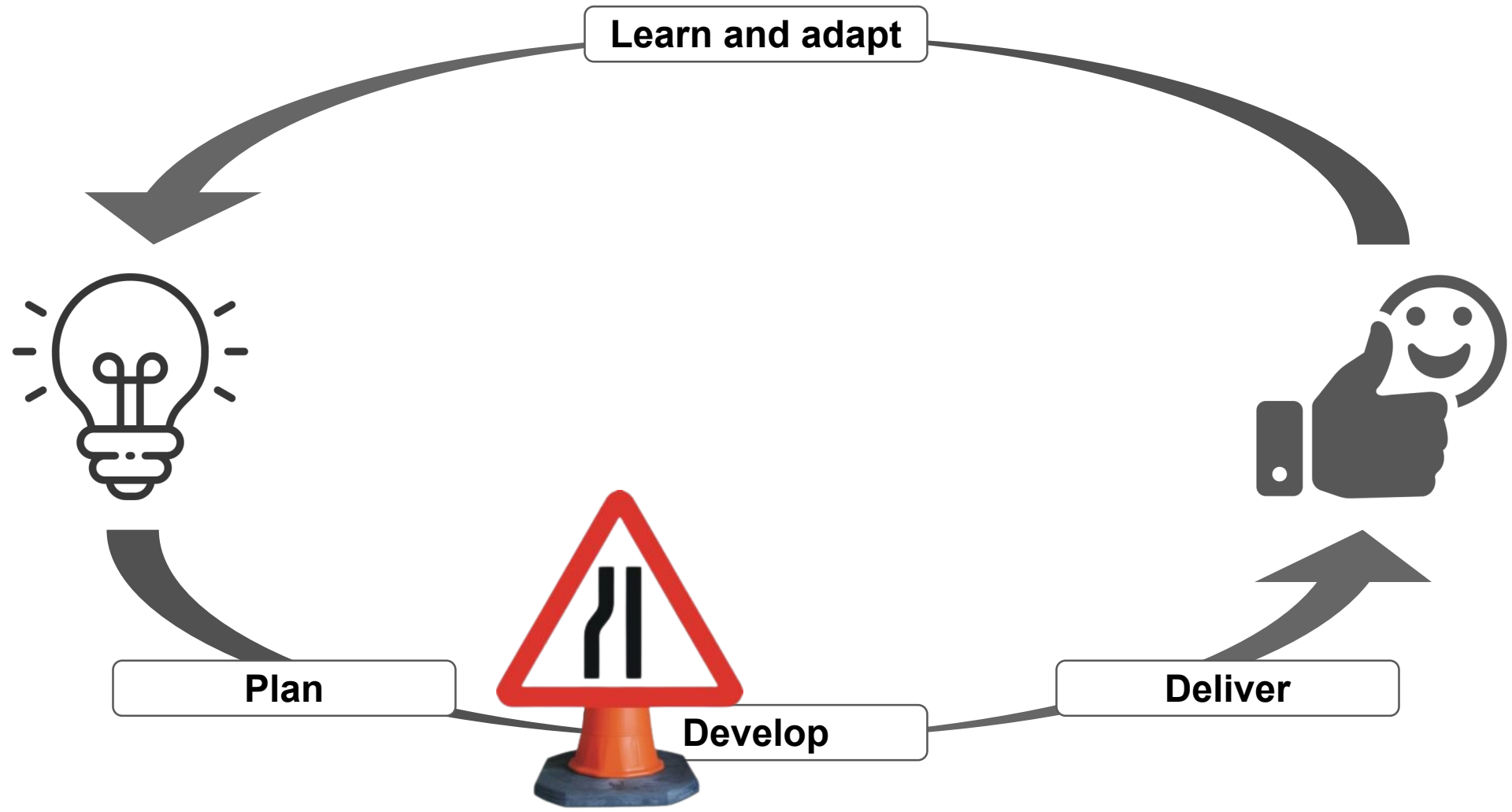


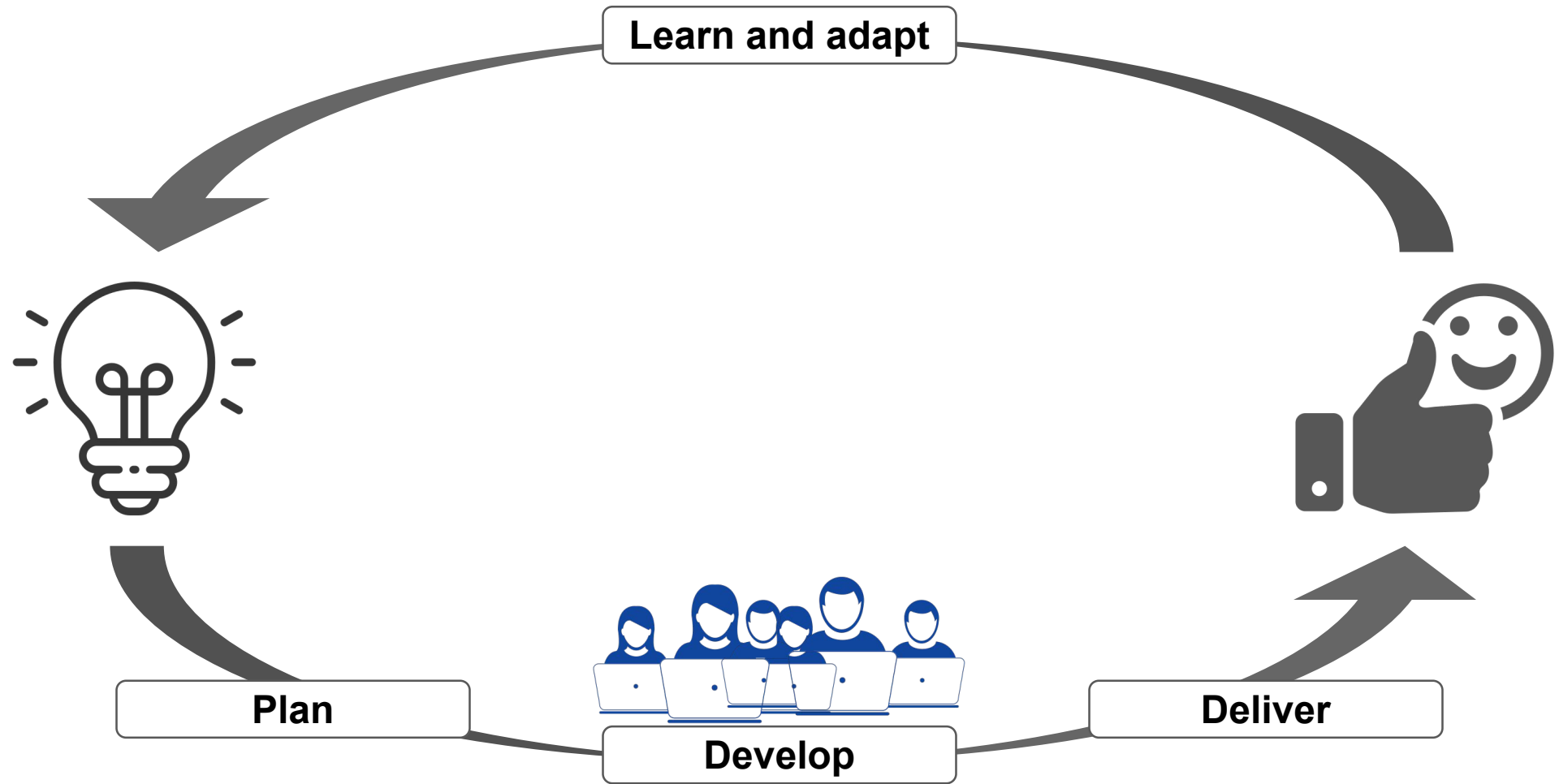


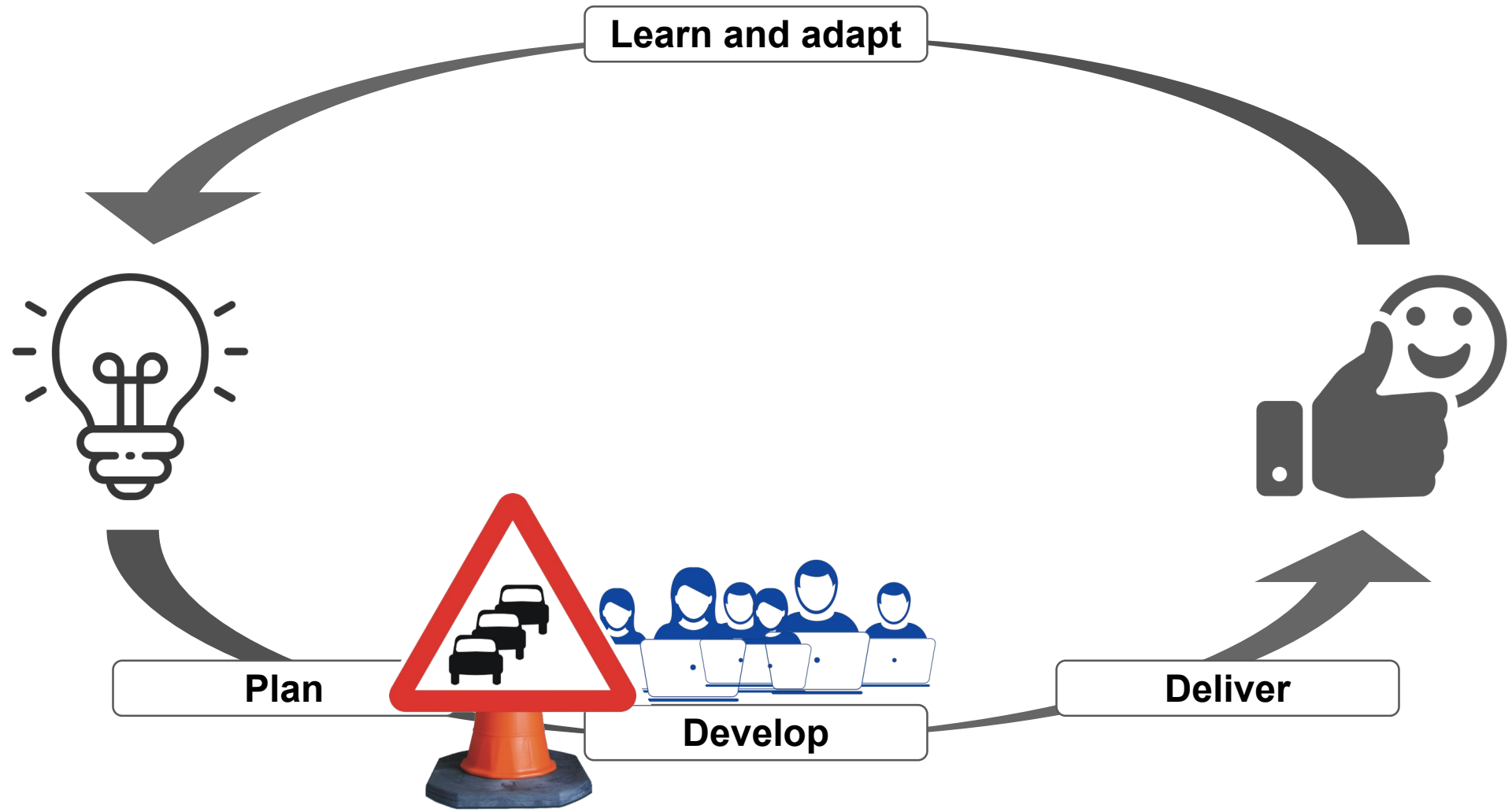
**But in a city of canals,
top speed isn't the bottleneck**

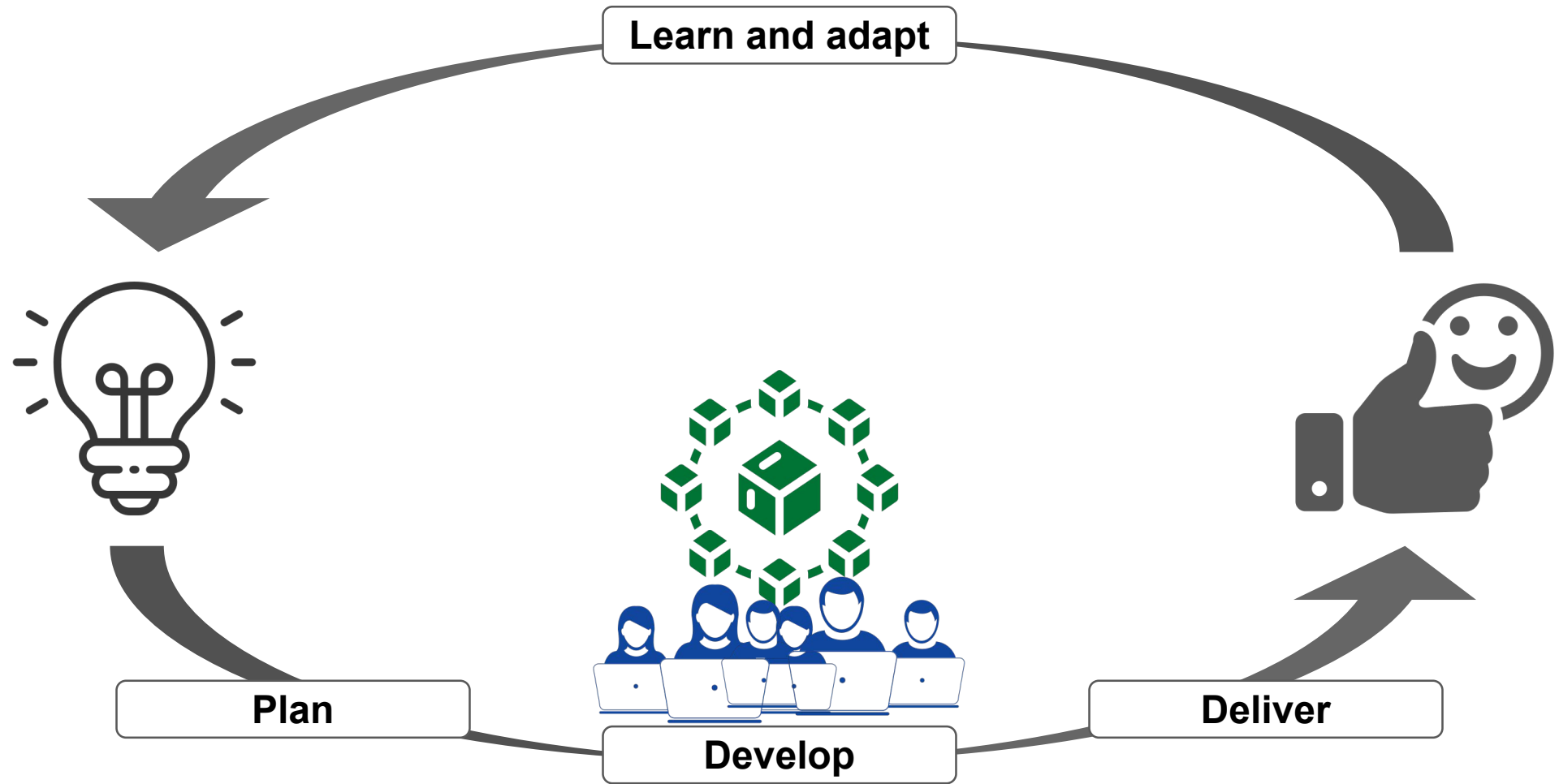
Thank you Paco Bernabé for the picture!











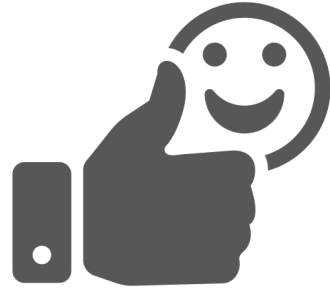
Learn and adapt

**System optimisation continues
chasing the ever moving itch**

Plan

Develop

Deliver



Truths about system optimisation

1

A bottleneck defines the scale limit for a system.

2

Any optimising outside the bottleneck is wasted effort.

3

All systems have a bottleneck, it just moves as optimisations are made.

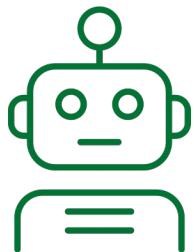
Truths about system optimisation

1

A bottleneck defines
the scope of the problem.

**AI has changed the bottleneck.
Dependencies are now in focus.**

Systems have a bottleneck,
it just moves as optimisations are made.



```
$ agent
```

```
> Create a new app that provides value
```

- I understand the goal of creating a new application. For this I need a database. Requesting a database from the platform.
- Ran ticket issue create...
Waiting for response...



Agents can only move as fast as their platform

```
$ agent
```

```
> Create a new app that will make $$$
```

- I understand the goal of creating a new application. For this I need a database. Requesting a database from the platform.
- Ran ticket issue create...
Waiting for response...
- It's been 84 years...

Principles of Optimized Platform Engineering



API first self-service

Full lifecycle
business compliance

Fleet management by
capability providers

**"Platform Capabilities are for life"
just like puppies**



Optimized Platform Engineering

MISSING QR
CODE

Principles

API first self-service

Full lifecycle
business compliance

Fleet management by
capability providers

Strategies

Buy what is not unique
to you

Build only for multiple
use cases

Evolve continuously

Tactics

Platform Engineers
build marketplaces

Platforms set and
support standards

Enable producers as
first-class users



**The biggest bottleneck in
Platform Engineering is supply**

Vertical scaling

add platform engineers



Vertical scaling

add platform engineers



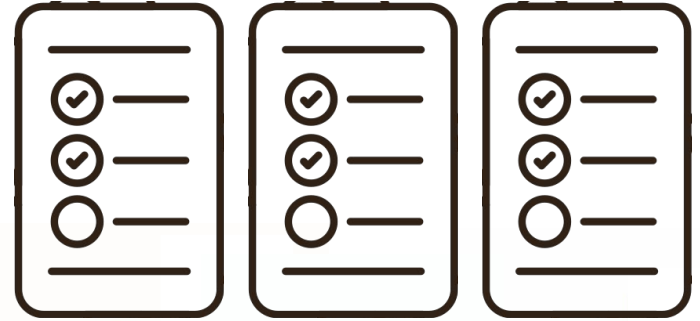
Vertical scaling

add platform engineers



Horizontal scaling

add independent producers



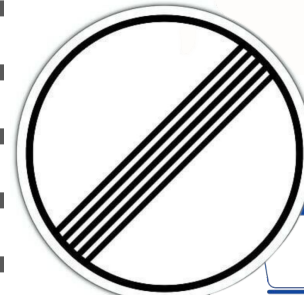
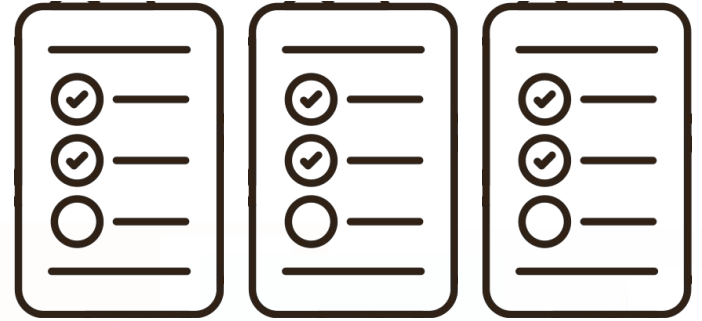
Vertical scaling

add platform engineers



Horizontal scaling

add independent producers



Enable producers *as customers*



Internal *multiplayer* platform



Self-service capability creation API



Standards



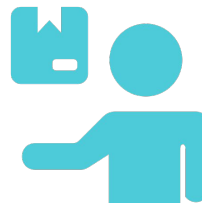
Support



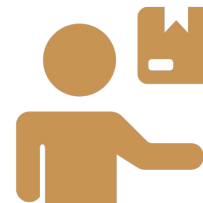
Tools



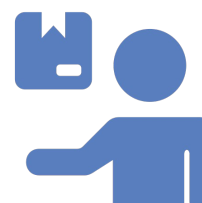
Security



Data



Vendor



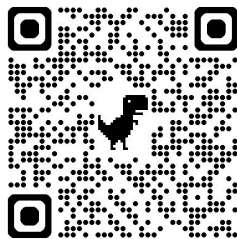
Finance

Consumer API



The platform defines
a good *consumer* citizen.

Adhering provides benefits.



THE TWELVE-FACTOR APP

The platform defines
a good *consumer* citizen.

Adhering provides benefits.

I. Codebase

One codebase tracked in revision control, many deploys

II. Dependencies

Explicitly declare and isolate dependencies

III. Config

Store config in the environment

IV. Backing services

Treat backing services as attached resources

V. Build, release, run

Strictly separate build and run stages

VI. Processes

Execute the app as one or more stateless processes

VII. Port binding

Export services via port binding

VIII. Concurrency

Scale out via the process model

IX. Disposability

Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity

Keep development, staging, and production as similar as possible

XI. Logs

Treat logs as event streams

XII. Admin processes

Run admin/management tasks as one-off processes

The platform needs to define
a good *producer* citizen.

Adhering provides benefits.

Consumer API

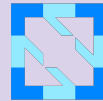
Internal *multiplayer* platform



kro



KUSION
STACK



The platform needs to define
a good *producer* citizen.

Adhering provides benefits.

Consumer API

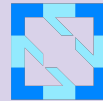
Internal *multiplayer* platform



kro



KUSION
STACK

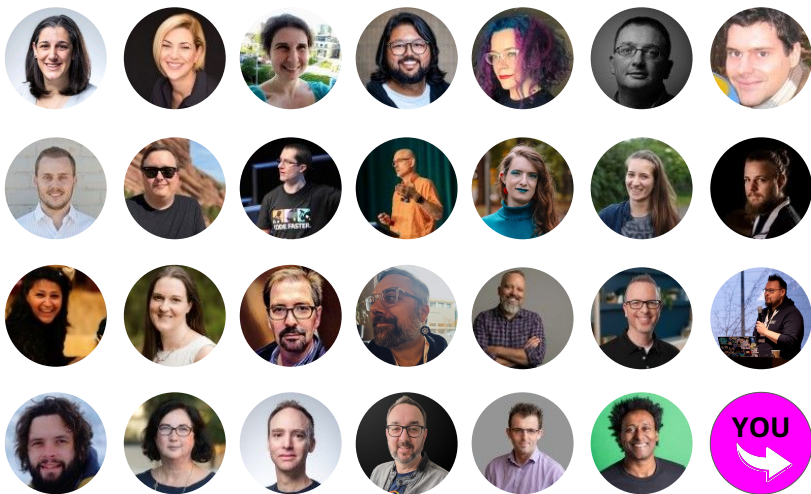


Producer API

The platform needs to define
a good *producer* citizen.

Adhering provides benefits.

1. Produced by API
2. Consumed by API
3. Self-managed Value Stream
4. Reconciled State
5. Implementation Agnostic
6. Fleet-Managed Instances
7. Operational clarity
8. Continuous Governance



Thank you to all of the early reviewers! Note that early participation does not indicate final approval.



CLOUD NATIVE PLATFORM ENGINEERING COMMUNITY



[Initiative] Call for Contributors - Platform Engineering Maturity Model Update (for completion by Kubecon EU) #63

[Open](#)

Labels

[Platform Maturity Model](#)

[Initiative] Call for Contributors - Platforms Whitepaper Update (for completion by Kubecon EU) #55

[Open](#)

Labels

[Platform Whitepaper](#)



[Community] Platform producer factors #79

[Open](#)

Architecting for producers will optimise platforms to meet growing expectations





**Architecting for producers will optimise
platforms to meet growing expectations**

**Community driven standards will
lower the cost to achieving this vision**