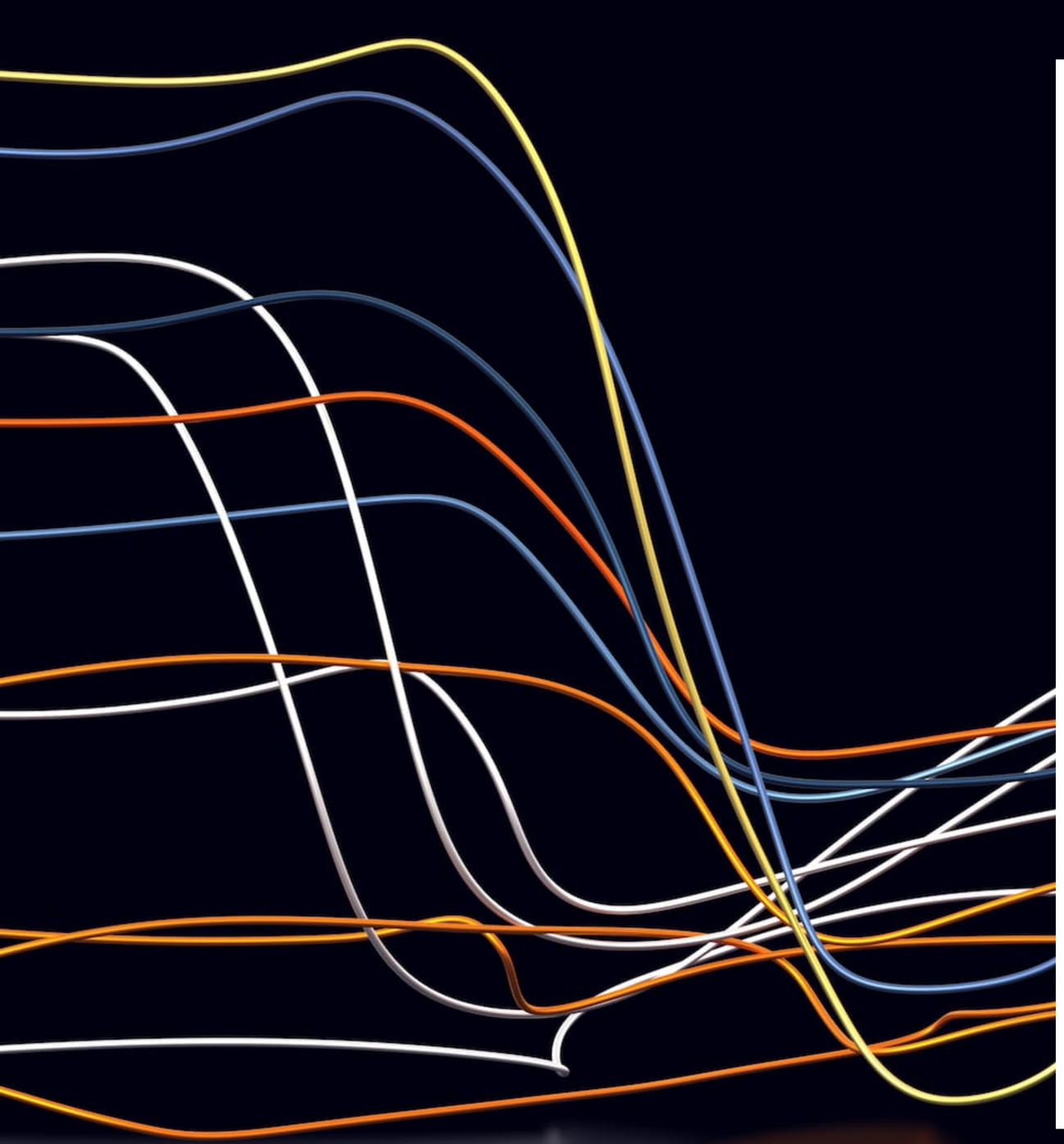


Technology Day 2022

dagger.io

(Not only) local CI/CD pipelines without the YAML hell

INNOQ

FABIAN KRETZER
INNOQ.SOCIAL/@FABIAN

Our journey

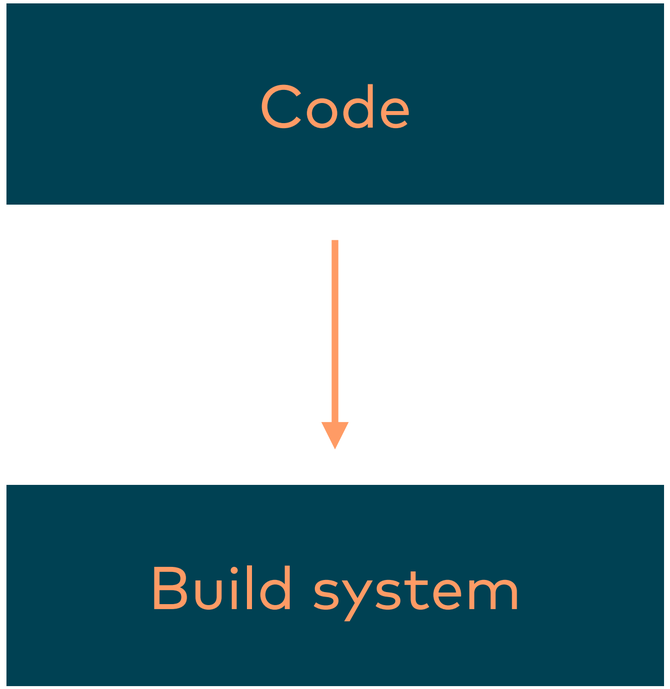
- Why?
- Origins
- Building blocks
- Concepts
- Example
- Future
- Opinion(s)

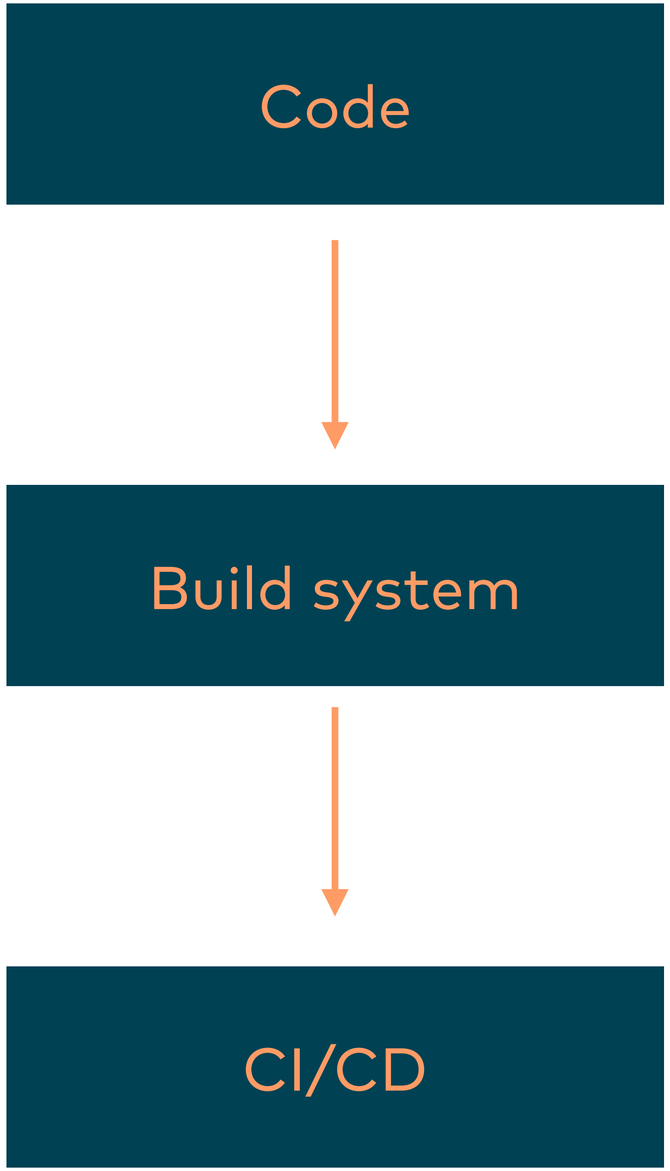
© dagger.io

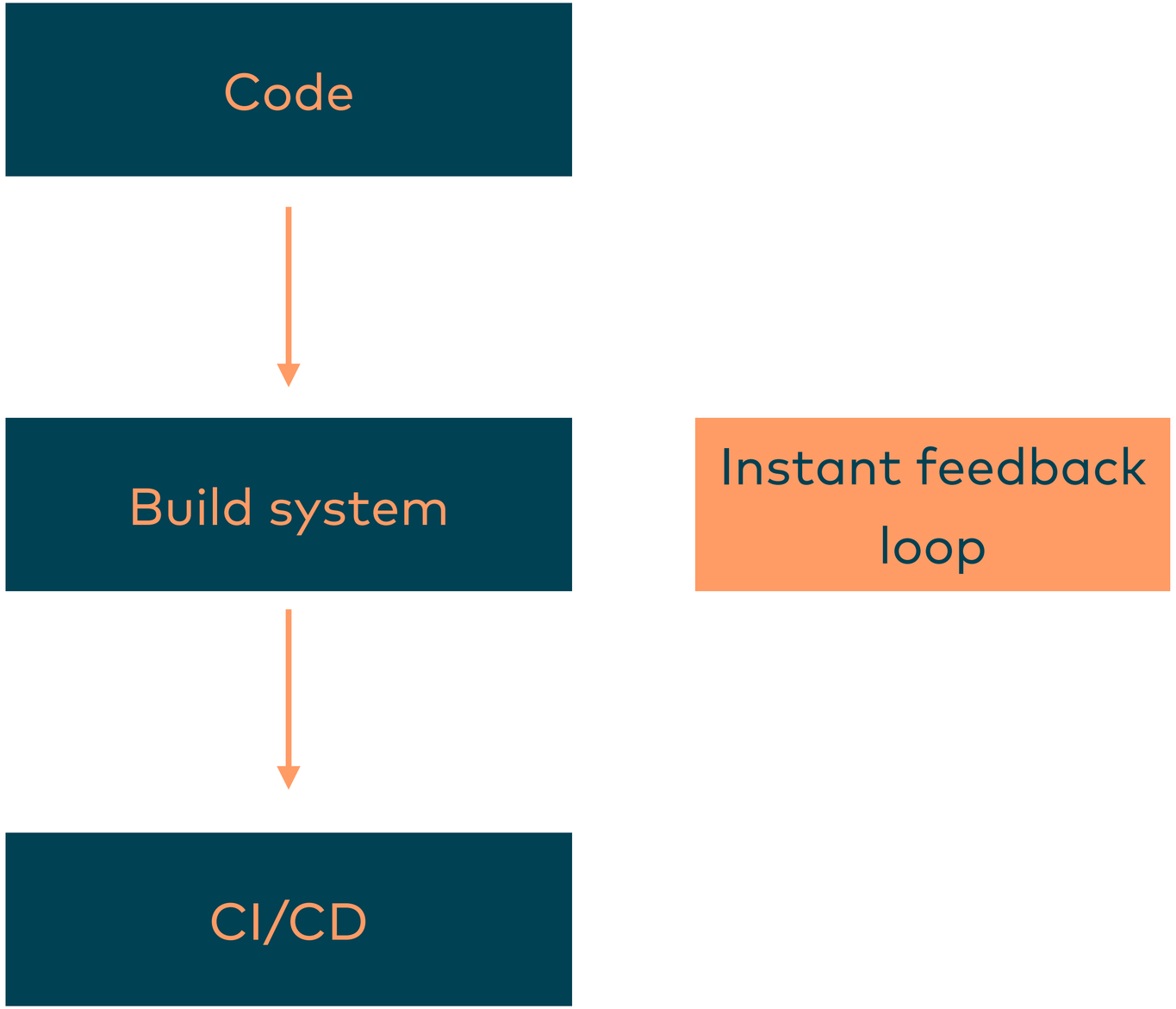
© cuelang.org

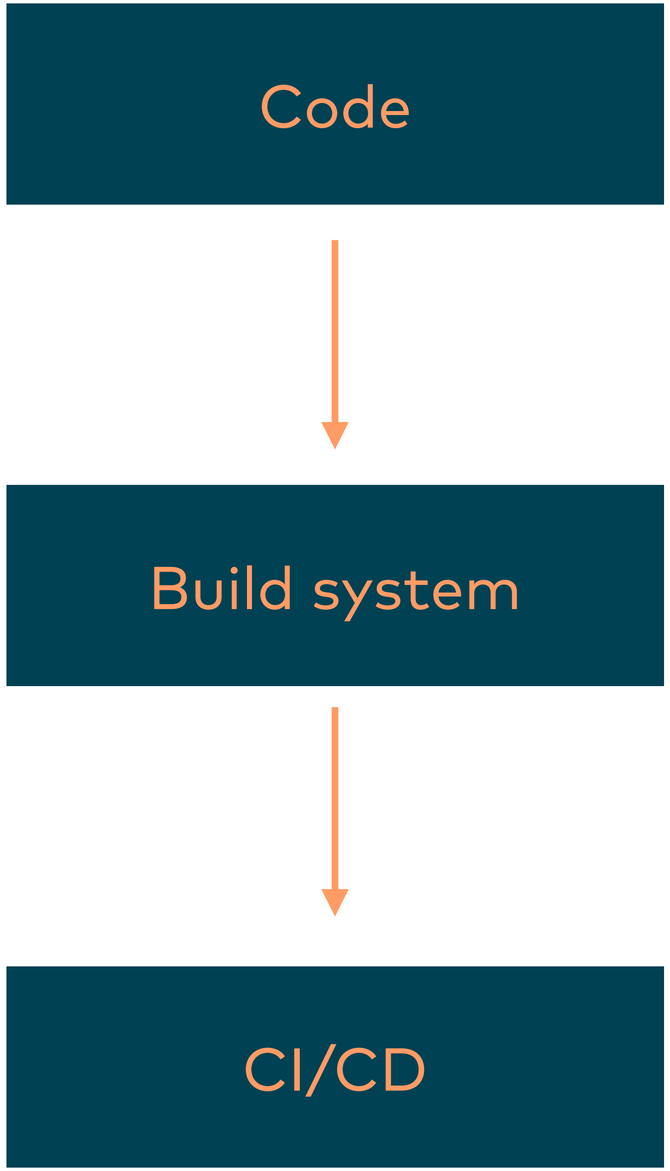
But why?!

Code


Code
Build system

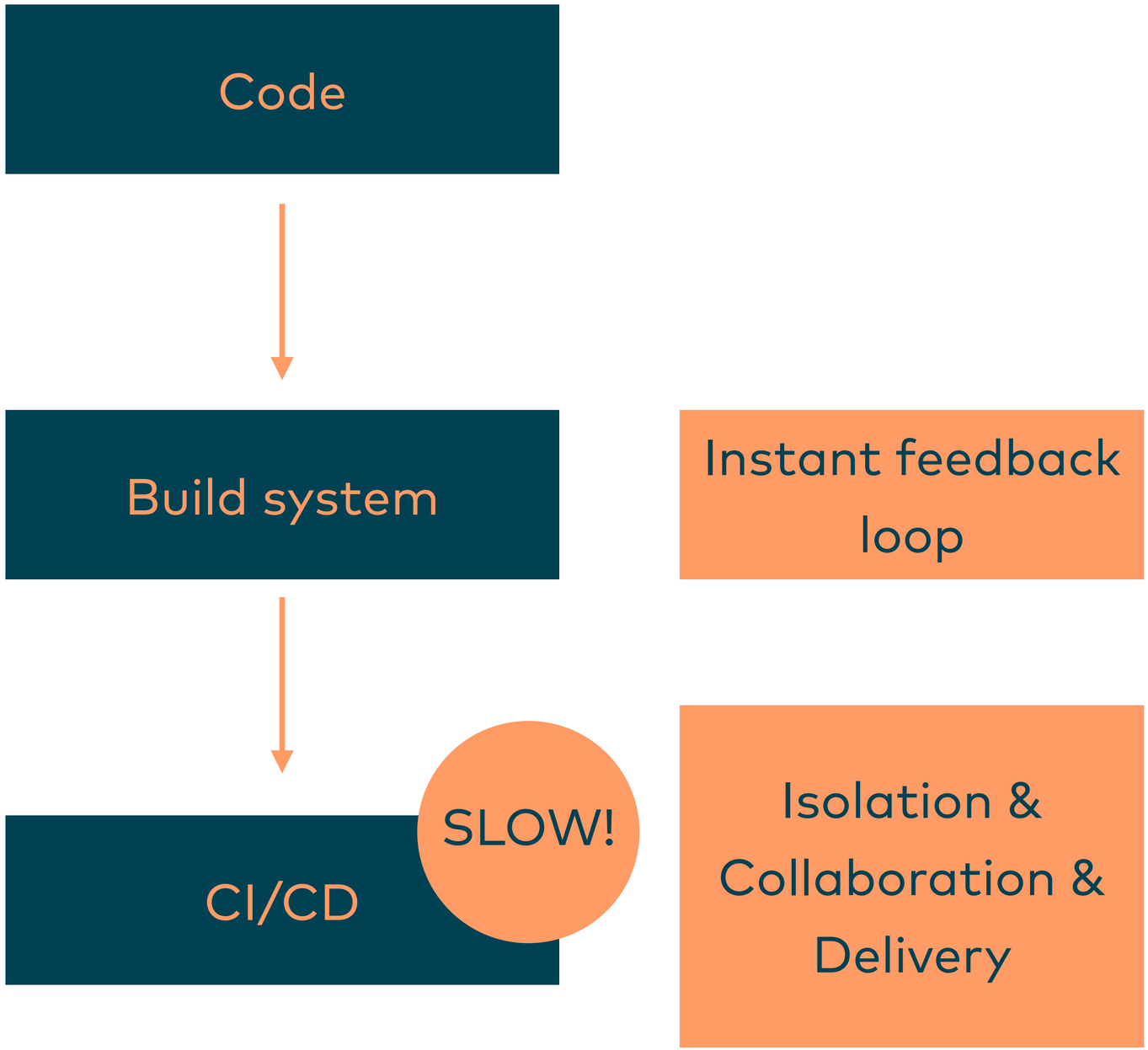

Code
Build system
CI/CD

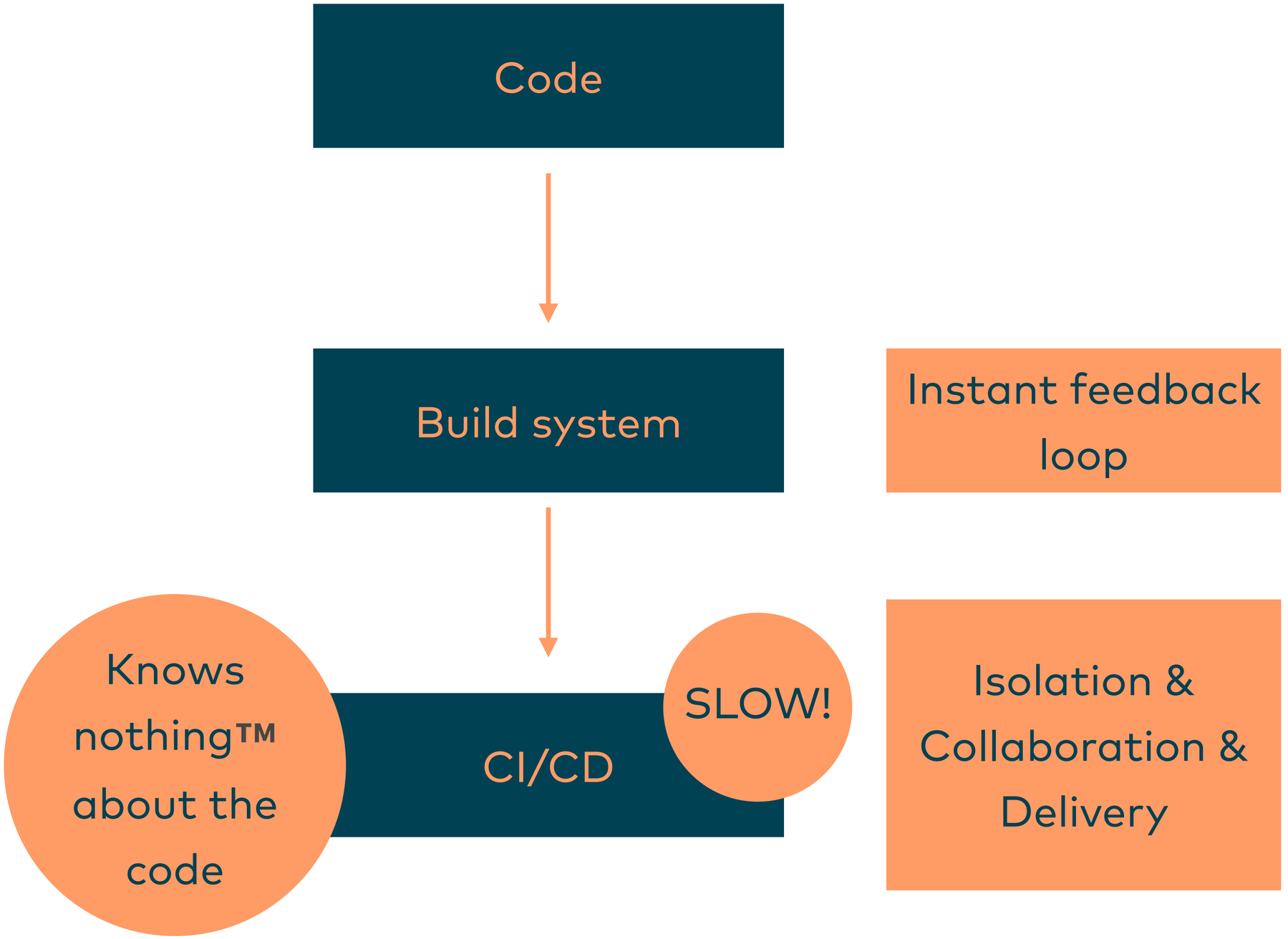

Code
Build system
CI/CD
Instant feedback loop

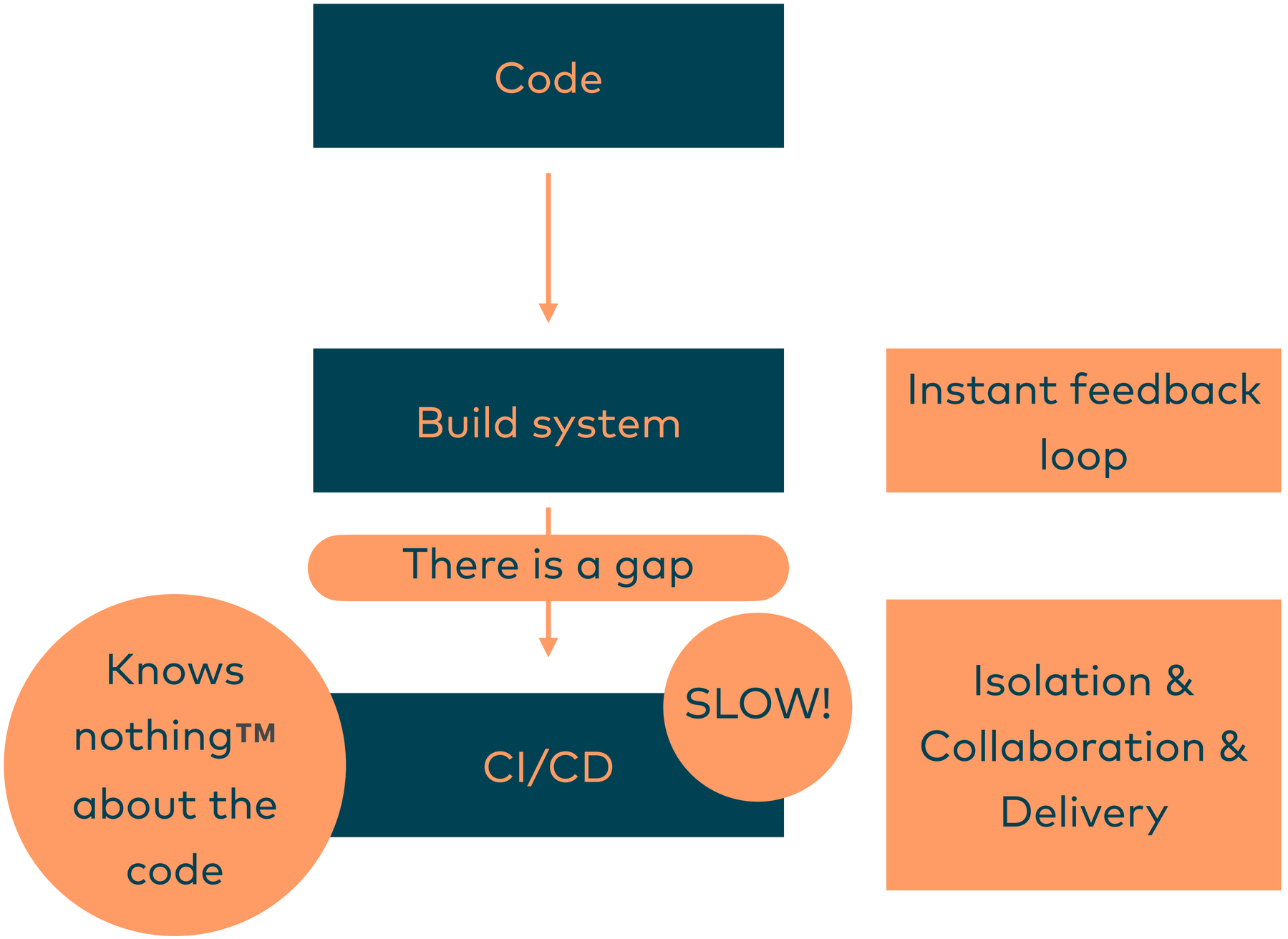


Instant feedback loop

Isolation & Collaboration & Delivery

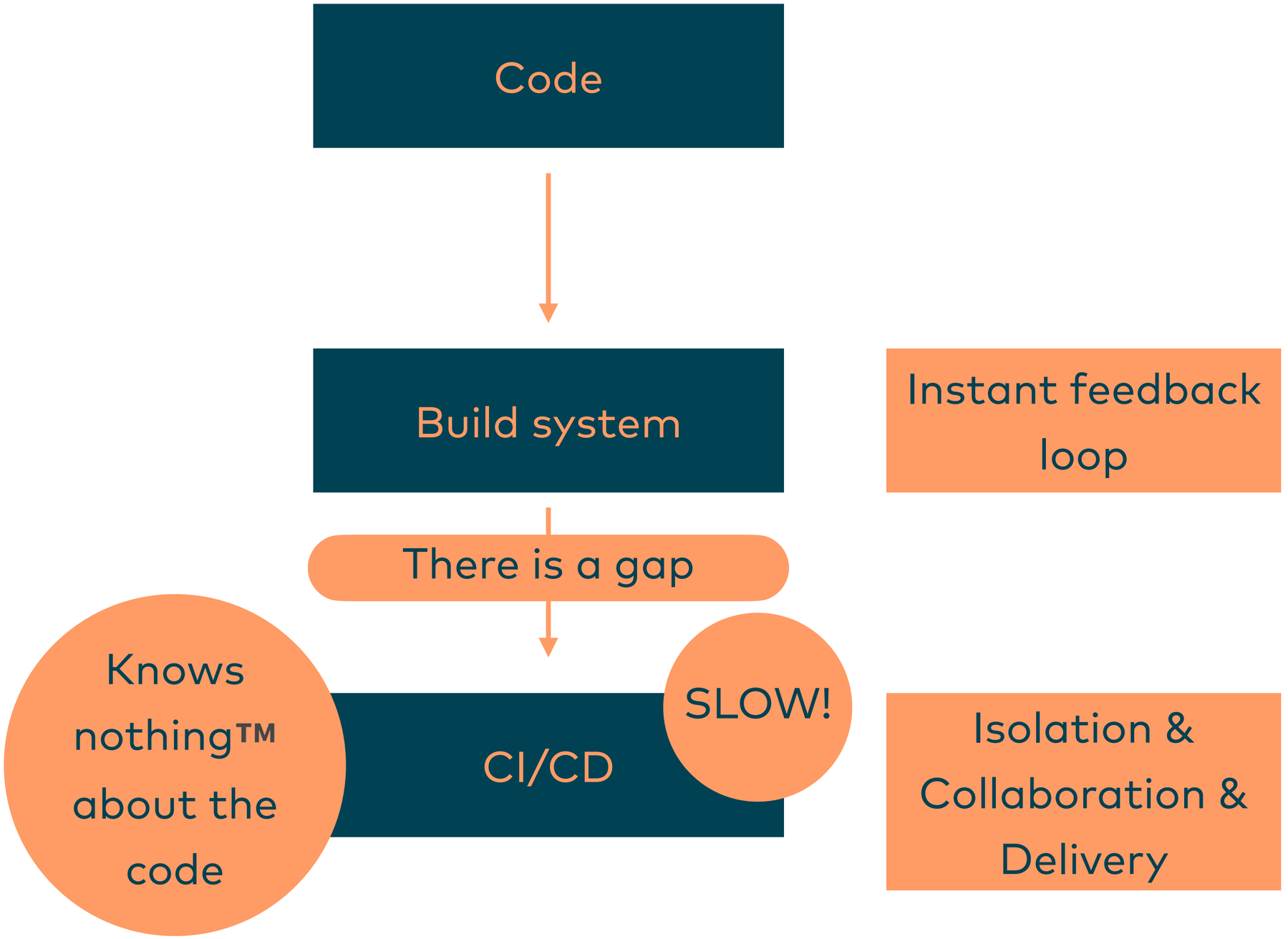

Code
Build system
CI/CD
SLOW!
Instant feedback loop
Isolation & Collaboration & Delivery

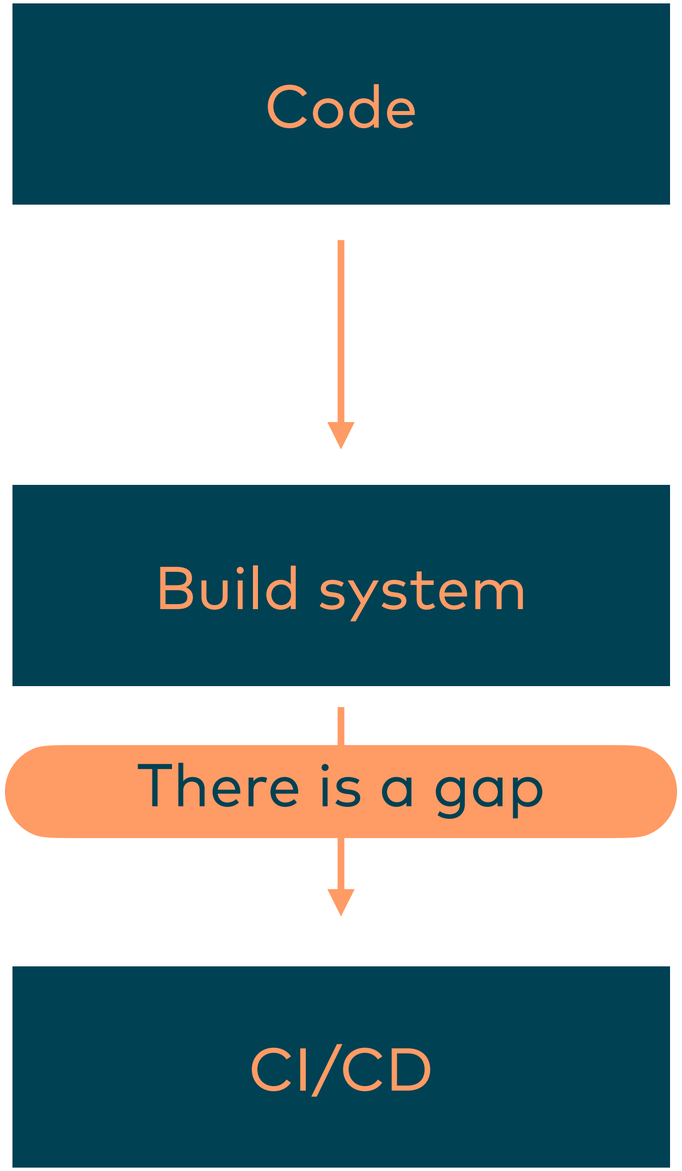

Code
Build system
CI/CD
Knows nothing™ about the code
SLOW!
Instant feedback loop
Isolation & Collaboration & Delivery

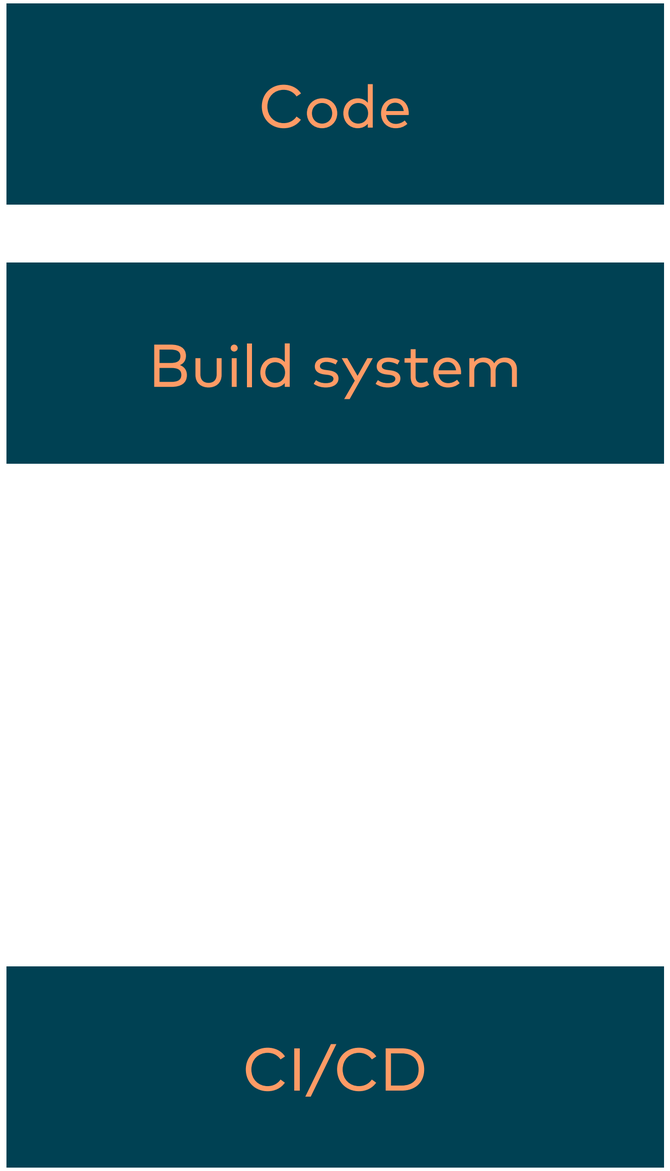

Code
Build system
Instant feedback loop
There is a gap
Knows nothing™ about the code
SLOW!
CI/CD
Isolation & Collaboration & Delivery

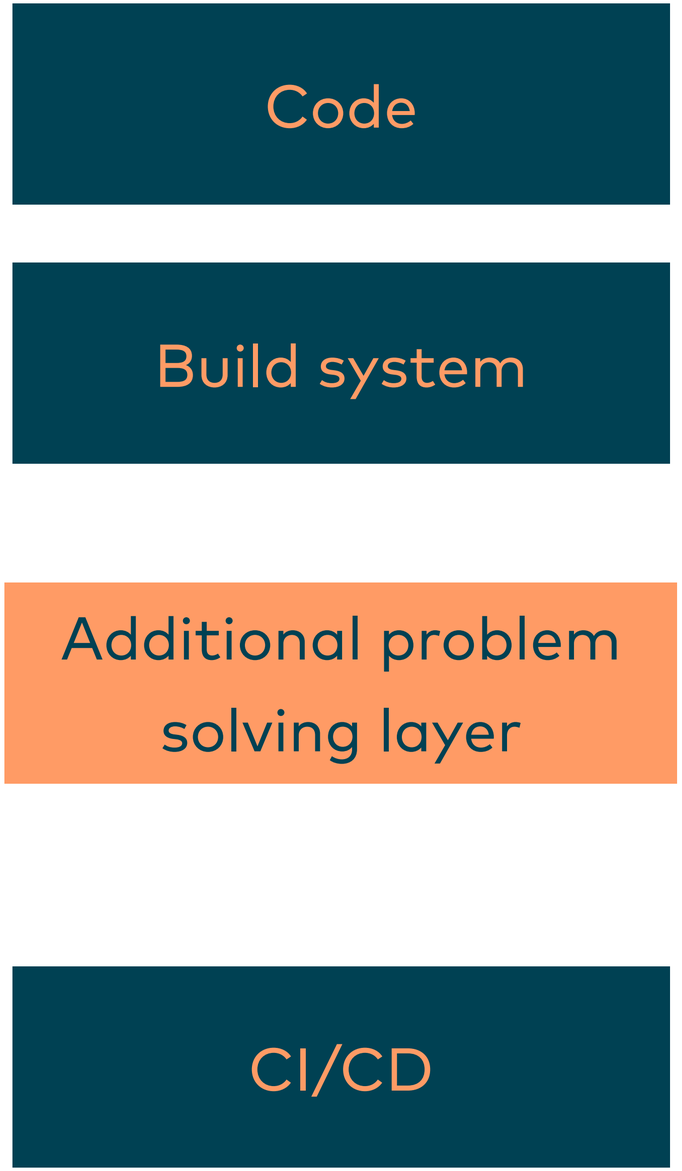
„Everything can be solved by an additional layer of indirection“

- *Unknown wise person*


Code
Build system
Instant feedback loop
There is a gap
Knows nothing™ about the code
SLOW!
CI/CD
Isolation & Collaboration & Delivery


Code
Build system
There is a gap
CI/CD


Code
Build system
CI/CD


Code
Build system
Additional problem solving layer
CI/CD

Code

Build system

Additional problem solving layer

Instant feedback & Isolation & Delivery

CI/CD

Imperative vs. declarative

- Gradle vs. Maven vs. Jenkinsfile vs. .gitlab-ci.yml
- Its not a binary decision, but a continuum
- Reduce mental load -> Shift complexity to different layers
- Don’t hide complexity, but establish clear boundaries

Why – Summary

- Save interface between Build and CI
- Local development with…
- … Instant feedback loop

We don’t want to replace either build or CI/CD systems, but bridge nicely between them while solving some problems of both systems along the way.

The origin story

From the people that brought you docker

Containers

It' about the developer experience

"Engine lead“ Docker project

BuildKit

Low-Level Build definition format

LLB

„At the core of BuildKit is a Low-Level Build definition format. <...>

<LLB> defines a content-addressable **dependency graph** that can be used to put together very **complex build definitions**.

It also supports features not exposed in Dockerfiles, like direct data mounting and nested invocation. <...>

Everything about execution and caching of your builds is defined in LLB“

cuelang.org

The data validation & configuration language

cuelang.org

The data validation & configuration language

- Built upon ~15 years of experience with Google GCL

cuelang.org

The data validation & configuration language

- Built upon ~15 years of experience with Google GCL
- **Combine constraints** from different sources to produce a **deterministic output**

cuelang.org

The data validation & configuration language

- Built upon ~15 years of experience with Google GCL
- **Combine constraints** from different sources to produce a **deterministic output**
- Bonus: Comparing schemas for backwards compatibility

cuelang.org

The data validation & configuration language

- Built upon ~15 years of experience with Google GCL
- **Combine constraints** from different sources to produce a **deterministic output**
- Bonus: Comparing schemas for backwards compatibility
- Limited scripting: explicitly **constrained** -> converges to a **valid state in finite time**

cuelang.org

The data validation & configuration language

- Built upon ~15 years of experience with Google GCL
- **Combine constraints** from different sources to produce a **deterministic output**
- Bonus: Comparing schemas for backwards compatibility
- Limited scripting: explicitly **constrained** -> converges to a **valid state in finite time**

https://cuelang.org/docs/about/#history

Origins – Summary

Origins – Summary

- People with right™ mindset

Origins – Summary

- People with right™ mindset
- Mature foundational technologies

Origins – Summary

- People with right™ mindset
- Mature foundational technologies
- Cuelang as a configuration language

Origins – Summary

- People with right™ mindset
- Mature foundational technologies
- Cuelang as a configuration language
- Everything gets better if you throw container technology at it and introduce a new – perfectly fitting – „programming“ language😉

Success of a technology is determined by its accessibility

The building blocks

Concepts

dagger.io

cuelang.org

Concepts

- Basic **cuelang** concepts

dagger.io

cuelang.org

Concepts

- Basic **cuelang** concepts
- **dagger.io** primitives, structure and lifecycle

dagger.io

cuelang.org

Concepts

- Basic **cuelang** concepts
- **dagger.io** primitives, structure and lifecycle
- How they interact (with docker)

dagger.io

cuelang.org

cuelang

Important concepts for dagger.io

Schema is data is ...

cuelang

Important concepts for dagger.io

Schema is data is ...

- Nodes (with fields) & constraints {a: int}
- Operators and Expressions (>=)
- Definitions (#)
- Unifications and Disjunctions (& |)
- Conditionals (if a > 5)
- Loops
- Templates

Go and play (later😉): https://cuelang.org/play/

cuelang

Important concepts for dagger.io

Schema is data is ...

cuelang

Important concepts for dagger.io

Schema is data is ...

```
innoqEvent: {
    name: string
    attendees: > 500 | *2000
    fun: "A lot!!!"
} & {
    name: "Technology Day"
    attendees: 600
}
```

cuelang

Important concepts for dagger.io

Schema is data is ...

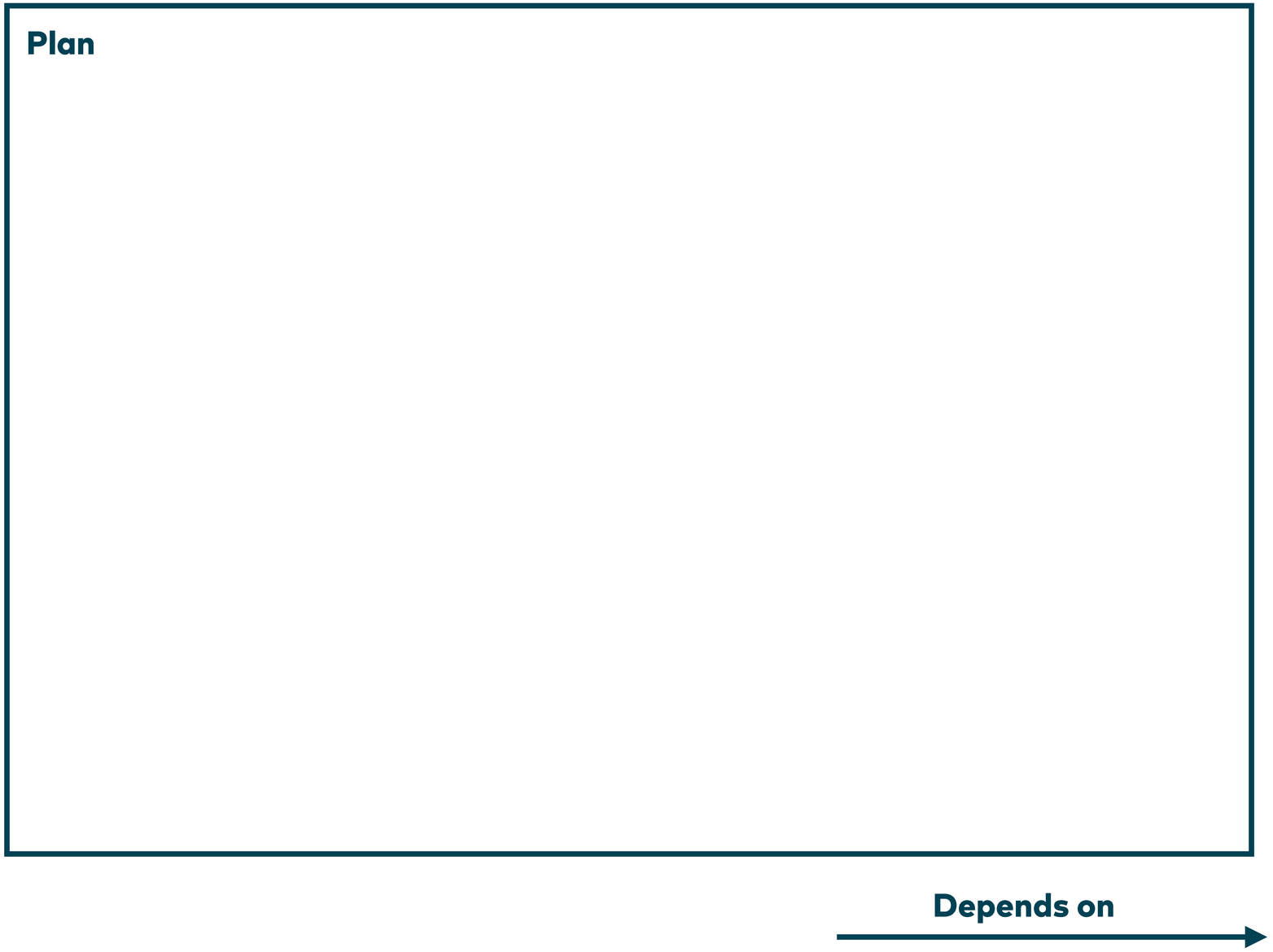
```
innoqEvent: {
    name: string
    attendees: > 500 | *2000
    fun: "A lot!!!"
} & {
    name: "Technology Day"
    attendees: 600
}
```

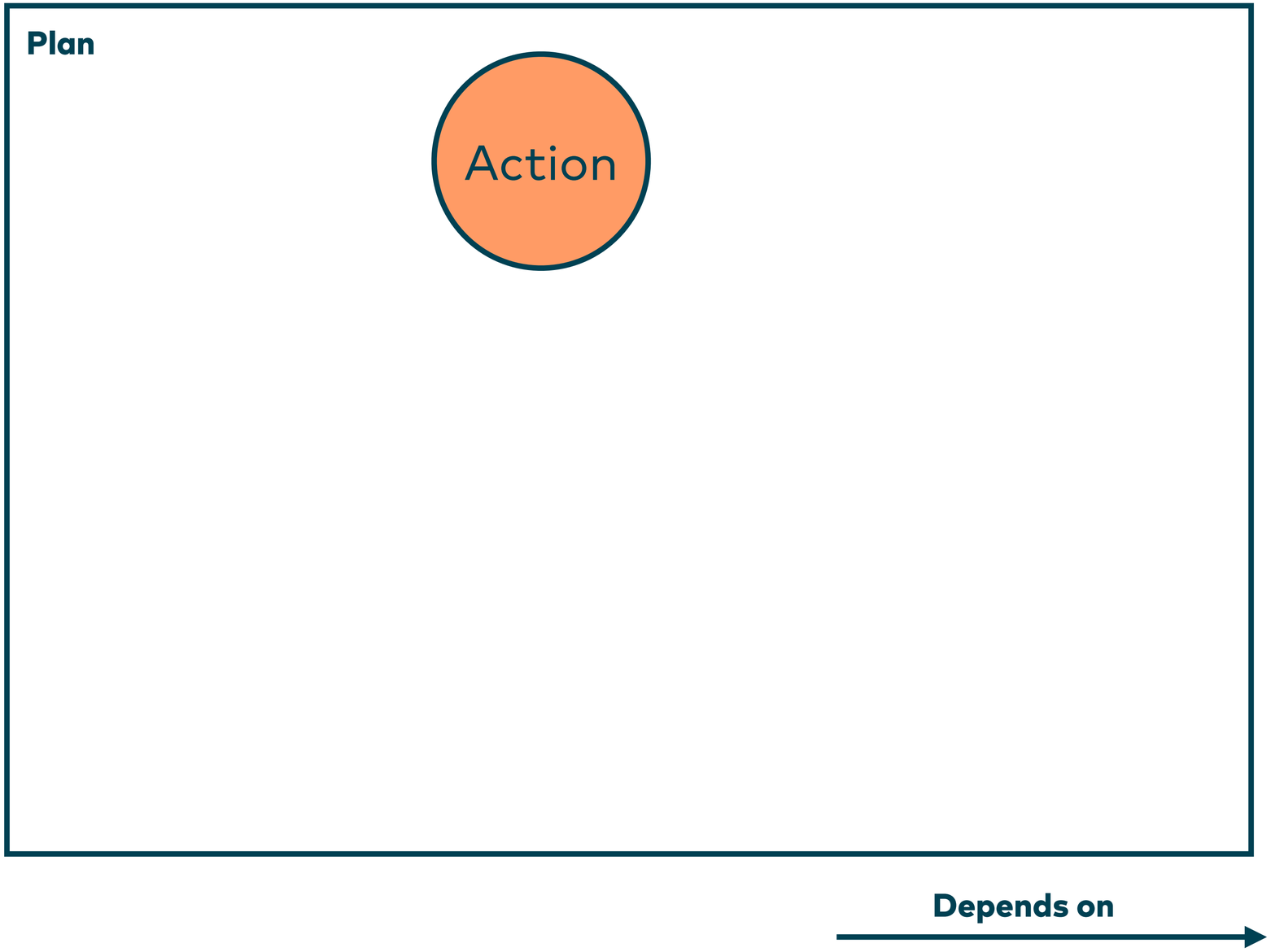
```
{
    "innoqEvent": {
        "name": "Technology Day",
        "attendees": 600,
        "fun": "A lot!!!"
    }
```

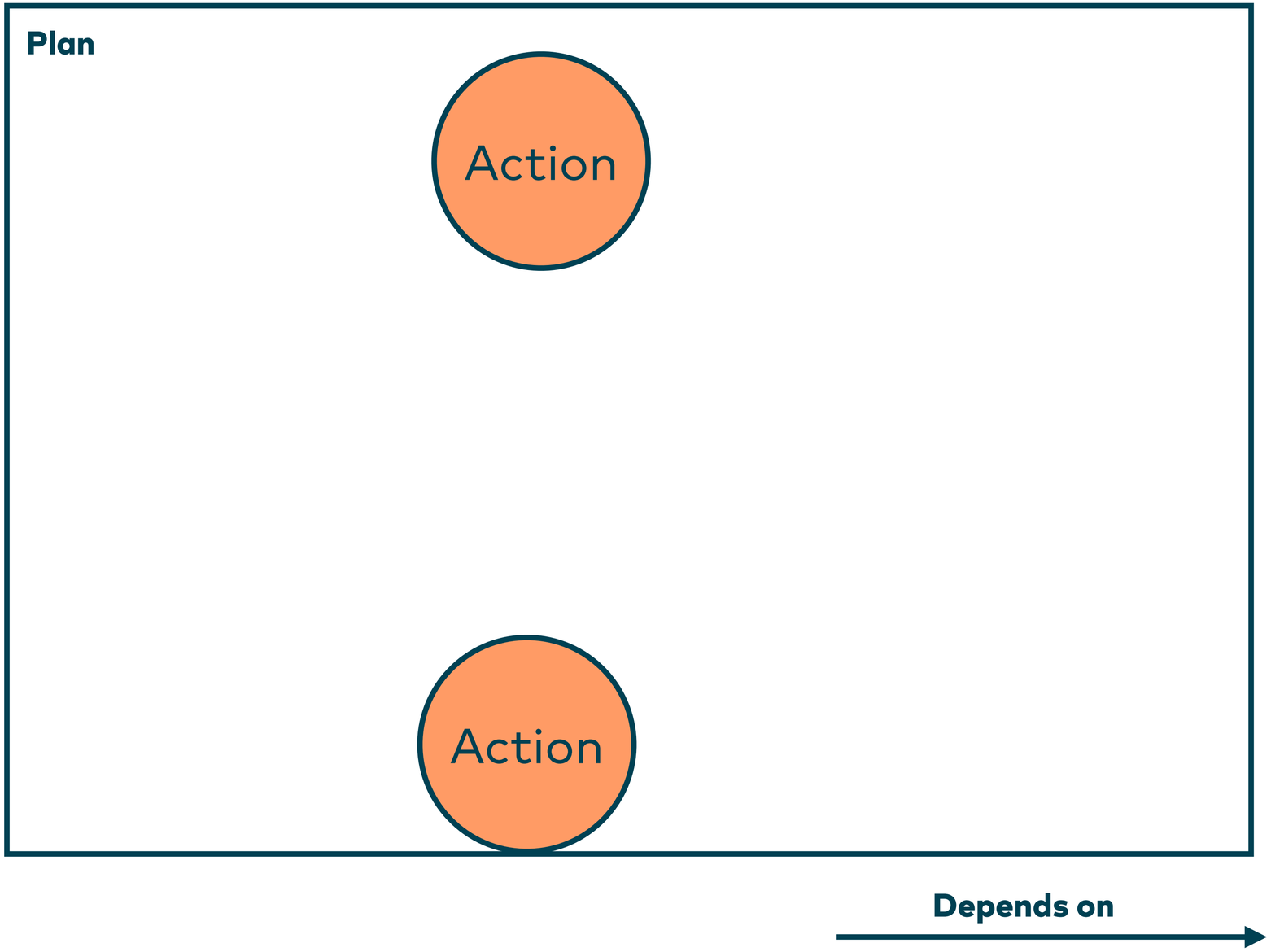
Using a directed acyclic graph to our advantage

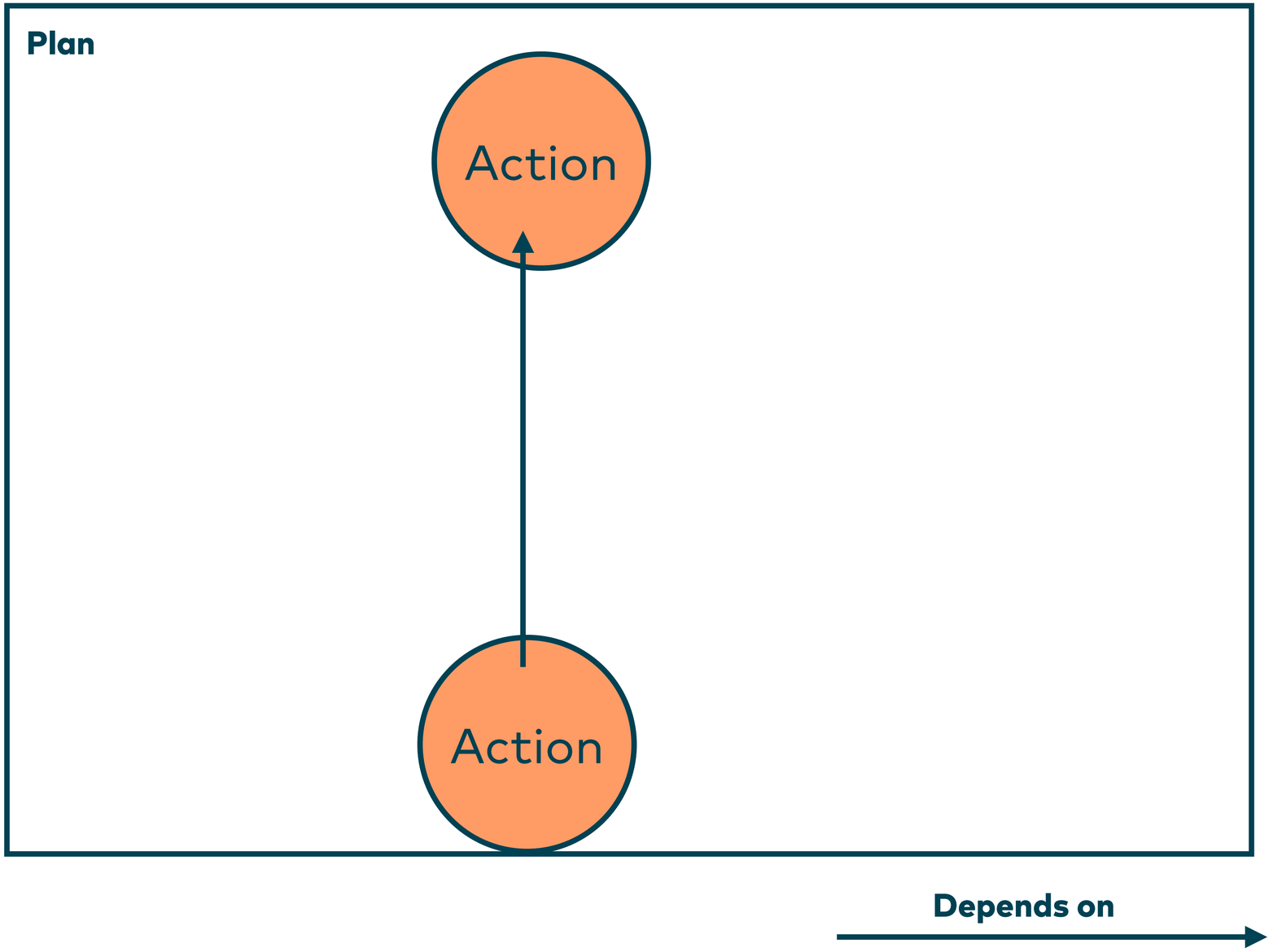
Putting a plan into actions

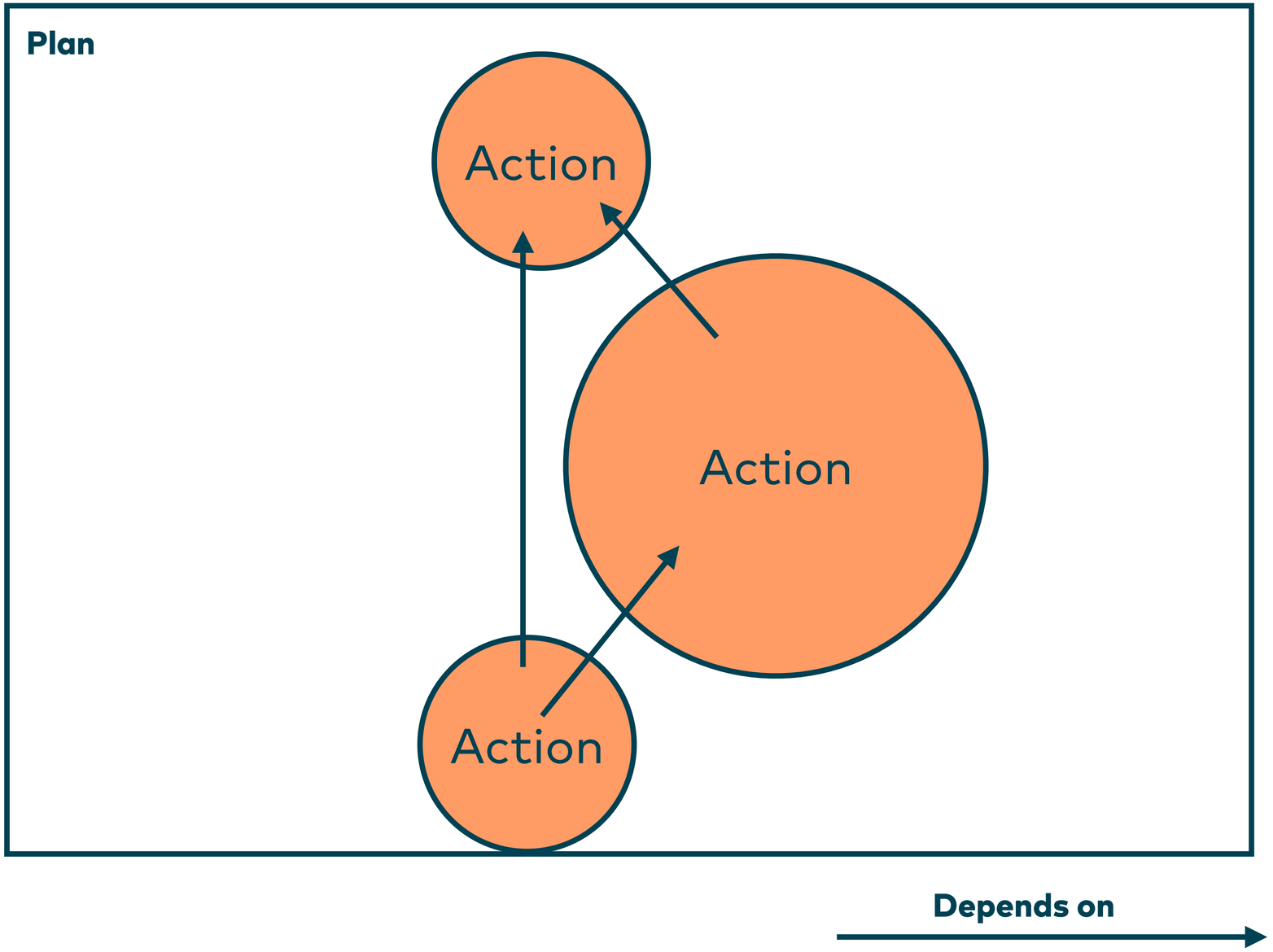

Depends on

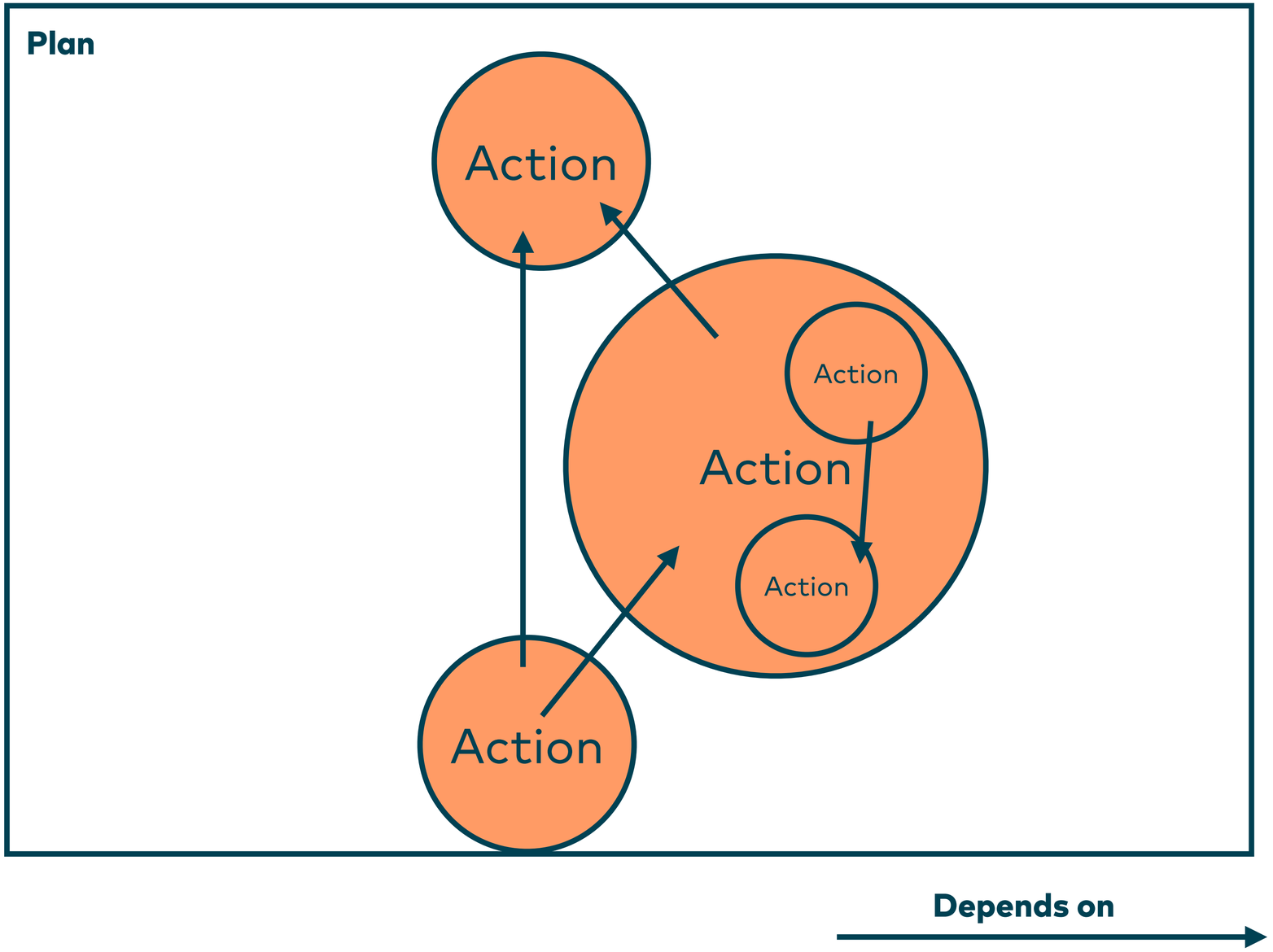

Plan
Depends on


Plan
Action
Depends on

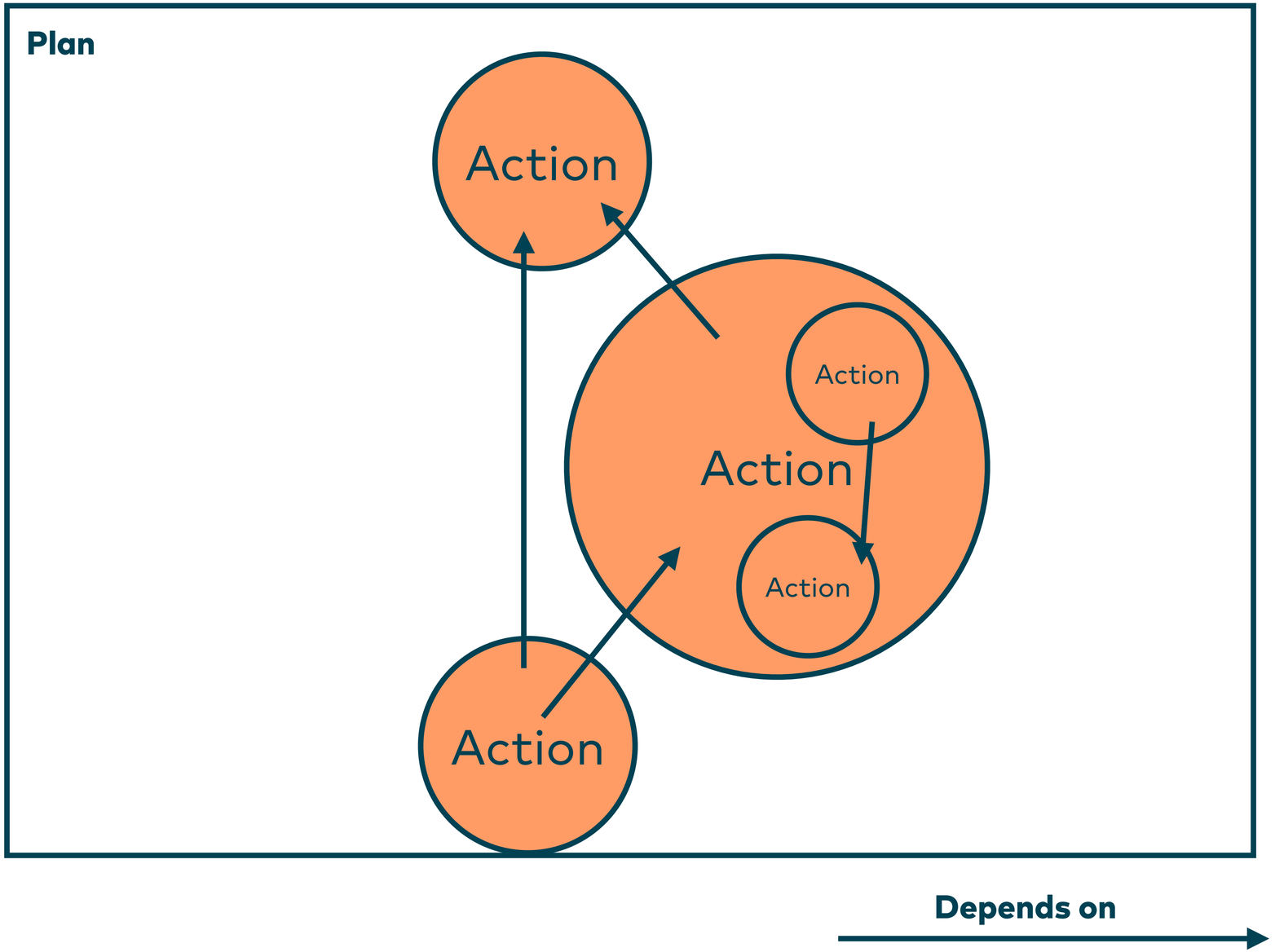

Plan
Action
Action
Depends on

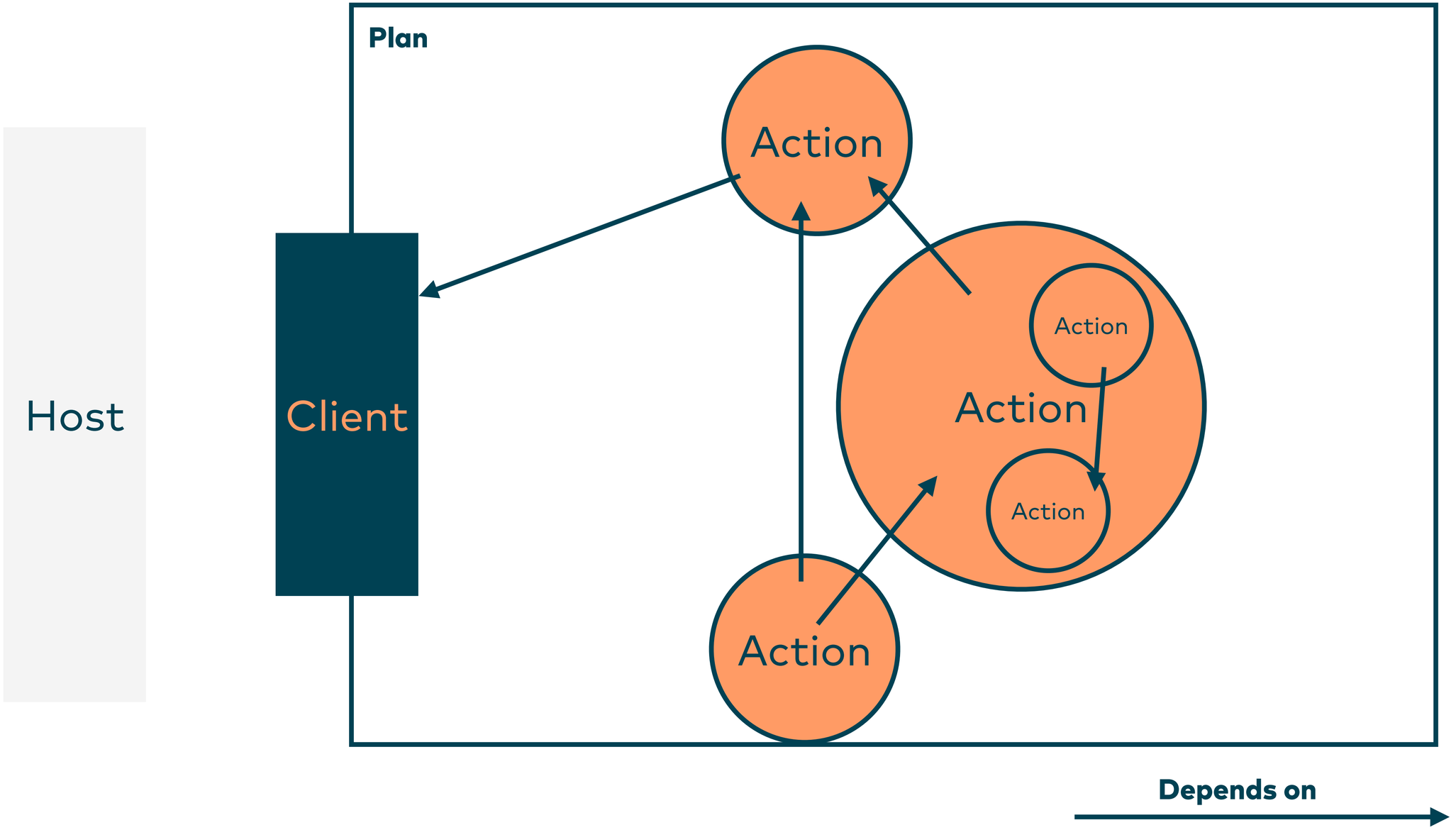

Plan
Action
Action
Depends on

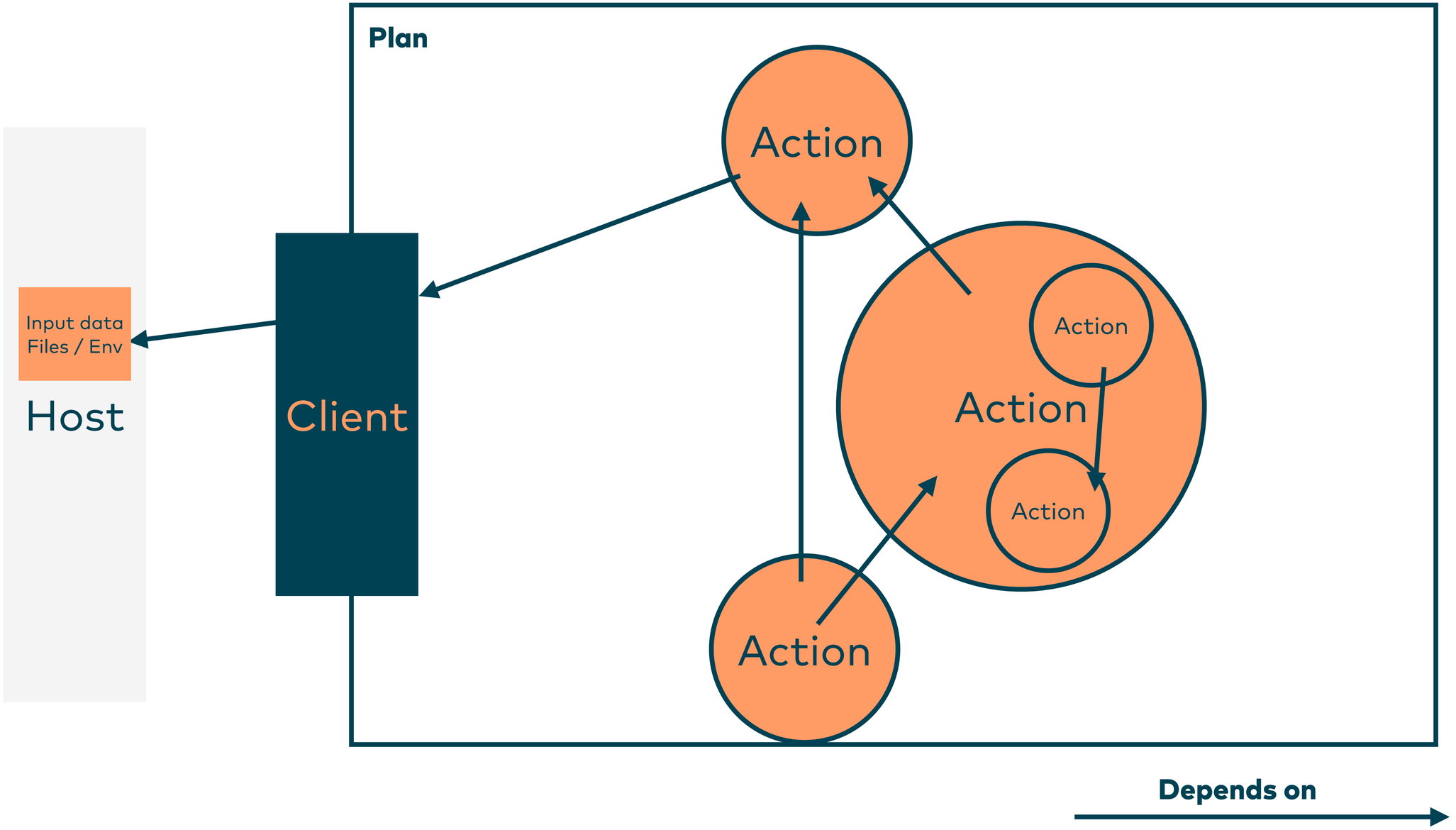

Plan
Action
Action
Action
Depends on


Plan
Action
Action
Action
Action
Action
Depends on

Host




Host
Plan
Client
Action
Action
Action
Action
Action
Depends on


Plan
Action
Action
Action
Action
Action
Input data
Files / Env
Host
Client
Depends on

Concepts – Summary

Concepts – Summary

- Cuelang: schema == data and order doesn't matter

Concepts – Summary

- Cuelang: schema == data and order doesn't matter
- dagger.io: Plan with composite and nested actions

Concepts – Summary

- Cuelang: schema == data and order doesn't matter
- dagger.io: Plan with composite and nested actions
- Docker engine: caching „for free“

Concepts – Summary

- Cuelang: schema == data and order doesn't matter
- dagger.io: Plan with composite and nested actions
- Docker engine: caching „for free“

Example - Lets blog!

- Build static site with goHugo
- Optimize images before deployment
- Deploy website via rsync

dagger.io

cuelang.org

Future

- cuelang not mandatory
- SDKs
  - Golang
  - Python
  - node.js

dagger.io

cuelang.org

Opinions

- Good mixture of people, mindset, concepts and foundational technology
- Boundary between imperative and declarative layers is good
- Nothing revolutionary, but an evolution and amalgamation of existing technologies

Feedback?

- Used dagger.io?
- Used cuelang.org?
- Can recommend similar / alternative tools?
- Declarative vs. imperative vs. mix of both?

- **Thanks for your attention!** ❤️

Feedback? Contact!

Fabian Kretzer

fabian.kretzer@innoq.com

innoq.social/@fabian

INNOQ

www.innoq.com

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin

Ludwigstr. 180E
63067 Offenbach

Kreuzstr. 16
80331 München

Hermannstrasse 13
20095 Hamburg

Erftstr. 15-17
50672 Köln

Königstorgraben 11
90402 Nürnberg