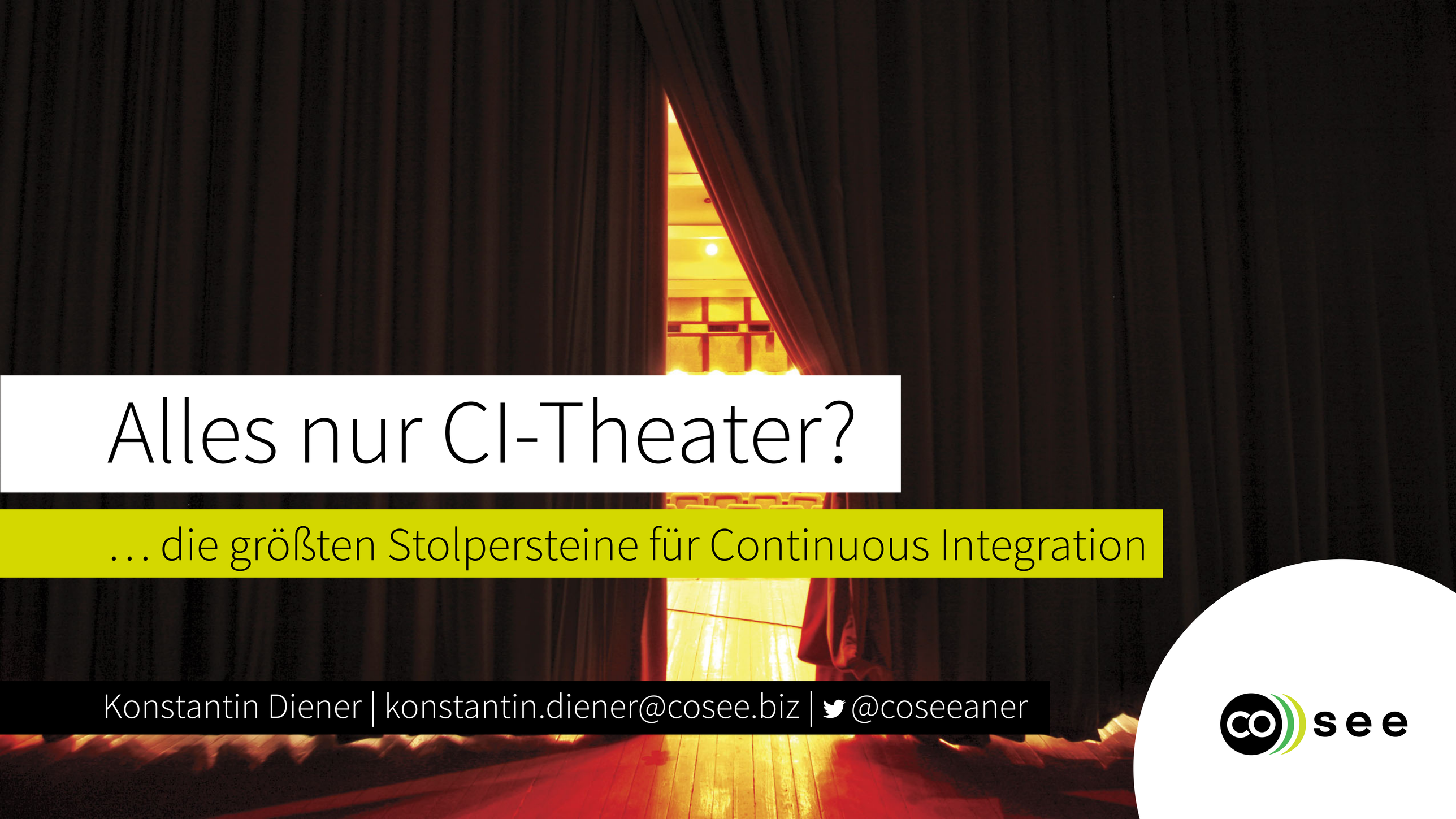


Alles nur CI-Theater?

… die größten Stolpersteine für Continuous Integration

Konstantin Diener | konstantin.diener@cosee.biz | @coseeaner

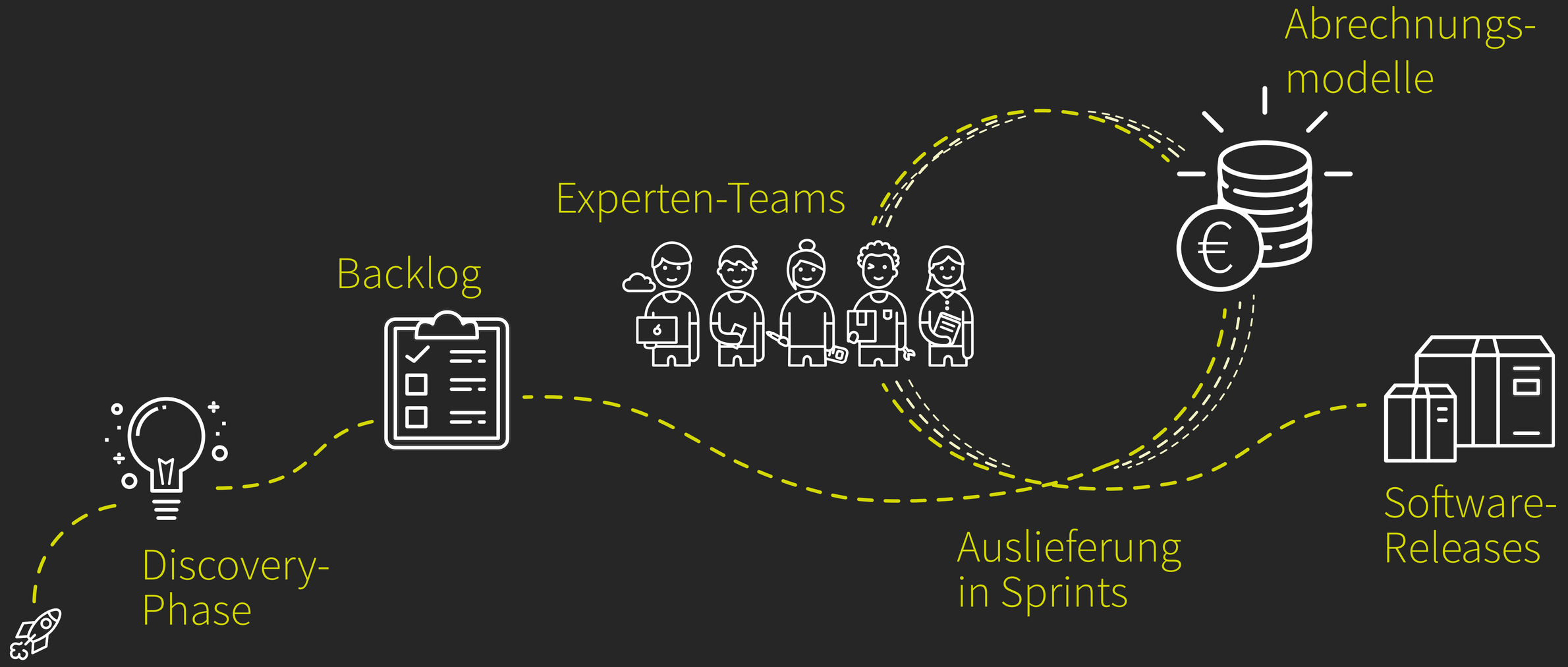
cosee

Konstantin Diener

CTO und Gründer von cosee

Produktentwicklung starten

Produktentwicklung skalieren


Abrechnungs-
modelle
Experten-Teams
Backlog
Discovery-
Phase
Auslieferung
in Sprints
Software-
Releases

Was ist eigentlich

Continuous Integration?

„Continuous Integration is a software development practice where **members of a team integrate their work frequently**, usually each person integrates **at least daily** - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to **significantly reduced integration problems** and allows a team to **develop cohesive software more rapidly**. ...“

https://martinfowler.com/articles/continuousIntegration.html

11 Praktiken

Maintain a Single Source Repository

Automate the Build

Make Your Build Self-Testing

Everyone Commits To the Mainline Every Day

Every Commit Should Build the Mainline on an Integration Machine

Fix Broken Builds Immediately

Keep the Build Fast

Test in a Clone of the Production Environment

Make it Easy for Anyone to Get the Latest Executable

Everyone can see what's happening

Automate Deployment


CI-Theater

“

„CI theatre” describes the **illusion of practising continuous integration** (CI) while not really practising it.

Suzie Prince, https://www.gocd.org/2017/05/16/its-not-CI-its-CI-theatre.html

Es ist die Übertragung von

Cargo Cult

auf Continuous Integration.

Top 3

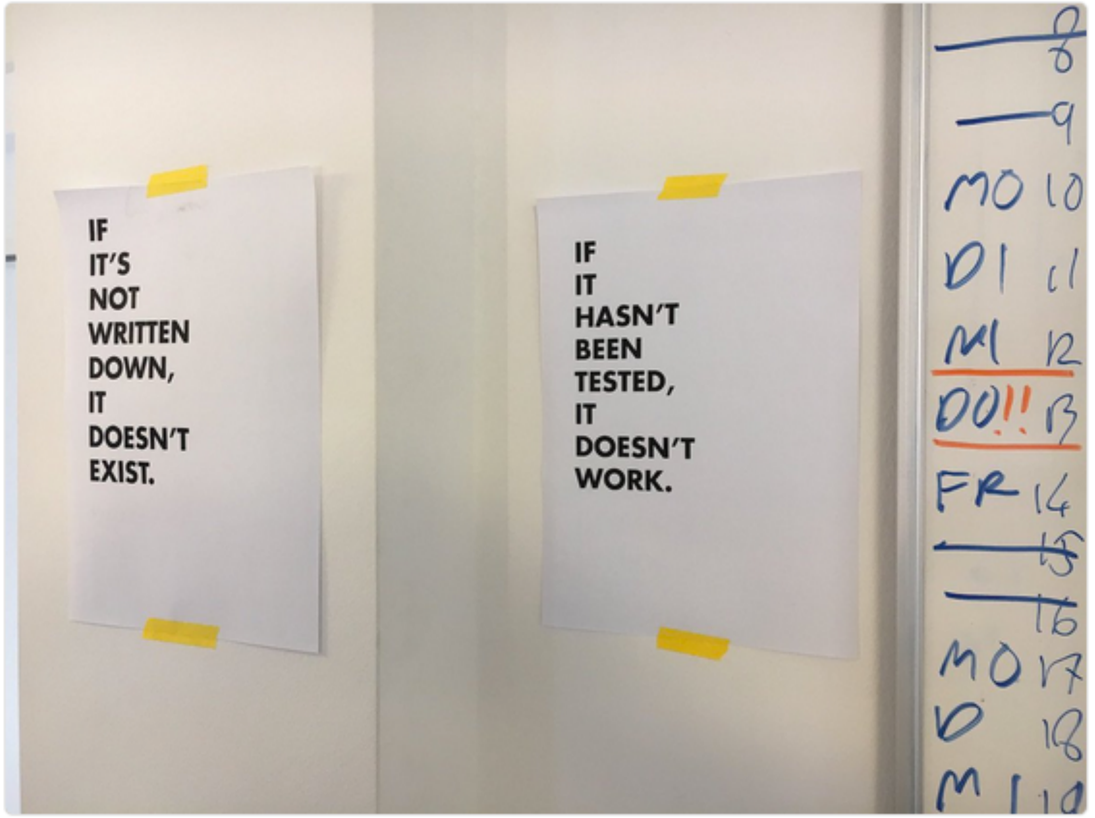
3

Make Your Build Self-Testing

espylaub
@espylaub

Folgen

Constant reminders.



01:43 - 14. Okt. 2016

331 Retweets **451** „Gefällt mir"-Angaben

„If you've skipped unit tests because you plan to refactor the code soon, you might not understand refactoring (or unit tests).“

@DocOnDev

```
mvn install -DskipTests
```

```
--no-verify
```

2

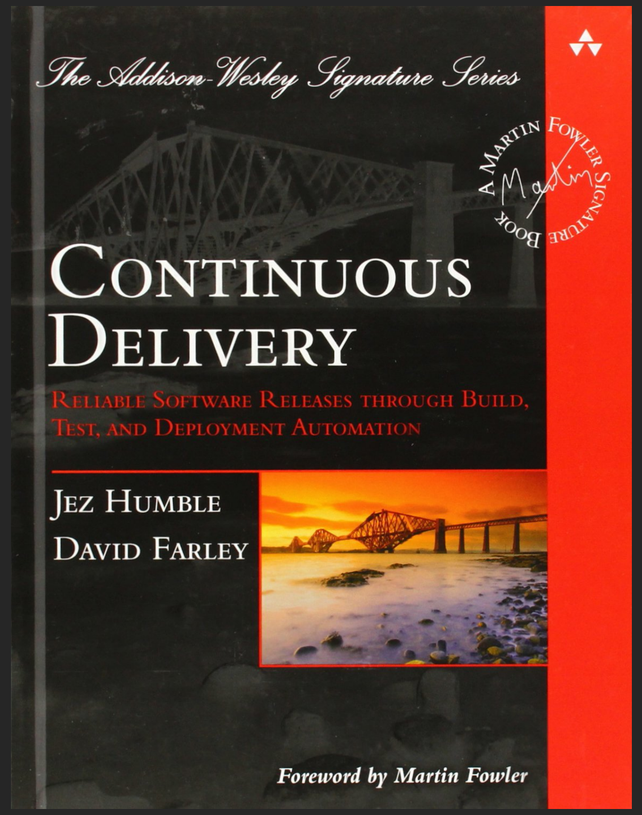
Fix Broken Builds Immediately

A phrase I remember Kent Beck using was **"nobody has a higher priority task than fixing the build"**. This doesn't mean that everyone on the team has to stop what they are doing in order to fix the build, usually it only needs a couple of people to get things working again. It does mean a conscious prioritization of a build fix as an urgent, high priority task.

Martin Fowler

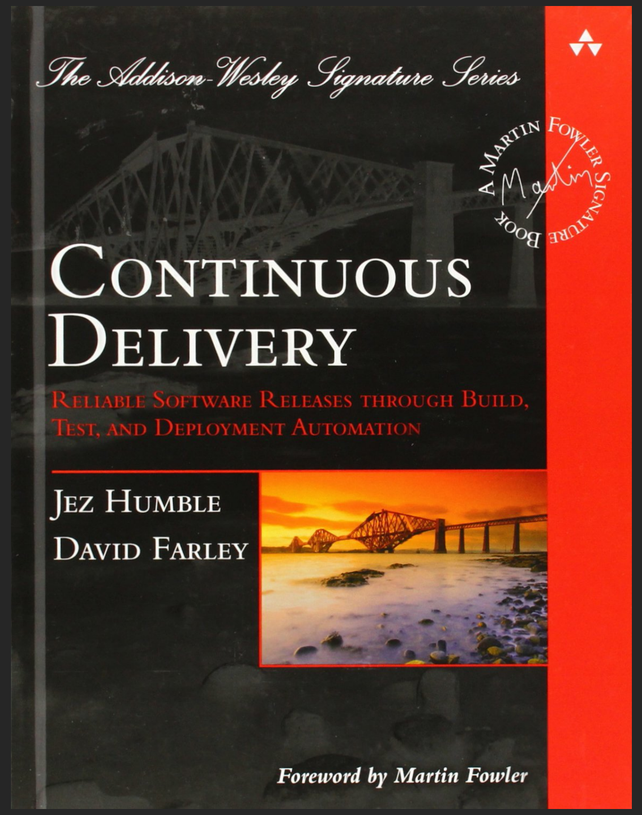
1

Everyone Commits To the Mainline Every Day



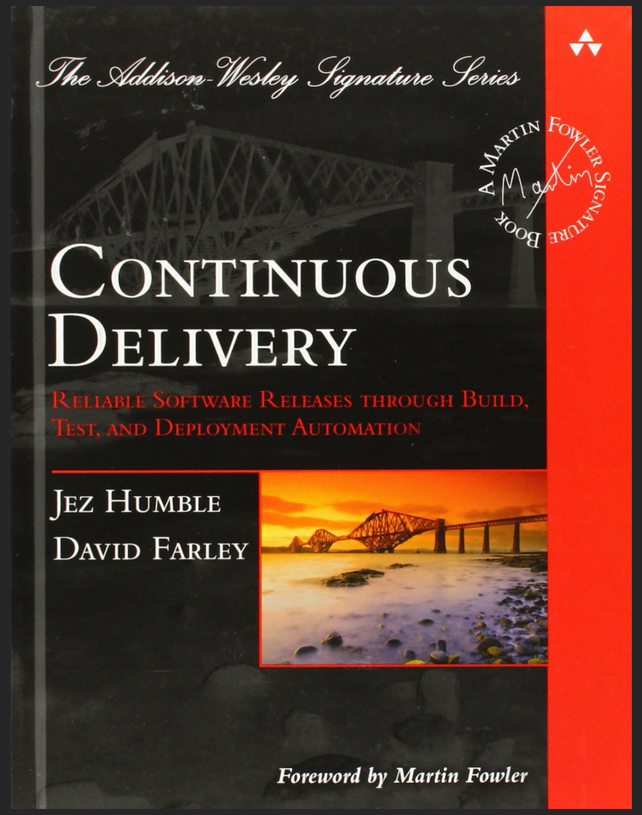
Continuous Delivery

Jez Humble, David Farley



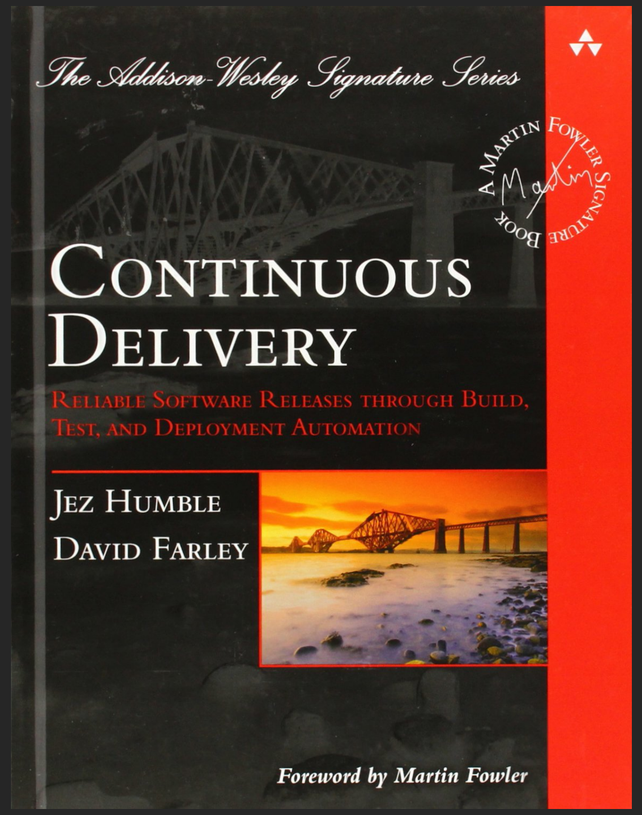
„. . . it has been our experience that **poor version control practices** are one of the **most common barriers to fast, low-risk releases.**“

Continuous Delivery



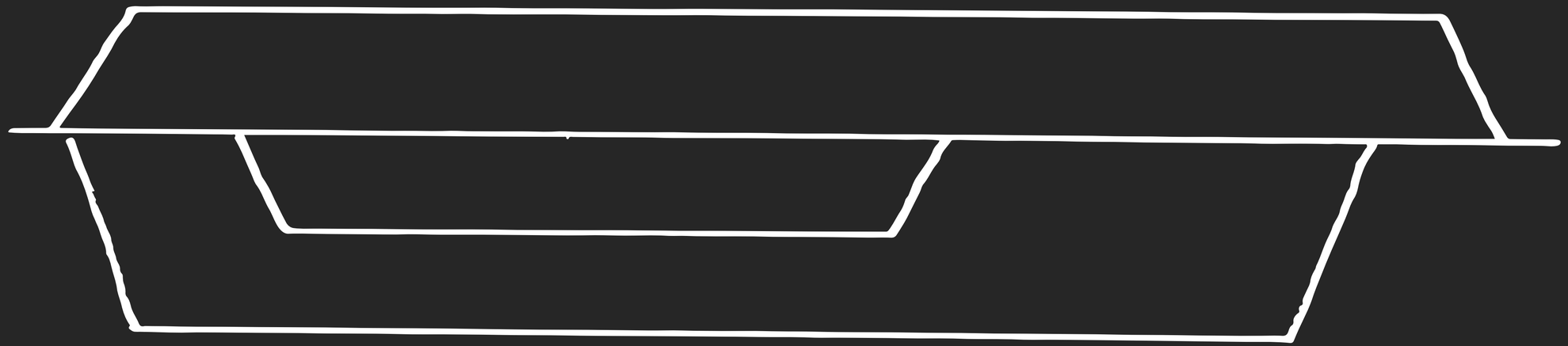
- **Develop on Mainline**
- **Branch for Release**
- **Branch by Feature**
- Branch by Team

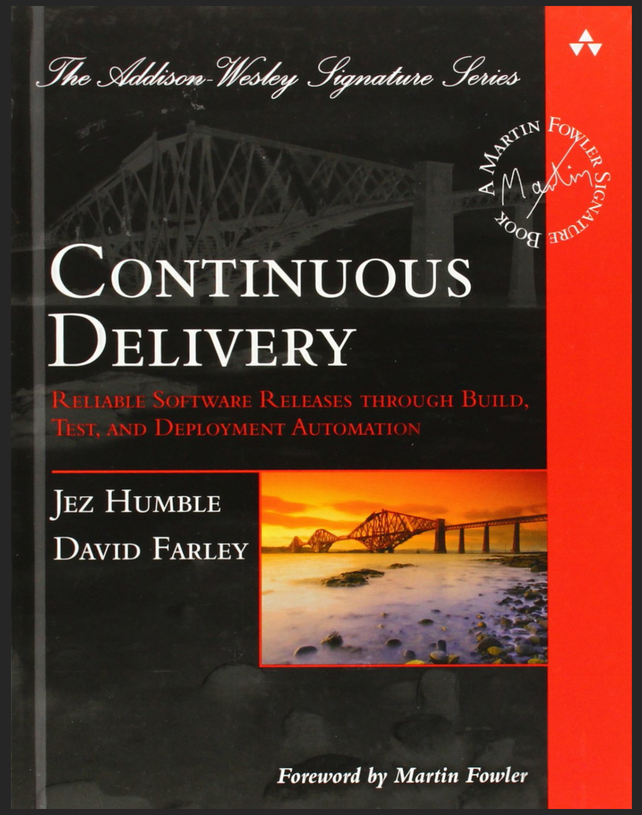
Continuous Delivery



- Develop on Mainline
- Branch for Release
- **Branch by Feature**

Continuous Delivery





„It is worth emphasizing that

branching by feature is really the antithesis of continuous integration,

…

Continuous Delivery

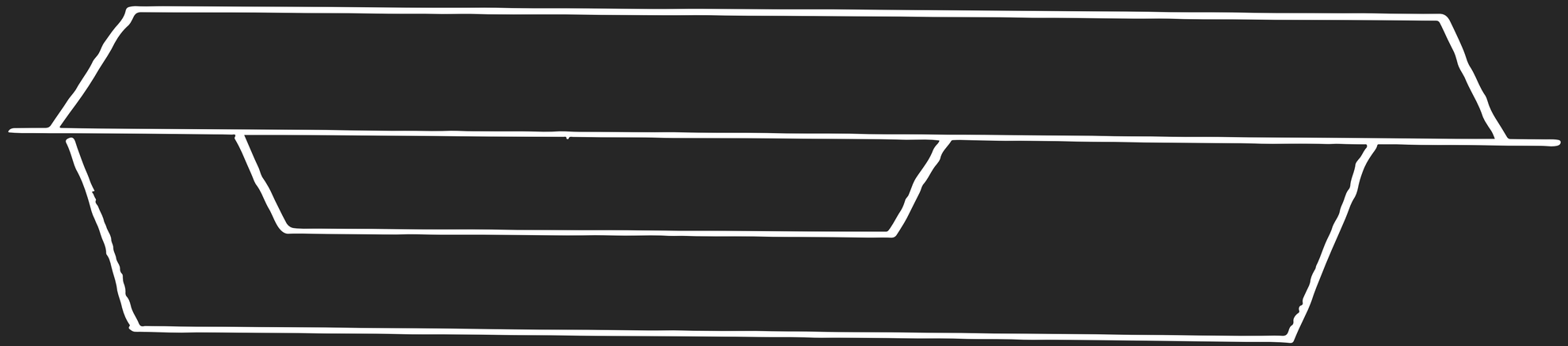
“

Branching is not the problem, **merging** is the problem.

Jez Humble

Jeder von uns kennt vermutlich die

Merge Hell.

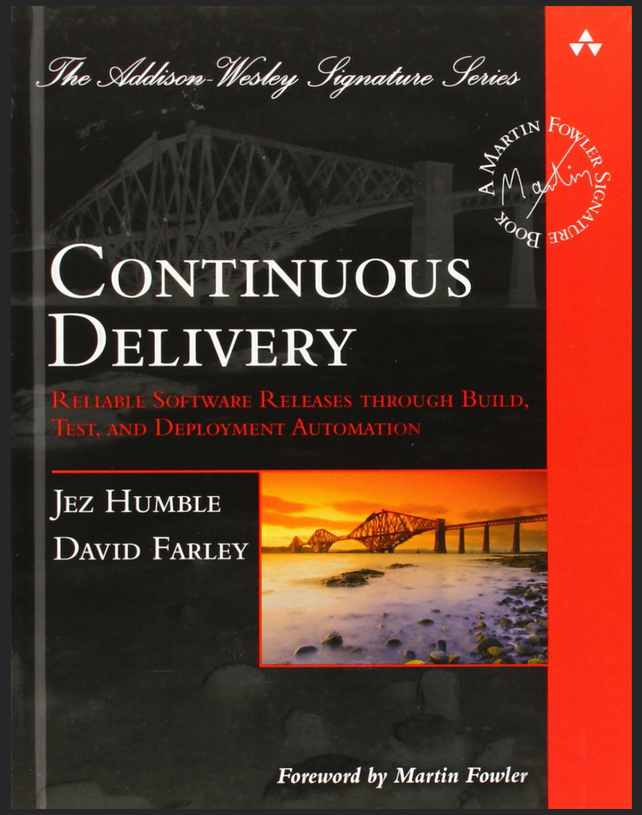


Auf Feature Branches zu arbeiten, bezeichnen die Autoren als

Continuous Isolation.

DON’T DO FEATURE BRANCHES!

Was denn sonst? 🤔



- **Develop on Mainline**
- **Branch for Release**
- **Branch by Feature**

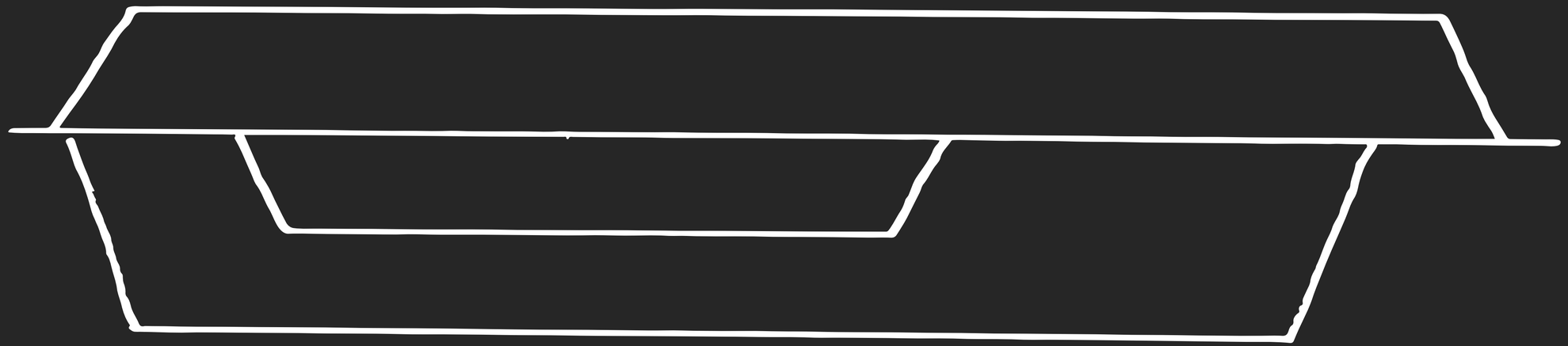
Continuous Delivery

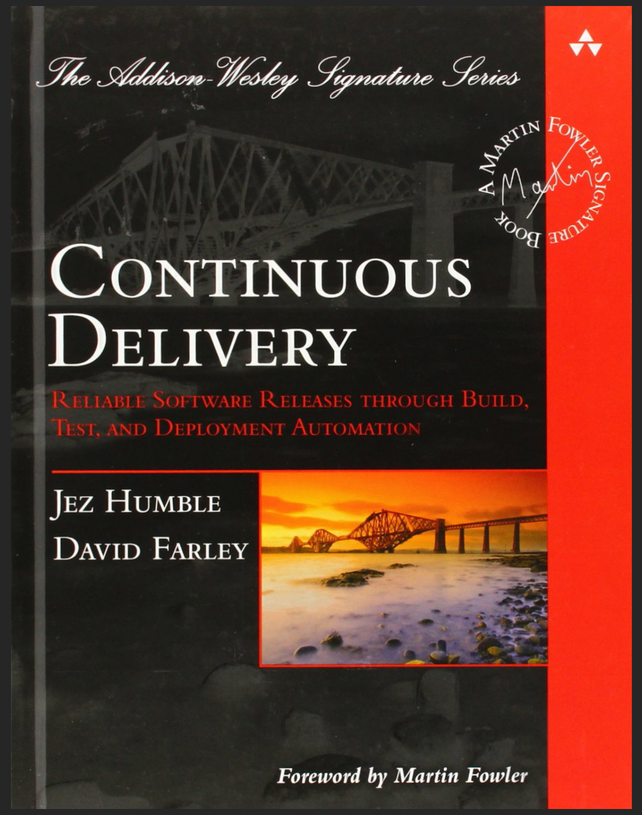
Auf der Mainline zu entwickeln, wird auch als **Trunk Based Development** bezeichnet.

Dadurch bekommen wir

early integration statt

late integration.



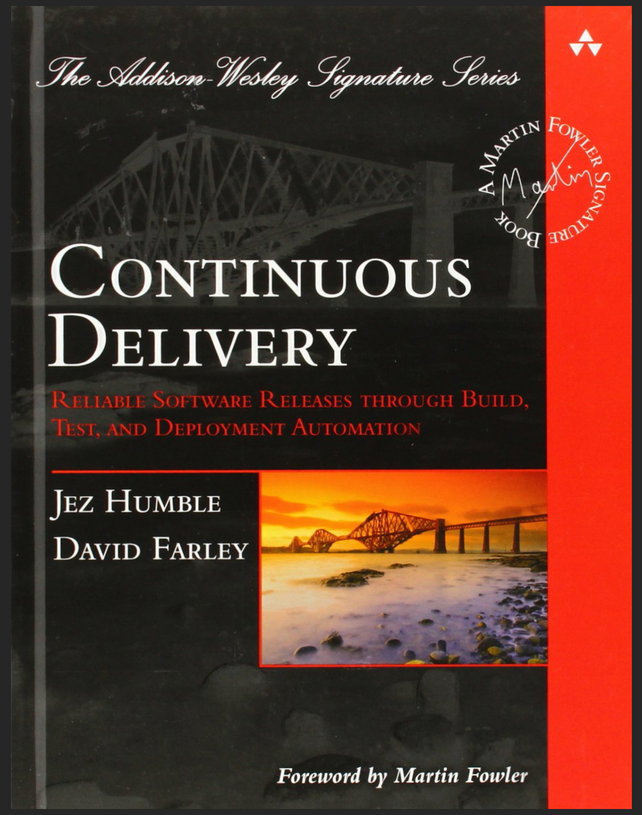


- Ensuring that **all code is continuously integrated**
- Ensuring **developers pick up each others' changes immediately**
- **Avoiding „merge hell“ and „integration hell“** at the end of the project

Aber wir bauen von der Mainline unsere

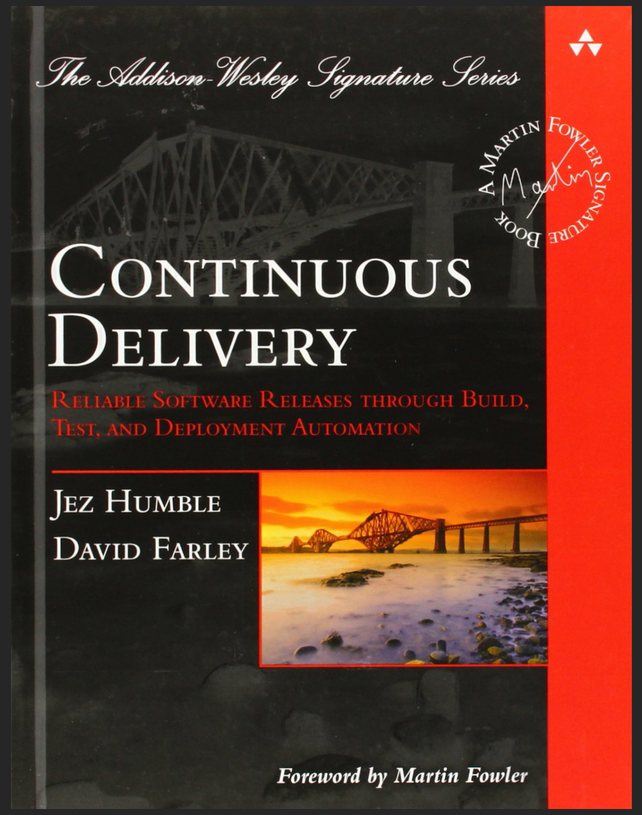
Releases!

Wie soll das gehen? 😱



- Develop on Mainline
- **Branch for Release**
- Branch by Feature

Continuous Delivery



„It is worth emphasizing that

branching by feature is really the antithesis of continuous integration,

…

Continuous Delivery

Ok, aber zum Release sind

nicht alle Features fertig!

Deswegen gilt:

„Keep your application releasable!“

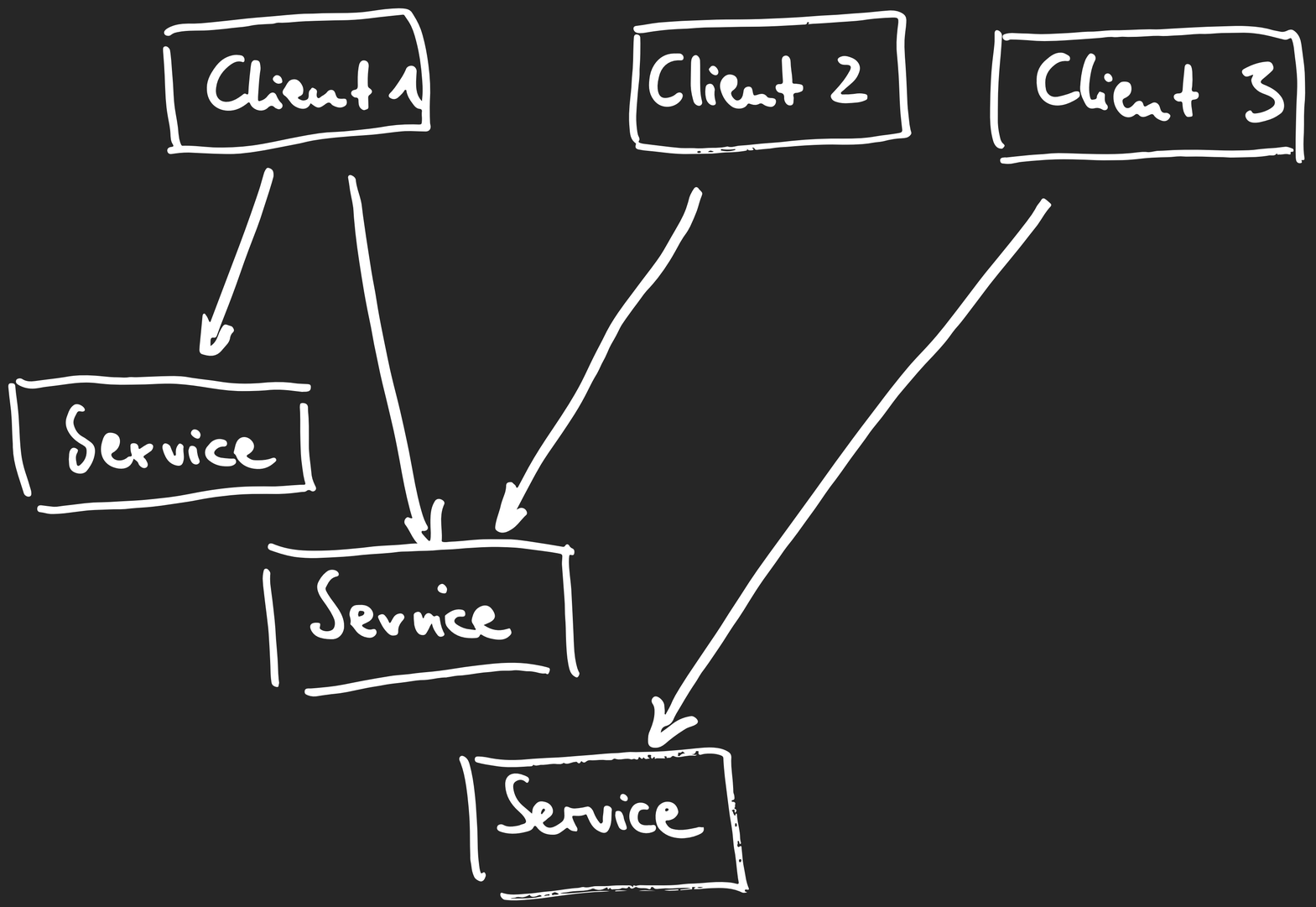

80
130

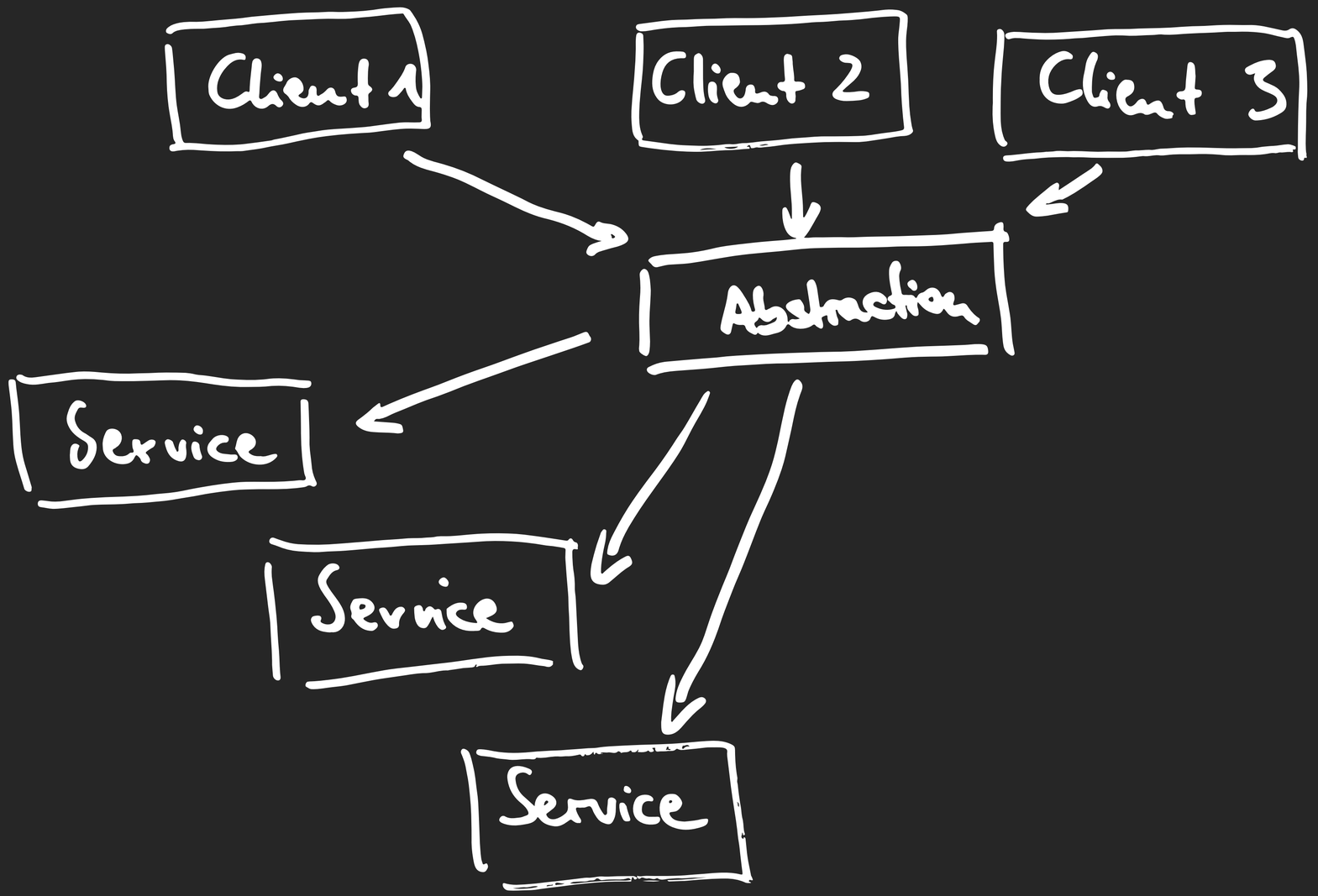
- **Hide new functionality** until it is finished.

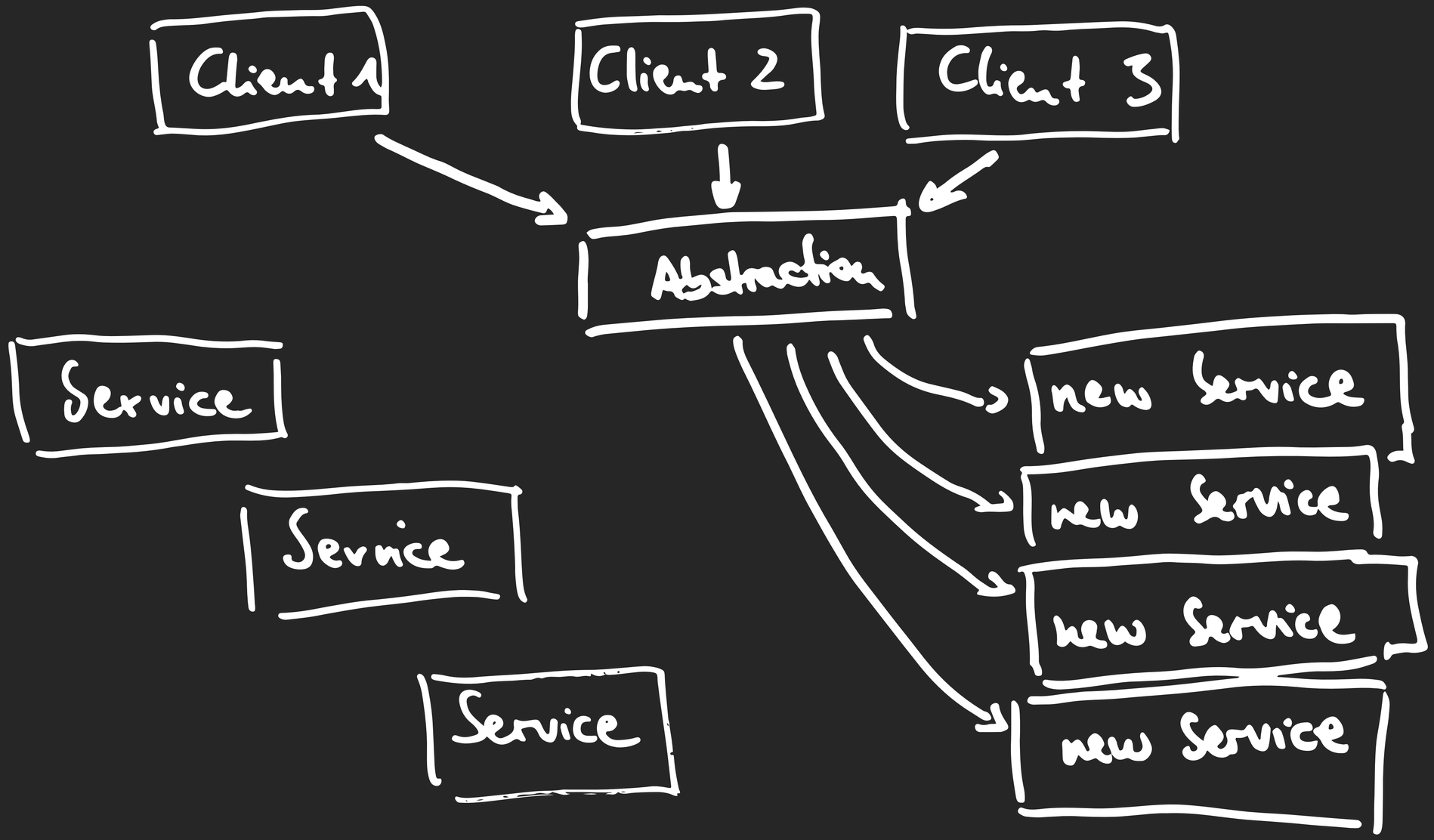
- Dark Releases
- Feature Toggles
- Build Time Switches

- **Hide new functionality** until it is finished.
- Make all **changes incrementally** as a series of **small changes**, each of which is releasable.

- **Hide new functionality** until it is finished.
- Make all **changes incrementally** as a series of **small changes**, each of which is releasable.
- Use **branch by abstraction** to make large-scale changes to the codebase.


Client 1
Client 2
Client 3
Service
Service
Service


Client 1
Client 2
Client 3
Abstraction
Service
Service
Service

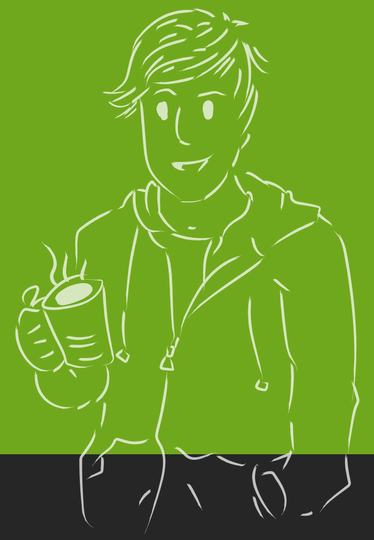

Client 1
Client 2
Client 3
Abstraction
Service
Service
Service
new Service
new Service
new Service
new Service

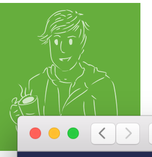
- **Hide new functionality** until it is finished.
- Make all **changes incrementally** as a series of **small changes**, each of which is releasable.
- Use **branch by abstraction** to make large-scale changes to the codebase.
- Use components to decouple parts of your application that change at different rates.


Fragen zum Vortrag?

Kolumne:
DevOps Stories

von Konstantin Diener





(Un)regelmäßige Integration: Stolpersteine für Continuous Integration

Nur noch wenige Stunden bis zum Sprint Review. Das MusicStore-Team arbeitet intensiv daran, letzte Stories fertigzustellen.

Martin: *„Tschakka, Erster!"*

Christian: *„Hä?"*

Martin: *„Ich habe meinen Feature Branch in den Master gemergt. Das war ein hartes Stück Arbeit. Ihr müsst bei euren Branches aufpassen. Ich habe vor allem in den Services für den Recommender und die Timeline einiges geändert."*

Christian: *„Super, ich auch. Und wenn ich das richtig verstanden habe, hat Lukas dort auch einiges geändert."*

Lukas: *„Ja, habe ich."*

Christian: *„Und Martin ist noch stolz drauf, dass wir jetzt die Merge Hell haben! Danke, Martin. Wir haben noch zwei Stunden bis zum Review und dürfen das jetzt unter Hochdruck zusammenpuzzeln."*

Martin: *„Sorry, aber das Feature war echt knifflig. Ich habe ja beinahe den ganzen Sprint für die Entwicklung gebraucht."*

Ruben: *„Der Build nach deinem Merge ist gerade gelaufen. Ich hätte erwartet, dass einiges an Tests dazukommt. Die Anzahl der Testfälle ist aber fast gleich geblieben."*

Martin: *„Ich habe doch gesagt, dass das Feature knifflig war. Für zusätzliche Tests hatte ich keine Zeit!"*

Ruben: *„In unserer Definition of Done steht doch aber, dass es für jede Klasse Tests gibt."*

Martin: *„Die gibt es ja auch. Nur halt nicht für die neuen Funktionen."*

Christian: *„Na, das ist doch sowieso egal. Ein Teil der vorhandenen Tests schlägt seit ungefähr einer Woche fehl, und das interessiert hier auch niemanden."*

Martin: *„Aber ich musste doch das Feature fertigmachen, damit wir es im Review zeigen können …"*

Lukas: *„Wie möchtest du das Feature denn im Review zeigen? Die Tests schlagen fehl und der Build ist rot. Der neue Code ist also nicht auf der Demo-Environment deployt worden."*

Martin: *„Ups, stimmt. Wollen wir dann die Tests fürs Review ausschalten und danach wieder an?"*

Christian: *„Was ein Quatsch. Dann können wir deinen Kram gleich von Hand auf der Umgebung deployen!"*

Martin: *„Aber dann hätten wir doch keine Continuous Integration mehr."*

Ruben: *„Haben wir die denn jetzt?"*

Continuous Integration heißt …

Rubens Frage ist absolut berechtigt. Ja, das Team hat einen Continuous-Integration-Server, der auf Änderungen im Versionskontrollsystem reagiert, den Build durchführt und im Erfolgsfall das entstandene Artefakt auf eine Umgebung deployt. Aber bringt dieser Prozess dem Team im aktuellen Zustand irgendeinen Erkenntnisgewinn?

Porträt

Konstantin Diener ist CTO bei cosee. Er beobachtet immer wieder die Probleme von Continuous Isolation und CI Theatre. Als CTO versucht er, die Teams bei der Anwendung von Trunk Based Development und sinnvoller CI zu unterstützen.

@onkelkodi https://cosee.biz

Kolumne:
DevOps Stories

von Konstantin Diener

cosee TechTalks

talks.cosee.biz

Dieser Vortrag bei euch?

cosee.biz

Picture credits:
Verkehrszeichen: https://www.shutterstock.com/g/FocusDzign
Theaterbühne: https://www.shutterstock.com/g/Mario+Lisovski
Autobahnbaustelle: https://www.istockphoto.com/de/portfolio/orinoco-art
Handzeichen: https://www.shutterstock.com/de/g/robertkneschke

Konstantin Diener | cosee GmbH

konstantin.diener@cosee.biz | @onkelkodi

cosee

Dieser Vortrag bei euch?

cosee.biz

Picture credits:
Verkehrszeichen: https://www.shutterstock.com/g/FocusDzign
Theaterbühne: https://www.shutterstock.com/g/Mario+Lisovski
Autobahnbaustelle: https://www.istockphoto.com/de/portfolio/orinoco-art
Handzeichen: https://www.shutterstock.com/de/g/robertkneschke

Konstantin Diener | cosee GmbH

konstantin.diener@cosee.biz | @onkelkodi