

Testing Casual Talks #2

大規模 Web サービスの ブラウザテスト自動化・高速化

2015/05/25 (Mon)

@deme0607



清水 直樹 (Naoki Shimizu)

@deme0607



Software Engineer @DeNA

2013年4月入社 (新卒)

SWET (Software Engineer in Test) 所属

Mission

ゲームプラットフォームの品質保証・向上

開発生産性の向上

2015年5月

株式会社ペロリ出向

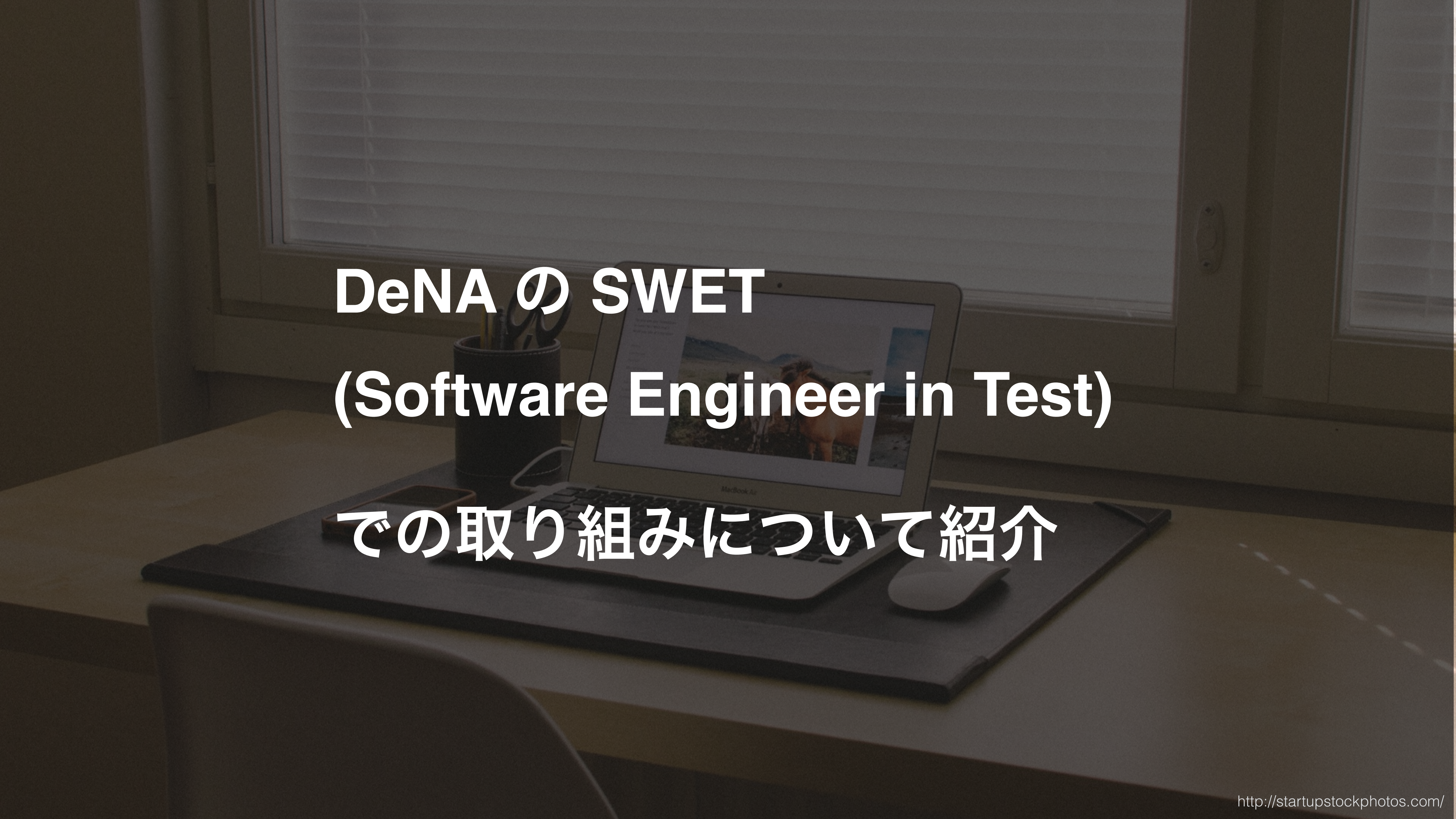
女性向けメディア

MERY の開発・運用



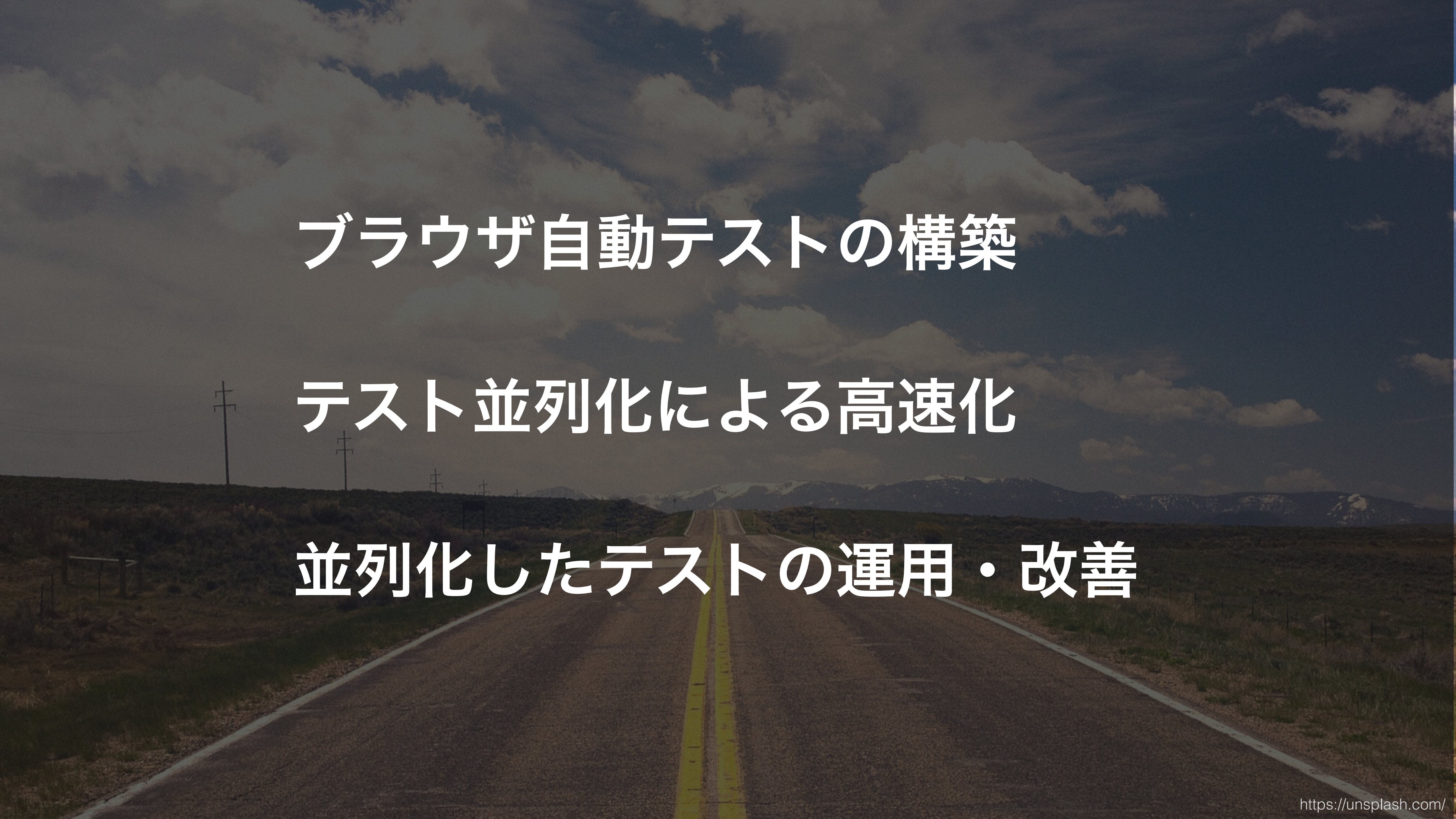


大規模 Web サービスの
ブラウザテスト自動化・高速化

A dimly lit desk with a laptop, mouse, and pen holder. The laptop screen shows a landscape with a horse. The text is overlaid on the image.

DeNA の SWET (Software Engineer in Test)

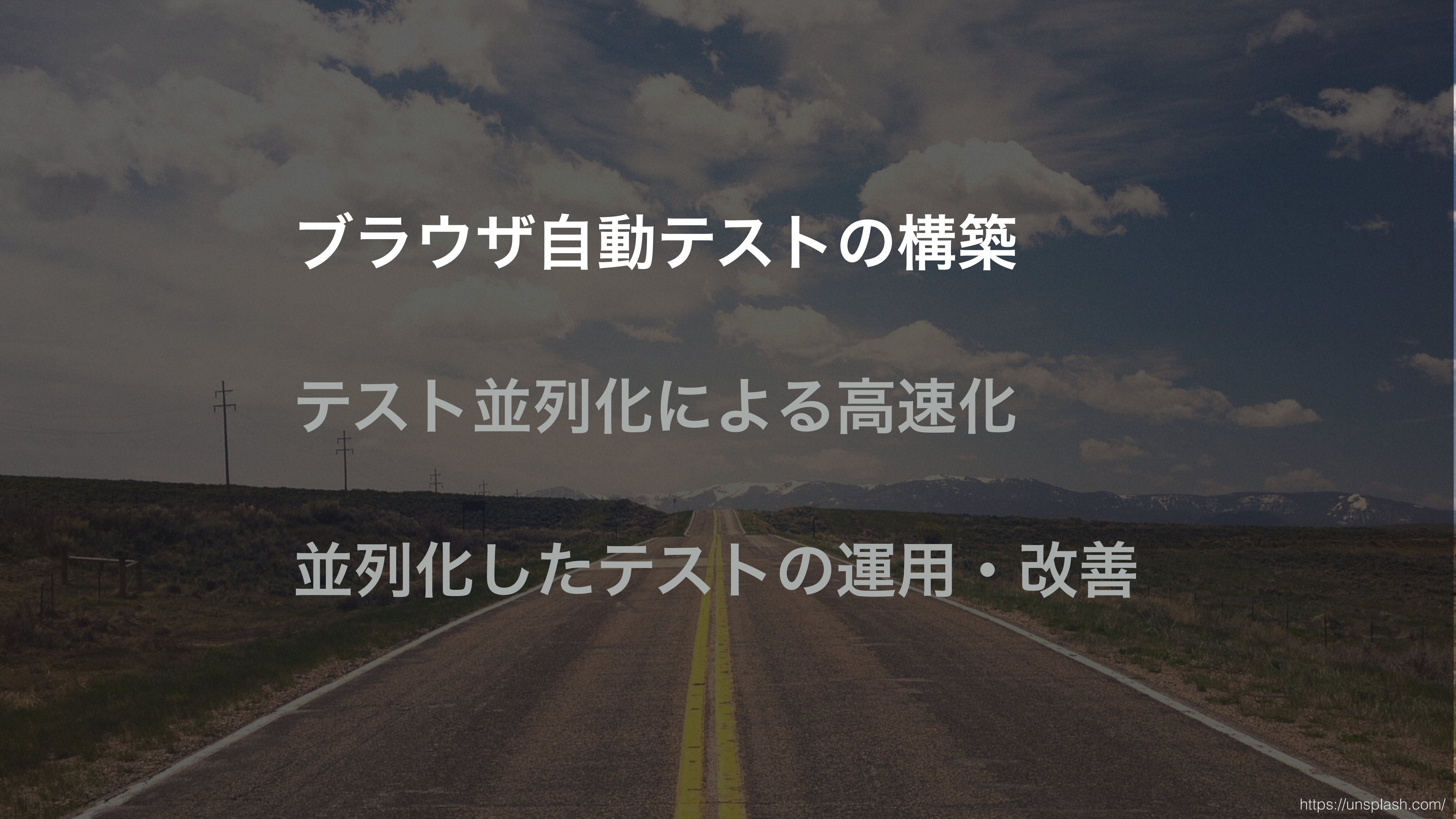
での取り組みについて紹介



ブラウザ自動テストの構築

テスト並列化による高速化

並列化したテストの運用・改善



ブラウザ自動テストの構築

テスト並列化による高速化

並列化したテストの運用・改善

NBPF

Next Browser Platform

次世代ブラウザゲームプラットフォーム

OpenID Connect + JavaScript SDK

従来の Mobage プラットフォームとも連携



NBPF に対する ブラウザ自動テストを構築

開発者によるユニットテスト + SWET による E2E テストのアプローチ

デプロイ・リリース時の受け入れテスト

SDK やプラットフォーム機能の正常系・異常系テスト操作を自動化

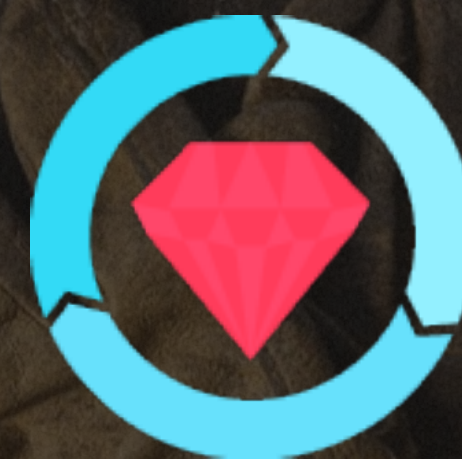
Selenium WebDriver

ブラウザ操作の自動化



RSpec / Capybara

テストフレームワーク (Ruby)



Jenkins

CI環境



ブラウザ自動テスト拡充の利点

#1 効率的なリグレッションテスト

#2 マニュアル検証工数の削減・集中



ブラウザ自動テスト拡充の利点 #1

効率的なリグレッションテスト

基本的なテストケースは完全自動化

hotfix や小規模な改善リリース時の目視・手動の確認がほぼ不要に

リリースサイクルの高速化・品質の向上



ブラウザ自動テスト拡充の利点 #2

マニュアル検証工数の削減・集中

自動テストでは担保できないこともある

UI/レイアウトの確認や機種依存問題の発見 etc

QA 工数を新機能や UI/レイアウト 検証に集中

新機能の検証方針を考えることはコンピュータにはできない

ブラウザ自動テスト拡充の問題点



ブラウザ自動テスト拡充の問題点

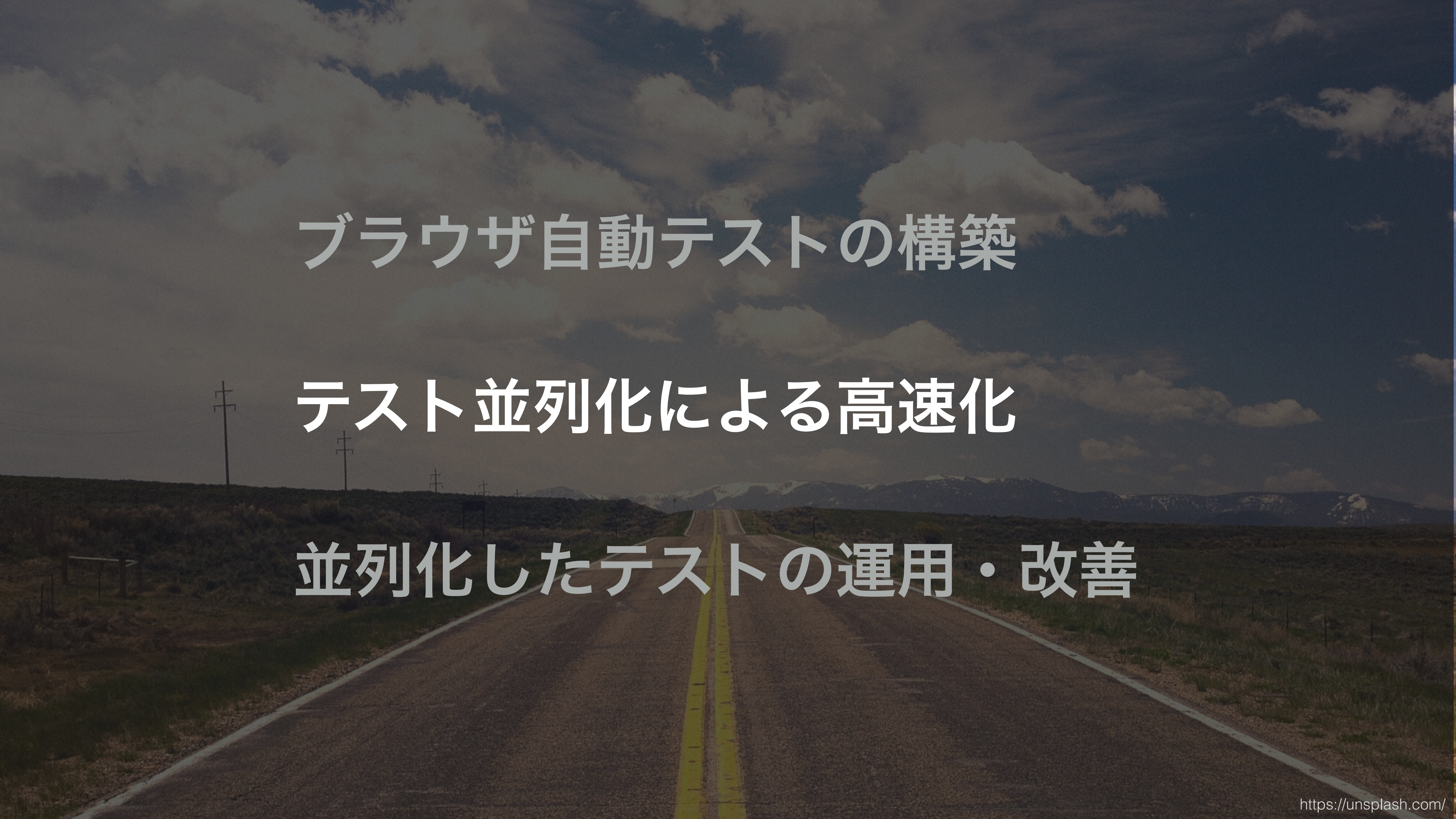
テスト実行時間の増加

ブラウザ自動テストは実行時間がかかる

HTTP リクエスト/レスポンス、Ajax、メール受信 etc.

実行するテストケースを削減することは避けたい

リグレッションテストとしての効果 down



ブラウザ自動テストの構築

テスト並列化による高速化

並列化したテストの運用・改善

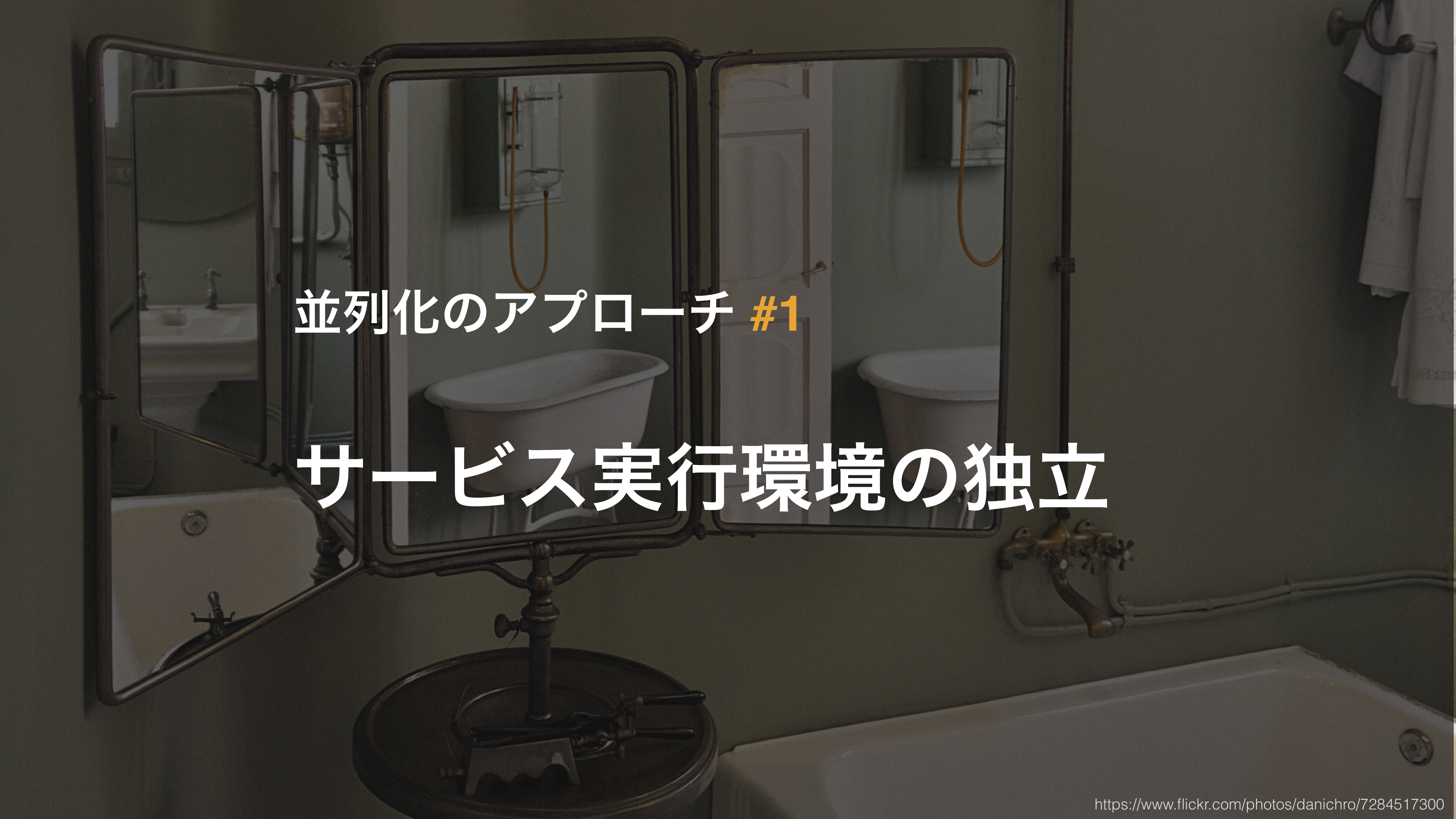
テストの並列化に必要なこと

並列実行しているテストの処理が互いに
衝突しないようにする

並列化のアプローチ

#1 サービス実行環境の独立

#2 テスト専用データの作成



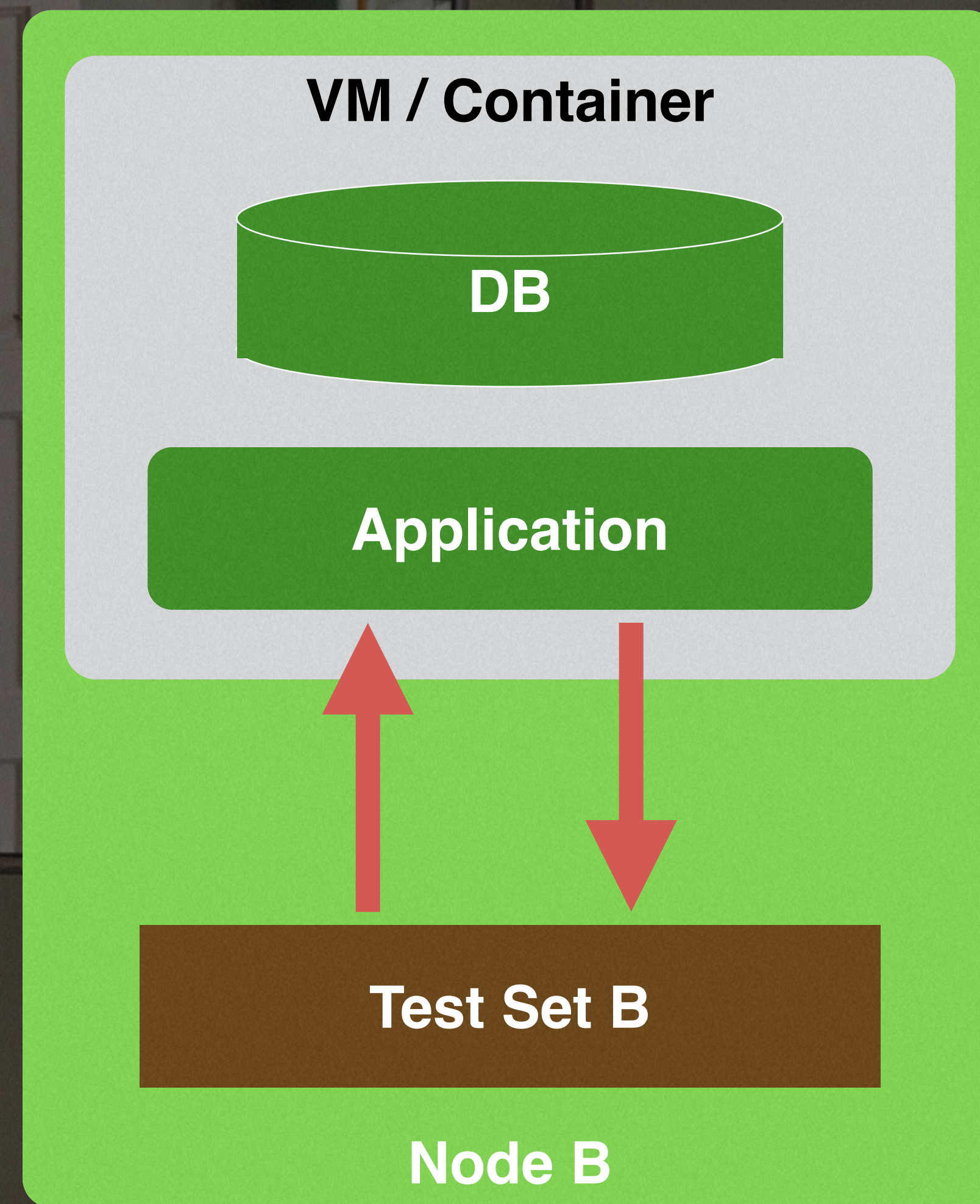
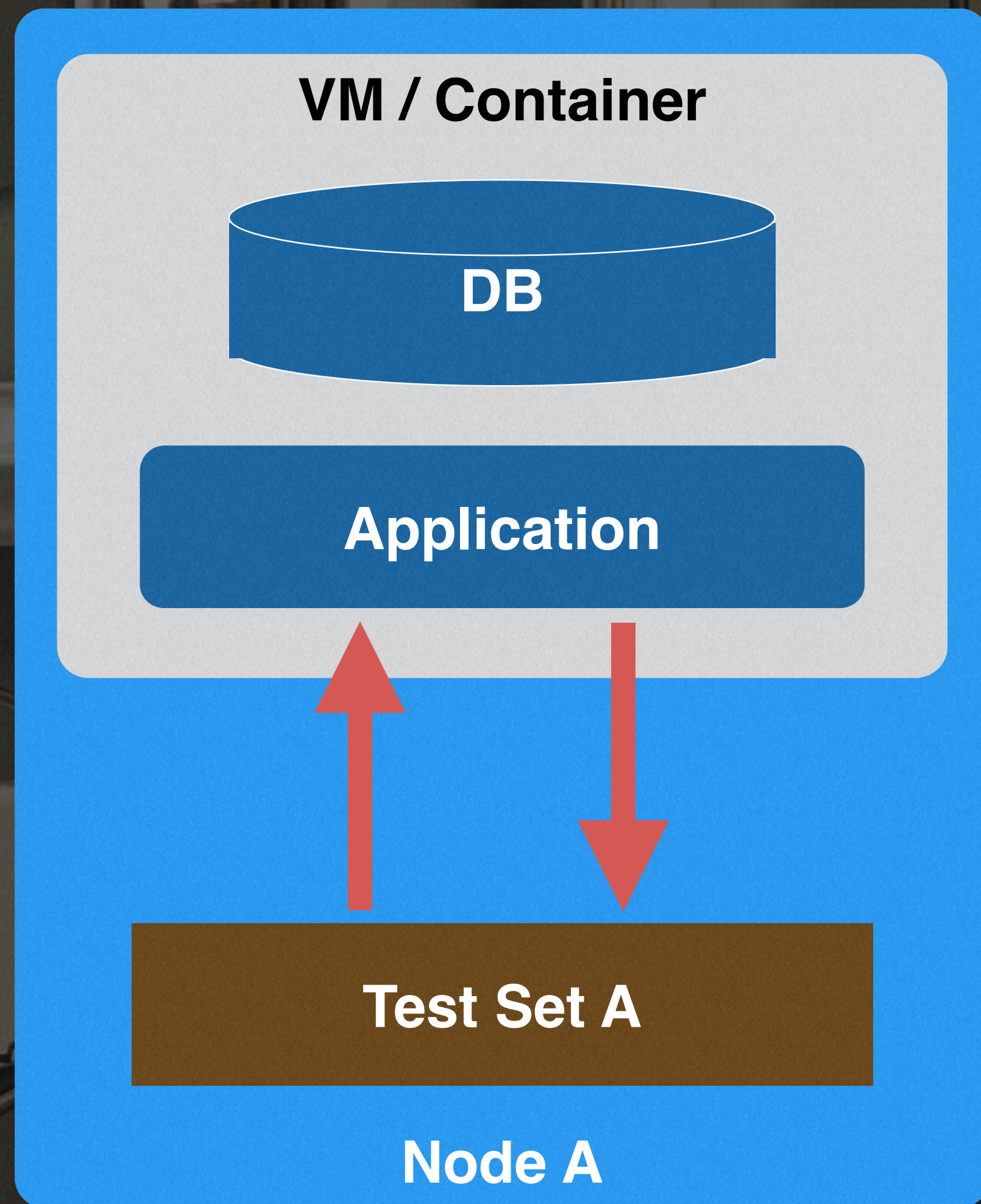
並列化のアプローチ #1

サービス実行環境の独立

テスト実行ノード毎に専用のサービス 実行環境を構築

Docker/Vagrant 等を利用して DB + アプリサーバを都度作成

各テストが専用の環境に向けて動作するので
衝突しない



問題点

大規模サービスの環境構築コストが大きい

複数サービスの依存関係解消

環境依存の問題にテストで気づけない

本番環境・実データでしか発生しない問題

並列化のアプローチ #2

テスト専用データの作成

テスト毎に専用のテストデータを作成

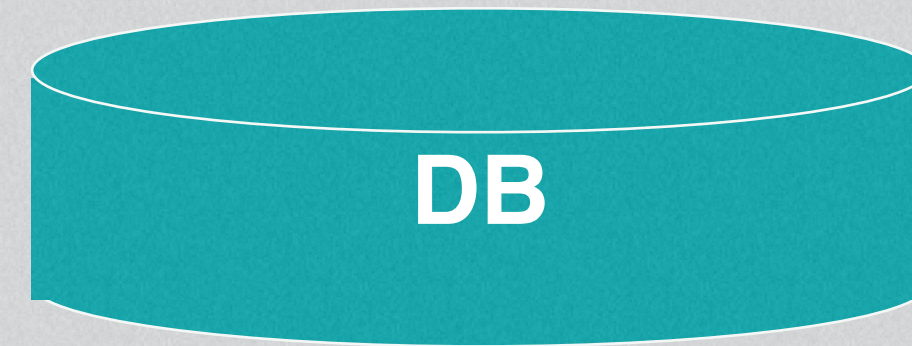
ユーザやゲーム ID etc

テストデータ作成手段

DB 操作

サービスで API を用意

Service Environment



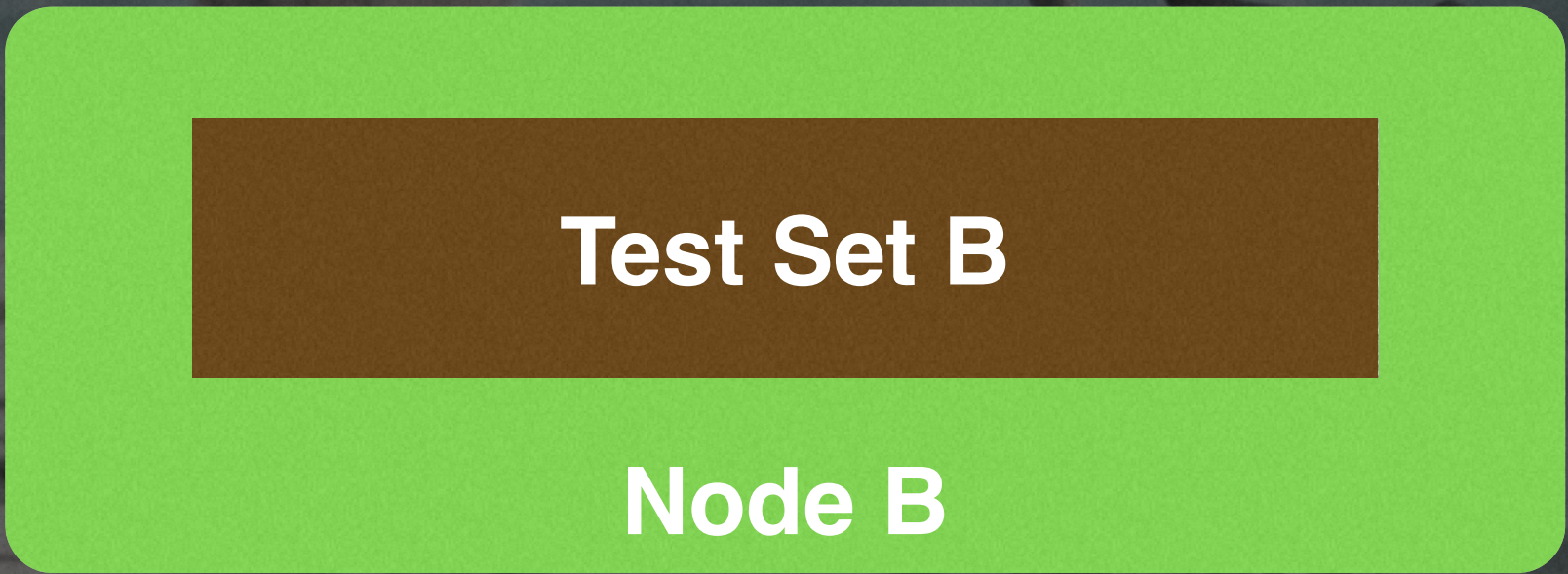
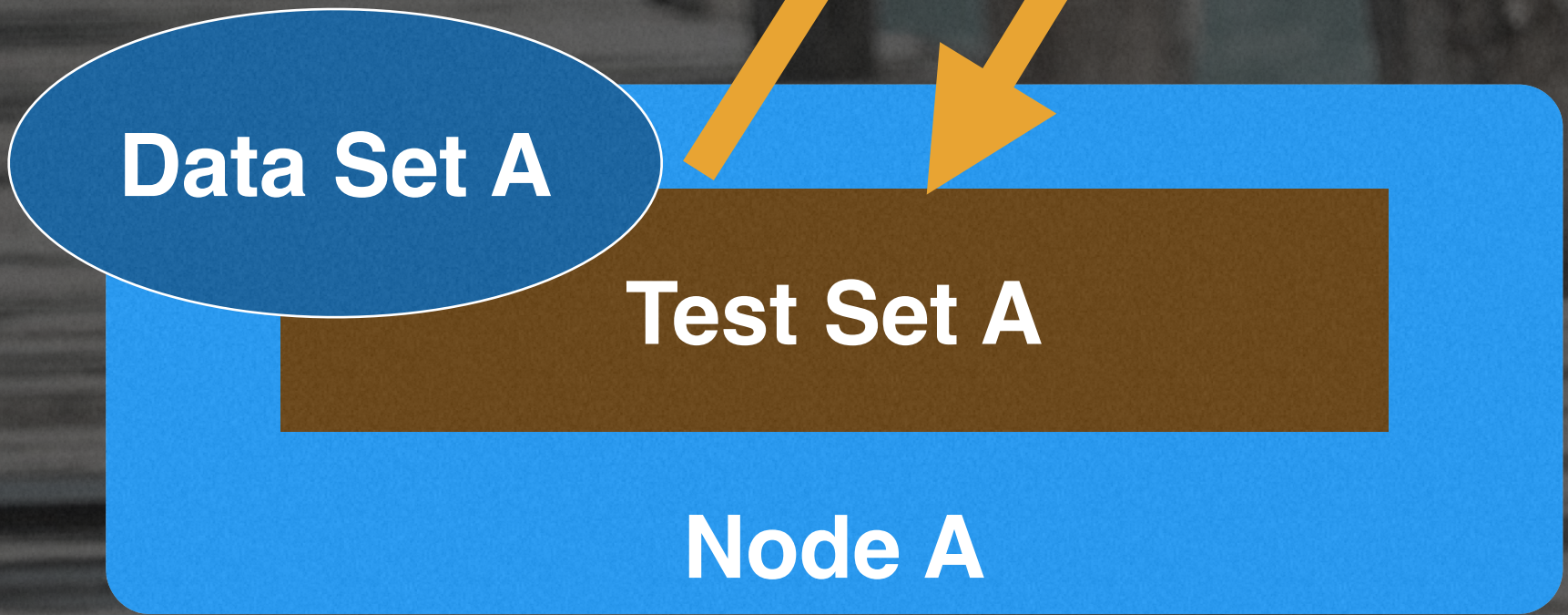
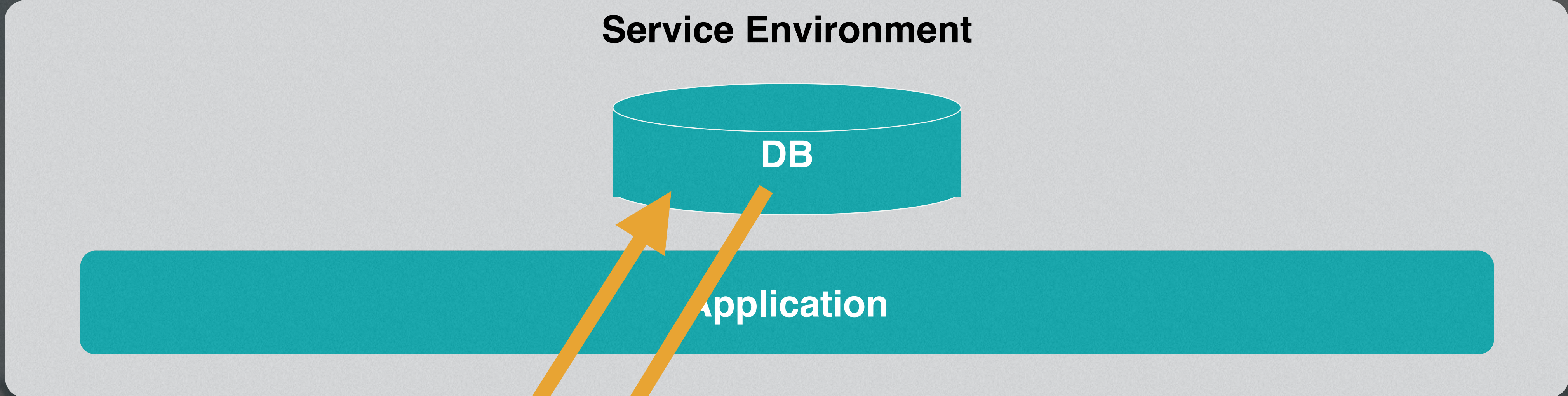
Application

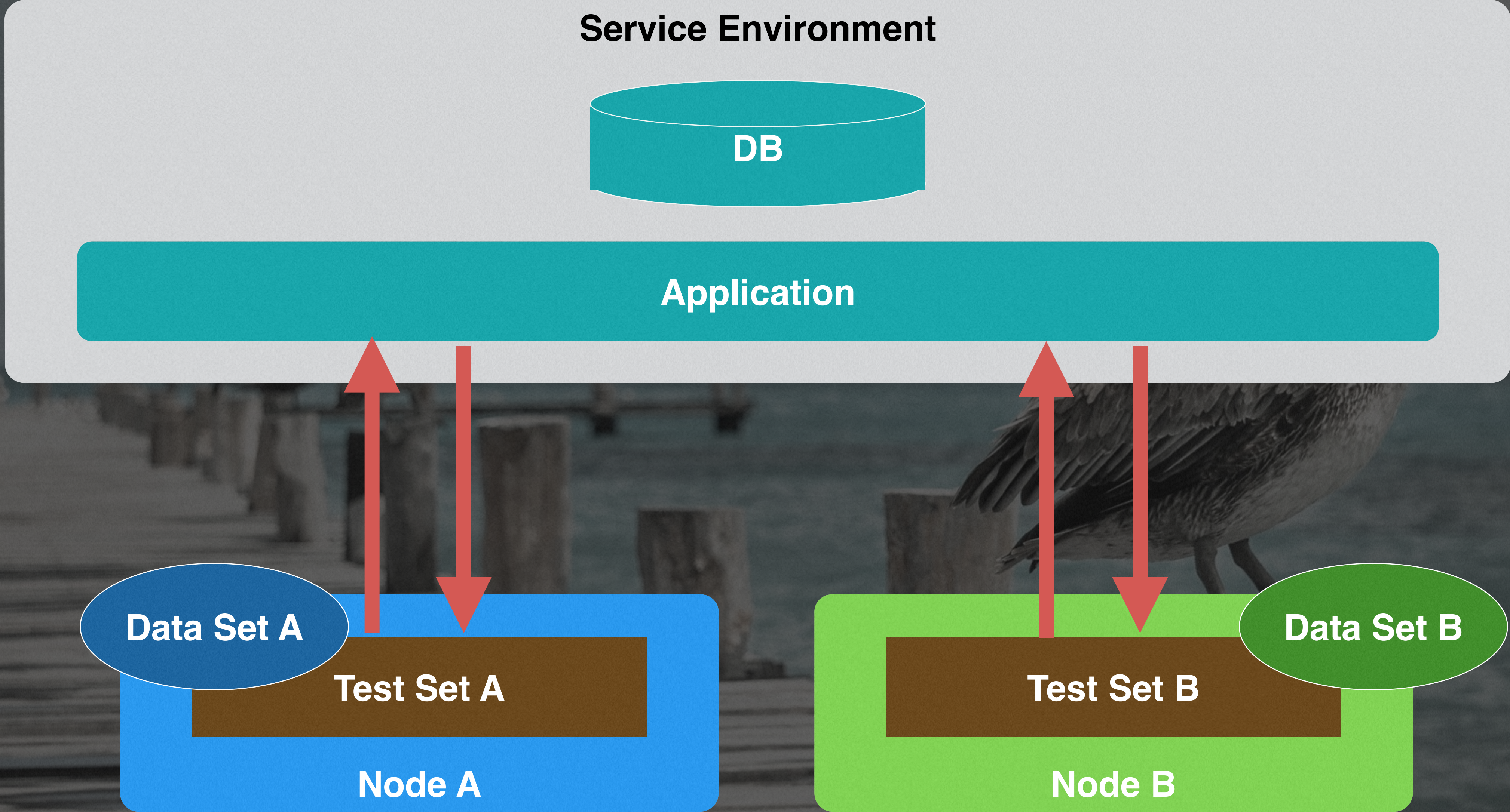
Test Set A

Node A

Test Set B

Node B





DB 操作によるテストデータ作成

仕様変更に従い辛くメンテナンスコスト高

サービス側のスキーマや処理が変わったらテストも修正

実環境の DB をテストから操作するリスク高

ユーザに不要なテストデータが見えてしまう

サービスに必要なデータを誤って削除



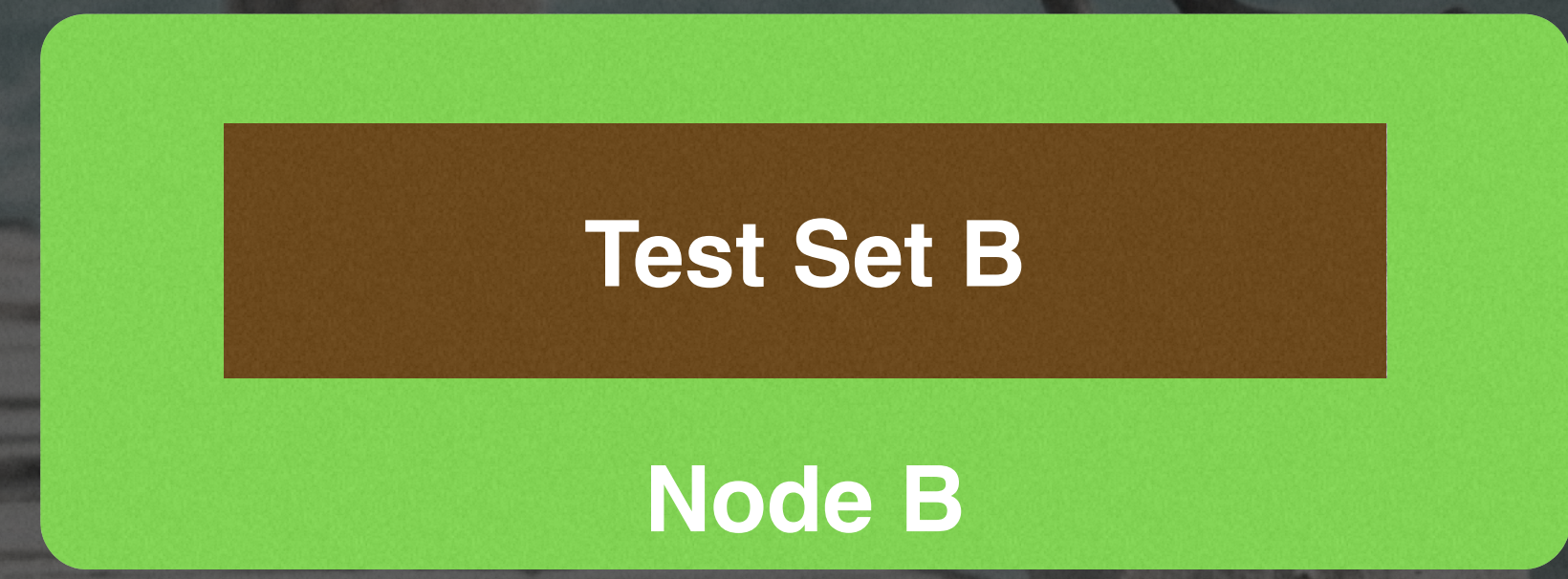
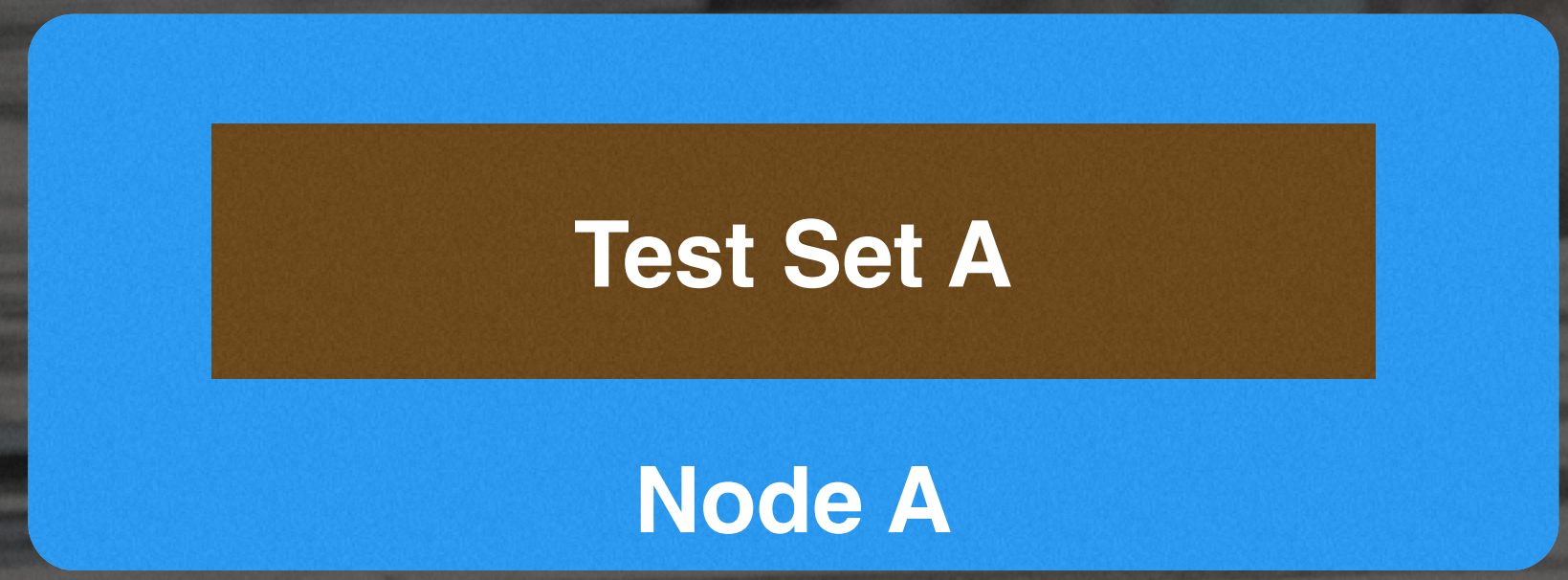
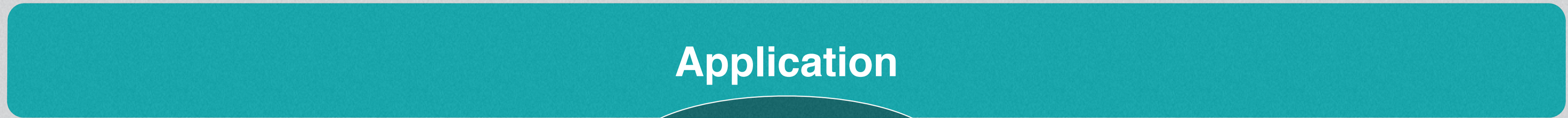
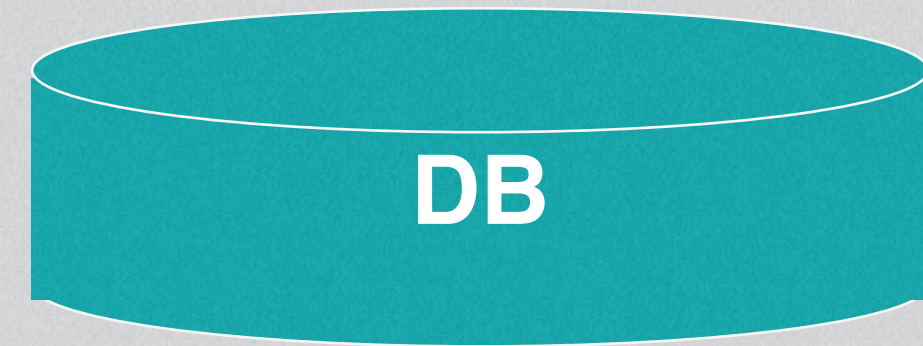
API によるテストデータの作成

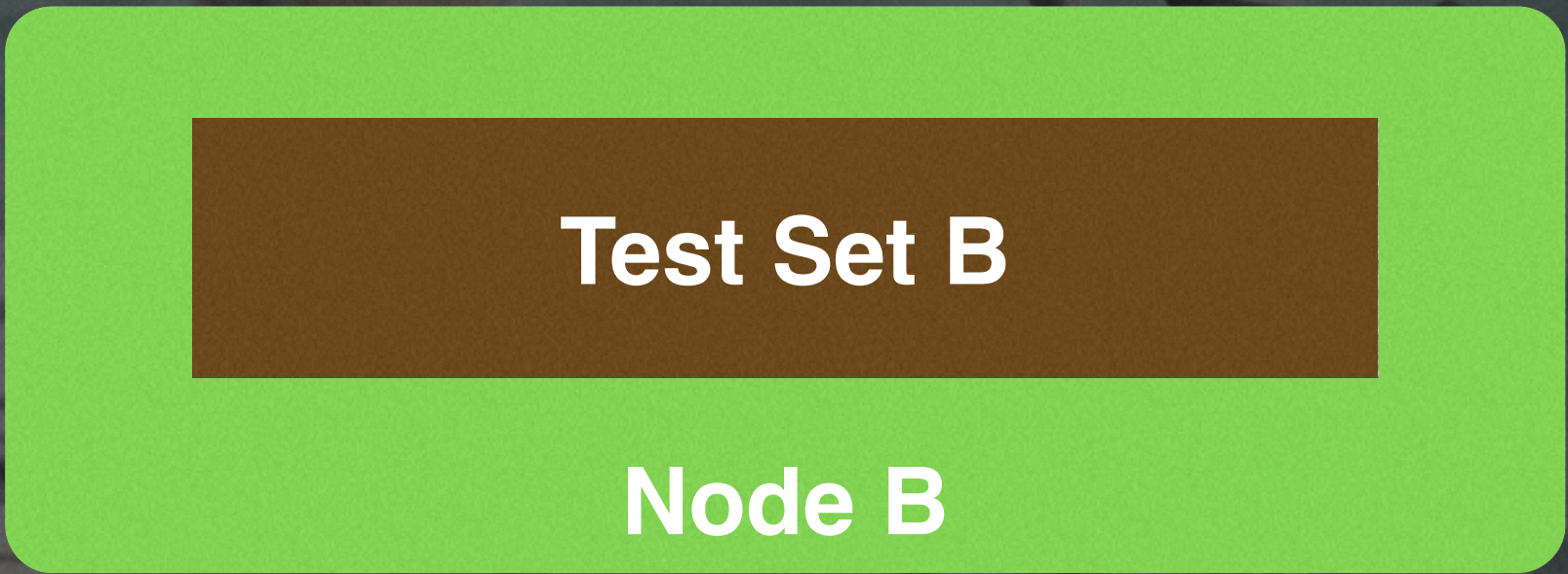
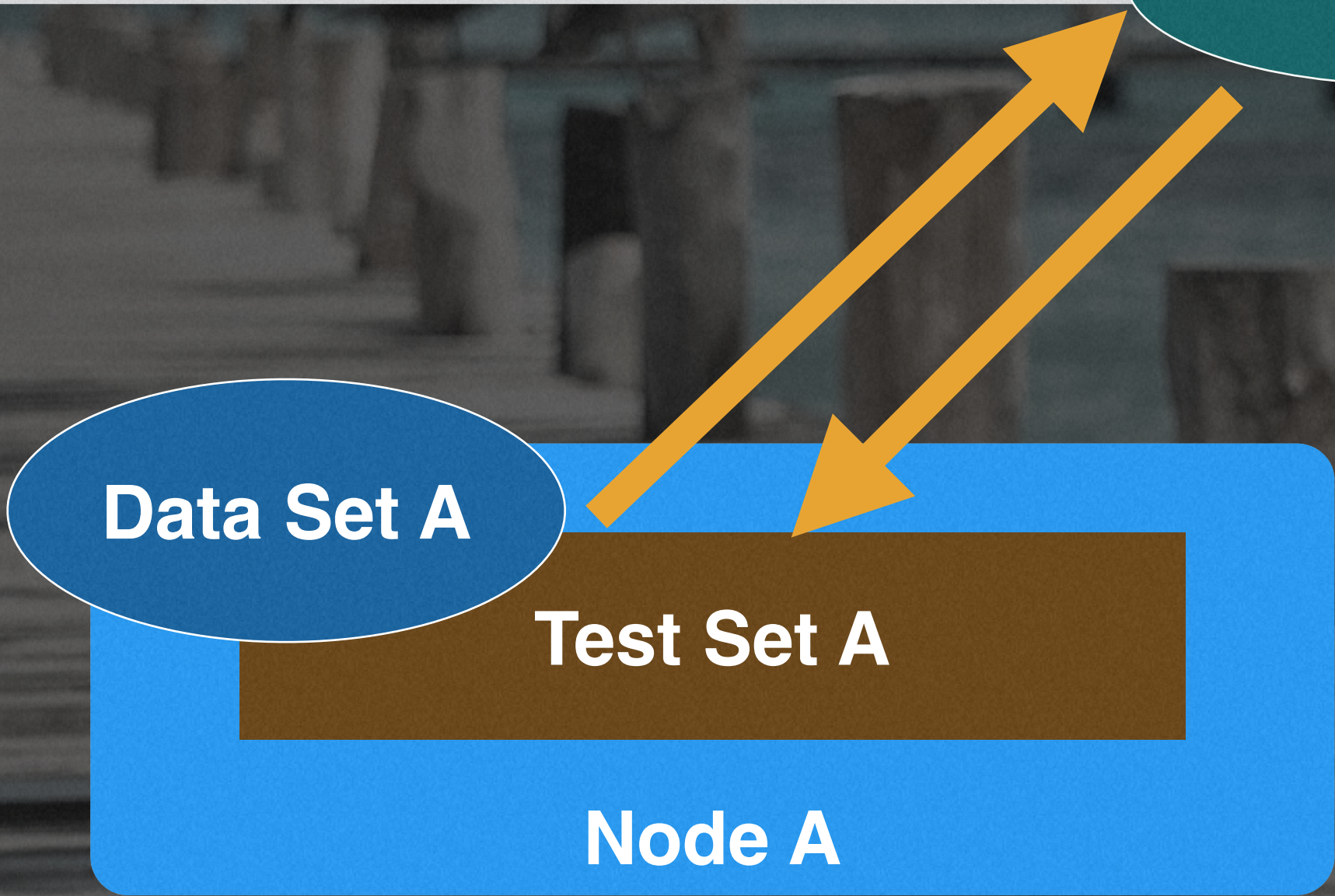
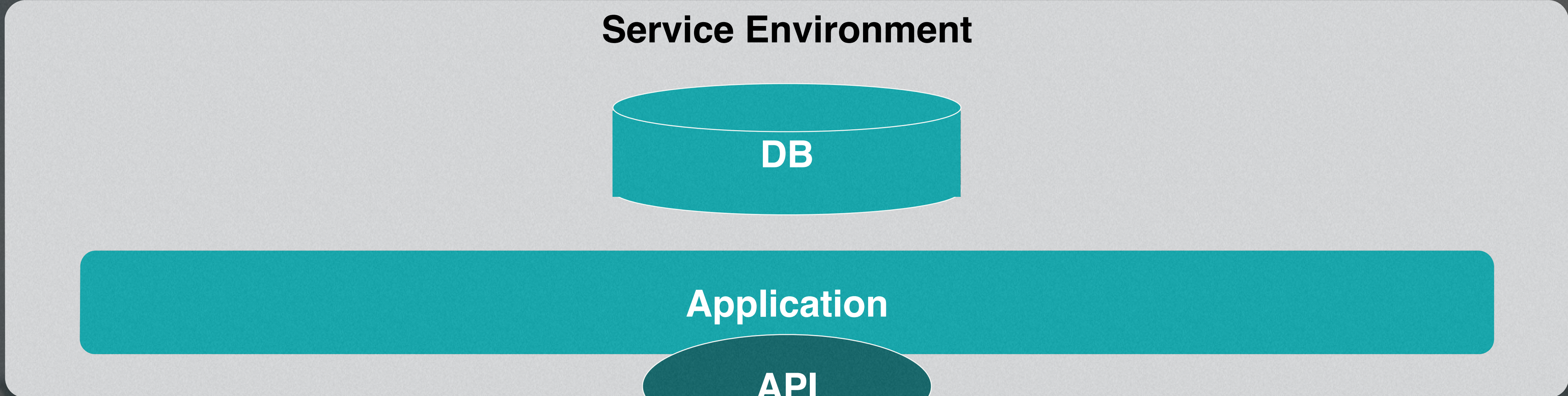
テストデータを作成できるAPIを提供

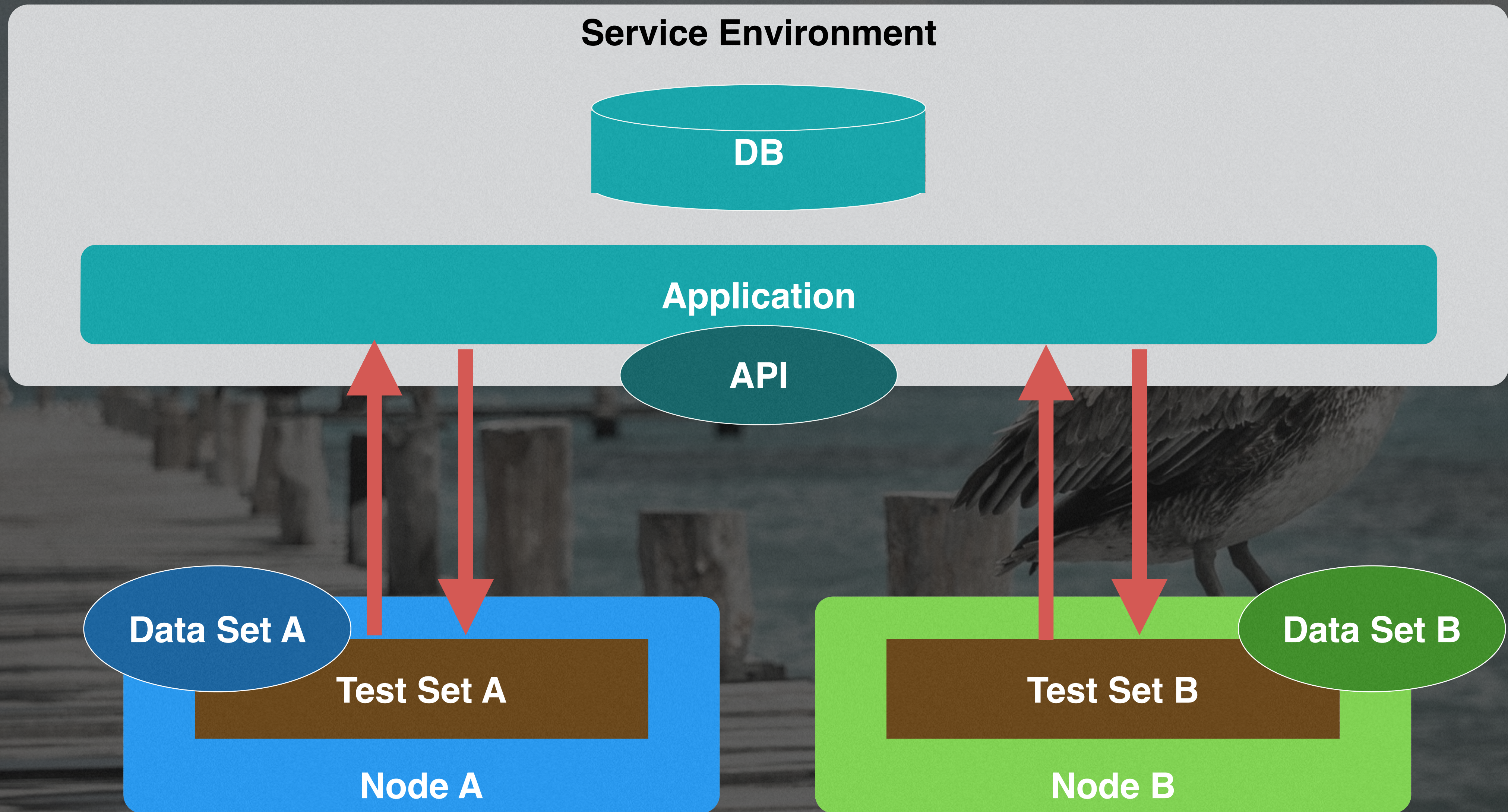
ユーザ作成 API・ゲーム作成 API etc.

テスト毎に API で専用データを作成

Service Environment









API が用意されているならばベストな選択肢

サービスも同じ API を利用してデータを作成

テストとサービス間でデータ構造に乖離が発生するのは NG

サービスの規模・構造によっては導入コスト高

長期運用によって大規模化したサービス/モノリシックな構造のサービス etc.

NBPF のテスト並列化

全データソースの API 移行は未達成

NBPF リリース: 2014年

モバゲーリリース: 2006年

DB 操作はリスクが大きい

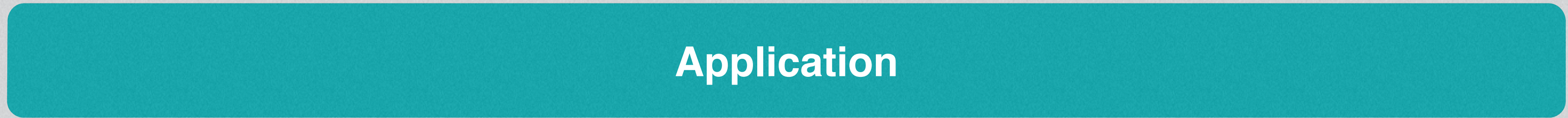
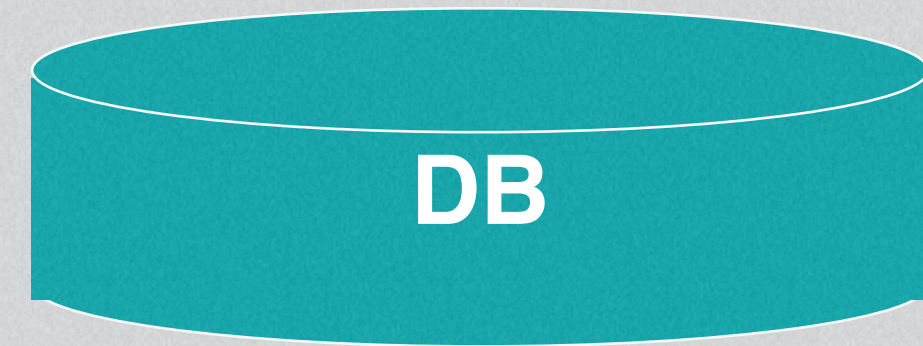
数千万ユーザがサービスを利用

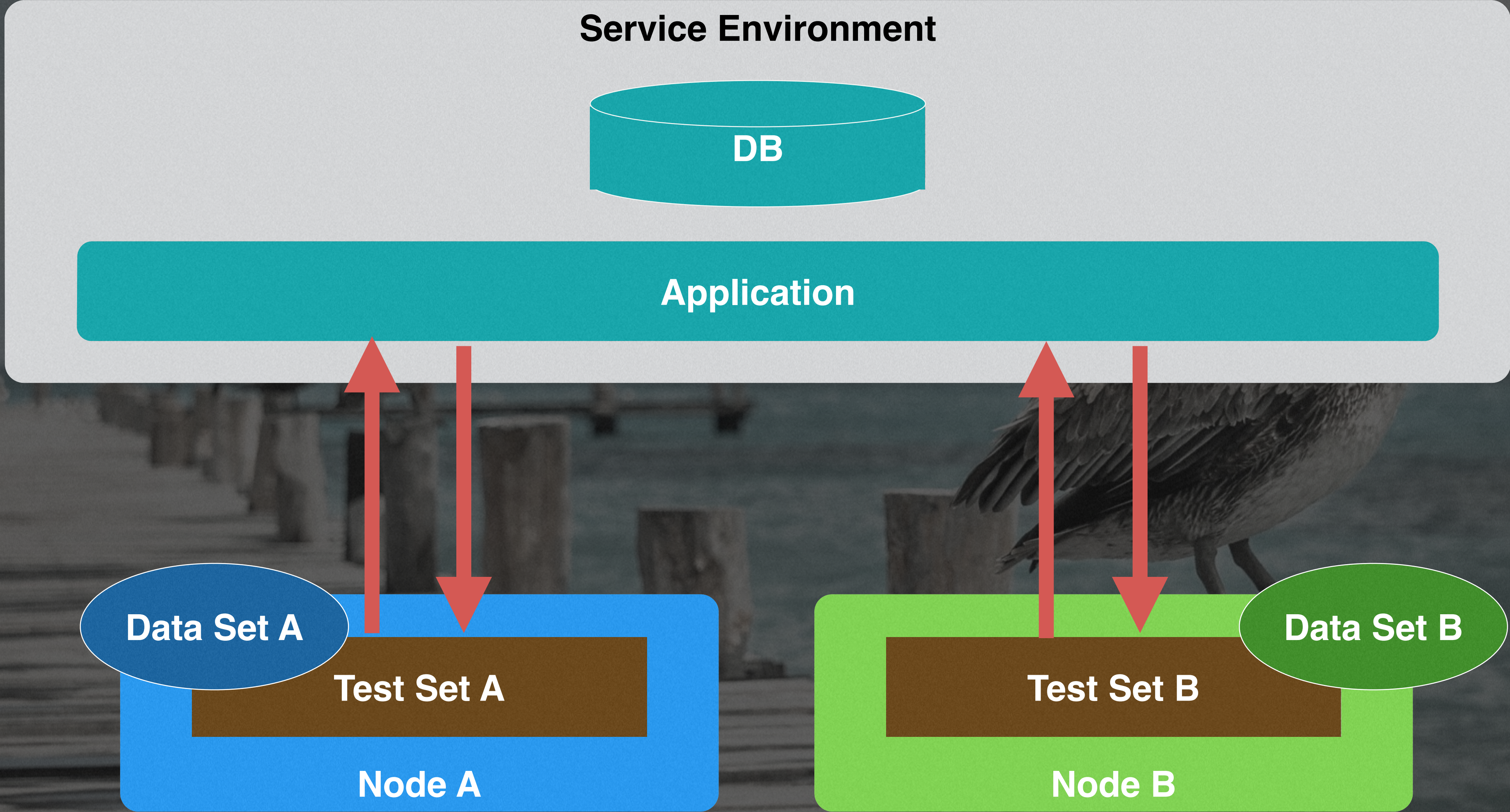


テスト実行時に作成できないデータは
事前に準備

各テスト実行ノードに専用のテストデータを
割当

Service Environment





サービスの UI 経由でユーザを事前作成

事前作成手順は WebDriver で自動化

UI 経由のデータ作成は実行時間がかかる

事前作成データの使い回しが必要

A muscular man is shown from the chest up, holding a clear plastic container filled with yellow pills in his left hand and a syringe in his right hand. The background is a plain, light-colored wall. The overall tone is serious and clinical.

事前作成データ利用

テストの後処理に注意

テストデータが壊れると次回以降のテストに影響

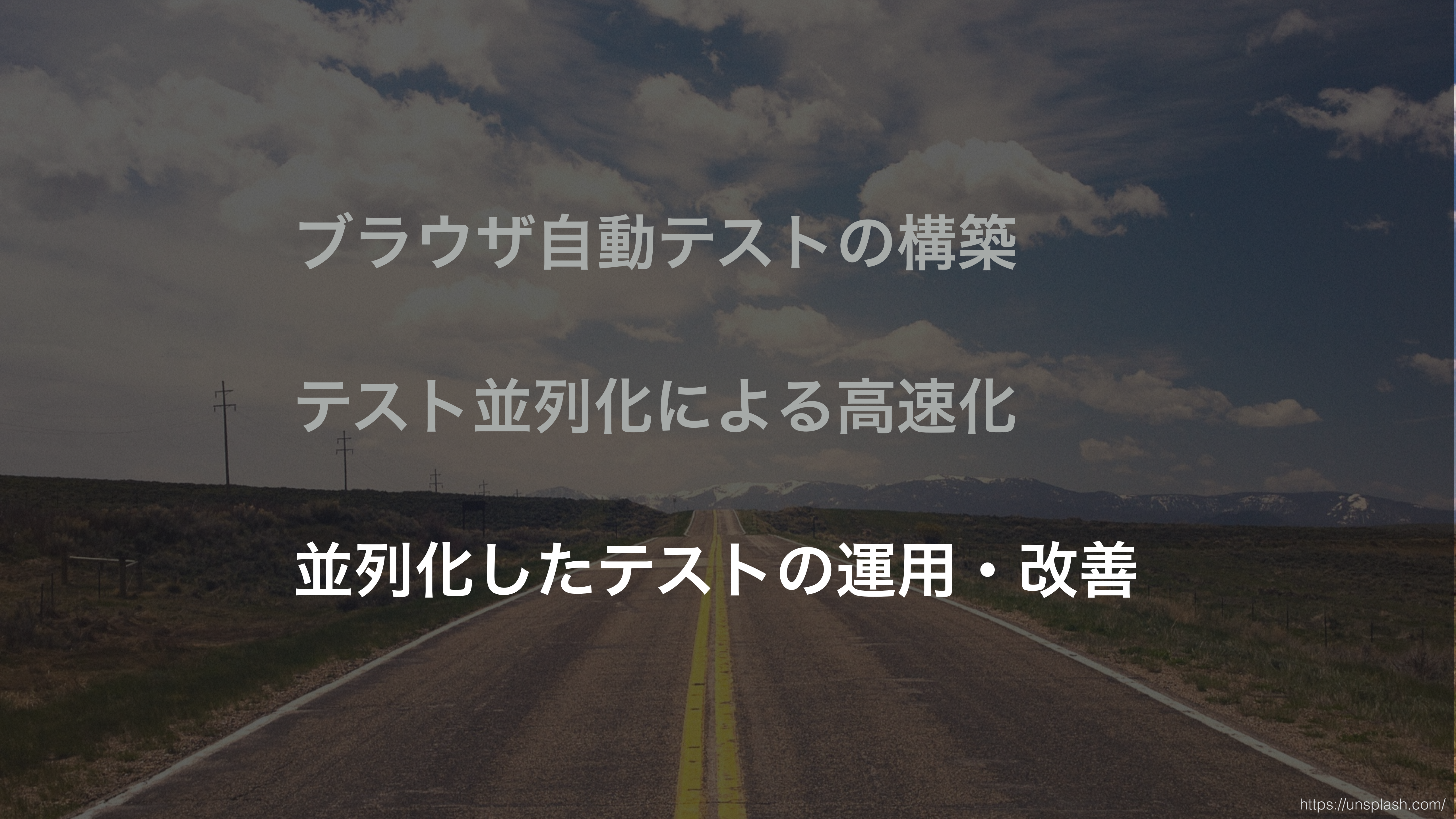
API で作成不可なデータが必要なら有効な選択肢

ブラウザ自動テストの並列実行環境が完成

8並列テスト on Jenkins

実行時間 2.5h → 30min

カジュアルに全テストケースのリグレッション
テストが実行可能



ブラウザ自動テストの構築

テスト並列化による高速化

並列化したテストの運用・改善

ブラウザ自動テストの並列実行環境が完成

8並列テスト on Jenkins

実行時間 2.5h → 30min

カジュアルに全テストケースのリグレッション
テストが実行可能

新たに顕在化した問題

Concourse A ↑
ターミナル Terminal
手荷物受取所 Luggage Claim
券 Hotel

#1 不安定なテスト

#2 不適切な並列実行計画

A Newton's cradle with a pen nib resting on a cylindrical base. The background is a dark, blurred landscape with a sunset or sunrise sky.

新たに顕在化した問題 #1

不安定なテスト

ブラウザ自動テストは不安定になりがち

タイムアウトエラー・Ajaxのタイミング etc.

テストに失敗していることに慣れてしまうと
本物のバグに気づけない

不安定なテストの解決策

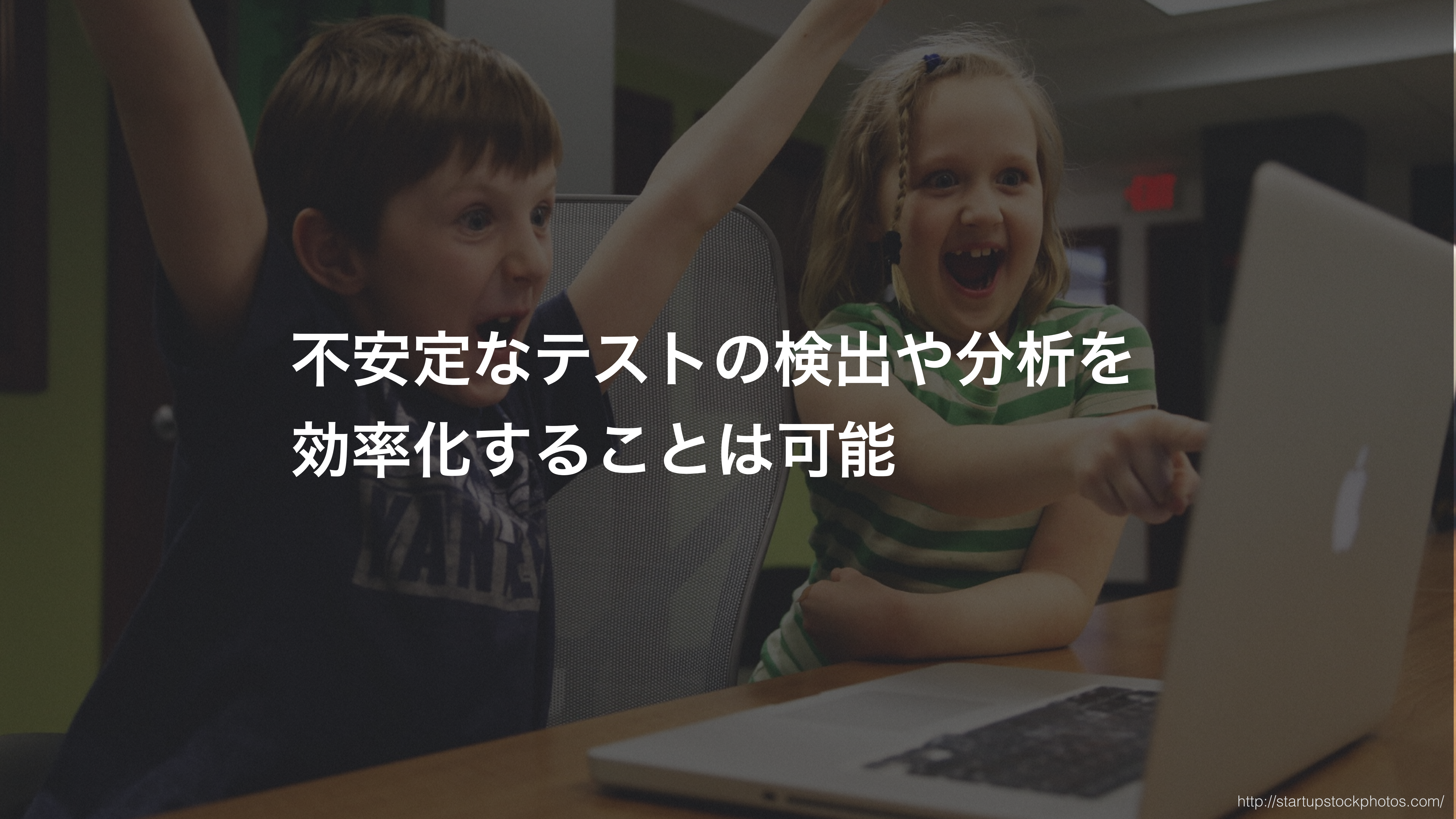
A photograph of a person's muscular torso and arms, holding a syringe filled with a yellow liquid. The image is overlaid with white Japanese text. The background is a plain, light-colored wall.

不安定なテストの解決策

地道にテストを修正していく

適切な待ち時間の挿入

失敗したテストのリトライ

A young boy and girl are sitting at a desk with two laptops. The boy is on the left, wearing a dark blue shirt, and the girl is on the right, wearing a green and white striped shirt. Both children have their arms raised and are looking at the laptops with expressions of excitement and joy. The background is a dimly lit office or classroom setting.

不安定なテストの検出や分析を
効率化することは可能

テスト結果の集計 API を作成

テスト実行毎に結果を POST し集計

環境横断で失敗しやすいテストの分析

Status	Region	Environment	Usage	filter					
all	all	all	all						
file_path	description	jp							
		dev		stg		prod			
		service	sandbox	service	sandbox	service	sandbox		
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable		
テストファイル・テストケース名		stable	unstable	stable	stable	stable	stable		
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable		
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable		
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable		
テストファイル・テストケース名		unstable	stable	failure	unstable	stable	stable		

環境横断で失敗しやすいテストの分析

1地域 x 3環境 x 2用途

Status	Region	Environment	Usage	filter
all	all	all	all	

file_path	description	jp					
		dev		stg		prod	
		service	sandbox	service	sandbox	service	sandbox
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable
テストファイル・テストケース名		stable	unstable	stable	stable	stable	stable
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable
テストファイル・テストケース名		unstable	stable	failure	unstable	stable	stable

環境横断で失敗しやすいテストの分析

Status	Region	Environment	Usage	filter					
all	all	all	all						
file_path	description	jp							
		dev		stg		prod			
service	sandbox	service	sandbox	service	sandbox				
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable		
テストファイル・テストケース名		stable	unstable	stable	stable	stable	stable		
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable		
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable		
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable		
テストファイル・テストケース名		unstable	stable	failure	unstable	stable	stable		

不安定なテスト

環境横断で失敗しやすいテストの分析

Status	Region	Environment	Usage	filter					
all	all	all	all						
file_path	description	jp							
		dev		stg		prod			
service	sandbox	service	sandbox	service	sandbox				
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable		
テストファイル・テストケース名		stable	unstable	stable	stable	stable	stable		
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable		
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable		
テストファイル・テストケース名		stable	stable	stable	stable	stable	stable		
テストファイル・テストケース名		unstable	stable	failure	unstable	stable	stable		

失敗したテスト

テストの安定性の評価

突然不安定になったテストケースを Slack に通知



slackbot 18:53



@channel: 3 test results status changed to failure from stable. Please check

テスト結果確認用 URL と不安定なテスト一覧を通知

failed test cases:

test_a

test_b

test_c

新たな問題 #2

不適切な並列実行計画

Test Case 1
3 min

Test Case 2
5 min

Test Case 3
4 min

Test Case 4
6 min

4 Test Cases, 18 min
Node A

Test Case 5
3 min

Test Case 6
3 min

Test Case 7
3 min

Test Case 8
3 min

4 Test Cases, 12 min
Node B

Test Case 1
3 min

Test Case 2
5 min

Test Case 3
4 min

Test Case 4
6 min

4 Test Cases, 18 min
Node A

Test Case 5
3 min

Test Case 6
3 min

Test Case 7
3 min

Test Case 8
3 min

Waste 6 minutes!!

4 Test Cases, 12 min
Node B

不適切な並列実行計画

マシンリソースを最大限活用できない

実行時間が長くなる



テスト結果集計 API を利用

過去のテスト実行時間から

各ノードの推定テスト実行時間が均等になる

実行計画を作成

Test Support API

Test Case 4
6 min

Test Case 1
3 min

Test Case 5
3 min

Test Case 6
3 min

4 Test Cases, 15 min
Node A

Test Case 2
5 min

Test Case 3
4 min

Test Case 8
3 min

Test Case 7
3 min

4 Test Cases, 15 min
Node B

今後の課題

#1 データ作成 API の充実

#2 分散実行ノードを動的に増やす仕組み

現状社内 Jenkins サーバ
クラウド化を検討

#3 並列実行計画の改善

Push 型より Pull 型のアプローチ

まとめ

大規模サービスのブラウザ自動テストを拡充

並列化による実行時間の短縮

テスト安定化・高速化のため独自 API で

テスト結果の集計・分析を実施