# デプロイ頻度x回/日を支えるテスト戦略

~ソフトウェアテスト自動化カンファレンス2023~

(株)スマートショッピング
エンジニア　野島大誠

# 自己紹介

- スマートショッピングにてエンジニア
  をしています
  - 最近はテストへの関心高め
  - スマートショッピングは、
    重量ベースで在庫管理をおこなう
    IoTプロダクト
- スマートショッピングのプロダクト
  いい感じの画像

# 今日のお題

- 弊社では1日x回のデプロイと、高頻度でデプロイをおこなっています
- これを支えている1要素としてテスト戦略を共有します
    - 詳細な技術要素については話しません

# 目次

- テスト戦略概観🔍
- Unit Test🔧
- Integration Test👨‍🔧
- End to End Test👮‍♂️
- Manual Test💪
- まとめ👀

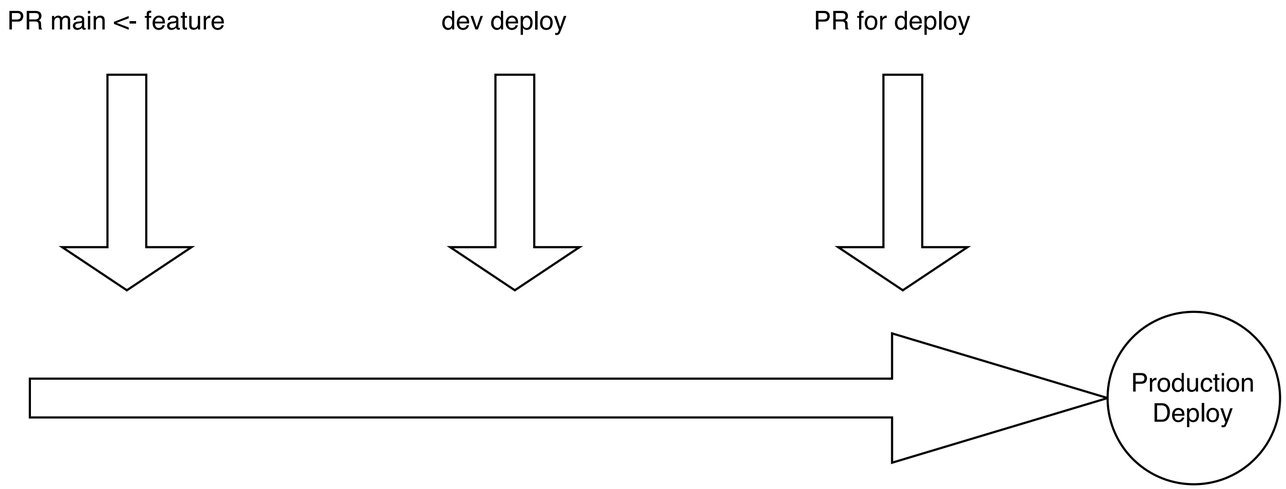
# テスト戦略概観🔍

# テスト戦略概観

- 1スプリント2週間を基本とする開発サイクル内で4種のテストが存在

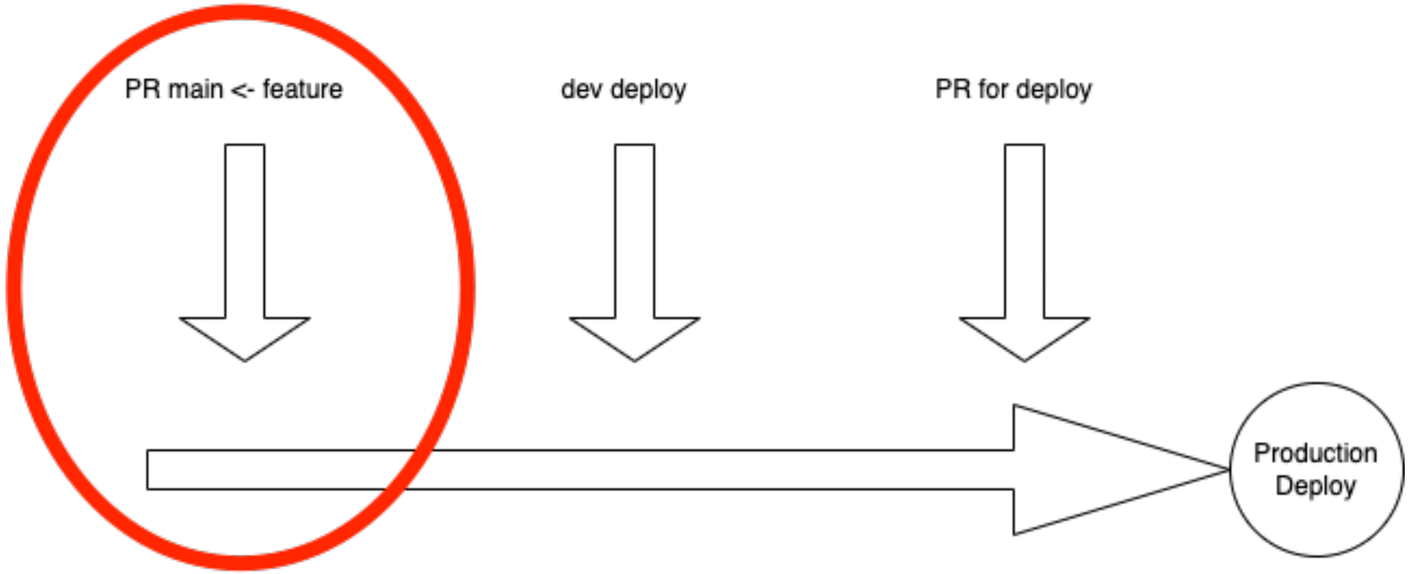
# テスト戦略概観

- 1スプリント2週間を基本とする開発サイクル内で4種のテストが存在
- **3つの自動テスト+手動テスト**
  - 1. Unit Test
  - 2. Integration Test
  - 3. End To End Test



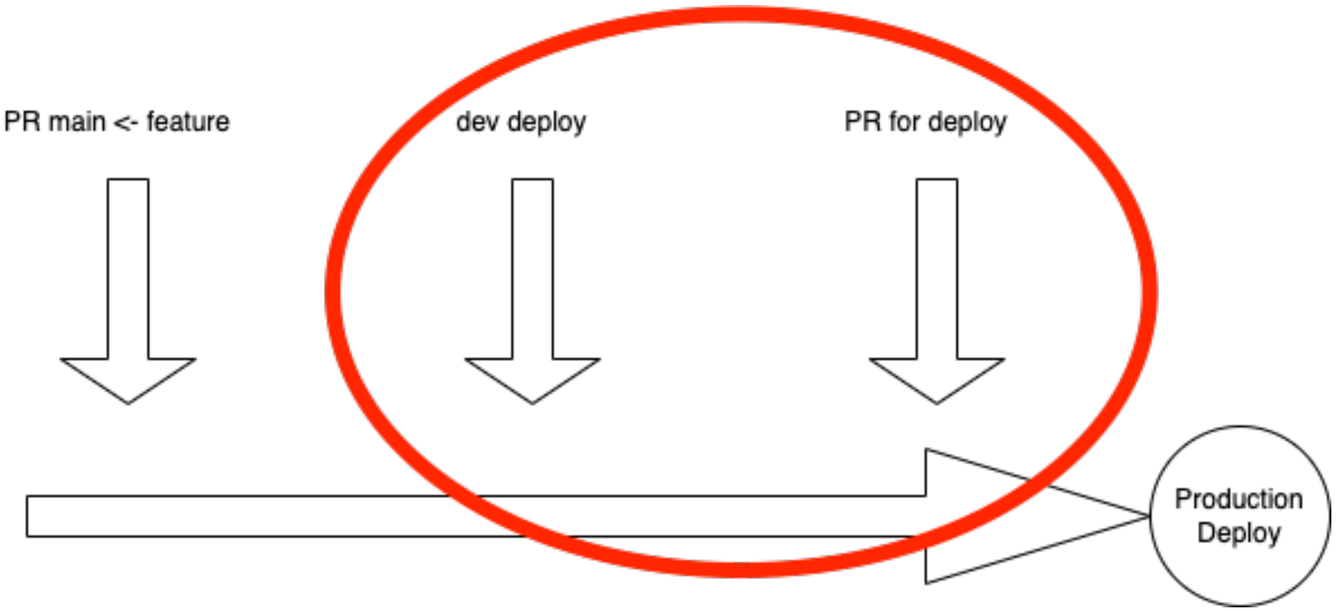
# テスト戦略概観

- 1スプリント2週間を基本とする開発サイクル内で4種のテストが存在
- 3つの自動テスト+手動テスト
  - **1. Unit Test**
  - 2. Integration Test
  - 3. End To End Test



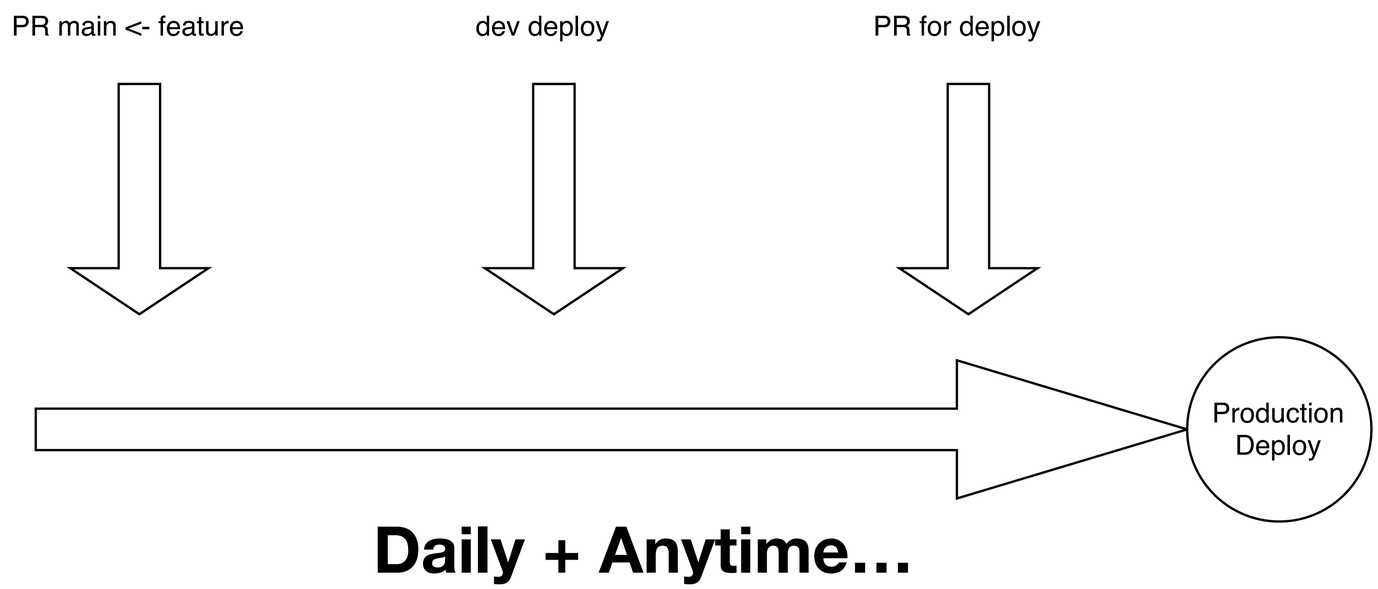
# テスト戦略概観

- 1スプリント2週間を基本とする開発サイクル内で4種のテストが存在
- 3つの自動テスト+手動テスト
  - 1. Unit Test
  - **2. Integration Test**
  - 3. End To End Test



# テスト戦略概観

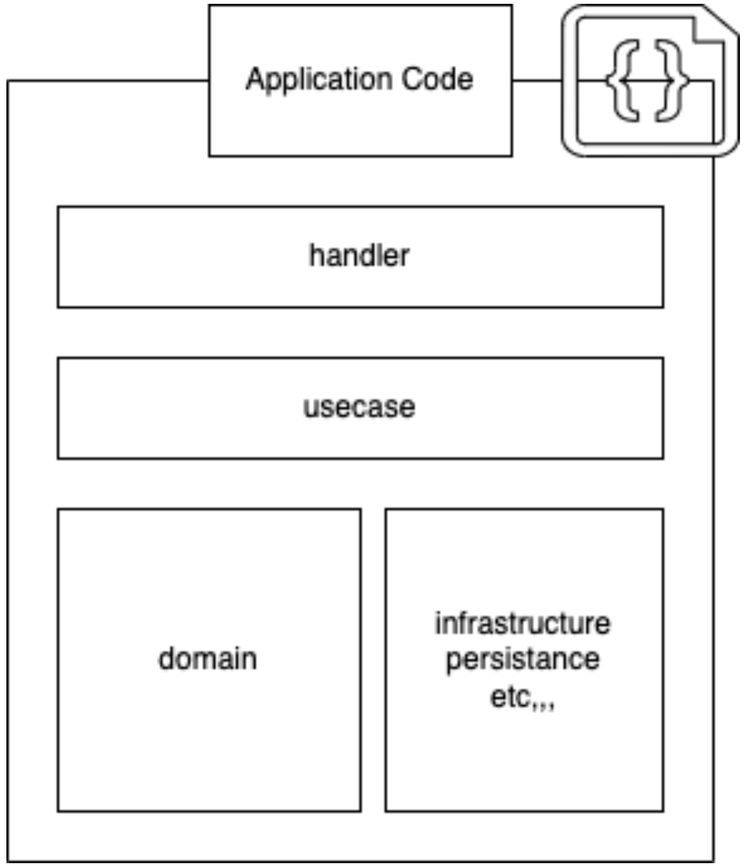
- 1スプリント2週間を基本とする開発サイクル内で4種のテストが存在
- 3つの自動テスト+手動テスト
  - 1. Unit Test
  - 2. Integration Test
  - **3. End To End Test**



# Unit Test🔧

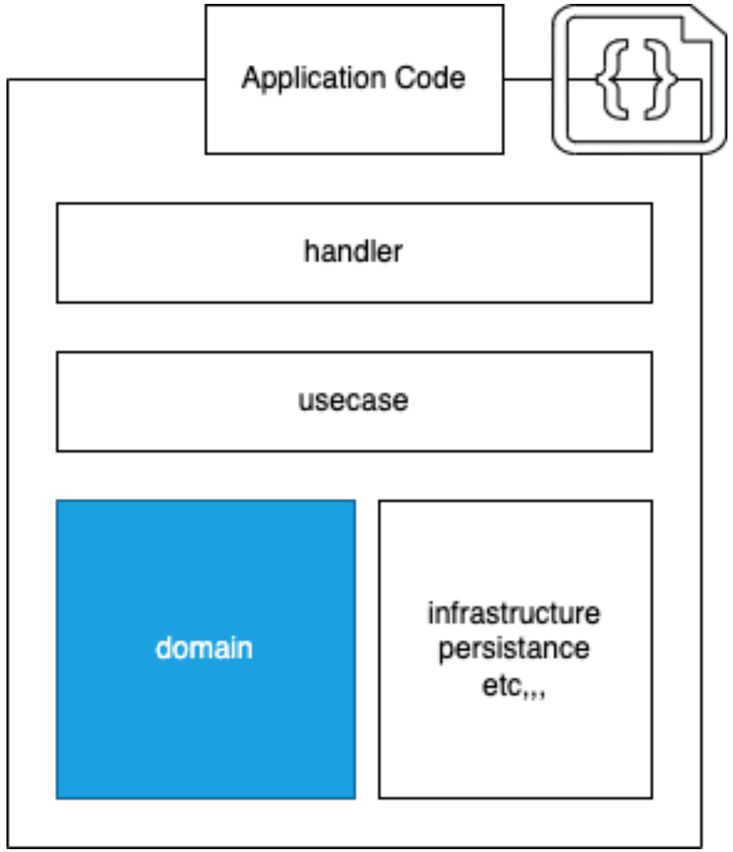
# Unit Test

- 関数、メソッドレベルでテストを作成



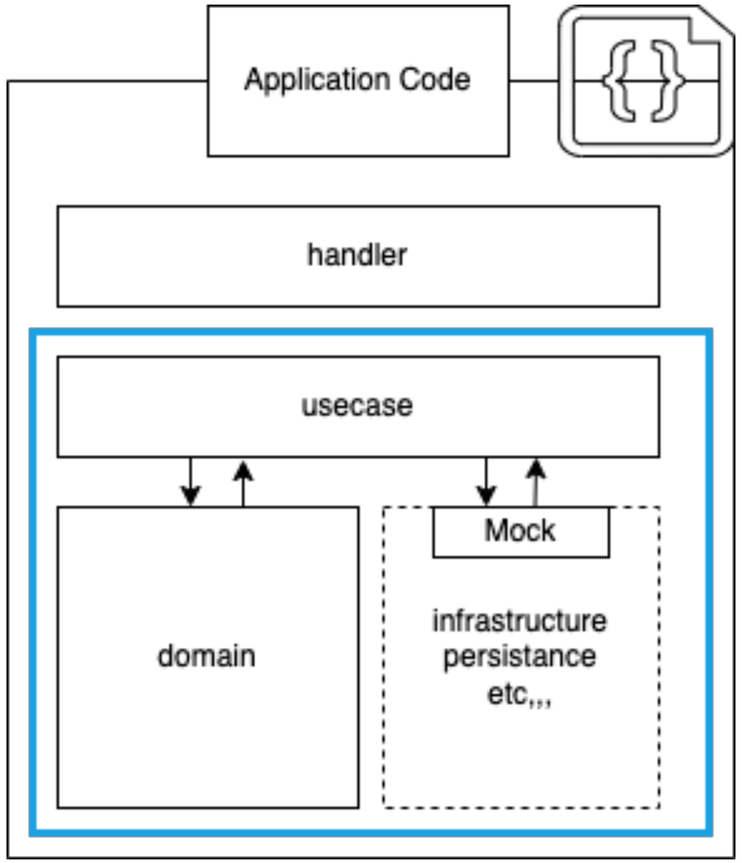
# Unit Test

- 関数、メソッドレベルでテストを作成
- **主に業務ロジック、ドメインオブジェクトが対象**



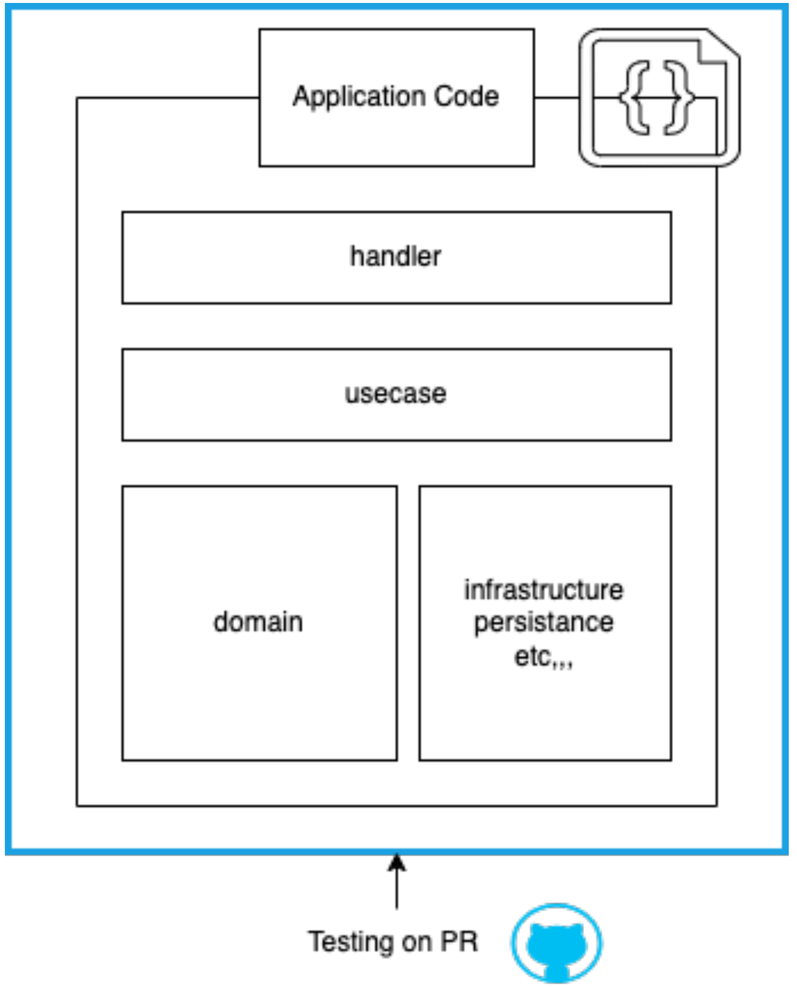
# Unit Test

- 関数、メソッドレベルでテストを作成
- 主に業務ロジック、ドメインオブジェクトが対象
- **Infra層をモックしてユースケース層をテストする場合もある**



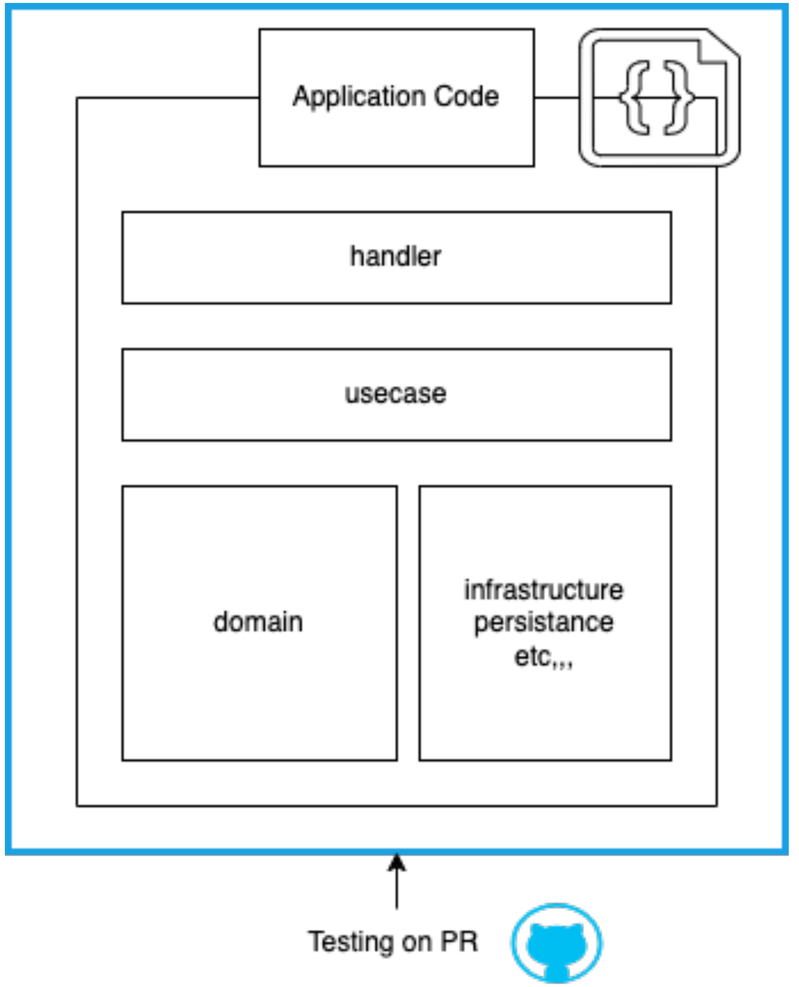
# Unit Test

- 関数、メソッドレベルでテストを作成
- 主に業務ロジック、ドメインオブジェクトが対象
- Infra層をモックしてユースケース層をテストする場合もある
- **PR上でUTは自動実行される(Github Actions)**



# Unit Test

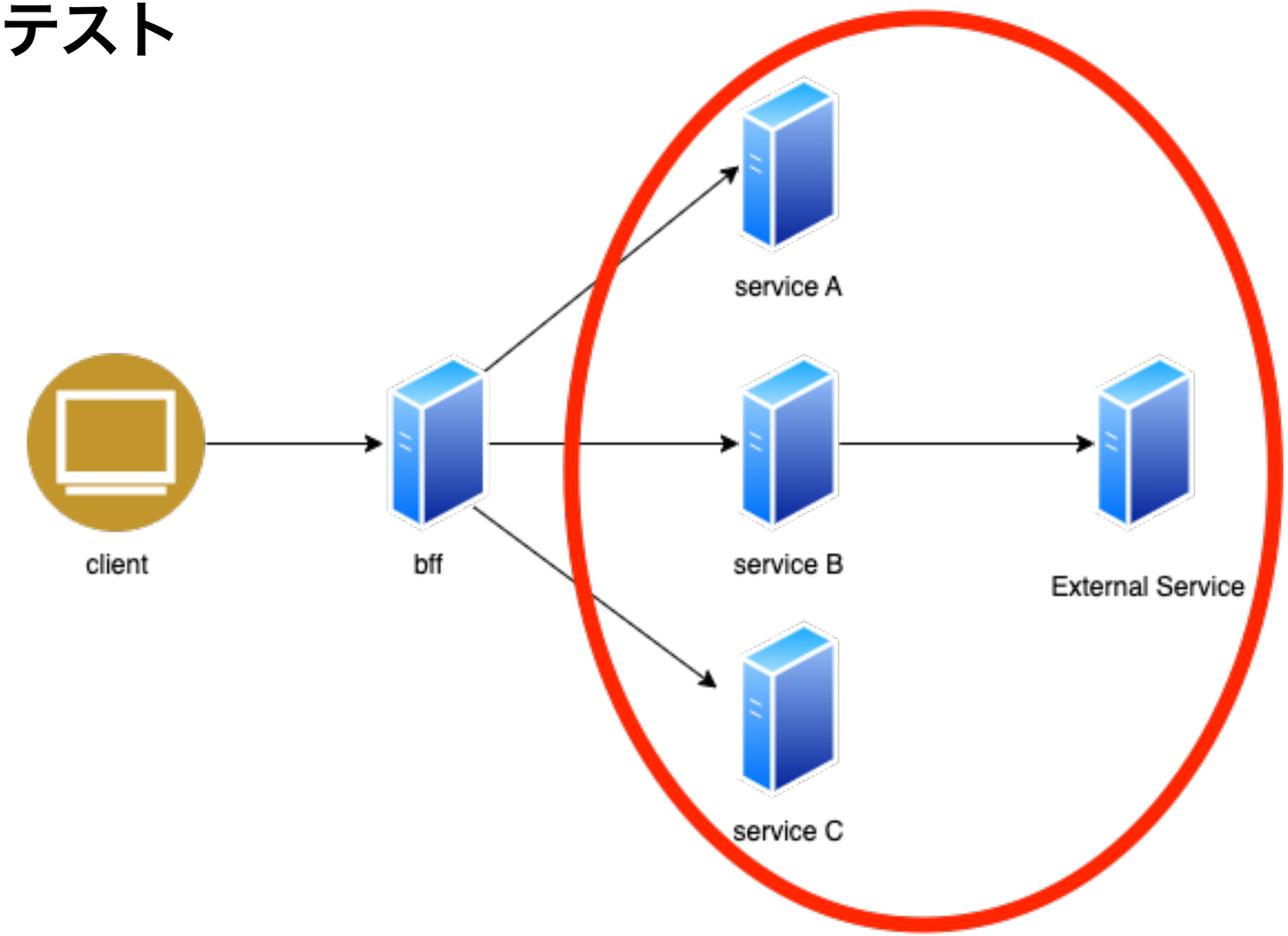
- 関数、メソッドレベルでテストを作成
- 主に業務ロジック、ドメインオブジェクトが対象
- Infra層をモックしてユースケース層をテストする場合もある
- PR上でUTは自動実行される(Github Actions)
  - **OKにならならないとマージできない**



# Integration Test 👨‍🔧

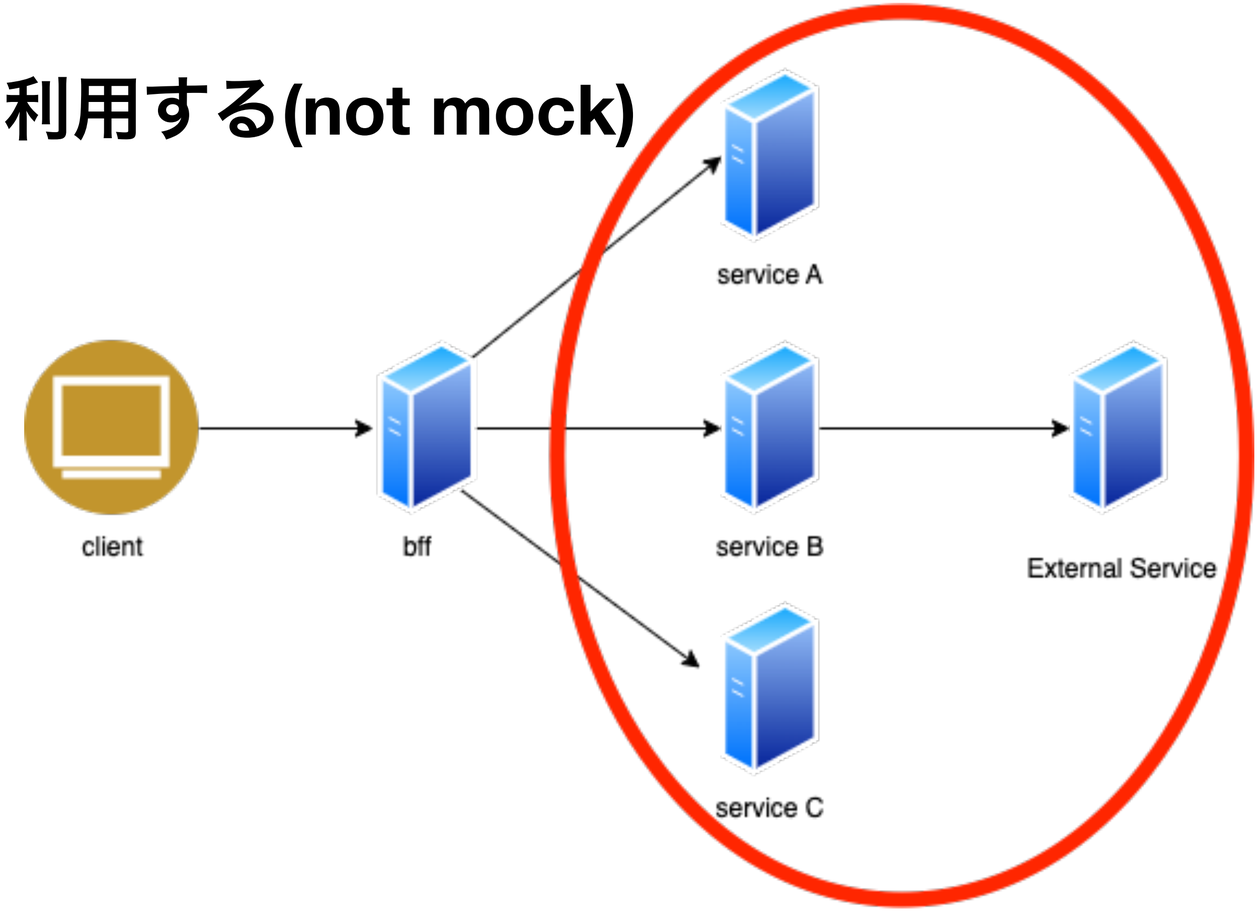
# Integration Test

- サーバーアプリ群を対象にテスト



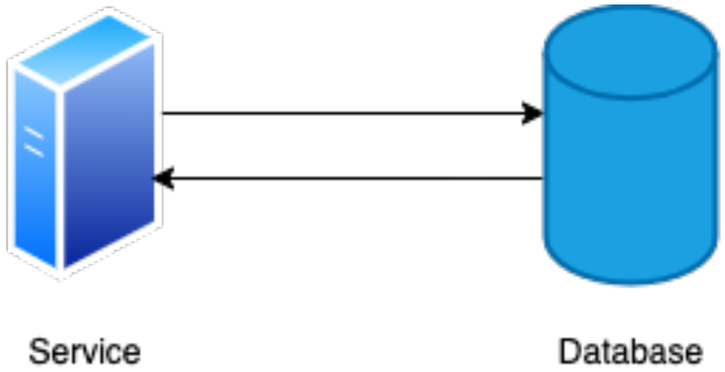
# Integration Test

- サーバーアプリ群を対象にテスト
- **アプリの依存はできるだけ本物を利用する(not mock)**



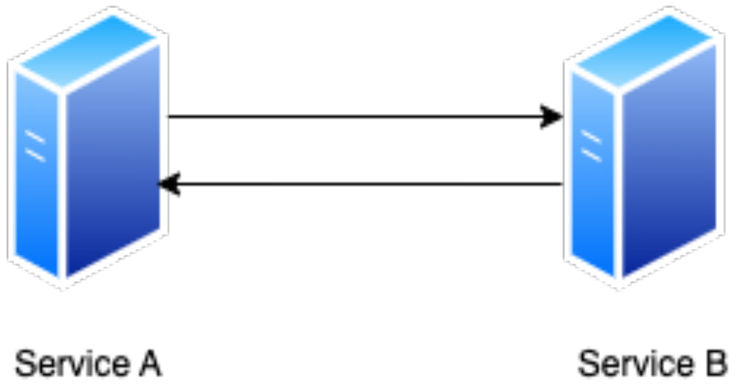
# Integration Test

- サーバーアプリ群を対象にテスト
- アプリの依存はできるだけ本物を利用する(not mock)
  - **DBとの連携**



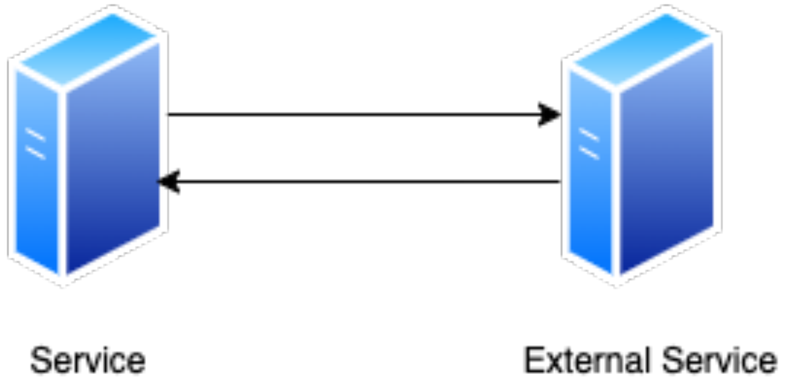
# Integration Test

- サーバーアプリ群を対象にテスト
- アプリの依存はできるだけ本物を利用する(not mock)
  - DBとの連携
  - **サーバーアプリの連携**



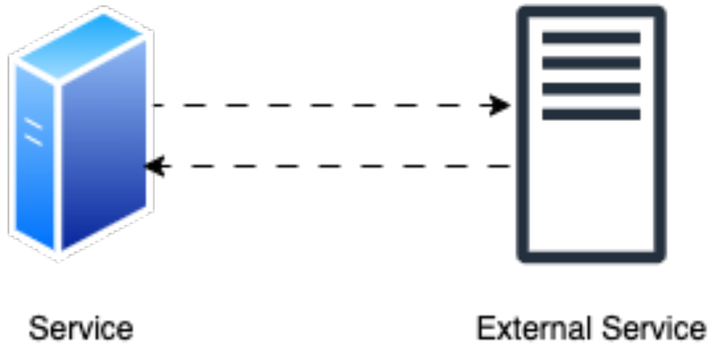
# Integration Test

- サーバーアプリ群を対象にテスト
- アプリの依存はできるだけ本物を利用する(not mock)
  - DBとの連携
  - サーバーアプリの連携
  - **外部サービスとの連携**



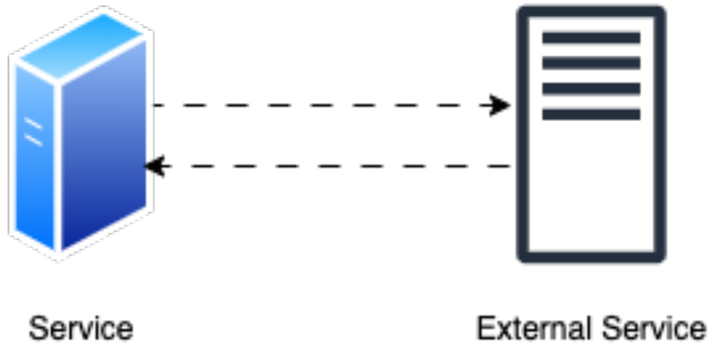
# Integration Test

- サーバーアプリ群を対象にテスト
- アプリの依存はできるだけ本物を利用する(not mock)
  - DBとの連携
  - サーバーアプリの連携
  - 外部サービスとの連携
    - **外部サービスはmockを利用する場合もある**



# Integration Test

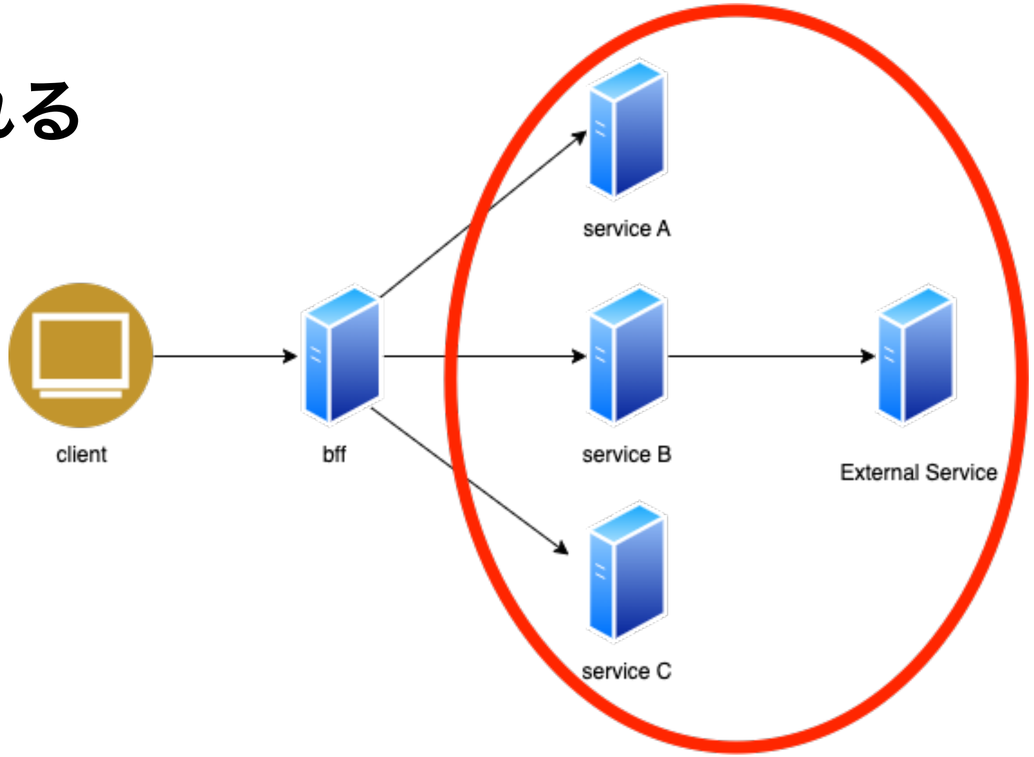
- サーバーアプリ群を対象にテスト
- アプリの依存はできるだけ本物を利用する(not mock)
  - DBとの連携
  - サーバーアプリの連携
  - 外部サービスとの連携
    - 外部サービスはmockを利用する場合もある




- **サービス間のコミュニケーションのルールが守られていることを検証**

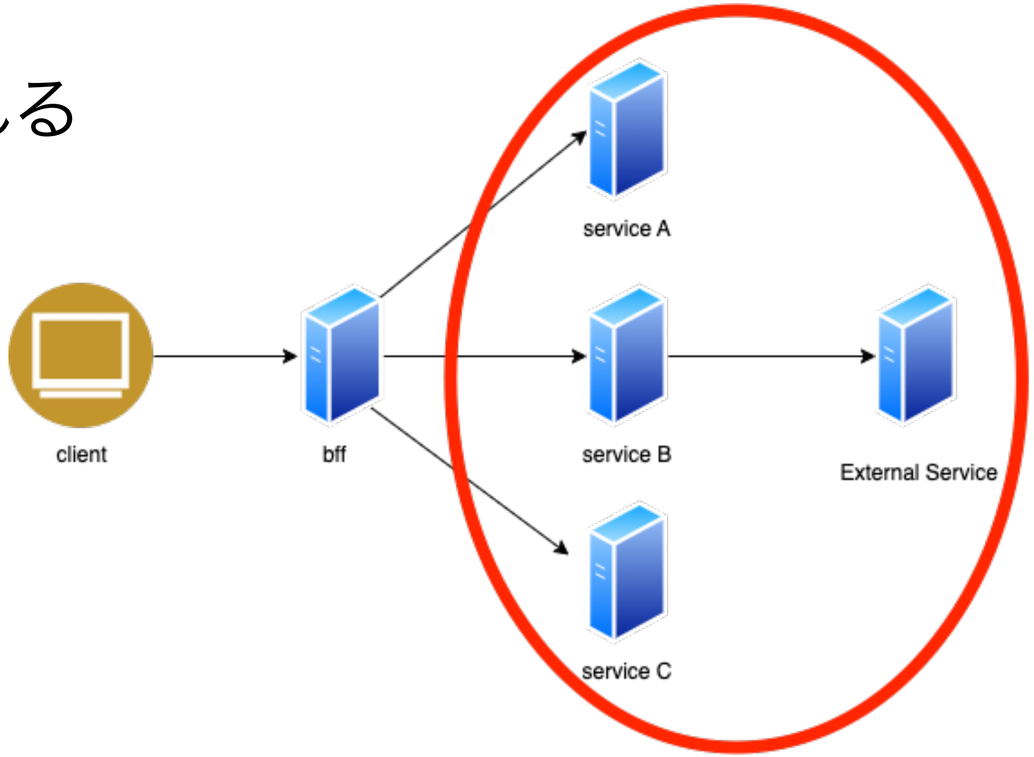
# Integration Test

- サーバーアプリ群を対象にテスト
- アプリの依存はできるだけ本物を利用する(not mock)
- **テストはPRがマージされる度に実行される**



# Integration Test

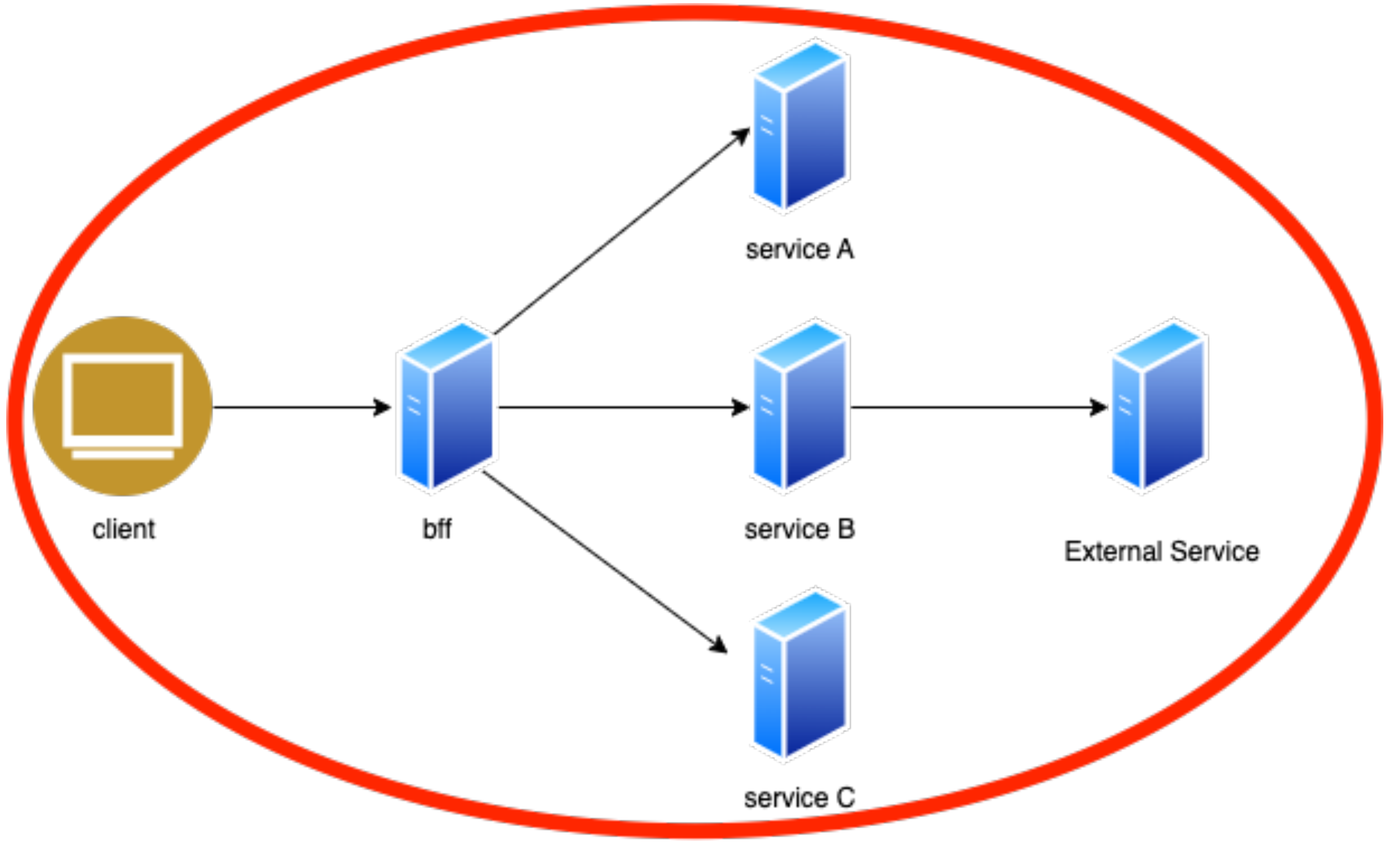
- サーバーアプリ群を対象にテスト
- アプリの依存はできるだけ本物を利用する(not mock)
- テストはPRがマージされる度に実行される
  - **開発環境上のソースが自動で更新され**
  - **テストもそのタイミングで実行**



# End to End Test 

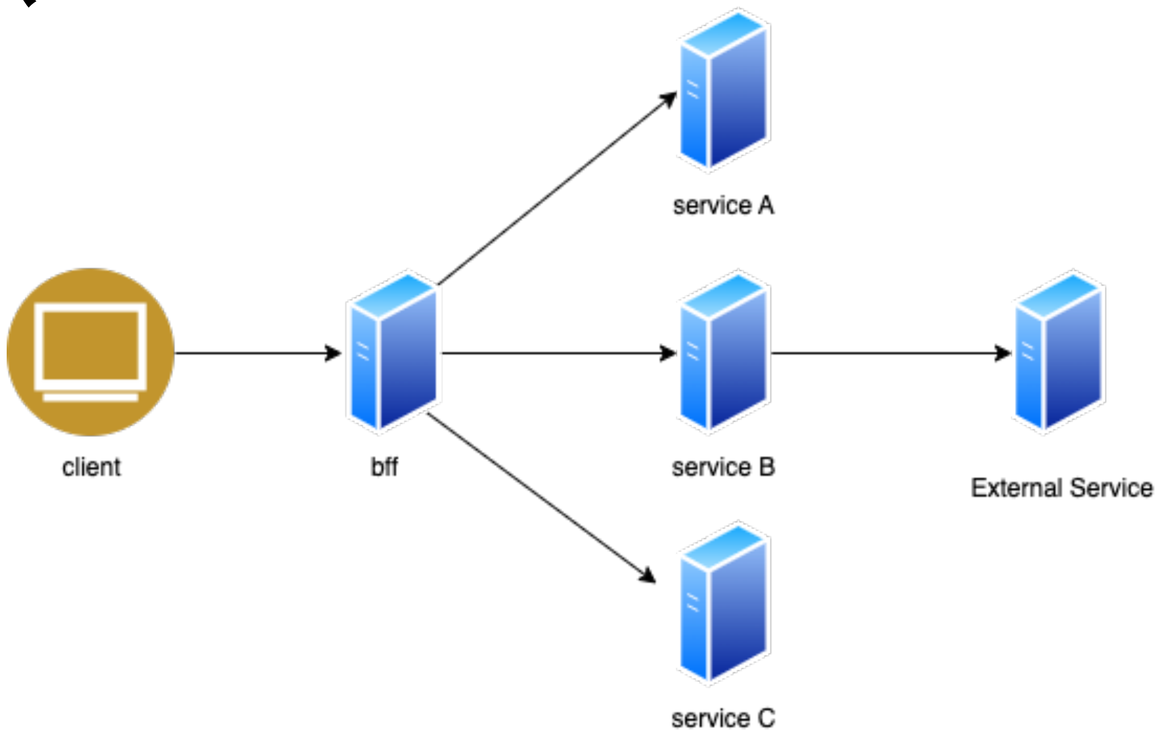
# End to End Test

- 画面を操作してシステム全体を一気通貫にテスト



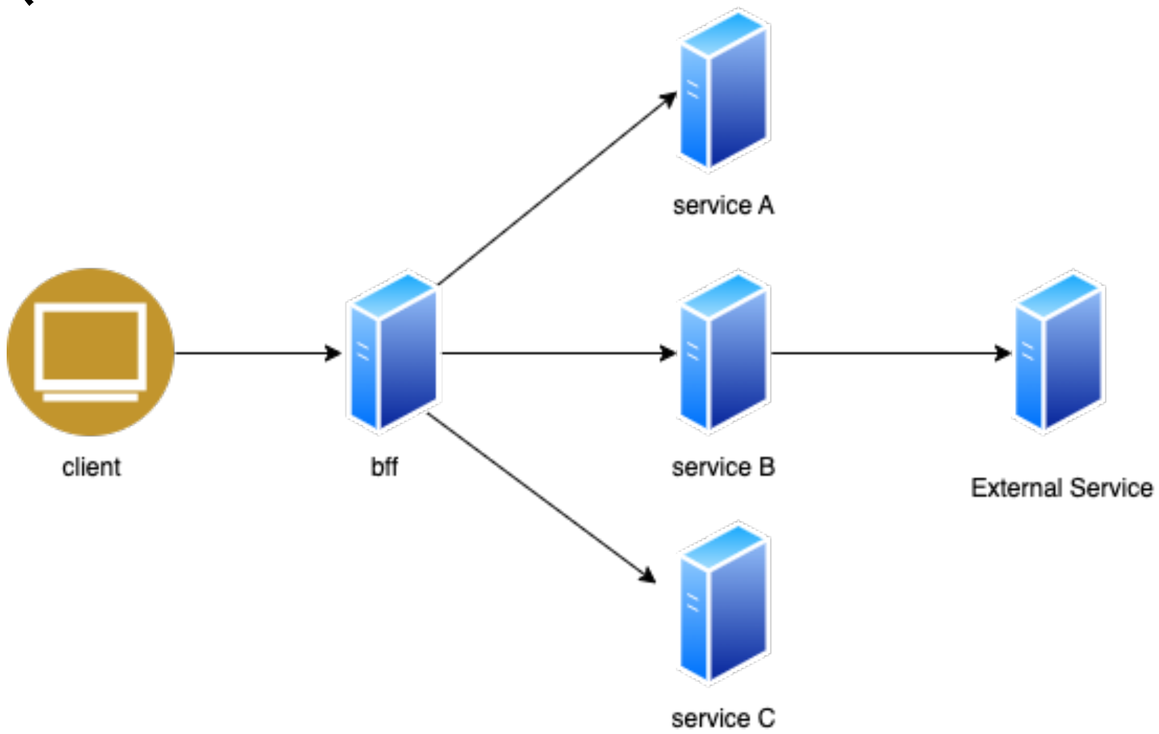
# End to End Test

- 画面を操作してシステム全体を一気通貫にテスト
- **ケース数を絞りつつ重要な機能をカバー**



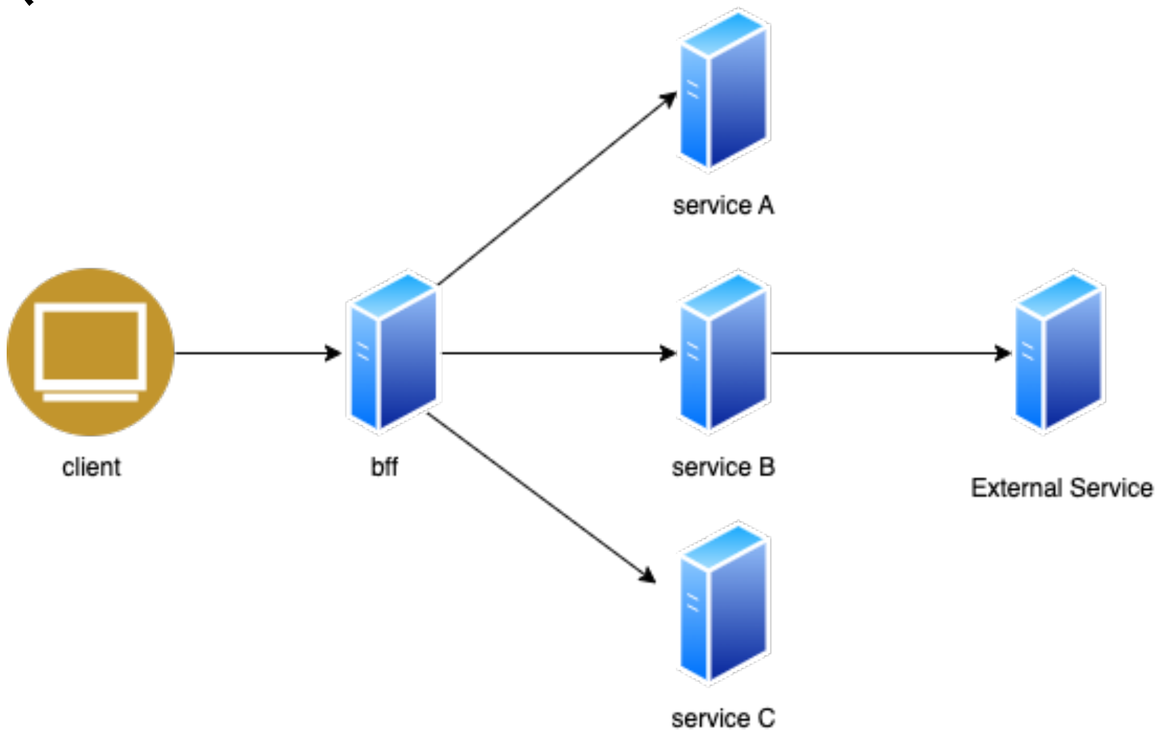
# End to End Test

- 画面を操作してシステム全体を一気通貫にテスト
- ケース数を絞りつつ重要な機能をカバー
- **日次での自動実行**



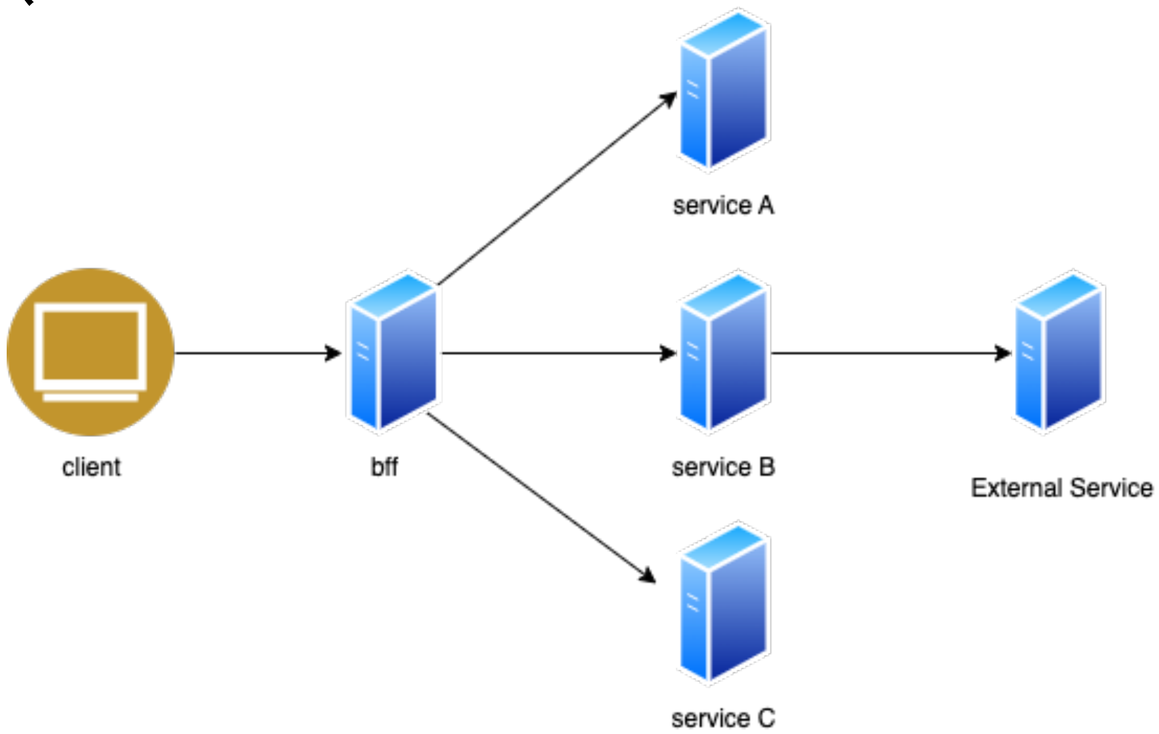
# End to End Test

- 画面を操作してシステム全体を一気通貫にテスト
- ケース数を絞りつつ重要な機能をカバー
- 日次での自動実行
  - **実行時間が長い**



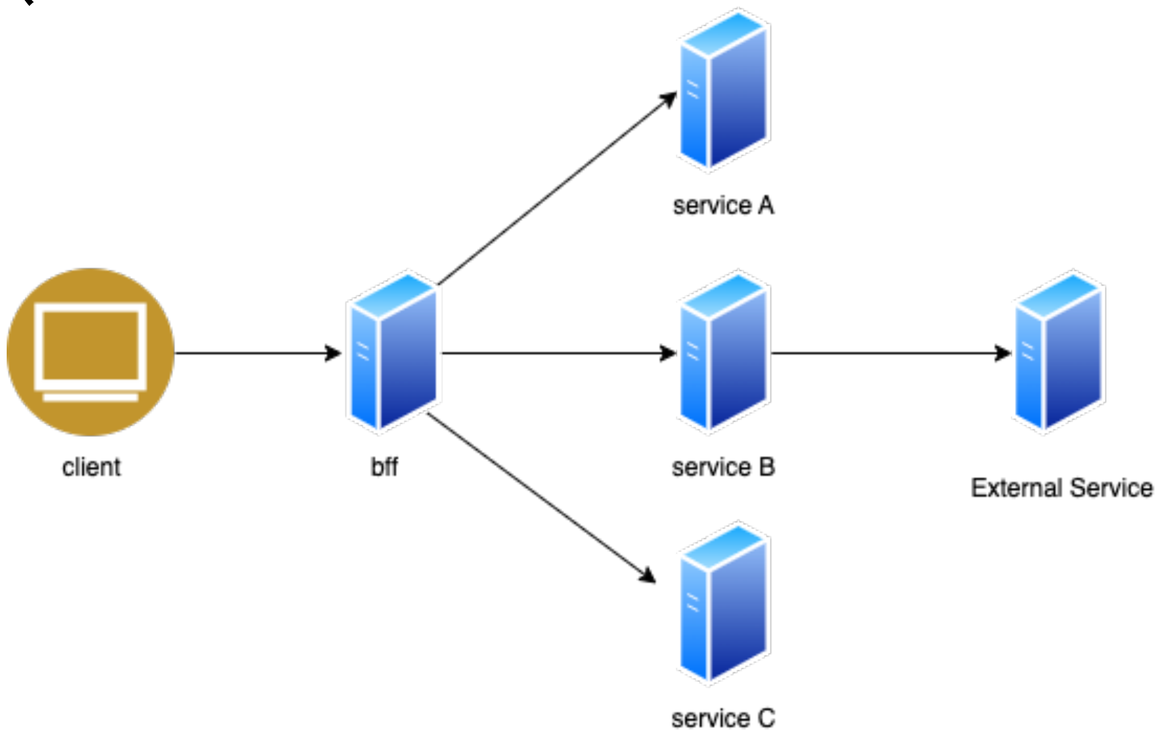
# End to End Test

- 画面を操作してシステム全体を一気通貫にテスト
- ケース数を絞りつつ重要な機能をカバー
- 日次での自動実行
  - 実行時間が長い
  - **Flakyという事情から**



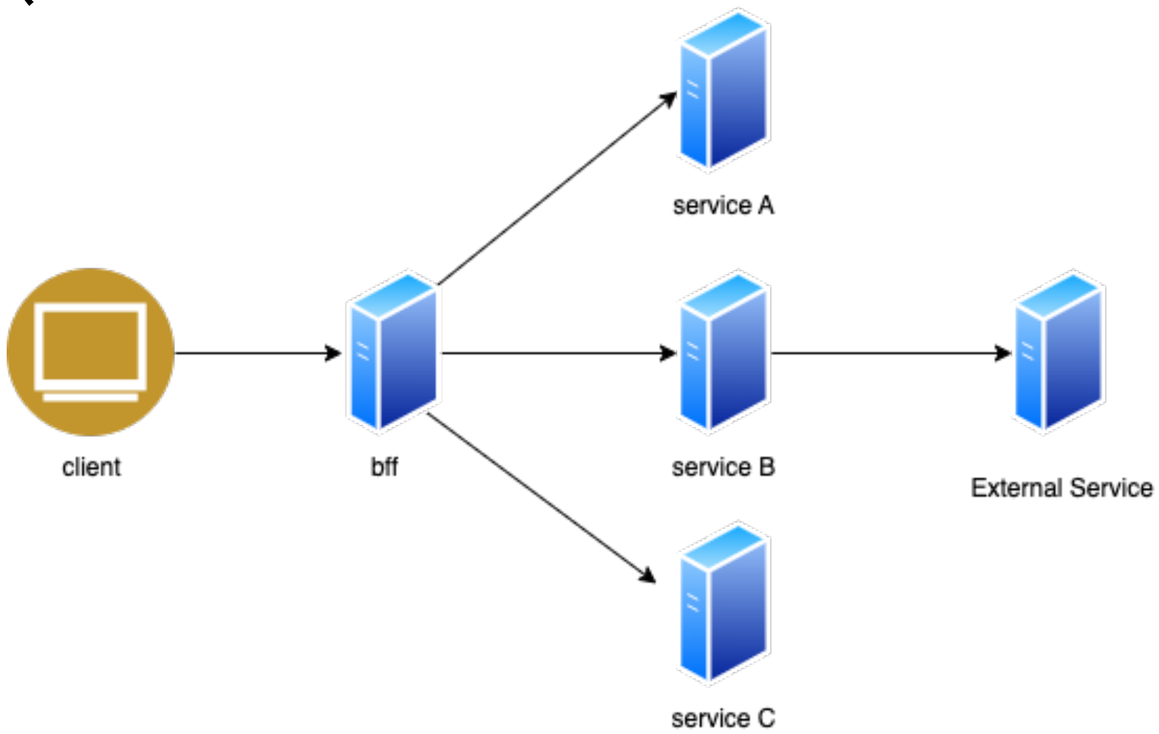
# End to End Test

- 画面を操作してシステム全体を一気通貫にテスト
- ケース数を絞りつつ重要な機能をカバー
- 日次での自動実行
- **最後の門番👮のイメージ**



# End to End Test

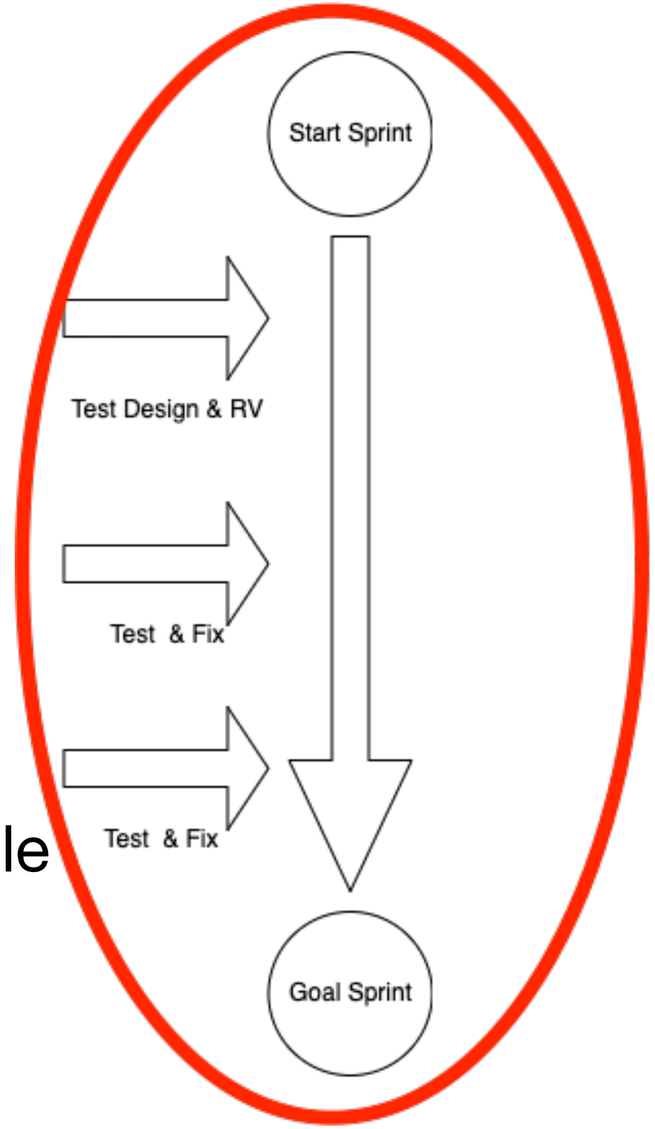
- 画面を操作してシステム全体を一気通貫にテスト
- ケース数を絞りつつ重要な機能をカバー
- 日次での自動実行
- 最後の門番👮のイメージ
  - **失敗している場合は緊急度が高い**



# Manual Test💪

# Manual Test

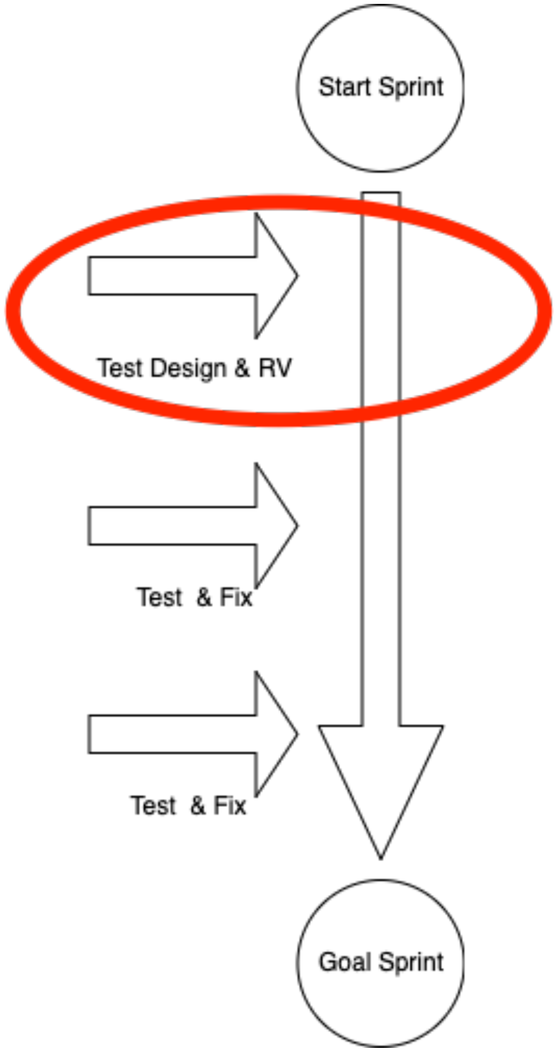
- 2週間を1スプリントとして開発サイクルを回す




2 weeks per cycle

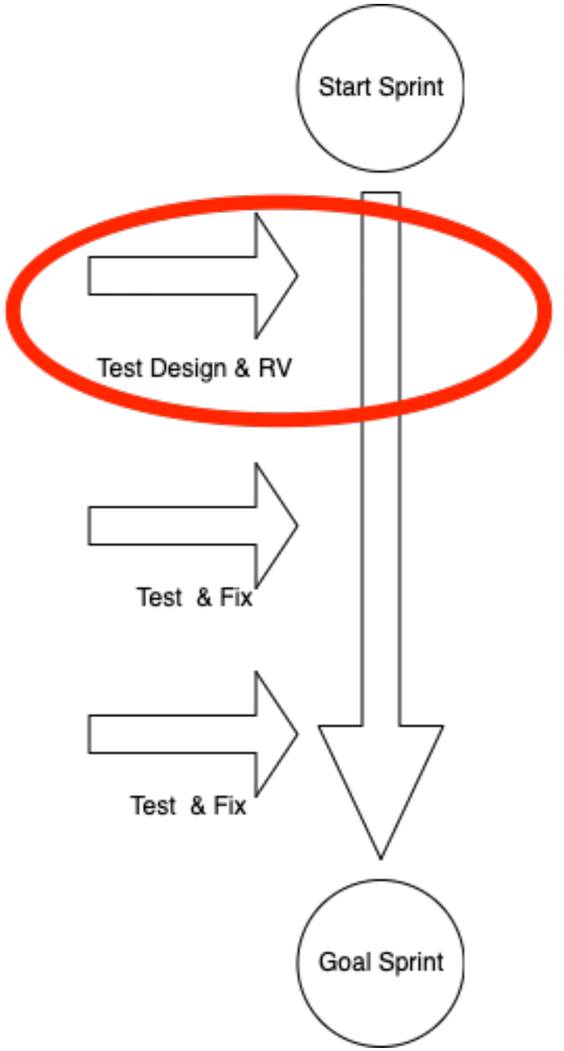
# Manual Test

- 2週間を1スプリントとして開発サイクルを回す
- **スプリント前半にテスト設計/RVを行う**



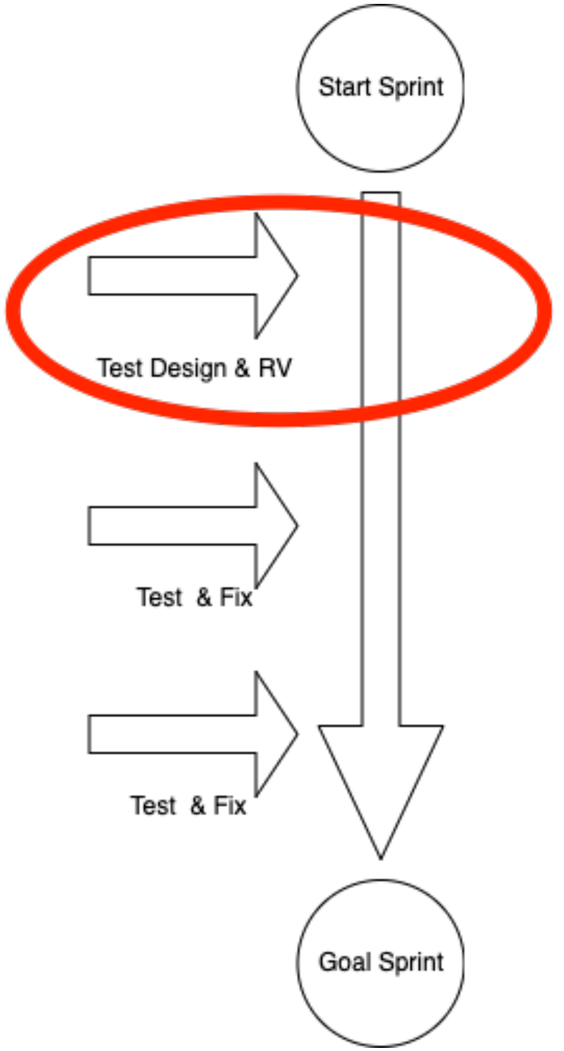
# Manual Test

- 2週間を1スプリントとして開発サイクルを回す
- スプリント前半にテスト設計/RVを行う
  - **テストはPBLごとに設計/実施**



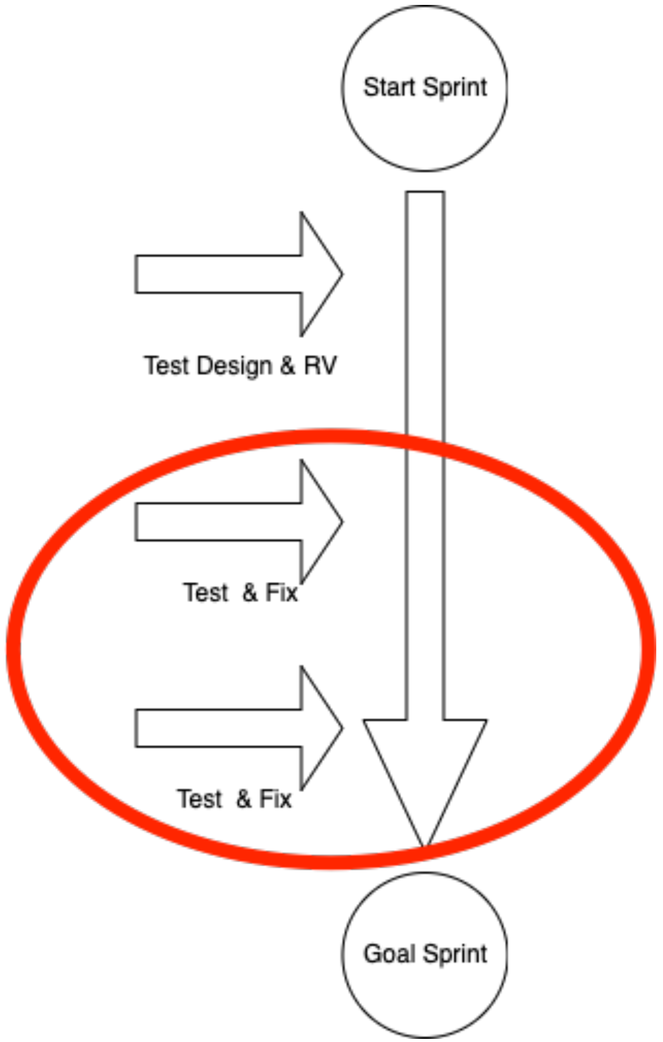
# Manual Test

- 2週間を1スプリントとして開発サイクルを回す
- スプリント前半にテスト設計/RVを行う
  - テストはPBLごとに設計/実施
  - **受入条件を明らかにして実装するため**



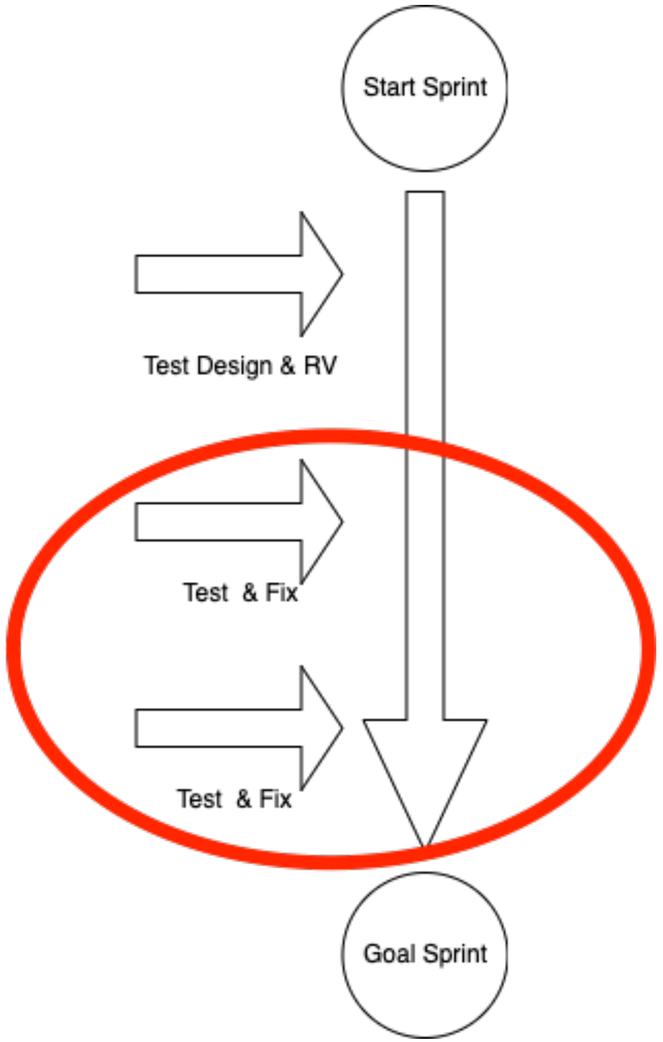
# Manual Test

- 2週間を1スプリントとして開発サイクルを回す
- スプリント前半にテスト設計/RVを行う
- **実装完了した部分から順々にテスト実施/修正**



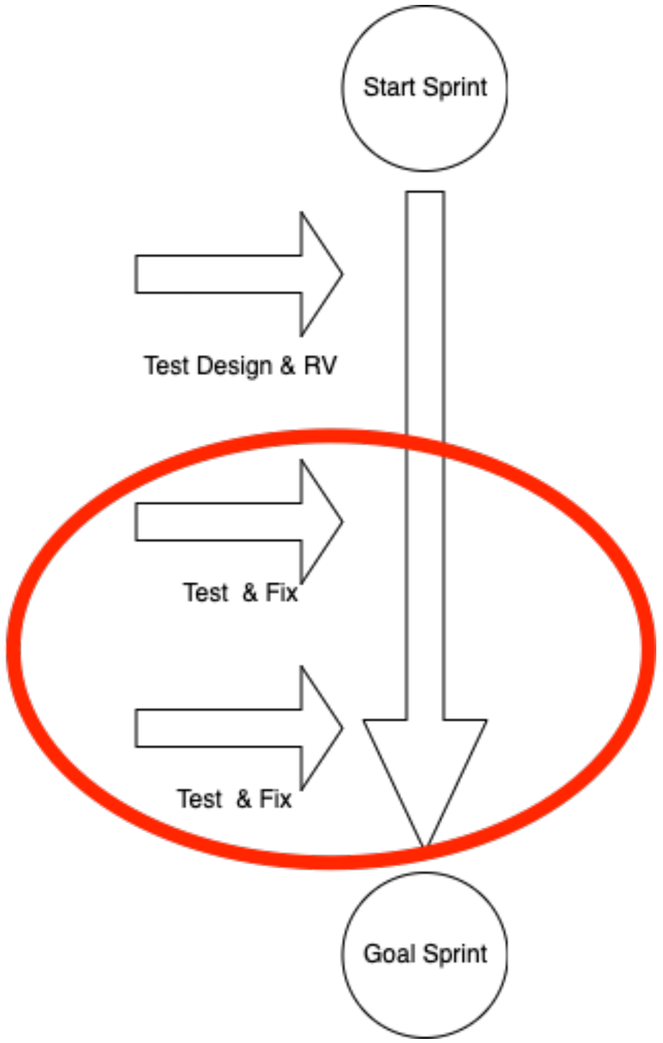
# Manual Test

- 2週間を1スプリントとして開発サイクルを回す
- スプリント前半にテスト設計/RVを行う
- 実装完了した部分から順々にテスト実施/修正
  - **デプロイは都度行っておりテスト実施は待たない**



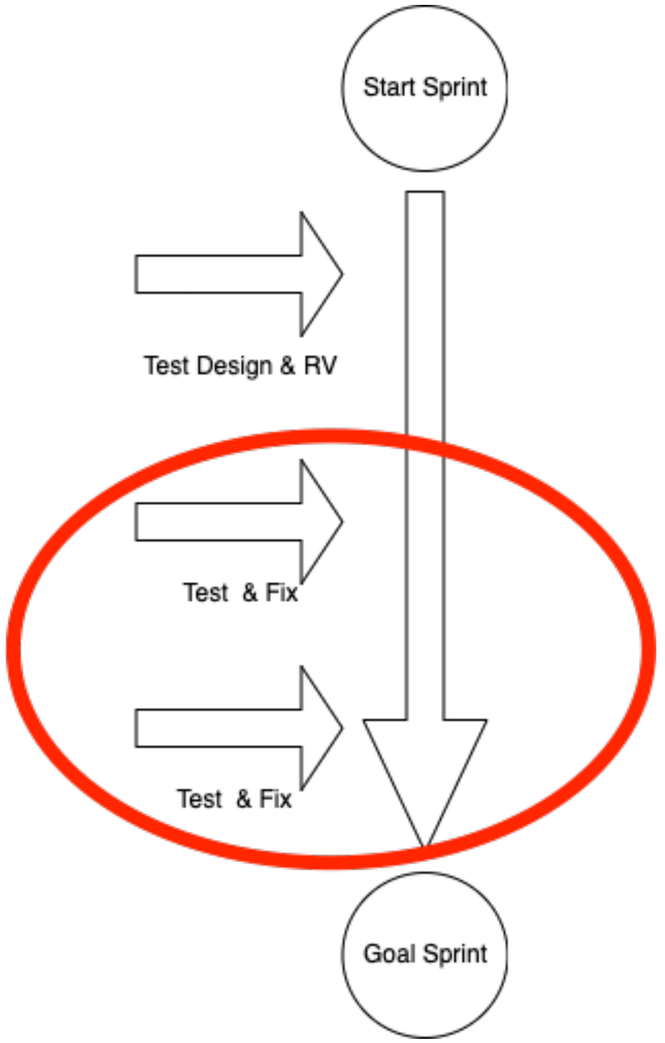
# Manual Test

- 2週間を1スプリントとして開発サイクルを回す
- スプリント前半にテスト設計/RVを行う
- 実装完了した部分から順々にテスト実施/修正
  - デプロイは都度行っておりテスト実施は待たない
    - **既存に影響が出ないように実装**



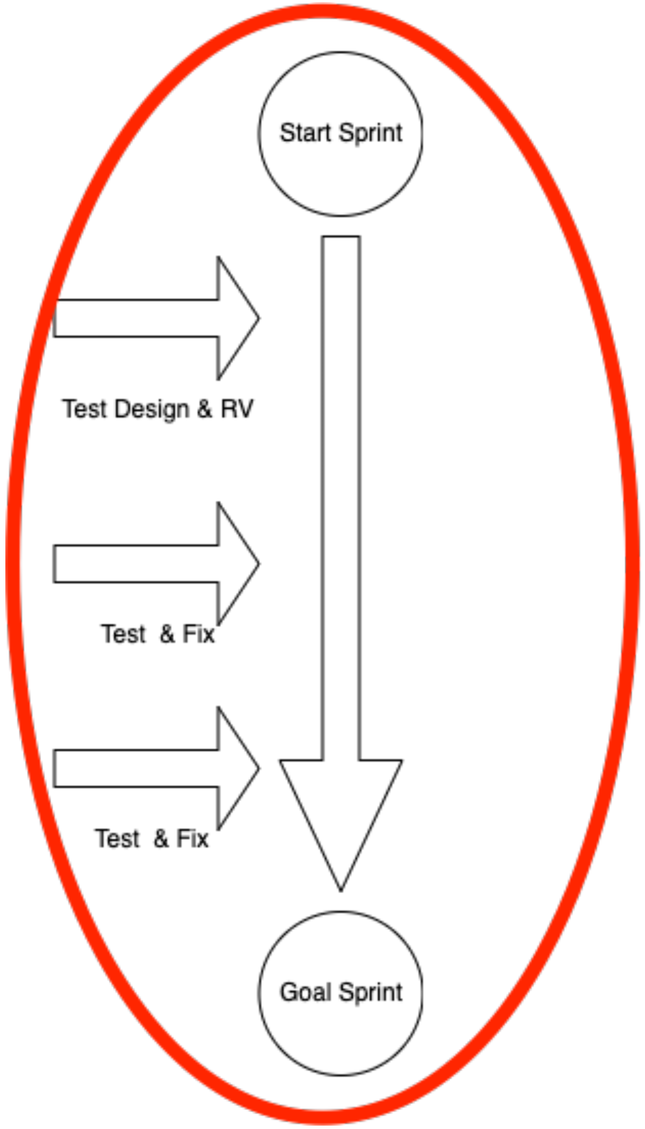
# Manual Test

- 2週間を1スプリントとして開発サイクルを回す
- スプリント前半にテスト設計/RVを行う
- 実装完了した部分から順々にテスト実施/修正
  - デプロイは都度行っておりテスト実施は待たない
    - 既存に影響が出ないように実装
    - **リグレッションテストは自動テストで賄う**



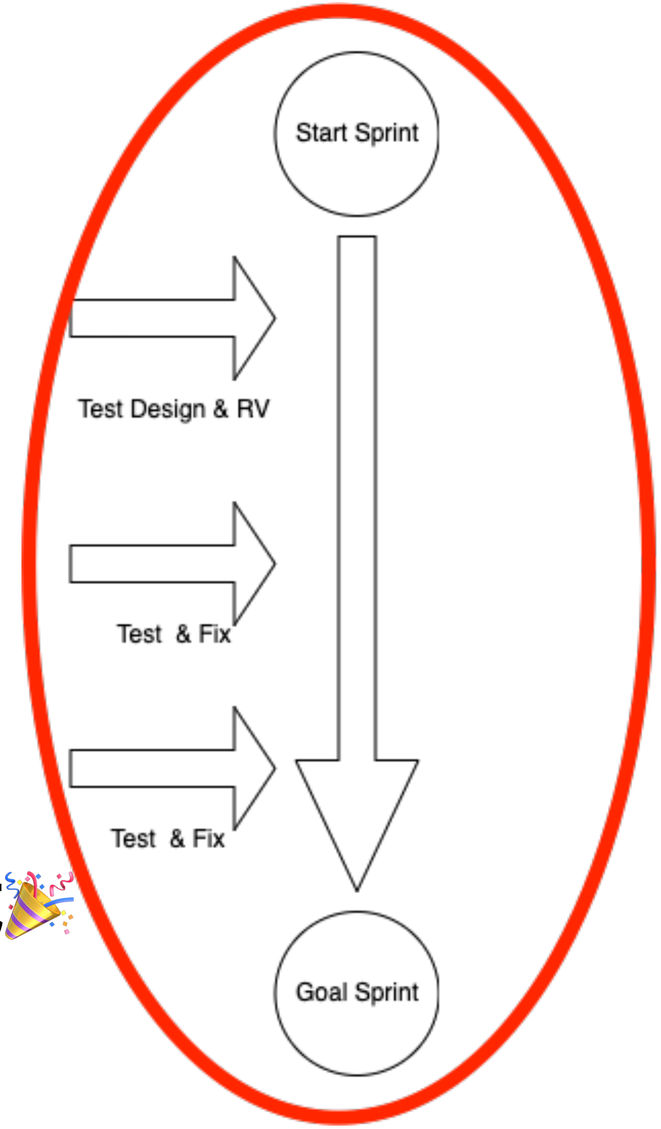
# Manual Test

- 2週間を1スプリントとして開発サイクルを回す
- スプリント前半にテスト設計/RVを行う
- 実装完了した部分から順々にテスト実施/修正
- **スプリント期間中に全てのテスト結果OKを目指す**



# Manual Test

- 2週間を1スプリントとして開発サイクルを回す
- スプリント前半にテスト設計/RVを行う
- 実装完了した部分から順々にテスト実施/修正
- スプリント期間中に全てのテスト結果OKを目指す
  - **テストALL OK &POの承認でスプリントゴール達成**🎉



# まとめ👀

# まとめ

- 3つの自動テスト と手動テストを組み合わせた、テスト戦略を紹介しました
- 既存機能へ影響がでないことの担保は自動テストで行いつつ、新機能開発については手動テストを行います
- テスト戦略とそれを可能にするインフラ基盤の組み合わせが、高頻度デプロイの要素のひとつとなっています