

Infrastructure as Code自身 のテストとは何か？

TIS株式会社 IT基盤技術推進部
斎藤辰徳

Agenda

- Infrastructure as Codeとは
- Infrastructure as Codeのテスト
- ライブコーディング
- まとめ

自己紹介

- ・ 斎藤辰徳(サイトウ・タツノリ)
 - ・ TIS株式会社 IT基盤技術推進部 所属
 - ・ ミッション：**ITインフラ構築・運用・保守の自動化推進**
 - ・ HN：**bbrfkr(ビビリフクロウ)**で活動中！
 - ・ 興味・関心

Infrastructure as Code



Itamae



ANSIBLE



SERVERSPEC

仮想化・クラウド・コンテナ

oVirt



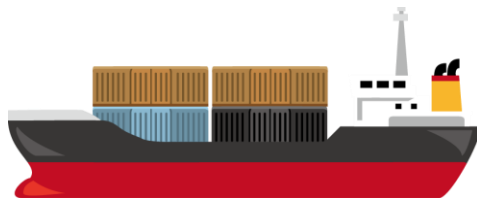
docker



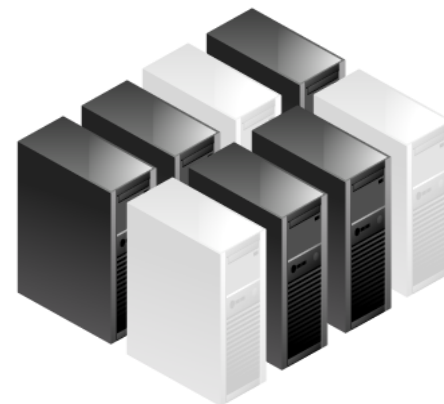
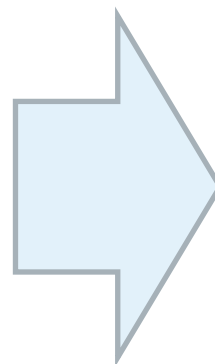
openstack®



Infrastructure as Codeとは



クラウド/コンテナ
技術の流行



ITインフラの
頻繁な作成・削除

手作業では構成管理が追い付かない！！



Infrastructure as Code(IaC)の登場

- ITインフラを**コード化**



- アプリケーションのプラクティス**をITインフラに適用



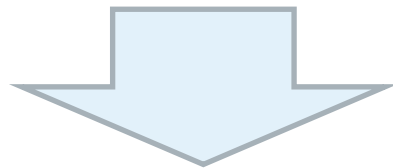
git



Jenkins

- IaCの実施

- 構成管理作業の自動化
- 構成情報のバージョン管理
- 構成情報の継続的インテグレーション



ITインフラ構築・運用・保守の
品質生産性向上

クラウド/コンテナ時代に

IaCは必須！



- 構成管理ツール



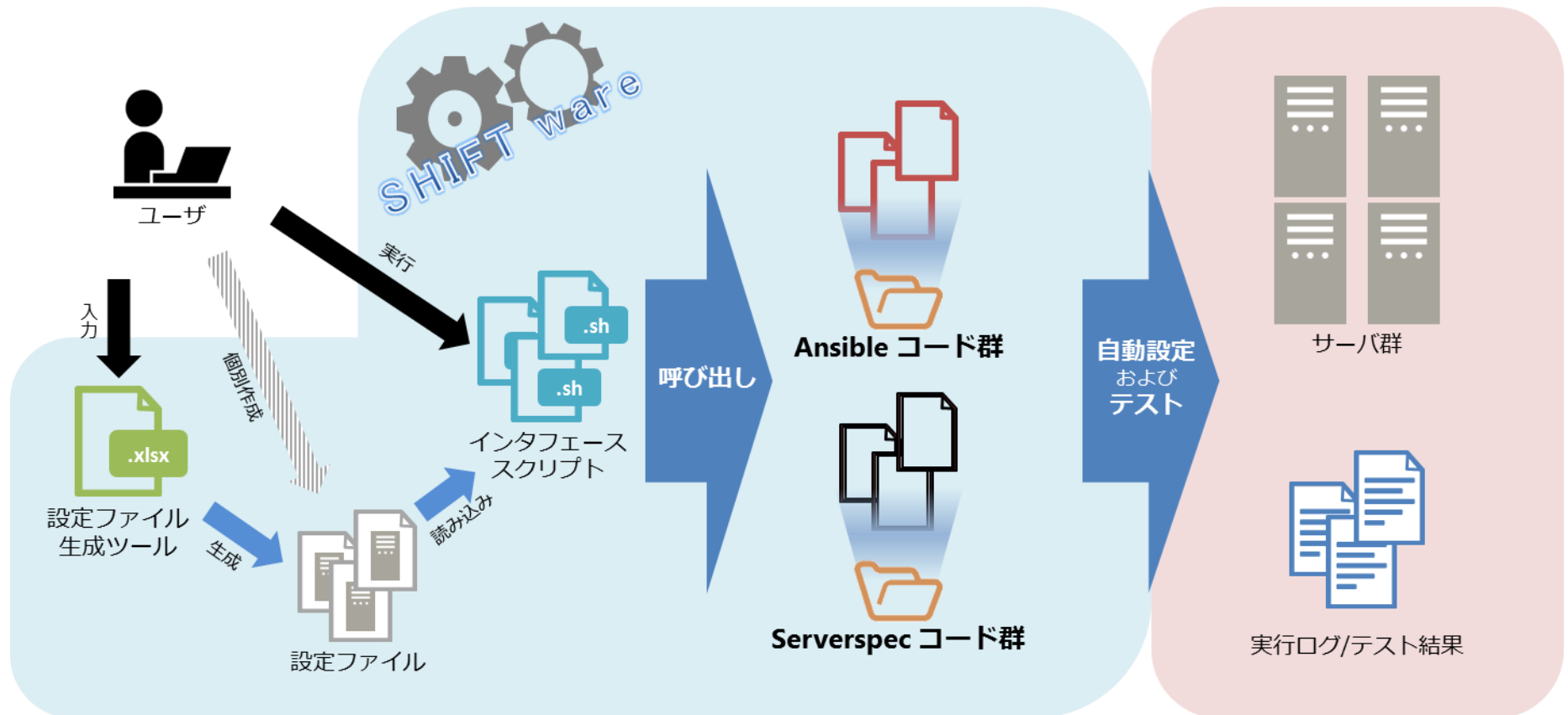
- インフラテスト自動化ツール



Infrataster

• SHIFT ware

- Ansible/Serverspecを用いた、**IaCフレームワーク**
- Ansible/Serverspecに**詳しくなくても利用可能**

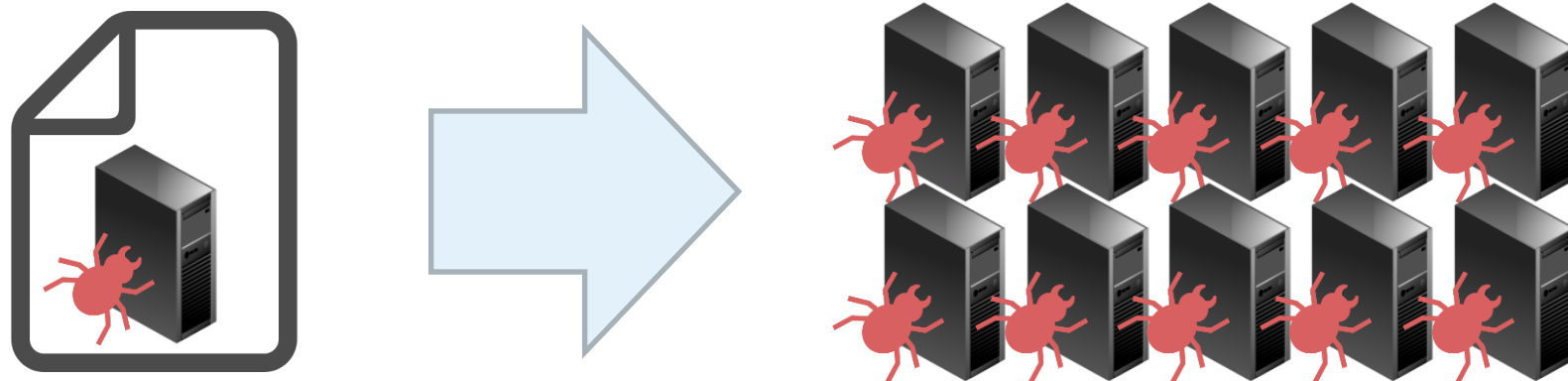


IaCフレームワーク
SHIFT ware

今秋OSS公開予定
coming soon...

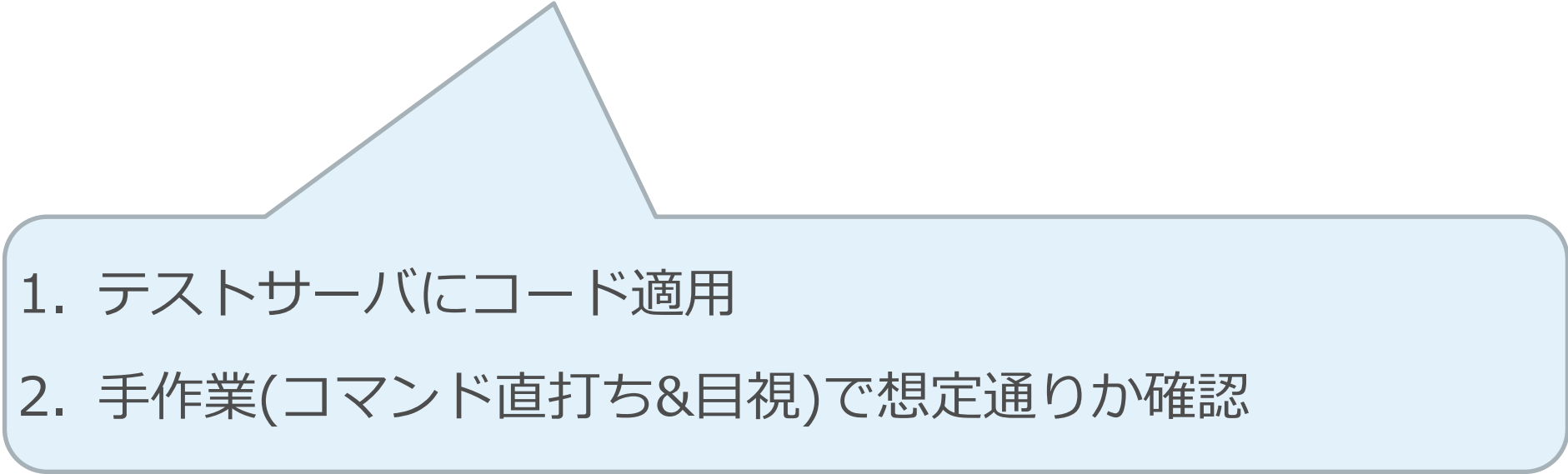
Infrastructure as Codeのテスト

- 一つのミスの**影響範囲が大きい**



コードの**テストが非常に重要**

まさか、**手動テスト**してませんか…？

- 
1. テストサーバにコード適用
 2. 手作業(コマンド直打ち&目視)で想定通りか確認

手動テストの恐ろしさ

- コードを変更するたびに行う回帰テスト

手動テストの恐ろしさ

- コードを変更するたびに行う回帰テスト
⇒やってられない



手動テストの恐ろしさ

- コードを変更するたびに行う回帰テスト
⇒やってられない
- その時その時の目的が達成されたかだけ確認する



手動テストの恐ろしさ

- コードを変更するたびに行う回帰テスト
⇒やってられない
- その時その時の目的が達成されたかだけ確認する
- 後で、以前できたことができなくなっていることに気付く



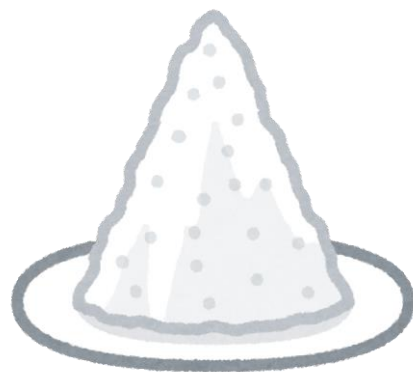
手動テストの恐ろしさ

- コードを変更するたびに行う回帰テスト
⇒やっつけられない
- その時その時の目的が達成されたかだけ確認する
- 後で、以前できたことができなくなっていることに気付く
- **動くコードの修正に臆病になっていく**



手動テストの恐ろしさ

- コードを変更するたびに行う回帰テスト
⇒やってられない
- その時その時の目的が達成されたかだけ確認する
- 後で、以前できたことができなくなっていることに気付く
- **動くコードの修正に臆病になっていく**



コードが塩漬け状態に！

リファクタリングもできず **保守性は最悪**…

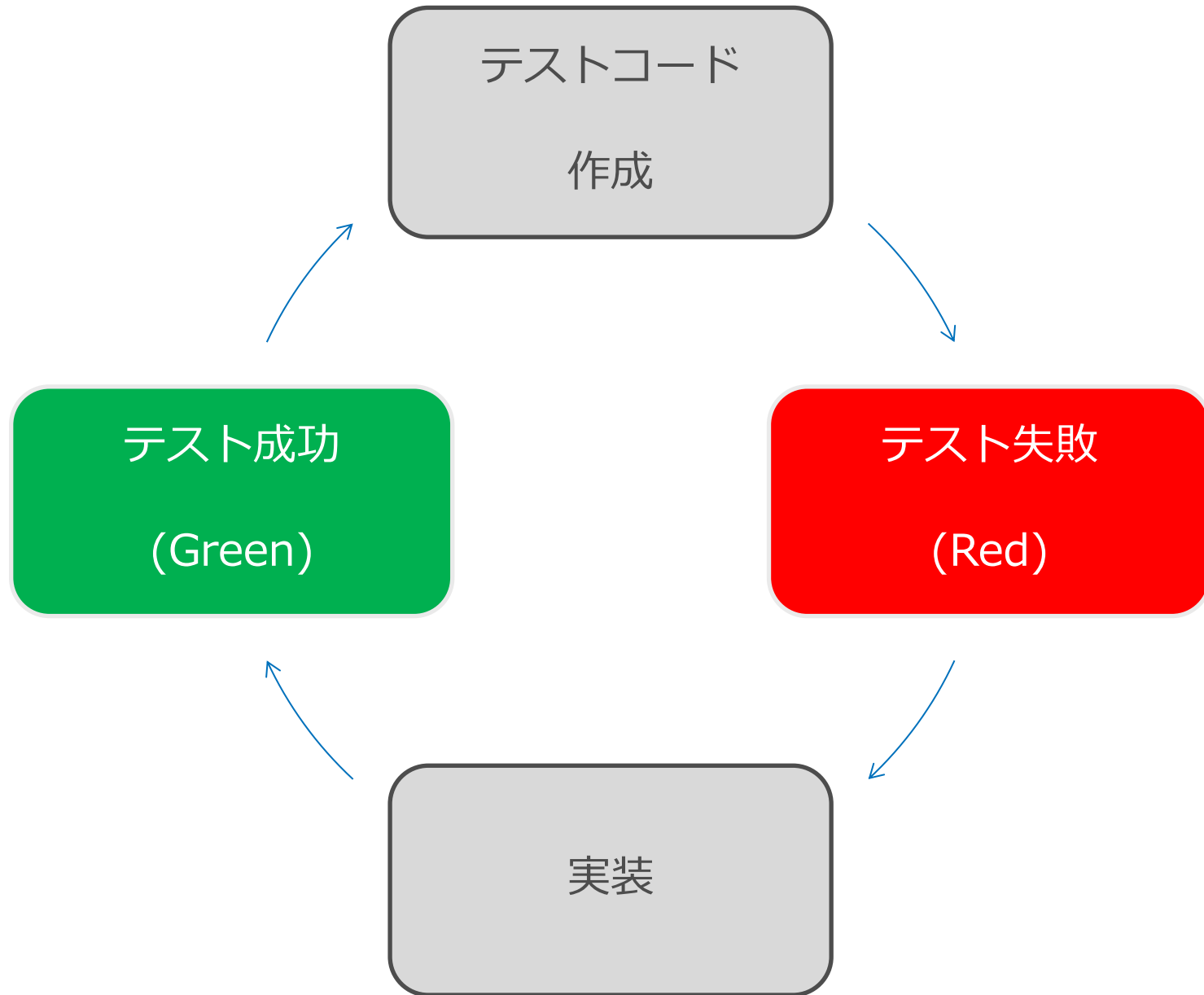
自動テストしましょう！

でも正直
動くコードができているのに
テストコード書くなんて面倒…



開発にテストコード作成を組み込む
テスト駆動開発！

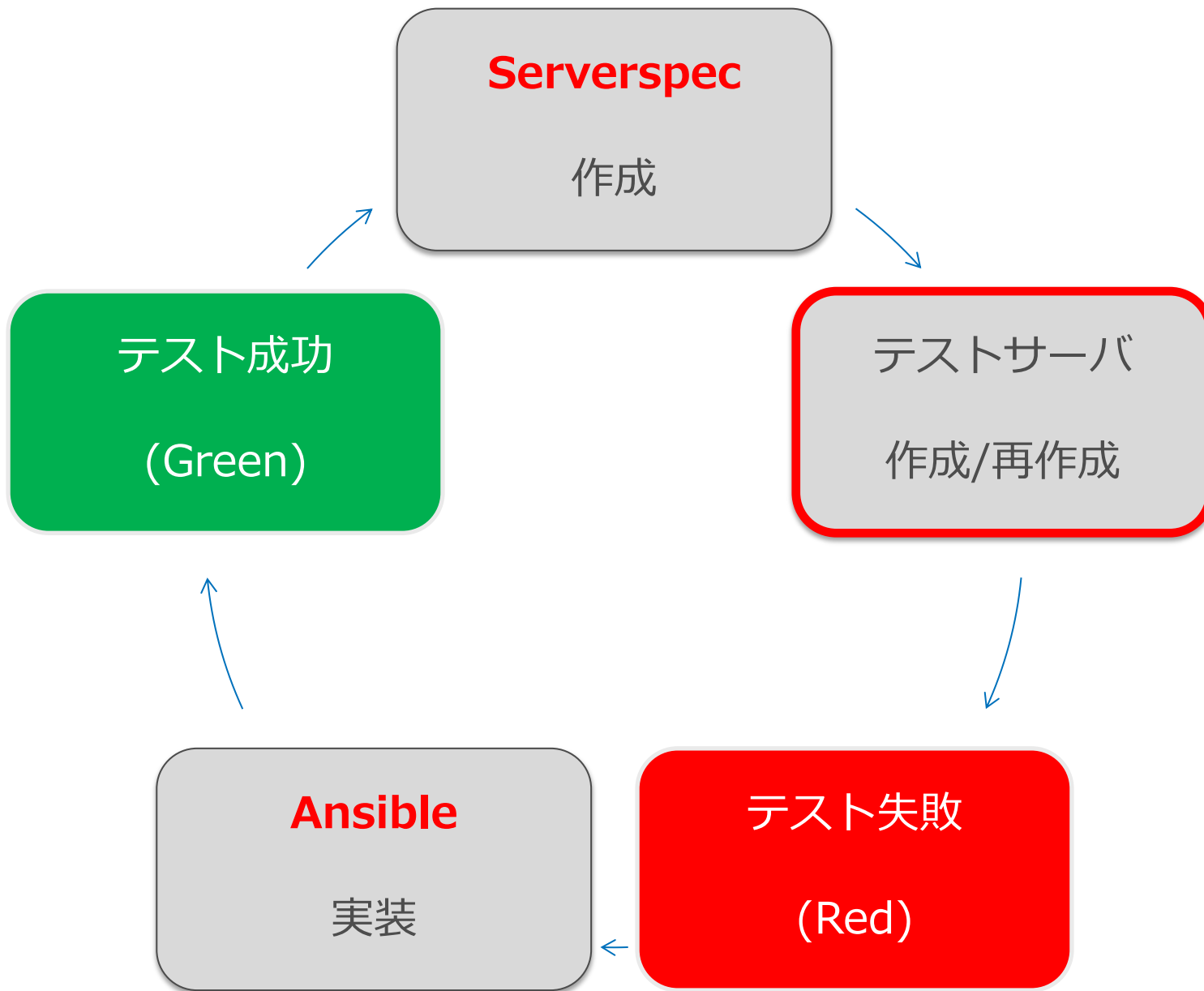
テスト駆動開発(TDD)



- TDDのメリット

- 無理なくテストコードを書ける
- **自信を持ったコーディング**ができる
- コードの**バグ**に早く気付ける
- **回帰テストが容易**にできる
- 恐れずに**リファクタリング**ができる
- 結果として高い品質

具体的なやり方



ライブコーディング

お題



ターゲット
(tdd_target)

httpd
service



起動状態
自動起動設定



Listen
ポート
X/TCP

httpd
package



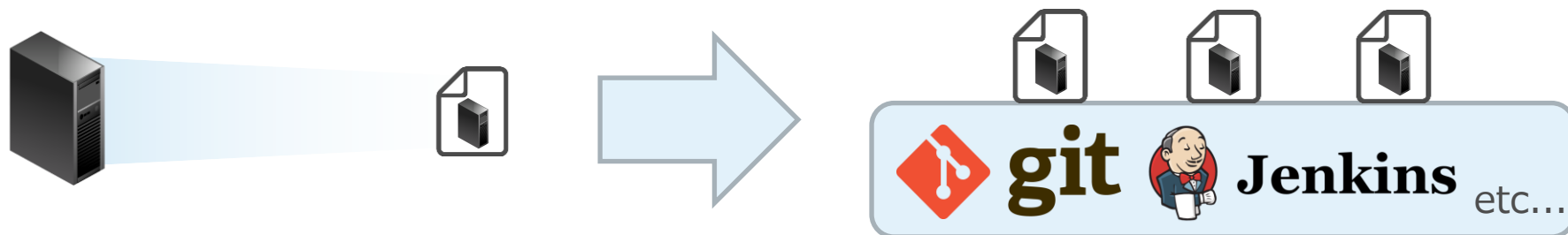
インストール済み



CentOS

まとめ

- Infrastructure as Code(IaC)とは



- IaCの効果

ITインフラ構築・運用・保守の

品質生産性向上

- テスト駆動開発(TDD)

テストコード作成 → テスト失敗 → 実装 → テスト成功



• TDDのメリット

- 無理なくテストコードを書ける
- **自信を持ったコーディング**ができる
- コードの**バグ**に早く気付ける
- **回帰テストが容易**にできる
- 恐れずに**リファクタリング**ができる
- 結果として高い品質

IaCフレームワーク **SHIFTware**

今秋OSS公開予定
coming soon...

ご清聴

ありがとうございました

Q1.

テストを書く粒度はどのくらい？

A1.

TDDにどの程度慣れているか、またコーディング対象についてどれだけ詳しく知っているかによって変わります。

TDDに慣れていない、また対象をあまり詳しく知らなければ、コードの挙動をよく確かめながら進めるため、タスク単位でテストを作成することになるでしょう。一方でTDDに慣れており、対象についても熟知しているのであれば、目標とする状態やその確認方法が明確であるため、実装するのは目標状態であることを確認するテストだけでよいでしょう。

Q2.

Serverspecで実施し辛いテストはどうする？

A2.

TDDで全てのテストを実装しようとしていないでしょうか？

TDDはあくまで開発手法であり、テスト手法ではありません。TDDで行うテストはあくまでコーディングを推進するために行うものであり、品質を担保するためのものではありません。したがって開発者の想定したコード、つまり書き起したテストをパスするコードが書ければそれでよいのです。実施し辛いテストをわざわざテストコードに起こす必要はありません。

TDDを実施しても必ず品質が上がるわけではないのはこの考え方によるもので、TDDを実施したとしても、品質担保のためのテスト(機能テストの一部、受け入れテスト、非機能テスト)は別途実施されるべきなのです。