Ansible・Serverspecベースの OSS開発のCIの成功・失敗談

2017.5.27

TIS株式会社 IT基盤技術推進部
八代 光平

アジェンダ

- 自己紹介
- なぜSIerがこんな話をするのか
- 障害を越えるための取り組み
- IaCのCI
  - 開発開始
  - CI環境
  - テスト実行
  - 合格判定
- まとめ

3 自己紹介

- 名前：八代　光平
- 所属：TIS株式会社　IT基盤技術推進部

- 略歴：
  - 2015年TIS株式会社に入社
  - インフラの自動化（SDI・IaC）関連の検証・開発
  - 社内へのインフラ自動化推進活動

なぜSIerがこんな話をするのか

5 インフラエンジニアの抱える課題

- インフラ構築時の人的ミス
- ドキュメントと実機の乖離
- 属人化と品質

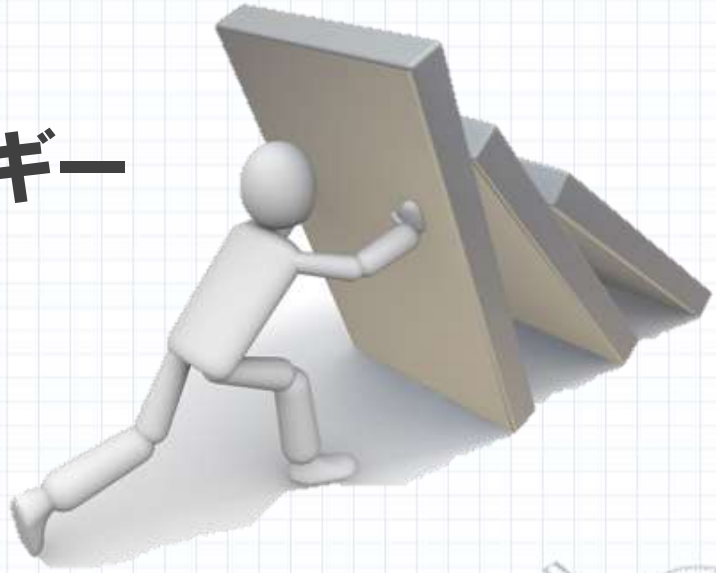
=> **より早く・より安く・でも品質は変えないで!!**

6 新しい技術

- 仮想化技術
  - VMware vSphere、VirtualBox
- クラウド
  - AWS、Azure
- コンテナ
  - Docker、LXD
- IaC
  - Ansible、Serverspec

7 新技術導入への課題

- お客様環境
- 新しい技術へのアレルギー
- 技術習得とコスト



課題を解決するための取り組み

Ansbile・ServerspecベースのOSS開発

Ansibleとは

Ansibleのplaybook

```
- name: network configuration
  os_network:
    cloud: demo
    name: demo_network

- name: subnet configuration
  os_subnet:
    cloud: demo
    name: demo_subnet
    network_name: demo_network
    cidr: 192.168.151.0/24

- name: router configuration
  os_router:
    cloud: demo
    name: demo_router
    network: provider
    interfaces: demo_subnet
```

- インフラの構成管理ツール
- コードに基づきターゲットを構成
  （≒自動設定）
- べき等性
- エージェントレス（SSH/WinRM）

Serverspecとは

- インフラの自動テストツール
- 設定を確認(≒単体テスト)
- エージェントレス(SSH/WinRM)

Serverspecのspecファイル

```
describe user(“user01”) do
  describe (“が存在すること") do
    it { should exist }
  end

  describe (“のuidが500であること) do
    it { should have_uid “500" }
  end

  describe (“がgroupに所属すること“)   do
    it { should belong_to_group group}
  end
```

IaCの恩恵

- インフラ構築時の人的ミス

  ⇒自動化による手作業の抑制

- ドキュメントと実機の乖離

  ⇒コード自体で構成管理

- 属人化と品質

  ⇒機械化で品質は均一化

受け入れられないIaC

本当にちゃんと動くの？

やり方を変えたくない

プログラムなんて書けない

SHIFTWare

Ansible、Serverspecに関連するコード集

＋

コードライブラリを利用するための

フロントエンドツール

■Excel to Parameters for SHIFT■

メニュー	利用方法 Shift管理者用→
サーバ設定ファイルを作成 ←②	①設定値(Ansible)列、確認値(Serverspec)列に値を記入してください ・黄色のセルは入力必須です。 ・設定または確認が不要なパラメータは値をブランク、または行を削除してください。 ・入力した値に不備があるとセルが赤色になります。入力可能リストをご確認ください。 ・同一設定を複数行う場合は、対象の行をコピーしてください。(例:3ユーザを作成、確認する) ・行のコピー、削除は↑↓間を1単位として実施してください。 ※ターゲットマシン1台につき、1シートが必要です。 ※ターゲットマシンが複数の場合は、対象OSのシートをコピー、編集してください。 ※ターゲットマシンに含まれないOSのシートは削除してください。 ②パラメータの記入が完了したら、左のメニューから「サーバ設定ファイルを作成」ボタンを押してください。 ※カレントディレクトリ¥property、および¥property¥host_vars配下に全シート分のpropertyが出力されます。

No.	設定項目	設定値(Ansible) ①	確認値(ServerSpec) ①	備考	入力可能リスト
1	接続用) ホスト名/IPアドレス	192.168.127.40	192.168.127.40		
71	コンピュータ名				
72	所属するグループ				
73	ドメイン/ワークグループ				domain,workgroup
74	ドメイン名/ワークグループ名				
75	ドメイン参加/脱退用ユーザ				
76	ドメイン参加/脱退用パスワード				
77	ネットワークインターフェース				
78	インターフェース名	イーサネット	イーサネット		
79	DHCP有効/無効	FALSE	FALSE		
80	IPアドレス	192.168.127.40	192.168.127.40	IPv6非対応	
81	サブネットマスク長(bit)	23	23		
82	NetBIOS over TCP/IP	dhcp	dhcp		dhcp,disabled,enable
83	WINSアドレス(Primary)	192.168.127.1	192.168.127.1		
84	WINSアドレス(Secondary)	192.168.127.2	192.168.127.2		
85	状態(有効/無効)	Up	Up		Up,Disabled

コーディングの必要なし

SHIFTWare

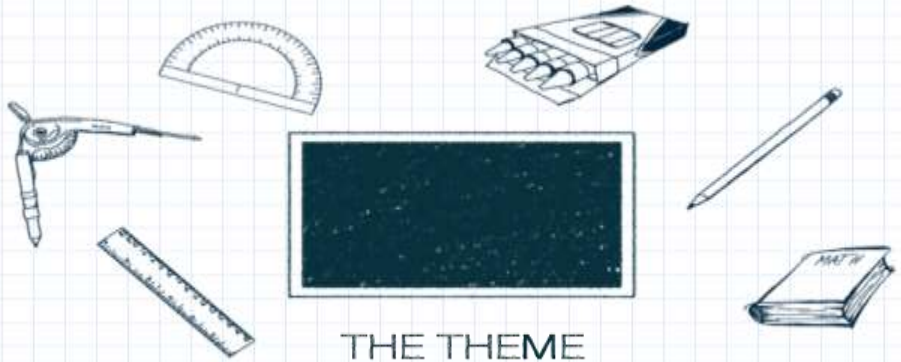
Ansible、Serverspecに関連するコード集

+

コードライブラリを利用するための

コンソールツール

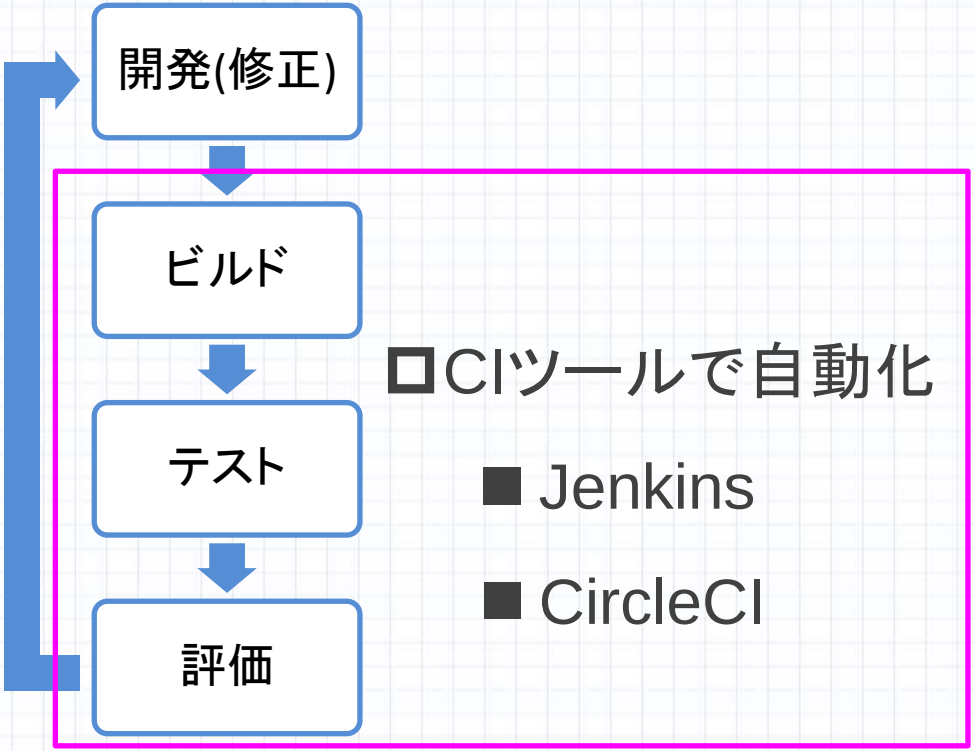
OSS化予定



IaCのCI

CIとは

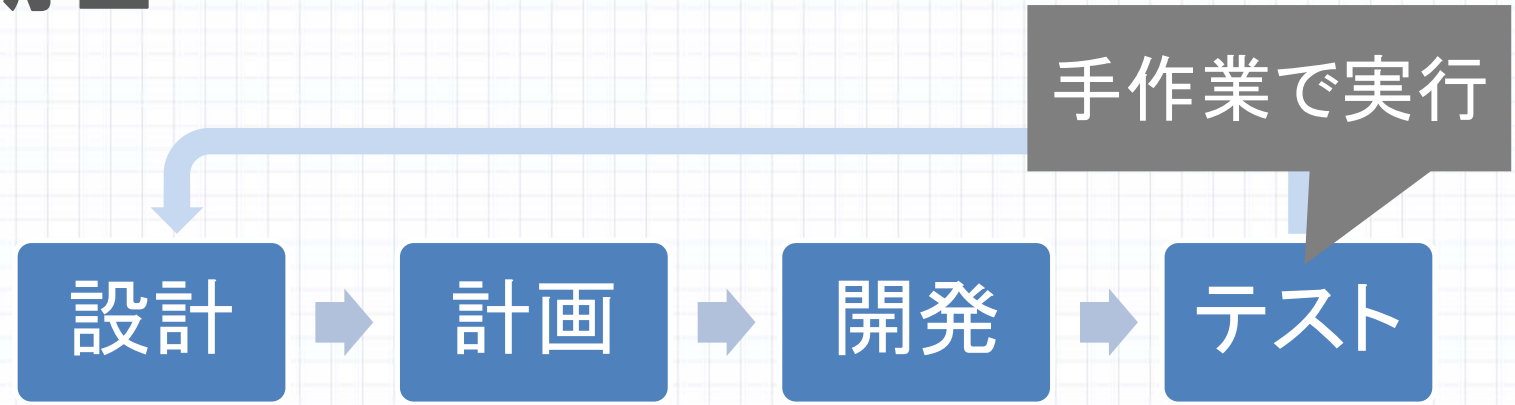
CI=継続的インテグレーション



開発開始

開発の進め方1

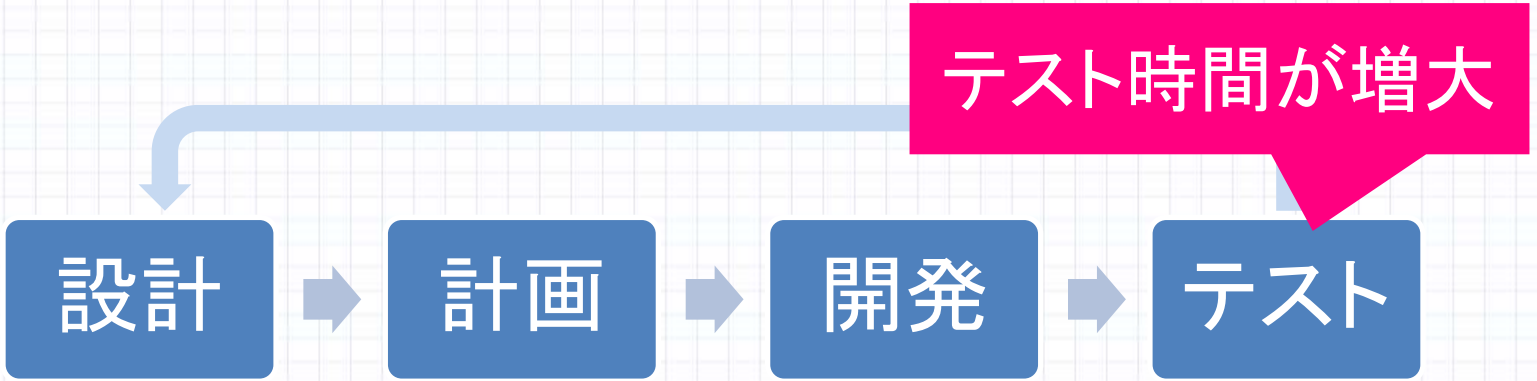
開発序盤



コード量が少ないうちは問題ないが…

開発の進め方2

開発中盤



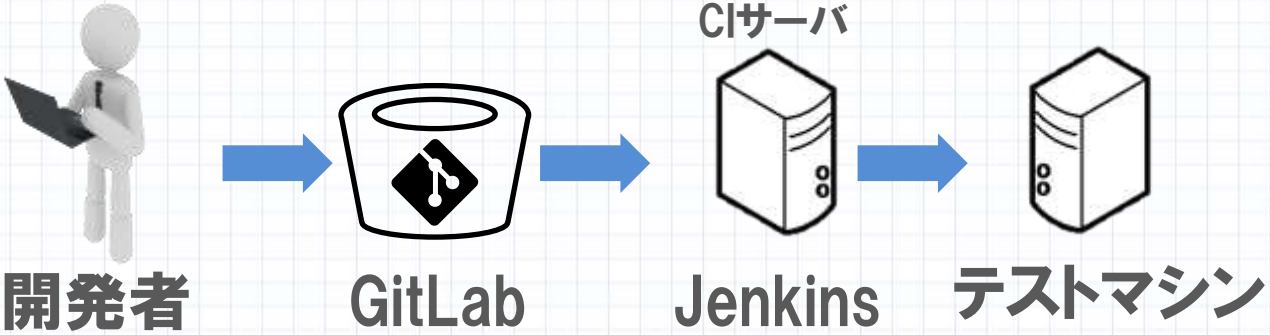
失敗：この段階でやっとCIが必要だと気づく

21 開発の進め方3

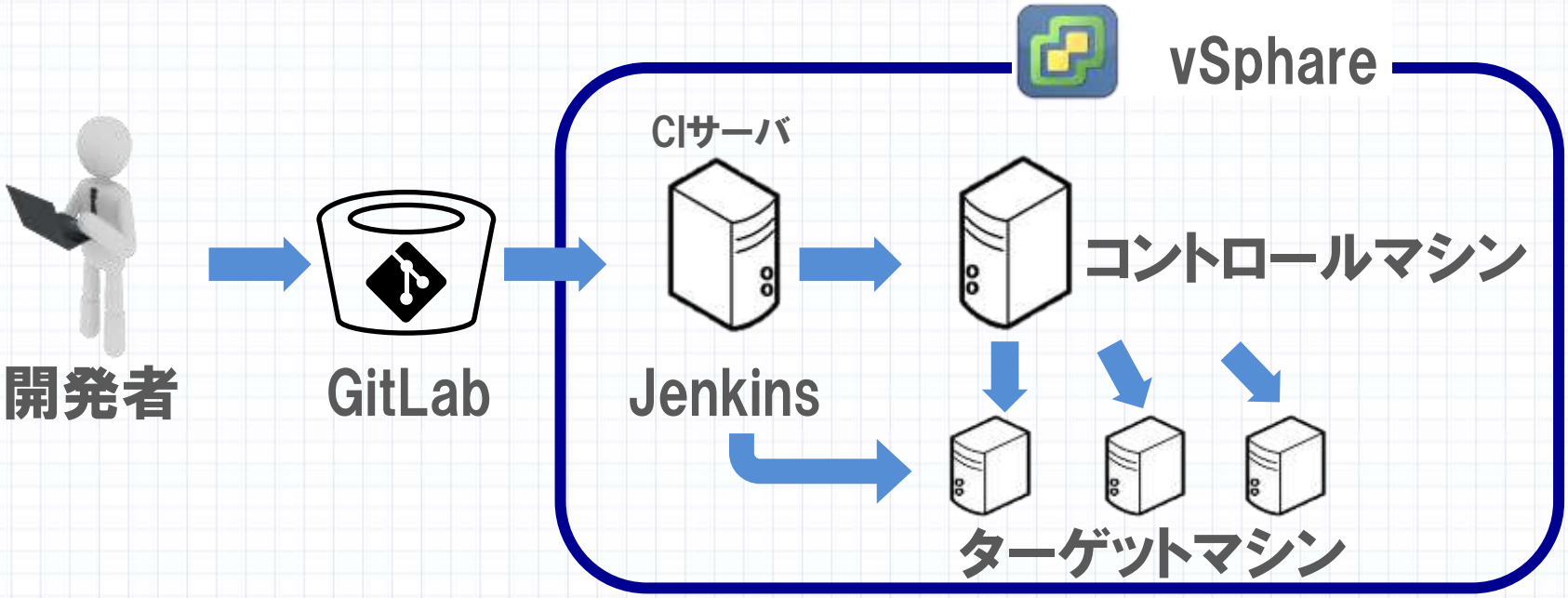
- 手法
  - アジャイル開発
  - テスト駆動開発
- ツール
  - バージョン管理ツール
  - バグ追跡ツール
  - CIツール

CI環境構築

CI環境1



CI環境2



成功：このCIと仮想環境の連携

テスト実行

テスト方法1

Ansible のテストを Serverspec で行う

テスト方法2

Serverspec のテストを  で行う

テストの基本的な流れ

- **成功:構成ツールとテストツールを一組としたテスト**
  1. Serverspecのテストが全て失敗
  2. Ansibleで構築成功
  3. Serverspecで確認
  4. Ansibleのべき等性確認
  5. Serverspecでべき等性確認の確認

合格判定

合格判定1

```
changed: [server1]
changed: [server2]

TASK [1-1301_ZabbixAgent : Update Packa

RUNNING HANDLER [1-1301_ZabbixAgent
```

```
should match /^HostnameItem=system¥

    Timeout=3が設定されていること
      content
192.168.127.31:      OK=14         NG=0
      should match /^Timeout=3/
```

ファイルのdiffをとって合格判定

```
TASK [1-1301_ZabbixAgent : Service State]
ok: [server2]
ok: [server1]

TASK [1-1301_ZabbixAgent : Auto Starting
changed: [server1]
changed: [server2]

server1 : ok=12 changed=9 failed=0
server2 : ok=12 changed=9 failed=0
```

```
14 examples, 0 failures
      content

        should match /^AllowRoot=0/
    Include=/etc/zabbix/zabbix_agentd.d
        should match /^Include=¥/etc¥/zabbi
192.168.127.31:      OK=14         NG=0
192.168.127.41:      OK=14         NG=0
```

合格判定2

```
changed: [server1]
changed: [server2]

TASK [1-1301_ZabbixAgent : Update Packa

RUNNING HANDLER [1-1301_ZabbixAgent
```

```
should match /^HostnameItem=system¥

    Timeout=3が設定されていること
      content
192.168.127.31:      OK=14        NG=0
      should match /^Timeout=3/
```

ファイルのdiffをとって合格判定

失敗：開発チームから大不評

```
TASK [1-1301_ZabbixAgent : Check Zabbix
ok: [server2]
ok: [server1]

TASK [1-1301_ZabbixAgent : Auto Starting
changed: [server1]
changed: [server2]

server1 : ok=12 changed=9 failed=0
server2 : ok=12 changed=9 failed=0
```

```
14      ok=14   NG=0 failed=0
      content

        should match /^AllowRoot=0/
    Include=/etc/zabbix/zabbix_agentd.d
        should match /^Include=¥/etc¥/zabbi
192.168.127.31:      OK=14        NG=0
192.168.127.41:      OK=14        NG=0
```

合格判定3

```
changed: [server1]
changed: [server2]

TASK [1-1301_ZabbixAgent : Update Packa

RUNNING HANDLER [1-1301_ZabbixAgent
```

```
should match /^HostnameItem=system¥

    Timeout=3が設定されていること
      content
192.168.127.31:      OK=14         NG=0
      should match /^Timeout=3/
```

実行結果のステータスの一部のみチェック

```
TASK [1-1301_ZabbixAgent : Service State]
```

```
14 examples, 0 failures
```

正規表現で柔軟に判定

```
TASK [1-1301_ZabbixAgent : Auto Starting
changed: [server1]
changed: [server2]

server1 : ok=12 changed=9 failed=0
server2 : ok=12 changed=9 failed=0
```

```
    Include=/etc/zabbix/zabbix_agentd.d
      should match /^Include=¥/etc¥/zabbi
192.168.127.31:      OK=14         NG=0
192.168.127.41:      OK=14         NG=0
```

まとめ

まとめ

- 開発開始前にCI環境準備
- 仮想環境は必須
- 構築ツールとテストツールは一組
- 開発効率と品質のバランスを調整