OSC2017 Tokyo/Spring

TIS IT Holdings Group | Go Beyond

システムインテグレータの働き方をシフトする、新しい自動化活用

2017.3.10
TIS株式会社 IT基盤技術推進部
髙木　光一郎

Agenda

- インフラを取り巻く環境
- Infrastructure as Code
- TISの取り組み
- Project SHIFT(仮)
- IaCのスキルアップに向けて

自己紹介

名前：髙木　光一郎
所属：TIS株式会社　IT基盤技術推進部

略歴：
・2006年TISに入社
・ずっとインフラをやっている
・OSC 2015 Tokyo FallでOSS運用管理ツール「Hinemos」について講演
・自動化歴はおよそ半年

インフラを取り巻く環境

インフラを取り巻く環境

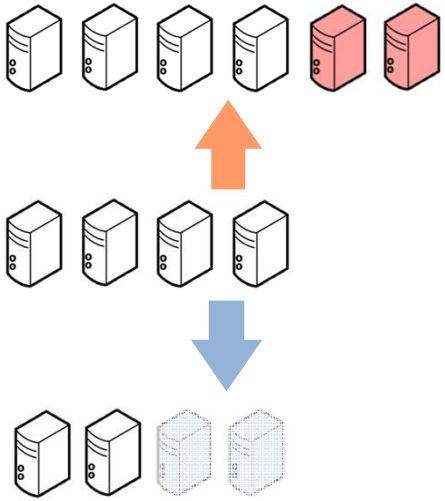
- **仮想化**
- **クラウドインフラ**
- **Software Defined xxx**
- **コンテナ**

インフラを取り巻く環境

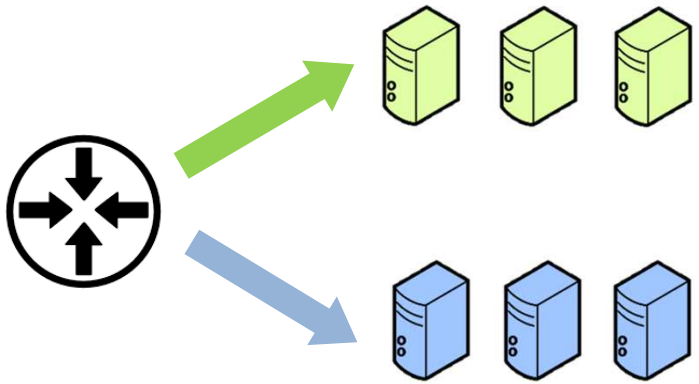
- **インフラ調達のスピードアップ**
- **リソースの柔軟性**
- **ハードウェア運用からの解放**
- **コストは使った分だけ**

インフラを取り巻く環境

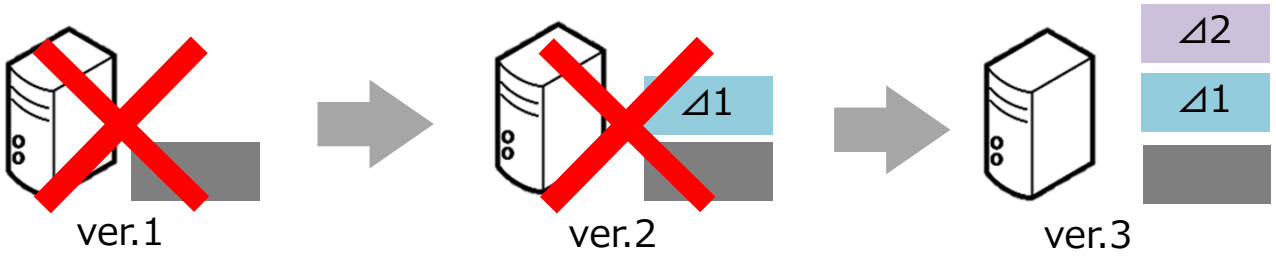
Anto Scaling



Blue Green Deployment



Immutable Infrastructure



従来の課題

新しい技術や手法が生まれる一方で、インフラエンジニアを悩ませる課題は今までと変わらない。

- 作業で発生するミス
- 属人化、品質のバラつき
- ドキュメントと実機が違う
- etc...

インフラエンジニアの課題

インフラエンジニアは、従来の課題をクリアしつつ、技術の変化にも対応しないといけない板挟み状態。

OLD

NEW

Infrastructure as Code

Infrastructure as Code

インフラエンジニアを悩ませる課題の解決策の
ひとつが「インフラの自動化」 もっと言うと、、

Infrastructure as Code (IaC)

Infrastructure as Code

- インフラ調達のスピードアップ
- リソースの柔軟性
- ハードウェア運用からの解放
- コストは使った分だけ
- インフラのソフトウェアによる抽象化

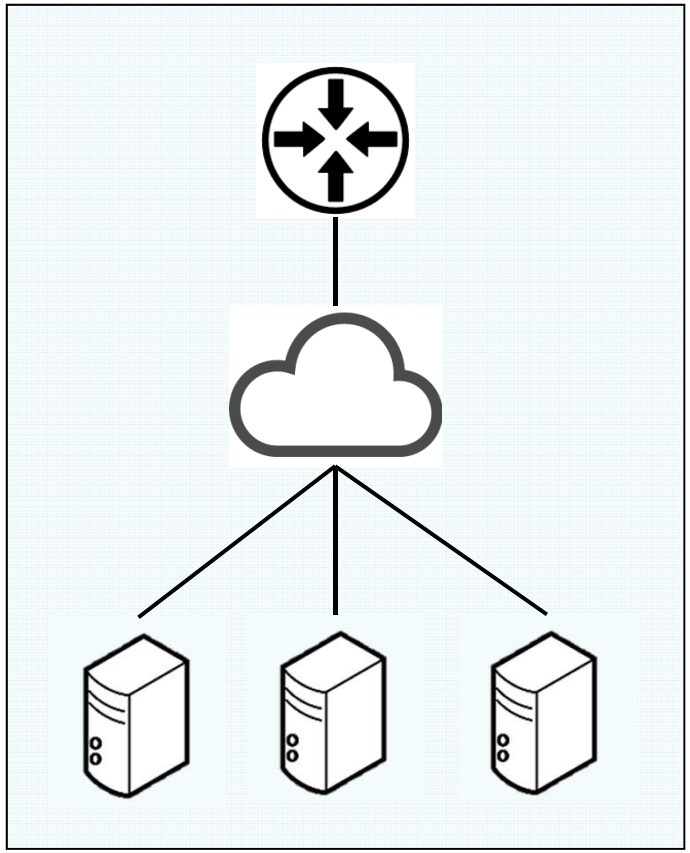
Infrastructure as Code

インフラの構成を、プログラムのコードで管理する手法

```
- 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

 - name: instance configuration
  os_server:
    cloud: demo
    name: demo_instance_{{ item }}
    image: cirros
    flavor: m1.nano
    network: demo_network
  with_items: [1,2,3]
```

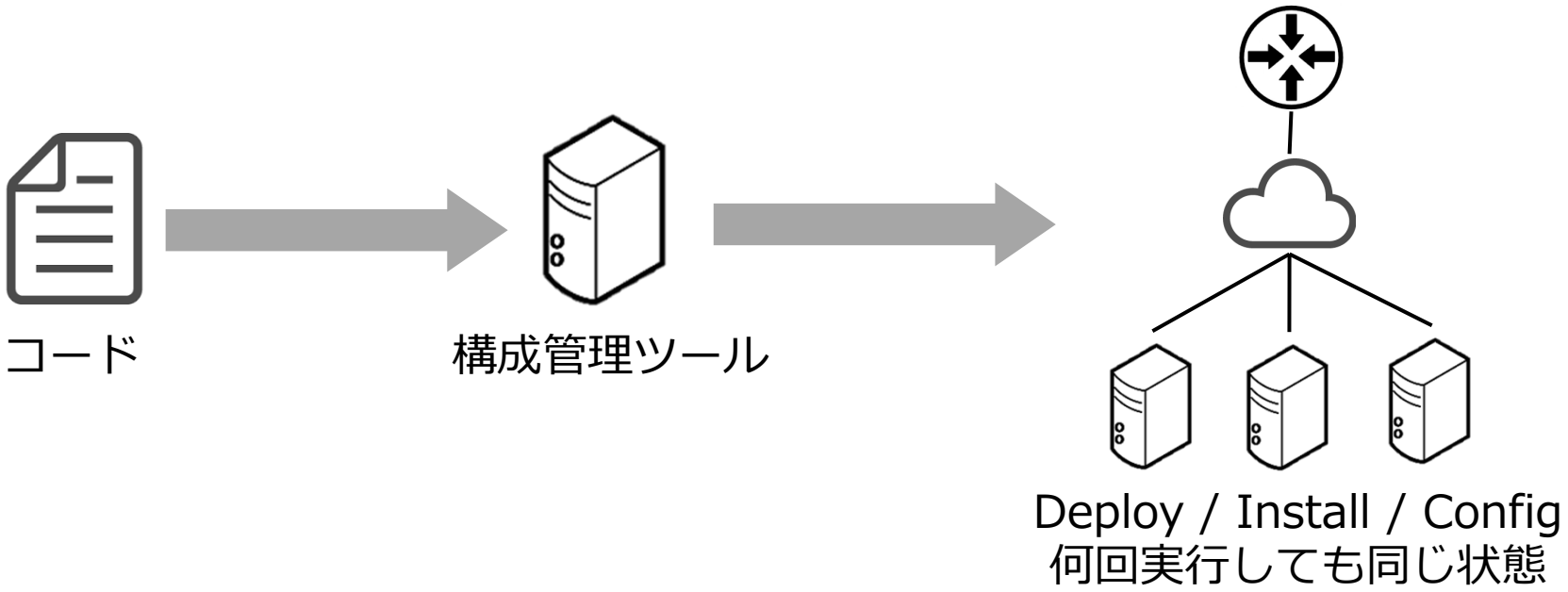
＝

イコール



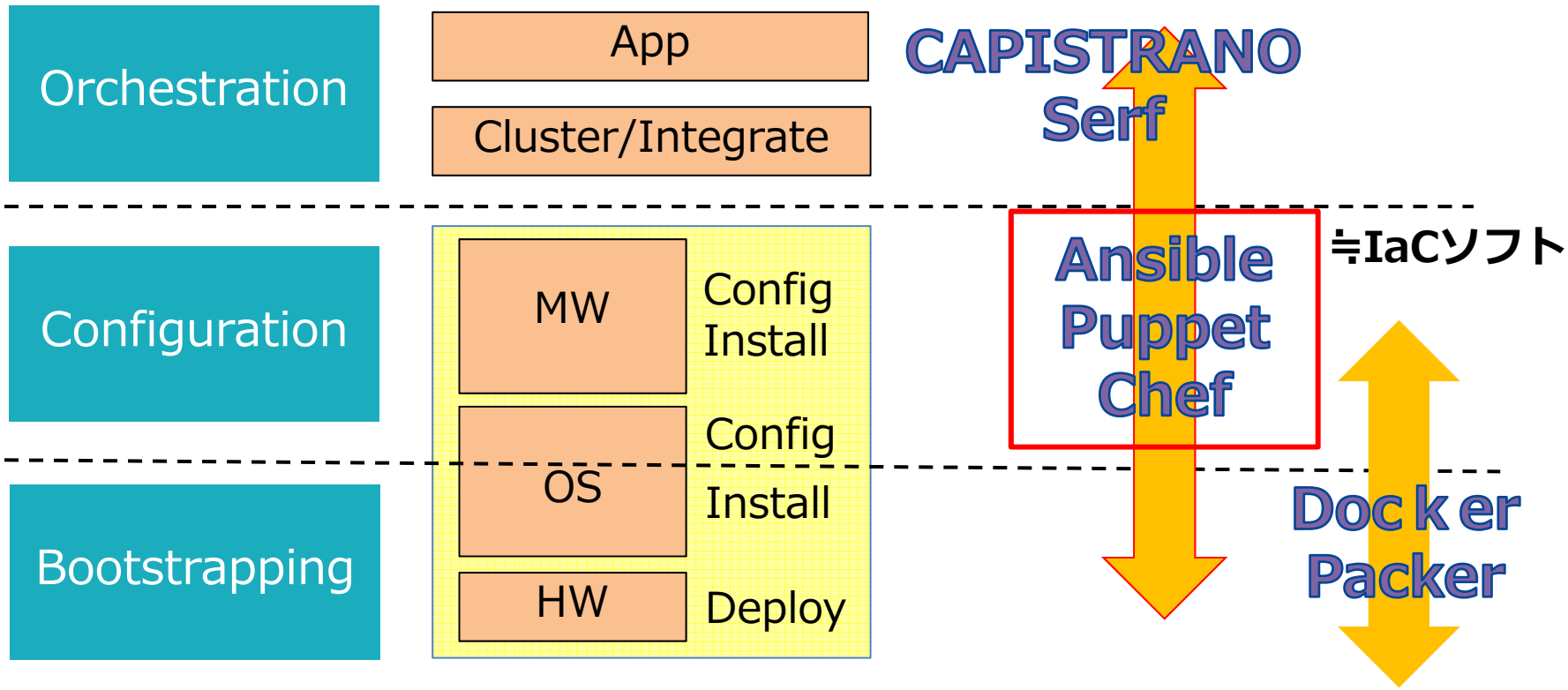
Infrastructure as Code

構成管理ツールによって、コードに基づきインフラが構成される。
コードは何度実行しても同じ状態に収束するため（べき等性）、コード＝構成である。



Infrastructure as Code

3 Layer & Tools



Infrastructure as Code

コード化することで、ソフトウェア開発の手法をインフラに適用することができる。

- バージョン管理
- 繰り返し可能なビルド
- テスト
- 継続的インテグレーション(CI)
- 継続的デリバリ(CD)

Infrastructure as Code

Infrastructure as Codeを活用することで、多くのメリットを得られる。

- 納期短縮
- 工数削減
- 品質向上
- 生産性向上

⇒つまり、楽していいものができる。

Infrastructure as Code

従来の課題に対しても有効

- 作業で発生するミス
  ⇒**自動化による手作業の抑制**
- 属人化、品質のバラつき
  ⇒**誰が実行しても同じものができる**
- ドキュメントと実機が違う
  ⇒**実行したコードそのもので構成を管理**

Infrastructure as Code

新旧課題は

OLD

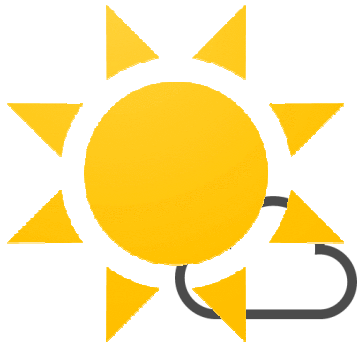
NEW

Infrastructure as Code

Infrastructure as Code で解決

CLEAR

CLEAR



TISの取り組み

IaC推進に向けての課題

しかしながら、すぐにIaCを実践できるわけではない。

学習・育成コスト
コードアレルギー
やり方が変わる
時間がない

TISの取り組み

課題に対する２方向のアプローチ

ハードルDOWN↓

- コードライブラリ
- フレームワーク
- スクリプト
- ツール

スキルUP↑

- 社内勉強会
- セルフハンズオン
- ナレッジ公開
- 案件利用支援

Project SHIFT(仮称)

SHIFT とは

Standard
Helpful
Infrastructure
Framework
of **T**IS

SHIFT とは

SHIFT とは、Infrastructure as Code およびその周辺機能を含んだインフラフレームワークの開発コード・プロジェクト名

SHIFTのコンセプト

従来の(延長線の)やり方から転換

(ユーザ目線で)便利な、
インフラ標準フレームワーク

SHIFT とは

SHIFTによって、Infrastructure as Codeを誰でも利用しやすく

ハードルDOWN↓
コードライブラリ
フレームワーク
スクリプト
ツール

スキルUP↑
社内勉強会
セルフハンズオン
ナレッジ公開
案件利用支援

SHIFT とは

実体は

Ansible、Serverspec のコード集

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

Ansible とは

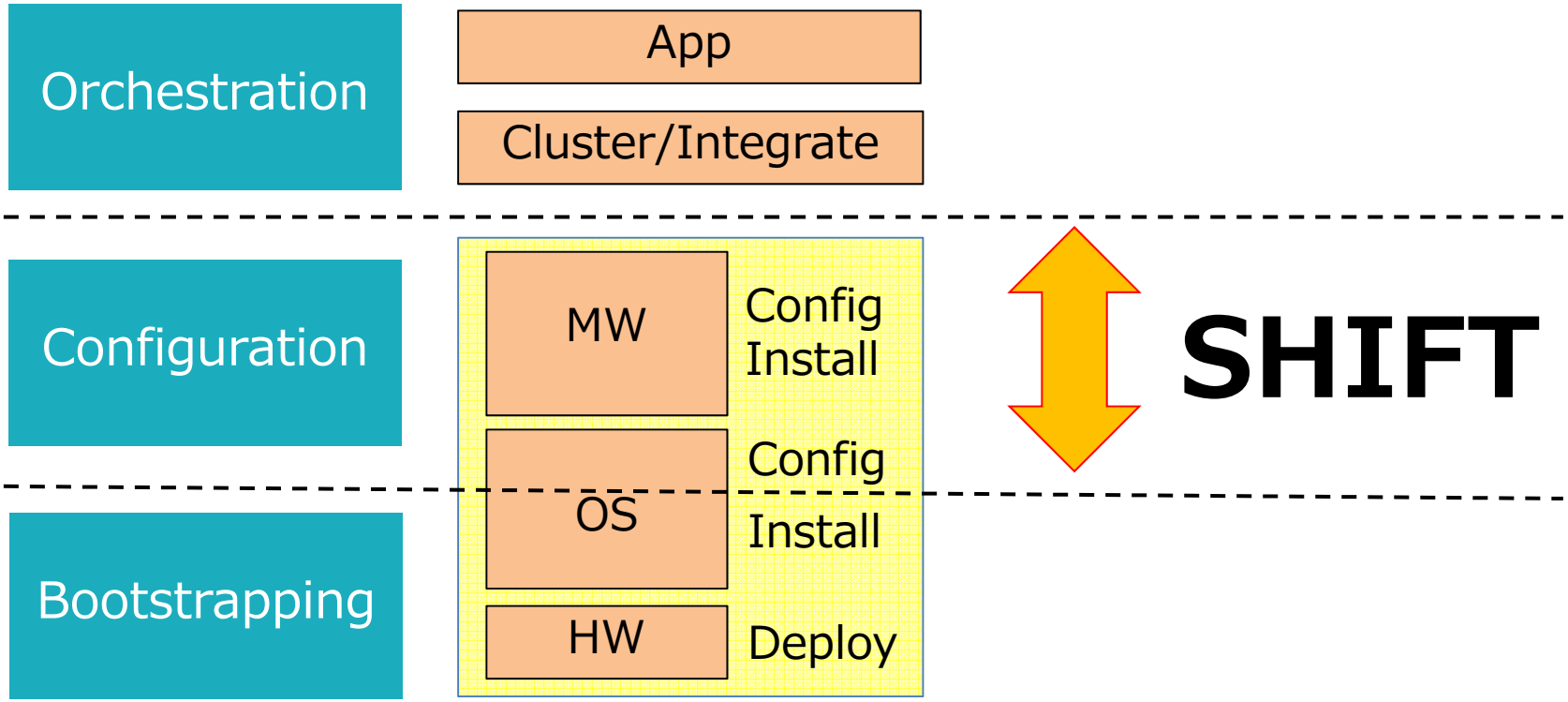
・インフラの構成管理ツール
・冪等性（べきとうせい）
・エージェントレス(SSH/WinRM)
・YAMLで定義(プログラミングではない)

Serverspec とは

- インフラの自動テストツール
- 設定を確認する
- エージェントレス(SSH/WinRM)
- 構成管理ツールに依存しない

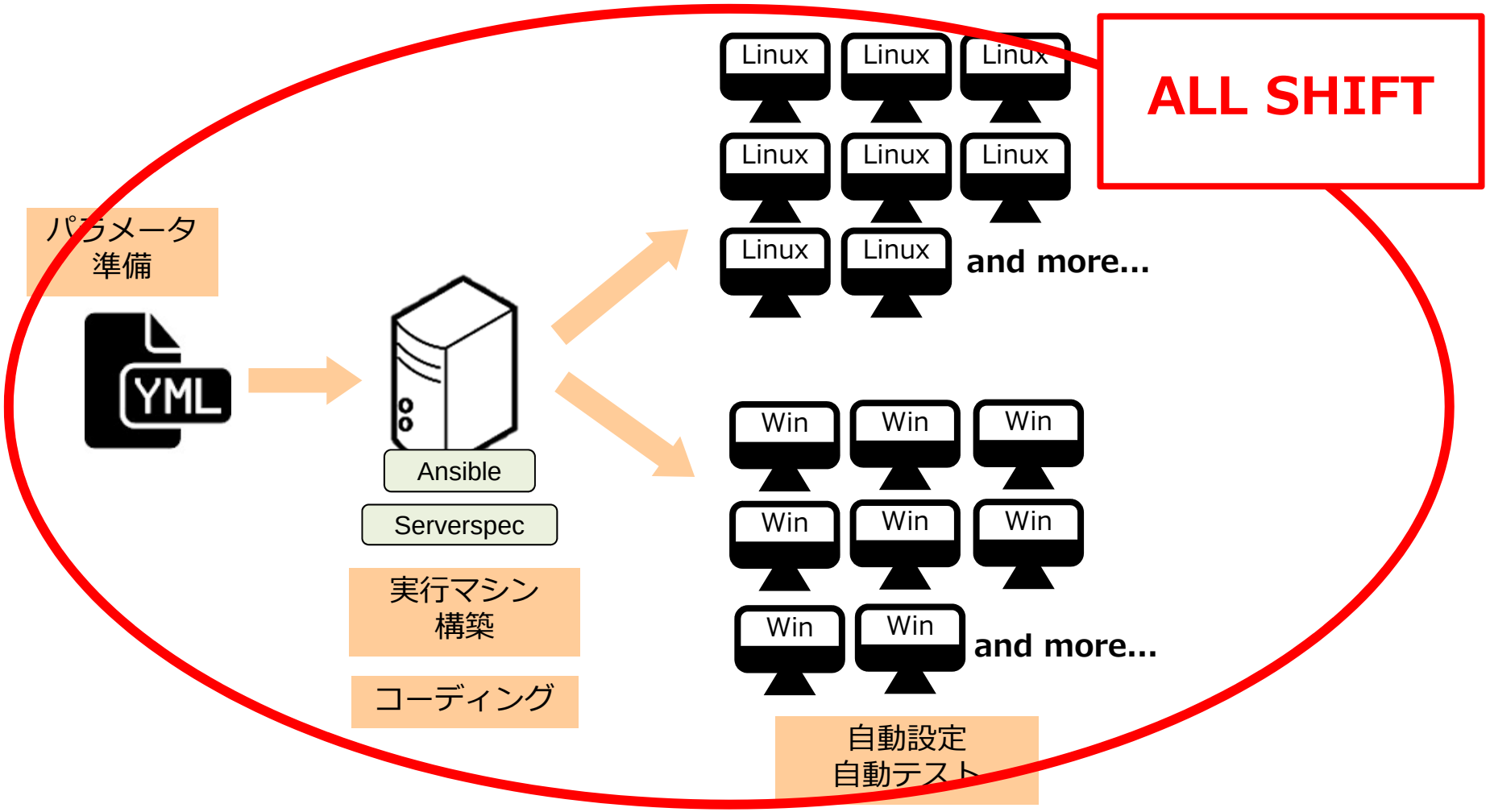
SHIFT で出来ること

OSの、
　自動設定 / 自動テスト
ミドルウェアの、
　自動インストール / 自動設定 / 自動テスト



SHIFT で出来ること

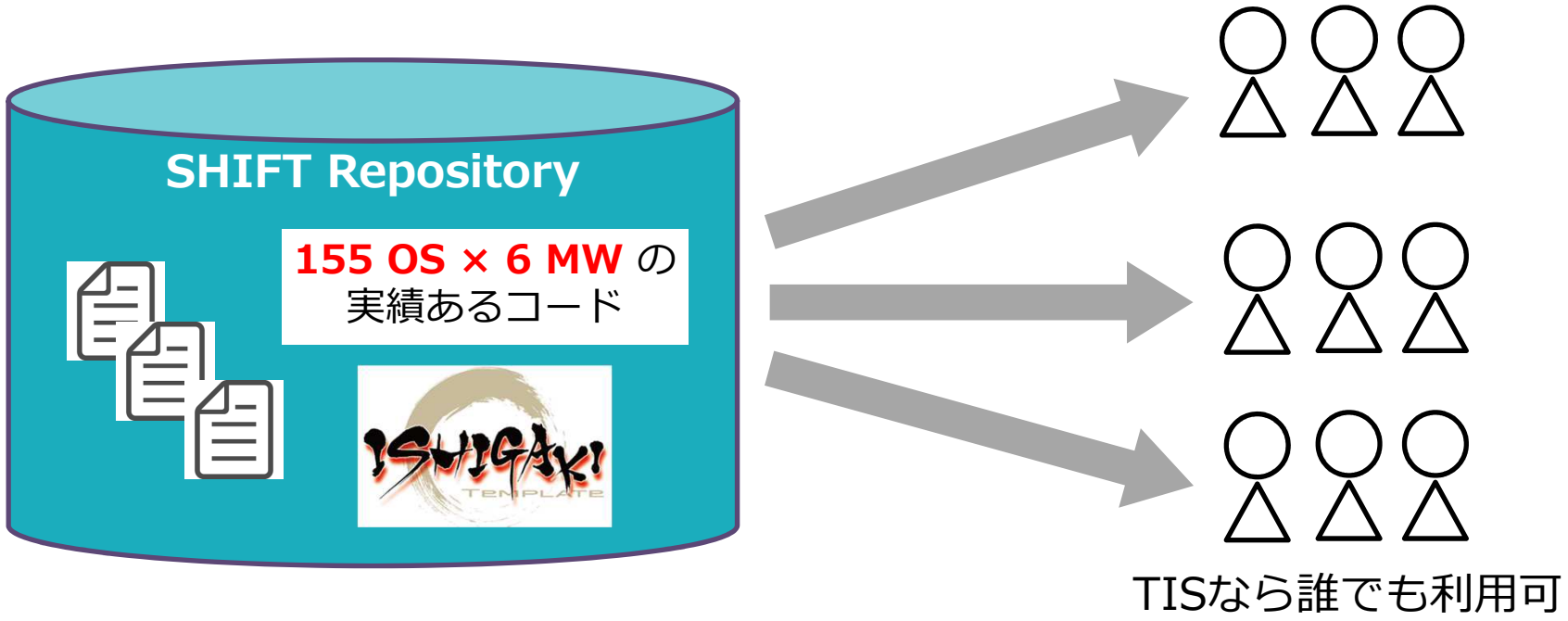
IaCの準備～実行まで、トータルでサポート



SHIFT Roles and Tools

SHIFT Roles and Tools

案件利用実績のあるコードを汎用化したものや
TISのOSS推奨スタック「Ishigaki Template」
のコードをシェアリポジトリとして社内に公開、
すぐに自動設定、テストで利用可能。



SHIFT Linux Roles

Linux Roles

1-0001_Base
1-0102_apache
1-0103_Tomcat
1-0104_postgreSQL
1-0105_openJDK
1-0106_PrivateCA
1-0201_RedHatSatellite
1-0401_JP1Agents
1-0501_OracleClient
1-0502_OracleJRE
1-0701_LogstorageAgent
1-0801_DynatraceAgent
1-0901_DeepSecurityAgent
1-1001_UnixAgentForLDAPManager
1-1101_ES1Acquire
1-1102_ES1Logscn

Linux Base Tasks

1-0001-04_Hosts
1-0001-05_Resolv
1-0001-07_Service
1-0001-08_Ntp
1-0001-10_Selinux
1-0001-12_Crontab
1-0001-14_Lang
1-0001-17_Logrotate
1-0001-20_Nsswitch
1-0001-23_Sshd
1-0001-39_Snmp
1-0001-40_Syslog
1-0001-13_OSGroup
1-0001-16_PasswordRules
1-0001-37_SystemRunLevel
1-0001-38_InterfaceNmcli
1-0001-24_OSUser.yml
1-0001-27_Kdump
1-0001-28_Keyboard
1-0001-29_Timezone
1-0001-40_Grub
1-0001-02_Interface
1-0001-03_Network
1-0001-22_Route
1-0001-39_Bonding
1-0001-35_SetDirectory

SHIFT Windows Roles

Windows Roles

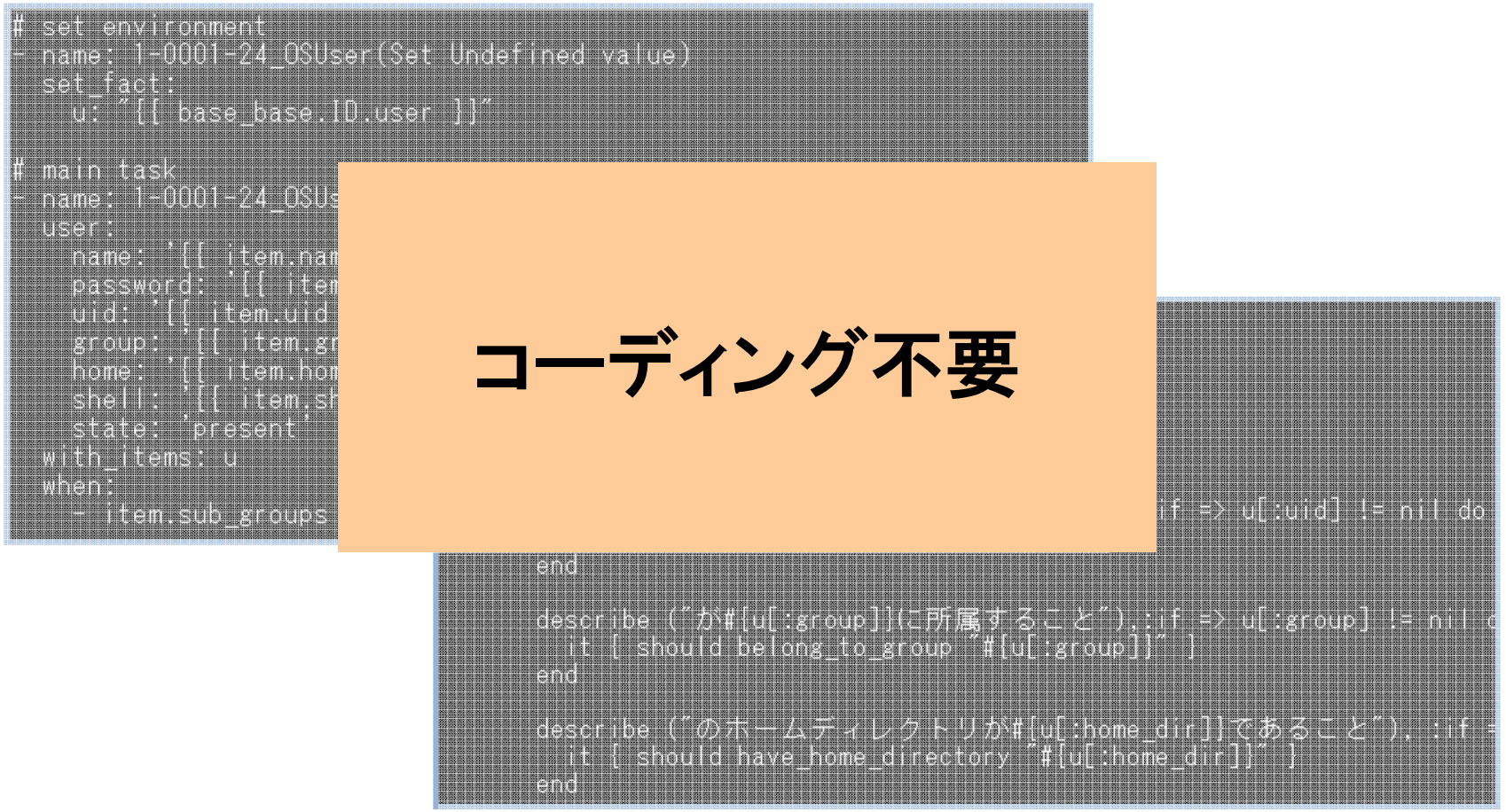
2-0001_Base
2-0101_Cygwin
2-0401_JP1Agents
2-0501_OracleClient
2-0701_LogstorageAgent
2-0901_DeepSecurityAgent
2-1002_LDAPManagerWindowsAgent
2-1101_ES1Acquire
2-1102_ES1Logscn

Windows Base Tasks

2-0001-002_Memory
2-0001-011_User
2-0001-012_UserGroup
2-0001-013_Uac
2-0001-024_Timezone
2-0001-025_RecoverOs
2-0001-051_Directory
2-0001-071_Hostname
2-0001-072_Interface
2-0001-073_Routing
2-0001-074_Firewall
2-0001-075_Teaming
2-0001-076_Domain
2-0001-077_IPv6Disable
2-0001-078_DnsSuffix
2-0001-079_NameResolve
2-0001-095_Feature
2-0001-096_Service
2-0001-098_Rdp
2-0001-102_ErrorReport
2-0001-103_WinUpdate
2-0001-104_Owner
2-0001-105_EventLog
2-0001-106_Registry
2-0001-107_PsExecPolicy
2-0001-109_Organization

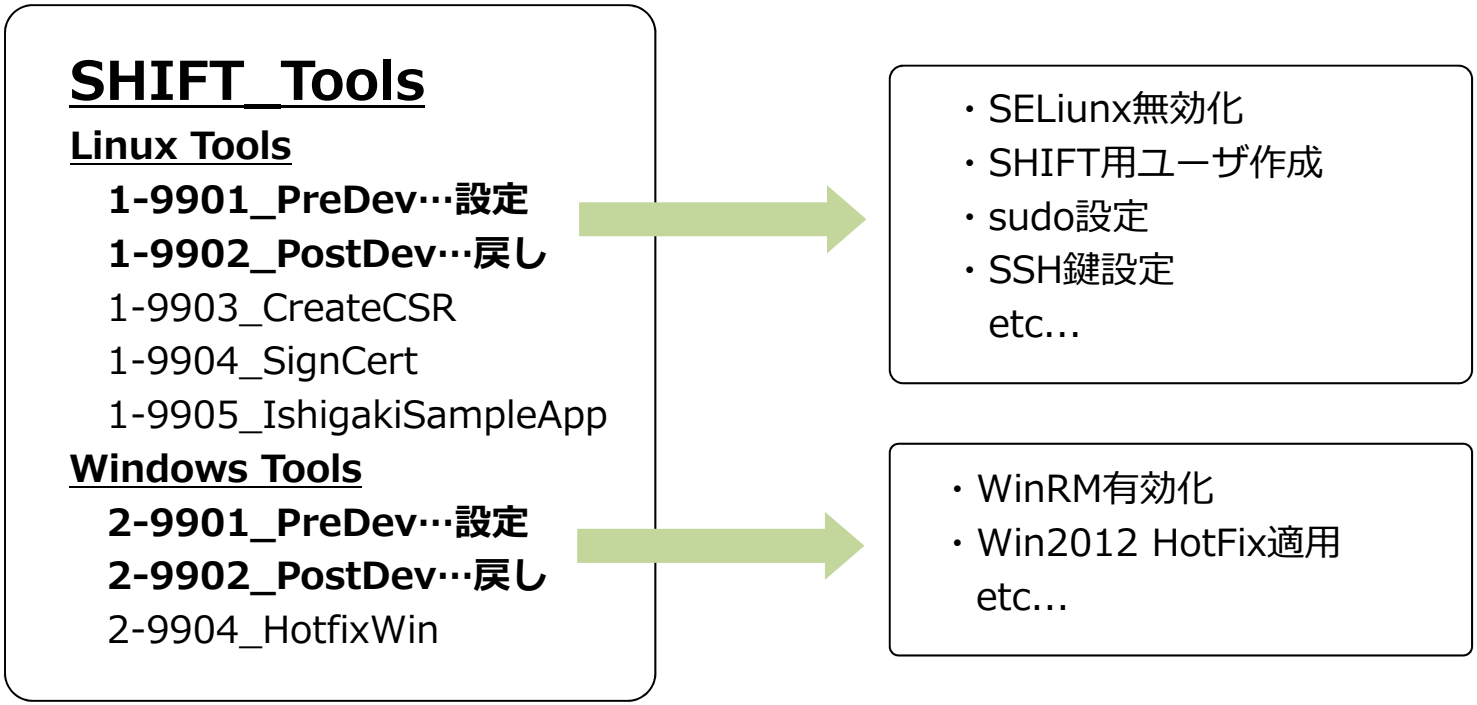
SHIFT Roles

リポジトリに実装されているロールはコーディングレスで自動設定・テストが可能。



SHIFT Tools

構成ではなく、動作を定義するコードをToolとして公開。Ansible、Serverspecを実行するための事前・事後設定もSHIFTで自動化。



Excel 2 YAML

Excel2YAML

「**Excel2YAML**」とは、Ansible、Serverspecのパラメータファイルを Excelで作成できるツール

（おまけとして作ったが、ウケがよく、需要は高い）

Excel2YAML

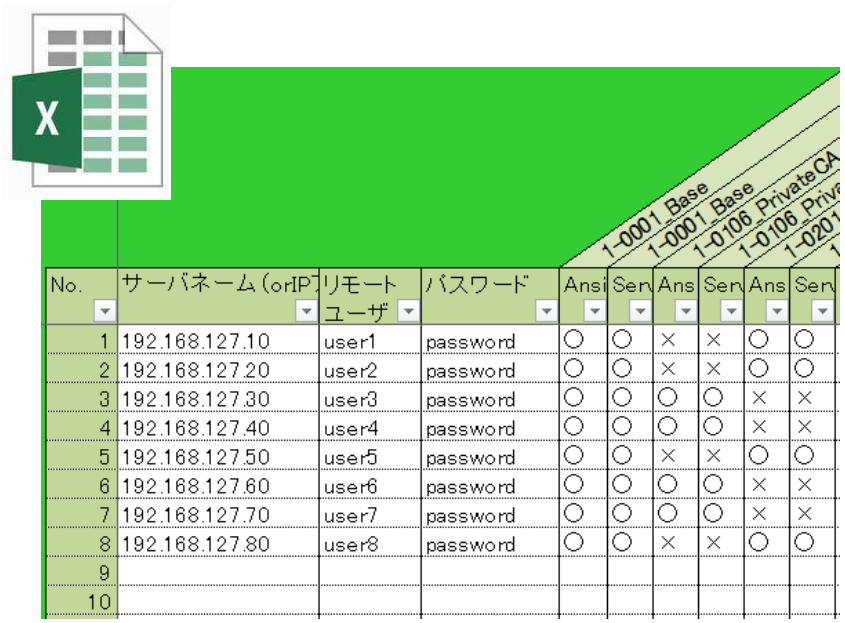
Property…設定・テストのパラメータを定義するファイル

No.	設定値(Ansible)	確認値(ServerSpec)
1 接続用）ホスト名/IPアドレス	192.168.127.10	192.168.127.10
9 ユーザ/グループ		
10 ユーザ設定		
11 ↑ ユーザ名	test1	test1
12 ｜ パスワード	password	
13 ｜ ユーザID(uid)	501	501
14 ｜ 所属グループ	test1	test1
15 ｜ ホーム	/home/test1	/home/test1
16 ｜ ログインシェル	/bin/bash	/bin/bash
17 ↓ セカンダリグループ	test2	test2
18 ↑ ユーザ名	test2	test2
19 ｜ パスワード	password	
20 ｜ ユーザID(uid)	502	502
21 ｜ 所属グループ	test2	test2
22 ｜ ホーム	/home/test2	/home/test2
23 ｜ ログインシェル	/bin/bash	/bin/bash
24 ↓ セカンダリグループ	test1	test1
25 グループ設定		
26 ↑ グループ名	test1	test1
27 ↓ グループID(gid)	503	503
28 ↑ グループ名	test2	test2
29 ↓ グループID(gid)	504	504
30 パスワードポリシー		
31 max_days	99999	99999
32 min_days	0	0
33 min_length	5	5
34 warn_age	7	7

```
nection_hostname: '192.168.127.10'
te_tmp_dir: '~root/.shift'
BASE:
  ID:
    user:
      - name: 'test1'
        password: 'password'
        uid: 501
        group: 'test1'
        home_dir: '/home/test1'
        shell: '/bin/bash'
        sub_groups: 'test2'
      - name: 'test2'
        password: 'password'
        uid: 502
        group: 'test2'
        home_dir: '/home/test2'
        shell: '/bin/bash'
        sub_groups: 'test1'
    user_group:
      - name: 'test1'
        gid: 503
```

Excel2YAML

Inventory…実行対象（ターゲット）と実行する処理（ロール）を定義するファイル



No.	サーバネーム(orIP)	リモートユーザ	パスワード	1-0001_Base Ansi	1-0001_Base Ser	1-0106_PrivateCA Ans	1-0106_PrivateCA Ser	1-0201 Ans	1-0201 Ser
1	192.168.127.10	user1	password	○	○	×	×	○	○
2	192.168.127.20	user2	password	○	○	×	×	○	○
3	192.168.127.30	user3	password	○	○	○	○	×	×
4	192.168.127.40	user4	password	○	○	○	○	×	×
5	192.168.127.50	user5	password	○	○	×	×	○	○
6	192.168.127.60	user6	password	○	○	○	○	×	×
7	192.168.127.70	user7	password	○	○	○	○	×	×
8	192.168.127.80	user8	password	○	○	×	×	○	○
9									
10									

Inventory

```
se_GROUP]
.10  ansible_user=user1  ansible_ssh_pass=password
192.168.127.20  ansible_user=user2  ansible_ssh_pass=password
192.168.127.30  ansible_user=user3  ansible_ssh_pass=password
192.168.127.40  ansible_user=user4  ansible_ssh_pass=password
192.168.127.50  ansible_user=user5  ansible_ssh_pass=password
192.168.127.60  ansible_user=user6  ansible_ssh_pass=password
192.168.127.70  ansible_user=user7  ansible_ssh_pass=password
192.168.127.80  ansible_user=user8  ansible_ssh_pass=password
[1-0102_apache_GROUP]
[1-0103_Tomcat_GROUP]
[1-0104_postgreSQL_GROUP]
[1-0105_openJDK_GROUP]
[1-0106_PrivateCA_GROUP]
192.168.127.30  ansible_user=user3  ansible_ssh_pass=password
192.168.127.40  ansible_user=user4  ansible_ssh_pass=password
192.168.127.60  ansible_user=user6  ansible_ssh_pass=password
192.168.127.70  ansible_user=user7  ansible_ssh_pass=password
[1-0201_RedHatSatellite_GROUP]
192.168.127.10  ansible_user=user1  ansible_ssh_pass=password
192.168.127.20  ansible_user=user2  ansible_ssh_pass=password
192.168.127.50  ansible_user=user5  ansible_ssh_pass=password
192.168.127.80  ansible_user=user8  ansible_ssh_pass=password
```

※AnsibleはYAMLではない

Serverspec Customize

Serverspec Customize

Serverspecは案件利用しやすいようにカスタマイズ

- AnsibleのRole、Varsと同じ粒度で管理/実行
  - ⇒ Inventoryでターゲットとテストを管理
  - ⇒ Propertyでターゲット毎のパラメータを管理

- 並列実行をするためにマルチプロセス化

Serverspec Customize

- ログ出力(CSV / Debug / Summary / Console)

CSV

	A	B	
1	192.168.127.211		
2	- OK=96		
3	- NG=0		
4	==============================		
5	- 011-OSユーザ		
6	- User "test1"		
7	- ユーザが存在すること		
8	- should exist	OK	/bin/sh -c id¥ test1
9	- のuidが501であること		
10	- should have uid "501"	OK	/bin/sh -c id¥ test1¥ ¥|¥ grep¥ --¥ ¥¥¥^uid¥¥¥=501¥¥¥(
11	- がtest1に所属すること		
12	- should belong to group "test1"	OK	/bin/sh -c id¥ test1¥ ¥|¥ sed¥ ¥'s/¥ context¥=.¥*//g¥'¥ ¥|¥ cut¥ -
13	- のホームディレクトリが/home/test1であること		
14	- should have home directory "/home/test1"	OK	/bin/sh -c getent¥ passwd¥ test1¥ ¥|¥ cut¥ -f¥ 6¥ -d¥ ¥':¥'¥ ¥|¥ g
15	- のログインシェルが/bin/bashであること		
16	- should have login shell "/bin/bash"	OK	/bin/sh -c getent¥ passwd¥ test1¥ ¥|¥ cut¥ -f¥ 7¥ -d¥ ¥':¥'¥ ¥|¥ g
17	- がセカンダリグループとしてtest2に所属していること		
18	- Command "grep test2 /etc/group"		
19	- stdout		
20	- should match /test1(,|¥s*$)/	OK	/bin/sh -c grep¥ test2¥ /etc/group

Debug

```
192.168.127.211
[1]
***Command***
/bin/sh -c id\ test1

***Stdout***
uid=501(test1) gid=503(test1) groups=503(test1),504(test2)
```

Other Tools

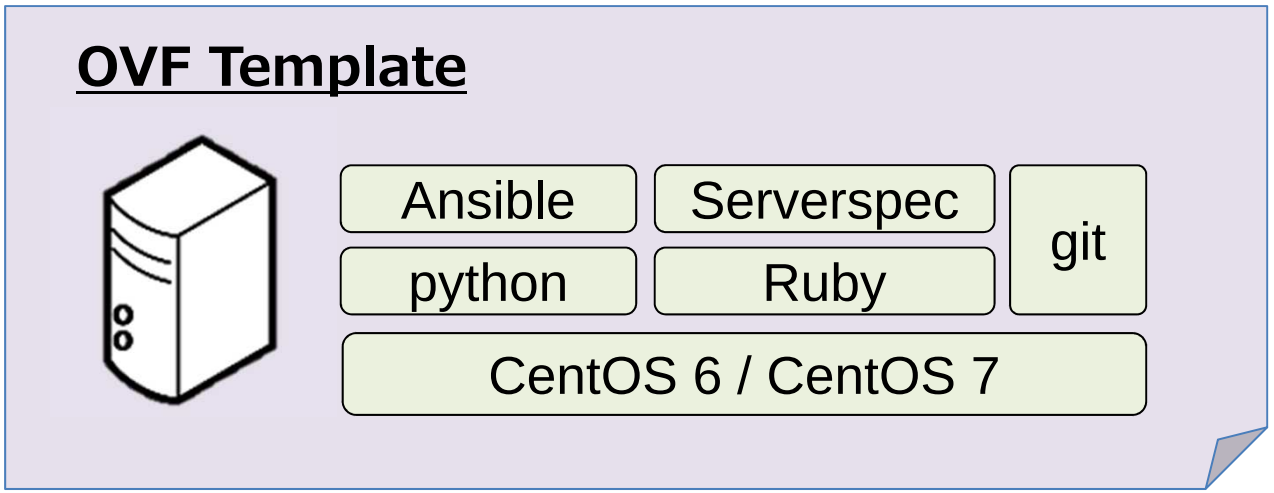
SHIFT Script

実行コマンドをスクリプト化してシンプルに

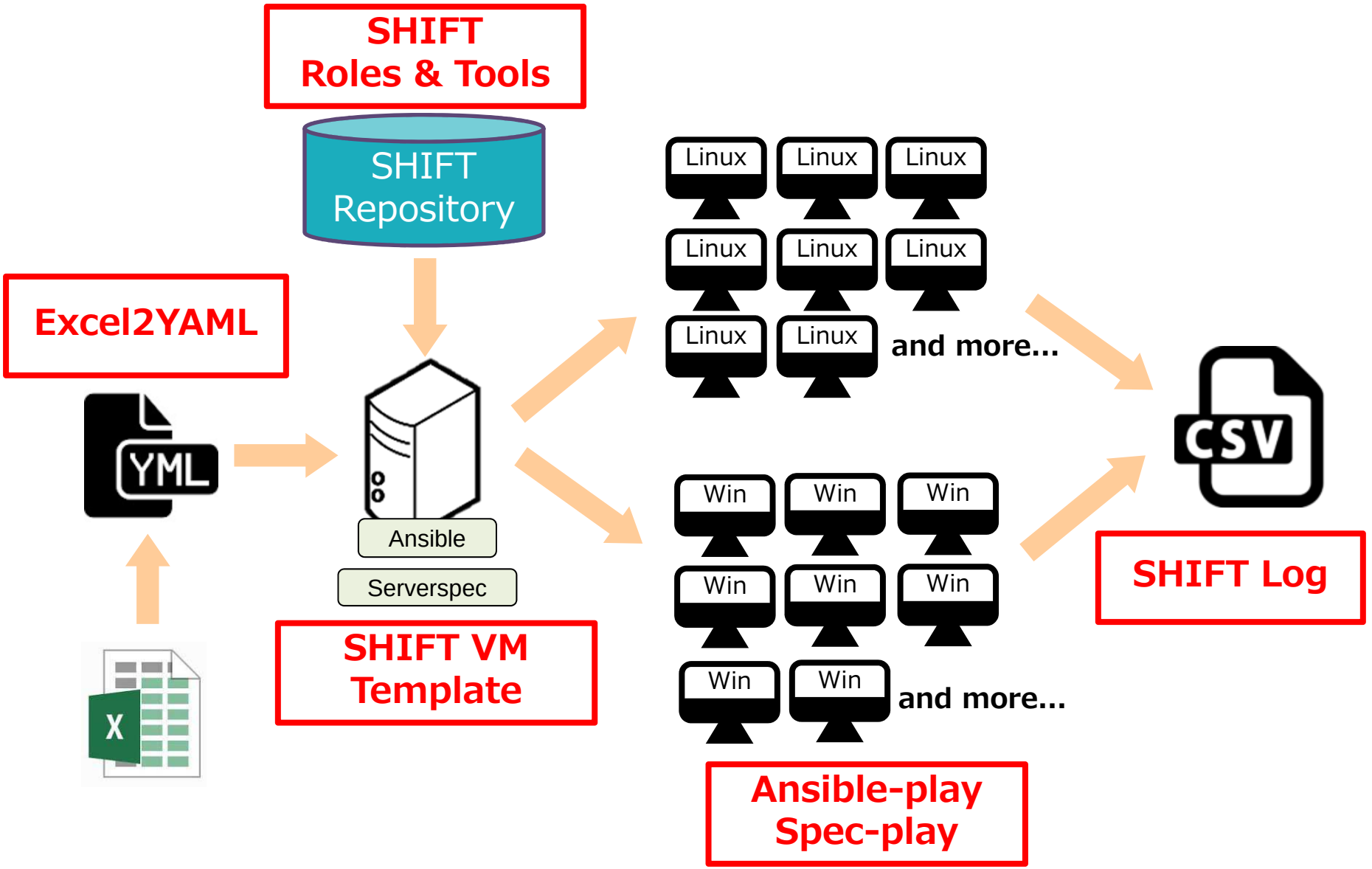
```
# cd $WORK_DIR
# ansible-playbook $SITE –i $INVENTORY –c paramiko
    ⇒ # Shift_Bin/Ansible-play.sh run

# cd $WORK_DIR
# for task in $@ do rake ${task} & wait $! …
    ⇒ # Shift_Bin/Spec-play.sh run
```

SHIFT VM Template

Ansible、Serverspec実行マシンはVMのテンプレートをデプロイしてすぐに利用可能



SHIFTを使ってカンタンIaC



SHIFTの利用シーン

SHIFTの利用シーン

SHIFTは2016/9に本部に公開、2017/1に全社に公開

↓

2017/3現在、およそ**20案件**がSHIFTを利用(把握している限りで)

要件やPJ担当者のスキルに合わせて、様々な形でIaCを実践

SHIFTの利用シーン①

SHIFTをそのまま使う。
SHIFTでできることはSHIFTで、できないことは手動で。

例）

Task	SHIFT	手動
ランレベル	○	
ユーザ・グループ	○	
インターフェース設定	○	
パッケージの追加		○
SYSLOG設定	○	
ログローテーション設定	○	
VNC設定		○
サービス自動起動	○	

SHIFTの利用シーン②

SHIFTをカスタマイズする。
部分的な追加、変更はそんなに難しくない。

例）パッケージのインストール処理をロールに追加

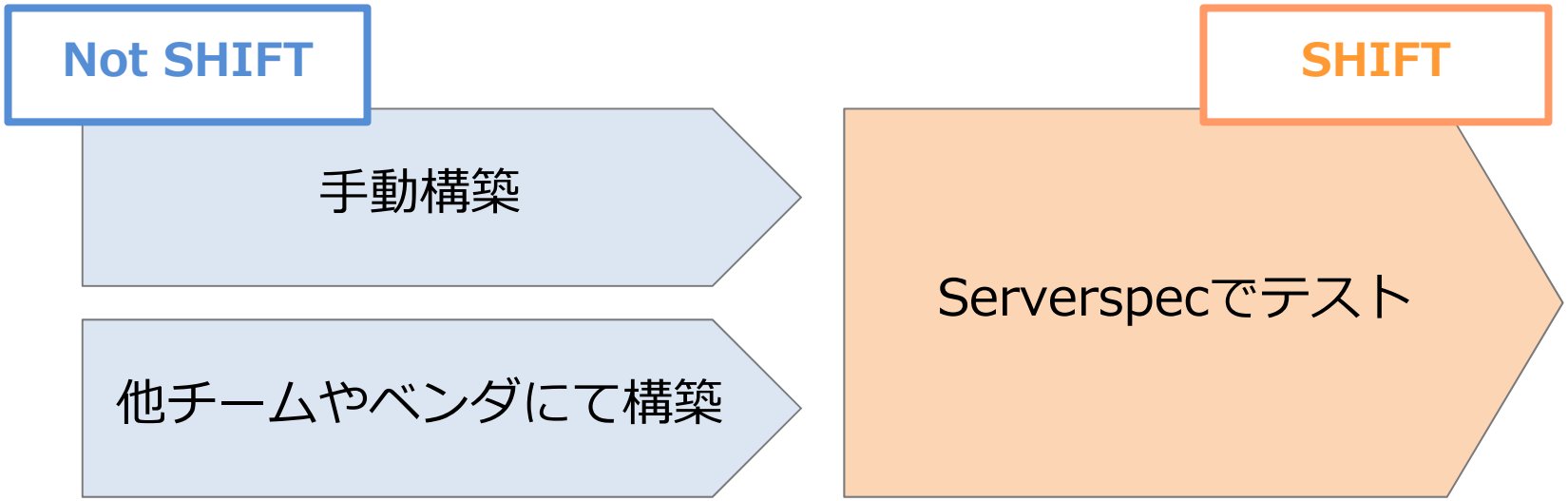
```
# Taskファイルを作成
- name: 1-0001-51_Package Install
  yum:
    name: “{{ item.name }}”
  with_items: base_advanced.packages
```

```
# host_varsを追加
packages:
  - name: ‘httpd‘
  - name: ‘postfix’
```

```
# main.ymlにタスクを追加
- include: ./1-0001-51_Package.yml
```

SHIFTの利用シーン③

テストメインで使う。
手作業で構築した部分のテストや、他者が構築したサーバの受け入れ確認をServerspecで。



IaCのスキルアップに向けて

IaCのスキルアップに向けて

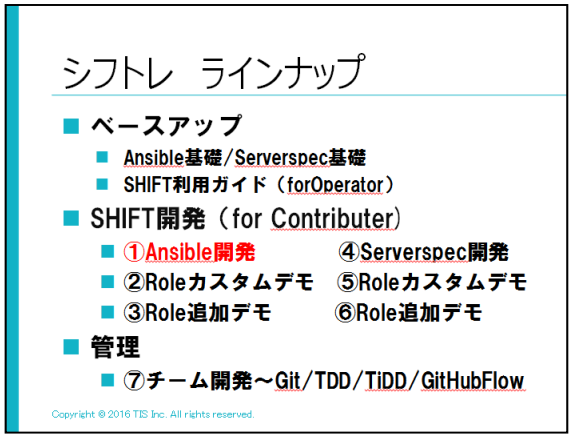
エンジニアのスキルアップに向けて継続的に活動

ハードルDOWN↓
- コードライブラリ
- フレームワーク
- スクリプト
- ツール

スキルUP↑
- **社内勉強会**
- **セルフハンズオン**
- **ナレッジ公開**
- **案件利用支援**

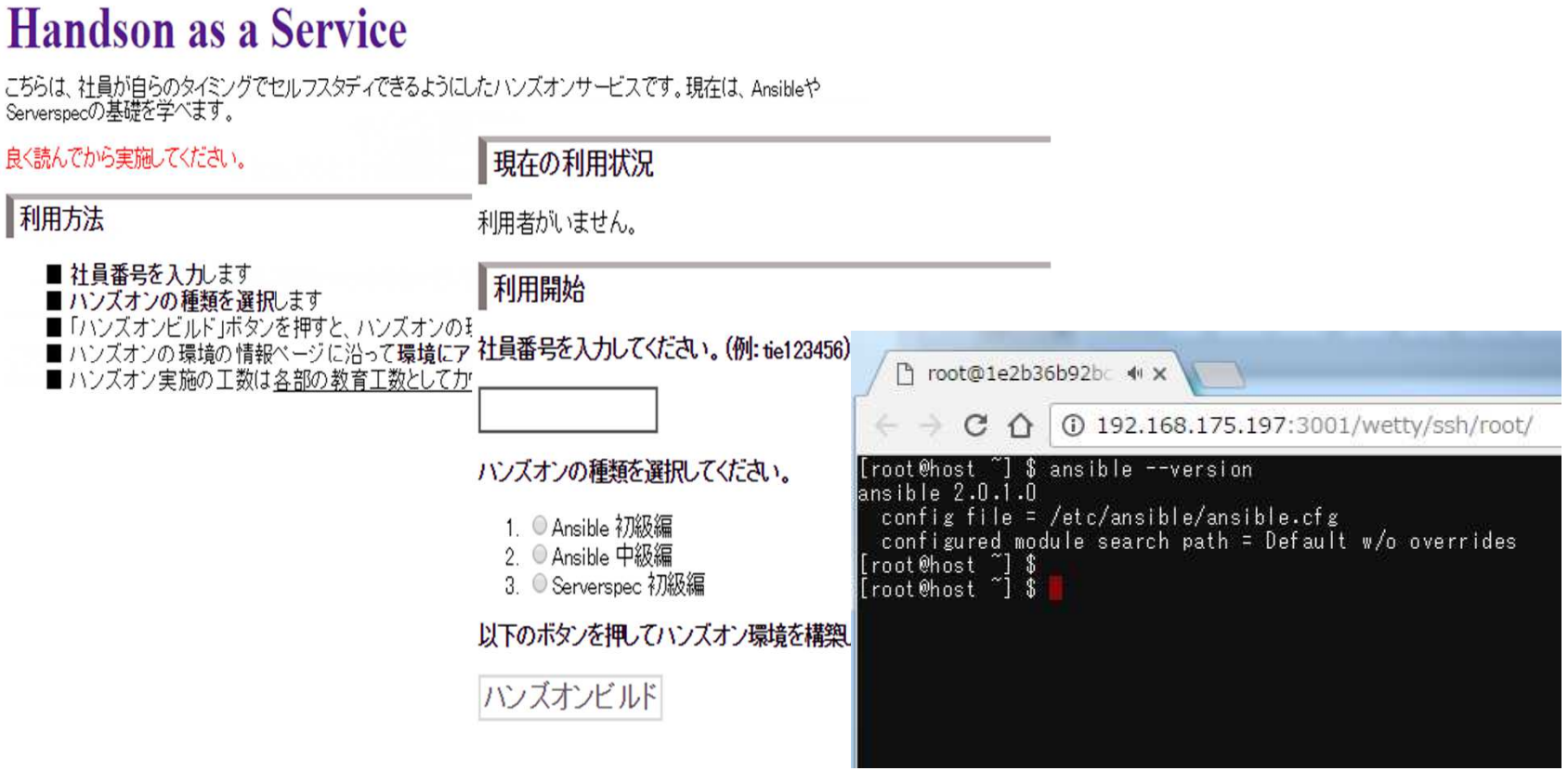
IaCのスキルアップに向けて

- 社内勉強会
  - SHIFT説明会
  - Ansible開発
  - Serverspec開発
- 案件利用支援
  - アセスメント
  - フィジビリティ
  - QA対応、OJT
  - etc...



IaCのスキルアップに向けて

セルフハンズオン
ブラウザだけで何度でも



IaCのスキルアップに向けて

日々の開発、メンテナンスで得たノウハウを Wiki、Tipsとして公開

■ Contents

はじめに

- SHIFT概要
- 利用にあたっての制限・前提条件・ライセンス
- とりあえずためしに使ってみる・ハンズオン
- 用語集

library

コードリポジトリ

- リリースノート・バグ情報
- リポジトリトップのリンク
- RolesList
- ToolsList
- (補足) SHIFTでのRoleとToolの定義・違いについて

ユーティリティ

- UtilityList
  - Exce2YAML
  - ControlMachine VM Template

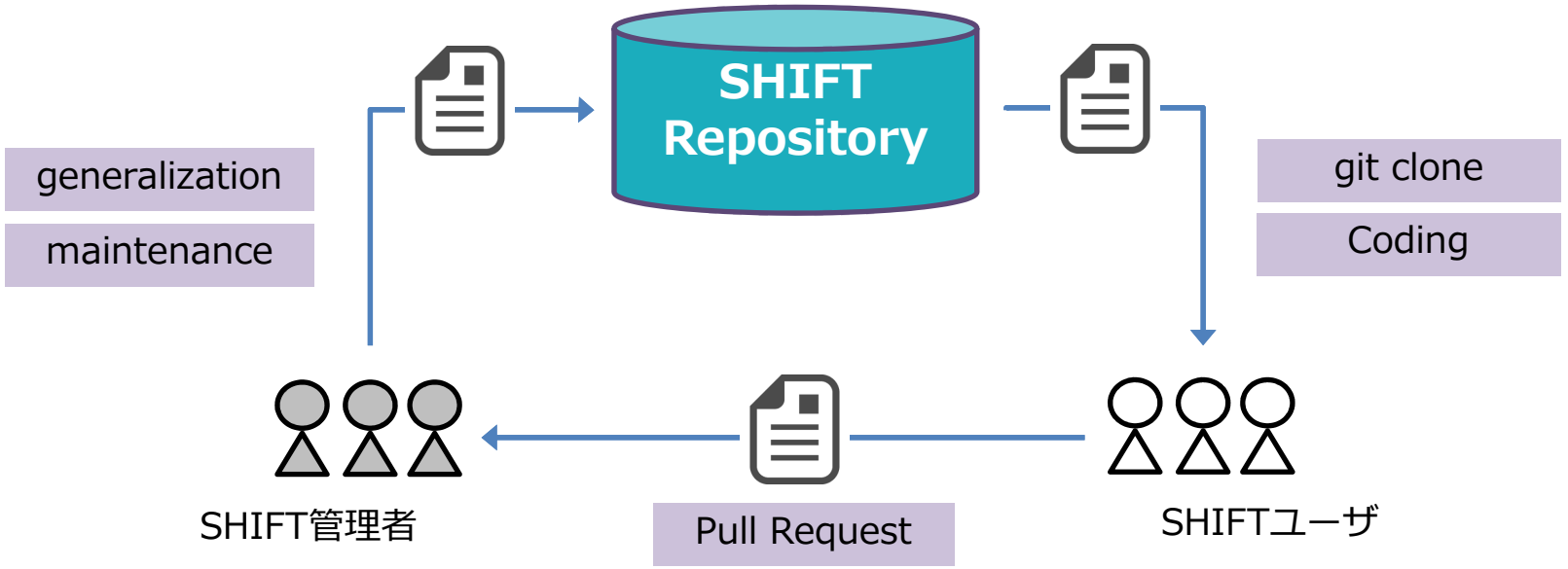
Ansible Tips

目次

- 全般
  - パスワード認証、公開鍵認証の優先度について
  - with_itemsを使用したタスクの結果を、次のタスクで使用する
  - handler の実行順序について
  - varsファイル内で変数を使用する
  - 変数が未定義の場合の扱い
  - ターゲットに生スクリプトを残す
  - int,stringの比較
  - when系の書き方
  - when, failed_when, changed_whenの優先度
- Linux
  - SSHで-kを使用して接続できないエラーについて
  - SELinuxの設定について
  - paramiko接続とOpenSSH接続の違い
  - sudo設定直後のターゲットに接続すると処理が停止する問題について

IaCのスキルアップに向けて

ゆくゆくは、SHIFTユーザの誰もが
「Operator」から「Contributor / Committer」へ



まとめ

まとめ

- インフラの自動化、Infrastructure as Codeを実践できれば、多くの課題はきっと解決
- TISは
  IaCフレームワーク「SHIFT」の開発
  ハンズオンや勉強会などの継続的な育成
  でIaCを推進しています
- これらの活動が実り、TISではIaCが徐々に普及し、働き方がシフトしつつあります

THANK YOU