Ansible　Terraform
技術検証結果報告書

新技術ワーキンググループ　Bチーム(自動化)

KEL

アジェンダ

1. 概要
    - -自動化とは
    - -Ansible
    - -Terraform
2. 検証内容・結果・所感
    - -検証①Ansible
    - -検証②③Terraform
    - -検証④連携
3. まとめ

アジェンダ

1. **概要**
   - -自動化とは
   - -Ansible
   - -Terraform
2. 検証内容・結果・所感
   - -検証①Ansible
   - -検証②③Terraform
   - -検証④連携
3. まとめ

自動化とは

=**IaC**（Infrastructure as Code）

インフラ構成を「**コード**」化して管理する手法。

手作業で発生する問題

・手作業実行によるヒューマンエラー増加
・重複変更による無駄なサービス停止
・対象が増えて管理しきれない

手作業はミスが多くなるし
時間もかかるし最悪..

ソースコード+自動化ツールで管理

・ヒューマンエラー防止
・重複した変更による無駄なサービス停止が減少
・作業の省人化と効率化（コスト削減）

ミスが減り、
膨大なサーバ管理も
簡単にできる！

Ansibleとは(概要)

→**Red Hat社**によってオープンソースとして公開された**構成管理ツール**

【構成管理ツールとは】

IT環境構築を自動化するプラットフォームであり、多数のアプリケーションとシステム導入を容易にする。

○特徴

- **一括管理**

  Ansibleがインストールされたサーバから一括で他の複数台のサーバに対して設定が可能。

- **オープンソースソフトウェア**

  カスタマイズが容易なため導入コストを低く抑えられる。

- **エージェントレス**

  Ansibleからネットワーク経由でマシンに繋がる限り事前準備不要

  ※Windowsノードでは導入作業が必要

Ansibleとは(事例)

◇BPOサービス業

背景	Windows7のサポート終了に伴い、PC5000台をWindows10に移行したい
課題	◆人的ミスの発生 →PCセットアップは1台ずつ手作業で実施 ◆PC運用管理の人員不足 →通常業務や障害対応に追われ、移行作業に人員を割くことが難しい ◆コストの増加 →移行を外部委託した際、新たなコストの発生 →初期セットアップ作業は既に外部委託をしている
Ansible導入	
結果	◆"playbook"による、設定作業の自動化 →**人的な設定ミスが 0** ◆PC100台のセットアップ作業の必要工数は4～5人月 → **1人月で可能** ◆自社で移行とセットアップの対応が可能に →**外注のコストは発生せず** →**既存コストの削減**

Terraformとは(概要)

→**HashiCorp**社によりオープンソースとして公開された**インフラストラクチャプロビジョニングツール**

【インフラストラクチャプロビジョニングツールとは】

インフラをコード管理することで、安全かつ効率的な構築・変更・バージョン管理を実行可能とするツール。

構成管理と組み合わせて利用でき、管理台数が増えるほど効果を発揮する。

○特徴

・マルチクラウド

クラウドサービスを提供する各社も独自の自動化ツールを提供してはいるが、Terraformを使用することで様々なクラウド環境の一括管理が可能。

・宣言的

最終的なプログラムのあるべき姿を宣言するのみで良い。

・リソース間の依存関係/オーケストレーション

各リソースの依存関係を把握しており、設定を変更した際の影響を事前に把握

Terraformとは(事例1)

◇情報・通信業

背景	開発組織の人員増加と併せ、マイクロサービスアーキテクチャを採用したい
課題	◆リソース管理が複雑 →クラウド環境にサービス単位でプロジェクトを分けなければならない ◆開発スピードの高速化 →人員増加に見合った開発環境が求められる

Terraform導入

結果	◆プロビジョニング作業を自動化 →**管理コストが低い** ◆インフラ管理を開発者自身でできる →**スピード感のある開発**が可能に

Terraformとは(事例2)

◇情報・通信業

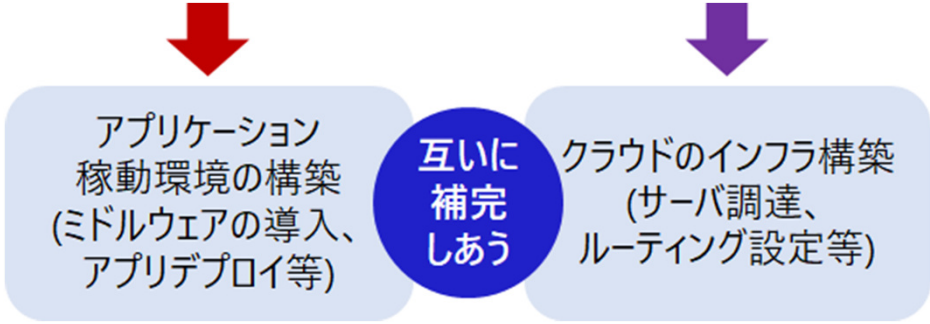
背景	業務環境のマルチクラウド化
課題	◆管理するサーバの数が多い →ミスが発生しやすい ◆クラウドごとにGUI画面仕様が異なる →開発者の学習コストが高い ◆各サーバの設定内容の管理が複雑 →作業ミス時に設定を戻すことが困難

Terraform Enterprise導入

結果	◆チームメンバーの作成したコードのポリシーチェックが可能 →操作・作業ミスなどからインフラを破壊してしまう**事故を防ぐ** ◆開発者はTerraformのコードのみ学習すればよい →**学習コストが低い** ◆プログラムのバージョン管理を行える →**障害対応が容易**

AnsibleとTerraformの比較

	Ansible	Terraform
リリース時期	2012年	2014年
開発社	Red Hat	HashiCorp
メイン機能	構成管理	インフラストラクチャ プロビジョニング
得意な管理	ソフトウェア	クラウドリソース
クラウドプロバイダ対応数	41	300以上
構成記述方式	YAML	HCL
情報量	英語/日本語とも多い	日本語情報は少ない



アジェンダ

1. 概要
    - -自動化とは
    - -Ansible
    - -Terraform
2. **検証内容・結果・所感**
    - -検証①Ansible
    - -検証②③Terraform
    - -検証④連携
3. まとめ

検証案

【検証目的】

1.KELビジネスにおいてAnsibleを扱うために必要な知見を得る
2.Terraformを扱うために必要な知見を得る
3.検証目的1および2で得た知見を基にTerraformとAnsibleの連携を検証する

【検証期間】

2020年10月6日～2021年2月末

【検証概要】

検証①：Ansibleを用いてAzure上で仮想マシンの作成/管理を行う
検証②：Terraformの基本的な挙動を検証する
検証③：Terraformを用いてAzure上で仮想マシンの作成/管理を行う
検証④：AnsibleとTerraformを組み合わせてAzure上で仮想マシンの作成/管理を行う

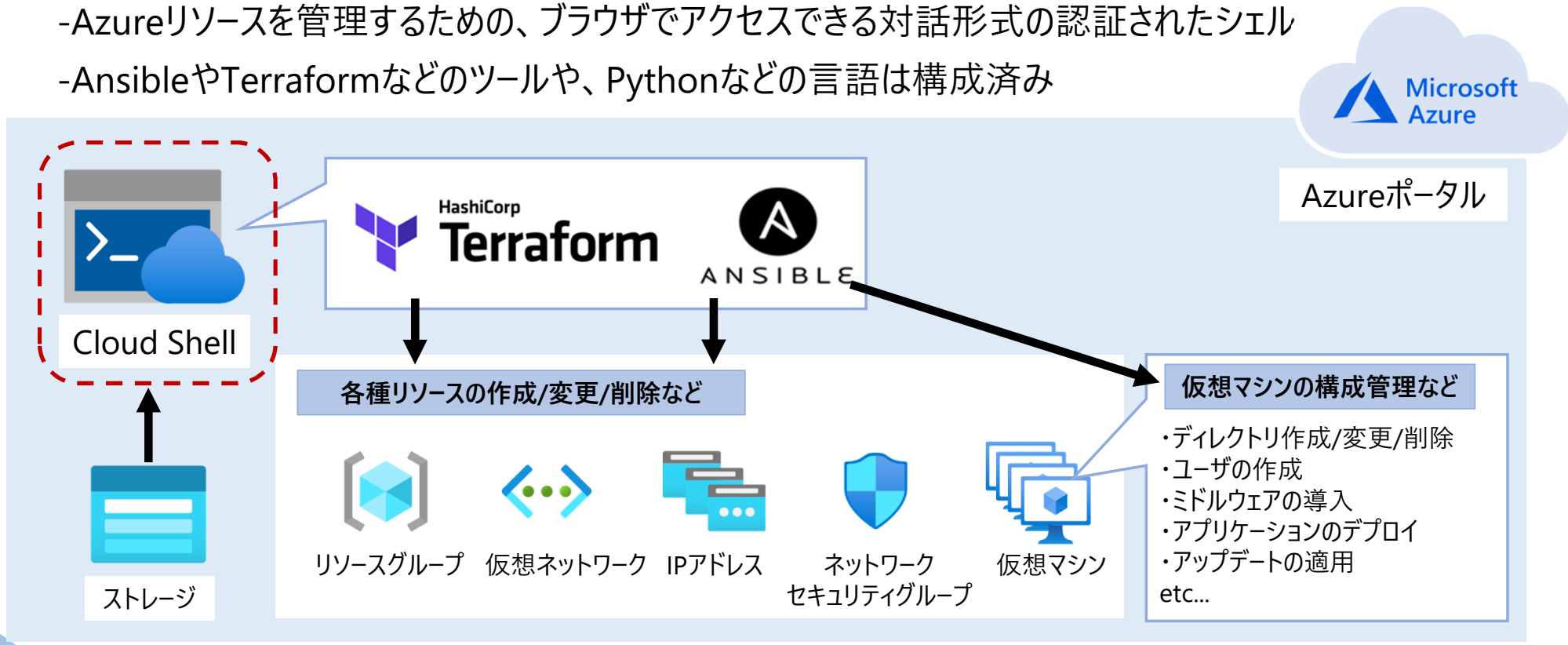
KEL

検証環境準備

AzureCloudShell環境構築（作業内容：Azureポータルからストレージをマウント）

-Azureリソースを管理するための、ブラウザでアクセスできる対話形式の認証されたシェル

-AnsibleやTerraformなどのツールや、Pythonなどの言語は構成済み



検証①Ansibleでの仮想マシン作成/管理

【検証】

・Ansibleを用いてAzure上で仮想マシンの作成/管理を行う

【目標】

・Azure上への仮想マシン作成がplaybookの実行のみで行えることを確認できること

・Azure上の仮想マシンへの設定変更/管理がplaybookの実行のみで行えることを確認できること

【検証に必要なもの】

・playbook(YAMLファイル)

検証①Ansibleでの仮想マシン作成/管理
評価・所感・考察

・既存リソースに対し自動で結びつけを行う（＝認識させる作業が不要な）ため、構築が容易。

・仮想マシンの構築時間はOSによる差が小さい。

・記述量が少なくシンプルなためYAMLファイル作成が容易。
ただし記述方式から少しでも外れるとエラーが出るため融通性は低い。

・学習コストを考えると少数マシンの自動化には非効率である。

検証②Terraformでの基本的な挙動確認

【検証】

・Terraformの基本的な挙動を確認する。

【目標】

・簡易的なタスクが記載されたtfファイルを作成し、Azureに対してタスクの実行が可能であるのかを確認する。

【検証に必要なもの】

・tfファイル

検証③Terraformでの仮想マシン作成/管理

【検証】

・Terraformを用いてAzure上で仮想マシンの作成/管理を行う

【目標】

・Azure上への仮想マシン作成がtfファイルの実行のみで行えることを確認できること

・Azure上の仮想マシンへの設定変更/管理がtfファイルの実行のみで行えることを確認できること

【検証に必要なもの】

・tfファイル

検証②・③Terraformでの仮想マシン作成/管理
評価・所感・考察

・既存リソースとの結びつけには手動でインポートが必要なため自動化の恩恵が薄い。

・エラー内容が分かりやすく修正作業の負担は比較的軽い。

・実行までの過程が定められており、実行直前の確認(plan)で成功した場合にも実行(apply)でエラーが発生するため、過程が設けられている意図が不明。

検証④AnsibleとTerraformの連携

【検証】

AnsibleとTerraformの双方を用いて仮想マシンの作成/管理を行う

【目標】

- AnsibleとTerraformを組み合わせてAzure上で仮想マシンの作成/管理を行う

【検証に必要なもの】

・tfファイル(仮想マシンを作成し、playbookを呼び出すremote execコマンドを記載)

・playbook(上記tfファイルで作成した仮想マシン内に新規ディレクトリ作成コマンドを記載)

⇒エラー発生

```
Error: Error running command 'ansible-playbook -i inventory mkdir.yml': chdir ~/kenshou4/ansible: no such file or directory. Output:
```

検証④ AnsibleとTerraformの連携
エラー原因と結果

```
Error: Error running command 'ansible-playbook -i inventory mkdir.yml': chdir ~/kenshou4/ansible: no such file or directory. Output:
```

エラー内容：作成対象のファイルやディレクトリが存在しない

・原因(技術)

⇒1つのtfファイルに仮想マシン作成とその仮想マシンへの設定変更を記述すると、Terraformによる仮想マシンのプロビジョニングが完了する前に、Ansibleがplaybookを実行してしまうため「対象が存在しない」とエラーが発生した。

・原因(その他)

⇒AnsibleとTerraformの連携については英語/日本語ともに参考資料が少なく、公式ドキュメントにエラーの解決方法が記載されていない場合は、自身で挙動確認及びエラー解決しなければならずプロジェクト期間内では解決できなかった。

検証④ AnsibleとTerraformの連携 評価・所感/考察

- AnsibleとTerraformを連携する公式ベストプラクティスがない。

- Ansible、Terraform双方の学習コストを考慮すると、使用用途に合わせどちらかを単体で活用した方が効率的である。

- Tfファイルにより仮想マシンが完成したあと呼び出したplaybookにエラーが発生した際、再度tfファイルを実行すると仮想マシンンを削除したあと作成し直すため、通常の倍時間を要する。

※補足

- AzureCloudShellは自身らでバージョン管理できないため、検証環境として不向きである。

アジェンダ

1. 概要
    - -自動化とは
    - -Ansible
    - -Terraform
2. 検証内容・結果・所感
    - -検証①Ansible
    - -検証②③Terraform
    - -検証④連携
3. **まとめ**

技術的な評価

項目		Ansible	Terraform
記述	書き易さ	シンプルで可読性が高い	変数を使用するため複雑
	リソースの依存関係	記述順序が厳しい	記述順序は緩い
	ルール	改行やインデントが厳しい	比較的緩い
	既存リソース	インポート不要	手動インポートが必要
エラー	表示	解読しづらい	行数まで細かく表示
実行	実行方法	実行のみ	init/plan/apply必要
	構築時間	短時間 OSによる差異は少ない	Windowsマシン1台構築の 所要時間は約3分

総括

■商材としての評価

⇒IaC(自動化ツール)を案件の構築業務で活用、知見をもとに教育ビジネスの展開

⇒Terraformは日本語の資料や、ユースケースが少なく学習コストが低いとはいえない

⇒Terraformは開発が進み挙動が安定するまでは使用をお勧めしない

■今後について

⇒AnsibleとTerraformの連携を継続して検証

⇒自動化ツールによるオンプレミス機器の設定

KEL

兼松エレクトロニクス株式会社

http://www.kel.co.jp