

「IaC活用研究会」(第2回)

IaCの取り組みと実績

グローバルデプロイ事業部

宮崎 剛, 若林寛之

2018年4月18日

DELL EMC

自己紹介

宮崎 剛

2004年12月よりEMCジャパンに勤務
シニアソリューションアーキテクト

主な仕事

- EMC製品(主にストレージ)設計に従事
- 主に仮想化・自動化系ソリューションを担当

若林 寛之

2015年4月よりEMCジャパンに勤務
インプリメンテーションスペシャリスト

主な仕事

- EMC製品(主にストレージ)構築に従事
- 主にソフトウェア系製品を担当



Agenda

1. IaCの取り組み
2. IaCの実績と学び

IaCの取り組み

1

取り巻く環境の変化

弊社に対するリクエストにも、IaCで改善するような項目が増えている

お客様の声

作業時のエラーを低減し、品質を高めたい

ドキュメントと実機で構成が異なることを防ぎ、最新の構成情報を取得したい

属人化、品質のばらつきを極小化したい

基盤運用業務を効率化したい

- ✓ ハードウェア増設時の設定自動化
- ✓ 仮想マシン構築と導入ソフト設定自動化
- ✓ 容易な構成変更 など

構成管理を効率化したい

- ✓ 機能アップデートパッチ適用の効率化
- ✓ セキュリティパッチ適用の効率化

パッチ適用対象選択をワンクリック(イメージ)で実施したい

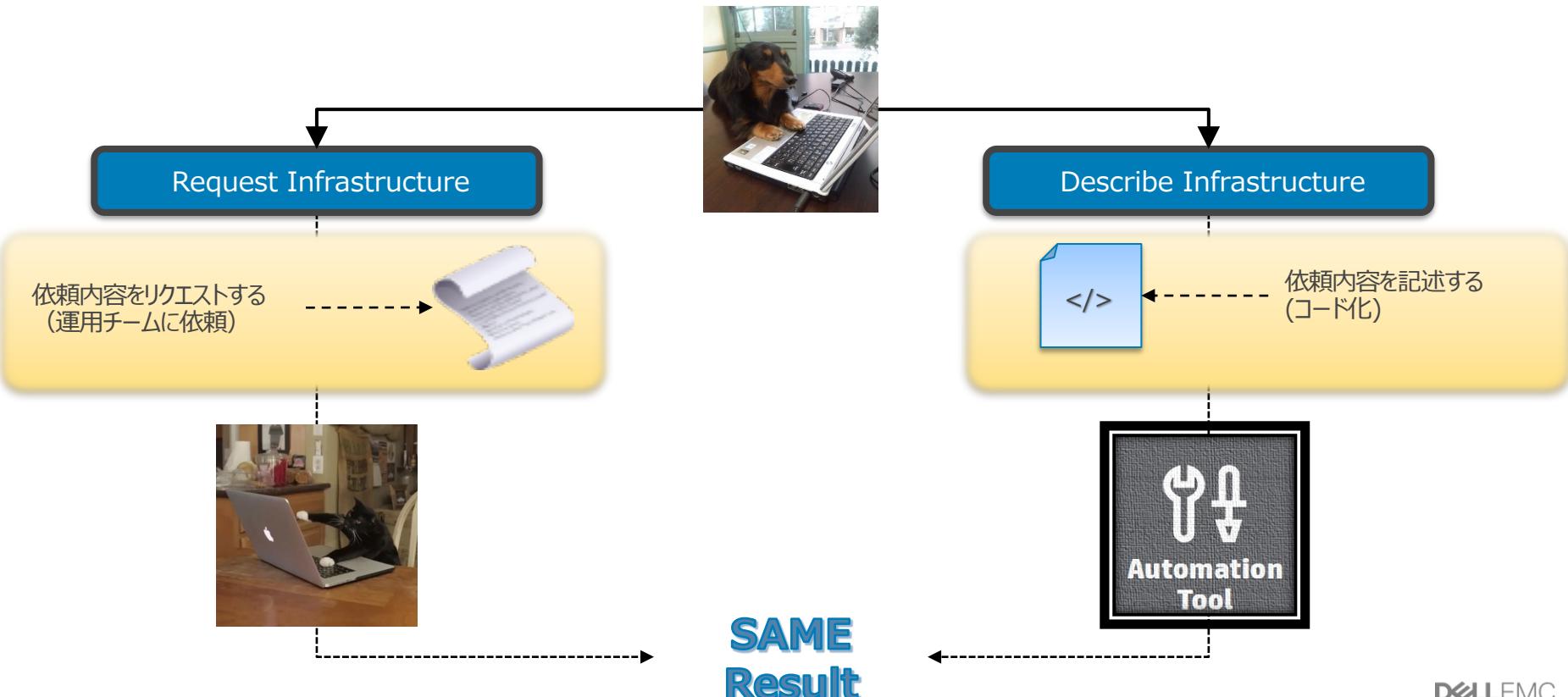
各機器に導入されているソフトウェアの情報を収集できること。ソフトウェアのバージョン情報等について構成管理できること

仮想サーバ構築時のエージェント/SWのインストール/設定の自動化

パッチ適用後におけるテストを自動化したい

Infrastructure as code

手作業で行っていたインフラの構築や変更作業をコードで定義／実装／表現



DELL EMCが開発をリードしているOSSツール例



REX-Ray

永続ストレージをDockerコンテナにベンダー依存せずに提供する、Linux環境におけるストレージ管理用ツール。

<https://github.com/rexray/rexray>



RackHD

ベアメタルサーバに対して、OSのインストール、ノードをリブート等のタスク実行を可能にする管理ツール。

<https://github.com/RackHD/RackHD>



SCALEIO

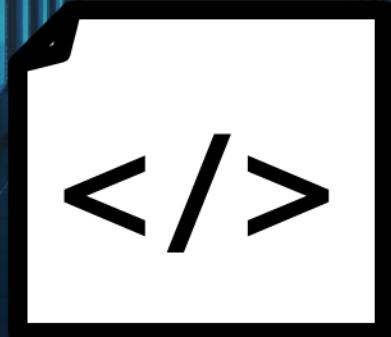
ansible-scaleio

Ansibleを利用して、EMCのSoftware Defined StorageであるScaleIOを構築するツール。

<https://github.com/sperreault/ansible-scaleio>

目指すべき方向性

Infrastructure as codeで
データセンター全体を効率化する！！



IaCの理想と現実

理想

IaCで工数削減！
IaC最高！！

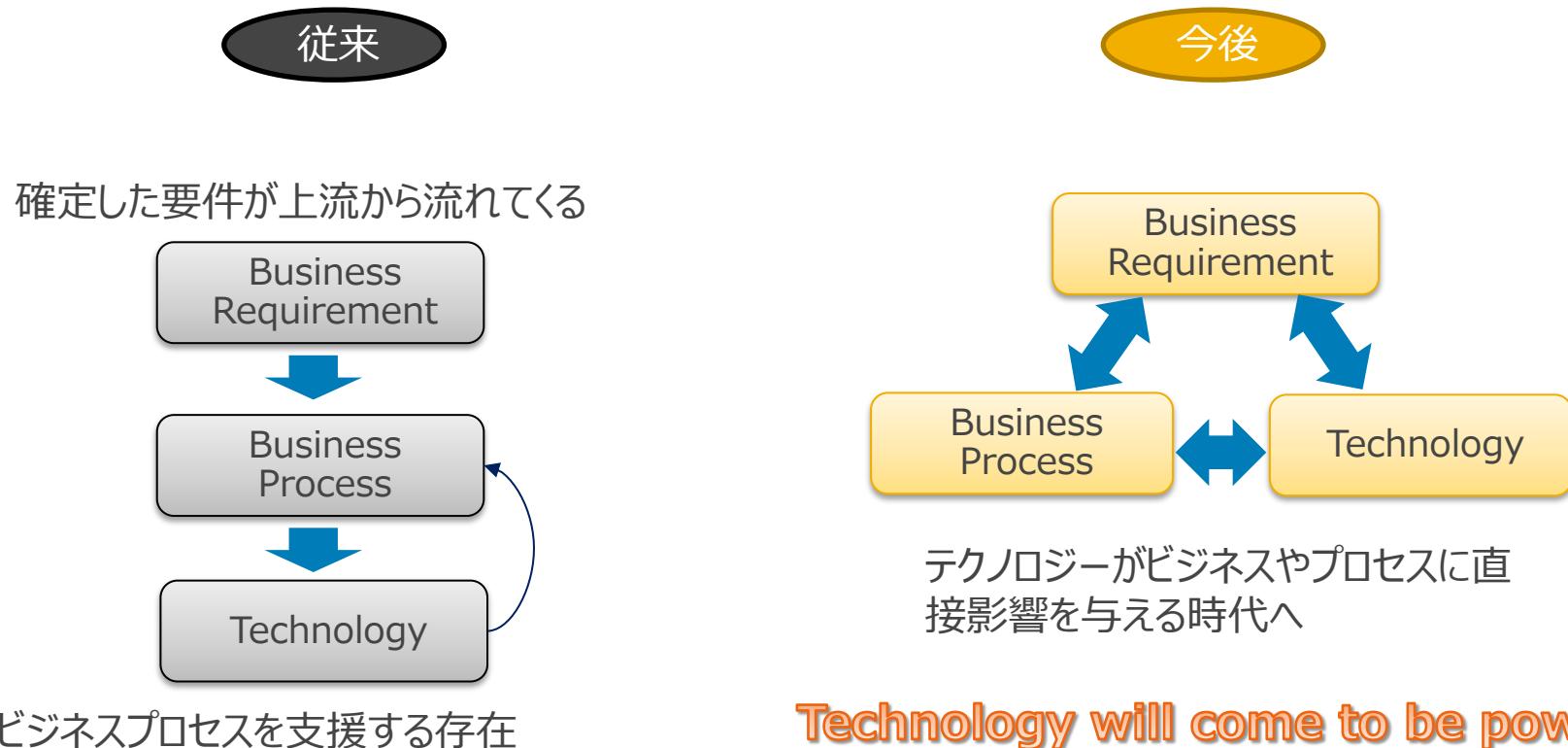
ギャップ

簡単
ではない

現実

試行錯誤で進めて
いるのが現状

ビジネスとテクノロジーの関係の変化

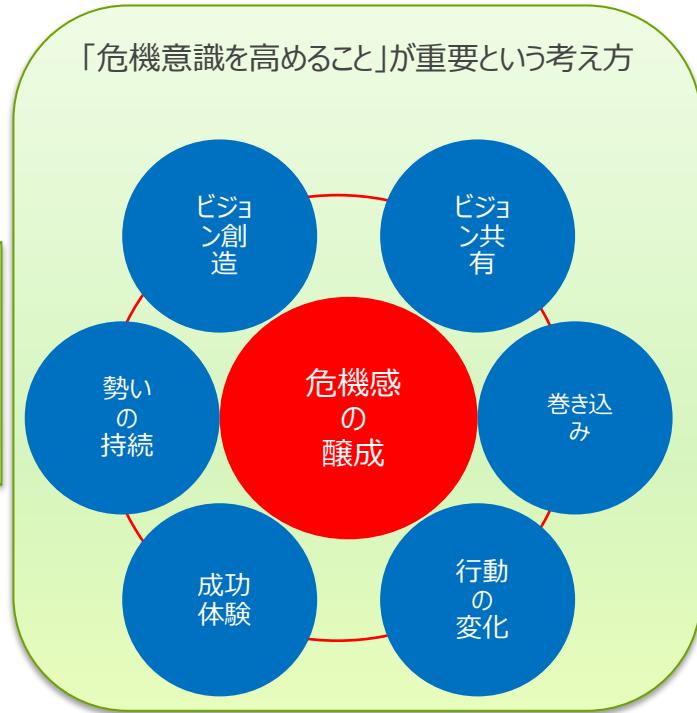


IaCを実現するために重要なと考へる事

3つの要件



※出典
森時彦「リーダーのための実践ファシリテーション」(BPUビジネス基礎講座)より



風土醸成するためには？

組織によって目標／時間軸が異なる
組織の数だけ風土が異なる

問題の複雑化／異種専門家チームの協働

チームによる問題解決／知識創造
チームによる合意形成を促進する技術
チームにおいて知的相互作用を促進する

発想を促す

コミュニケーションを
促す

行動を促す

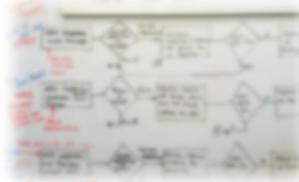


ファシリテーション・スキルを要する

ファシリテーションは、コミュニケーションのインフラ的な役割を果たす

具体的な実践方法

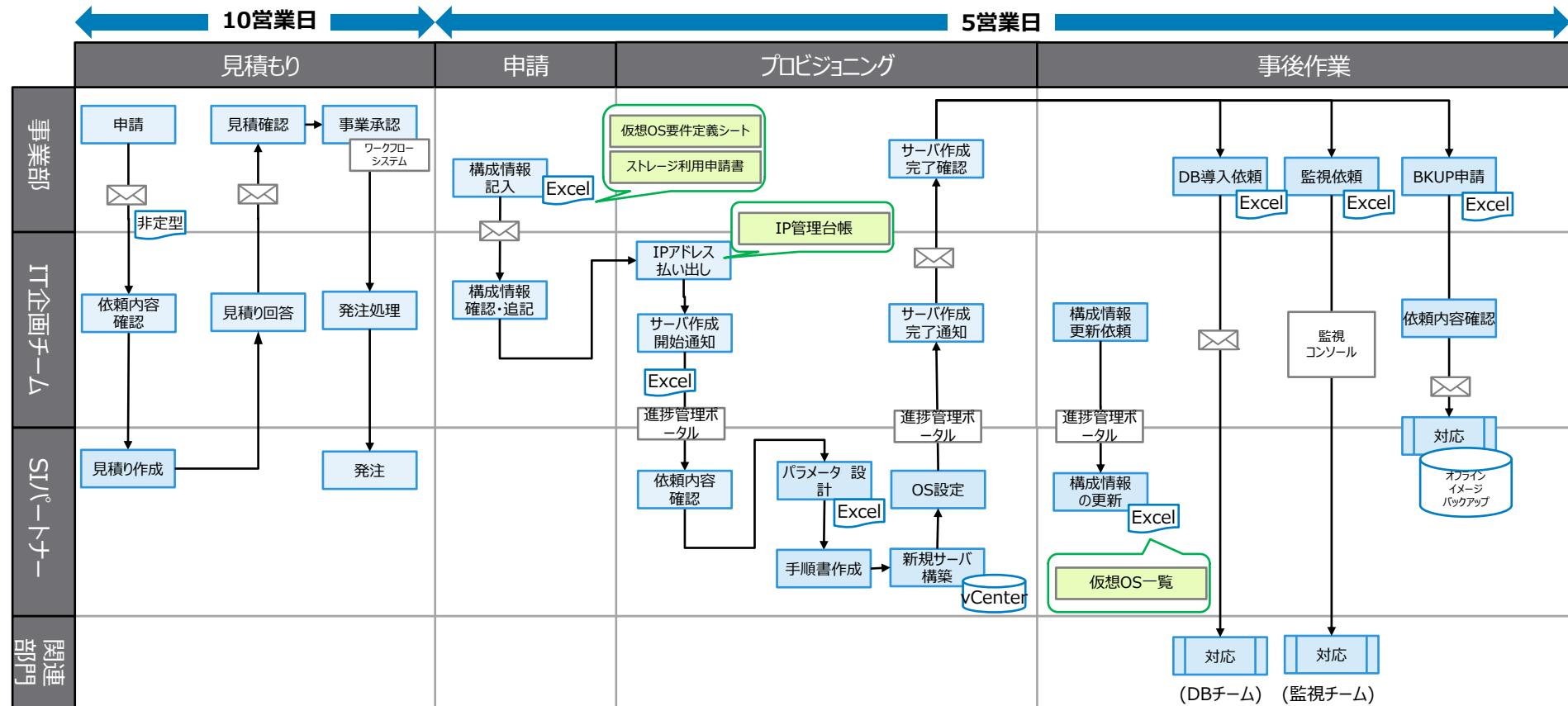
深掘りするテーマを決め、関係するアクターに参加してもらう



ビジネスプロセス責任者、テーマのビジネスフローをよくご存じの方に、可能な限り、参加いただき、それぞれの立場からよりプライオリティの高い課題へ認識を共通化してからチームで取り組んでいく。

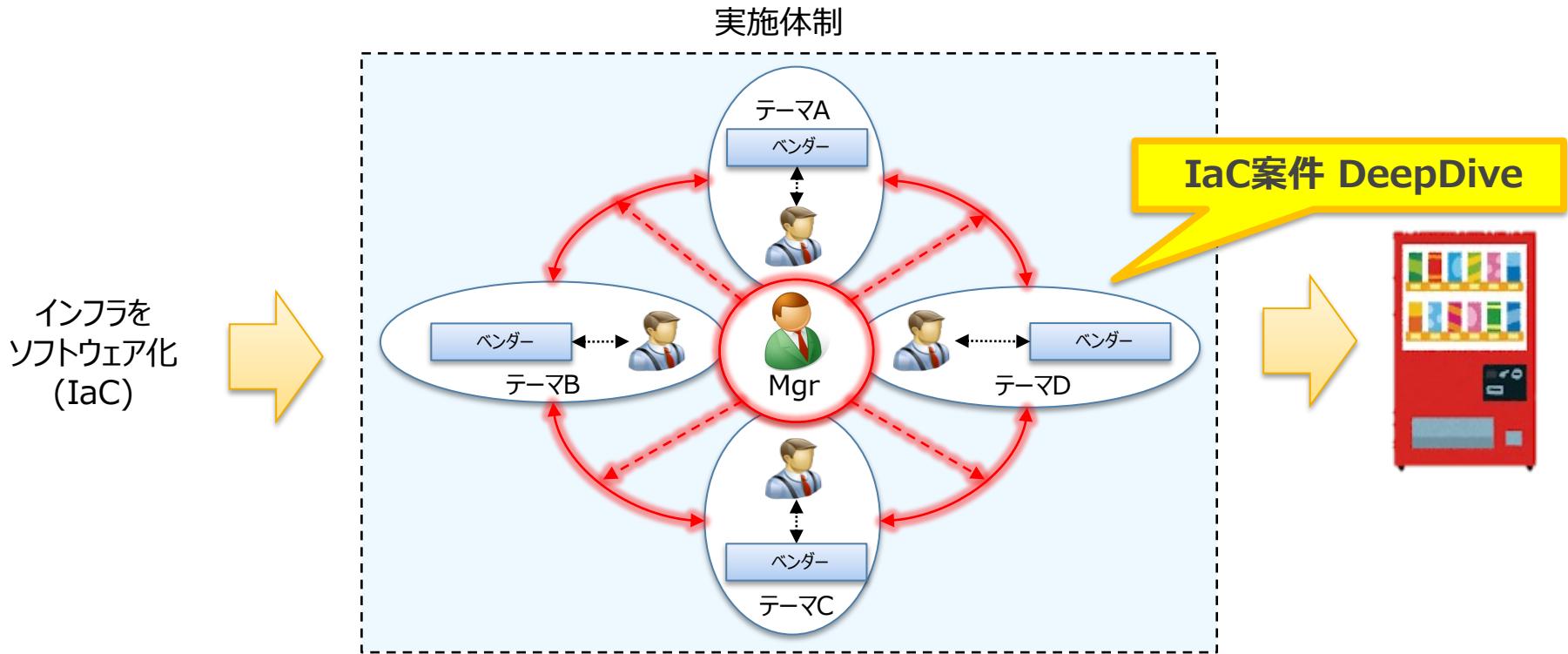


検討資料の例（プロセスマッピング的な）



国内大手企業での成功事例

プロセスの改善で効率化が実現した！



2

IaCの実績と学び

DELLEMCにおけるIaCの取り組み(国内事例)

時期	プロジェクト概要	詳細
2016Q1	SDS導入自動化	Ansibleを利用した大規模ScaleIO※1環境の継続追加の自動化
2016Q2	サーバ導入自動化	RackHD※2、vSphere PowerCLIを利用したベアメタルからESXサーバ構築までの自動化
2016Q4	ストレージ作業自動化	Ansibleを利用したVNX※3ストレージのマスキング作業の自動化
2017Q1	ストレージ作業自動化	Pythonを利用したXtremIO※4ストレージのインストール作業の自動化
2017Q3	Kubernetes環境構築自動化	GitLabとAnsibleを連携することで、DELL GPUサーバ上にScaleIOとkubernetesクラスタ環境の自動構築
201710-継続中	OSパッチ管理自動化	Puppetを使ったOSパッチにおける構成確認、検証環境作成、適用、確認の自動化

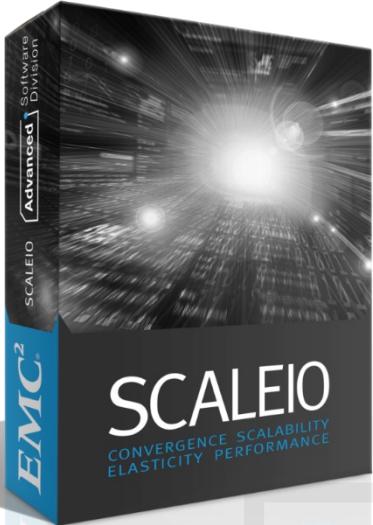
※1：コモディティハードウェアでスケールアウト対応のSDSを実現するソフトウェア。

※2：ベアメタルサーバを構成管理できるツール。

※3：DELLEMCのミッドレンジストレージ。

※4：DELLEMCのフラッシュストレージ。

ScaleIOとは

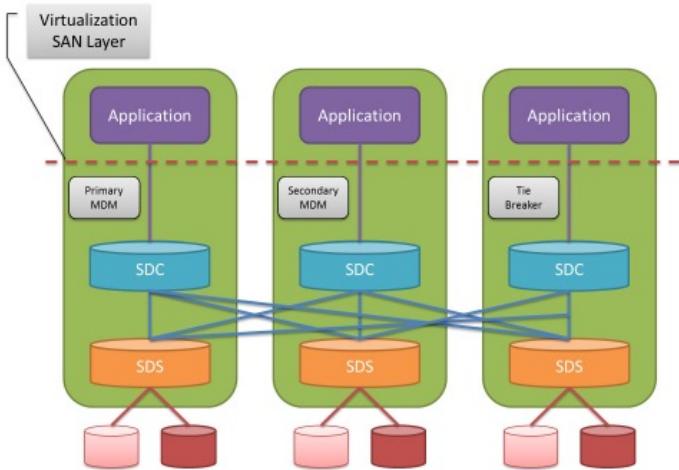


- コンポーネントの一部
 - 管理 : MDM(Meta Data Manager)
 - ストレージ : SDS(ScaleIO Data Server)
 - クライアント : SDC(ScaleIO Data Client)
- All-in-one構成/分離構成が可能

ソフトウェア的にサーバ内蔵ディスクを束ね、
ブロックストレージを構成するSoftware Defined Storage

ScaleIOのインストール作業

- 手動構築で必要になるrpmファイル



EMC-ScaleIO-mdm-1.32-4503.5.el6.x86_64.rpm
EMC-ScaleIO-sdc-1.32-4503.5.el6.x86_64.rpm
EMC-ScaleIO-sds-1.32-4503.5.el6.x86_64.rpm
EMC-ScaleIO-callhome-1.32-4503.5.el6.x86_64.rpm
EMC-ScaleIO-gateway-1.32-4503.5.noarch.rpm
EMC-ScaleIO-lia-1.32-4503.5.el6.x86_64.rpm
EMC-ScaleIO-tb-1.32-4503.5.el6.x86_64.rpm
RPM-GPG-KEY-ScaleIO_1.32.4503.5
sdc-1.32.4469.5-esx6.0.zip

大量のrpmファイルを適切にサーバに配置し、展開することが必要。。。

案件の概要と背景

■ 概要

- ・ サービス内容：Software Defined Storage(SDS)構築支援サービス
- ・ 規模：サーバ1000台以上(OpenStack/vSphere基盤)

■ 背景

- ・ お客様にて既に一定水準の内製開発に取り組み
 - ✓ 「有償製品/ハードウェアアプライアンス製品→OSS/ソフトウェア製品」への変革中
 - ✓ 既に構築用(OSインストール)のポータルを作成開始
 - ✓ EMC製品ScaleIOを利用したSDS化を図る
- ・ 当時のansible-scaleioプロジェクトからのFork
 - ✓ Globalで取り組んでいたもののローカライズ/日本での実績作り
 - ✓ Ansible社から提供されていたansible-vsphereモジュールの存在/実装にあたっての検証作業

共同開発の要素が強かった

目的	対応策(製品)
構築・構成管理	Ansible + ScaleIO
構成情報・性能情報の取得	Python + ScaleIO
構成情報・性能情報の可視化	Kibana + ElasticSearch
監視	Python + ScaleIO + Zabbix

Step : インフラ構成の決定

■ 通常のインフラ設計とは異なる、IaC特有の考慮点

- コンポーネントを分離するか否か
⇒ 「分離構成は要件をすべて満たせるか？」
⇒ 「不足・漏れはないか？」
- 対冗長性・パフォーマンス要件をどう満たすか
⇒ 「どうすれば壊れない？」
⇒ 「かつ、パフォーマンスを最大化できる構成は？」

⇒ 「既存Playbookを流用しやすいから分離構成を採用」
まず自動化された仕組みを作る(ソフトウェア化する)

⇒ 「壊れる前提でその影響をゼロにするためには？」
自動化が生んだ余剰でパフォーマンスの最大化を考える

- ▼
- 「要件は不变」
 - 要件に対する手戻りのなさ・網羅性を重視

- ▼
- 「要件は変わりうるもの」
 - 要件に対し優先度を持たせる、スピード感を重視

構成を決定する要素は「自動化のしやすさ」や「手間の少なさ」

構成決定後、やらなくてはいけないこと…

Task	Sub-Task	Script	Status	avail	commands		
ESXi の設定	SSHアクセス有効化	vim-cmd	*kickstartより指定				
	ESXiシェルアクセス有効化	vim-cmd	*kickstartより指定				
ホスト名/DNS設定	ansible2.0		*検証済み	○	vmware_dns_config		
管理ネットワーク用ネットワークアダプタの選択	vCLI(esxcli)			○	esxcli network ip interface set/ esxcli network nic set/ esxcli network nic up		
VLAN ID の設定	vCLI(esxcli)	#vssおよびpg必要		○	esxcli network vswitch standard portgroup set -p <portgroup> -vlanid <vLAN ID>		
ESXi の IP 設定の構成	vCLI(esxcli)			○	esxcli network ip interface ipv4 set		
ESXi の DNS の構成	vCLI(esxcli)			n/a	n/a(vmware_dns_configで同時実行可能)		
IPv6 無効化	vCLI(esxcli)			○	esxcli network ip set -e false		
ESXi ホスト syslog の構成	vCLI(esxcli)			○	esxcli system syslog config set		
ESXi Host ID取得	vCLI(esxcli)			○	esxcli system hostname get / esxcli system uuid get		
NTP設定	vCLI(esxcli)						
SSH Warning 抑止設定	vCLI(esxcli)			○	esxcli system settings advanced set -o /UserVars/SuppressShellWarning -i 1		
PasswordAuthentication有効化	△	PowerCLI実績のみ					
memory-cache-size-cap容量変更	△	PowerCLI実績のみ					
ドメイン認証設定	△	PowerCLI実績のみ					
ローカルセキュリティ認証サーバ自動起動	△	PowerCLI実績のみ					
SSHのrootログインを不可	Linuxコマンド(sed)				sed -i -e "s/PasswordAuthentication no/PasswordAuthentication yes/g" /etc/ssh/sshd_config		
メンテナンスマード解除							
vCenter/Cluster設定	データセンター作成	Task	Sub-Task	Script	Status	available	commands
クラスタ作成	vSphere ネットワーク設定	vSphere ネットワーク設定	標準スイッチの構成(vSwitch)	ansible2.0	*検証済み	○	vmware_vswitch
ESXi Host追加			ポートグループの構成	ansible2.0	*検証済み	○	vmware_portgroup
データストアへのファイルアップロード			分散仮想スイッチの構成(vSphere Distributed Switch)	ansible2.0	*検証済み	○	vmware_dvswitch
vSANクラスタの構成			ポートグループの構成	ansible2.0	*検証済み	○	vmware_dvs_portgroup
ESXi HostのLUN/バスの取得			分散スイッチへのホスト追加	ansible2.0	*検証中		vmware_dvs_host
VMkernel ネットワークの設定	VMkernel ネットワークの設定	VMkernel ネットワークの設定	VMkernel アダプタの作成	ansible2.0	*検証中	○	
			VmkernelインターフェイスのIP変更	ansible2.0	*検証済み	○	vmware_vmkernel_ip_config
			Vmkernelの移行(標準スイッチ→分散スイッチ)	ansible2.0	*検証中		
GuestVM設定	GuestVM設定	GuestVM設定	GuestVM作成	ansible2.0	*検証済み		
			GuestVMの情報取得	ansible2.0	*検証済み		
			GuestVMのシェル上でコマンド実行	ansible2.0	*検証済み		
			GuestVMの以降(標準スイッチ→分散スイッチ)	ansible2.0	*検証済み		
ScaleIO SDCインストール	ScaleIO SDCインストール	ScaleIO SDCインストール	SDCインストール	ansible2.0	*検証済み	○	
			SDC設定	ansible2.0	*検証済み	○	

インフラ構成決定での振り返り

自動化しやすい構成を考える



やるべきことの明確化(タスクをヒアリング・リスト化)



タスク遂行にあたり考えられる方法を検討



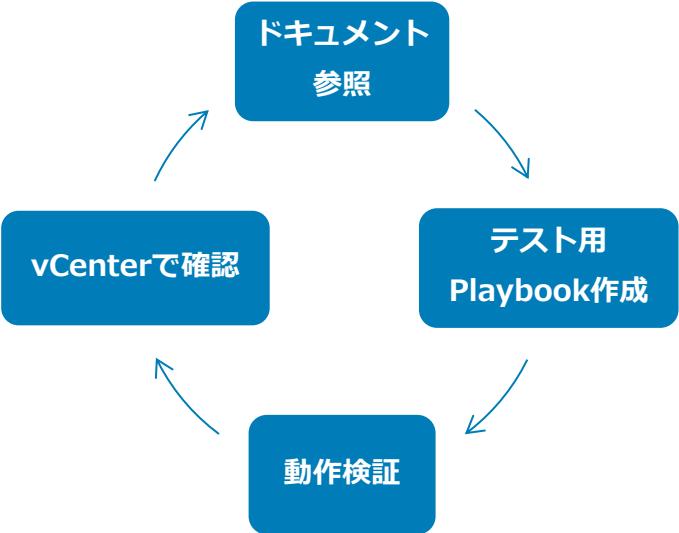
「それAnsibleでできそう？」を検討
(極力Ansibleでやりたい)

Step : 検証/実装

- 検証対象 : Ansible-vmware-module 19個

- 検証結果(当時は。。。) :

- 動作が確認できた利用可能Module : 14個
- エラーにより利用できなかったModule : 5個
 - vmware_datacenter(データセンタ作成)
 - vmware_dvs_host(分散スイッチへのホスト登録)
 - vmware_vmkernel(vmkernelの作成)
 - vmware_migrate_vmk(vmkernelを標準スイッチから分散スイッチに移行)
 - vmware_target_canonical_facts(ESXiホストのLUNパス取得)
- 一部はバッチ的な実装で回避



大きなPlaybookを書かずに、小さいコンポーネントから考えて書く。
小さく、シンプルな繰り返しを実践する。

Step : 検証/実装

Module(vmware_cluster)動作例

Playbook: create-cluster.yml :

```
---
- hosts: localhost
  tasks:
    - name: create cluster in vCenter
      vmware_cluster:
        hostname: 192.168.10.200
        username: administrator@vsphere.local
        password: P@ssw0rd
        datacenter_name: testdc
        cluster_name: cluster-add
        enable_ha: true
        enable_drs: true
        enable_vsan: true
        validate_certs: false
```

実行結果 :

```
[root@jumper wakabh]# ansible-playbook -i localhosts -v -k create-clstr.yml
No config file found; using defaults
SSH password:

PLAY [localhost]
*****
TASK [setup]
*****
ok: [127.0.0.1]

TASK [create cluster in vCenter]
*****
changed: [127.0.0.1] => {"changed": true, "failed": false}

PLAY RECAP
*****
**
127.0.0.1 : ok=2    changed=1    unreachable=0    failed=0
```

プロジェクトを振り返って

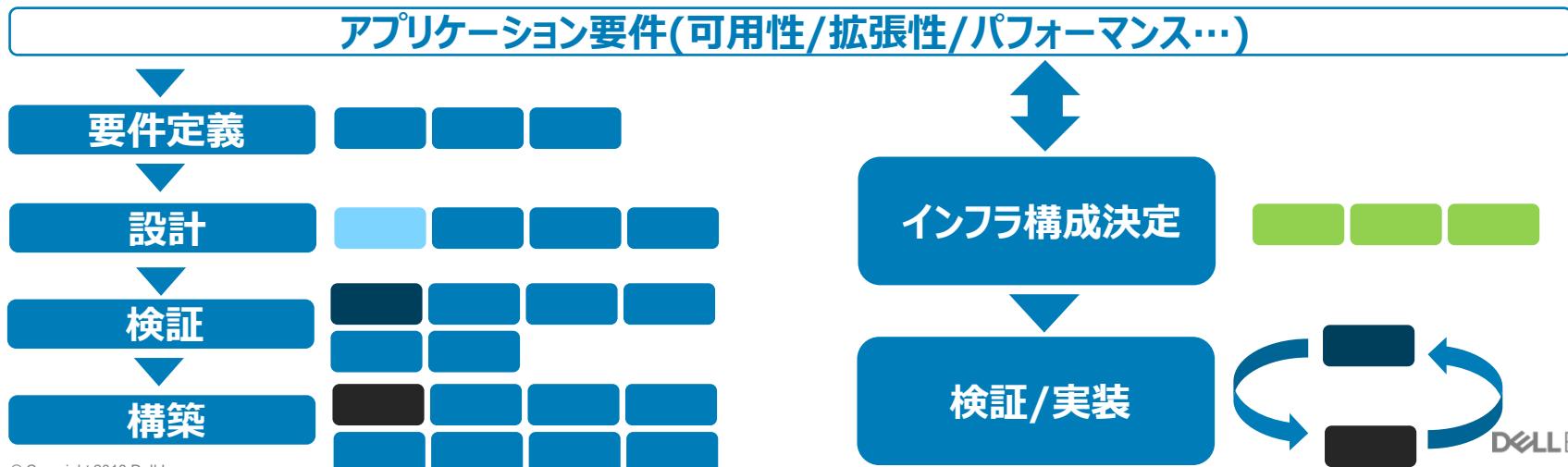
Before

- OS構築 + SDS構築に多くのオーバヘッド
(構築管理がバラバラ)
- 構築時の証跡保存・パラメータ記録
- Excelや膨大な設計書・パラメータシート

After

- Ansibleからの統一された自動構築/管理
(SDS構築をPlaybook実行によりAnsibleから一元的に行う)
- 設定情報のPlaybook化
- ymlファイルが成果物

<サブタスクの完了を待って進める>



まとめ

「自動化しやすいインフラ(ストレージ)をつくること」

- 足りなければ自分たちで作る、作ることができる能力
- 機能拡張を待つのではなく、機能拡張のコントリビュータになる
- とはいえ、コードを書けるだけでは不十分。顧客の運用を知り、詳細にヒアリングする能力も重要

「いかに部品化するか」

- OSSであっても、規模が大きいほど、検証やトラブルシュートは多くなる
- 小さく、シンプルに考え、ループを作る・ライブラリ化する。再利用できるものはする

