



# 概要ご紹介

## 世の中はクラウドネイティブなシステムを目指して切磋琢磨しています



## 課題はクラウドネイティブなシステムを提供できる技術者の不足です



## 現行システムの構築・運用を自動化して技術者を確保する必要があります

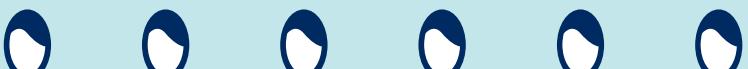


## 現行システムの構築・運用を自動化して技術者を確保する必要があります

新技術を取り込みにくい

密結合で保守しづらいアプリ

従来のスタティックな環境



(2)張り付いているITエンジニアを解放して

自動構築・自律運用

(1)構築・運用を自動化・省力化して

Mobile

Social

IoT

5G

Tech

スケーラブルなアプリ

コンテナ  
サービスメッシュ  
イミュータブルインフラストラクチャ  
宣言型API

近代的でダイナミックな環境

パブリッククラウド  
プライベートクラウド  
ハイブリッドクラウド

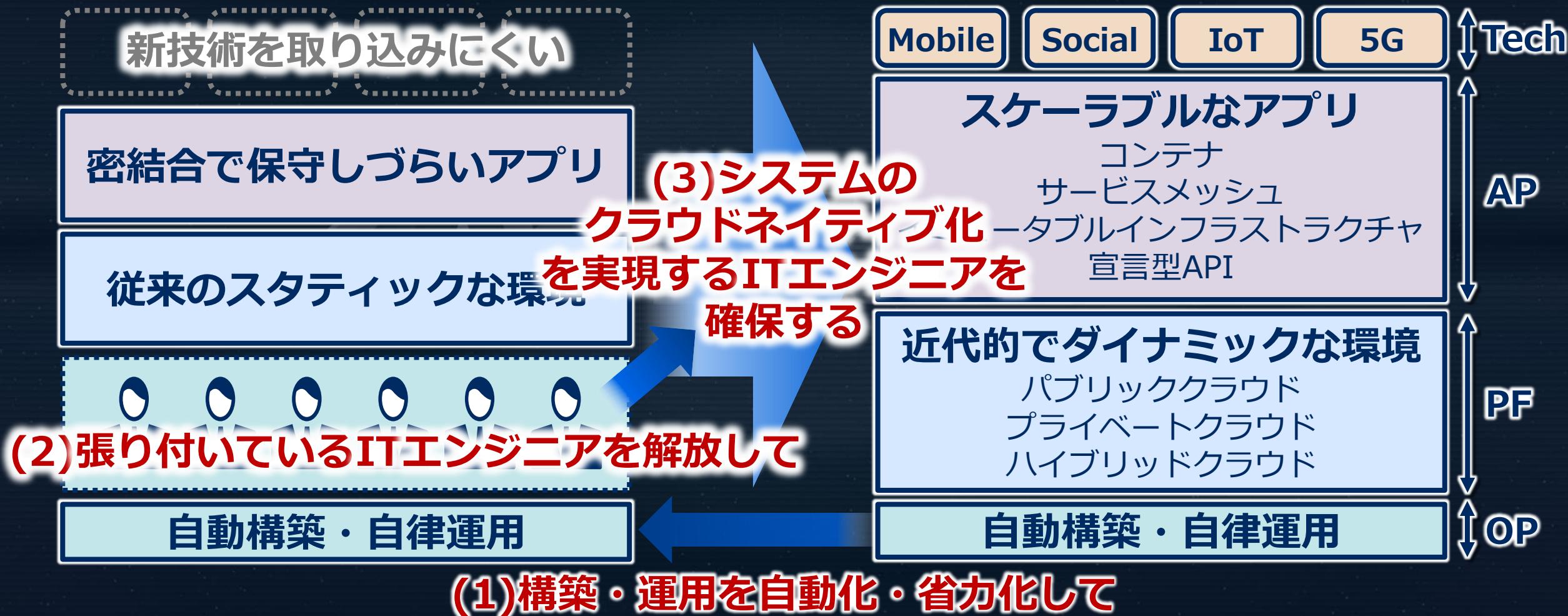
自動構築・自律運用

AP

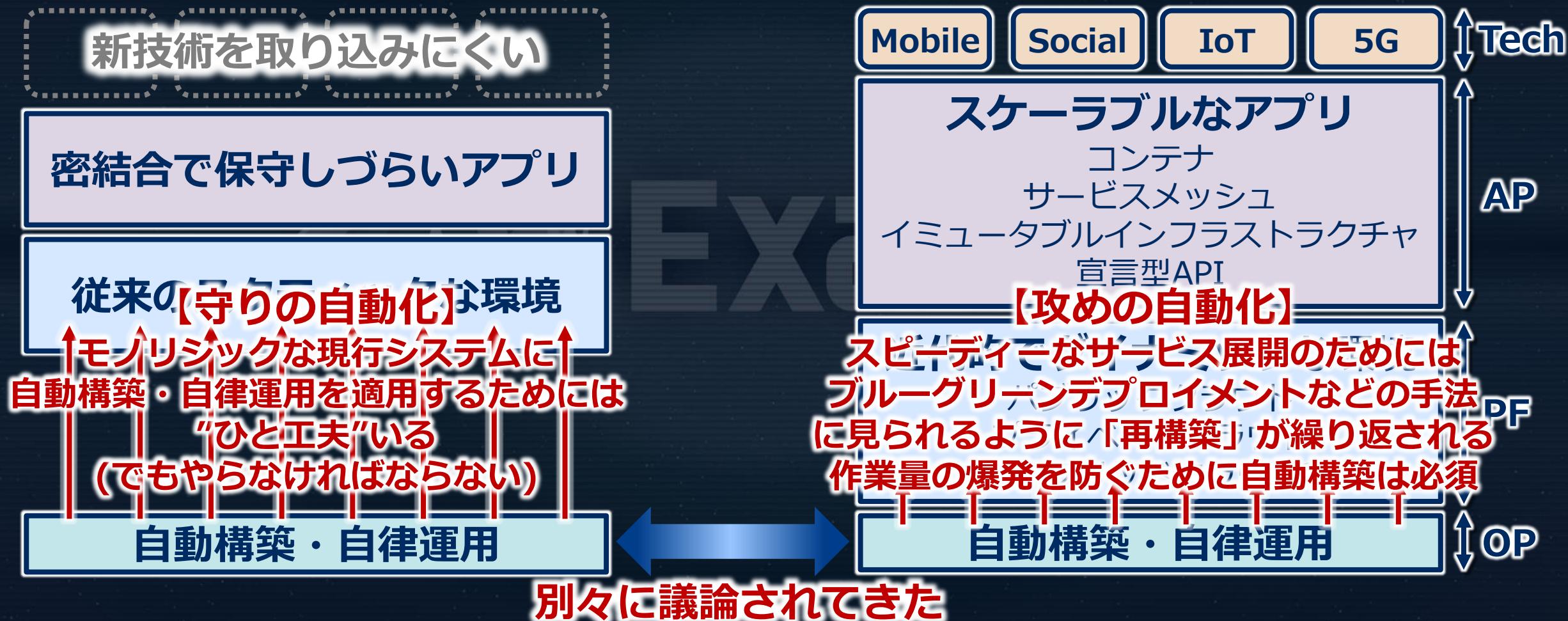
PF

OP

## 現行システムの構築・運用を自動化して技術者を確保する必要があります



## これまで2つの領域のアプローチが別々に議論されてきました



## 2つの領域の自動化は財務的な視点でも目的を異にします



使い切れないシステムリソースを「仮想化」して使い切ることで主にハードウェア保守を効率化できる

結果的にシステムリソースをソフトウェア的に扱えることになり、主にシステム運用(維持)の負荷を「自動化」で軽減することが可能となる

システム運用(維持)の負荷を軽減するために取り組まれてきた「自動化」は、早いサイクルでサービス提供する際の作業量の爆発を防ぐための「自動化」として今まで以上に注目を集めている

## 守りの自動化はOPEX(維持費)の効率化を目的とすることが多いです



## 攻めの自動化はROI(投資対効果)の引き上げを目的とすることが多いです



しかし2つの領域は実際にはアプローチを融合して考える必要があります



## それでは「守りの自動化」と「攻めの自動化」についてご説明します

新技術を取り込みにくい

密結合で保守しづらいアプリ

従来のスタティックな環境

モノリシックな  
システムの自動化  
(守りの自動化)

Mobile

Social

IoT

5G

Tech

スケーラブルなアプリ

コンテナ  
サービスメッシュ  
イミュータブルインフラストラクチャ  
宣言型API

近代的でダイナミックな環境

クラウドネイティブな  
システムの自動化  
(攻めの自動化)

AP

PE

【守りの自動化】  
モノリシックなシステムの構築・運用に携わる  
ITエンジニアの「苦」



## モノリシックシステムの SIに携わるITエンジニアの現場の声をまとめてみました



設計



作業準備



作業実施

- チーム間の情報伝達に遅延やミスが発生する
  - データの二重管理や独自文言が設計ミスにつながる
  - 多重開発により設計書(帳票)の管理が煩雑化する
  - 結果として設定の前後性を確認できない
- 
- チーム間の作業順序が複雑で毎回タイムチャートを作成しては使い捨てる
  - 作業ごとに手順書を作成/レビューしては使い捨てる
  - 手順ごとにコンフィグを埋め込んでいて、新機種／新OSを追加するごとに手順書のパターンが増える(マルチベンダー対応の障壁)
- 
- 人手作業なので作業時間が一定でない  
⇒チーム間で作業待ちが発生
  - 人手作業なので人為ミスの懸念から逃れられない

## モノリシックシステムの 運用に携わるITエンジニアの現場の声をまとめてみました



管理不足

- 運用上変更してよいパラメータと変更してはいけないパラメータが把握できていない
- システムのパラメータの現在値や過去の変更履歴を管理できていない
- 結果としてせっかくパラメータ化されているのに運用で利用できていない



高負荷

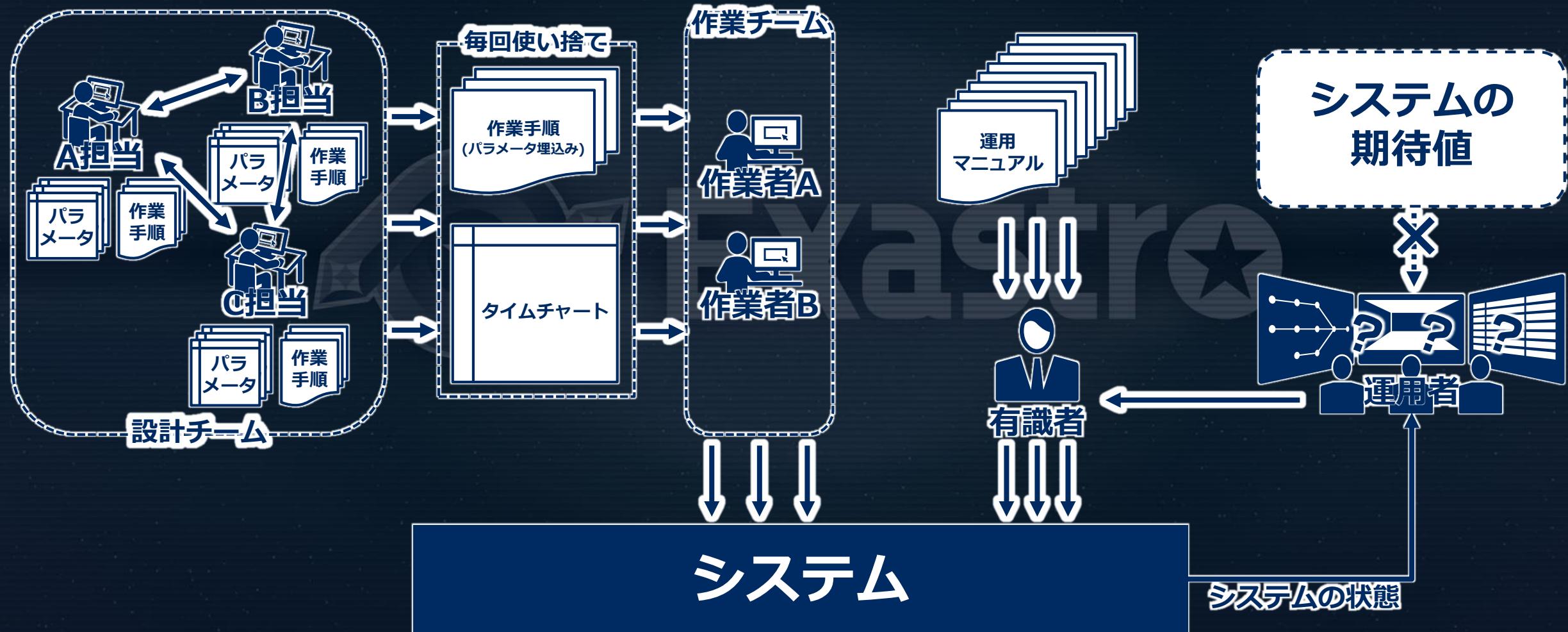
- システムは複雑化の一途を辿っており作業量は増大するばかり
- 何か起こるとExcelで書かれた大量のマニュアルを読み替えながら複数体制で1作業ずつ慎重に実行するしかない
- 結果としてシステムの故障時間が長くなりサービスにも影響が出る



属人化

- 有識者不在により作業が進まない
- 有識者がいなくなるとノウハウは消失する
- 既知事象/未知事象の切り分けが難しく有識者の経験に頼らざるを得ない
- 結果として有識者を異動させられない

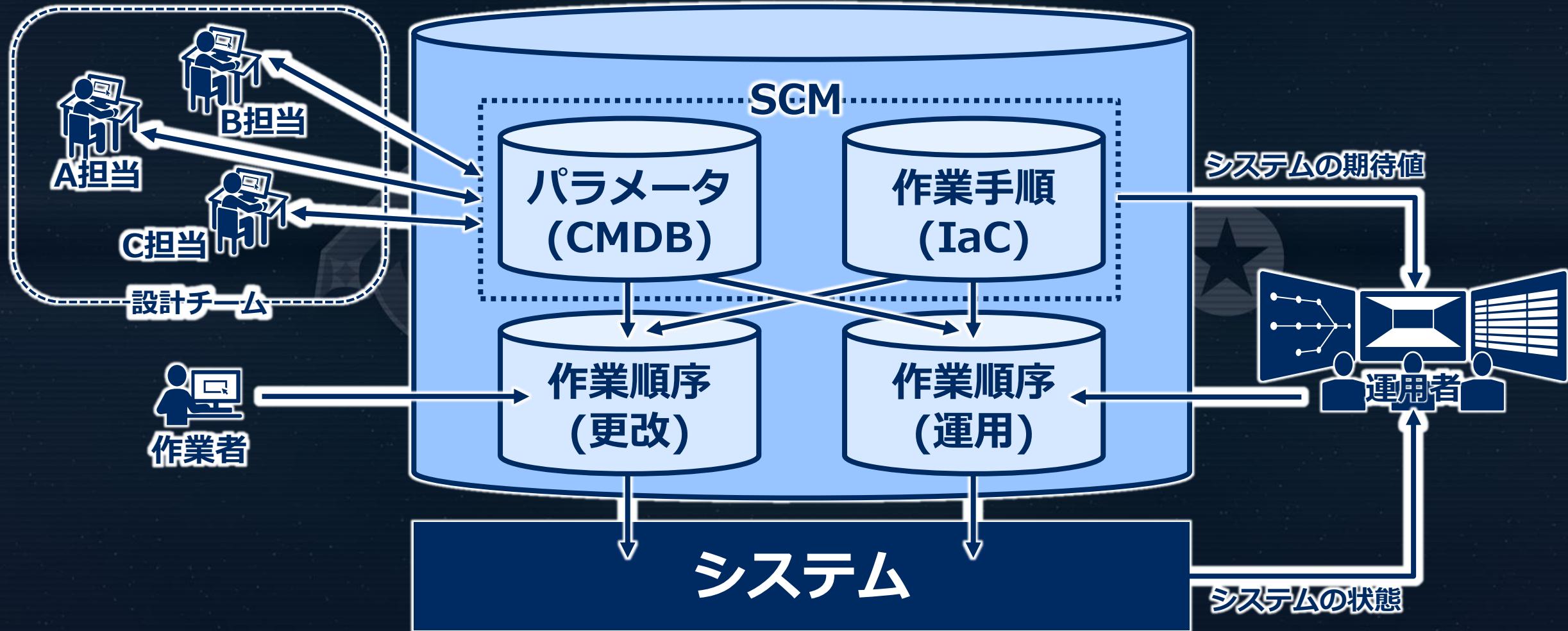
関係者は散乱した情報を正確に伝えることに多くの時間を費やしています



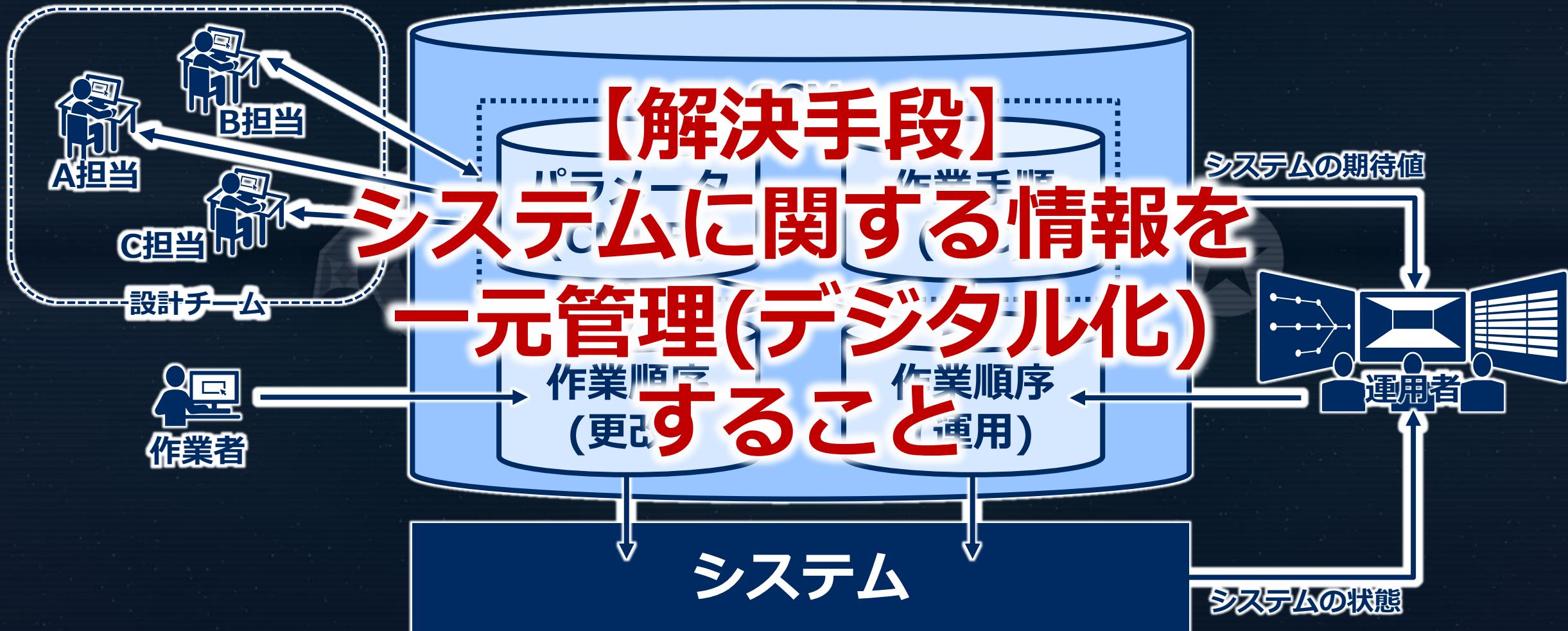
関係者は散乱した情報を正確に伝えることに多くの時間を費やしています



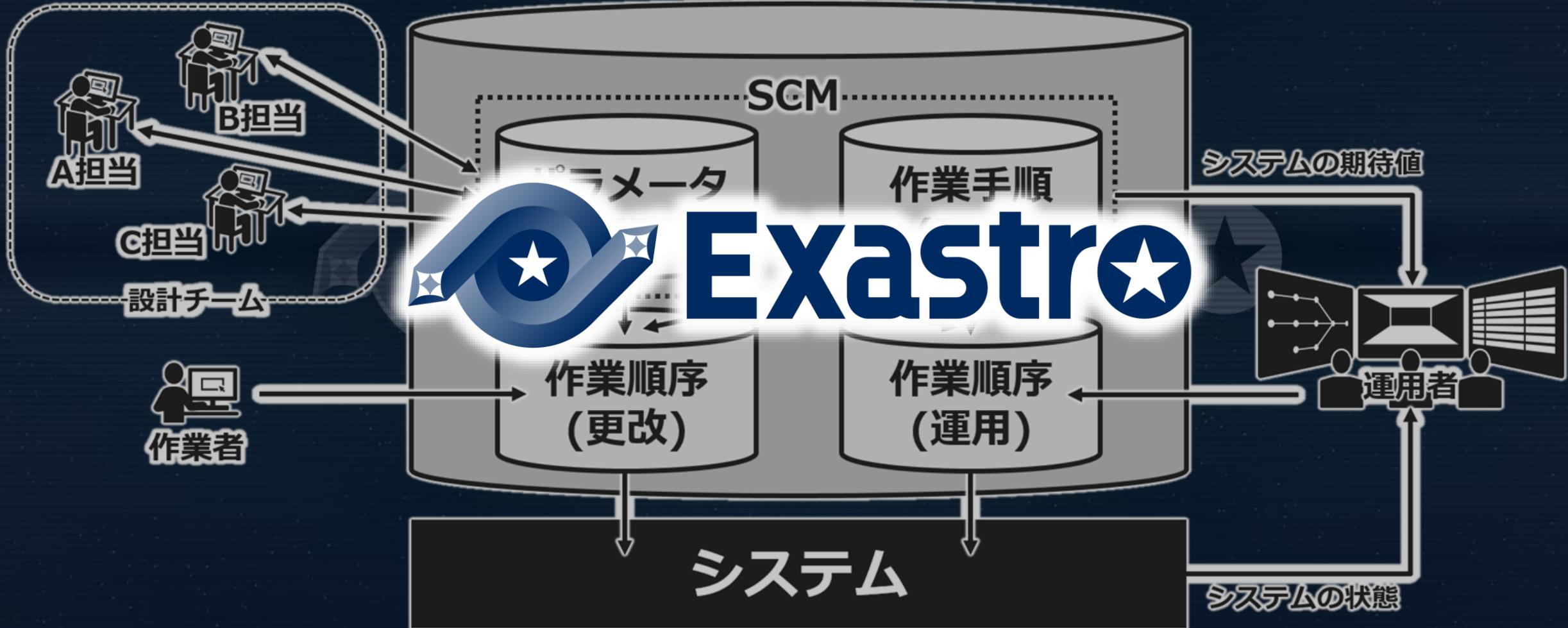
システムに関する情報をデジタル化して一元管理すればよいのですが…



システムに関する情報をデジタル化して一元管理すればよいのですが…



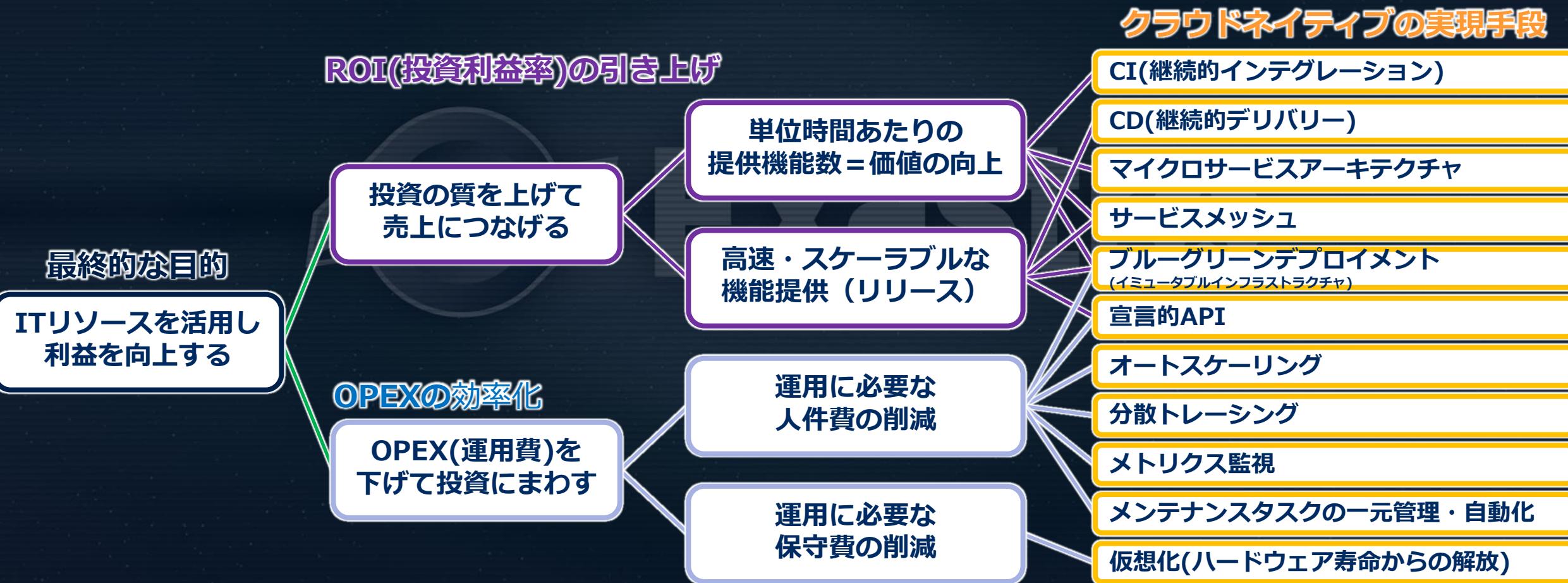
Exastroはシステム情報をデジタル化して一元管理するのに役立ちます



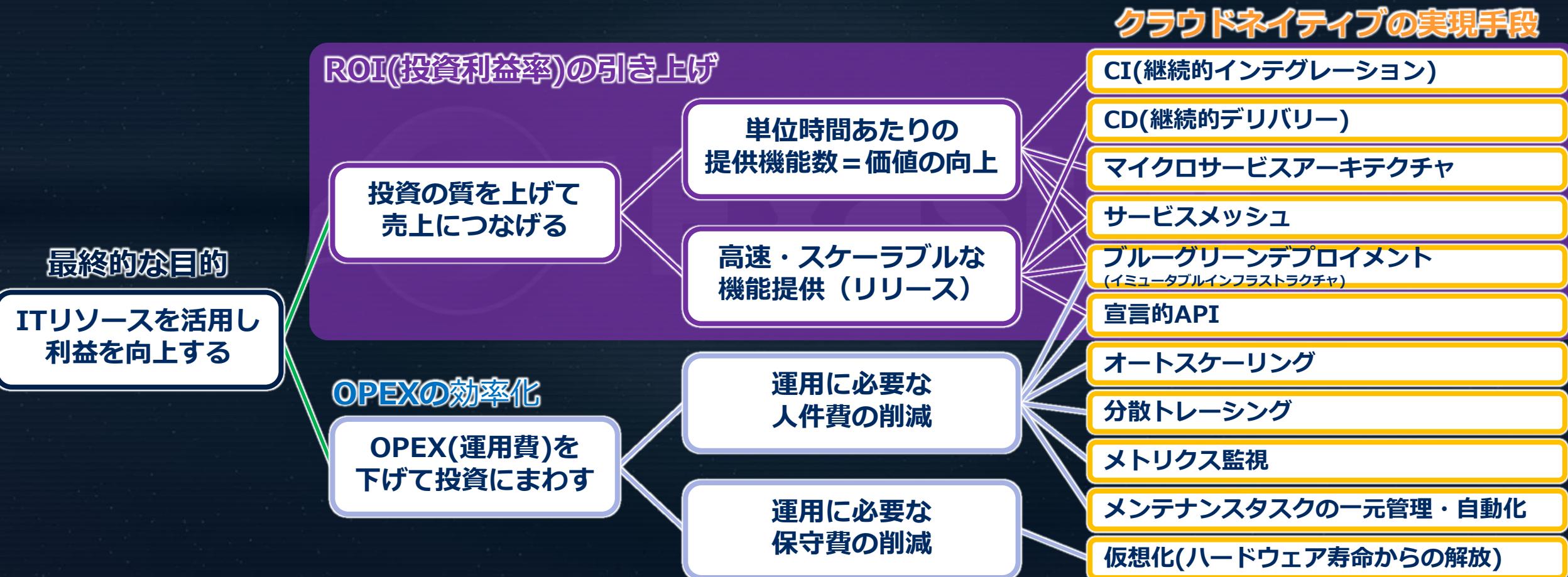
**【攻めの自動化】**  
**ROI(投資対効果)を引き上げるために必要な  
クラウドネイティブのテクニック**



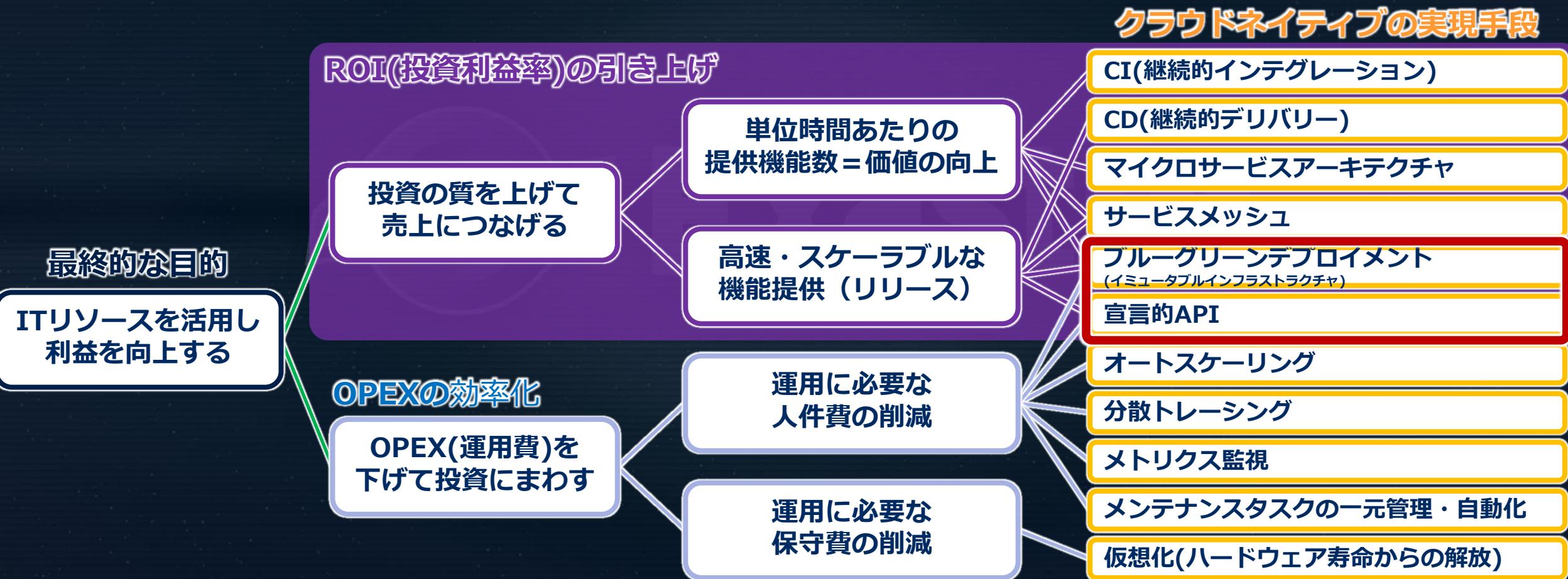
## ITリソースを活用し利益を向上したいという最終的な目標のために 様々な「クラウドネイティブな手法」が考えられています



## そのうち下表の「CI/CD」～「宣言的API」といった手法では ROI(投資対効果)の引き上げを見込むことができます



さらに「ブルーグリーンデプロイメント」と「宣言的API」は自動化と密接な関係があり、「攻めの自動化」に不可欠な要素です



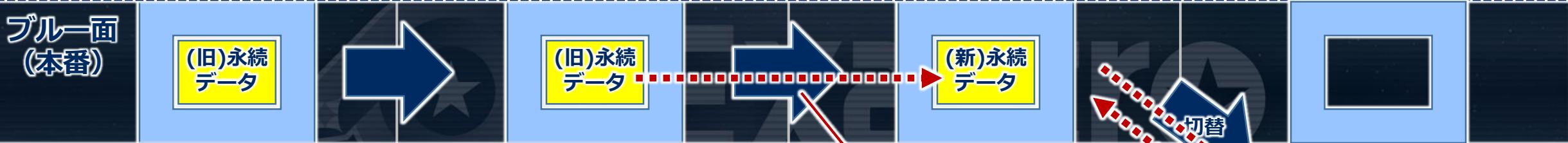
## ブルーグリーンデプロイメントでは部分的に「新規構築」を繰り返します 手作業でやっていたのでは作業量が爆発しますので自動構築は必須です

Step1  
本番環境のみの状態

Step2  
新たな環境を構築(試験実施)

Step3  
永続データを真データへ返還

Step4  
系切替+データの付替えで、  
新環境をリリース



1回のリリースのために  
2環境分の「新規構築」  
を実施する必要がある  
(再現性も重要)

「新規構築」以外の移行作業もある  
(自動化しなければならない)

※切り戻し可能な状態を保つために互換性を保つ。検証可能とする必要がある

「新規構築」可能な部分は宣言的API(るべき姿を定義)が効果的です  
一方で、幾許か残る移行作業には命令型IaCでの自動化が必要です

### 宣言型のIaC (宣言的API)

最終的に以下の通り。  
○○が1個  
□□が2個  
△△が3個

クラウドリソースの  
新規構築に向いている



### 命令型のIaC (手続き的)

1. まず〇〇を1個用意する
2. 次にAとBを混ぜて△△を3個作る
3. 最後にCとDを混ぜて□□を2個作る

クラウドリソース以外は  
命令型IaCで作業を自動化  
する必要がある



**Exastro® は宣言型と命令型の両IaCを駆使した自動化に役立ちます**

**まとめ：「攻めの自動化」を実現するうえで重要なポイント**

**I. 作業量に糸目をかけない。**

**そのためにあらゆる作業は例外なく自動化する。**

**(短いインターバルで何度も「新規構築」を可能とする)**

**II. 宣言的APIはとっても重要。**

**ただし、それだけではあらゆる作業を自動化できない。**

**クラウドネイティブでも命令型IaCの必要性を忘れない。**

【攻めと守りの自動化】

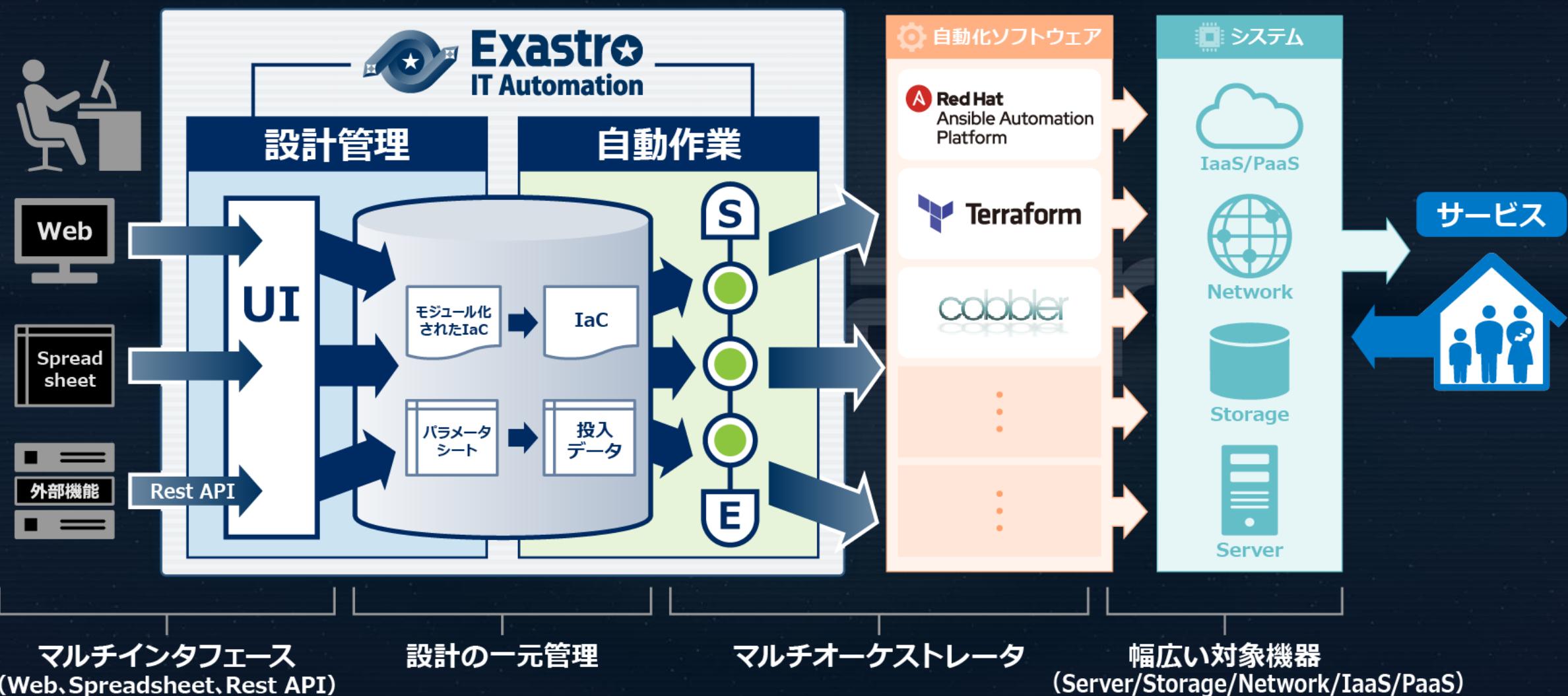
# Exastro IT Automation



# Exastro IT Automation : システム情報をデジタル管理するためのフレームワーク

設計フェーズ

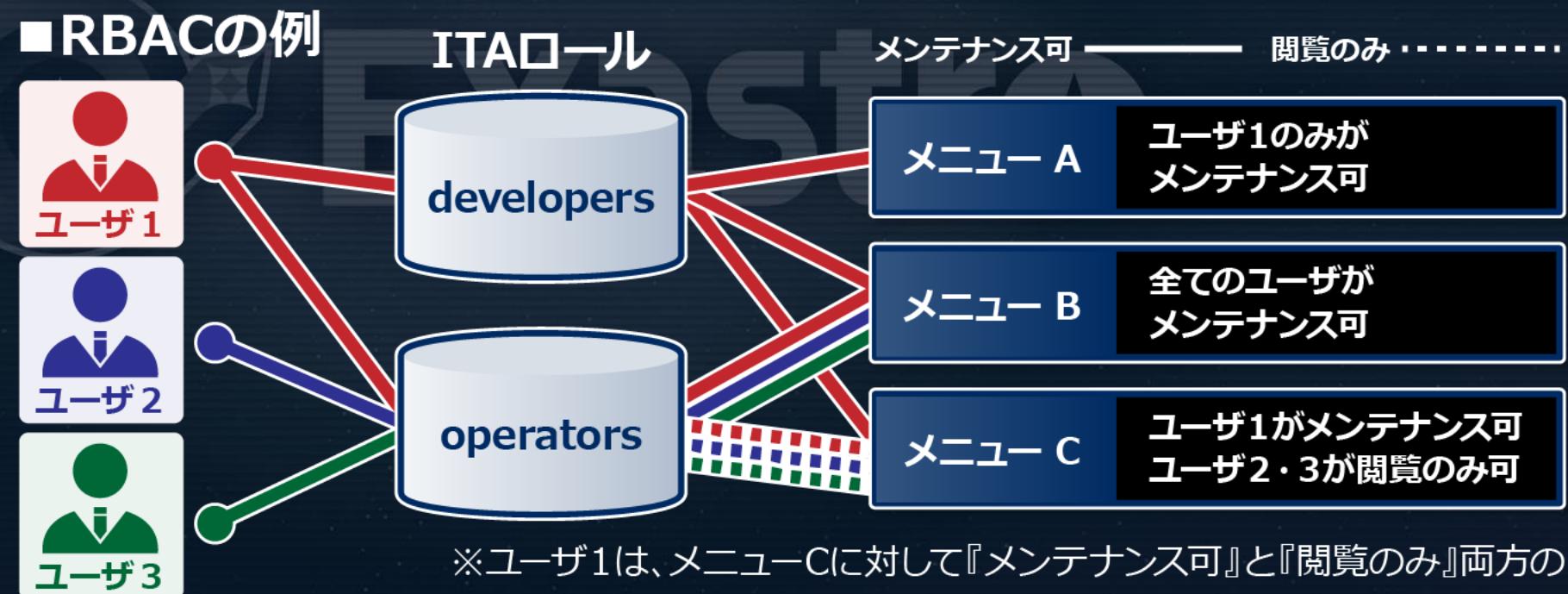
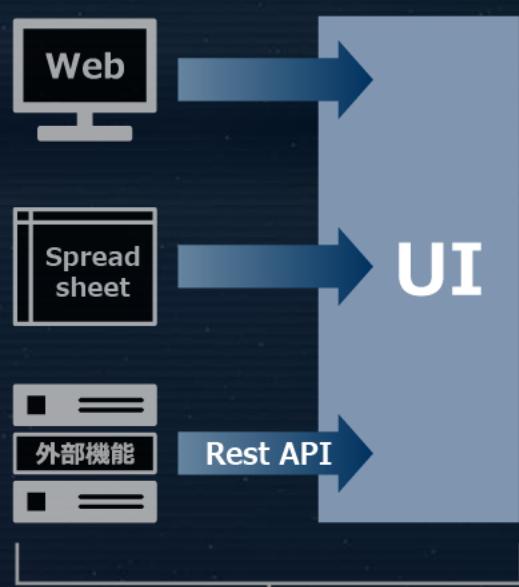
作業フェーズ



- 
- 1 マルチインターフェースとRBAC
  - 2 パラメータをグルーピング／履歴管理する
  - 3 IaCを解析して変数を切り取る
  - 4 IaCをモジュール管理して再利用性を高める
  - 5 複数の自動化ソフトウェアを繋げて実行する
  - 6 自動化を止めない最後の切り札Pioneerモード
  - 7 実行状況をリアルタイムで監視する

# 1つめの特徴 - マルチインターフェースとRBAC

ユーザ操作を3種類のI/F(Web, Excel, RestAPI)から実行可能  
どのI/Fからの操作でも「誰が・いつ・何をしたか?」を記録する  
RBACを備えており、開発者、作業者、運用者といった役割りを定義でき  
その役割りごとに出ること(参照のみ、更新、実行)を制御できる



## 2つめの特徴 - パラメータをグルーピング／履歴管理する (1/2)

### システムのパラメータ情報をグルーピング／履歴管理する



## 2つめの特徴 - パラメータをグルーピング／履歴管理する (2/2)

パラメータシートは履歴管理機能を標準装備  
設計履歴から抽出した情報を使ってシステム更改する仕組み

ITA の履歴管理機能つきパラメータシート

ホスト	オペレーション		パラメータ				設計日
	日時	作業名	P1	P2	P3	…	
hostA	12/20	クリスマス対応	1024	512	2048	…	10/1
hostA	10/9	hostB 増設	512	256	1024	…	8/3
hostA	9/3	システムリリース	256	128	512	…	7/7
hostB	12/20	クリスマス対応	16	32	64	…	10/1
hostB	10/9	hostB 増設	32	64	128	…	8/3

設計者は設計に集中できる

例えば  
“10/9”で  
パラメータを  
抽出すると

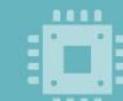
“10/9” のシステムの期待値

ホスト	パラメータ				設計日
	P1	P2	P3	…	
hostA	512	256	1024	…	8/3
hostB	32	64	128	…	8/3

運用者は 運用に  
集中できる

システム更改

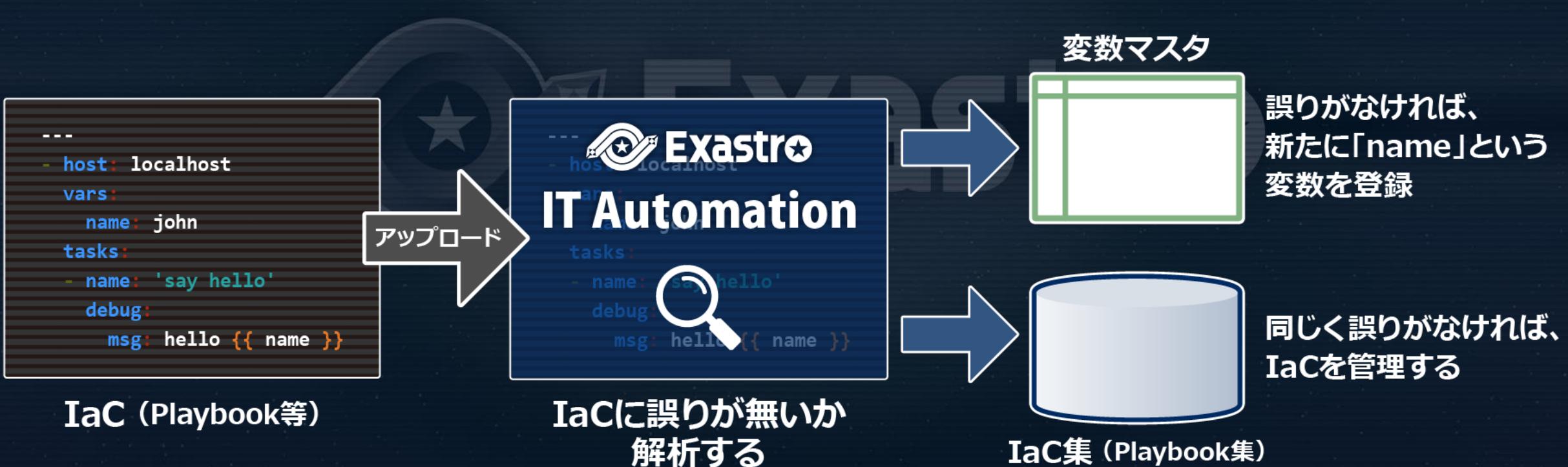
妥当性確認



システム

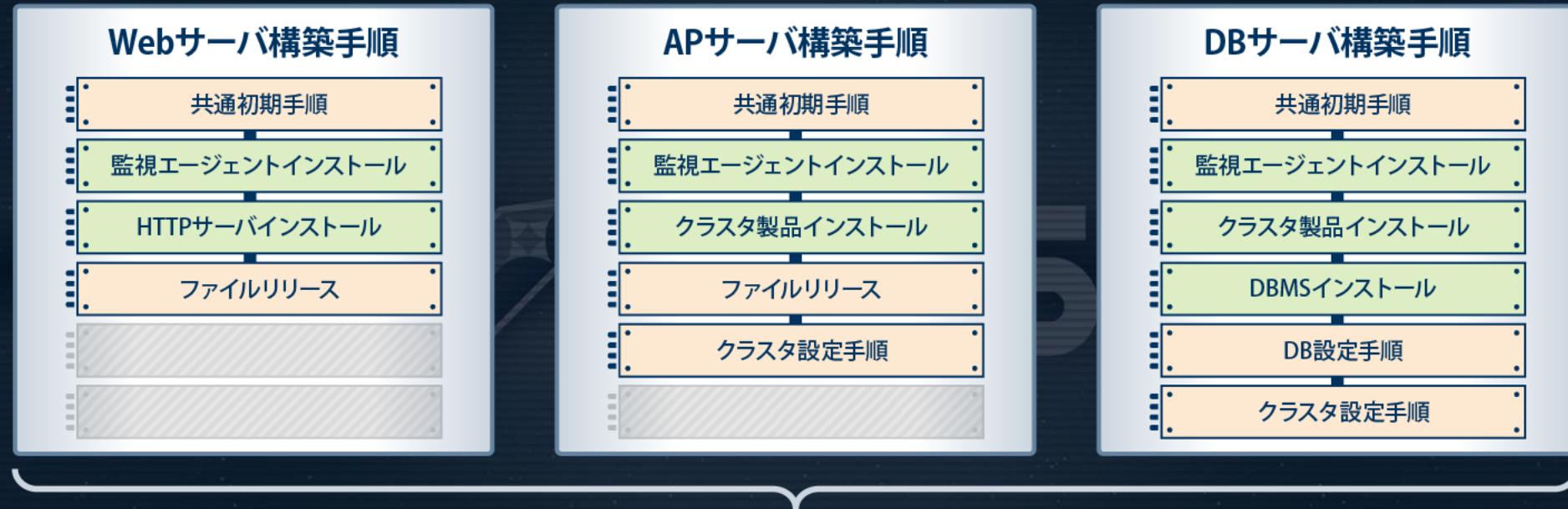
## 3つめの特徴 - IaCを解析して変数を刈り取る

IaCがアップロードされるとまずIaCに誤りが無いか解析する  
誤りがなければ、IaCの記述から変数名を刈り取って管理する  
変数名を選択式で利用し誤植等のヒューマンエラーを防止する

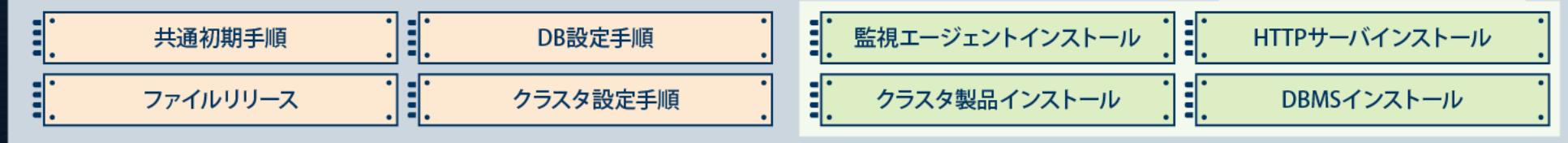


## 4つめの特徴 - IaCをモジュール管理して再利用性を高める

IaC(Playbook等)を一発モノで終わらせず再利用して利用し続けられる  
ように、モジュール化して作業時に組み立てる仕組み



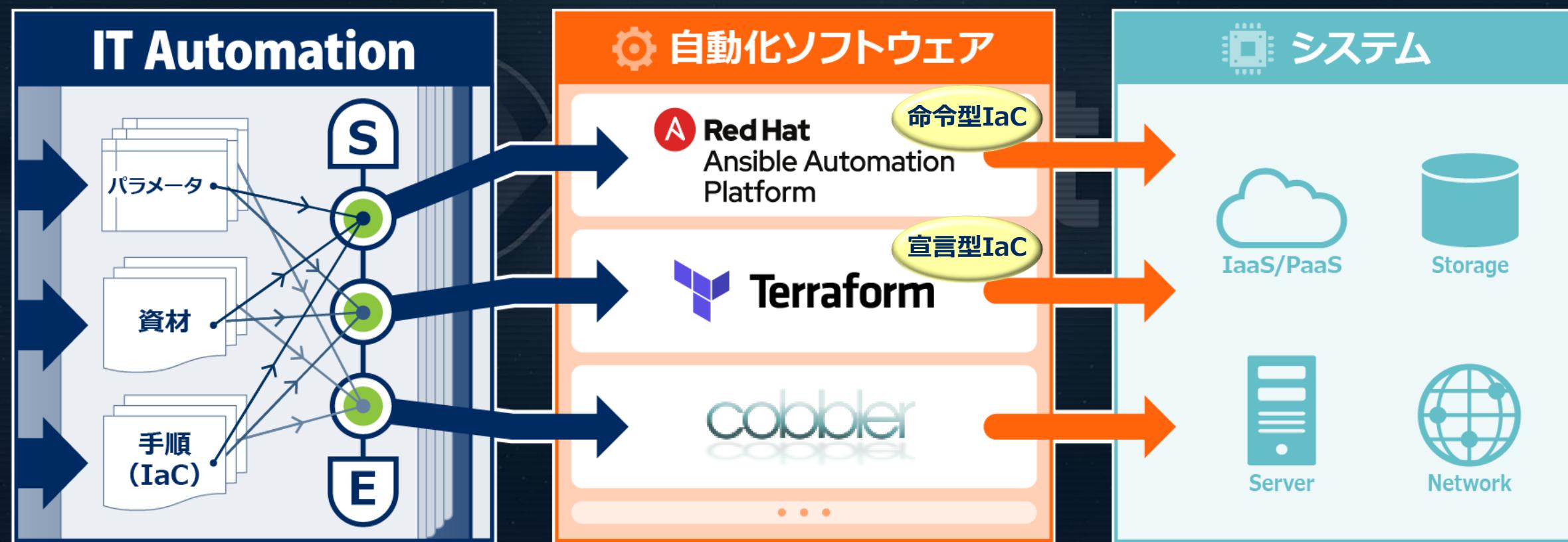
共通の手順はモジュール化し再利用できるように管理する



## 5つめの特徴 - 複数の自動化ソフトウェアを繋げて実行する

複数の自動化ソフトウェアを繋げて一本の作業フローを定義できる  
また自動化ソフトウェアの動作に必要な投入データを自動生成する

例) (Ansibleの場合) 必要なPlaybookを集めて繋げ、ノード毎にパラメータからhost\_varsを作る



## 6つめの特徴 - 自動化を止めない最後の切り札Pioneerモード

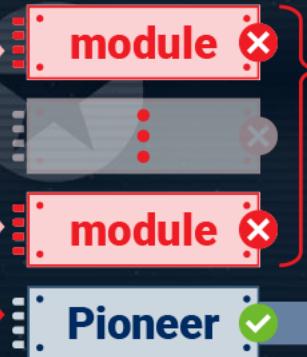
Ansibleのどのモジュールを使っても自動化できない場合に、手動作業を挟んでしまうと自動化のメリットが半減する。そこで、**自動化を止めない最後の切り札として、ITAではPioneerモードを提供。**

### ▼ Pioneer専用「対話ファイル」

expectコマンド  
+  
変数埋め込み  
+  
条件分岐/繰り返し  
を可能とした  
独自のIaC



A Red Hat  
Ansible Automation  
Platform



どれを使っても  
自動化できない場合



ssh / telnet

ITA専用のAnsibleモジュール  
ssh / telnet のいずれかOKであれば対話可能

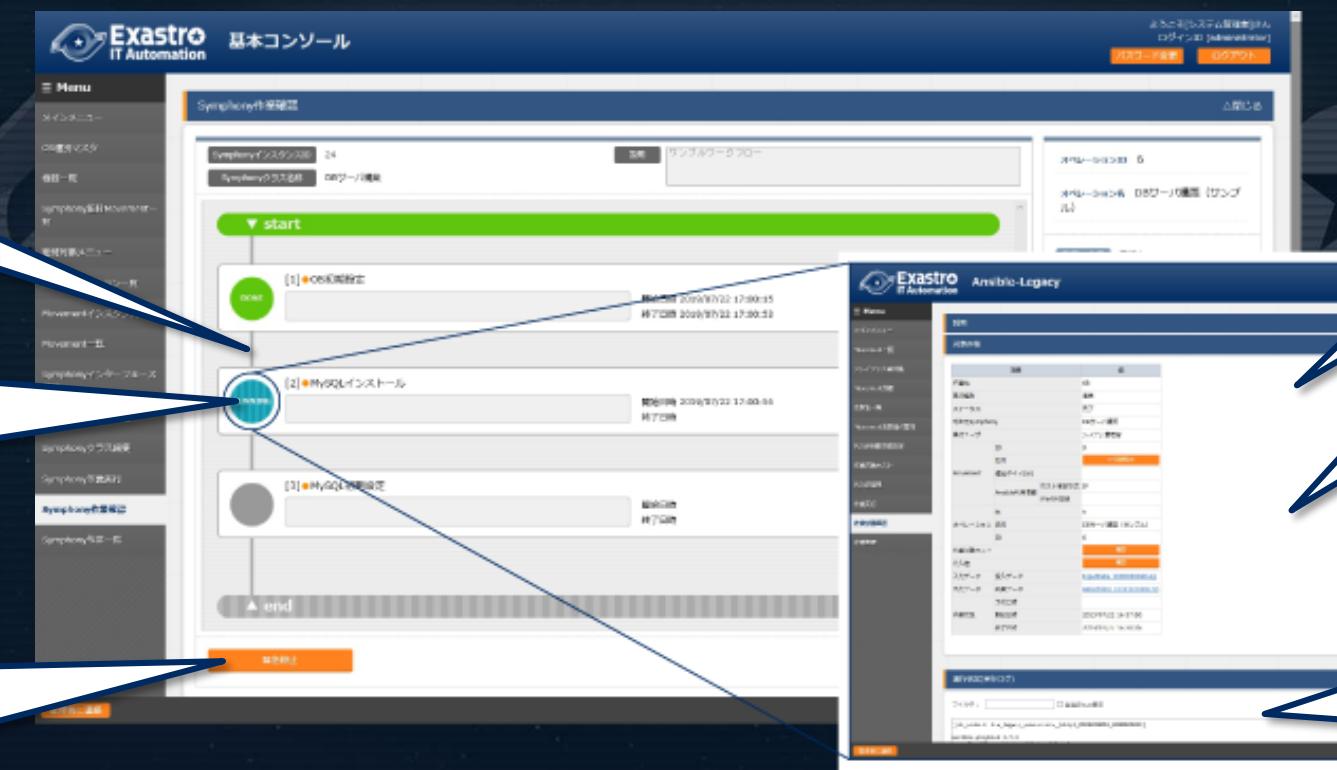
## 7つの特徴：⑦実行状況をリアルタイムで監視する

手動作業と比較して遜色なく実行状況をリアルタイム把握することを重視  
また実行記録(作業エビデンス)を管理し欲しい時にダウンロード可能

作業フローの途中に  
「保留ポイント」  
を設定可能

実行状況をクリック  
すればドリルダウン  
が可能

非常時には「緊急停  
止」で作業をストッ  
プすることが可能

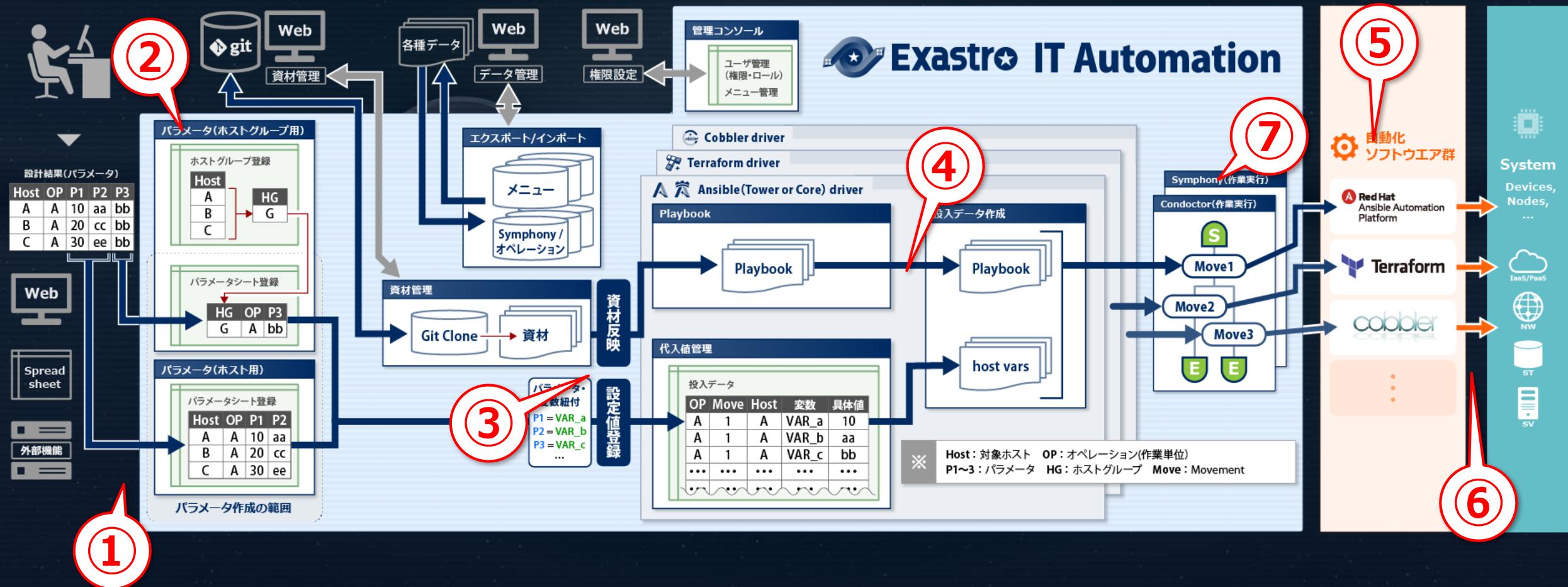


投入データ(自動生成)  
がダウンロード可能  
(zip)

実行結果(作業エビデ  
ンス)がダウンロード  
可能(zip)

自動化ソフトウェア  
の実行状況をリアル  
タイムで表示

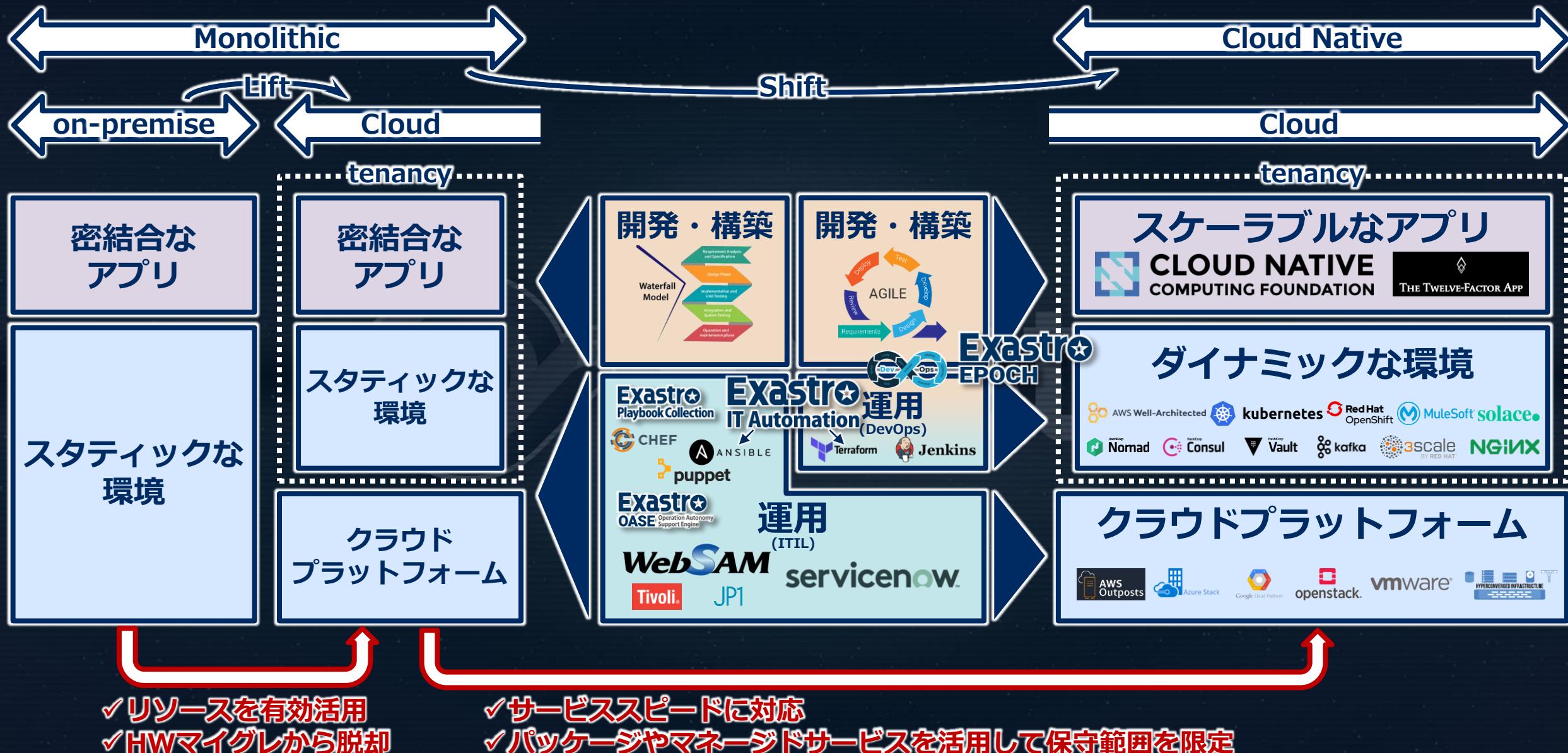
ご紹介した「7つの特徴」の他にも様々な工夫を凝らして、システム情報をデジタル管理できるようにしています。



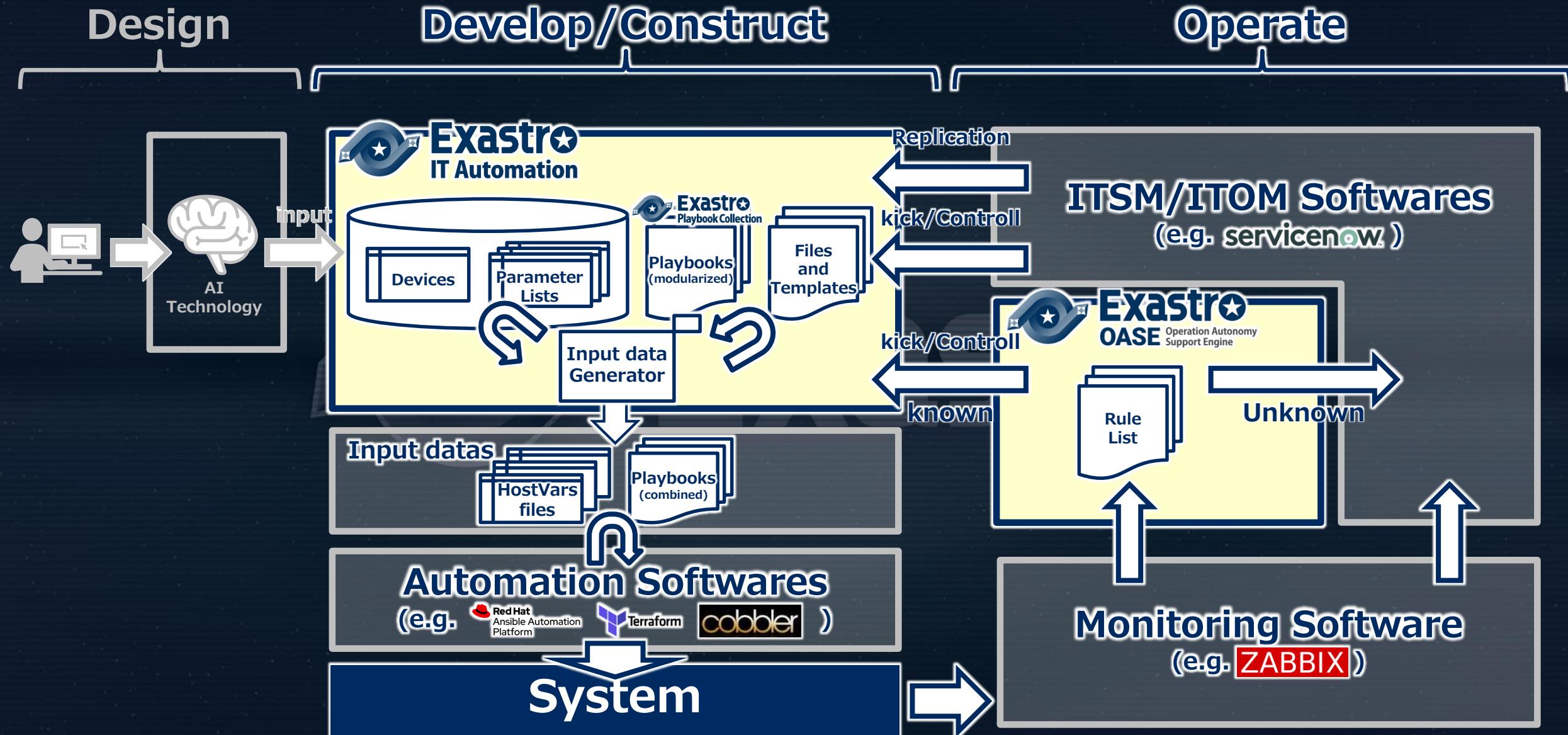
# 参考



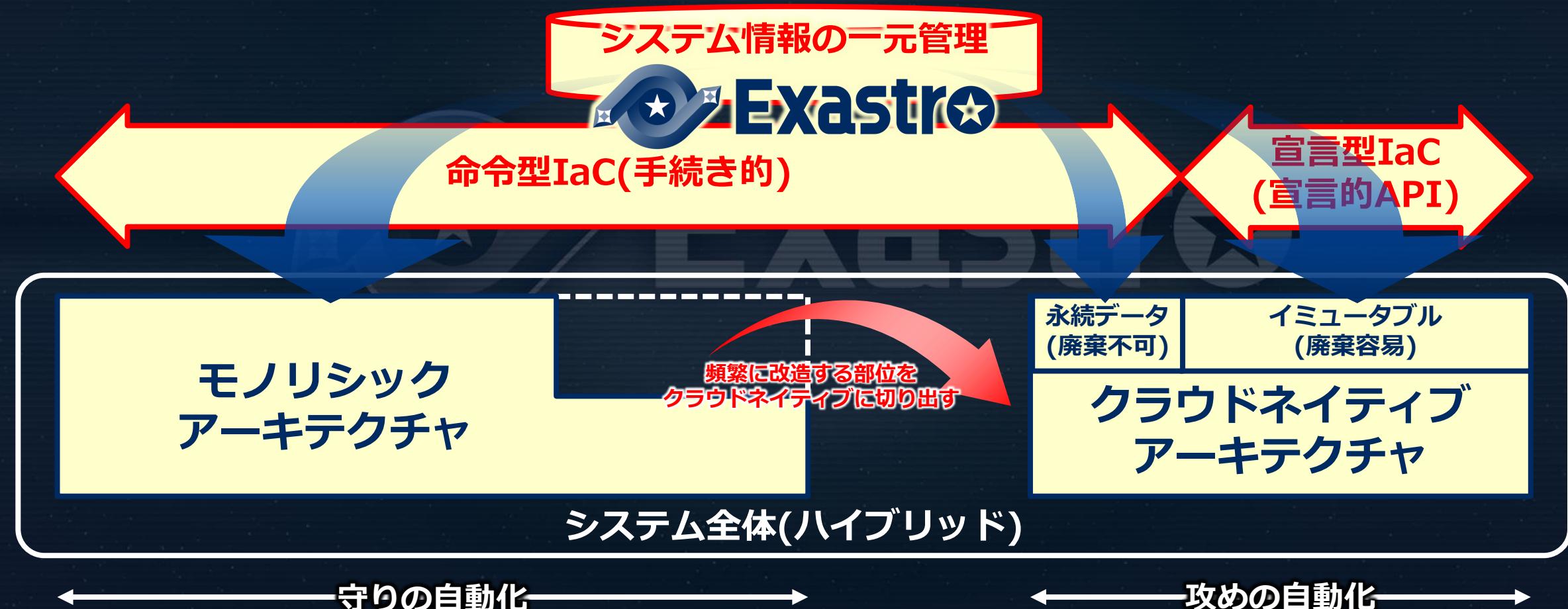
# 【守りの自動化・攻めの自動化】システムを取り巻く状況とExastroの位置づけ



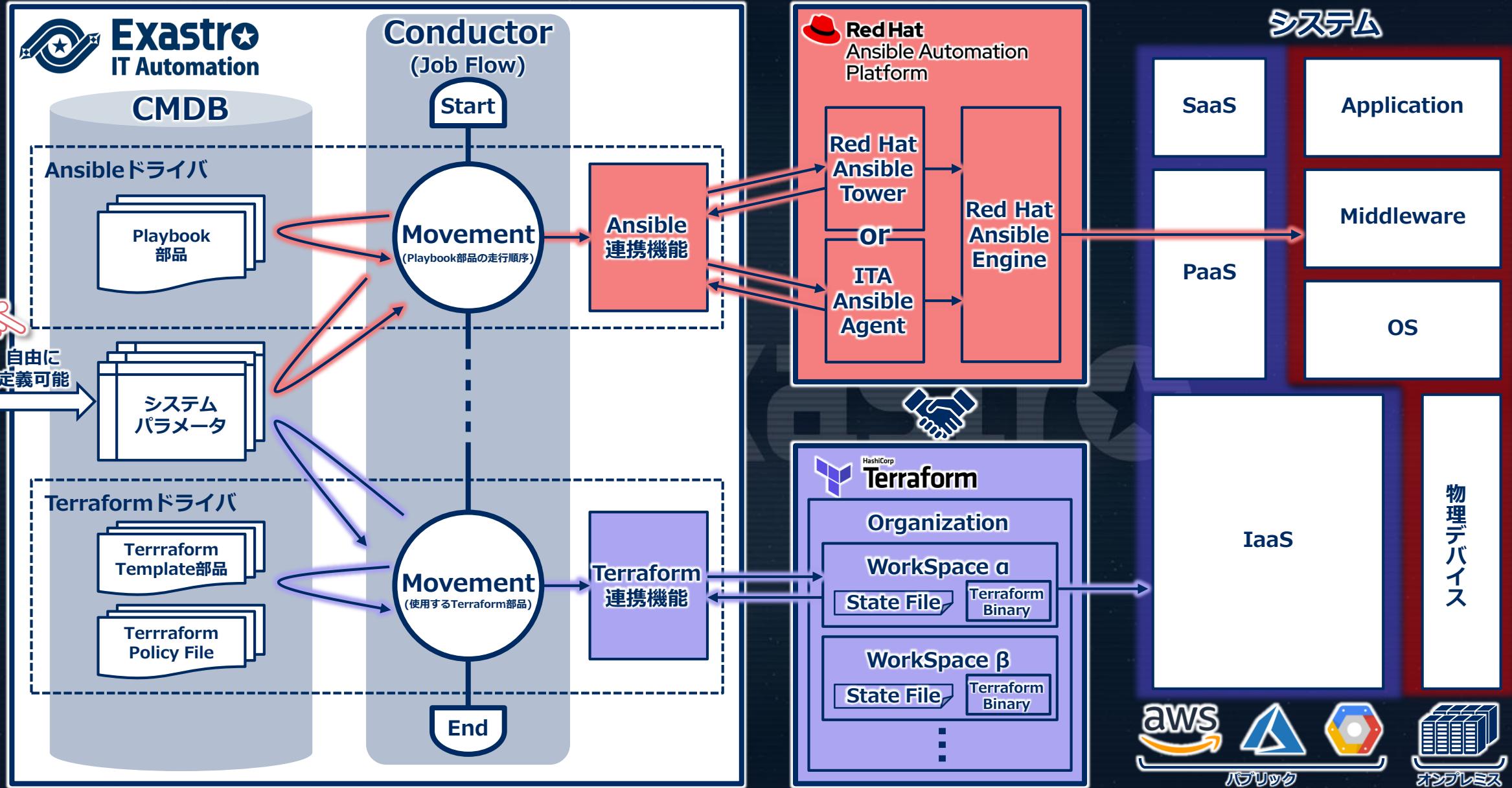
# 【守りの自動化】Exastro Suiteが目指す自動化・自律化の概要イメージ



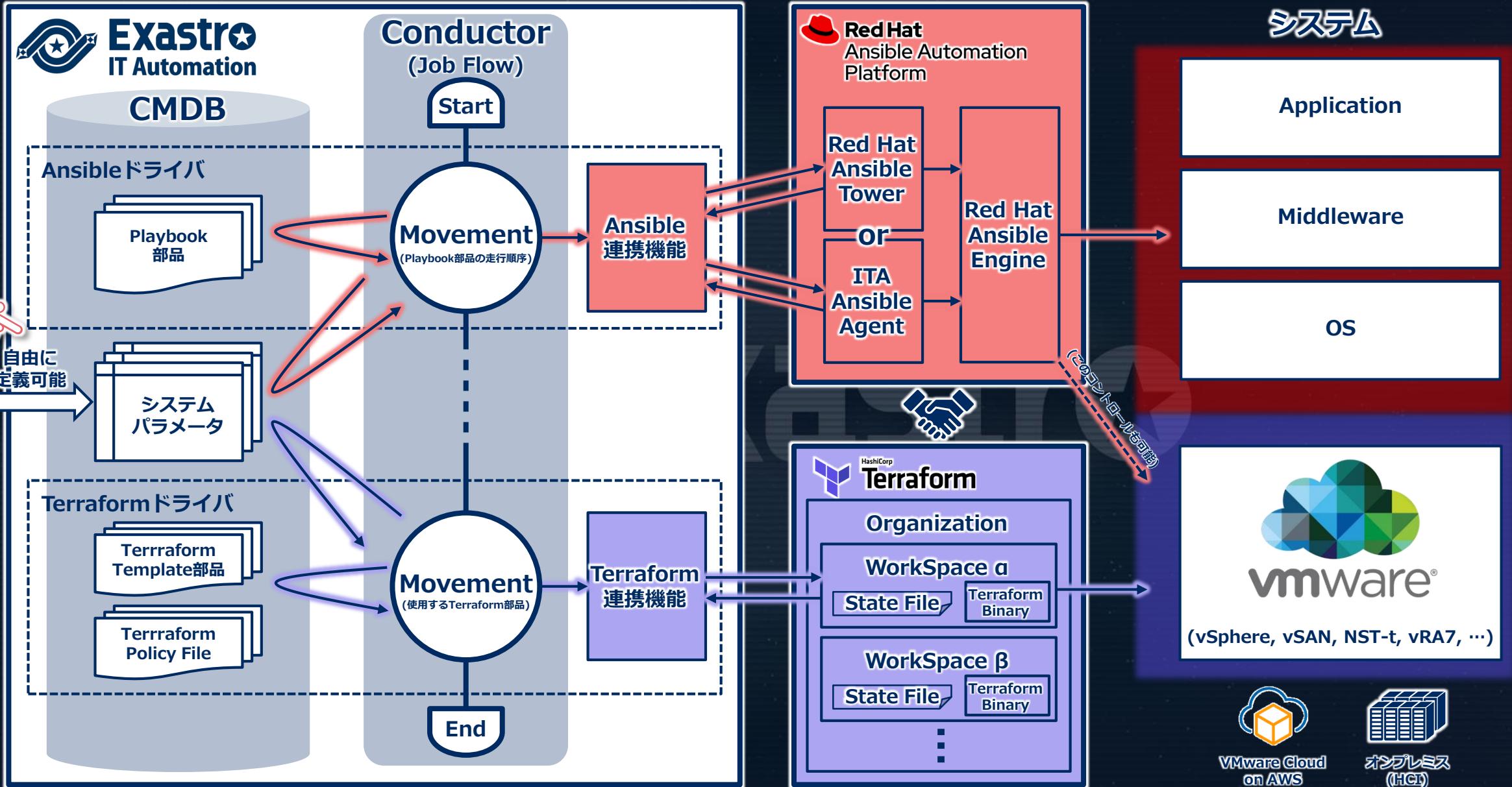
Exastroはモノリシック/クラウドネイティブを別々ではなく  
システム全体の情報を一元管理します



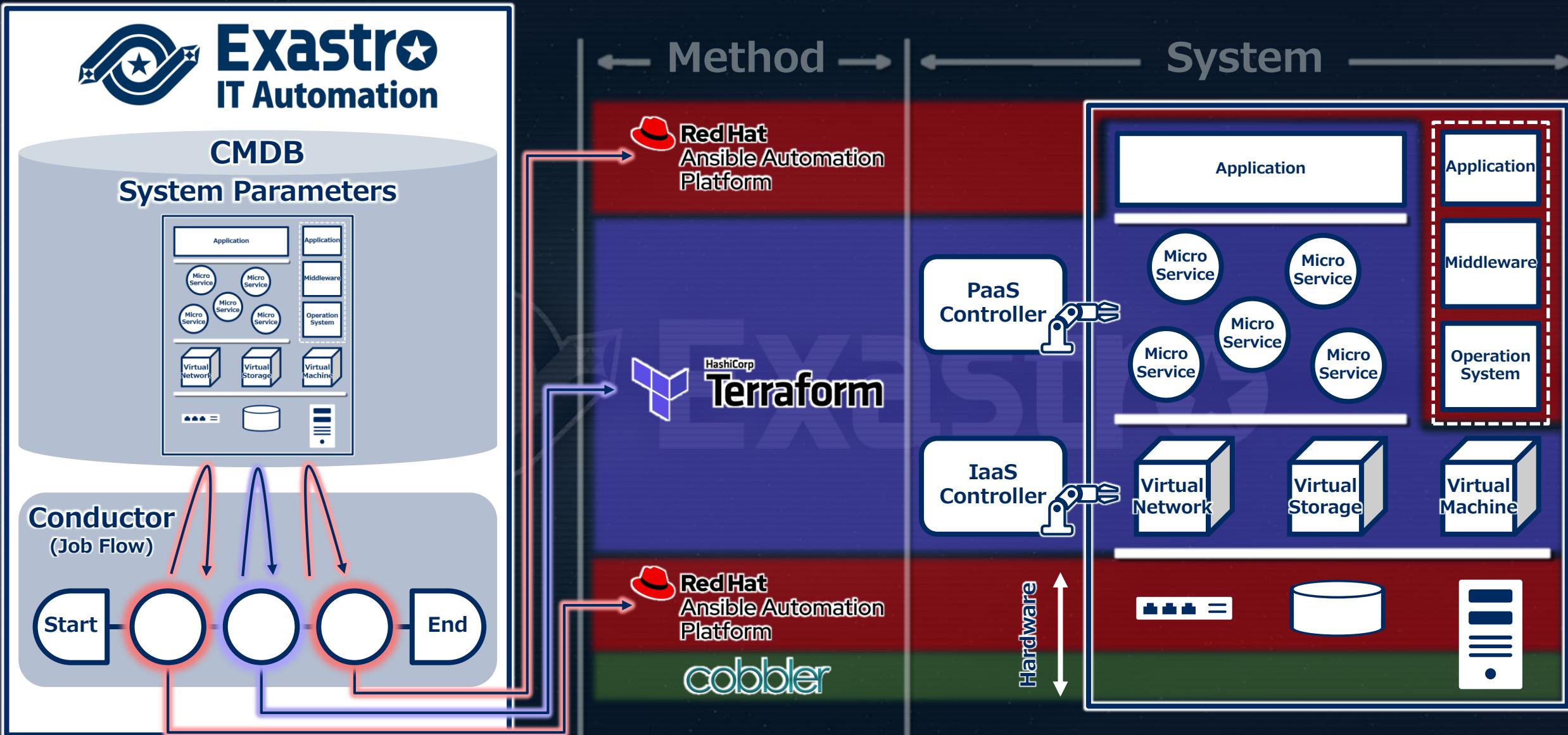
# 【守りの自動化・攻めの自動化】Exastro ITAからAnsibleとTerraformを両方活用する狙い



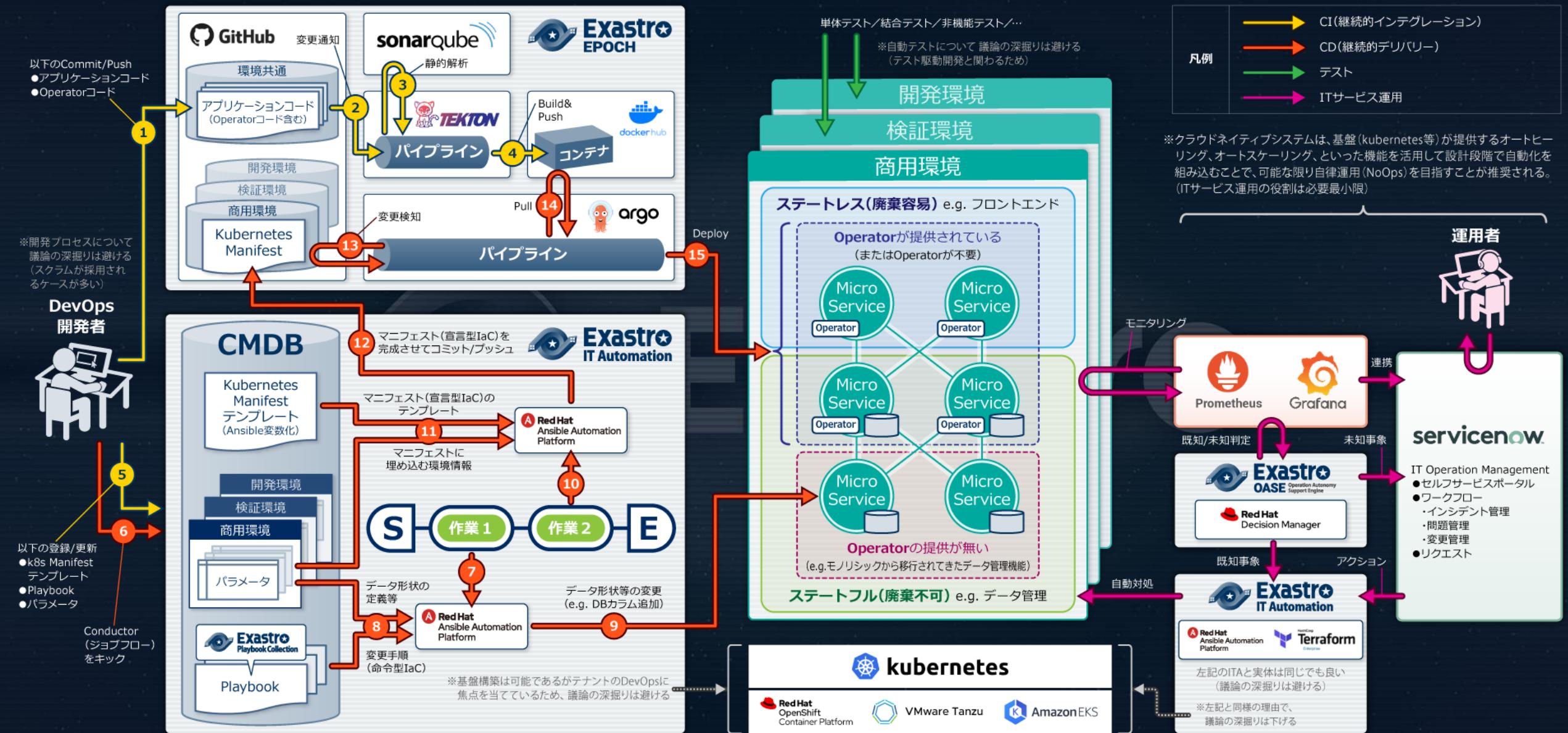
# 【参考】VMware基盤を中心とした仮想上のシステムの自動化(効率化)



# 【守りの自動化・攻めの自動化】システムスタックと使用するメソッド



# 【攻めの自動化】クラウドネイティブでのCI/CD : kubernetesの場合



# 【参考】Exastro の Red Hat Kubernetes Operator Projectへの参加

Red Hat OpenShift Operator認定で認証されたソフトウェアは、OpenShift上で稼働することが保証され、マルチクラウド対応、および運用ノウハウのコード化による自律的運用が実現できます。

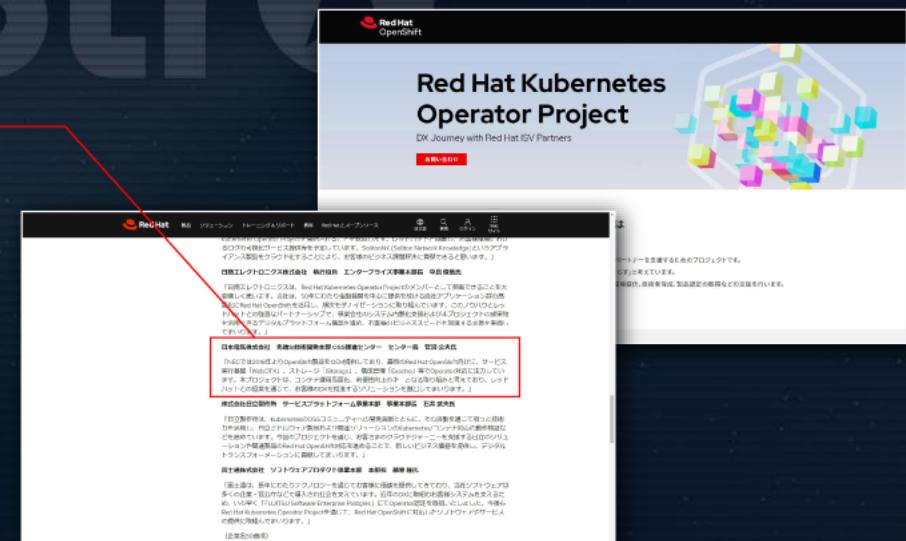
Exastroは、「Red Hat Kubernetes Operator Project」にも参加しており、OpenShift上でExastroをデプロイできるようにOperator対応しています。

日本電気株式会社 先端SI技術開発本部

OSS推進センター センター長 菅沼 公夫氏

「NECでは2016年よりOpenShift製品をOEM提供しており、最新のRed Hat OpenShift向けに、サービス実行基盤「WebOTX」、ストレージ「iStorage」、構成管理「Exastro」等でOperator対応に注力しています。本プロジェクトは、コンテナ運用高度化、利便性向上のキーとなる取り組みと考えており、レッドハットとの協業を通じて、お客様のDXを推進するソリューションを創出してまいります。」

出典：<https://www.redhat.com/ja/about/press-releases/red-hat-starts-kubernetes-operator-project-in-japan>



# 【参考】Exastro ITAとAnsible Towerを連携することによる効果

## IT Automation

設定データを蓄積/管理し、Ansibleが実行するために必要なディレクトリ/コンフィグファイルを生成します。

## AnsibleTower

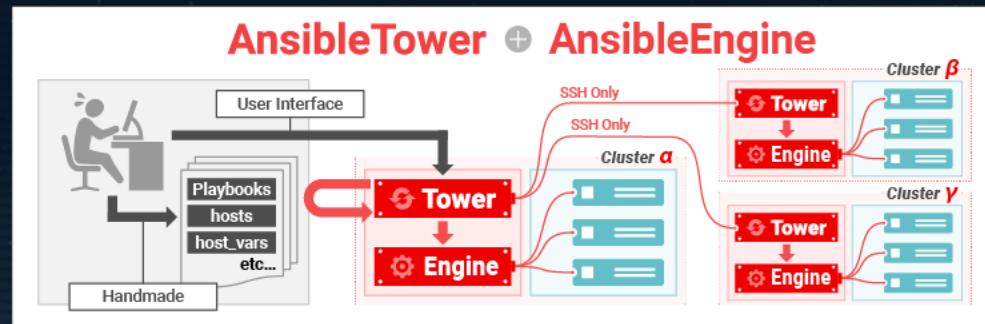
クラスタ間通信をセキュアに、そして異なるバージョンのAnsibleEngineをコントロールします。

## AnsibleEngine

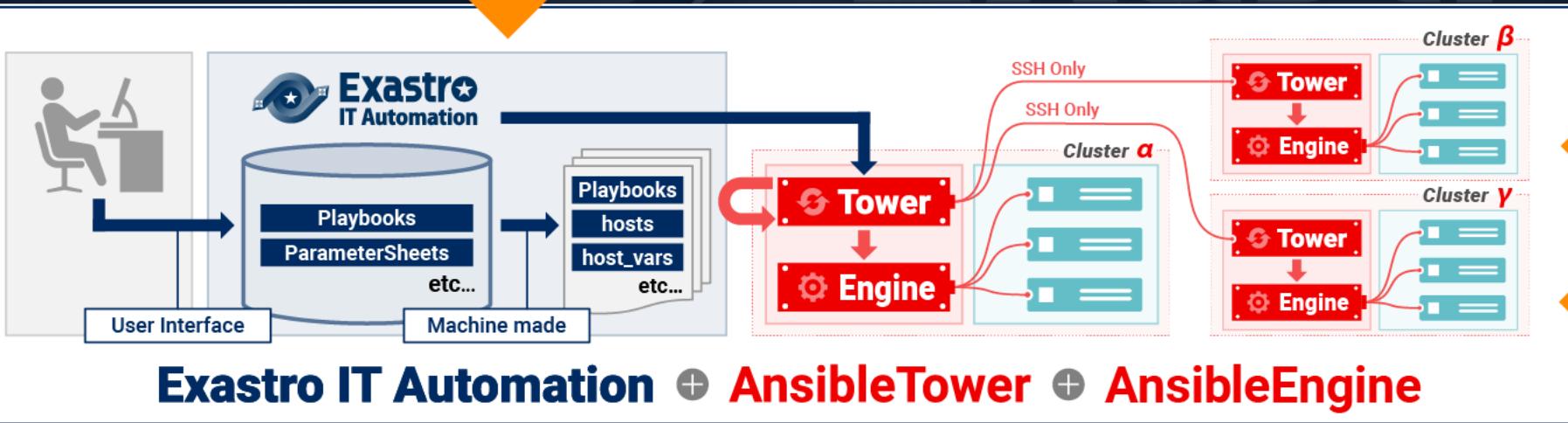
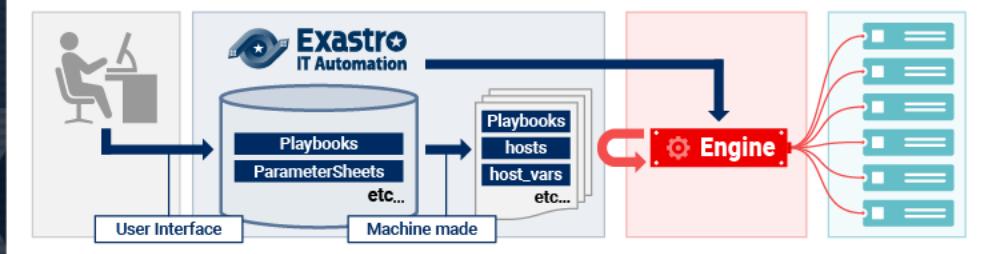
ansible playbookを実行するエンジンです。

それぞれの特徴を組み合わせた、**IT Automation + AnsibleTower + AnsibleEngine** で構成された

自動構築システムで作業の効率化・省力化が実現できます。



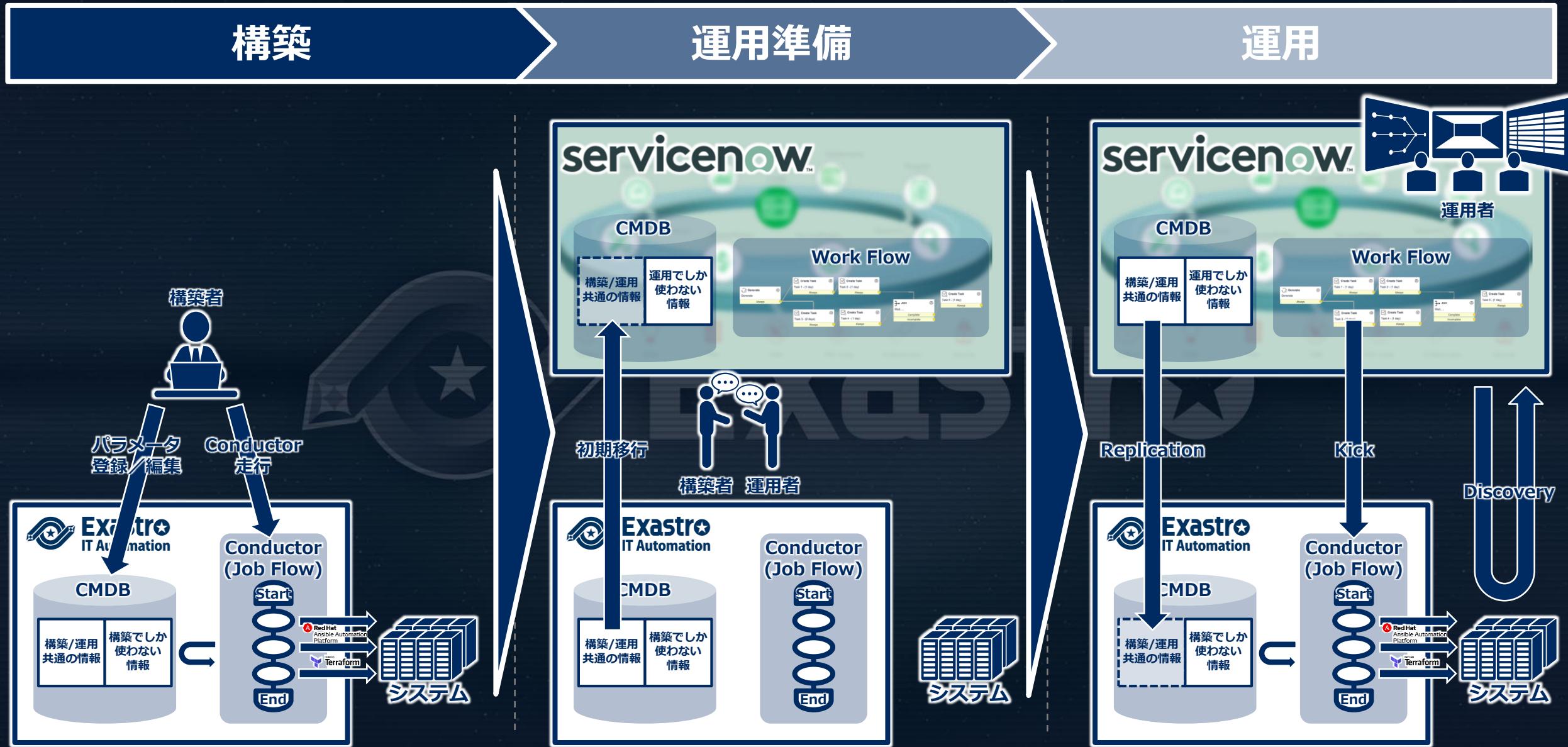
## Exastro IT Automation + AnsibleEngine



設計管理  
(インプット自動生成)

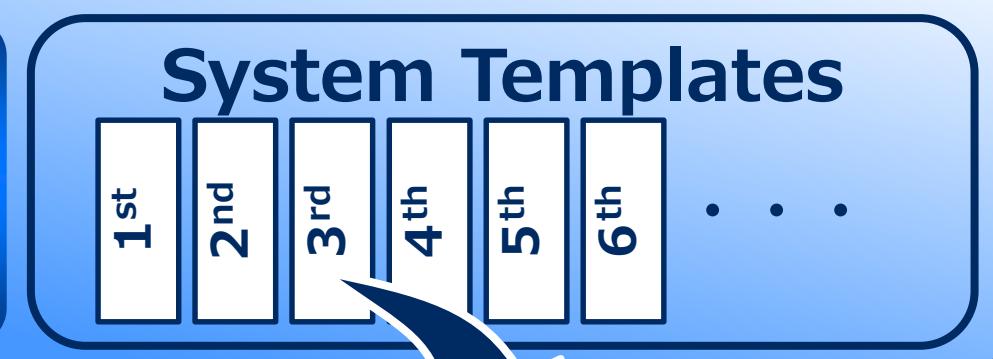
セキュリティ確保(クラスタリング)・  
複数EngineVer共存

# 【参考】Exastro ITAとITOM(e.g. ServiceNow)の役割分担



## カートリッジ式のシステムテンプレートをSetting Samplesにラインナップ

システムテンプレート  
をラインナップし  
SIをメニュー化  
**(カートリッジ式)**

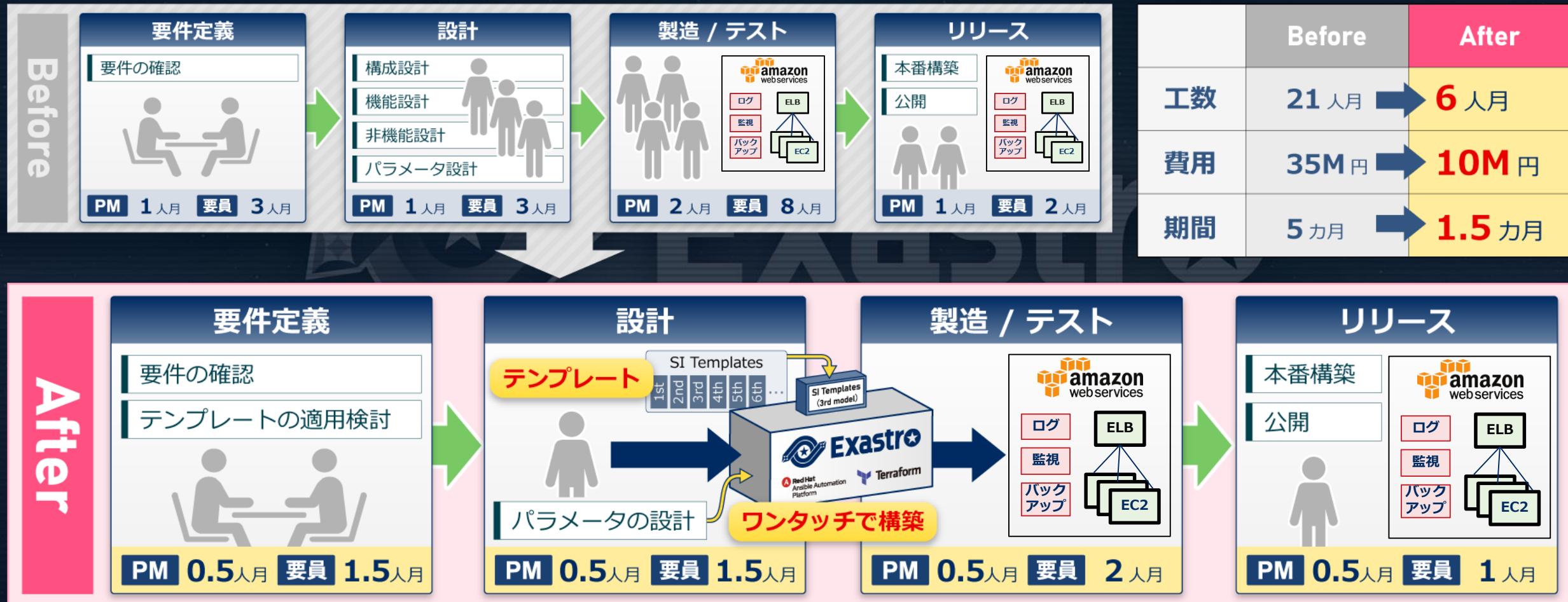


Exastroは  
システムテンプレート  
の実行基盤として  
使い易さを追求

様々なクラウド環境に  
SIメニューを開く



## Setting Samples (1<sup>st</sup> model)の導入効果(試算値)





**Exastro** 