



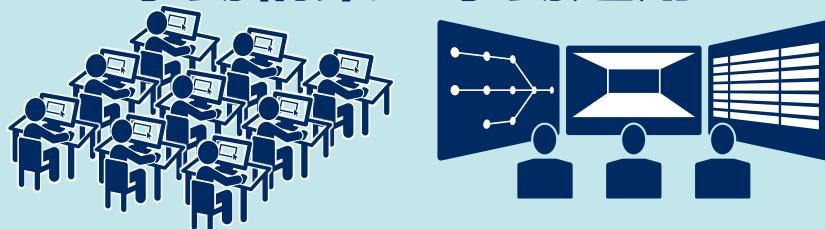
# 概要ご紹介

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

手動構築・手動運用



モノリシック  
(物理機器)

仮想化技術がまだ普及していなかった頃…

- ✓ サービス停止させないために  
**「最繁トラヒック×バッファ×冗長2倍」**  
のリソースを物理的に並べていた
- ✓ また「CPU、Mem、I/O」の選択肢が少  
なく、**リソースを使いきれなかつた**
- ✓ それでもなるべく少ない機器点数で多くの  
サービスを捌けるように、**複数機能をうま  
く混載(コンソリ)することが“正しい”とさ  
れていた**
- ✓ **アプリケーションもこれらを前提に作られ  
ていた(それが正しかつた)**

# Introduction 「攻めの自動化」「守りの自動化」とは?

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

手動構築・手動運用



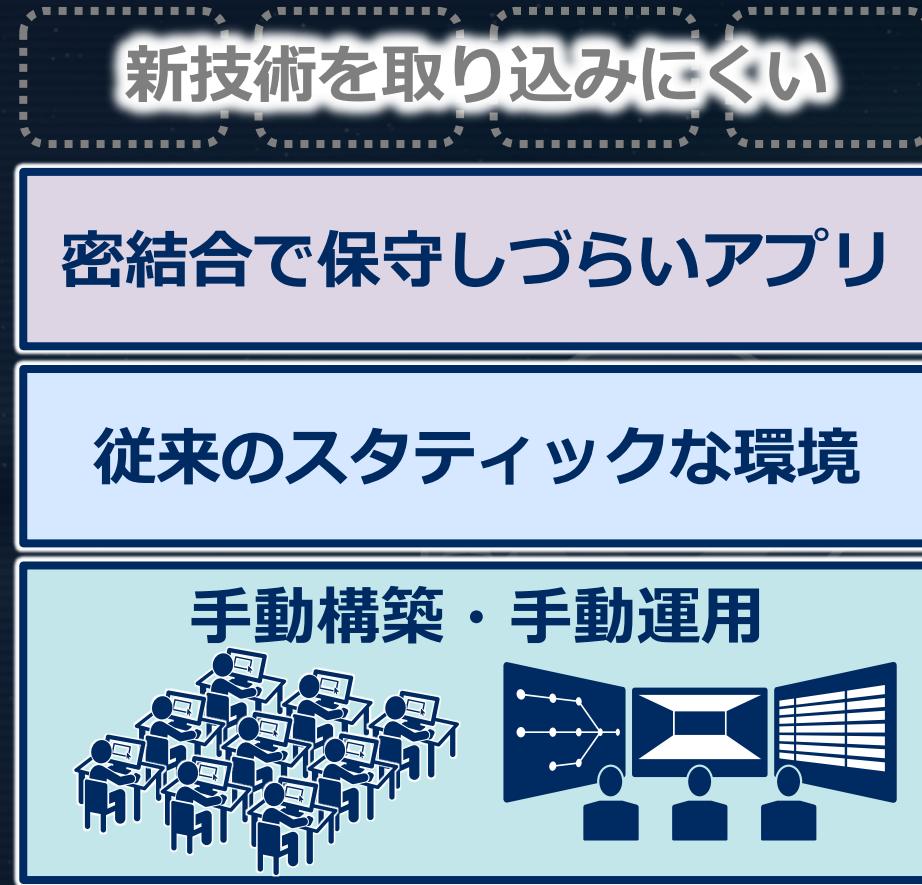
モノリシック  
(クラウドリフト)

仮想化技術という武器を手に入れた結果

- ✓ 「CPU、Mem、I/O」が選択可能に
- ✓ 「オートスケールWeb」くらいの簡単な動的システムは組めるようになった
- ✓ 結果、システムサイズはコンパクトになりハードウェア寿命から脱却できた
- ✓ 複数機能を混載(コンソリ)させないことが“正しい”と変わり始めた

しかし、以下の課題は解決されず…

- ✓ アプリケーションを作り直さない限り多くの機能で冗長2倍は継続  
→クラウドシフトへの期待 … ①
- ✓ 根本的にはシステム形状は変わらないので運用の手間は大きくは変わらない  
→運用自動化への期待 … ②



モノリシック  
(クラウドリフト)

## クラウドシフトへの期待(①)

### OPEXの効率化

アプリ設計の段階から自動化を組み込むことで、必要な時に必要なリソースを使う「**本格的な動的システム**」  
(リソースの冗長確保からの脱却)

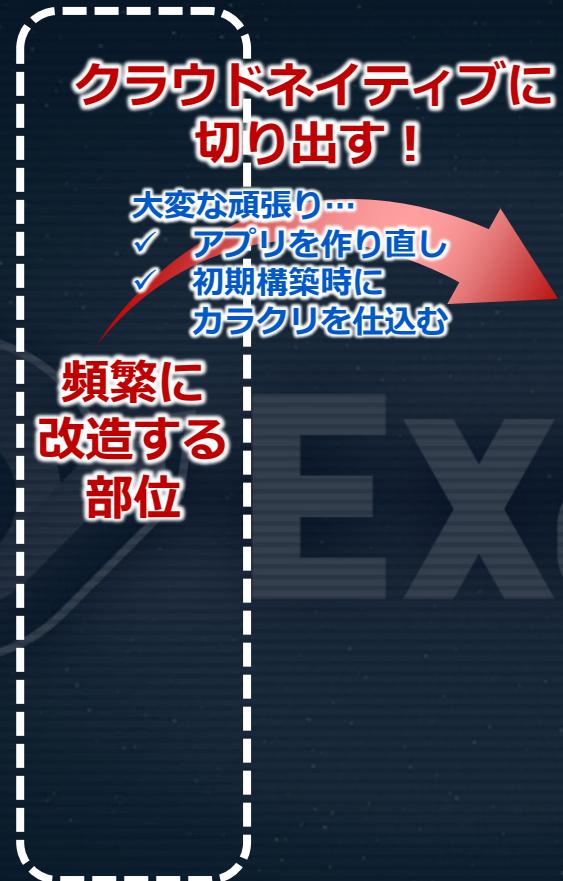
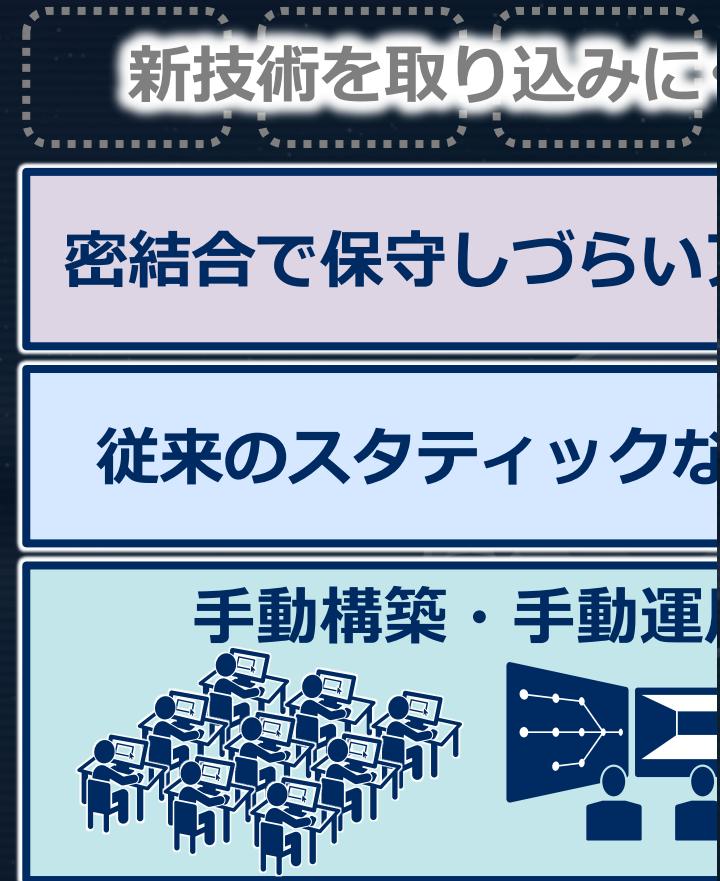
### ROI(投資利益率)の引き上げ

アプリ設計の段階から自動化を組み込むことで、運用しながら機能追加できる  
「**廃棄容易なシステム**」(DevOpsの実現)

## クラウドシフトの問題点

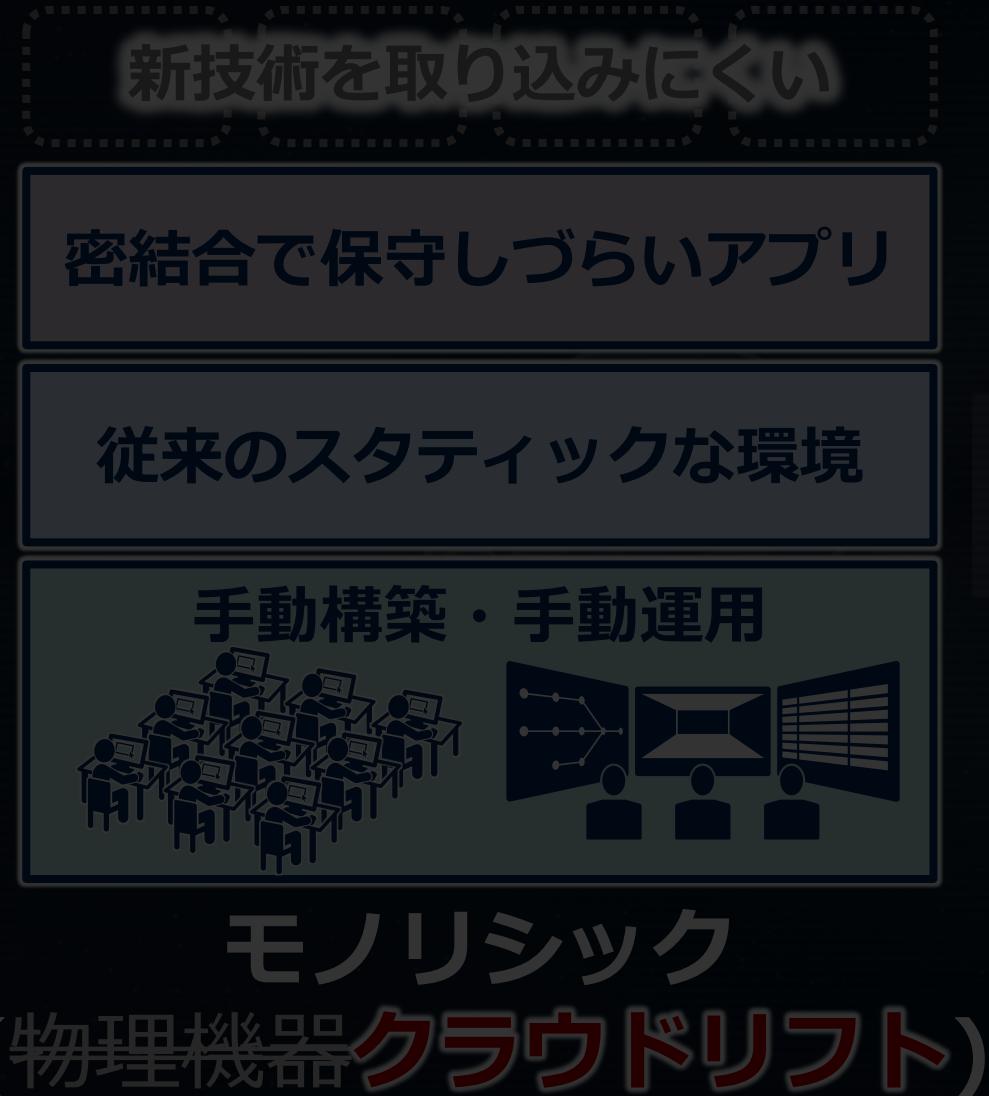
- ✓ アプリの作り直しを伴う
- ✓ 初期構築時にカラクリを仕込むのが大変  
⇒とにかく敷居が高い！  
頻繁に改造したい部位を選択しなければ…

# Introduction 「攻めの自動化」「守りの自動化」とは？



クラウドネイティブ

# Introduction 「攻めの自動化」「守りの自動化」とは？



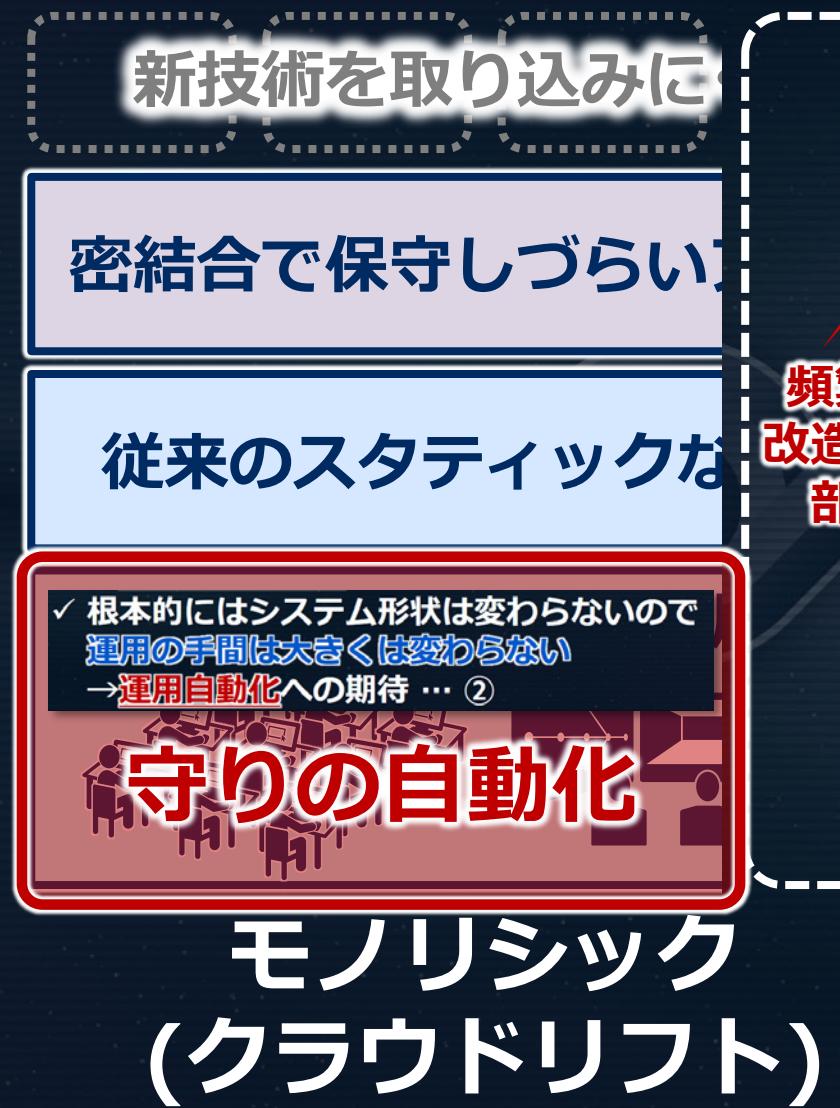
## 仮想化技術という武器を手に入れた結果

- ✓ 「CPU、Mem、I/O」が選択可能に
- ✓ 「オートスケールWeb」くらいの簡単な動的システムは組めるようになった
- ✓ 結果、システムサイズはコンパクトになりハードウェア寿命から脱却できた
- ✓ 複数機能を混載(コンソリ)させないことが“正しい”と変わり始めた

## しかし、以下の課題は解決されず…

- ✓ アプリケーションを作り直さない限り多くの機能で冗長2倍は継続  
→クラウドシフトへの期待 … ①
- ✓ 根本的にはシステム形状は変わらないので運用の手間は大きくは変わらない  
→運用自動化への期待 … ②

# Introduction 「攻めの自動化」「守りの自動化」とは?



クラウドネイティブ

# 「攻めの自動化」の進め方



ところで…

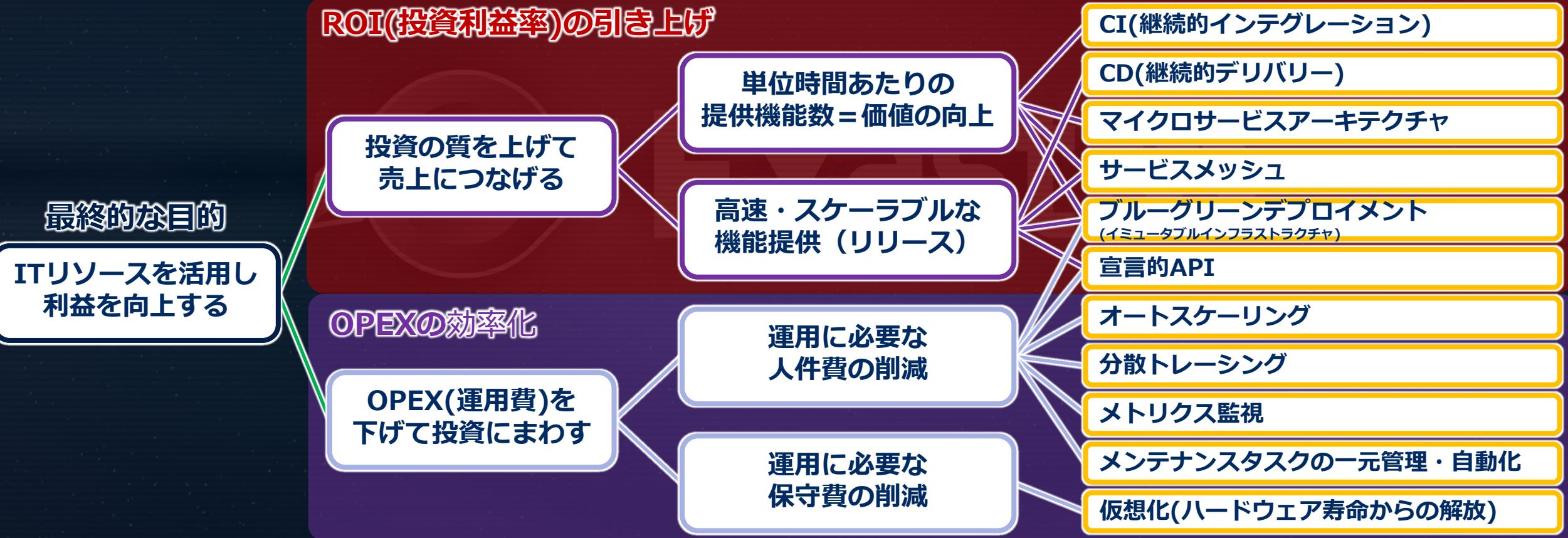
クラウドネイティブのカラクリとは？  
初期構築時にどんなカラクリを仕込んでおけば、  
気持ちよく DevOps をまわせる？

## クラウドシフトで期待できること

人

## 11のカラクリ (10の技 + 仮想化)

### クラウドネイティブの実現手段



## クラウドシフトで期待できること

11のカラクリ  
(10の技+仮想化)

クラウドネイティブの実現手段

### ROI(投資利益率)の引き上げ

投資の質を上げて  
売上につなげる

単位時間あたりの  
提供機能数=価値の向上

高速・スケーラブルな  
機能提供(リリース)

CI(継続的インテグレーション)

CD(継続的デリバリー)

マイクロサービスアーキテクチャ

サービスメッシュ

ブルーグリーンデプロイメント  
(immutable infrastructure)

宣言的API

最終的な目的

ITリソースを活用し  
利益を向上する

OPEXの効率化

OPEX(運用費)を  
下げて投資にまわす

特にこの6つの手段が  
「ROI(投資利益率)の引き上げ」という  
クラウドネイティブ最大のテーマに直結する

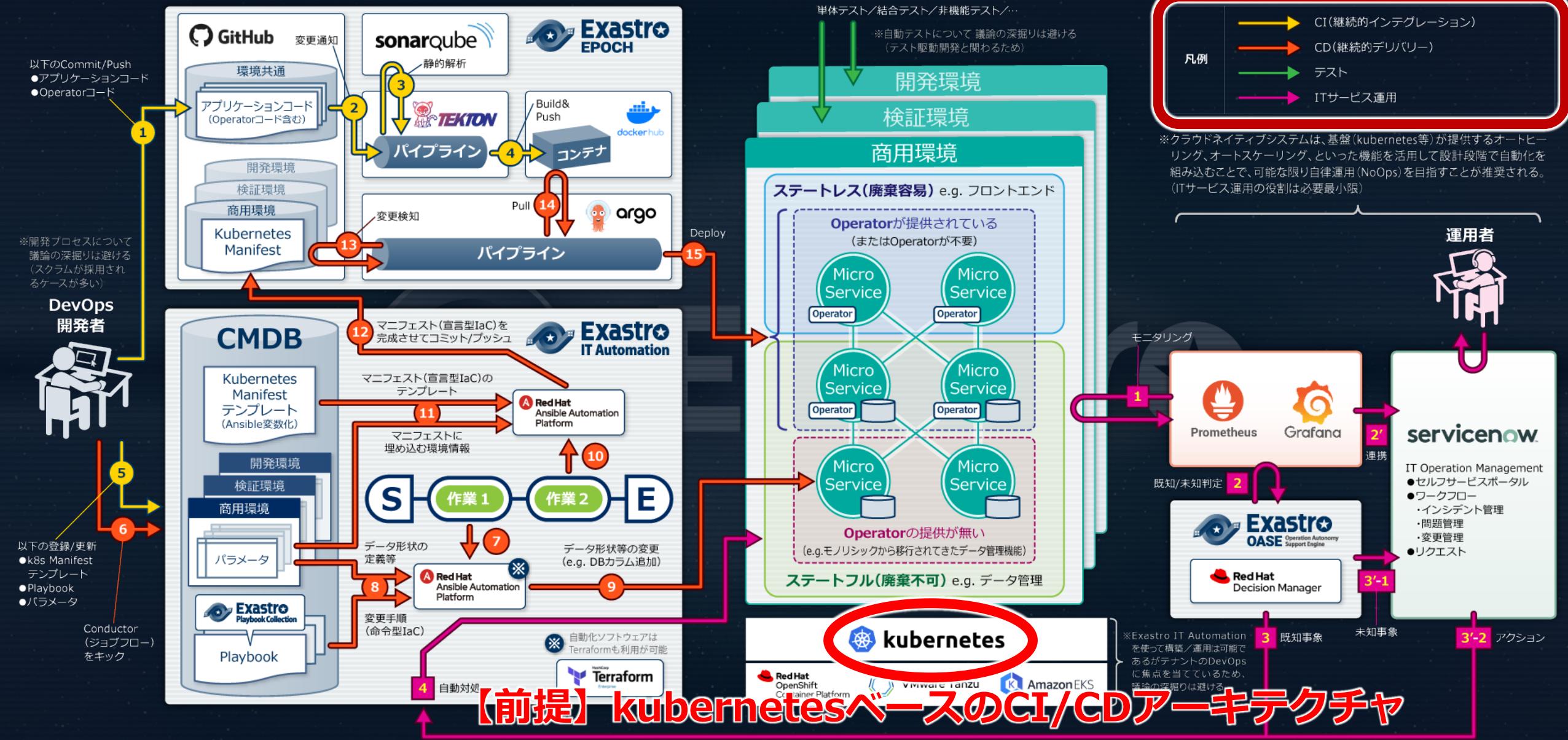
保守費の削減

オートスケーリング

分散トレーシング

仮想化(ハードウェア寿命からの解放)

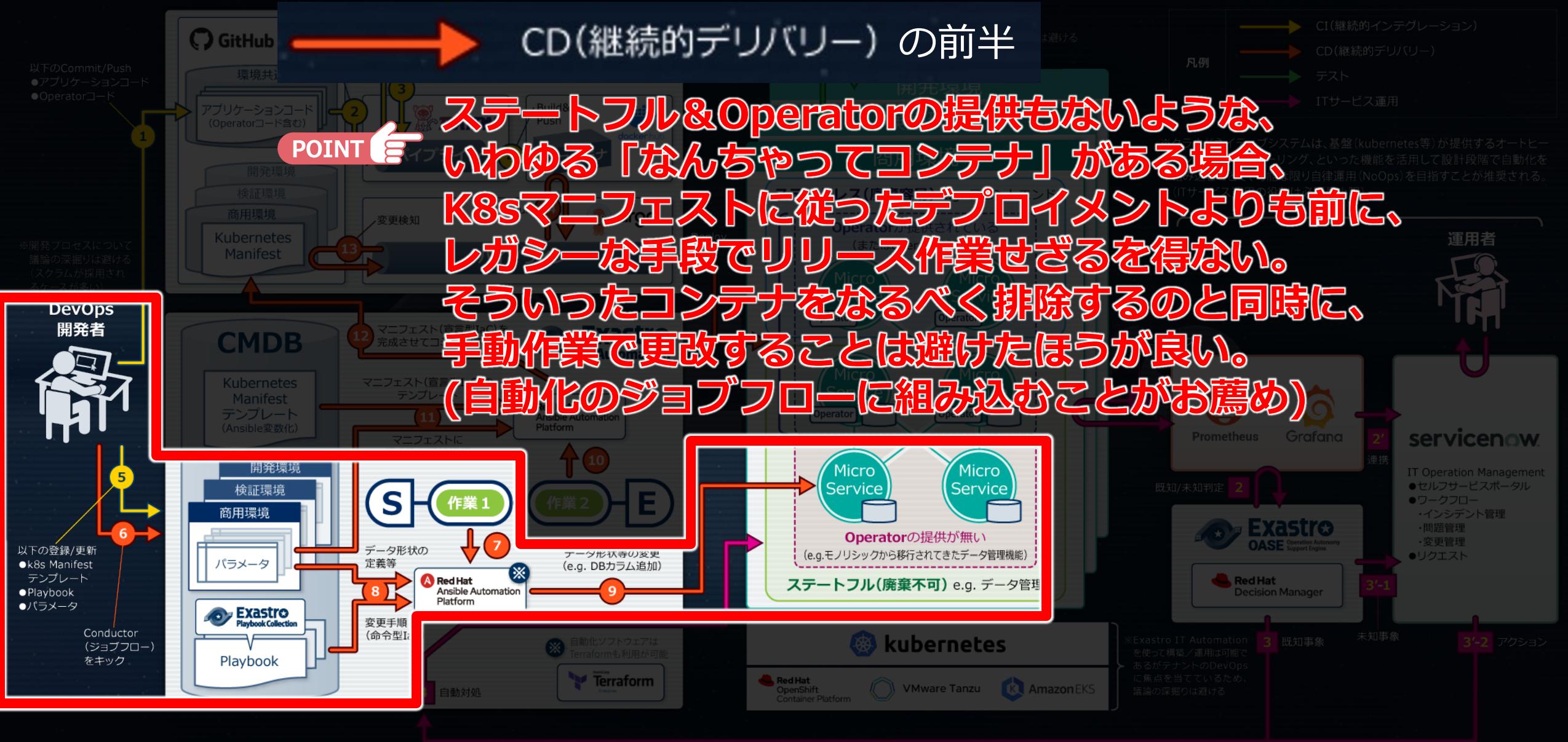
# 「攻めの自動化」の進め方



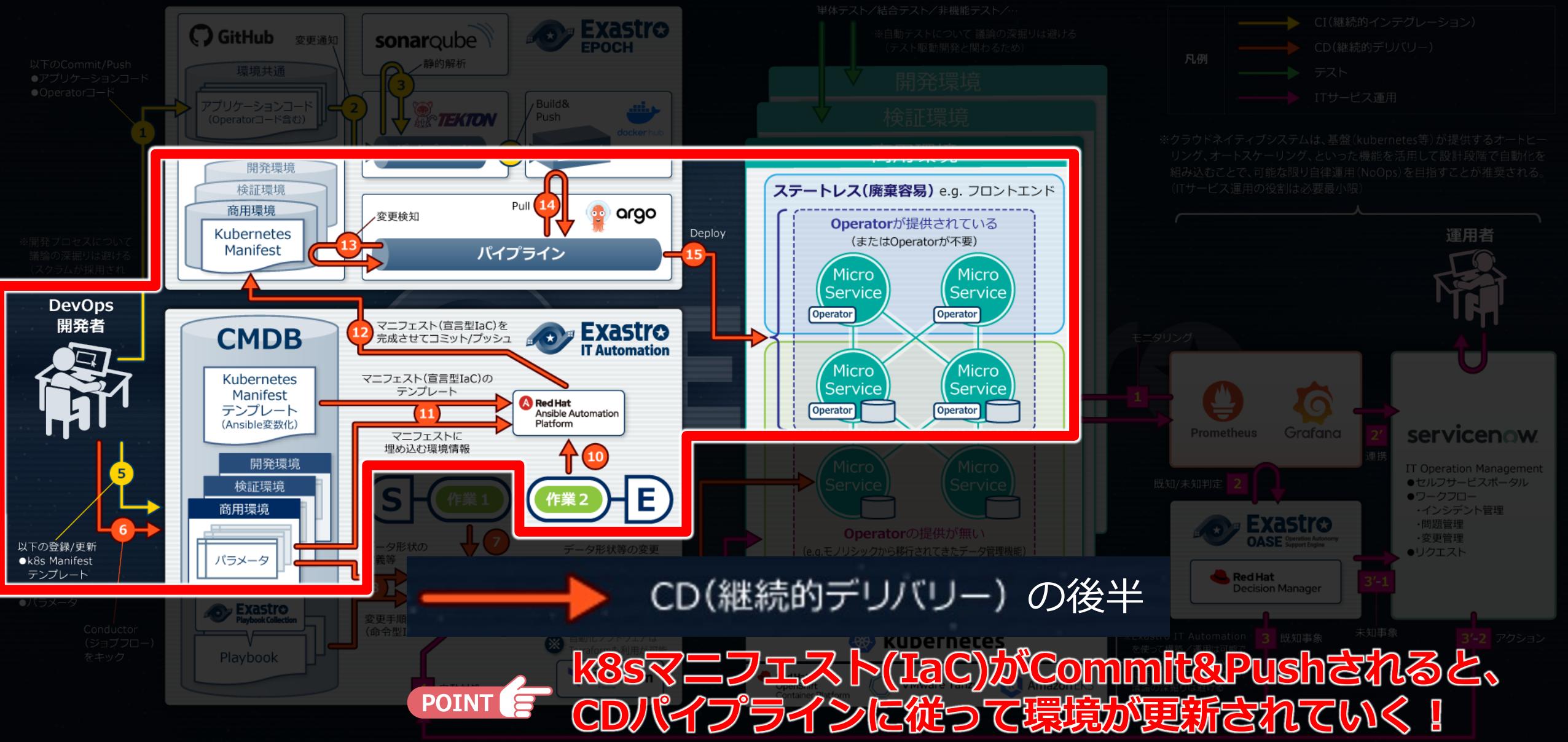
# 「攻めの自動化」の進め方



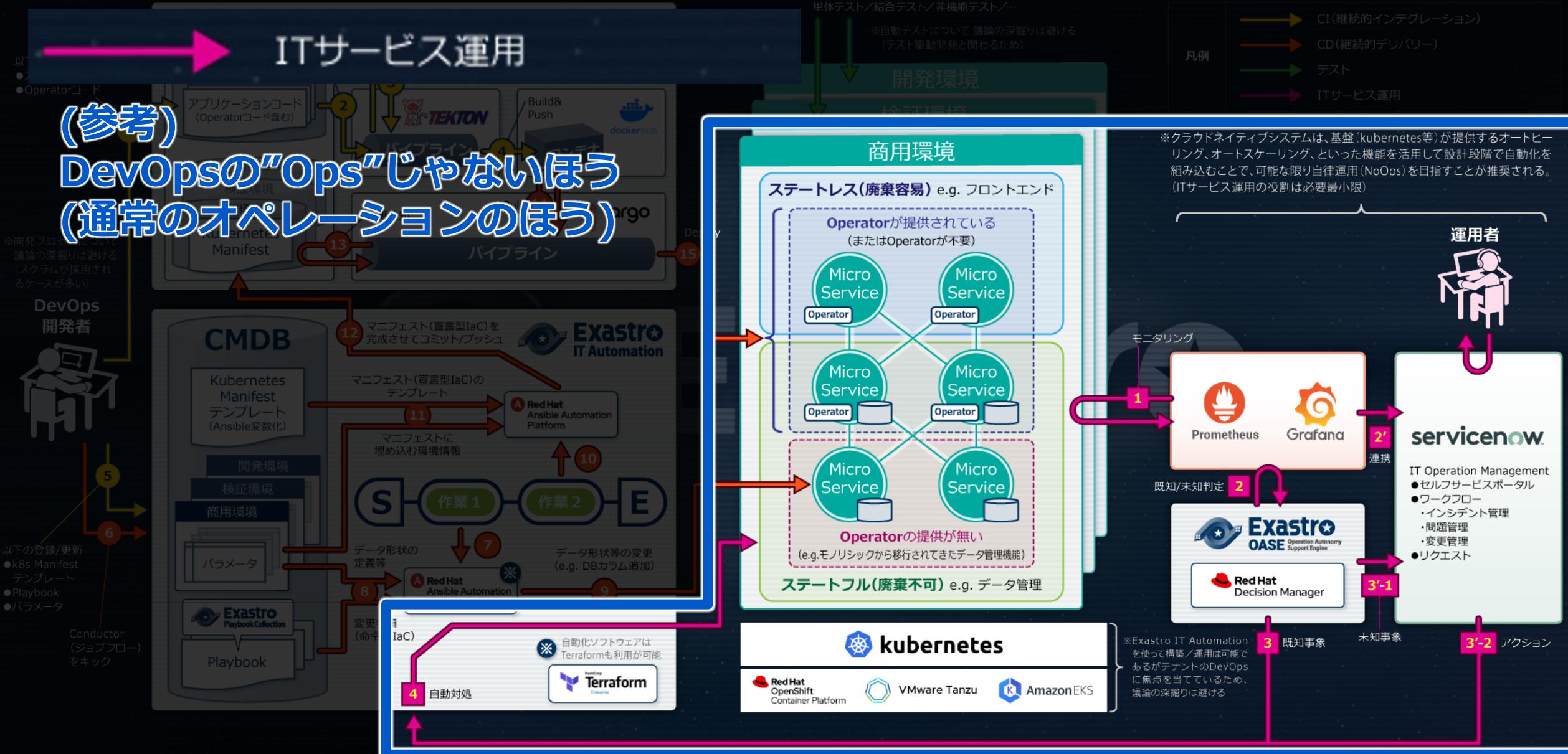
## 「攻めの自動化」の進め方



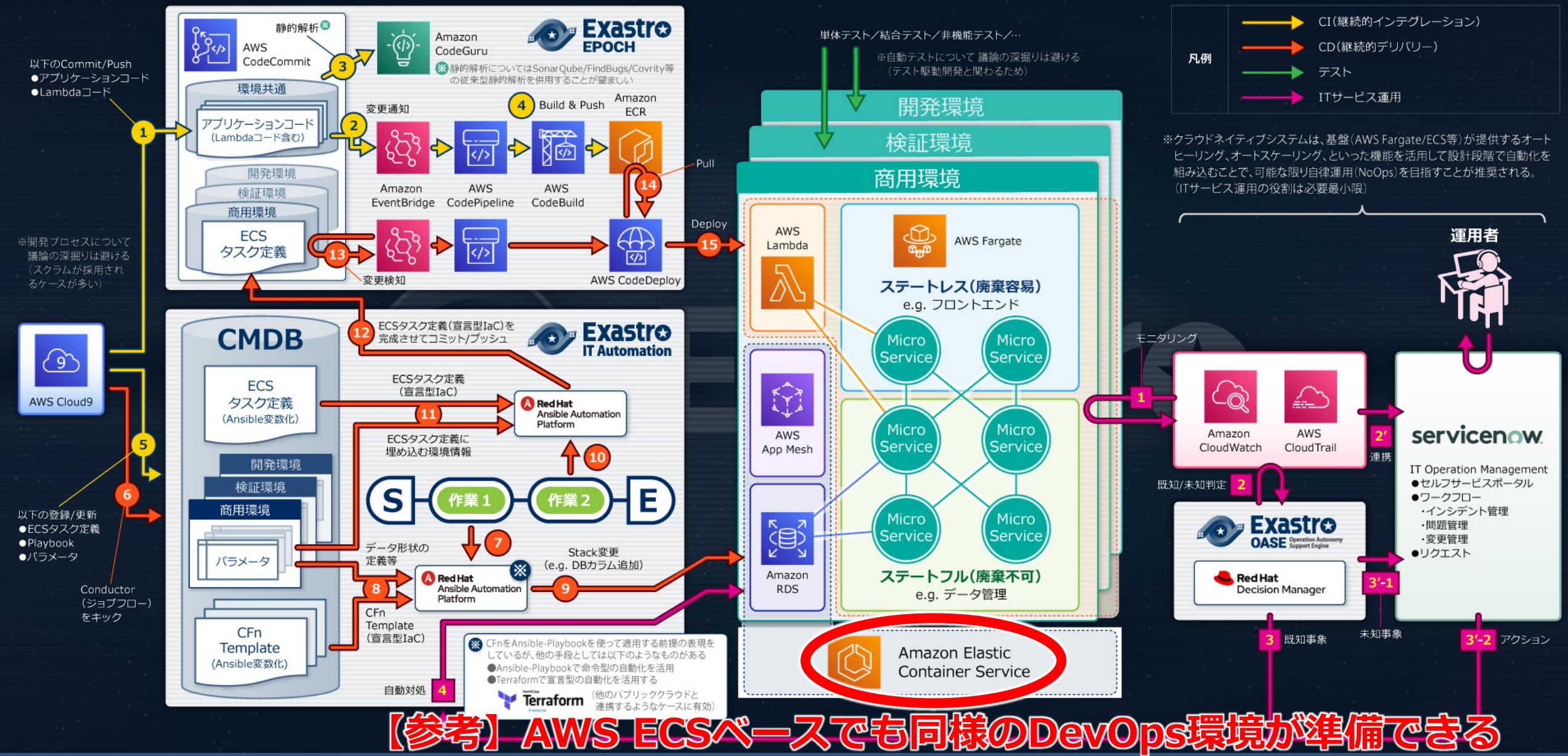
# 「攻めの自動化」の進め方



# 「攻めの自動化」の進め方



## 「攻めの自動化」の進め方



# 「守りの自動化」の進め方



## 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



- 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

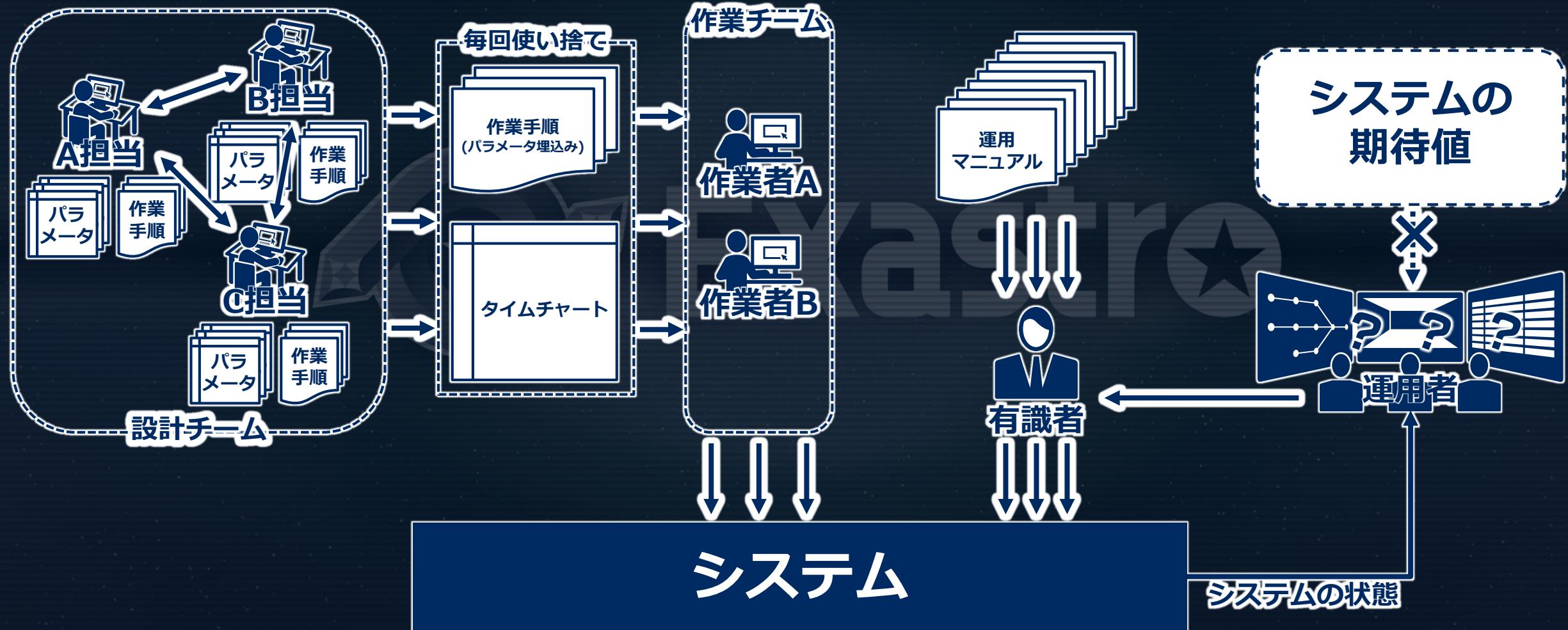
- システムは複雑化の一途を辿っており作業量は増大するばかり
- 何か起こるとExcelで書かれた大量のマニュアルを読み替えながら複数体制で1作業ずつ慎重に実行するしかない
- 結果としてシステムの故障時間が長くなりサービスにも影響が出る



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

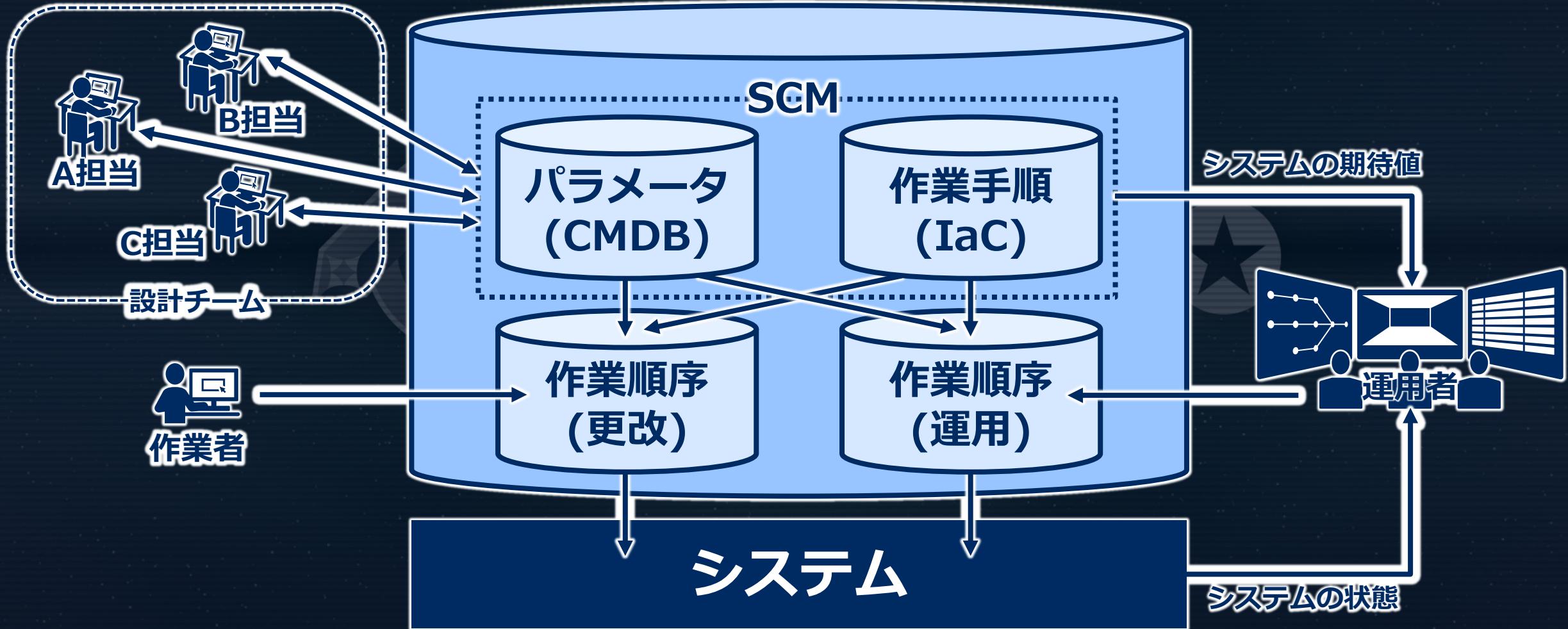
関係者は散乱した情報を正確に伝えることに多くの時間を費やしています



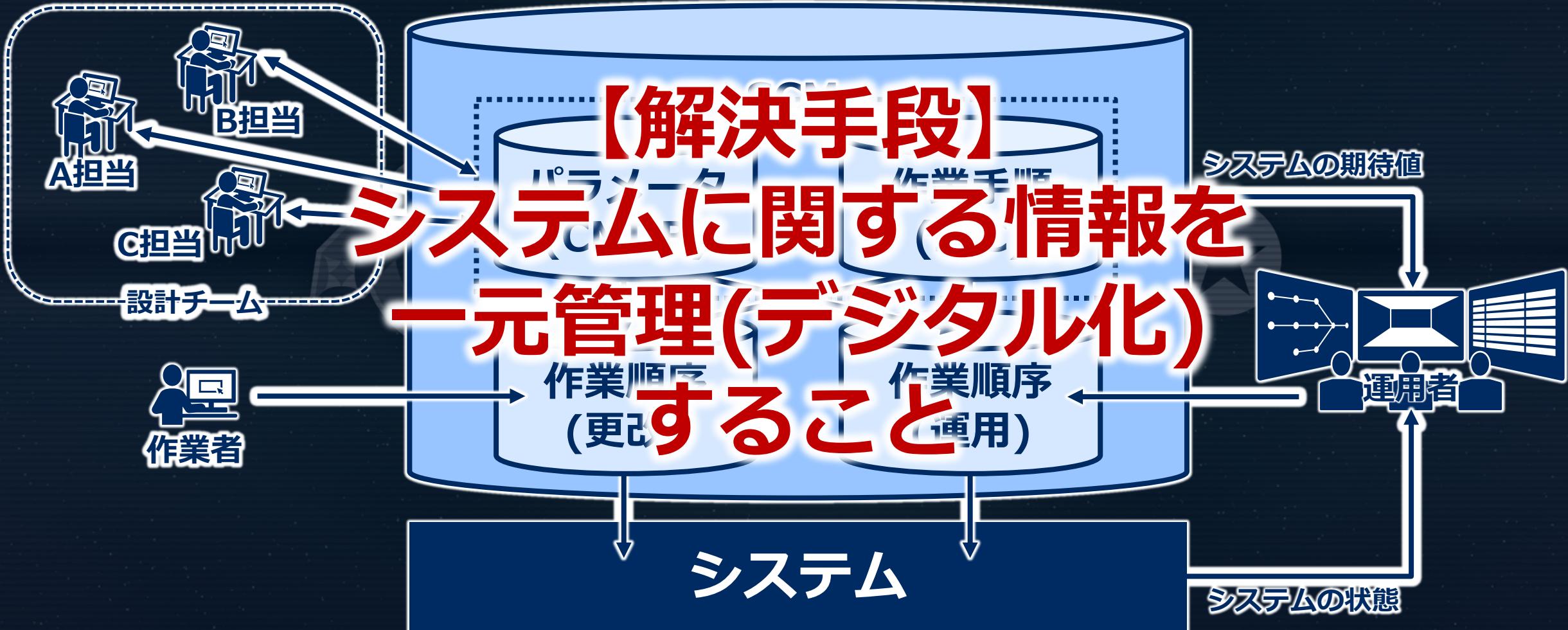
関係者は散乱した情報を正確に伝えることに多くの時間を費やしています



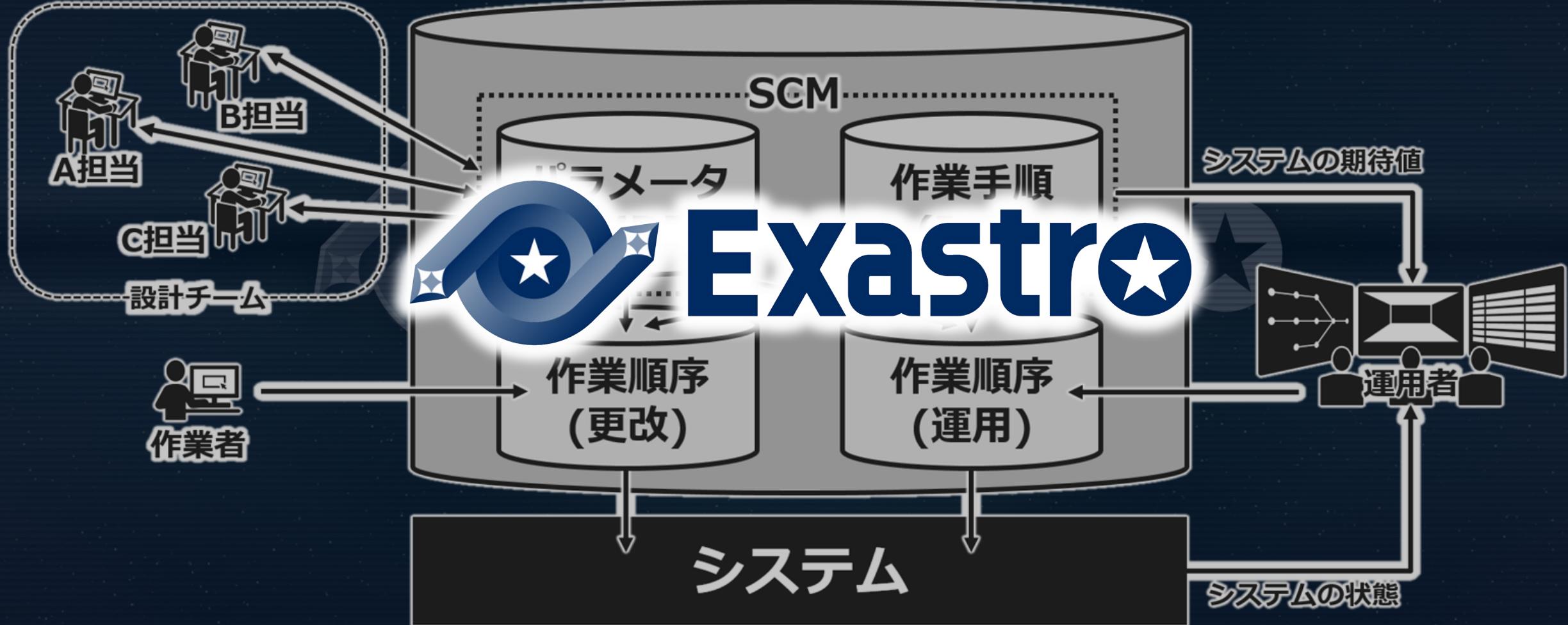
システムに関する情報をデジタル化して一元管理すればよいのですが…



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…



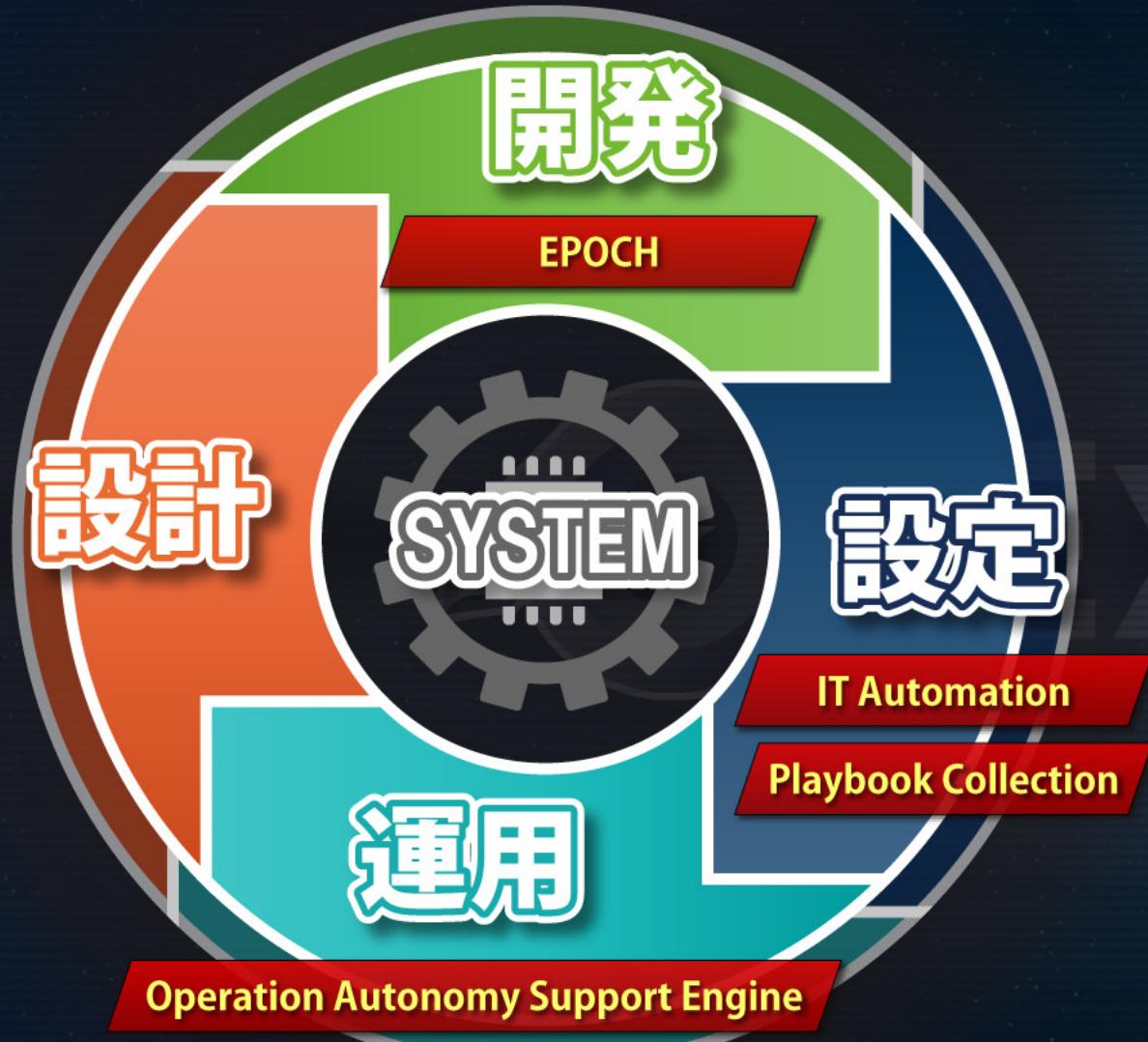
Exastroはシステム情報をデジタル化して一元管理するのに役立ちます



# Exastro Suite



# Exastro Suiteとは？



Exastroはシステムライフサイクルをデジタル化・自動化・省力化することを目的としたオープンソースのソフトウェアスイートです。

# Exastro Suiteが目指す自動化・自律化のロードマップ

## Exastro Suiteが目指す自動化・自律化のロードマップ



詳しくはコミュニティーサイトへ！



# Exastro

 Search

Exastro

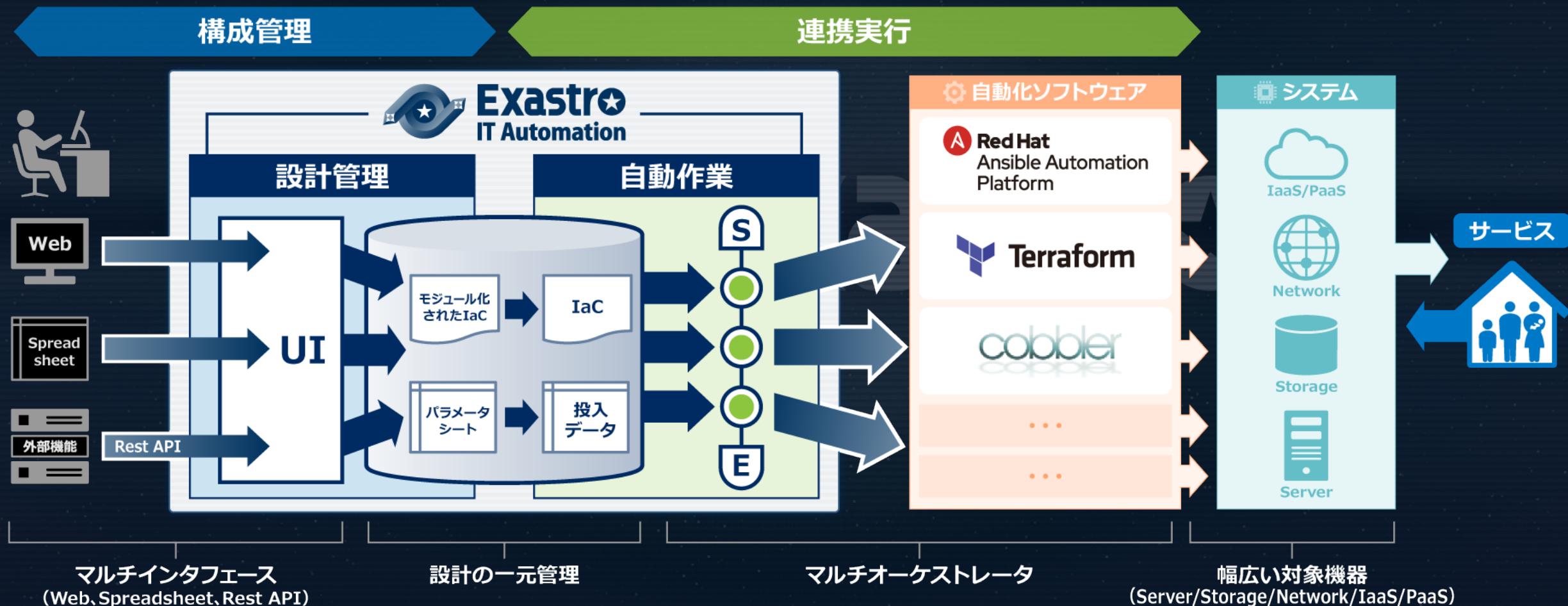
[https://exastro-suite.github.io/docs/index\\_ja.html](https://exastro-suite.github.io/docs/index_ja.html)



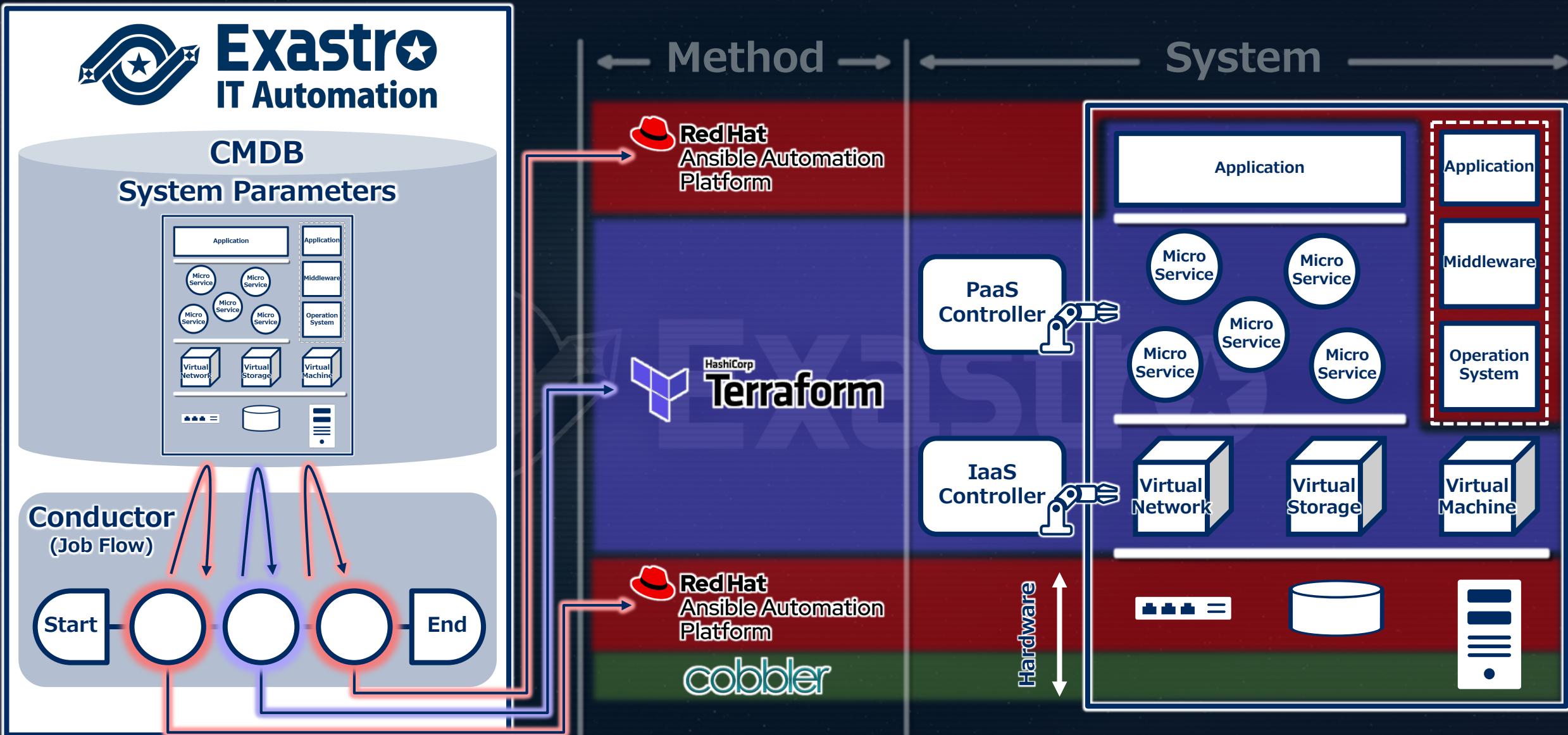
# Exastro IT Automation



## Exastro IT Automationは 「システム構成(IaC+パラメータ)を管理するためのフレームワーク」です



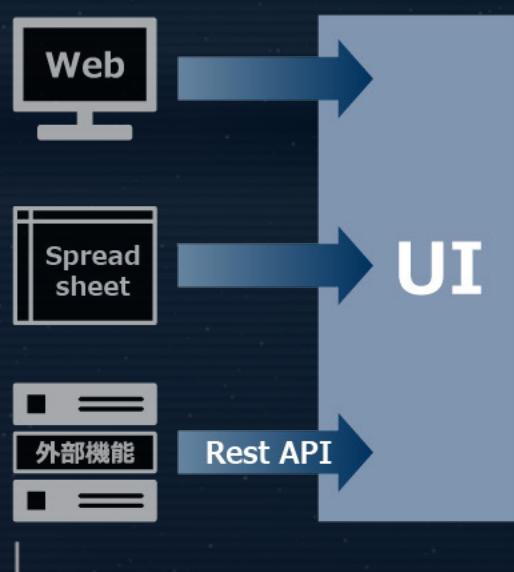
# Exastro IT Automation : システムスタックと使用するメソッド



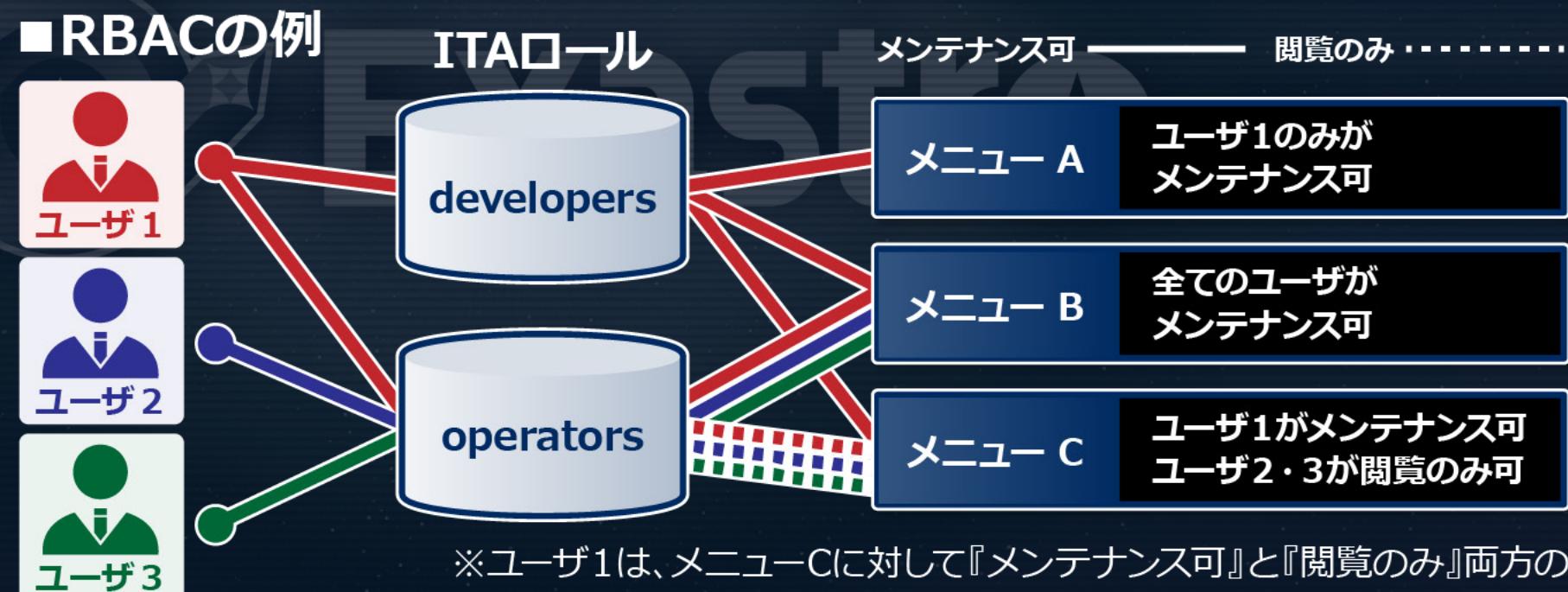
- 
- 1 マルチインターフェースとRBAC
  - 2 パラメータをグルーピング／履歴管理する
  - 3 IaCを解析して変数を切り取る
  - 4 IaCをモジュール管理して再利用性を高める
  - 5 複数の自動化ソフトウェアを繋げて実行する
  - 6 自動化を止めない最後の切り札Pioneerモード
  - 7 実行状況をリアルタイムで監視する

# 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



マルチインターフェース  
(Web, Spreadsheet, Rest API)



## 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

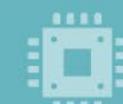
“10/9” のシステムの期待値

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

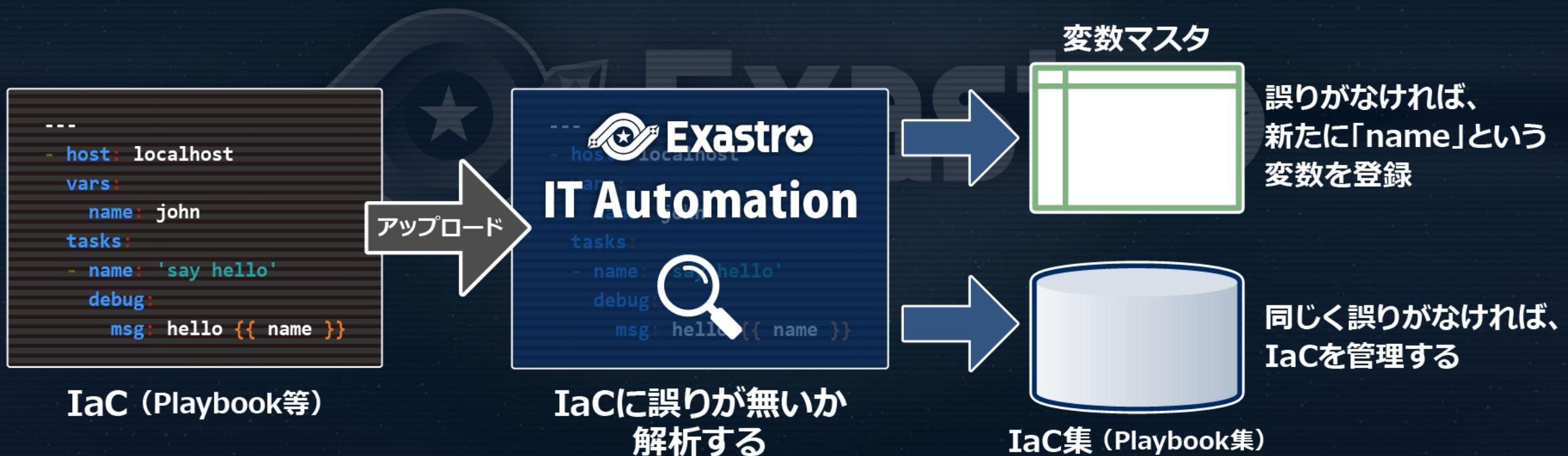
妥当性確認



システム

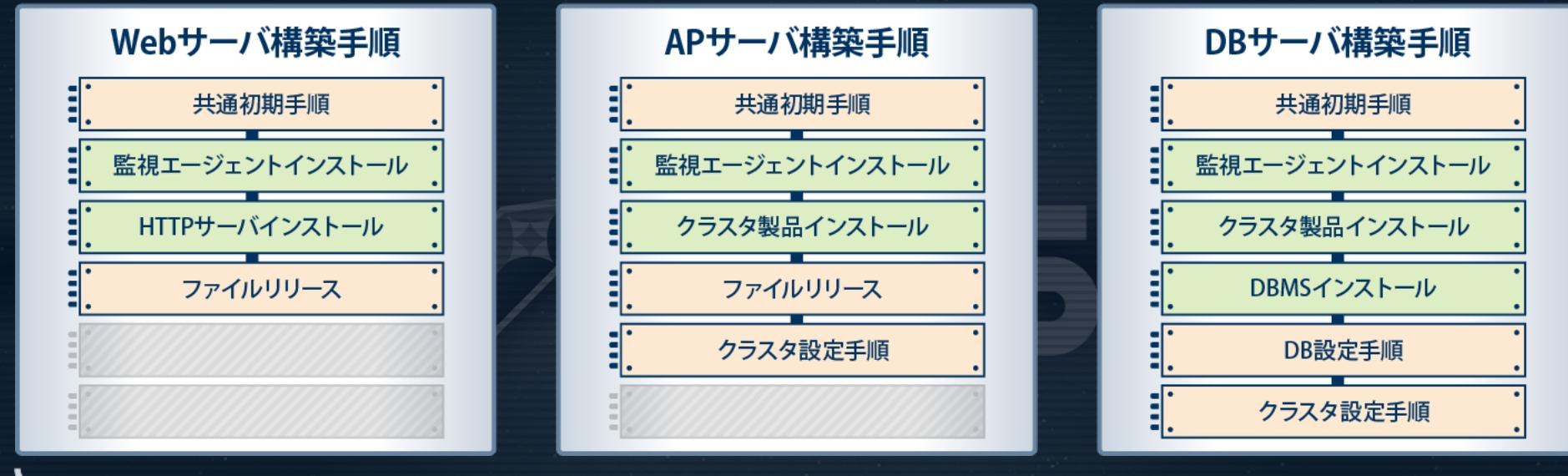
## 3つめの特徴 - IaCを解析して変数を刈り取る

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

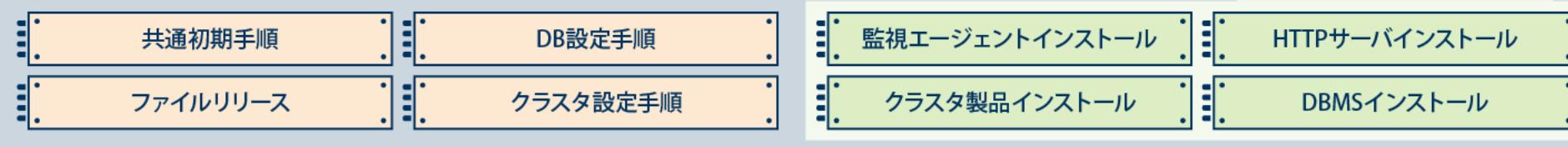


## 4つめの特徴 - IaCをモジュール管理して再利用性を高める

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



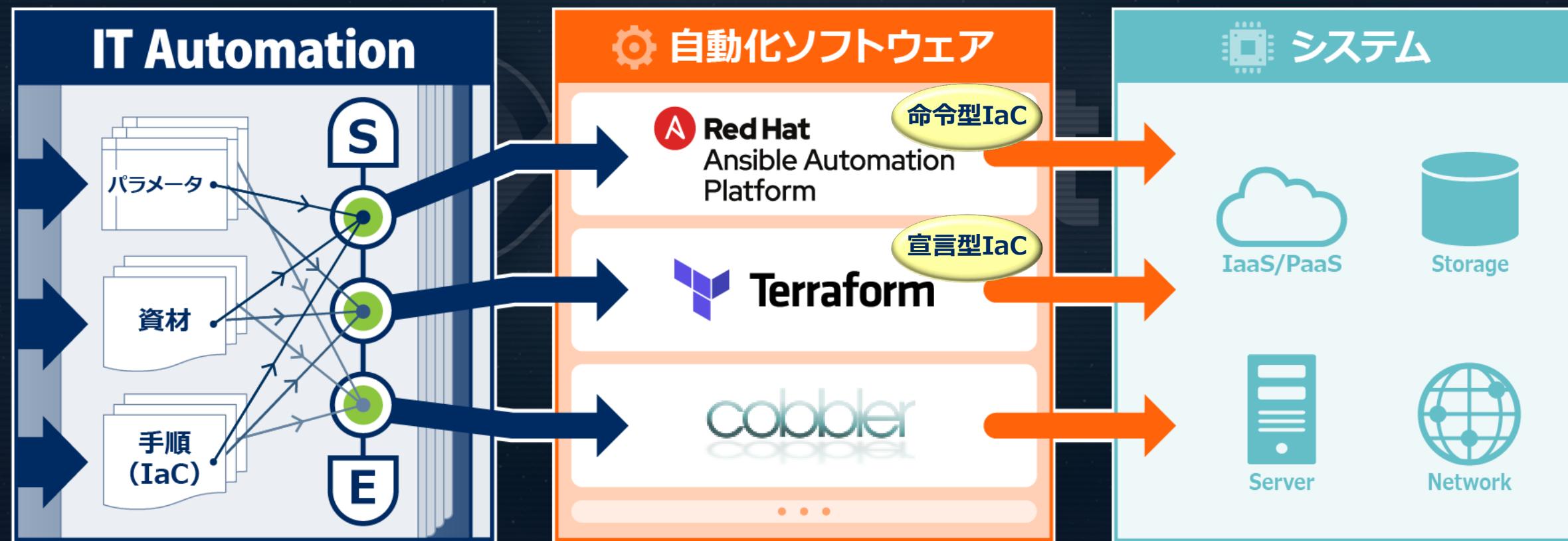
共通の手順はモジュール化し再利用できるように管理する



## 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」



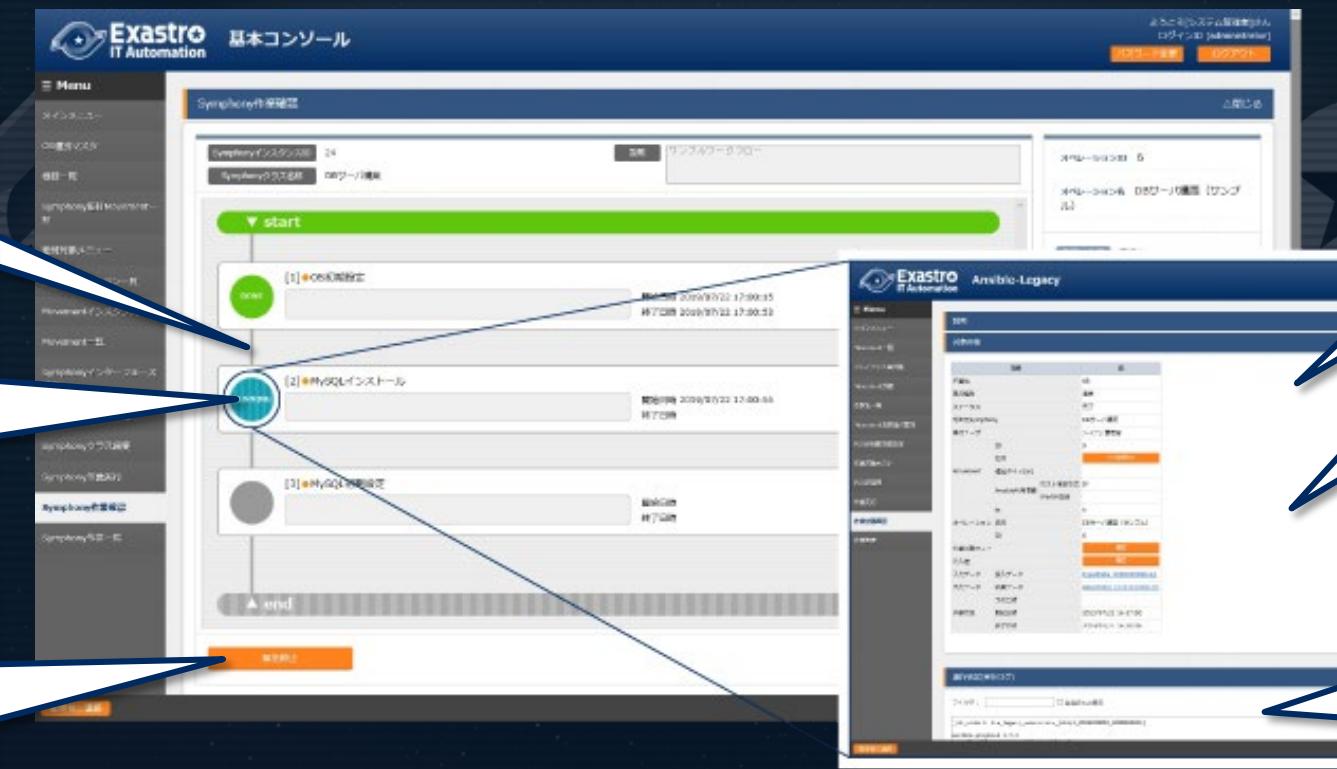
## 7つの特徴：⑦実行状況をリアルタイムで監視する

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

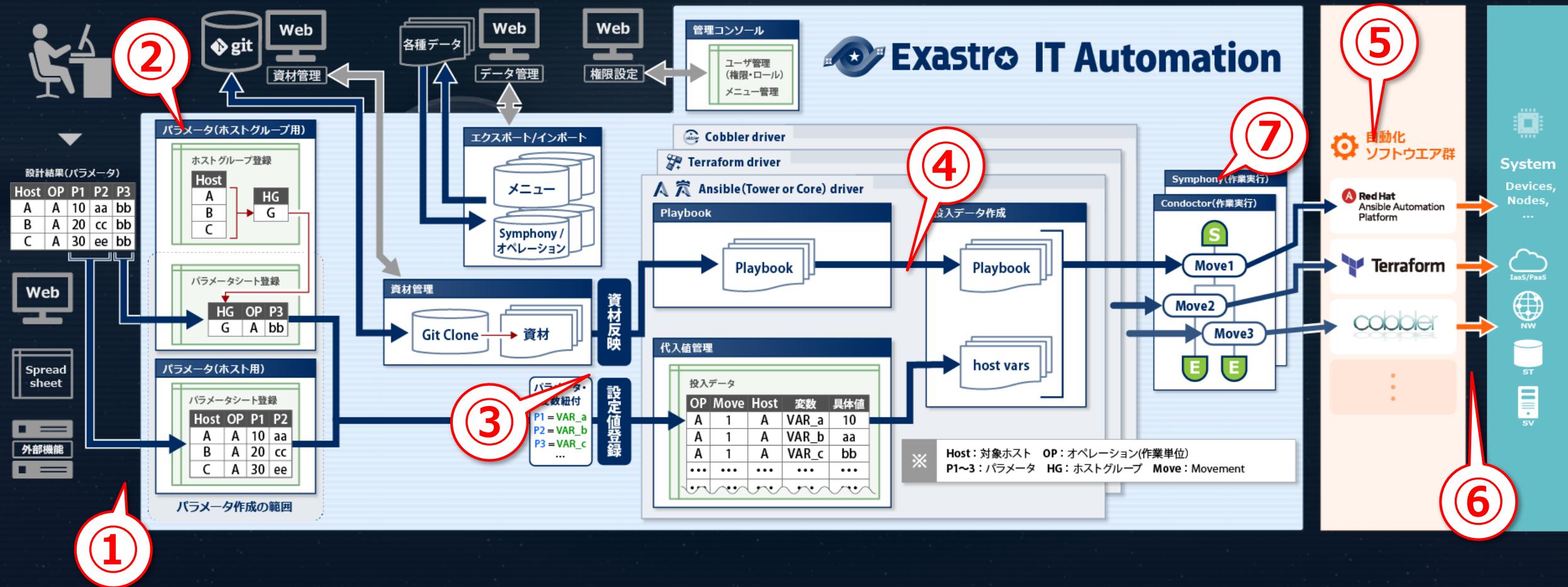


投入データ(自動生成)  
がダウンロード可能  
(zip)

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

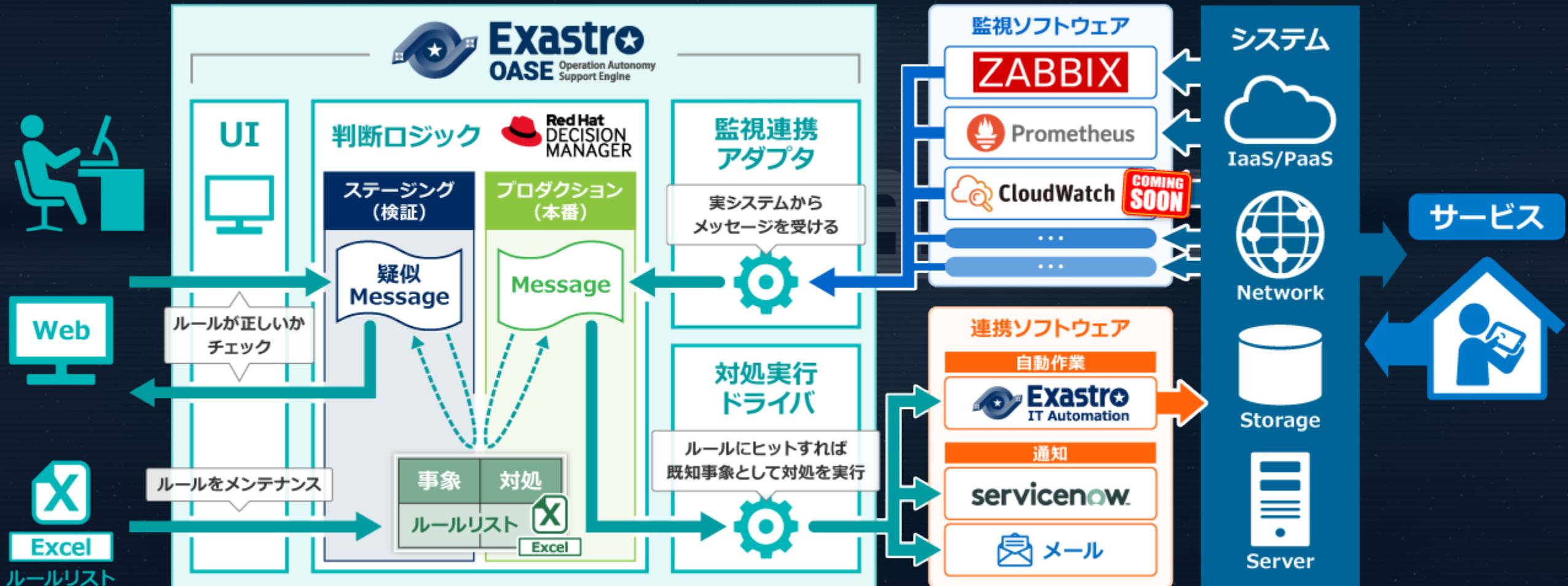
ご紹介した「7つの特徴」の他にも様々な工夫を凝らして、システム情報をデジタル管理できるようにしています。

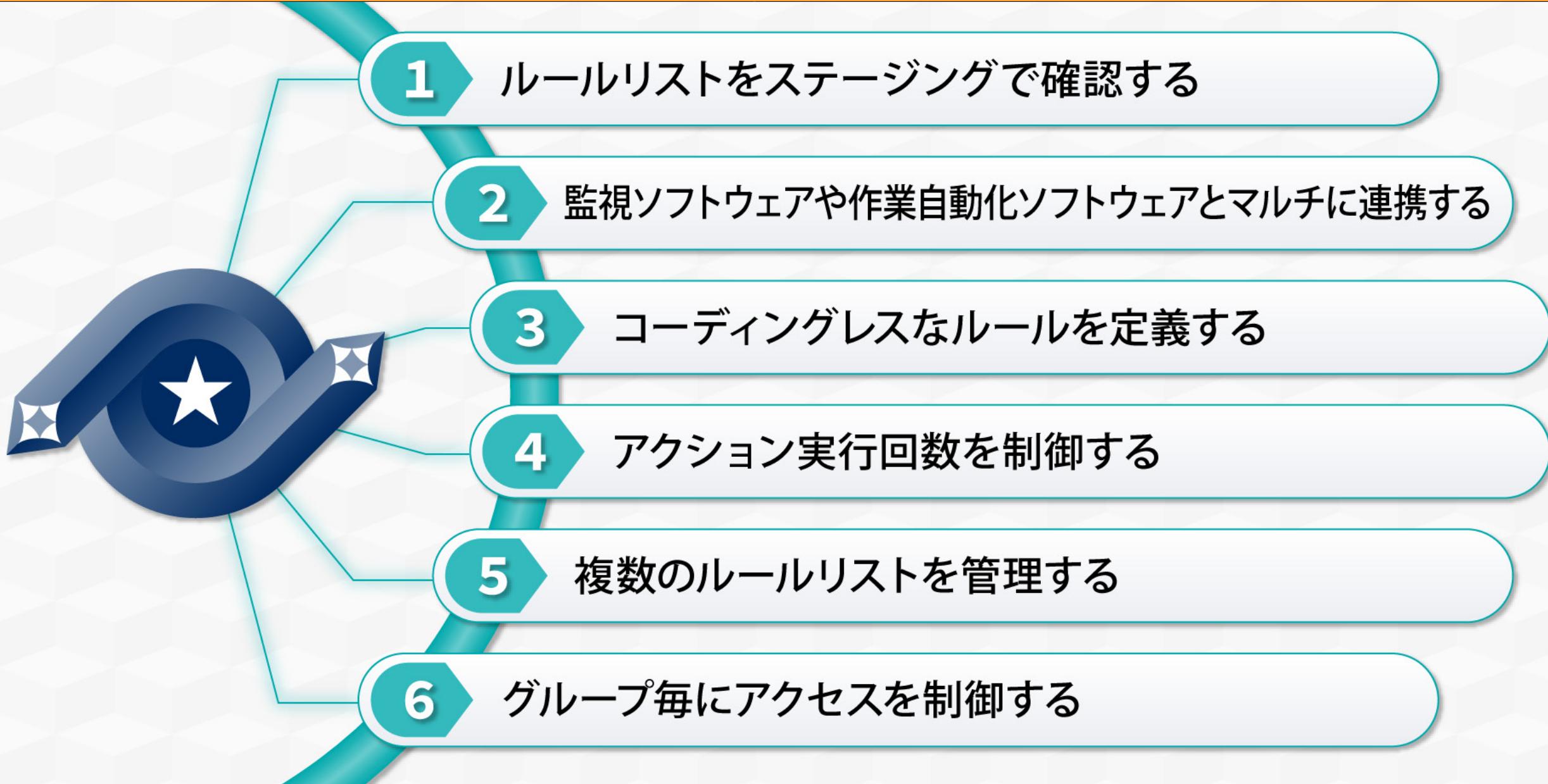


# Exastro OASE



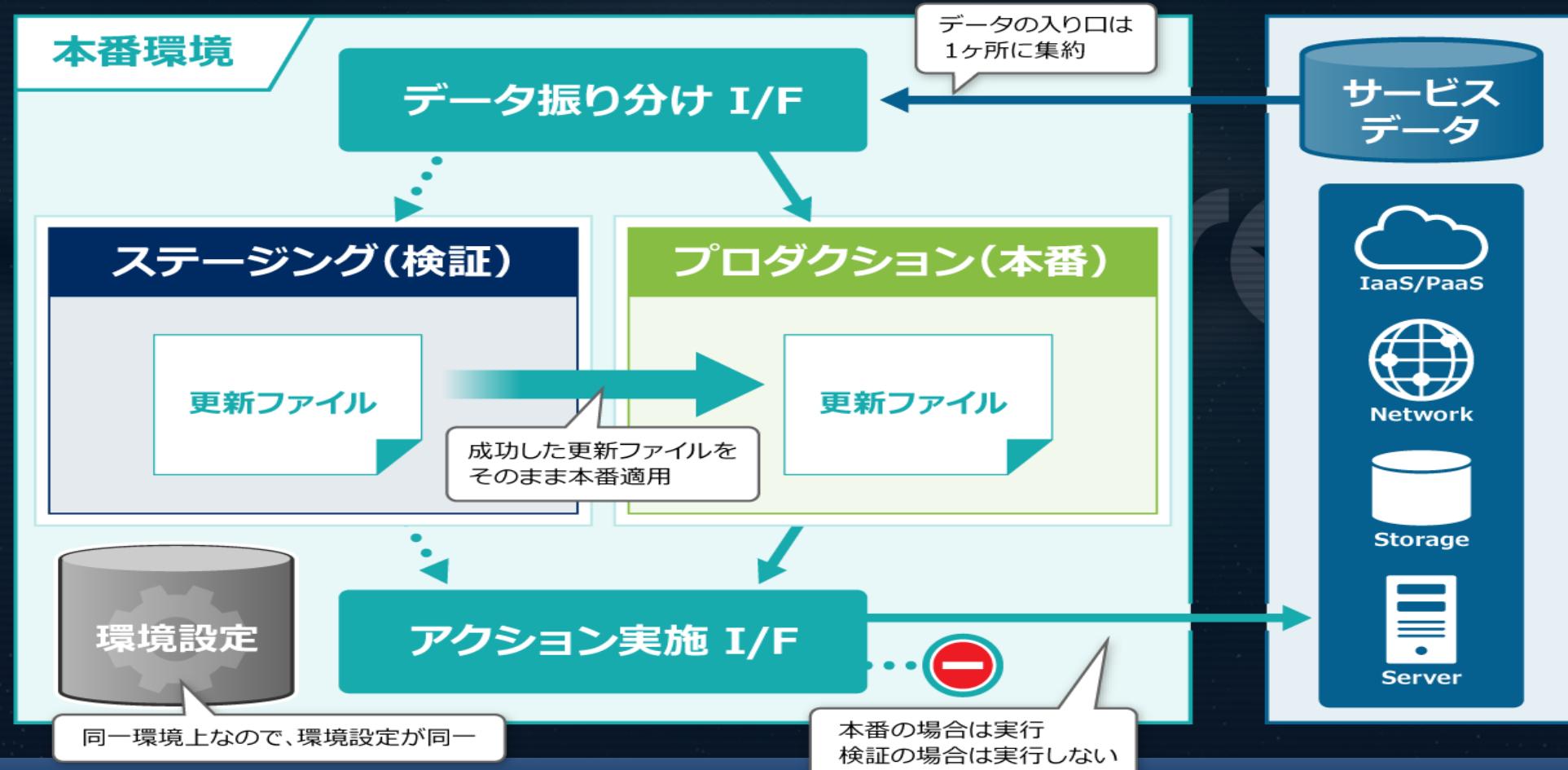
## Exastro Operation Autonomy Support Engineは 「システム運用の自動化を支援するためのフレームワーク」です





## 6つの特徴：①ルールリストをステージングで確認

ステージング確認ではルールリスト適用前に実際のアクションの直前までを実行し妥当性を確認できます。



6つの特徴：②監視ソフトウェアや作業自動化ソフトウェアとマルチに連携する

様々なツールとの連携ドライバやRestAPIを具備しているためマルチな連携ができます。

### 監視ソフトウェア

ZABBIX

Prometheus

CloudWatch

COMING SOON

+ 独自ツール



### 連携ソフトウェア

Exastro  
IT Automation

servicenow.

メール

+ 独自ツール

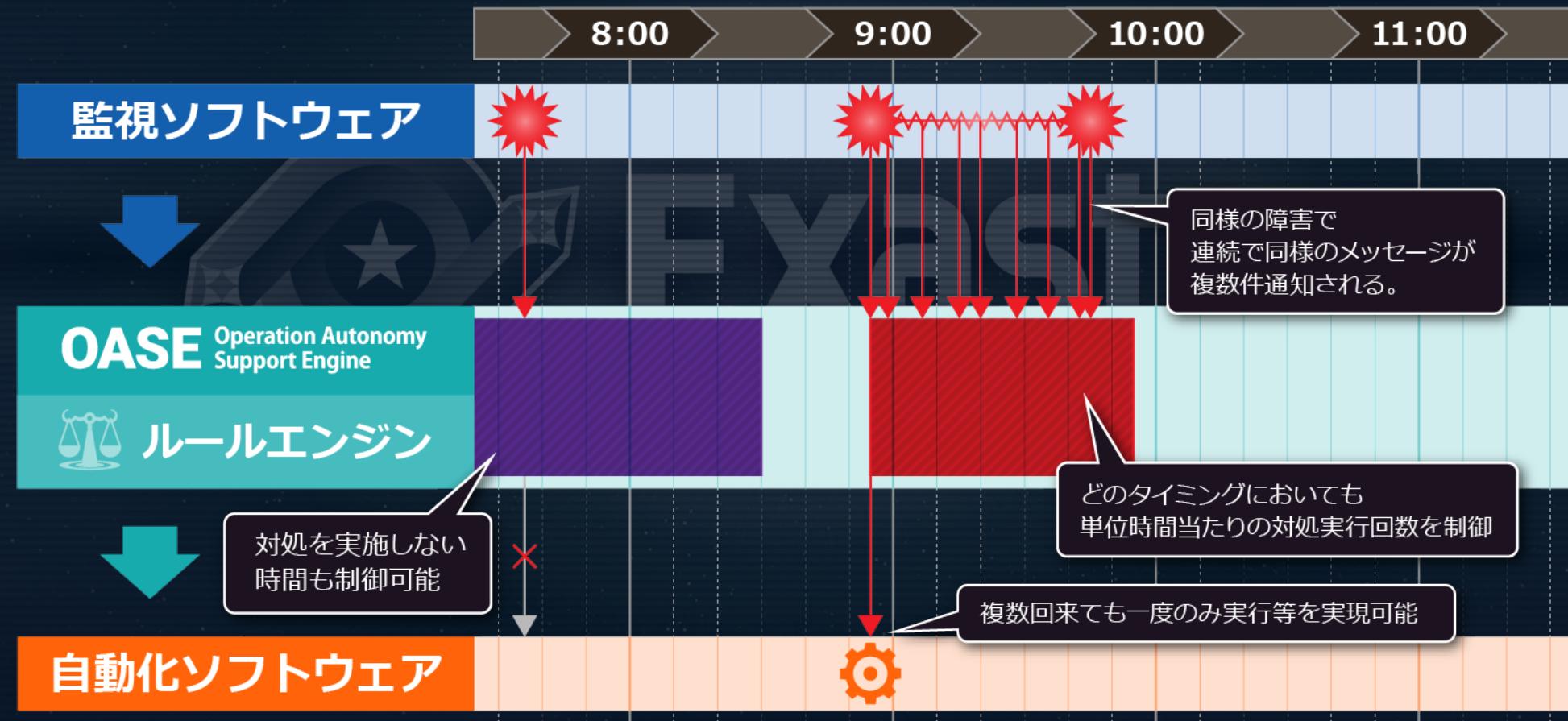
## 6つの特徴：③コーディングレスなルールを定義する

ルールフォーマットにExcelを採用しています。Web UI上からプログラムを自動で書き込んだExcelを作成できます。ユーザはコーディングレスなルールの定義ができます。



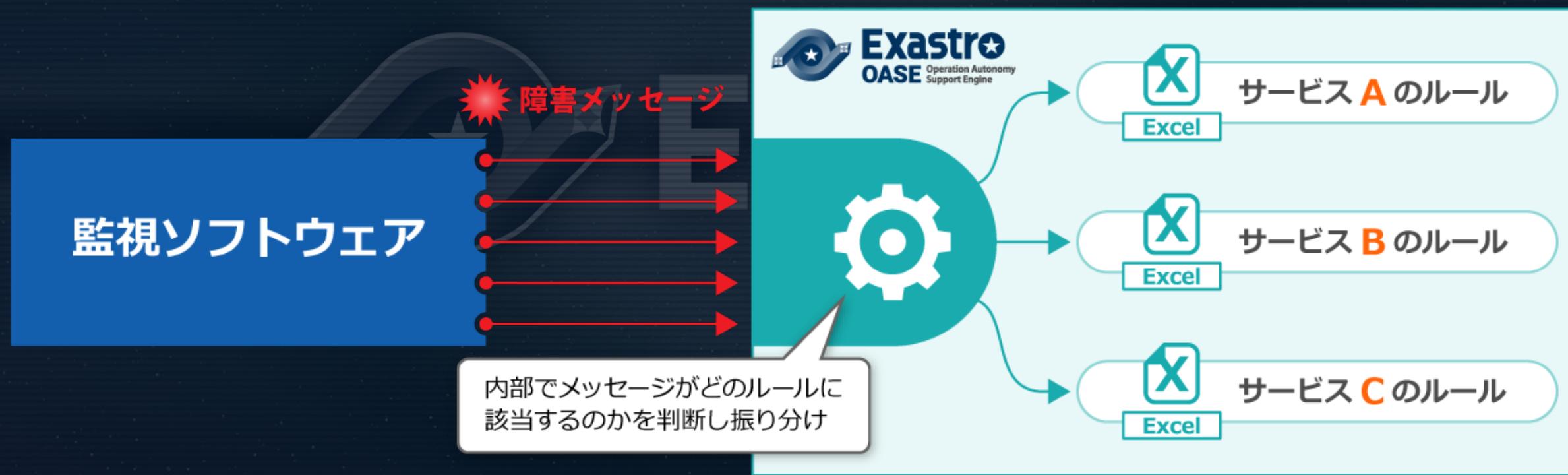
## 6つの特徴：④アクション実行回数を制御する

監視ツールから複数メッセージを受けても対処に応じて実行回数や実行間隔を設定できます。



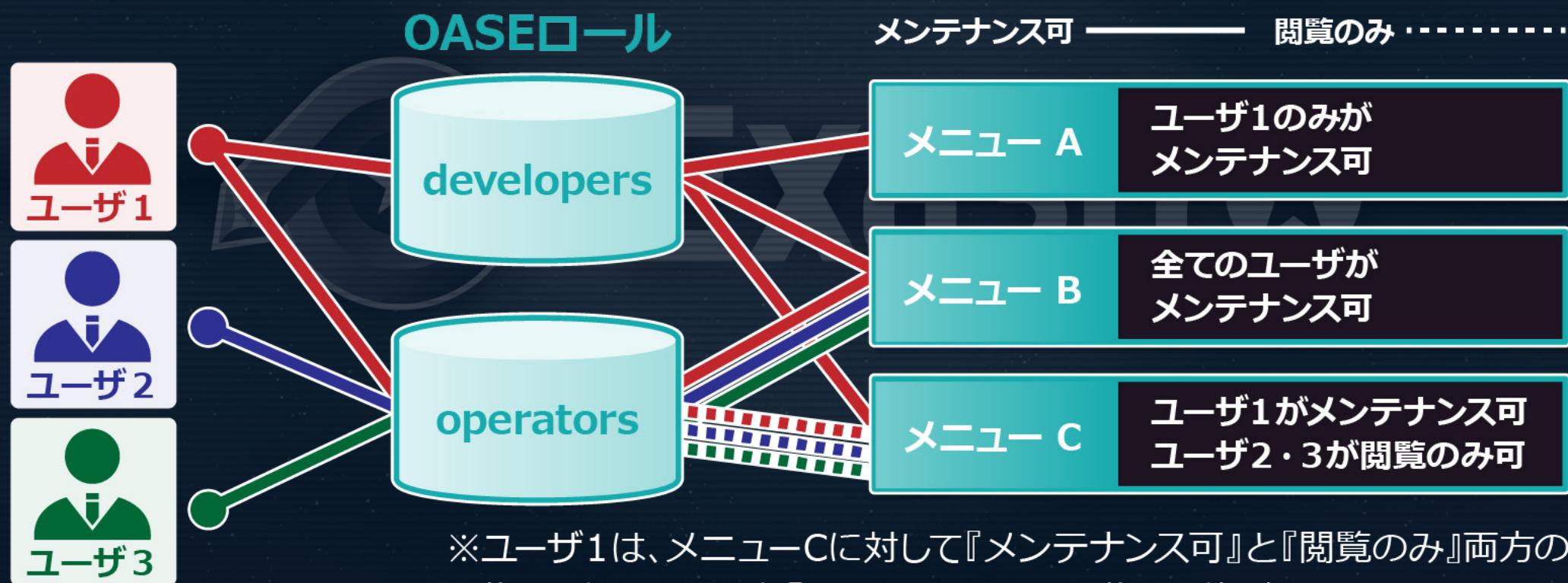
## 6つの特徴：⑤複数のルールリストを管理する

ルールをプロジェクトやシステム毎など任意の単位に切り分けて管理できます。また切り分けたルールごとにメッセージを振り分けることが可能です。



## 6つの特徴：⑥グループ毎にアクセスを制御する

ロールベースアクセス制御機能を備えているため参照・更新・実行をグループ毎に割当てて管理できます。



# AIOpsへの取り組み

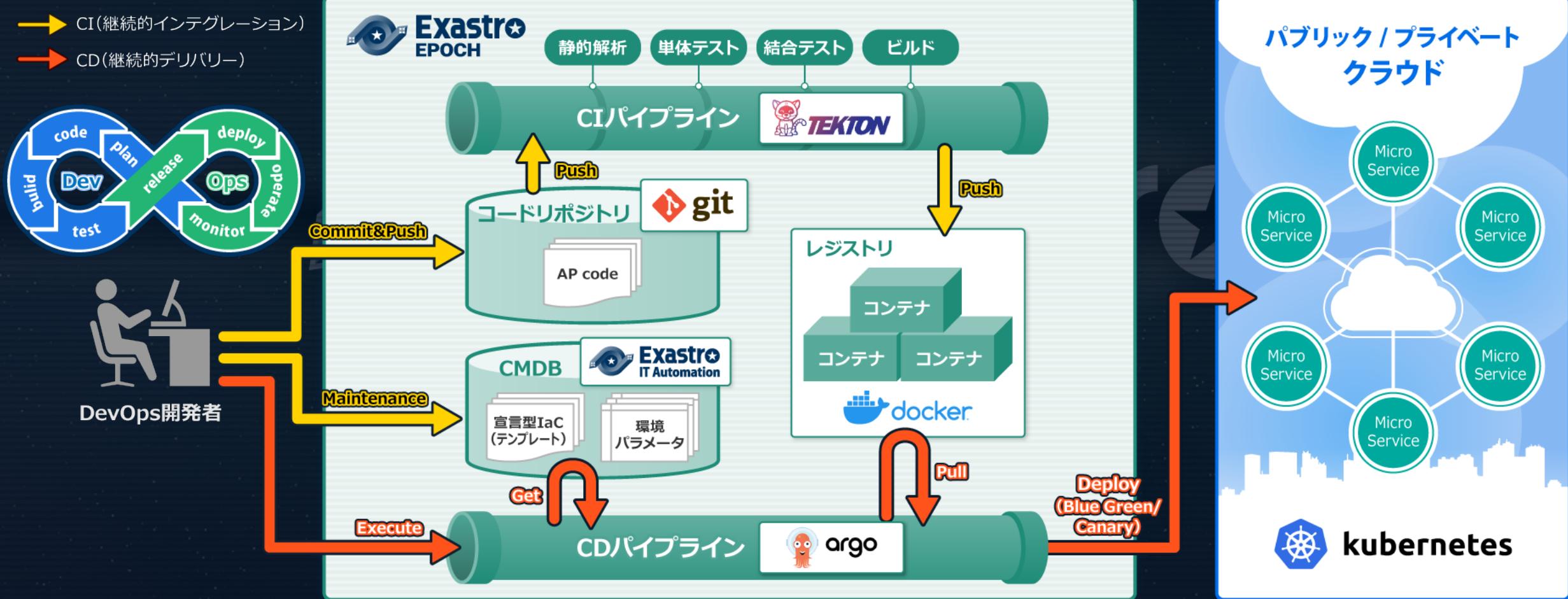
頻繁に発生する事象を早期に自動化/省力化するためにはルールベースが有効です。一方で、AIOpsは更なる自動化/省力化の効果が見込めますが、「AIが読み込むための学習データ」が必要となるため立ち上げに時間を要します。  
Exastro OASEでは、ルールベースで自動化/省力化に結果を出しながら、対処結果を蓄積してAIOpsの実現を支援します。



# Exastro EPOCH

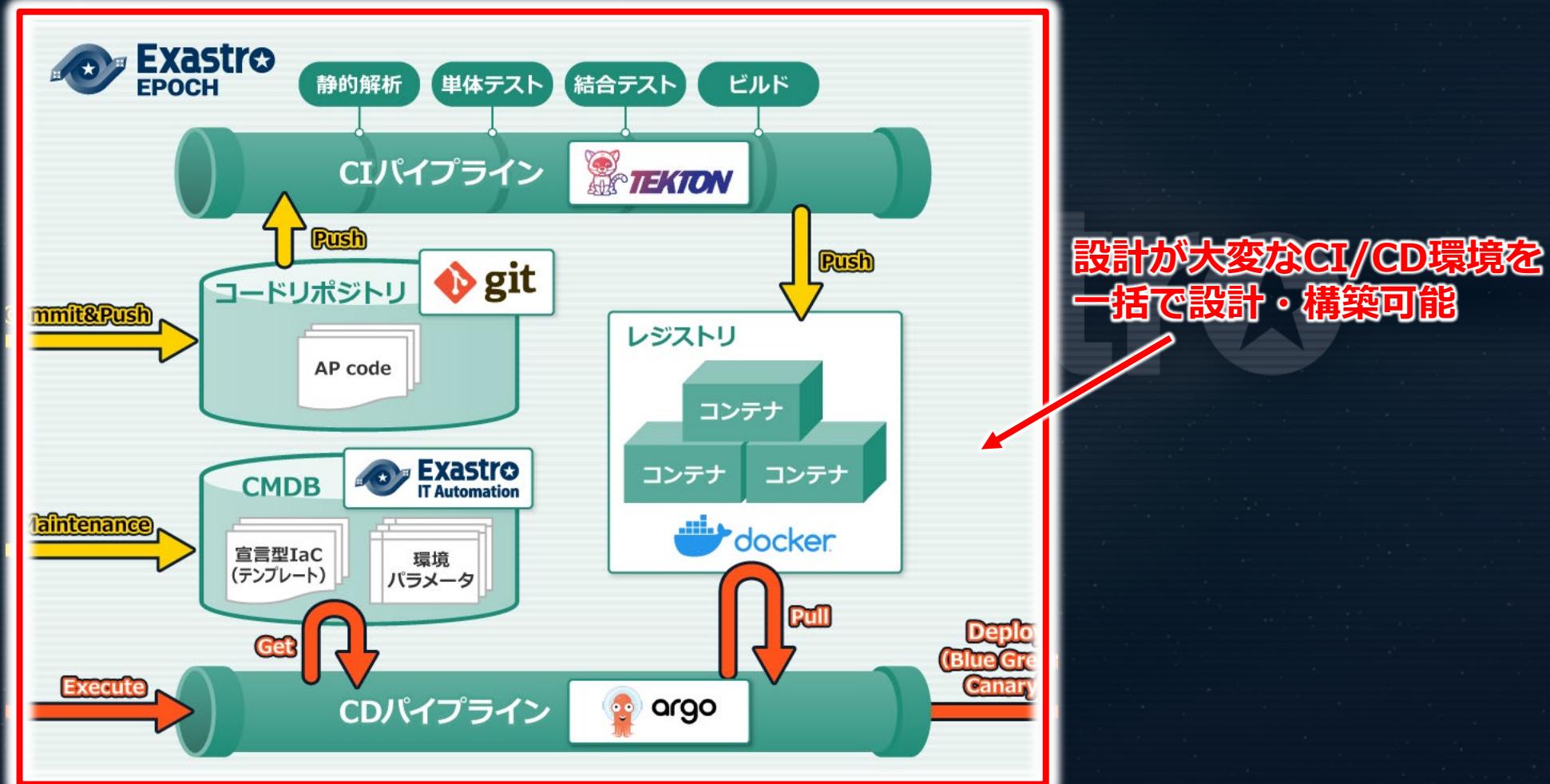


## Exastro EPOCHは 「クラウドネイティブシステム開発を加速するためのフレームワーク」です



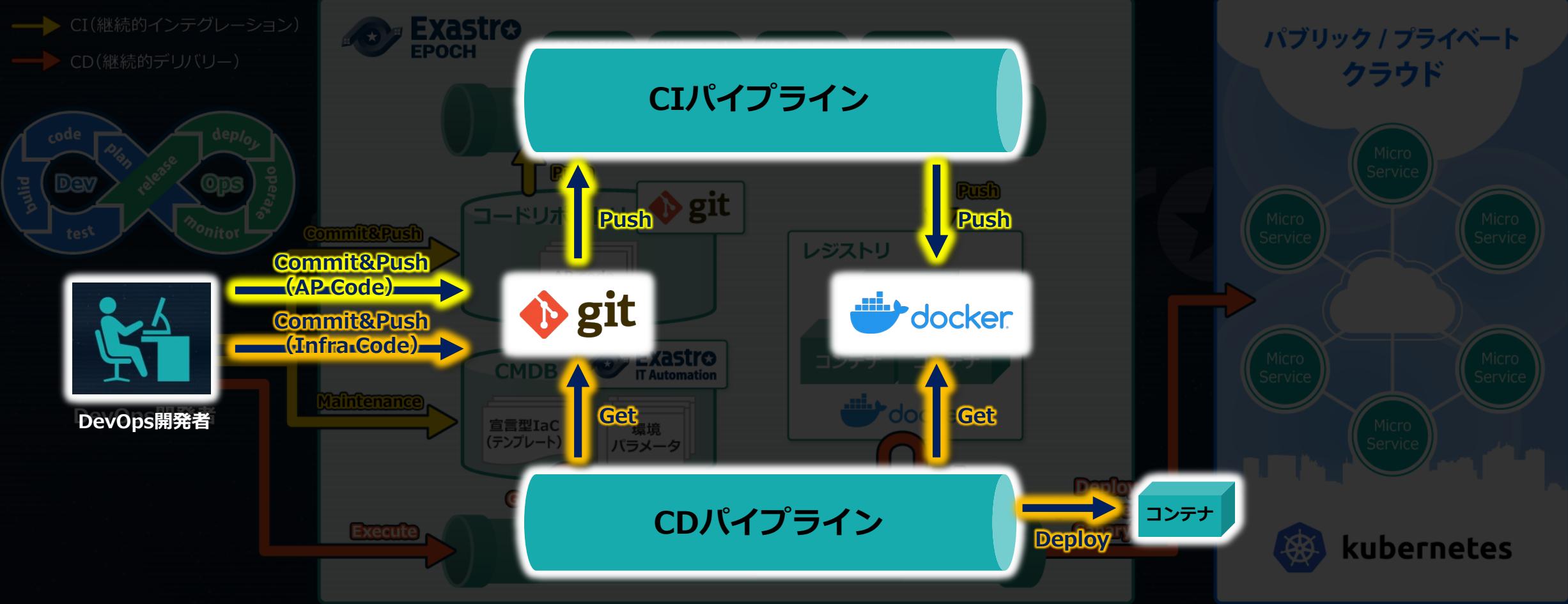


## 高速開発に必要なCI/CD環境を一元的に提供する



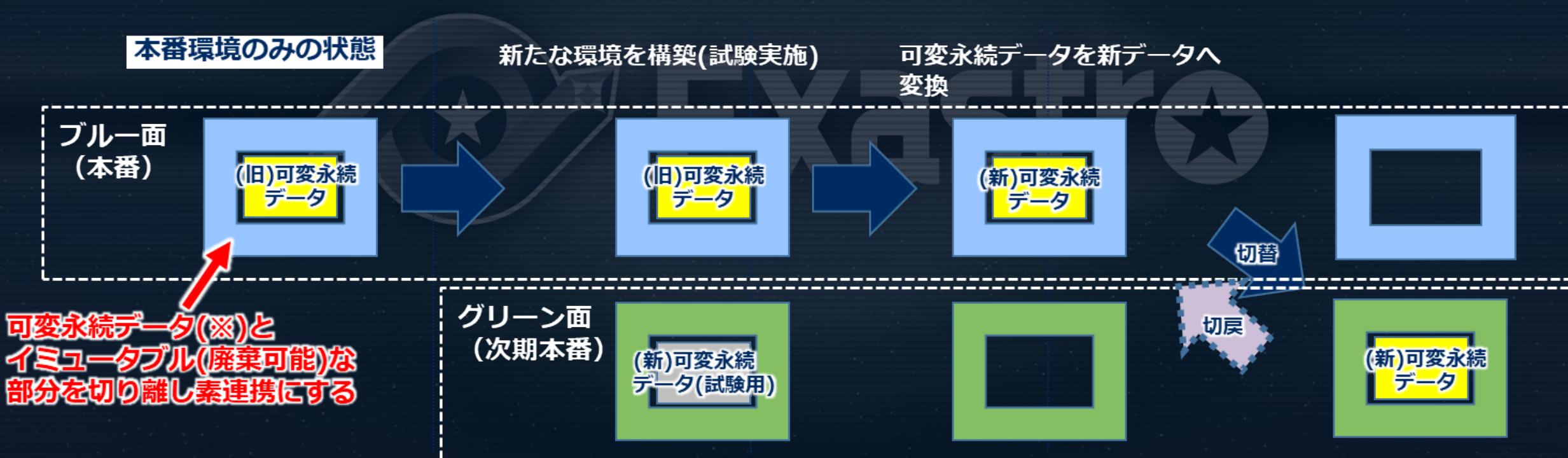
## 2つめの特徴 – GitOpsに対応した環境の展開

アプリケーションコードのコミット＆プッシュだけで、CI/CDパイプラインを実行可能 (GitOps)



### 3つめの特徴 - カナリアリリース/ブルーグリーンデプロイに対応

コンテナデプロイでは迅速な切り替え・切り戻しに有用な“カナリアリリース”と“ブルーグリーンデプロイ”が利用可能  
(例: ブルーグリーンデプロイ)



## 4つめの特徴 - ワークロードを統一

CI/CDのステータスを各CI/CDツールを介さずに一括で確認可能  
スムーズな状況確認・DevOpsを実施



## 5つめの特徴 – 環境一致を強力に支援する

宣言的IaCの環境依存パラメータをシステム上で管理  
CD時に自動的にパラメータ反映したIaCをコミットする

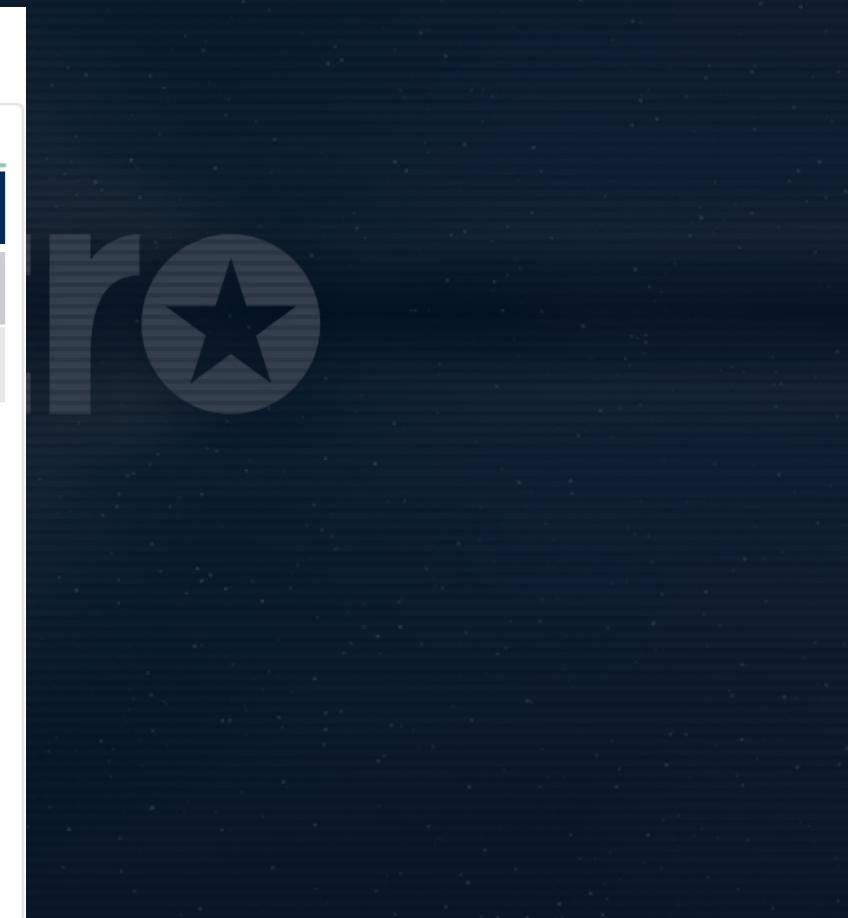
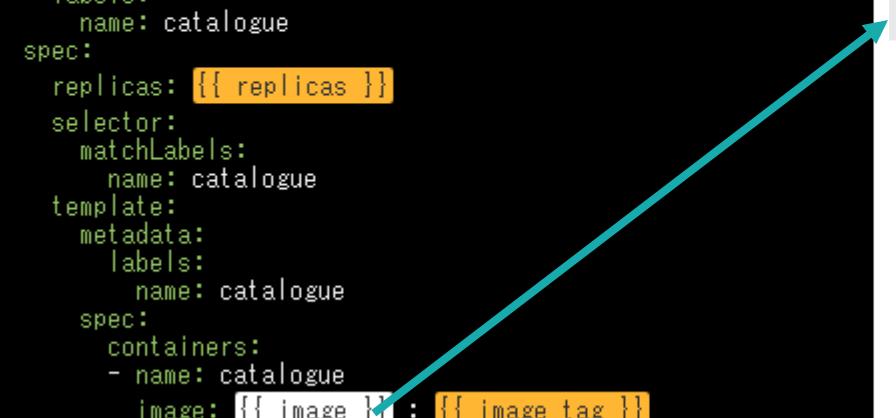
パラメータ入力

text.yaml

```
# epoch-template => file001
apiVersion: apps/v1
kind: Deployment
metadata:
  name: catalogue
  labels:
    name: catalogue
spec:
  replicas: {{ replicas }}
  selector:
    matchLabels:
      name: catalogue
  template:
    metadata:
      labels:
        name: catalogue
    spec:
      containers:
        - name: catalogue
          image: {{ image }} : {{ image_tag }}
          ports:
            - containerPort: 8000
      nodeSelector:
        beta.kubernetes.io/os: linux
```

環境

環境	Develop
環境①	パラメータA
環境②	パラメータB



# 事例



# 【事例①】大規模システムの様々な運用監視設定を効率化しました

## 課題



システムのパラメータが時系列で構成管理できておりらず、またPlaybookの部品化ができておらず、構築および運用における作業の自動化/効率化が実現できていなかった。

## Exastro IT Automationの機能で

- システムのパラメータを構成管理
  - Ansible Playbookを部品管理
- することで設計～構築、および運用のフェーズを繋げて、作業の自動化を実現した。

## 解決策



## 効果

### 例1: 運用監視設定作業

構成管理  
自動化前

年間4億円の作業工数で実施していた

4億円/年

構成管理  
自動化後

1億円/年

75% 削減

年間1億円の作業工数で  
実施できるようになった(75%削減)

### 例2: 運用コマンドリストメンテナンス作業

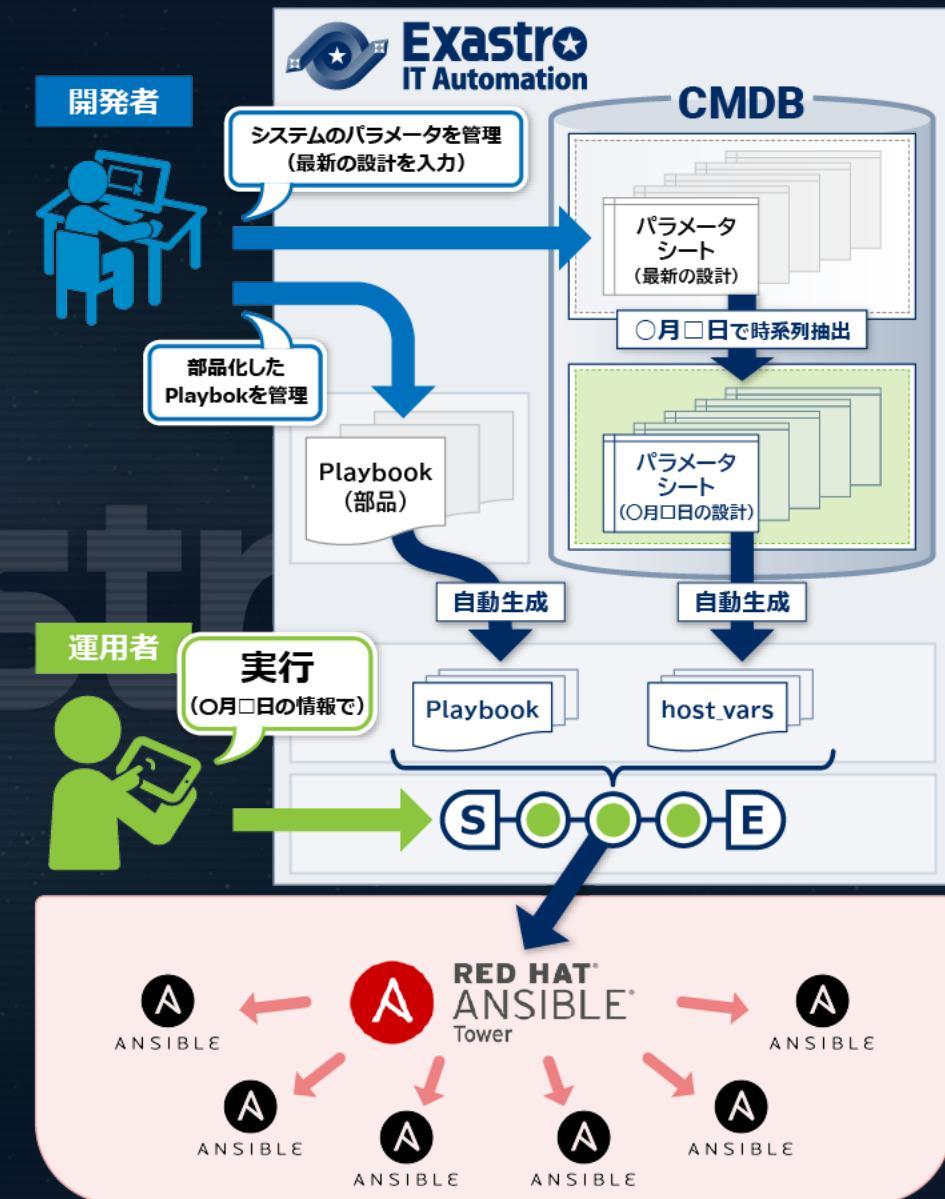
年間4,000万円の作業工数で実施していた

4,000万円/年

800万円/年

80% 削減

年間800万円の作業工数で  
実施できるようになった(80%削減)  
メンテナンスまでのリードタイムが不要となり  
即時反映がされることで利便性も向上した。



## 【事例②】大規模イベント向けに3万台の機器を自動キッティングしました

### 課題



大量のネットワーク機器の初期設定を行う必要があった。しかし、手作業では多くの工数が必要であり、また大量の設定値を表計算ソフトで管理するのにも限界があった。

### 解決策

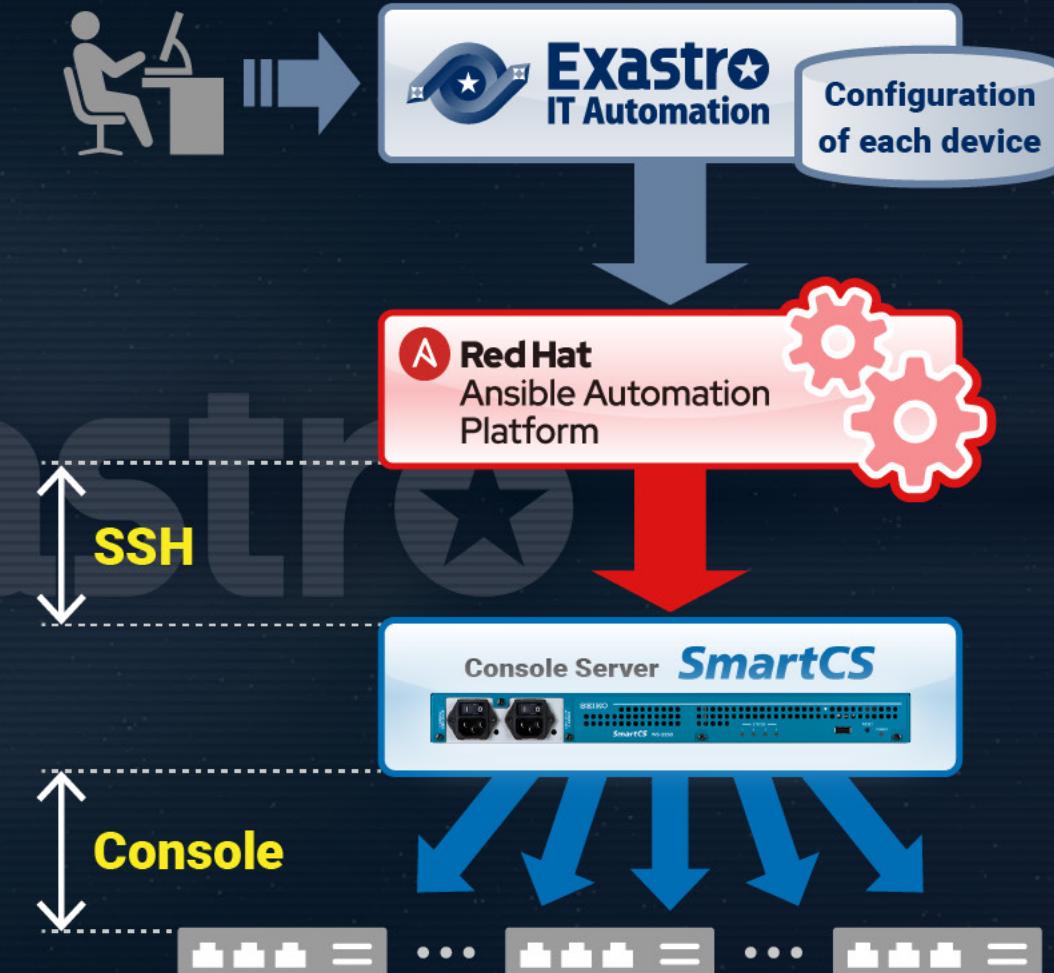


SmartCSを活用して複数のネットワーク機器を同時に設定可能にした。  
また、設定値は**CMDBで一元管理**し、**Ansibleで設定を自動投入**することで課題を解決できた。

### 効果



スイッチなど6種類のネットワーク機器の初期設定を自動化することができ、**2,000台/月のリリース**を達成した。



# 【事例③】システムの構成管理を活用して機器停止のサービス影響予測しました

## 課題



大規模キャリアシステムにおいて、機器の計画停止、および突発的な停止によるサービス影響の調査に多くの工数がかかっていた。

## 解決策

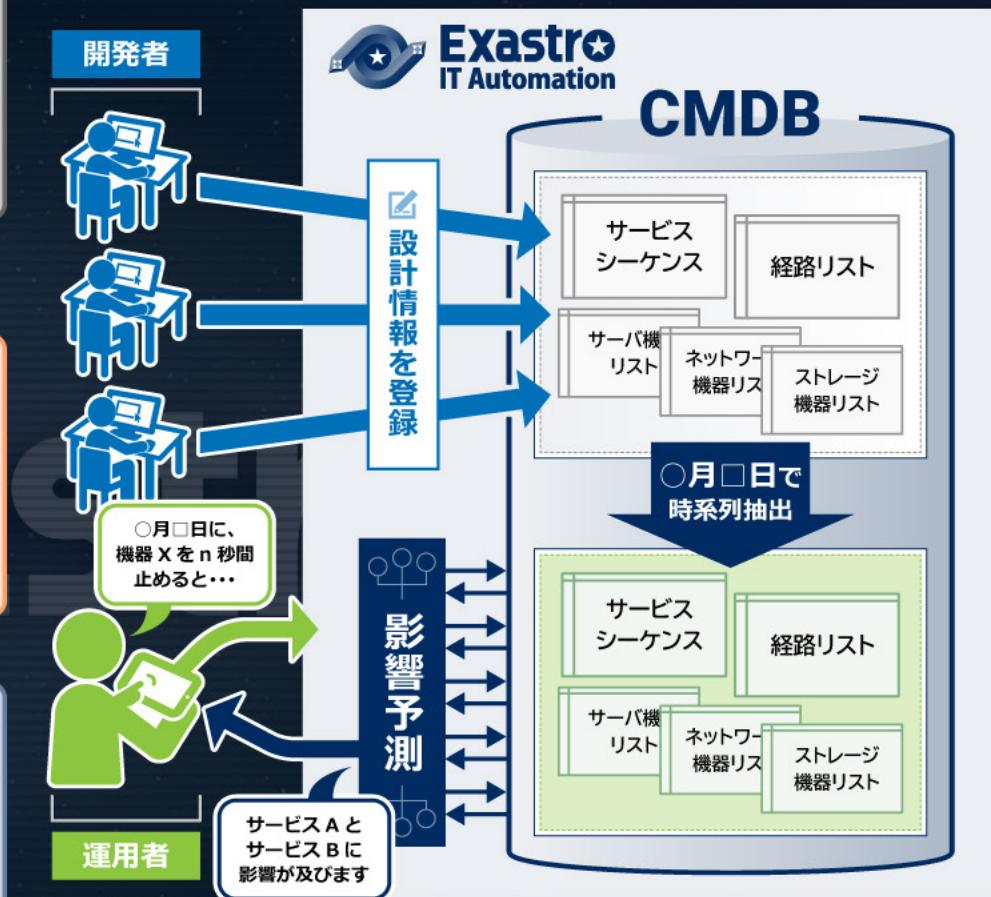


Exastro IT Automationの機能で**システムを構成管理**することにより、機器停止による**サービス影響を自動予測**することを可能とした。

## 効果



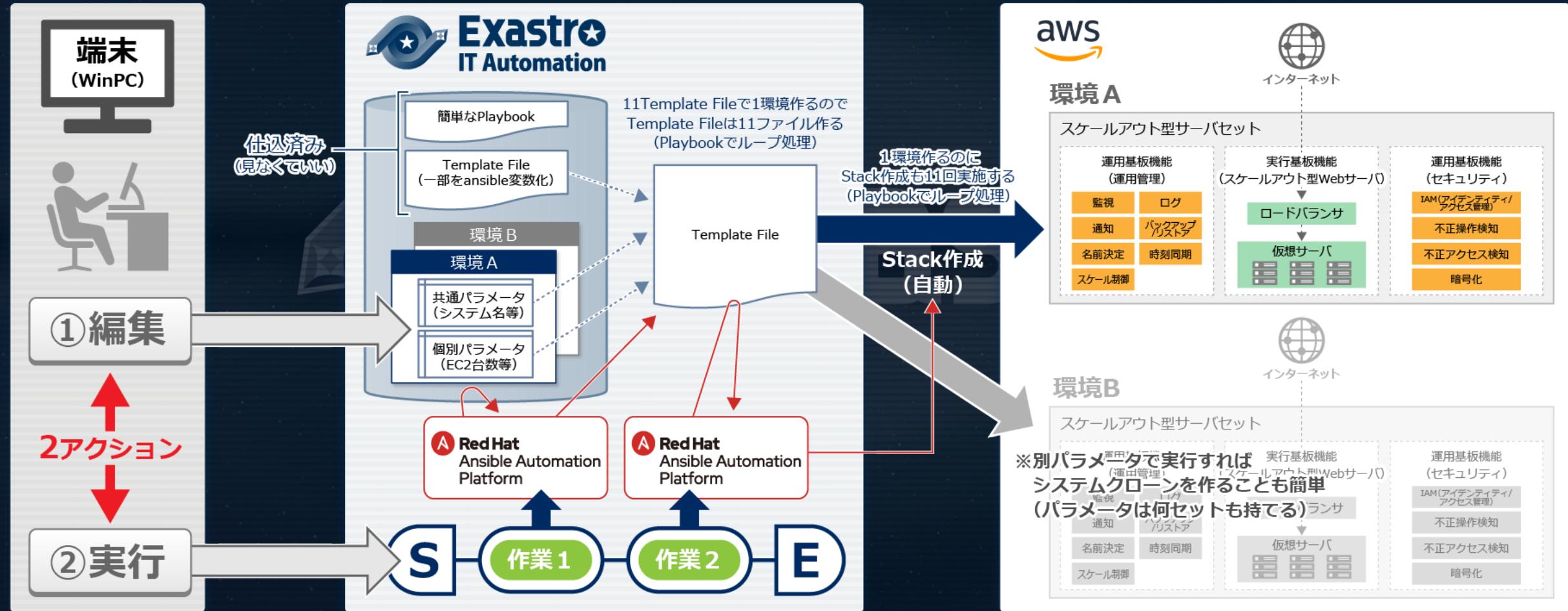
1回の調査で約80万円の工数/費用を要していたところ、本施策により工数不要となった。年間約120回の調査があったので**約9,400万円/年の削減効果**が確認された。



※影響予測はITAが提供する画面フレームワークを使ってカスタム開発。

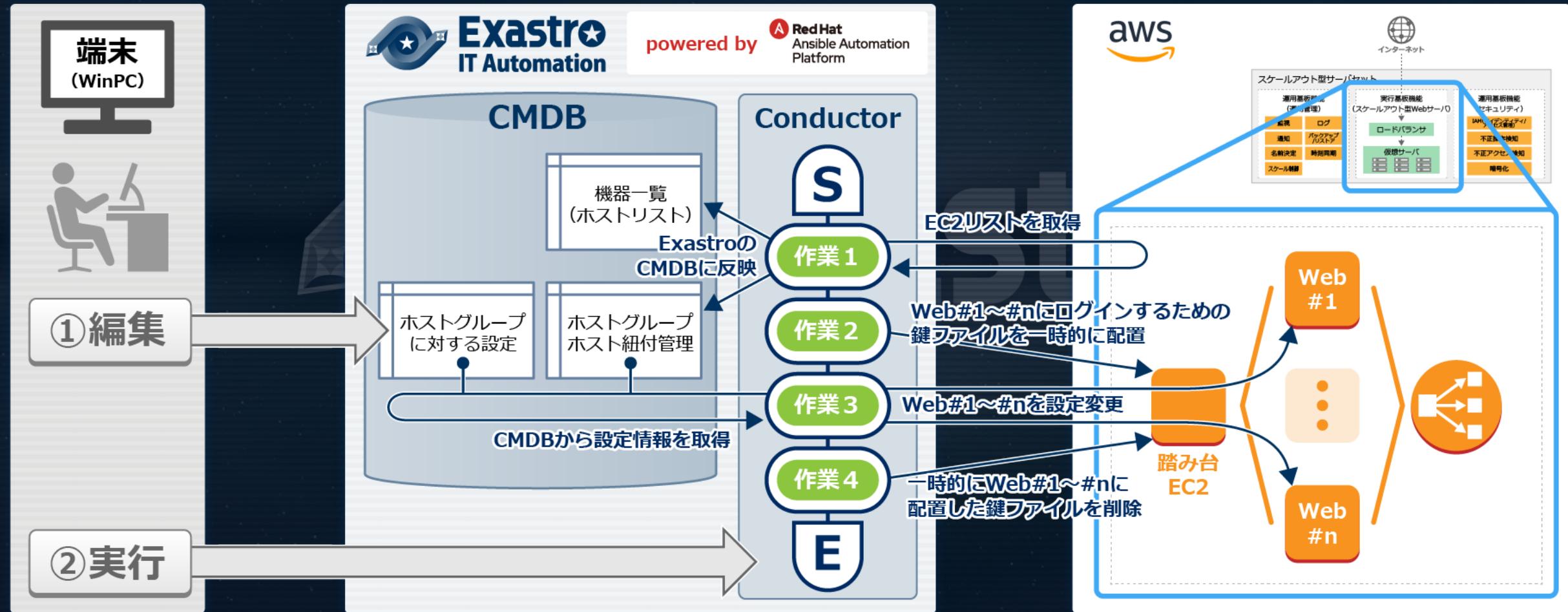
## 【事例④】企業の運用部門が各部門のAWSシステムを統制する仕組み (1/2)

Exastro IT Automationで「CloudFormationテンプレート」を管理し、各部門にガバナンスの効いたAWS環境を払い出す仕組みを提供しました。



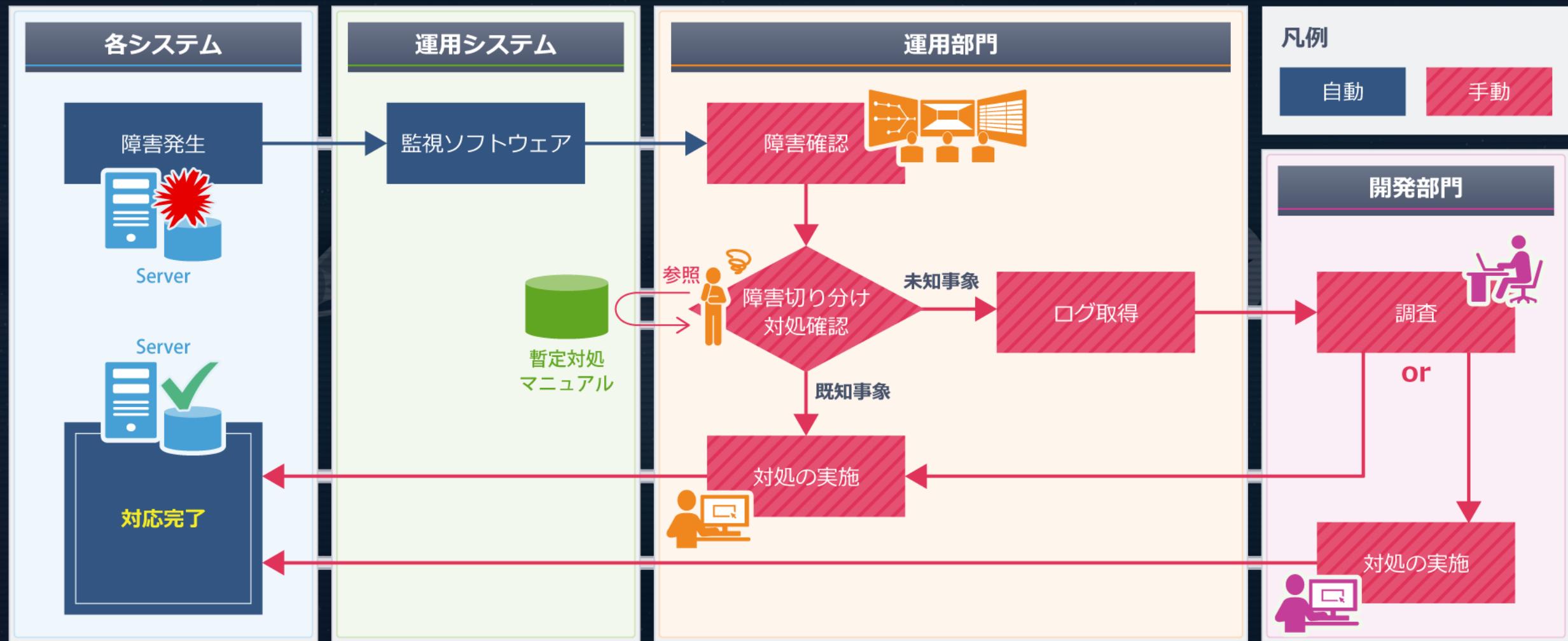
## 【事例④】企業の運用部門が各部門のAWSシステムを統制する仕組み (2/2)

稼働中のEC2(オートスケール)に緊急でパッチ適用する、  
といった運用シナリオにも対応しました



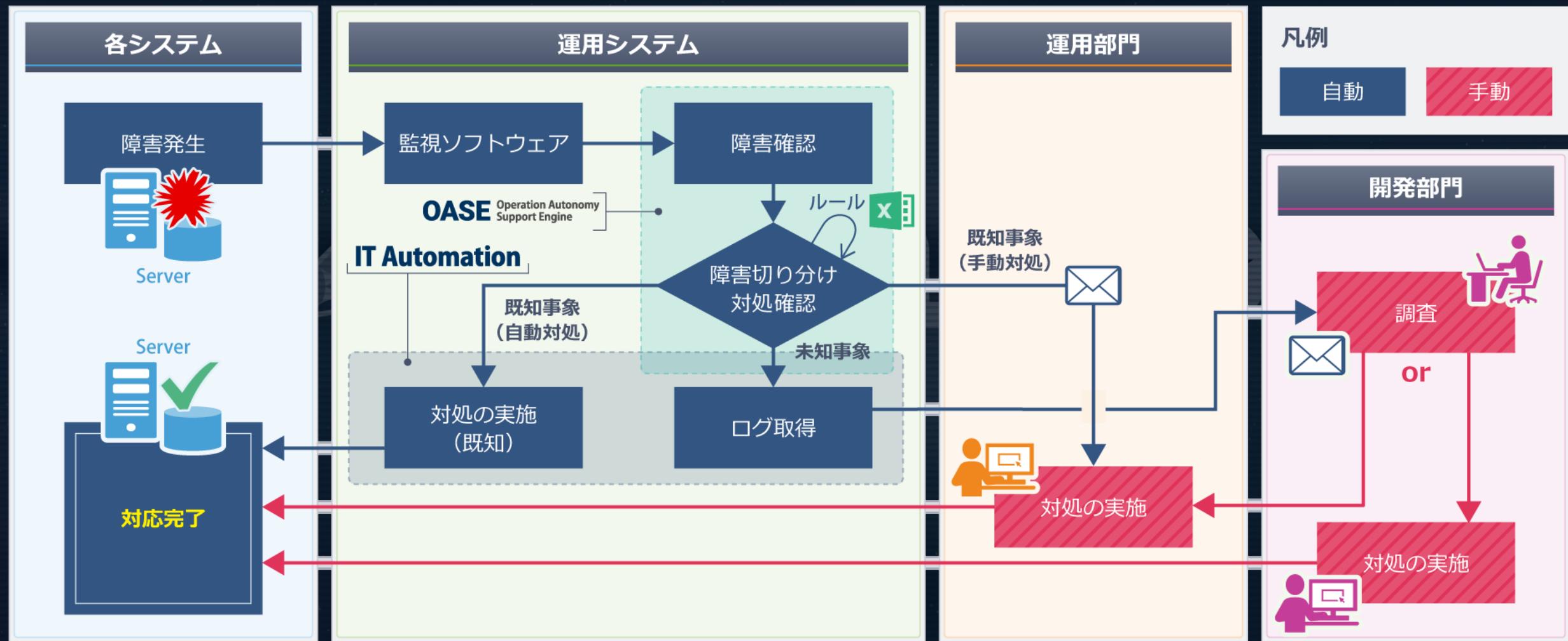
## 【事例⑤】運用業務における障害対処を自動化しました – Before –

障害の切り分けから対処まで、手動で作業を実施していました。



## 【事例⑤】運用業務における障害対処を自動化しました – After –

Exastroをご利用いただくことで、手動作業を最小限に抑えることができました。



# 参考① ExastroをOSS公開する目的

## Exastro Suiteは、市場で広く共有して他のソフトウェアとも連携してITエンジニアの「苦」を解消することを目的にOSSにしました。

NECは、システム構築の自動化を支援するソフトウェア「Exastro IT Automation (エグザストロ・アイティ・オートメーション)」を開発し、2019年4月にオープンソースソフトウェア(OSS)として公開しました。これまでの大規模システム構築の課題として、システムを構成する機器の設定など手作業による定型作業の繰り返しや、作業者の熟練度の違いによる効率や品質のばらつき等が挙げられます。Exastro IT Automationは、Infrastructure as Codeの技術を用いてこれらの課題を抱えるシステム構築の自動化を実現するソフトウェアであり、効率と品質の向上に貢献します。

NECはこれまでミッションクリティカルな大規模システム構築や開発を行い、SIのノウハウや実績を積み上げてきました。Exastro IT Automationはこれらの経験から出てきた課題を元に制作したソフトウェアであり、多岐にわたるシステム構築プロジェクトでExastro IT Automationを活用して自動化を進めてきました。しかし、昨今の大規模システム構築では、複数企業で行うことも多く、本ソフトウェアのような自動化ツールのニーズが高まっています。

この度Exastro IT AutomationをOSS化することで、市場で広く本ソフトウェアを共有して他のソフトウェアとも連携してエコシステムを形成することにより、新たなビジネスモデルを創出して社会価値創造を実践してまいります。

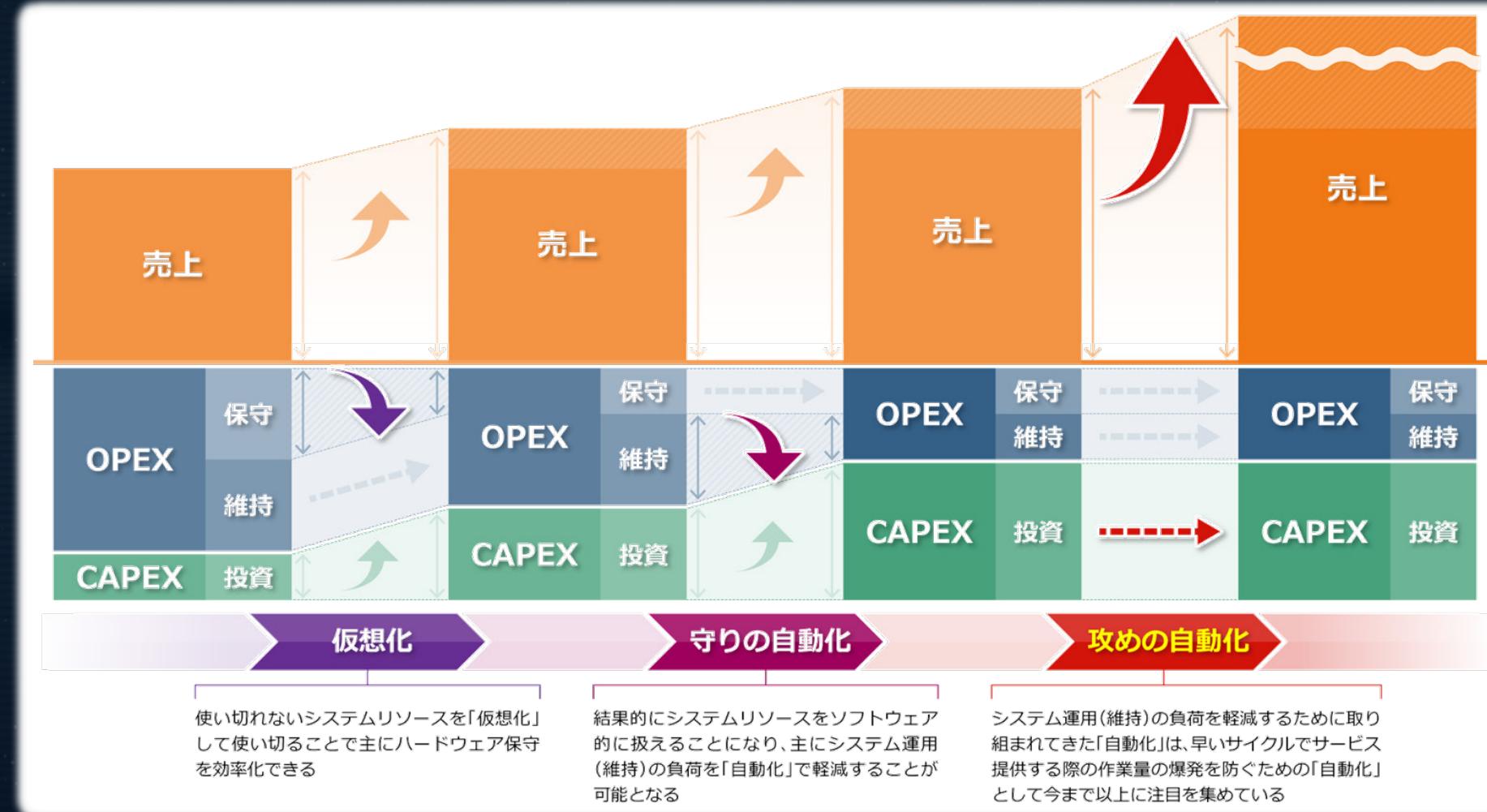
出典：<http://www.news2u.net/releases/165517/>

補足：旧称「astroll IT Automation」は「Exastro IT Automation」に書き直している



## 参考② 攻めと守りの自動化

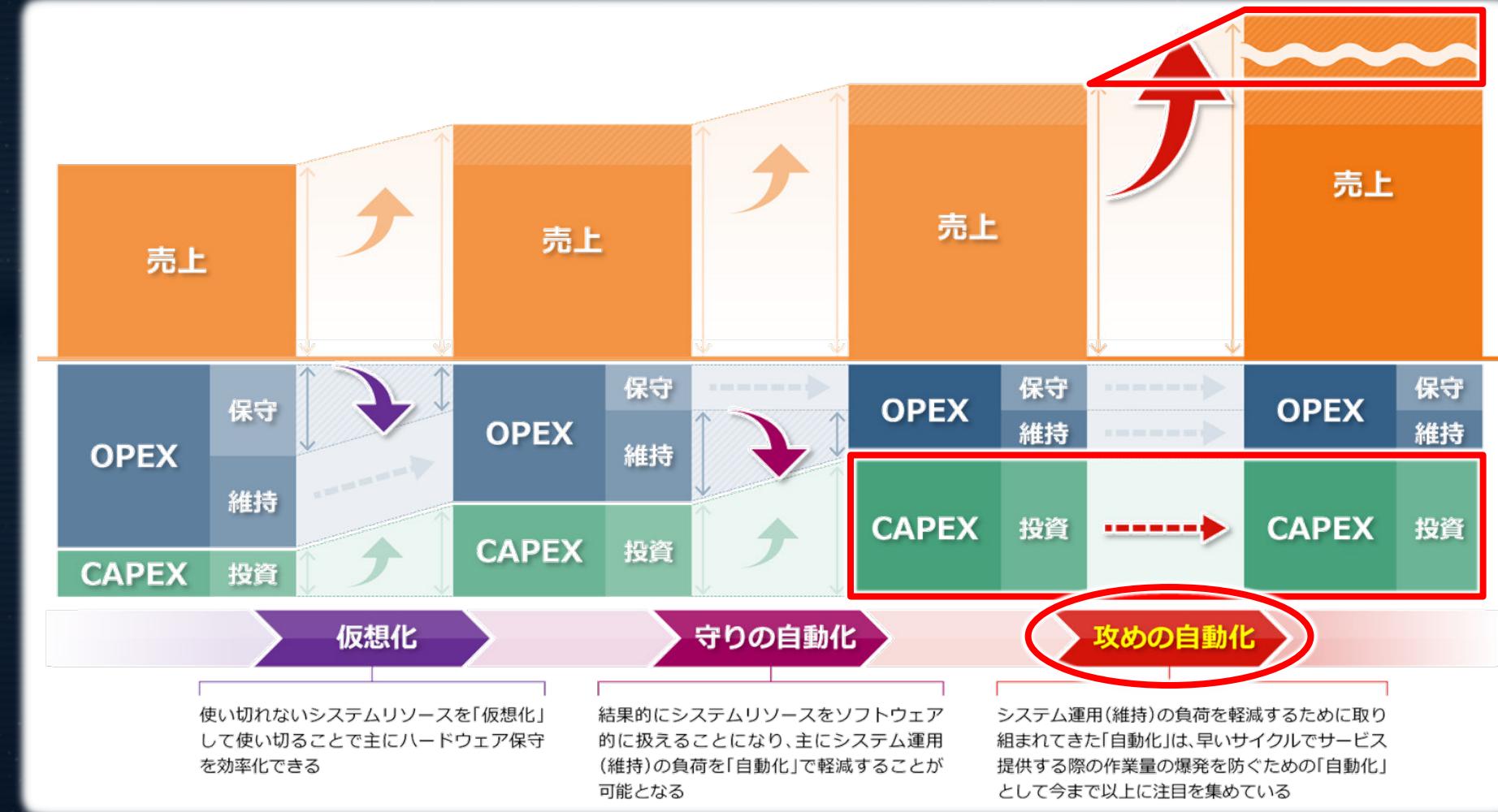
## 2つの領域の自動化は財務的な視点でも目的を異にします

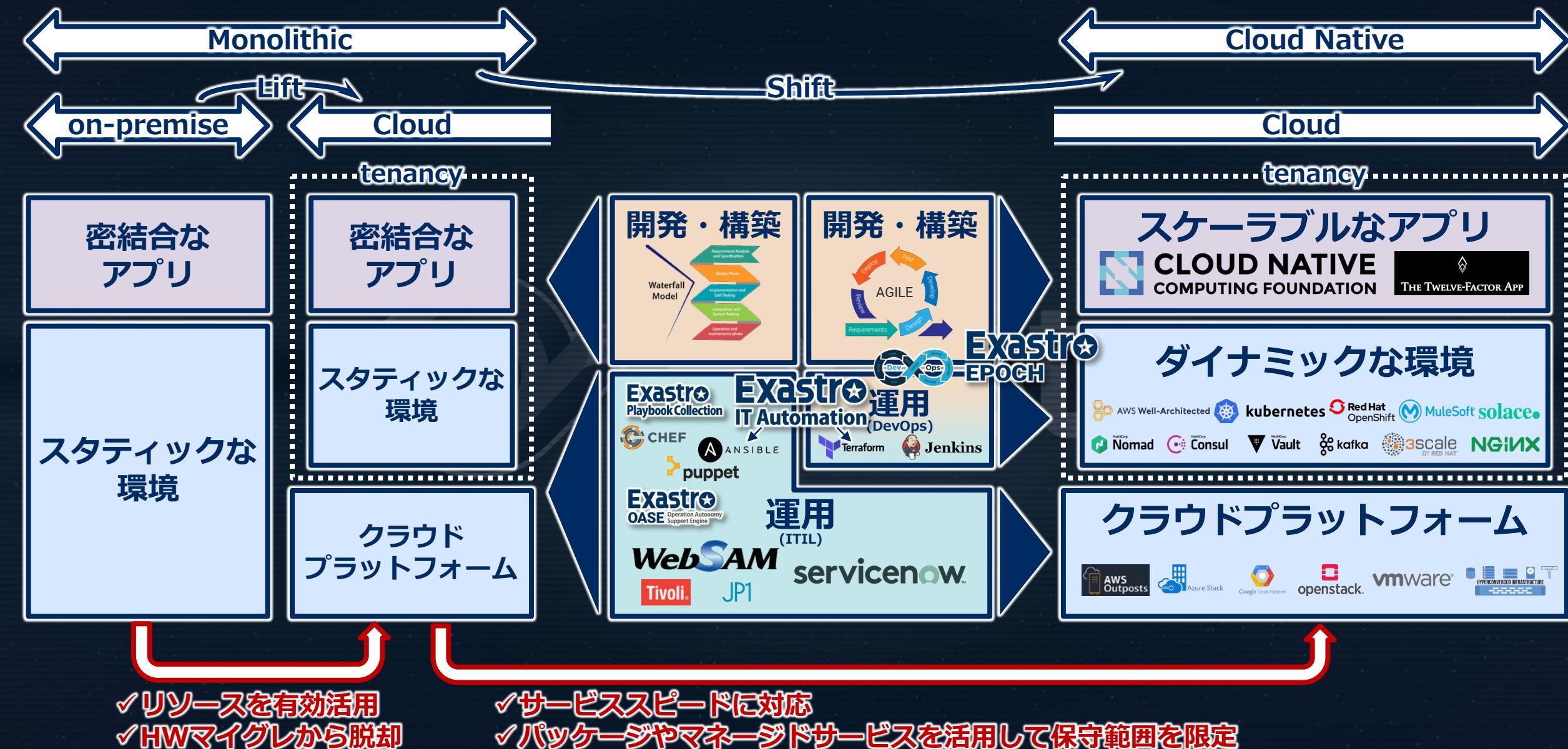


## 守りの自動化はOPEX(維持費)の効率化を目的とすることが多いです

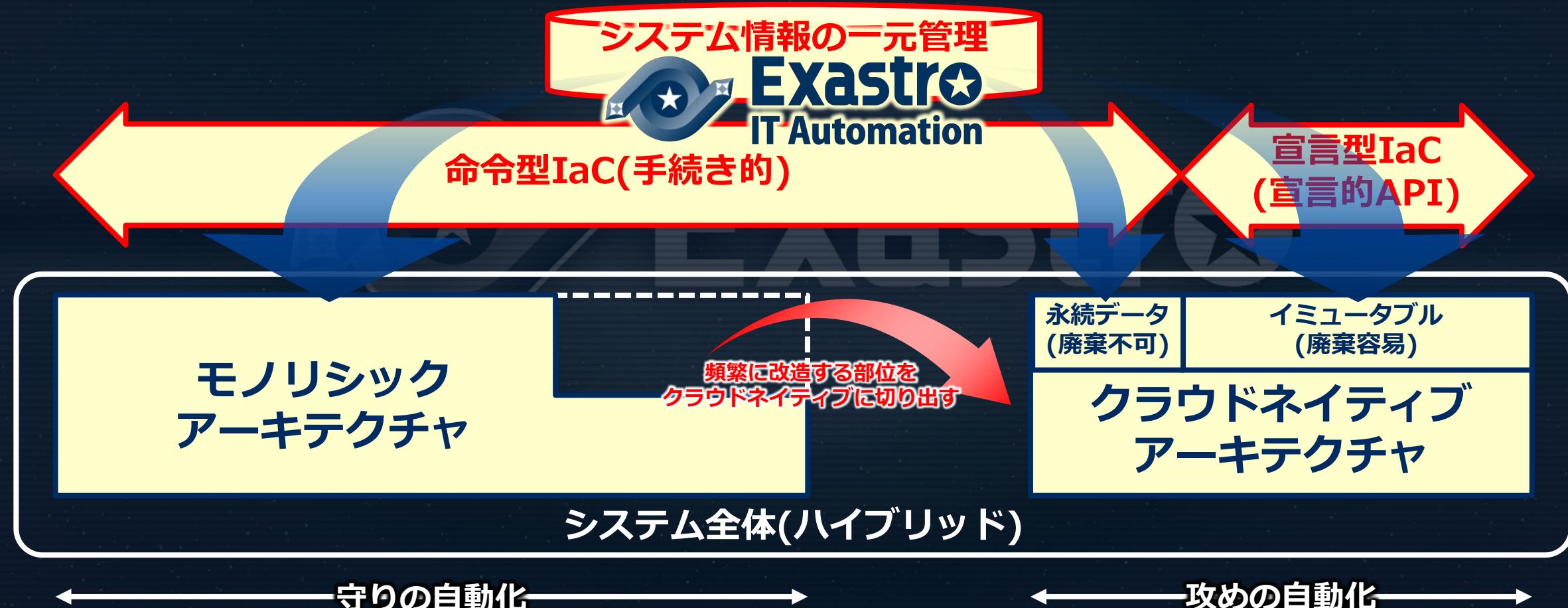


## 攻めの自動化はROI(投資利益率)の引き上げを目的とすることが多いです





## Exastro ITA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



## 参考③ 攻めの自動化



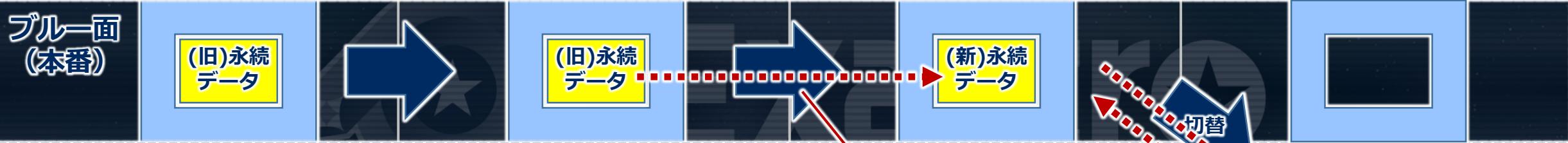
## ブルーグリーンデプロイメントでは部分的に「新規構築」を繰り返します 手作業でやっていたのでは作業量が爆発しますので自動構築は必須です

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)



ブルー面  
ミラー  
(切戻担保)

※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

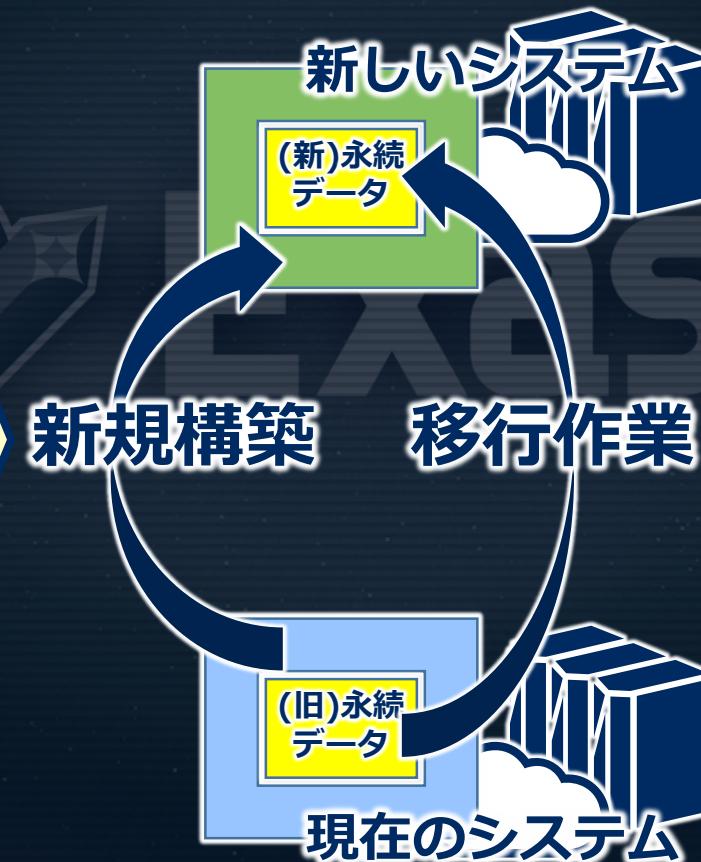
「新規構築」以外の移行作業もある  
(自動化しなければならない)

「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「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○○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

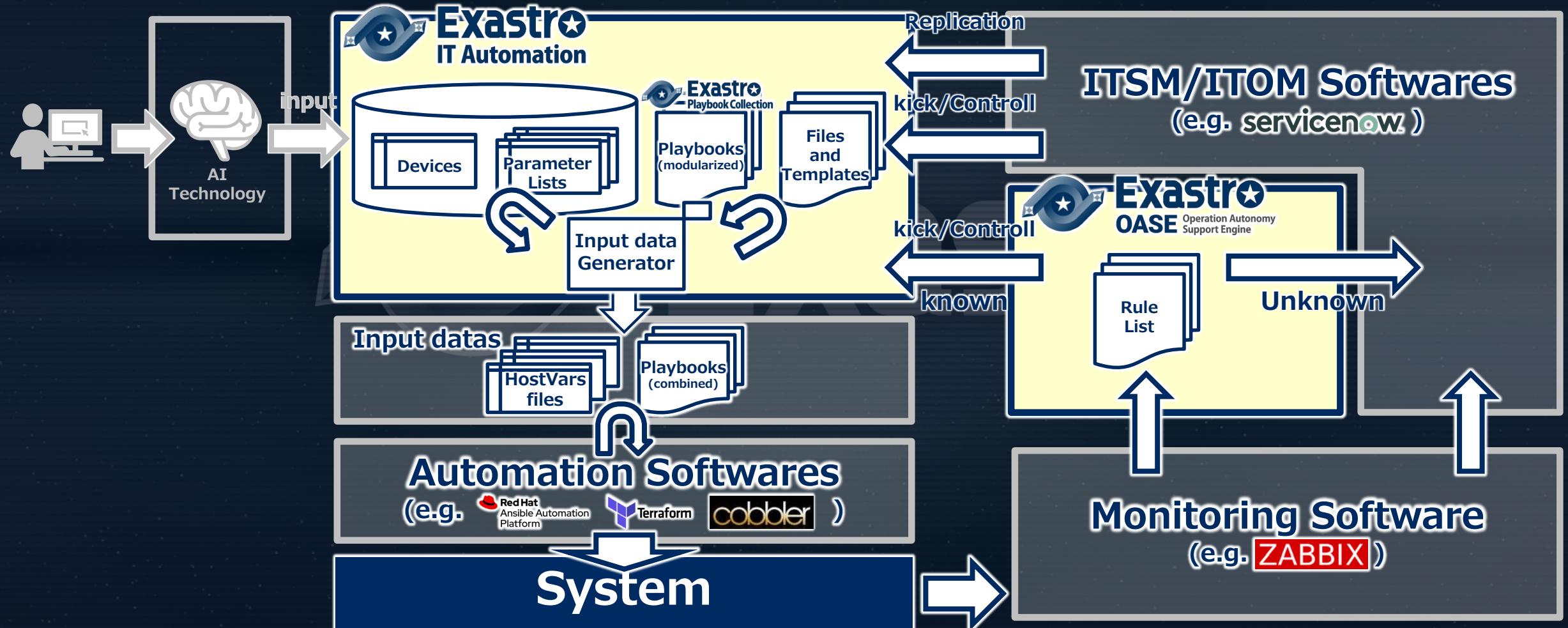
## 参考④ 守りの自動化

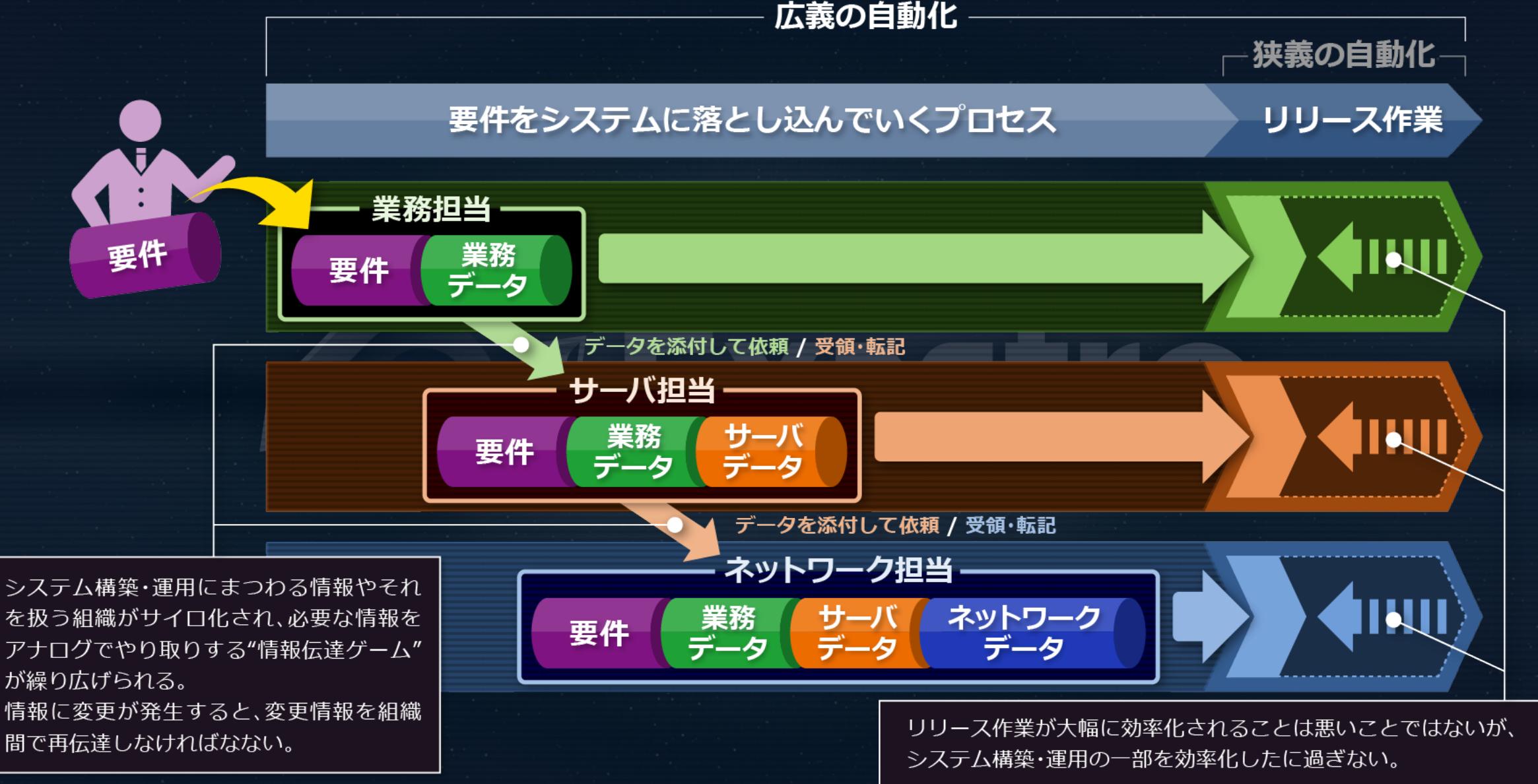


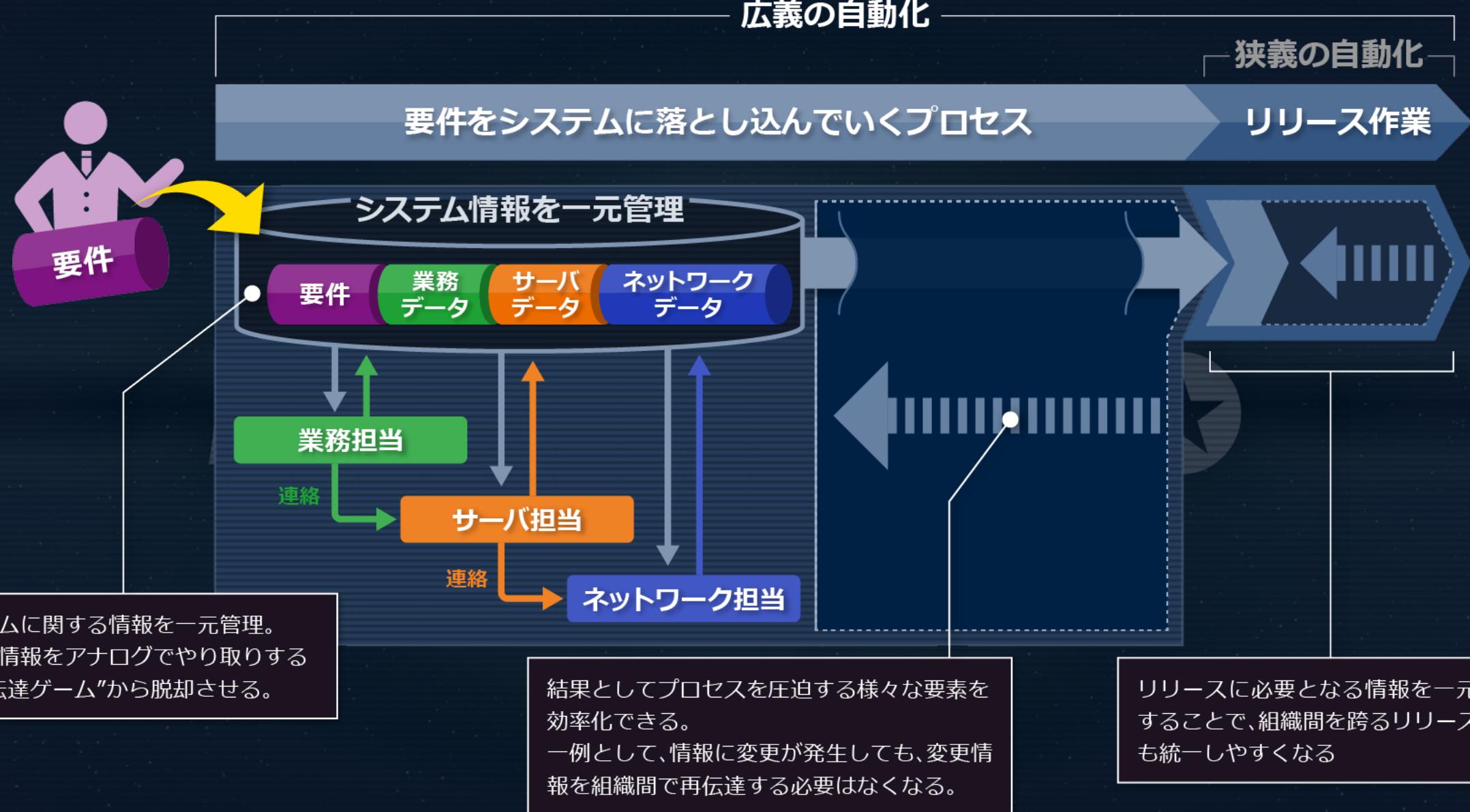
Design

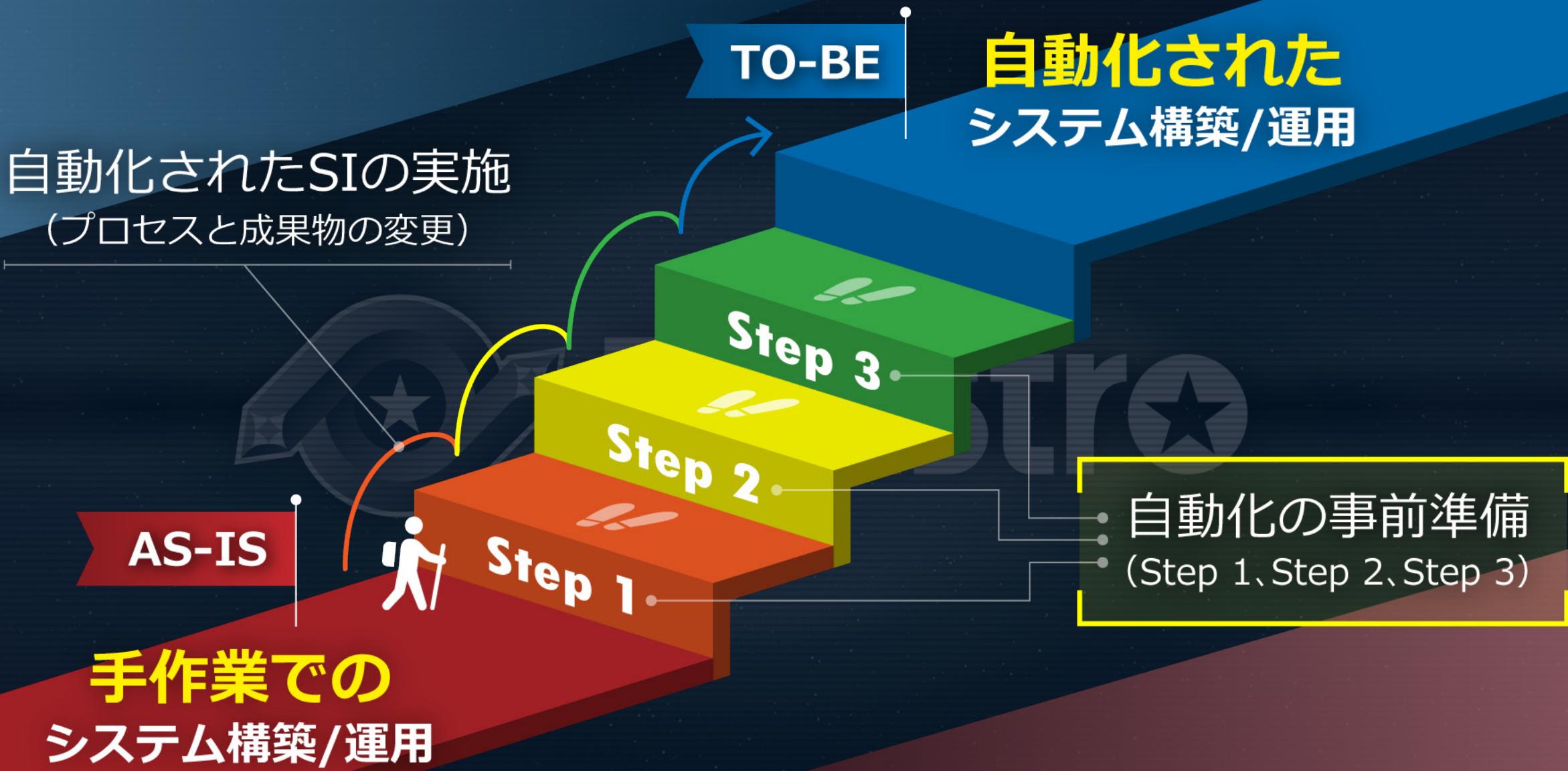
Develop/Construct

Operate





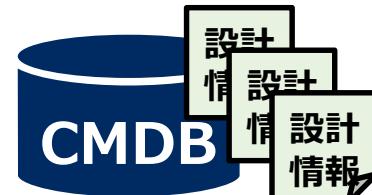




## これらの課題は大まかには3つのステップで解決できます



### Step 1 設計情報の一元管理

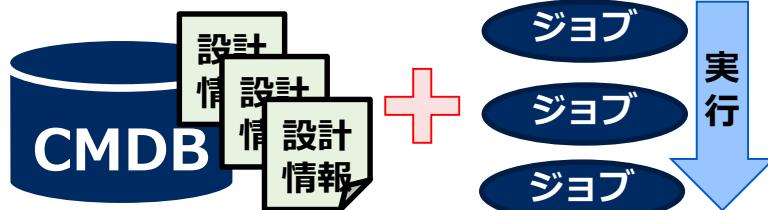


### Step 2 自動実行の実現



**連携**

### Step 3 一元管理と自動実行の連携



## 細かく刻めば12つのタスクで解決できます

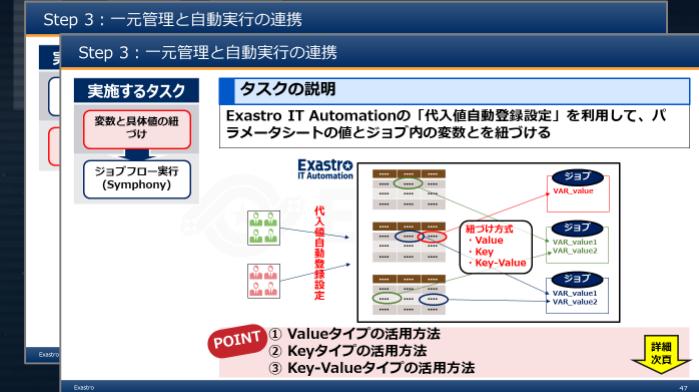
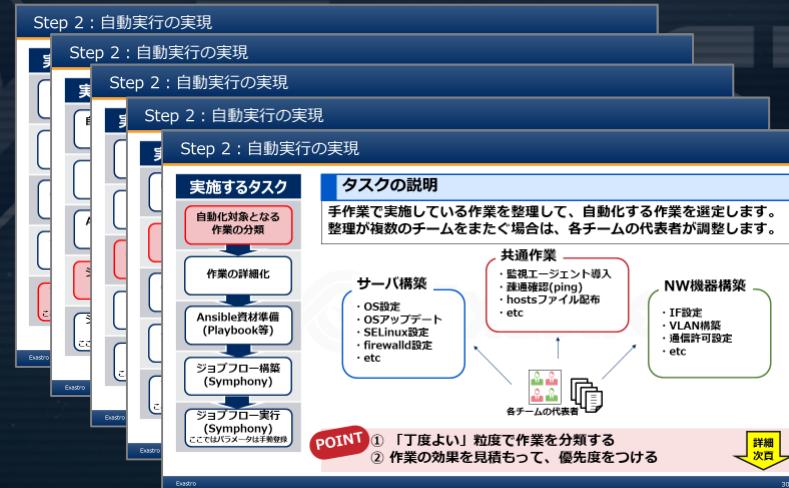
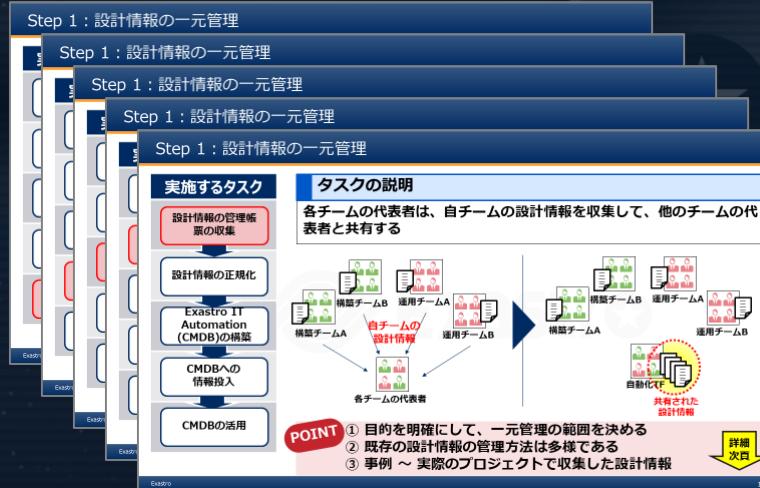
Step 1



Step 2



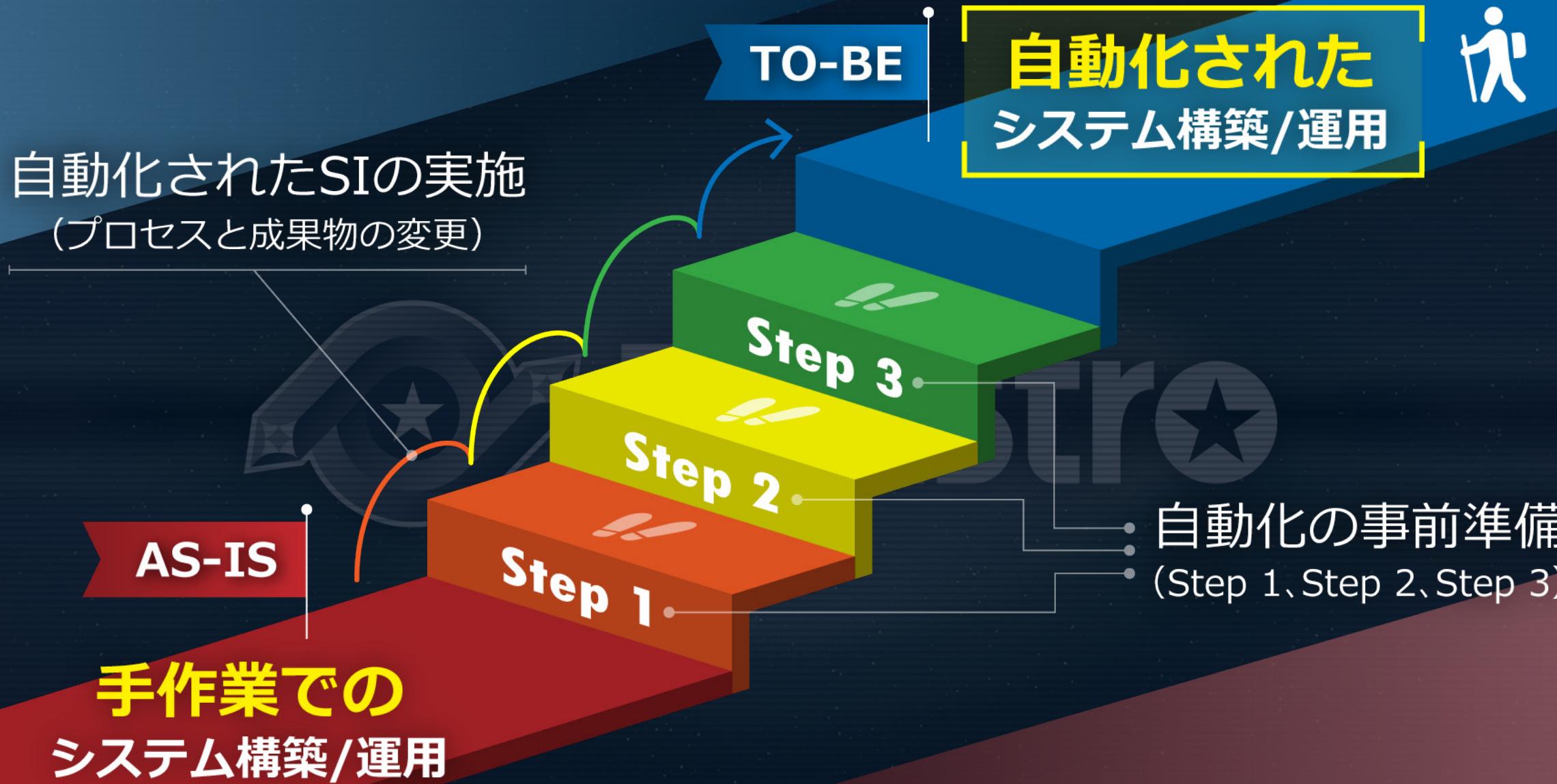
Step 3



5つのタスク

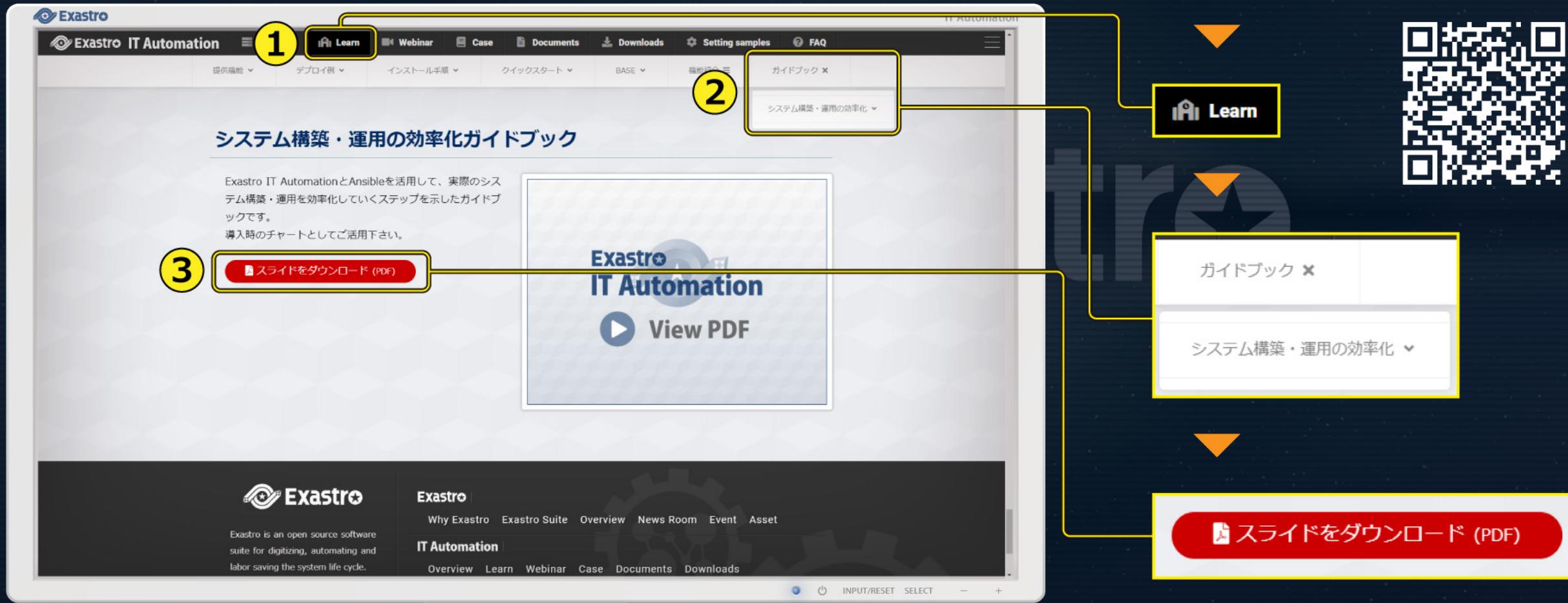
5つのタスク

2つのタスク



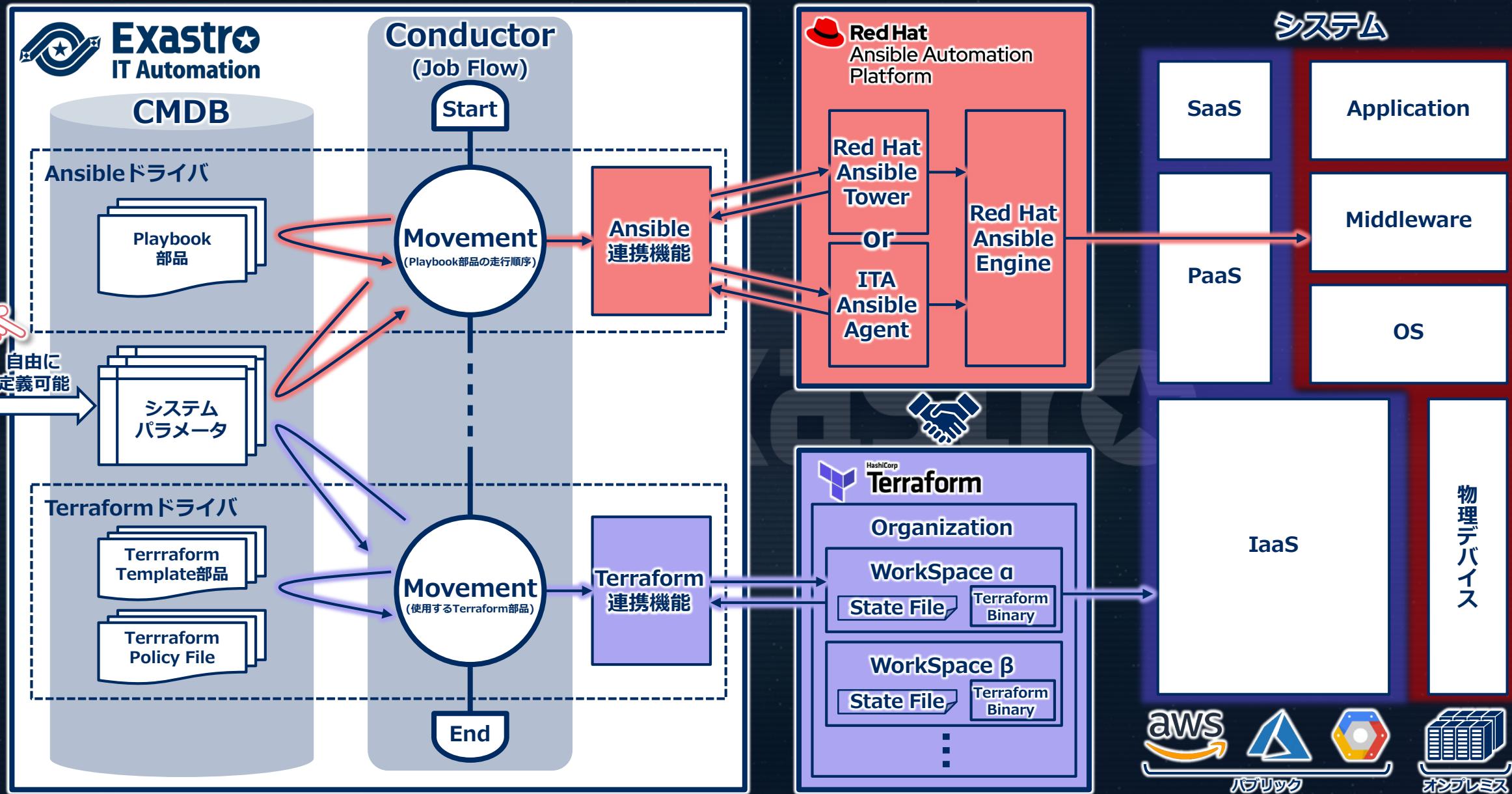
# ガイドはOSSコミュニティーサイトからダウンロードできます

Exastro IT Automation コミュニティサイト [https://exastro-suite.github.io/it-automation-docs/index\\_ja.html](https://exastro-suite.github.io/it-automation-docs/index_ja.html)

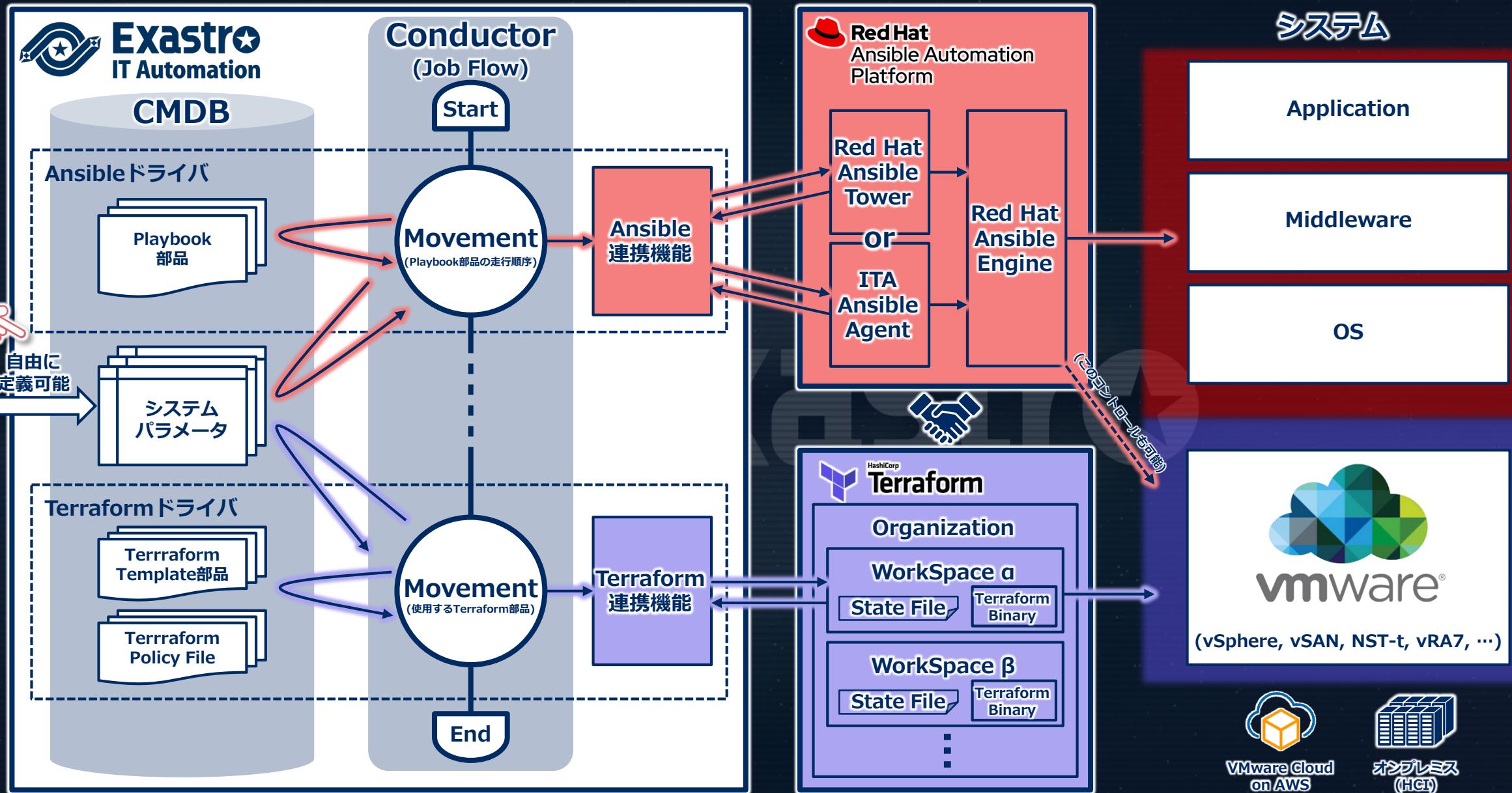


## 参考⑤ 関連ソフトウェアとのエコシステム

# Exastro IT Automation : 自動化ソフトウェアの使いどころ(クラウド等)



# Exastro IT Automation : 自動化ソフトウェアの使いどころ(VMware基盤)



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>



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

参考

## 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

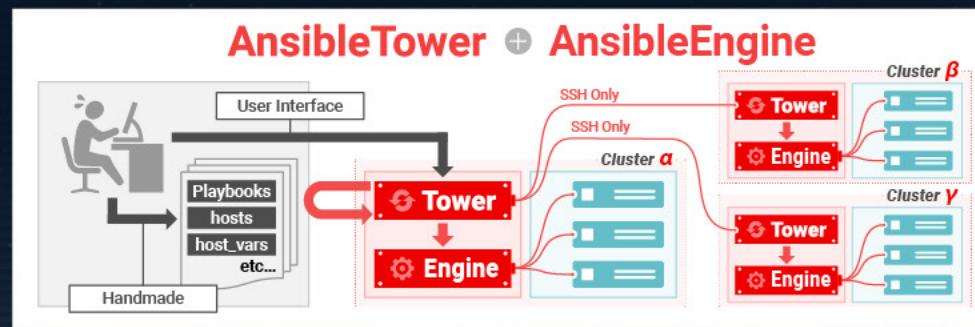
クラスタ間通信をセキュアに、そして異なるバージョンのAnsibleEngineをコントロールします。

## AnsibleEngine

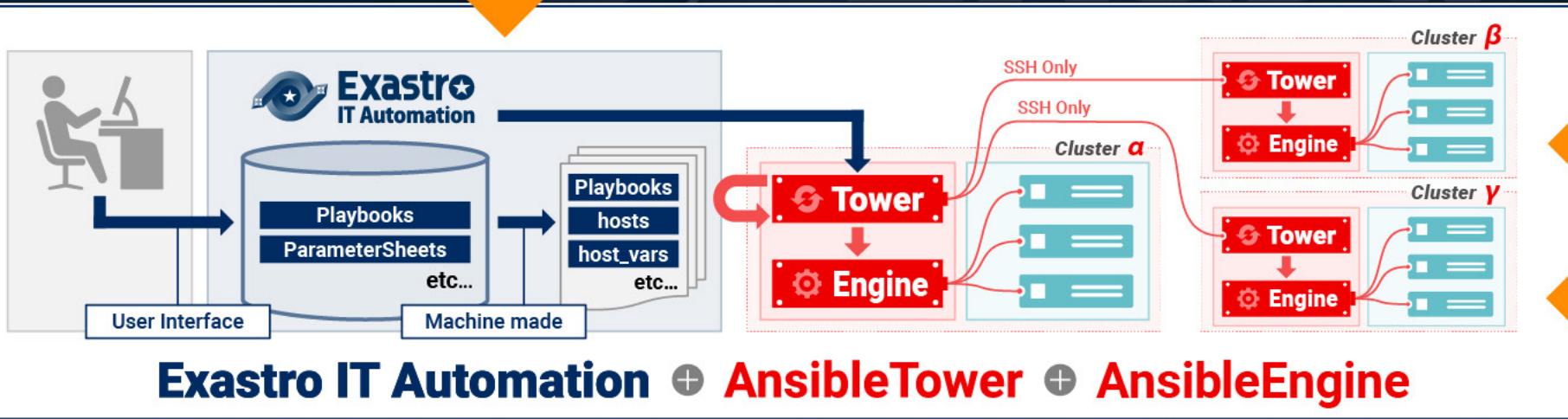
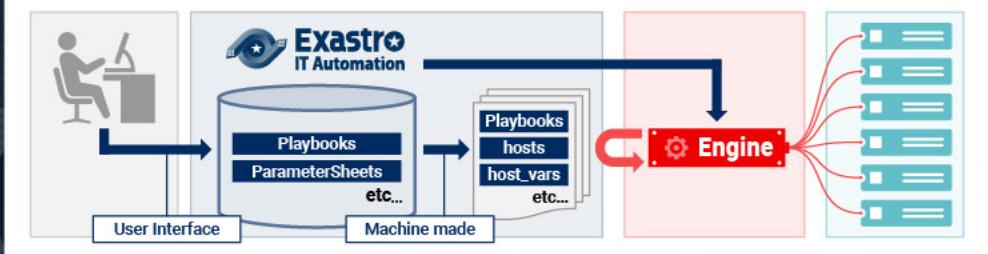
ansible playbookを実行するエンジンです。

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。



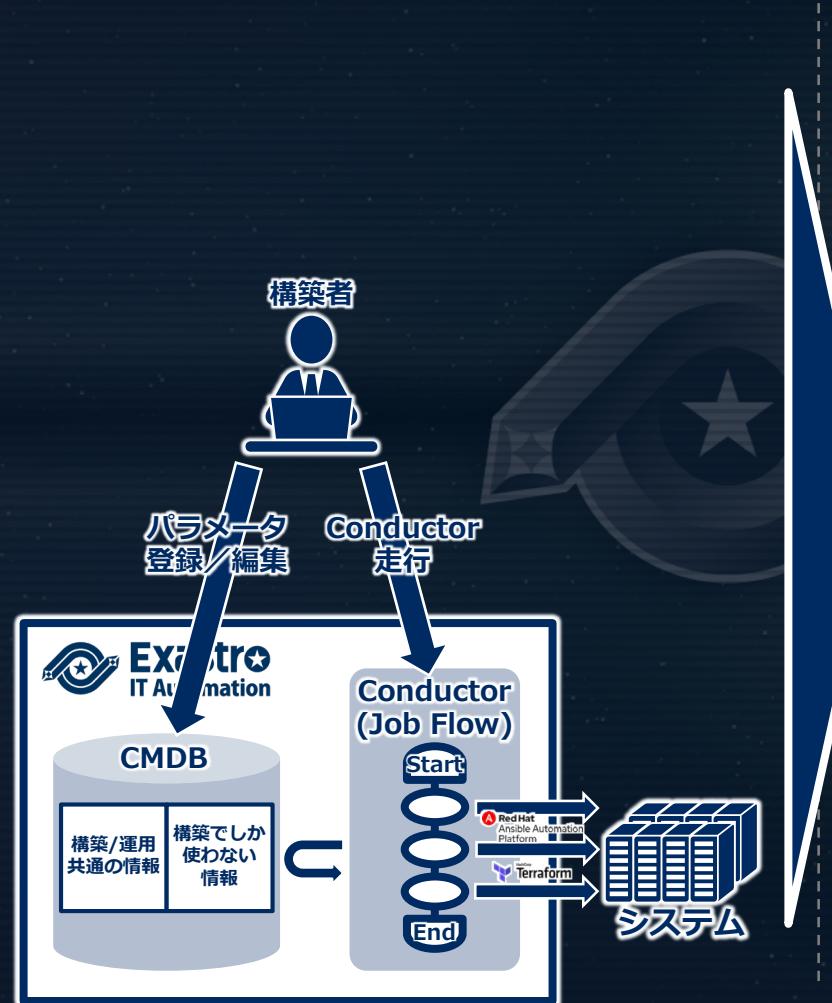
## Exastro IT Automation + AnsibleEngine



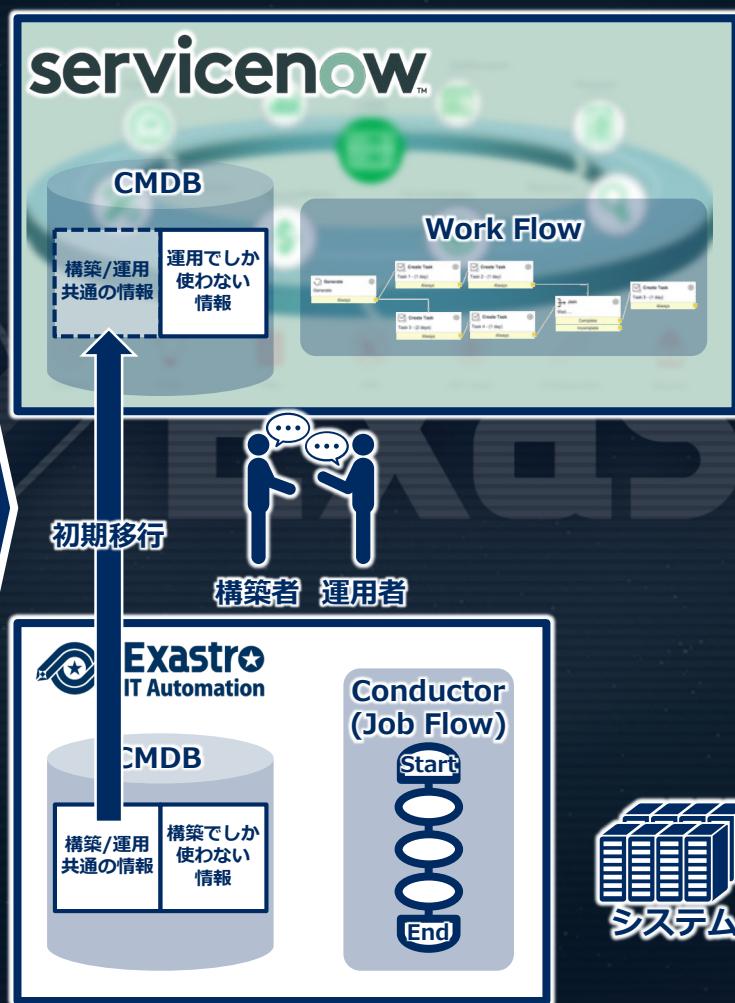
設計管理  
(インプット自動生成)

セキュリティ確保(クラスタリング)・  
複数EngineVer共存

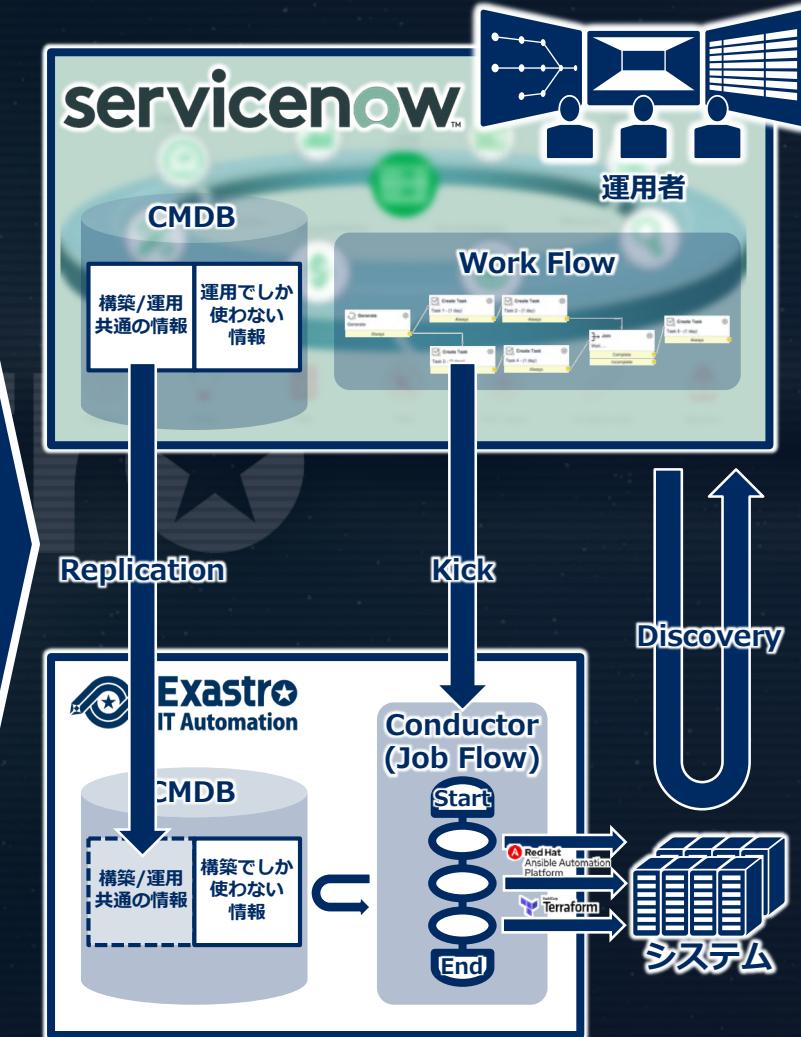
## 構築



## 運用準備



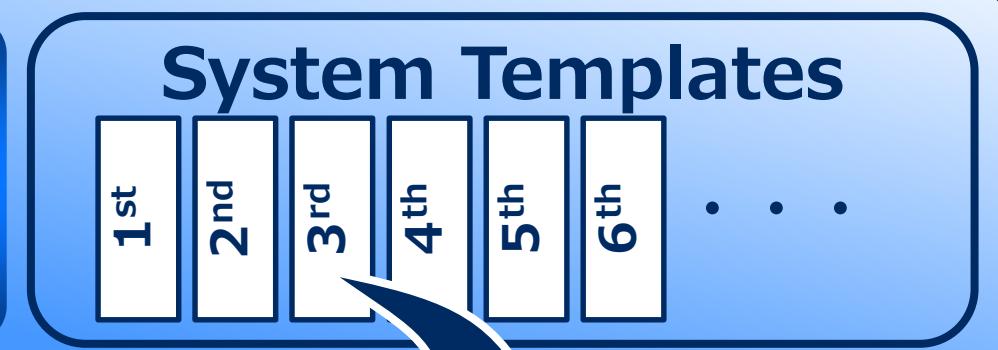
## 運用



# 参考⑥ Setting Samples

## 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)**

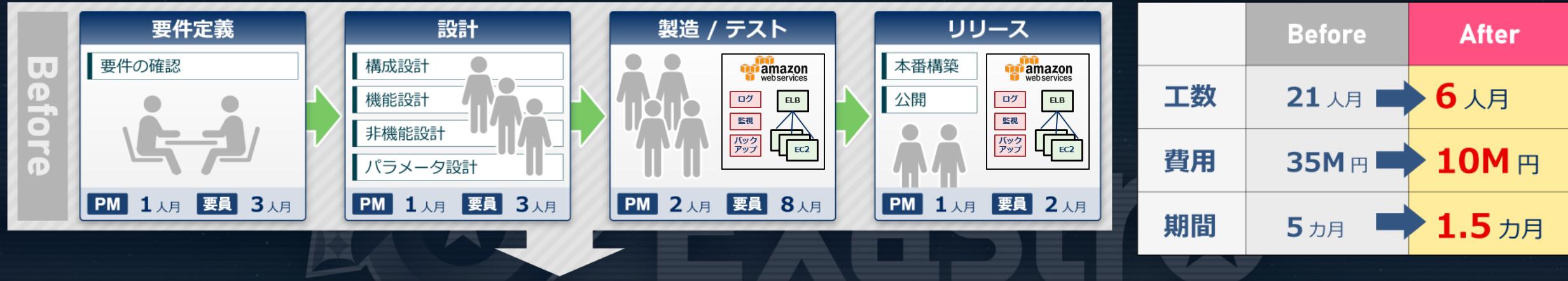


Exastroは  
システムテンプレート  
の実行基盤として  
使い易さを追求

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** 