ITmedia DX Summit Vol.5
プログラム -Day1- 9月14日 ～運用自動化・自律化・リモート～

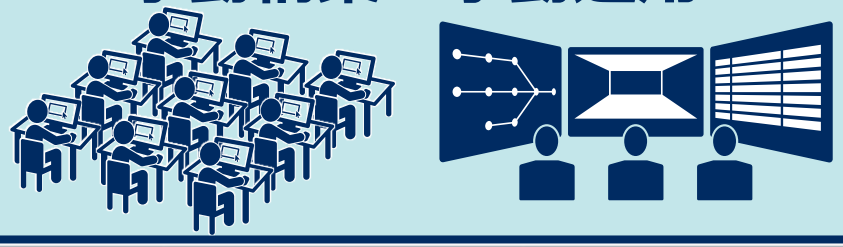
システム構築・運用の自動化を効率化の手段から攻めの武器に繋げる方法

Exastro developer
吉田 功一 Koichi Yoshida koichi-yoshida@nec.com

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

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



Mobile | Social | IoT | 5G — Tech

スケーラブルなアプリ — AP
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 — PF
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 — OP

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

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

多くのITエンジニアが張り付いている

ITエンジニア不足

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
AP

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
PF

自動構築・自律運用
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

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

自動構築・自律運用

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

AP

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

PF

自動構築・自律運用

OP

(1)構築・運用を自動化・省力化して

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

Mobile　Social　IoT　5G　Tech

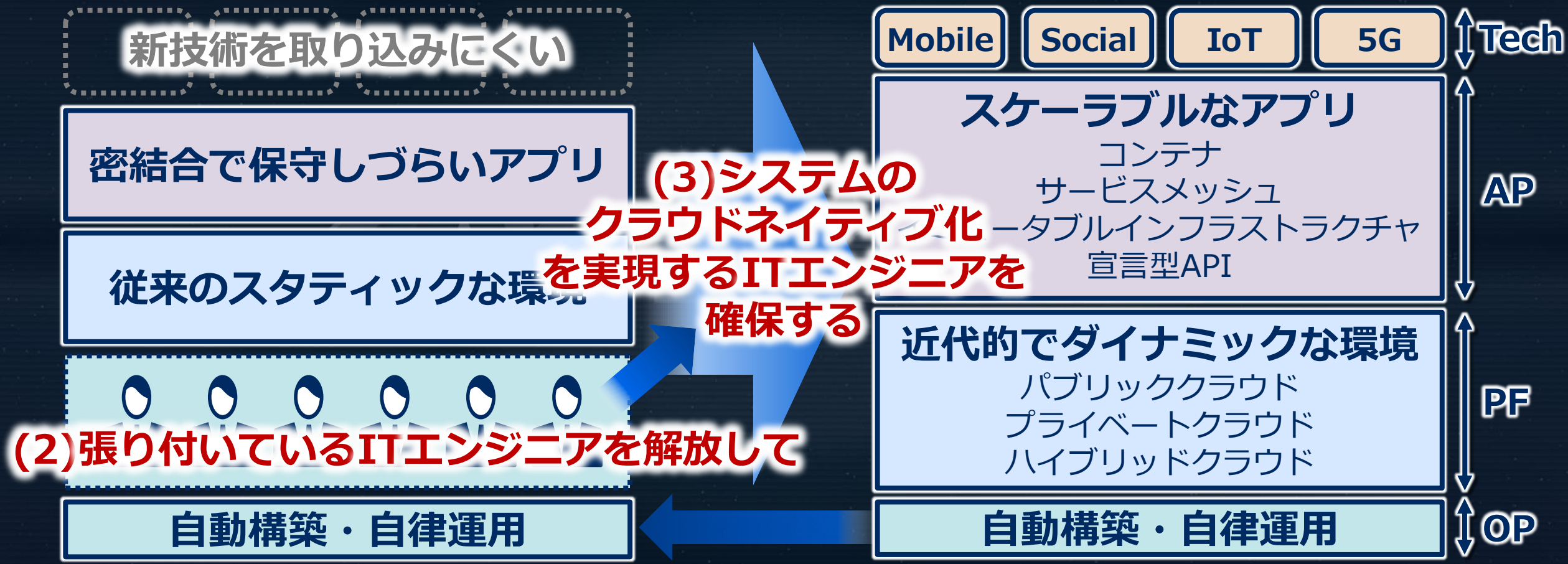
スケーラブルなアプリ
コンテナ
サービスメッシュ
イミュータブルインフラストラクチャ
宣言型API
AP

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
PF

自動構築・自律運用
OP

(1)構築・運用を自動化・省力化して

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

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[illegible]な環境

【守りの自動化】
モノリシックな現行システムに
自動構築・自律運用を適用するためには
"ひと工夫"いる
(でもやらなければならない)

自動構築・自律運用

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

AP

【攻めの自動化】
スピーディなサービス展開のためには
ブルーグリーンデプロイメントなどの手法
に見られるように「再構築」が繰り返される
作業量の爆発を防ぐために自動構築は必須

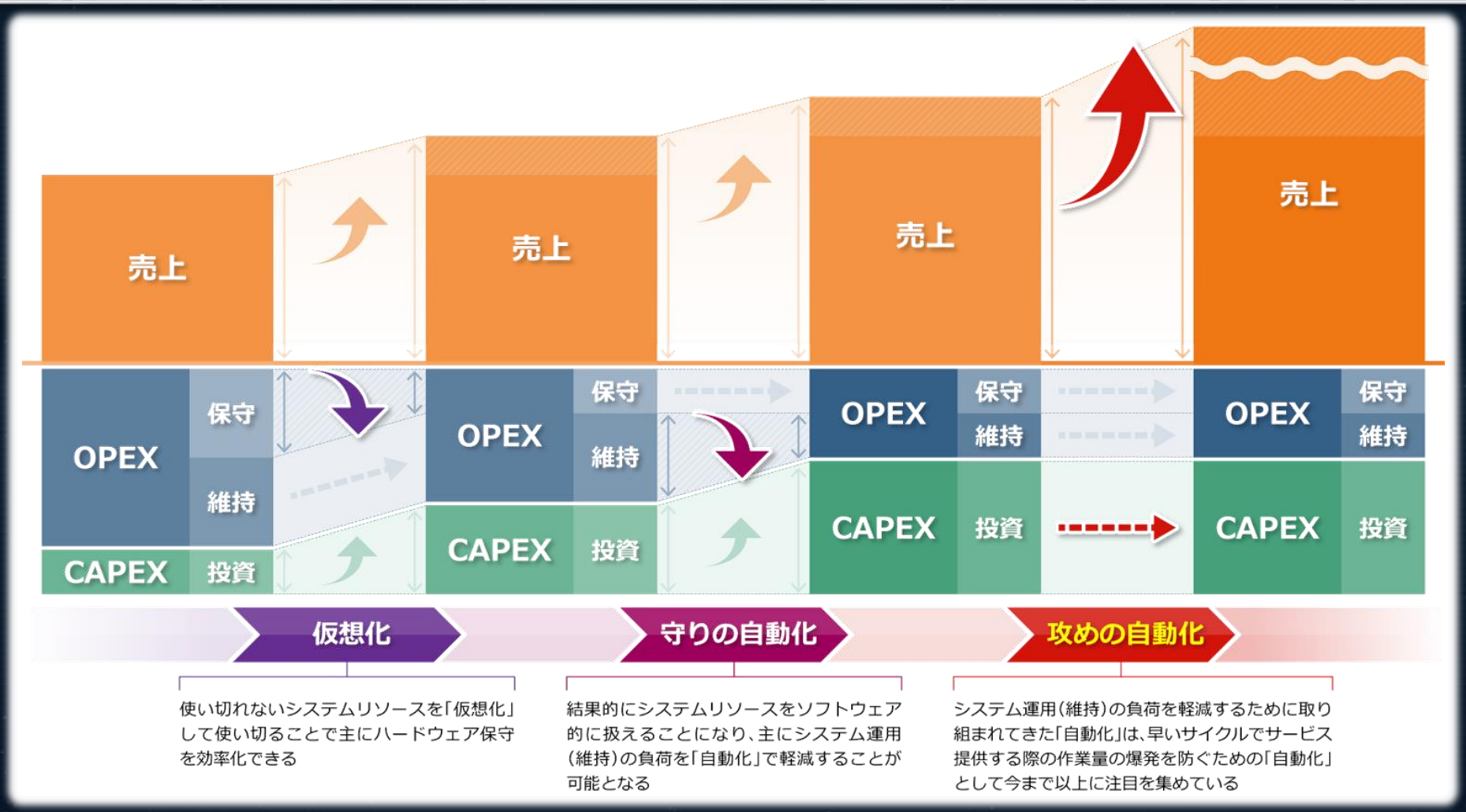
PF

自動構築・自律運用

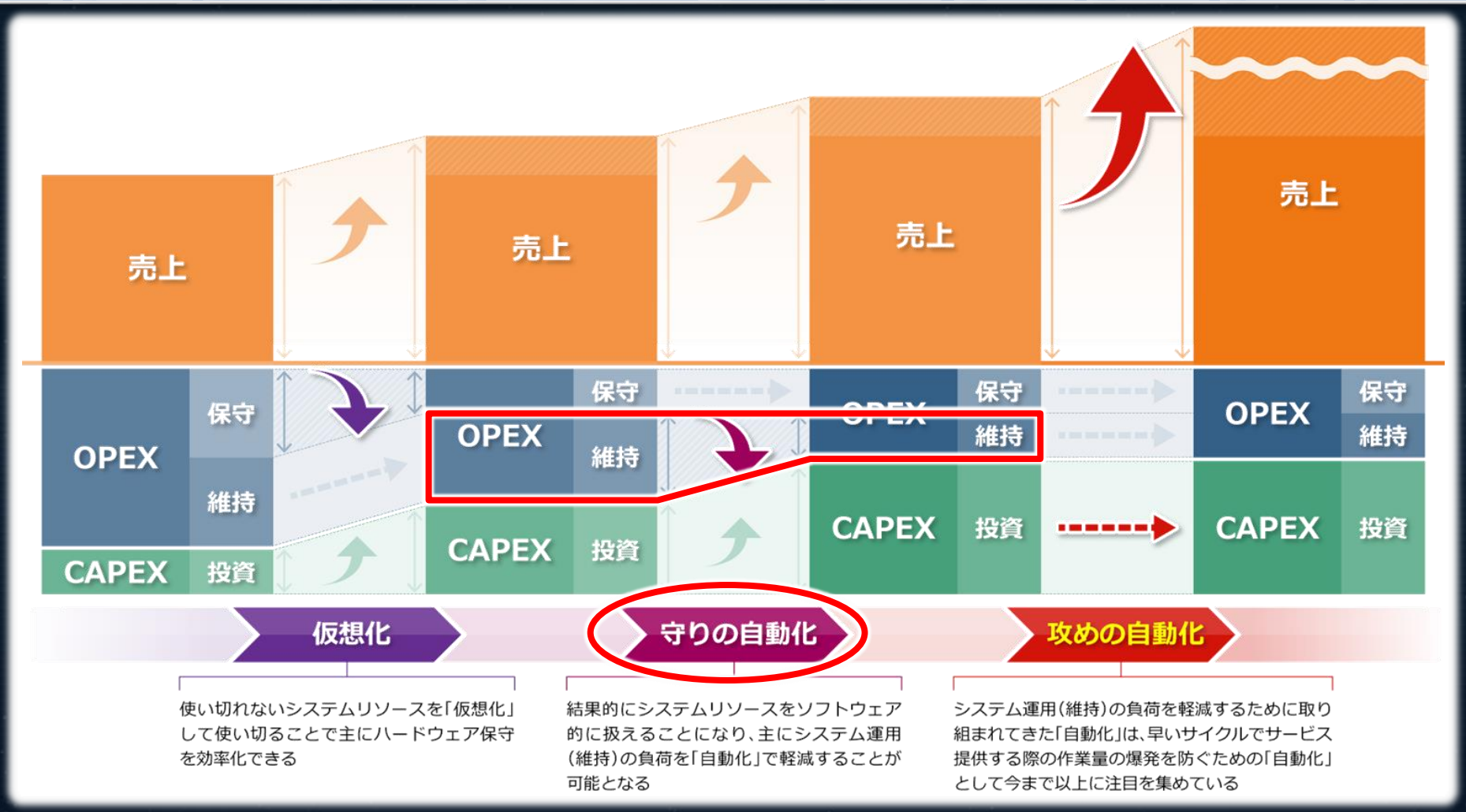
OP

別々に議論されてきた

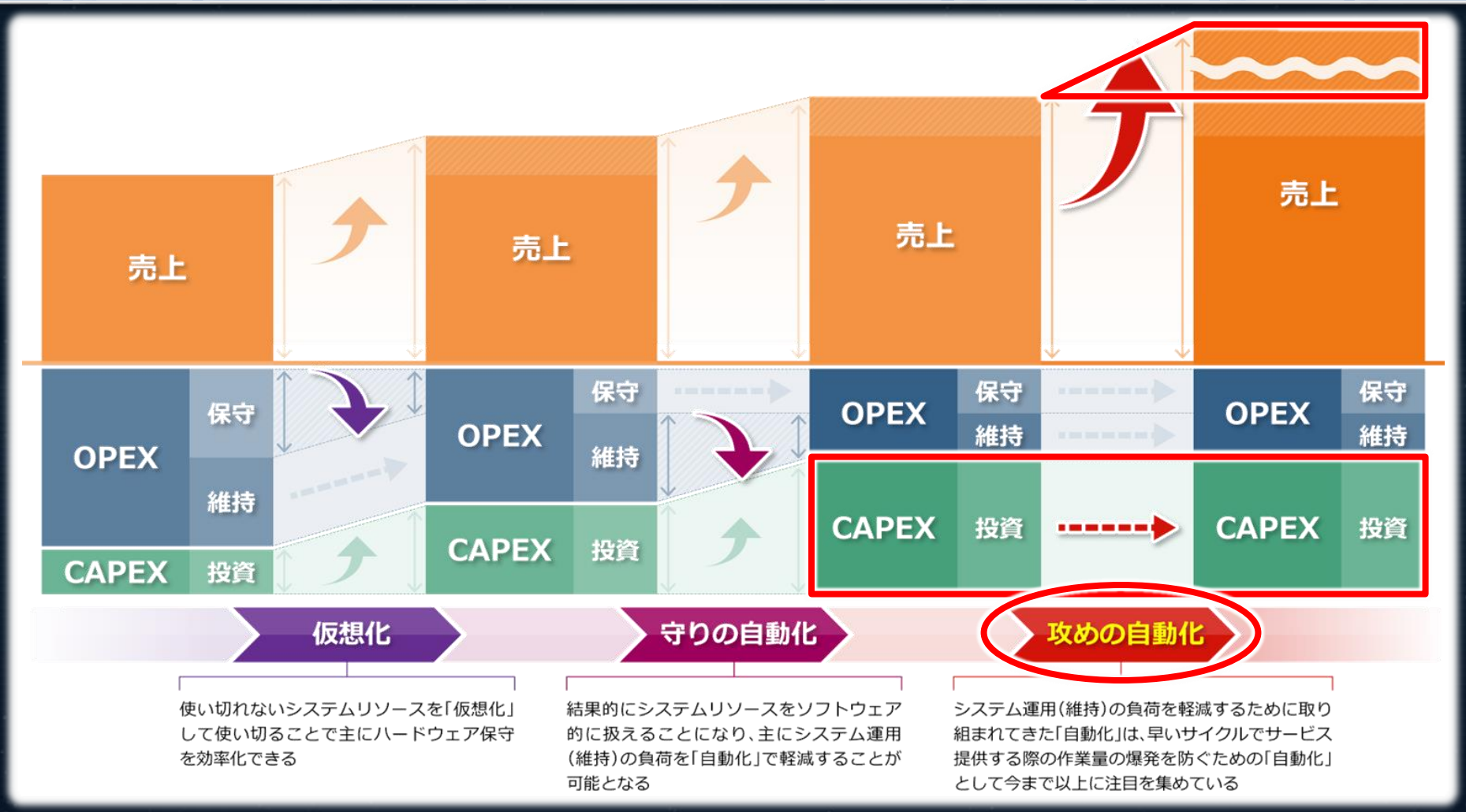
2つの領域の自動化は財務的な視点でも目的を異にします



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

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

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

AP

実際にはハイブリッドでシステムが構成されることが多く

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

PF

自動構築・自律運用

OP

モノリシック／クラウドネイティブの
ハイブリッドなシステムの構築・運用

それでは「守りの自動化」と「攻めの自動化」について解説します

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

AP

近代的でダイナミックな環境

PF

クラウドネイティブなシステムの自動化（攻めの自動化）

【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」

Exastro

モノリシックなシステムの「構築」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

- ☑ 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

モノリシックなシステムの「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
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

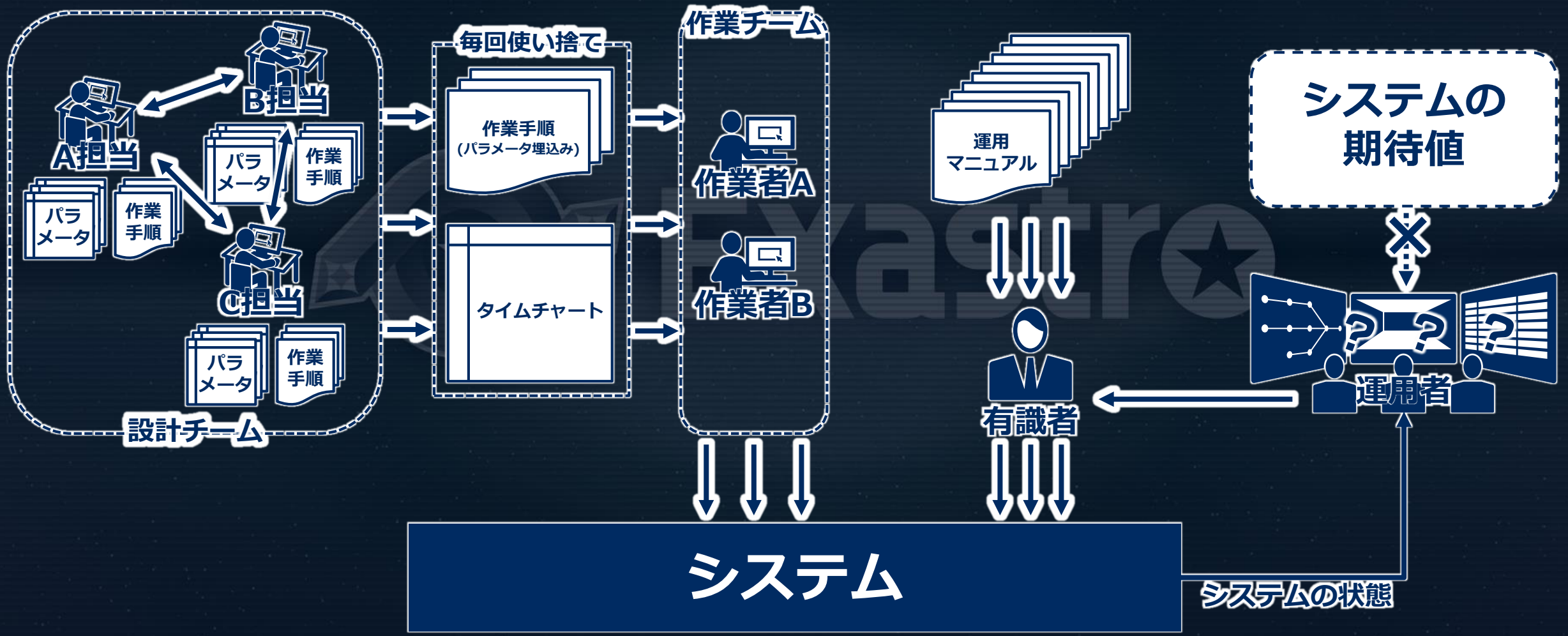
- ☑ 運用上変更してよいパラメータと変更してはいけないパラメータが把握できていない
- ☑ システムのパラメータの現在値や過去の変更履歴を管理できていない
- ☑ 結果としてせっかくパラメータ化されているのに運用で利用できていない

- ☑ 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人体制で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

- ☑ 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

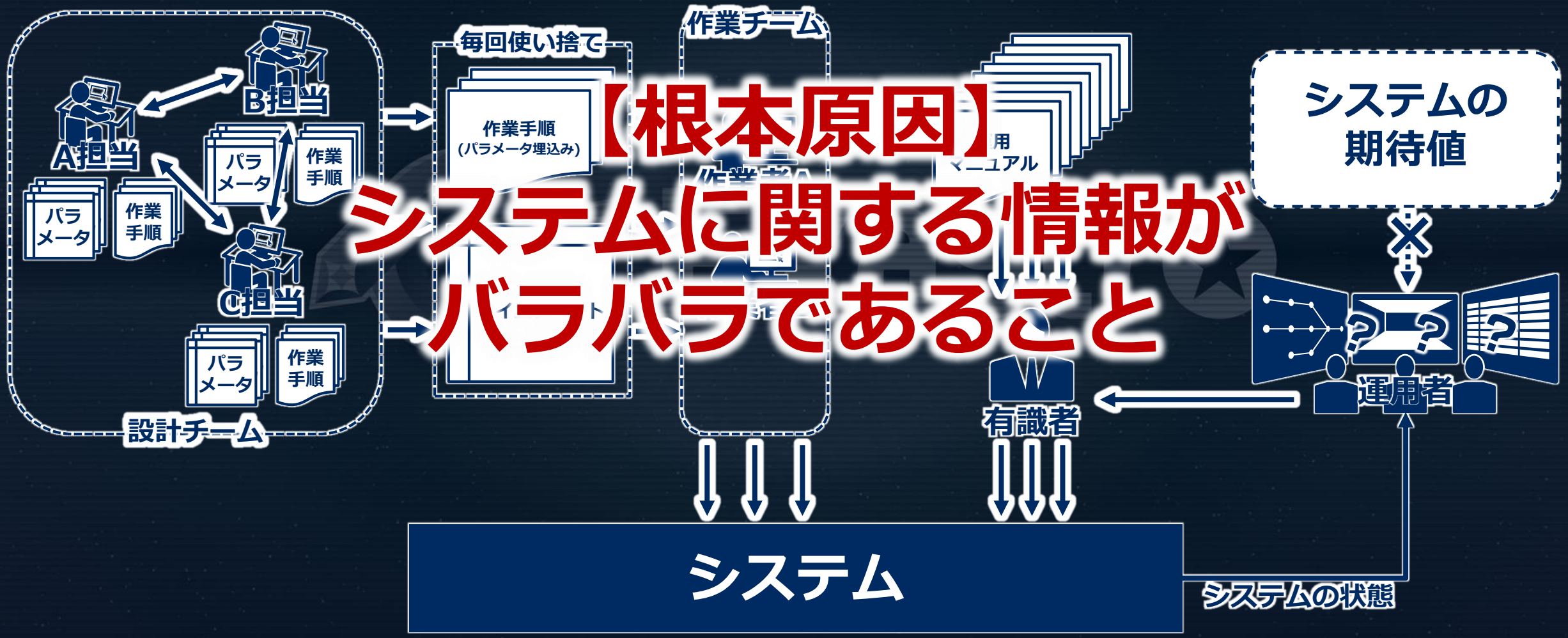
ITエンジニアの「苦」の根本的な原因

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



ITエンジニアの「苦」の根本的な原因

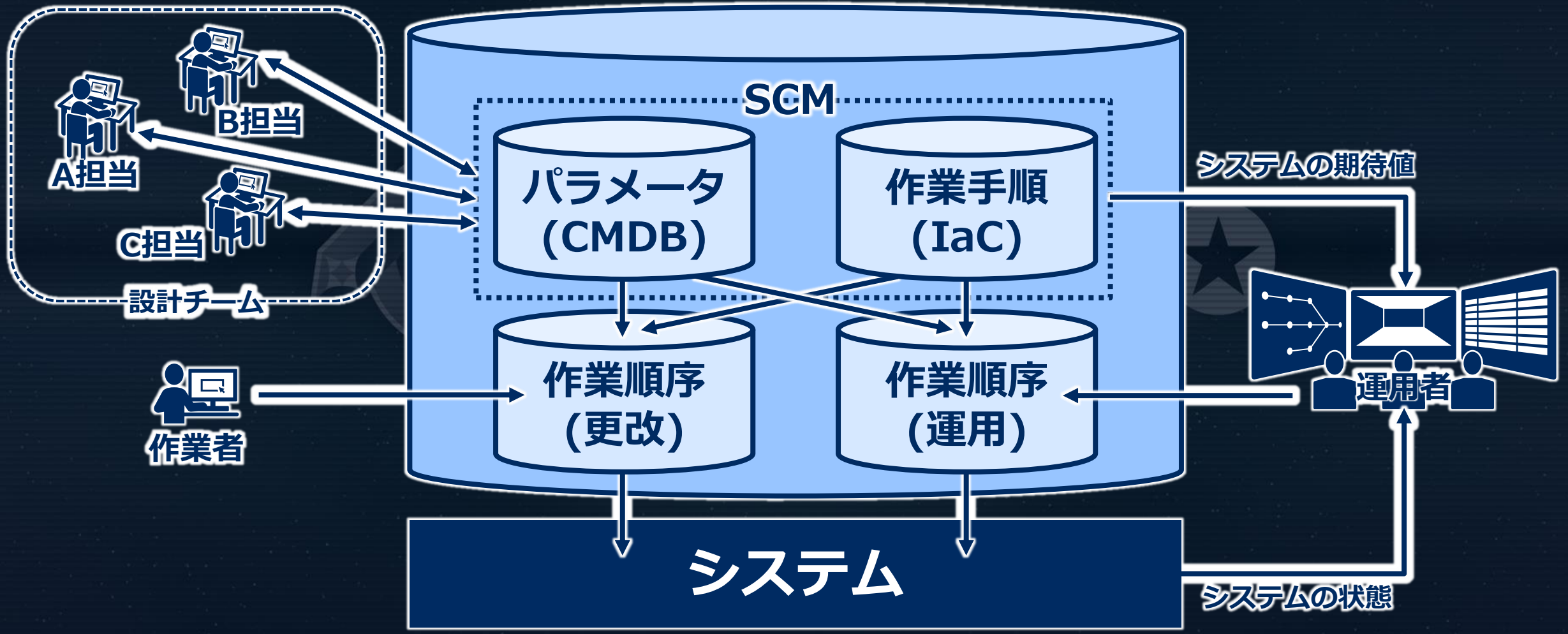
関係者は散乱した情報を正確に伝えることに多くの時間を費やしています



【根本原因】
システムに関する情報がバラバラであること

ITエンジニアの「苦」を解決するためには？

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・・・



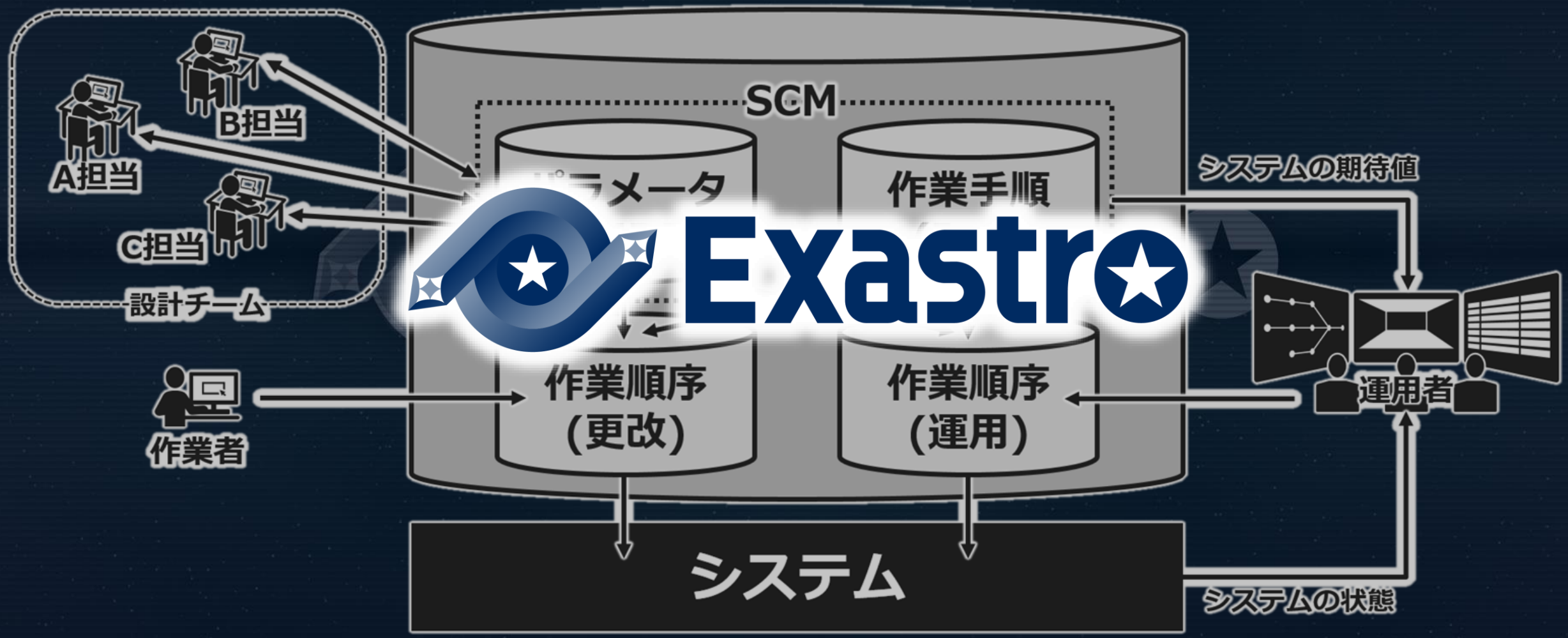
ITエンジニアの「苦」を解決するためには？

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・・・

【解決手段】
システムに関する情報を一元管理(デジタル化)すること

Exastro Suite(OSS) が使えます

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

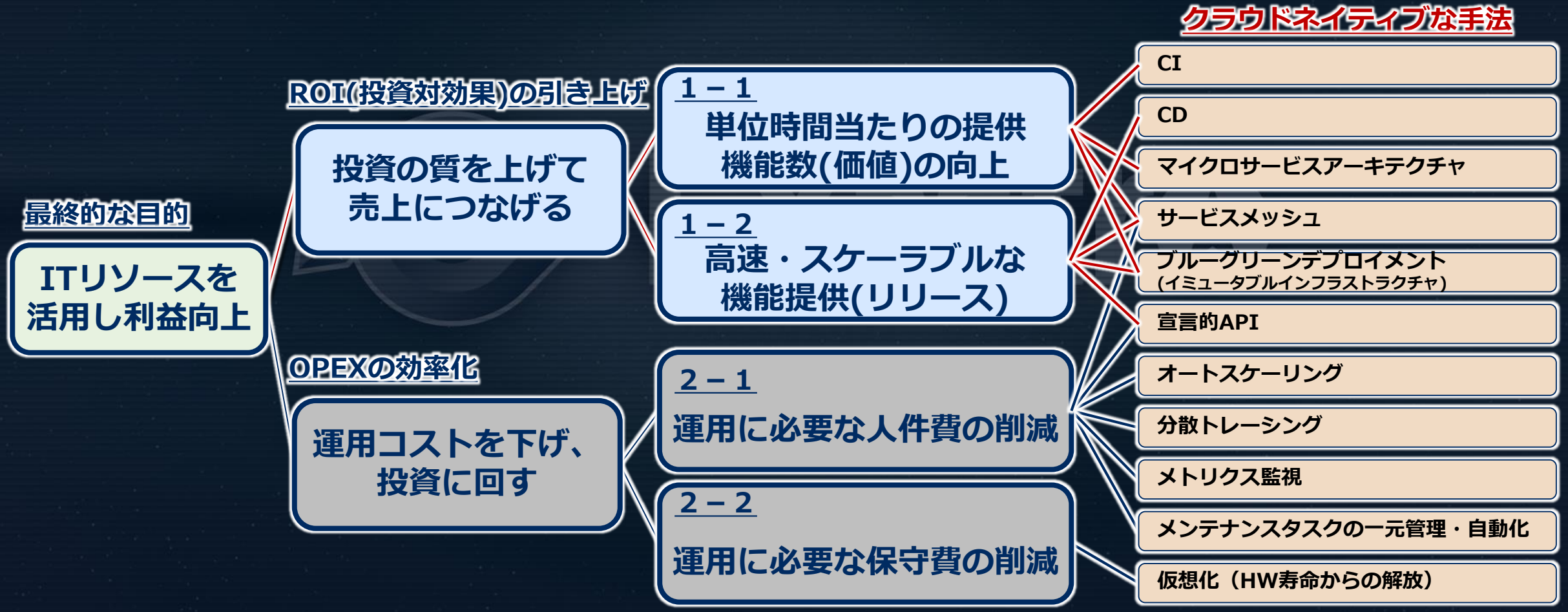


【攻めの自動化】
ROI(投資対効果)を引き上げるために必要なクラウドネイティブのテクニック

Exastro

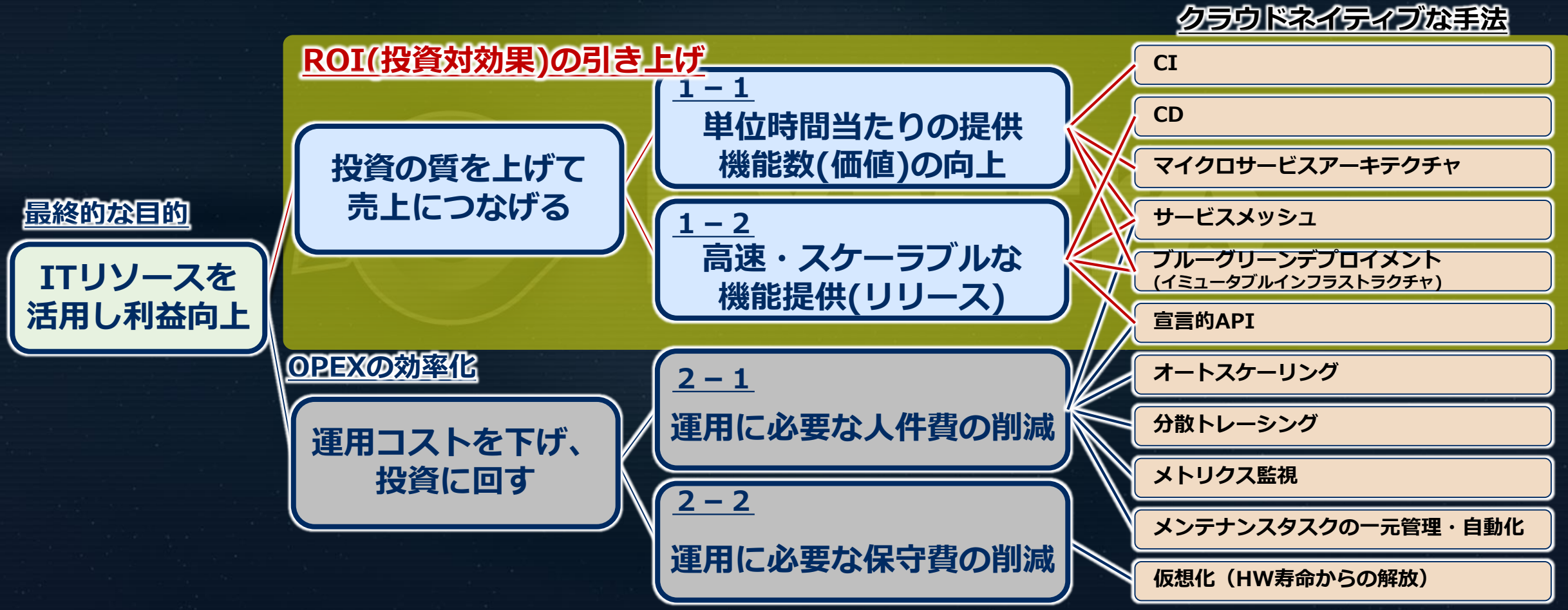
ROI(投資対効果)を引き上げるために必要なクラウドネイティブのテクニック

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



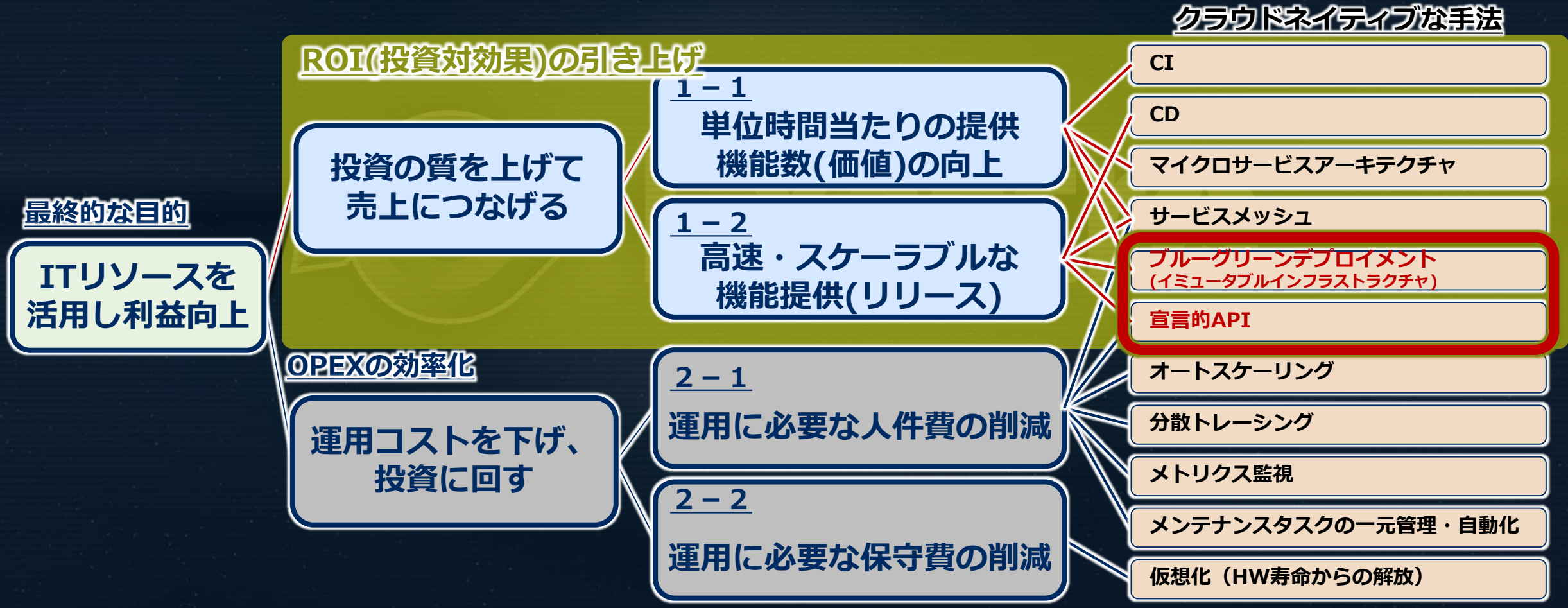
ROI(投資対効果)を引き上げるために必要なクラウドネイティブのテクニック

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



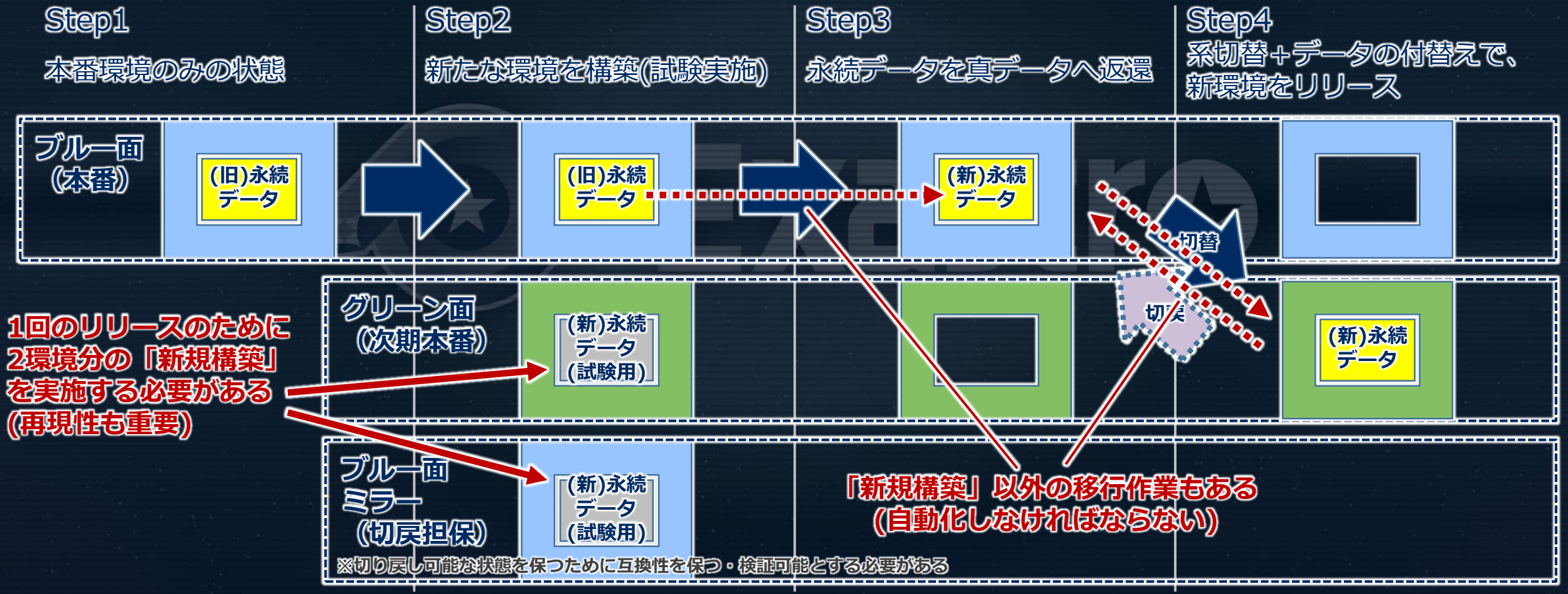
ROI(投資対効果)を引き上げるために必要なクラウドネイティブのテクニック

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



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

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



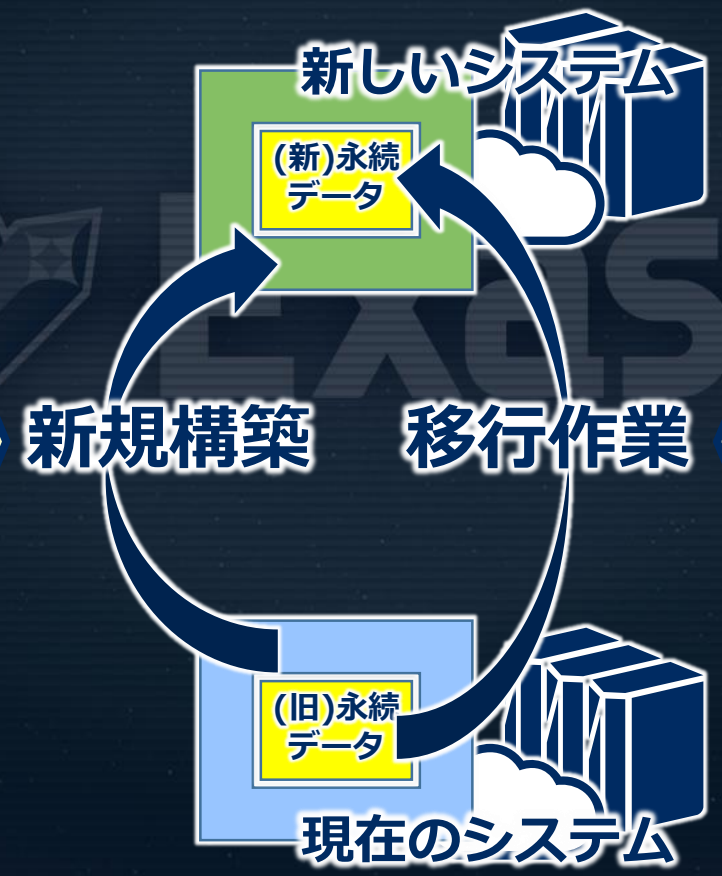
ROI(投資対効果)を引き上げるために必要なクラウドネイティブのテクニック

「新規構築」可能な部分は宣言的API(あるべき姿を定義)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。
〇〇が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

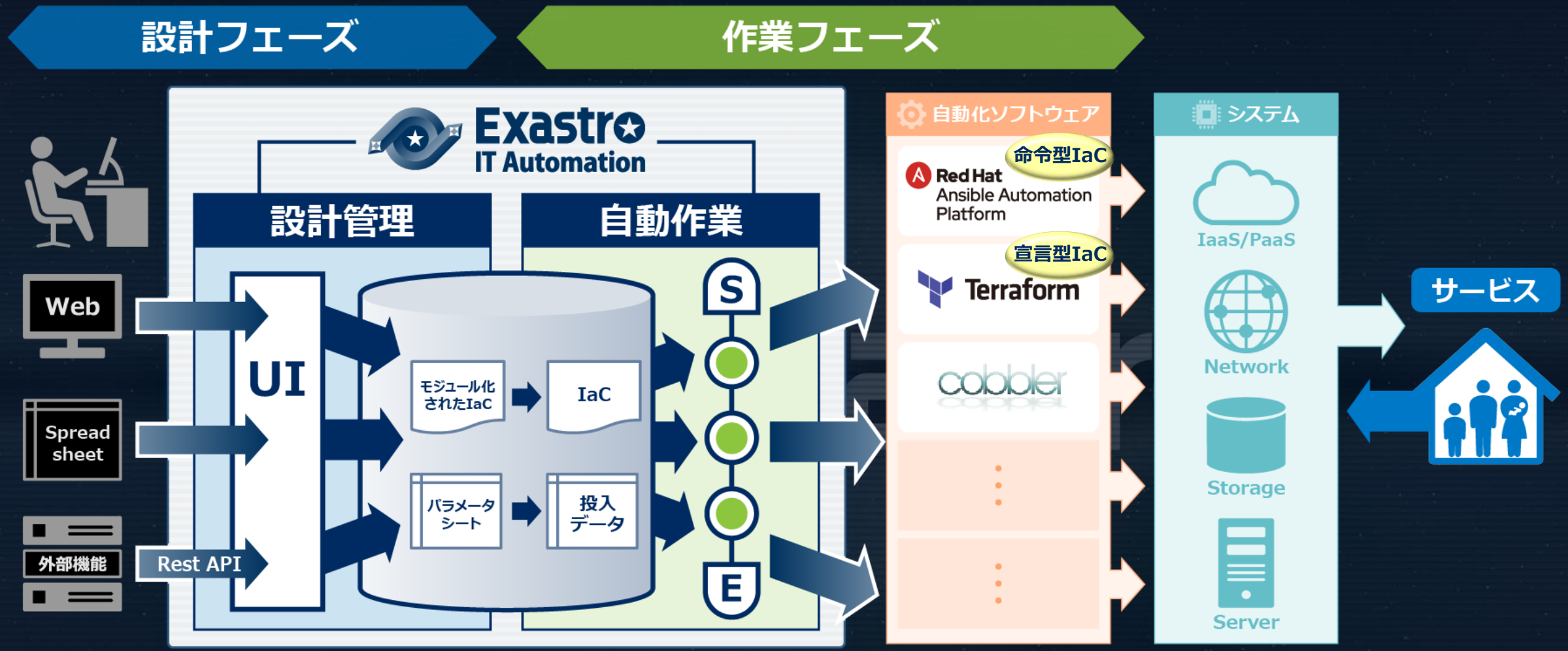
まとめ：「攻めの自動化」を実現するうえで重要なポイント

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でも「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。

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



7つの特徴

1. マルチインタフェースとRBAC
2. パラメータをグルーピング／履歴管理する
3. IaCを解析して変数を刈り取る
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

7つの特徴

1. マルチインタフェースとRBAC
2. パラメータをグルーピング／履歴管理する
3. IaCを解析して変数を刈り取る
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

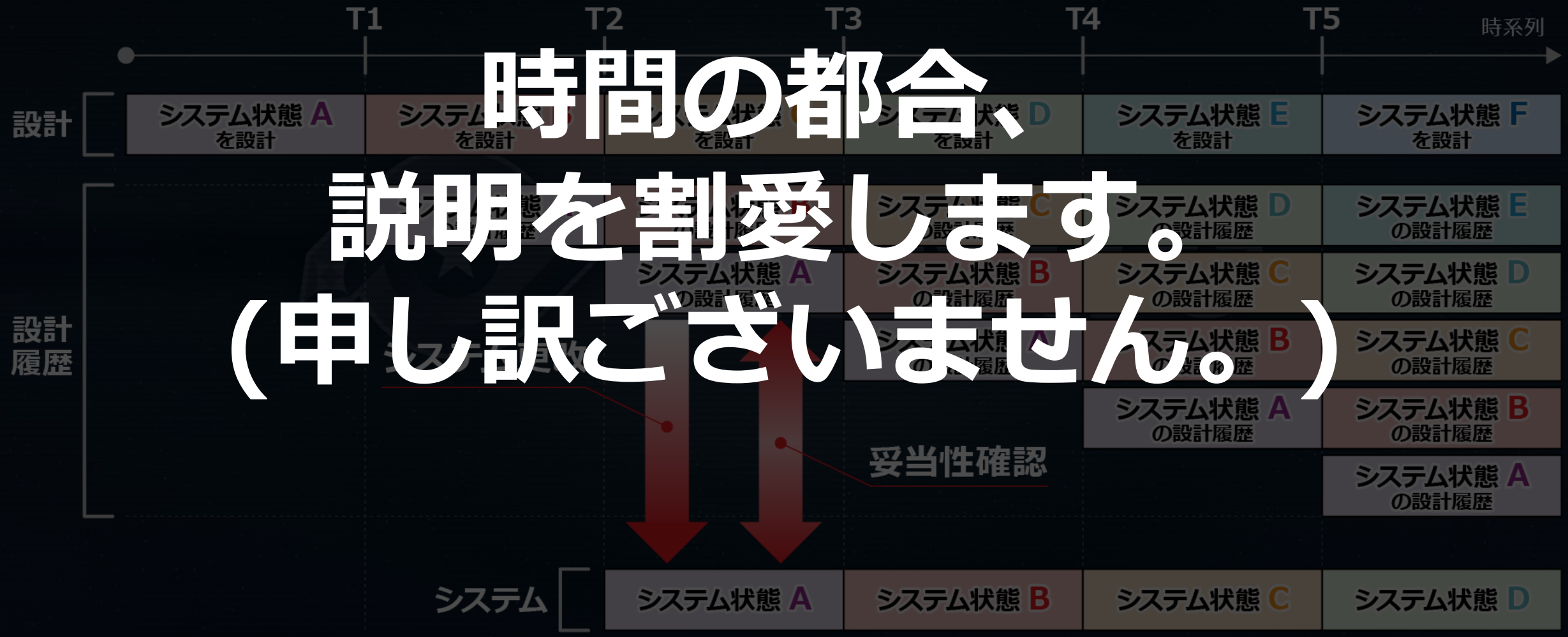
1つめの特徴 - マルチインタフェースとRBAC

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来る操作(閲覧のみ、実行)を制御できる

時間の都合、説明を割愛します。(申し訳ございません。)

※ユーザ1は、メニューCに対して『メンテナンス可』と『閲覧のみ』両方の権限がありますが、『メンテナンス可』の権限が優先されます。

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



時間の都合、
説明を割愛します。
（申し訳ございません。）

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

時間の都合、
説明を割愛します。
（申し訳ございません。）

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	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

"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

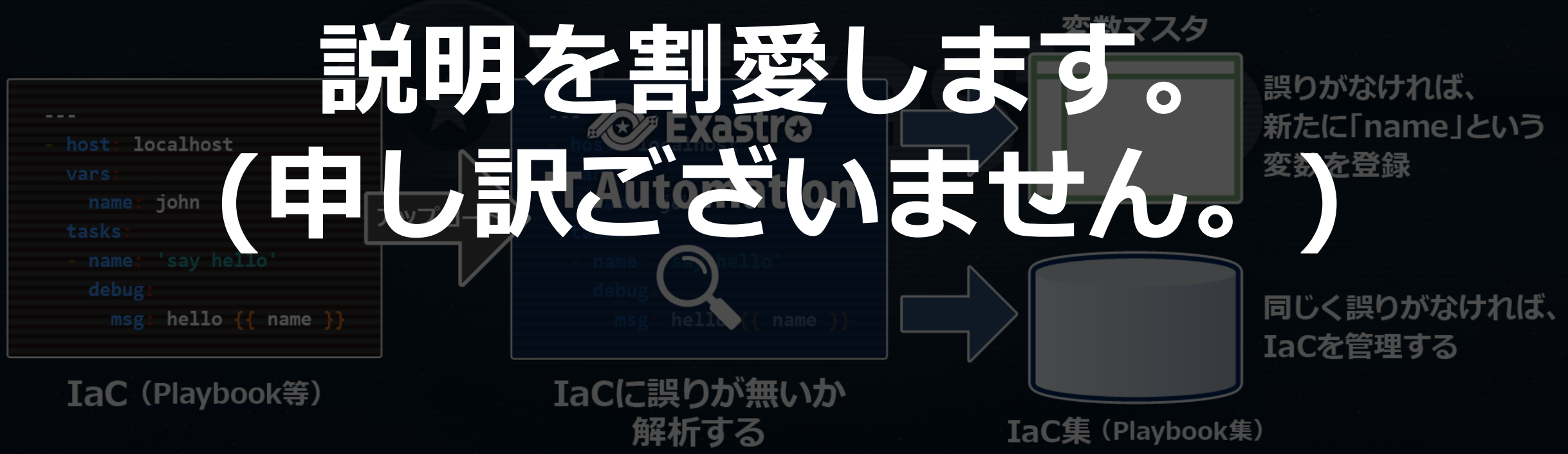
妥当性確認

システム

3つめの特徴 - IaCを解析して変数を刈り取る

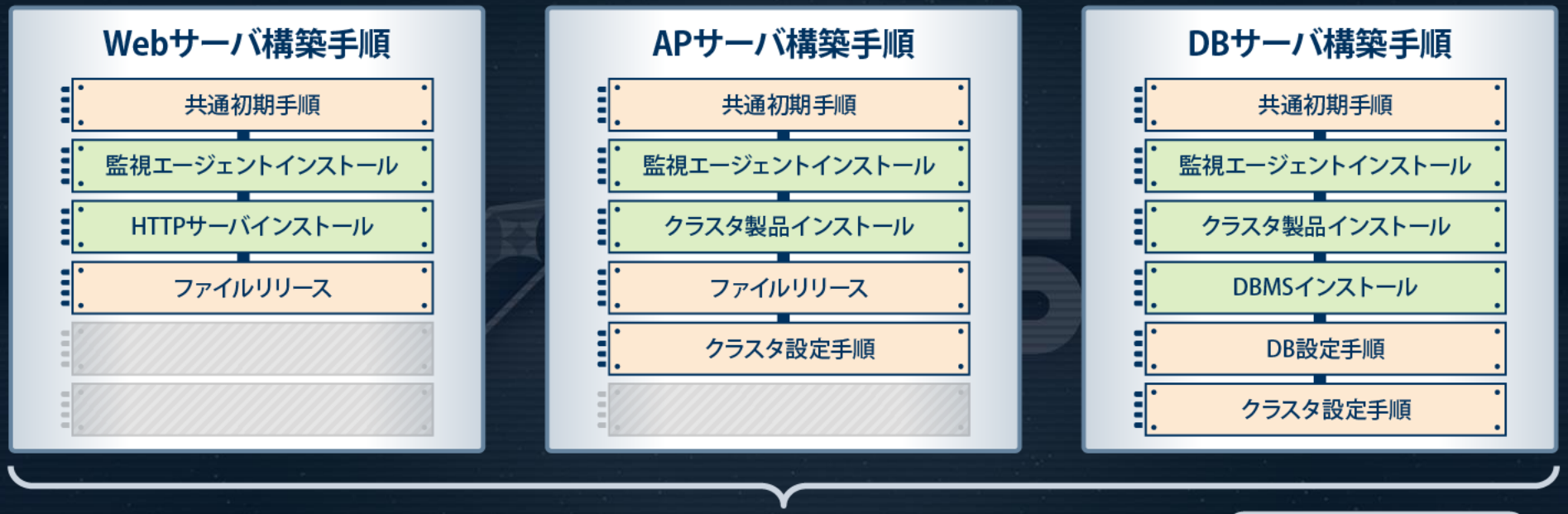
IaCがアップロードされるとまずIaCに誤りが無いか解析する
誤りがなければ、IaCの記述から変数名を刈り取って管理する
変数名を選択式で利用し記述等のヒューマンエラーを防止する

時間の都合、
説明を割愛します。
（申し訳ございません。）



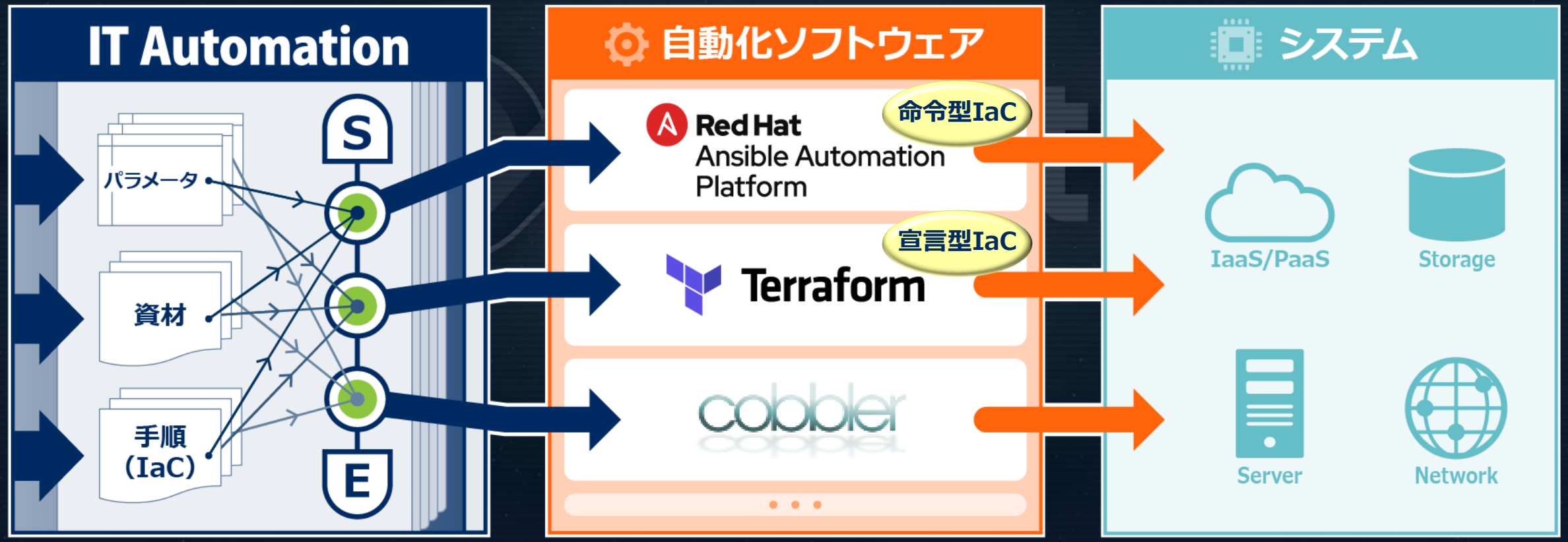
4つめの特徴 - IaCをモジュール管理して再利用性を高める

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



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

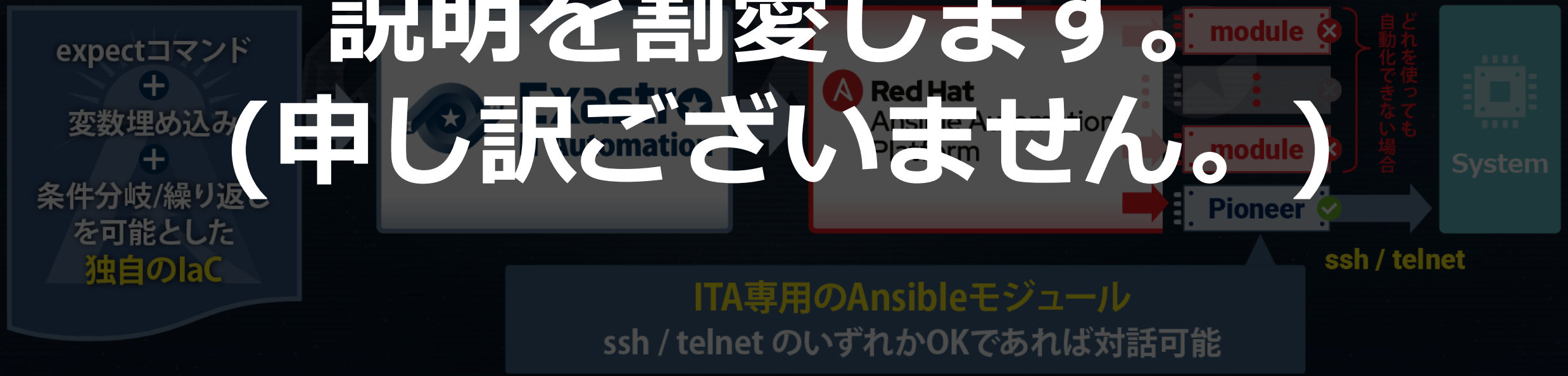
複数の自動化ソフトウェアを繋げて一本の作業フローを定義できる
また自動化ソフトウェアの動作に必要な投入データを自動生成する
例) (Ansibleの場合) 必要なPlaybookを集めて繋げ、ノード毎にパラメータからhost_varsを作る



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。

時間の都合、説明を割愛します。（申し訳ございません。）



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

時間の都合、
説明を割愛します。
(申し訳ございません。)

モノリシック／クラウドネイティブのハイブリッドなシステムの自動化

Exastro

モノリシック／クラウドネイティブのハイブリッドなシステムの自動化

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

活用事例 (時間が許す限りご紹介します…)

【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

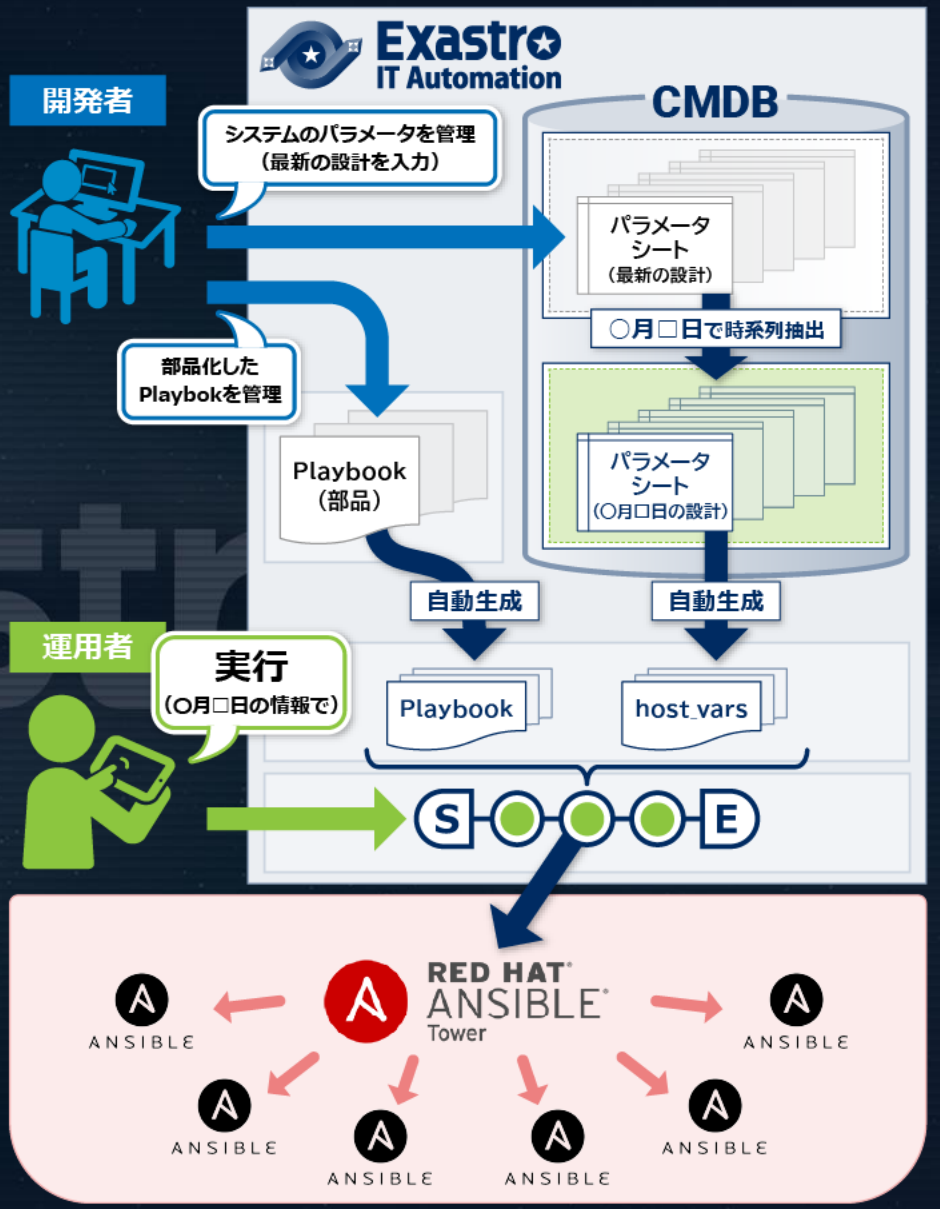
システムのパラメータが時系列で構成管理できておらず、
またPlaybookの部品化ができておらず、構築および運用における作業の自動化/効率化が実現できていなかった。

☑ 解決策

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。

👍 効 果

	例1:運用監視設定作業	例2:運用コマンドリストメンテナンス作業
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	年間4,000万円の作業工数で実施していた 4,000万円/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)	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

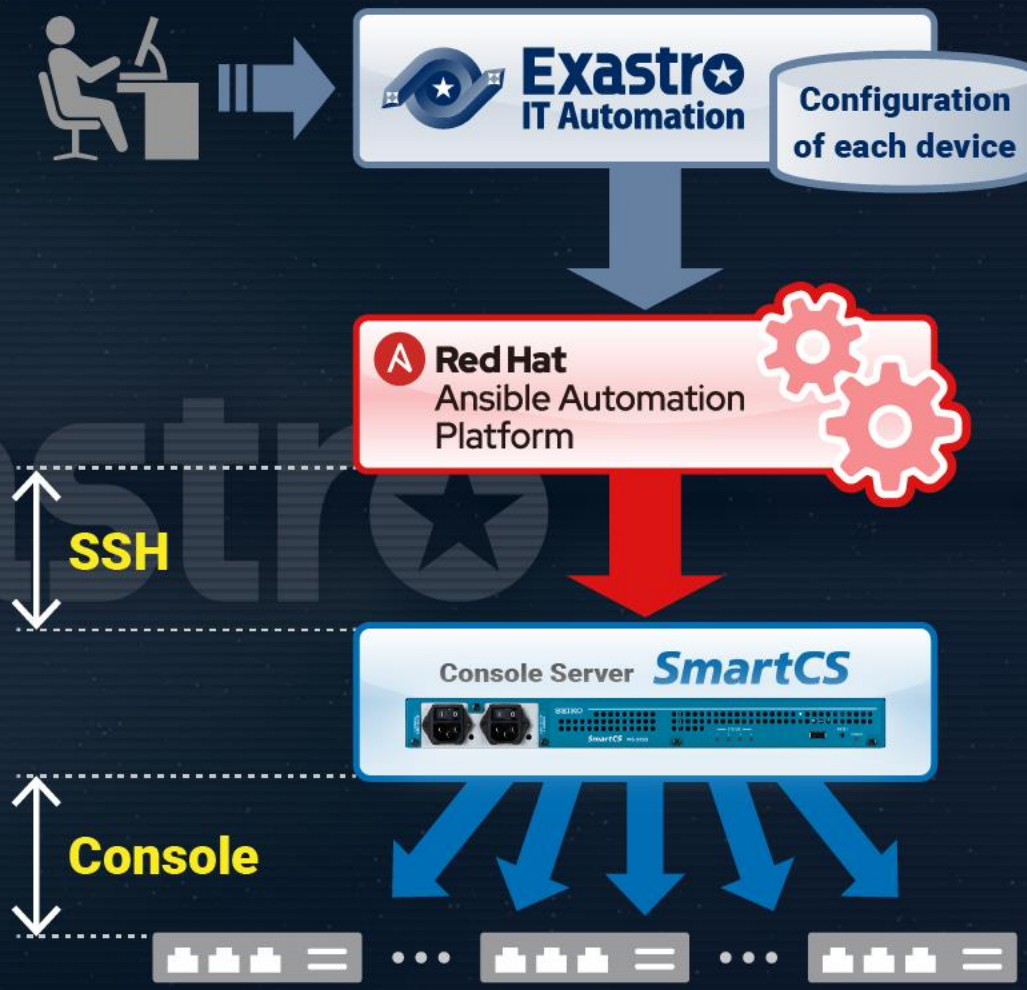
大量のネットワーク機器の初期設定を行う必要があった。しかし、手作業では多くの工数が必要であり、また大量の設定値を表計算ソフトで管理するのにも限界があった。

☑ 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

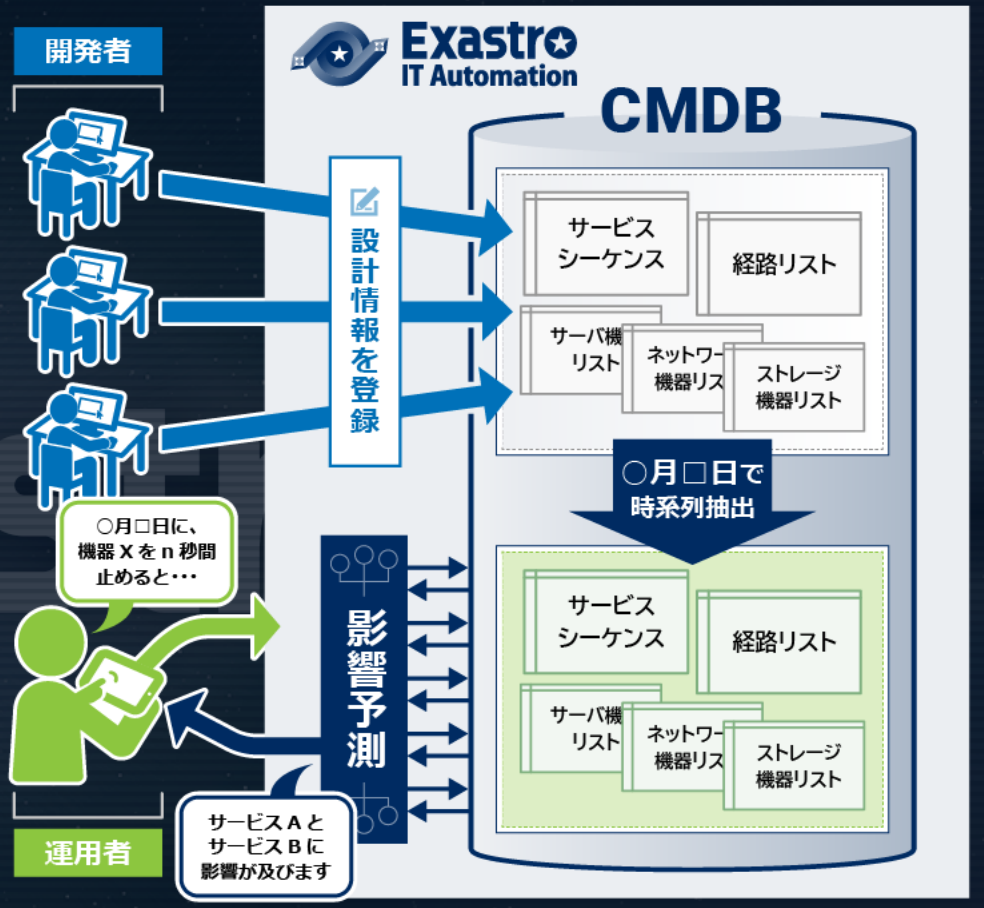
大規模キャリアシステムにおいて、計画および突発の機器停止によるサービス影響の調査に多くの工数がかかっていた。

☑ 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

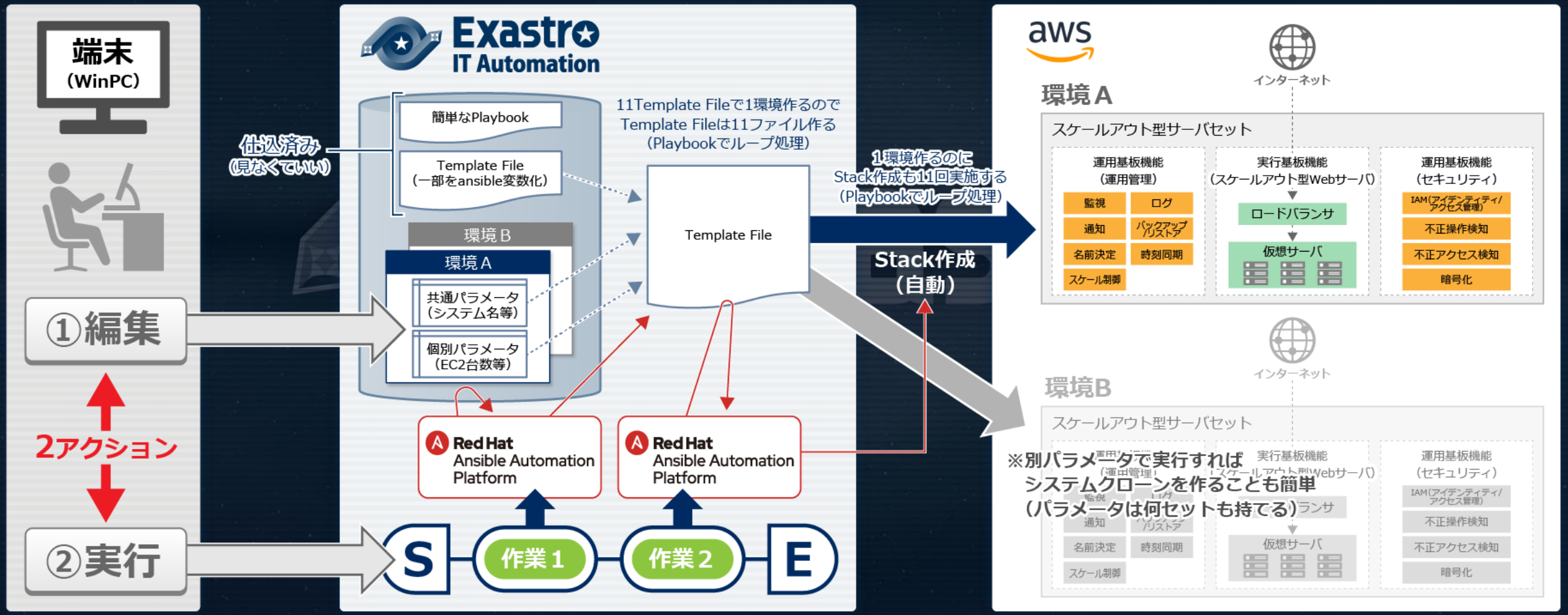
1回の調査で約80万円の工数/費用を要していたところ、本施策により工数不要となった。
年間約120回の調査があったので
約9,400万円/年の削減効果が確認された。



※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。

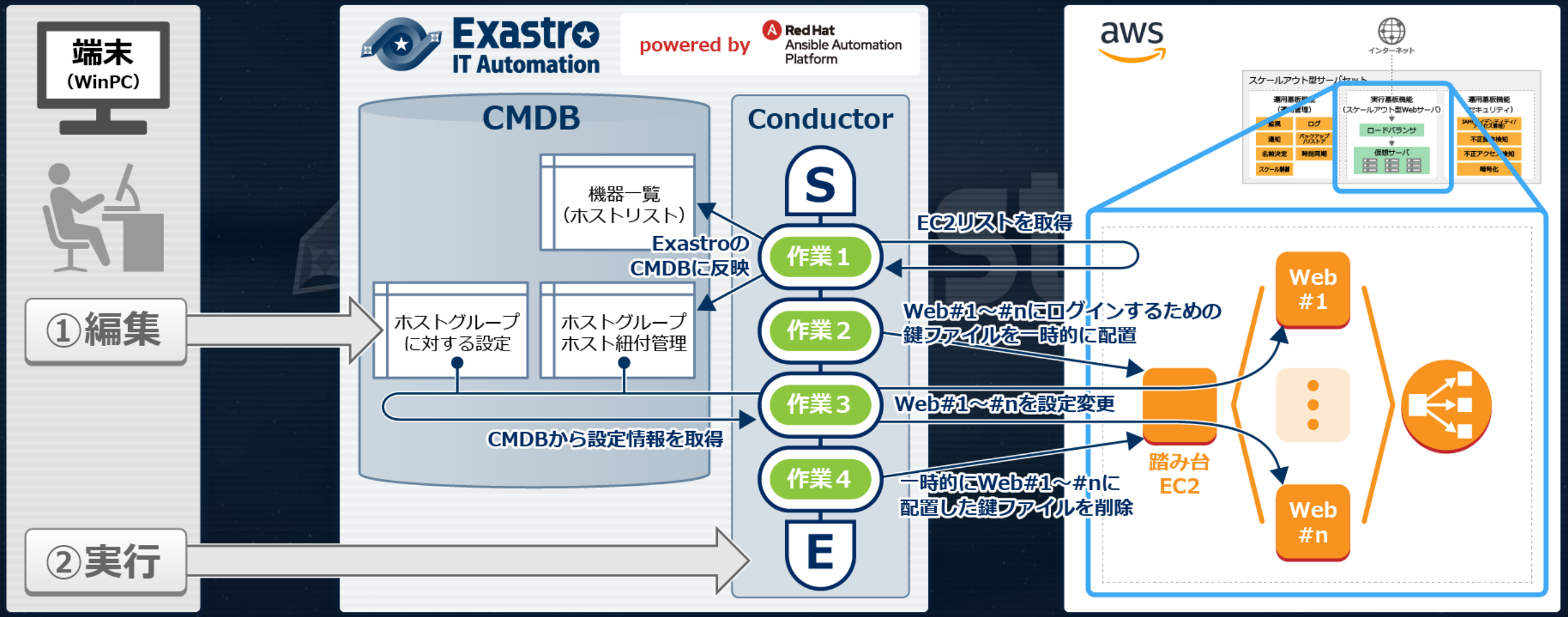
【事例④】企業の運用部門が各部門のAWSシステムを統制する仕組み (1/2)

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




Exastro