

TECH+ スペシャルセミナー

先進企業から学ぶ最新クラウド活用事例

～DXを加速！今始める攻めと守りの自動化2.0～



# DXの必需品 「攻めと守りの自動化」とは

NEC サービス＆プラットフォームSI事業部  
吉田 功一

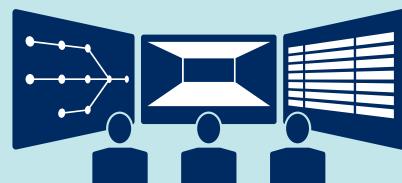
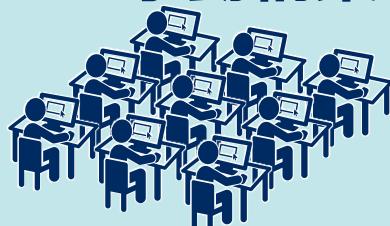
# Introduction 「攻めの自動化」「守りの自動化」とは？

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  
(物理機器)

仮想化技術がまだ普及していなかった頃…

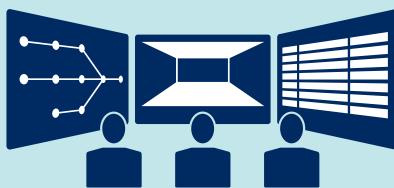
- ✓ サービス停止させないために  
**「最繁トラヒック×バッファ×冗長2倍」**  
のリソースを物理的に並べていた
- ✓ また「CPU、Mem、I/O」の選択肢が少  
なく、**リソースを使いきれなかつた**
- ✓ それでもなるべく少ない機器点数で多くの  
サービスを捌けるように、**複数機能をうま  
く混載(コンソリ)することが“正しい”とさ  
れていた**
- ✓ **アプリケーションもこれらを前提に作られ  
ていた(それが正しかつた)**

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

# 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)



## 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…**

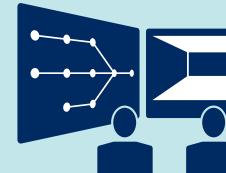
# Introduction 「攻めの自動化」「守りの自動化」とは？

新技術を取り込みに

密結合で保守しづらい

従来のスタティックな

手動構築・手動運用



モノリシック  
(クラウドリフト)

クラウドネイティブに  
切り出す！

- 大変な頑張り…  
✓ アプリを作り直し  
✓ 初期構築時に  
カラクリを仕込む

頻繁に  
改造する  
部位

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

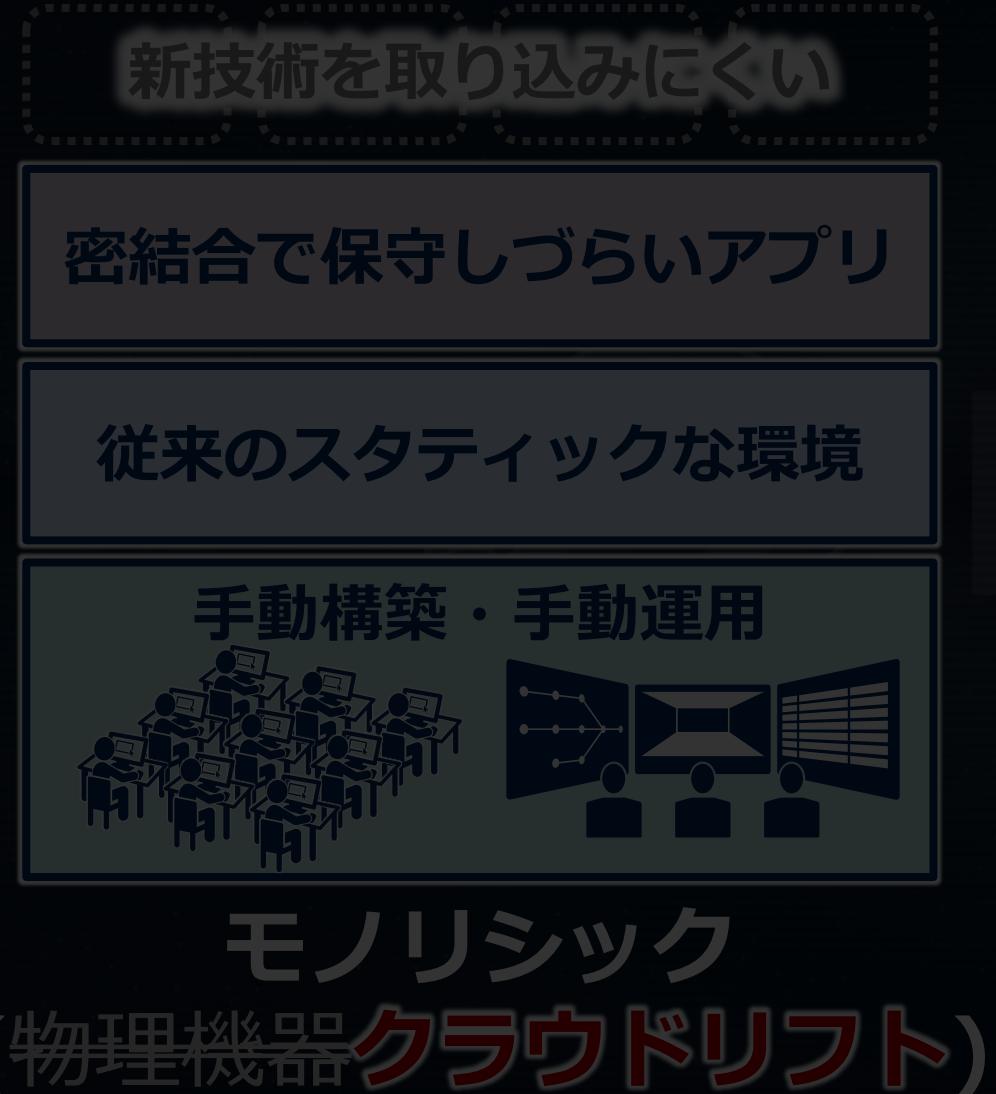
自動構築・自律運用

PF

クラウドネイティブ

OP

# Introduction 「攻めの自動化」「守りの自動化」とは？



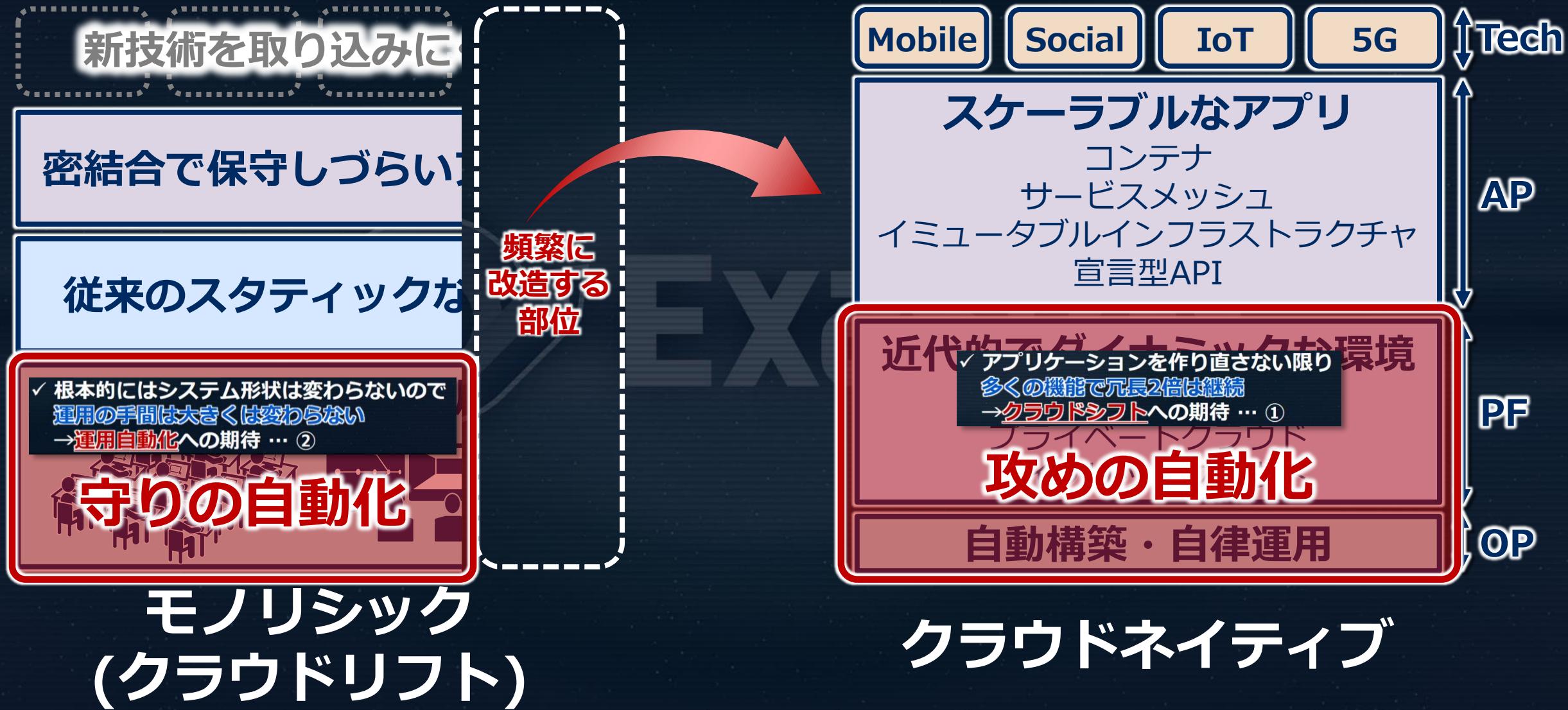
## 仮想化技術という武器を手に入れた結果

- ✓ 「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？



# 「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

11のカラクリ  
(10の技 + 仮想化)



最終的な目的  
ITリソースを活用し  
利益を向上する

### 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 = 値値の向上

高速・スケーラブルな  
機能提供 (リリース)

### OPEXの効率化

OPEX(運用費)を  
下げて投資にまわす

運用に必要な  
人件費の削減

運用に必要な  
保守費の削減

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

オートスケーリング

分散トレーシング

メトリクス監視

メンテナスタスクの一元管理・自動化

仮想化(ハードウェア寿命からの解放)

## 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用手段

### 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

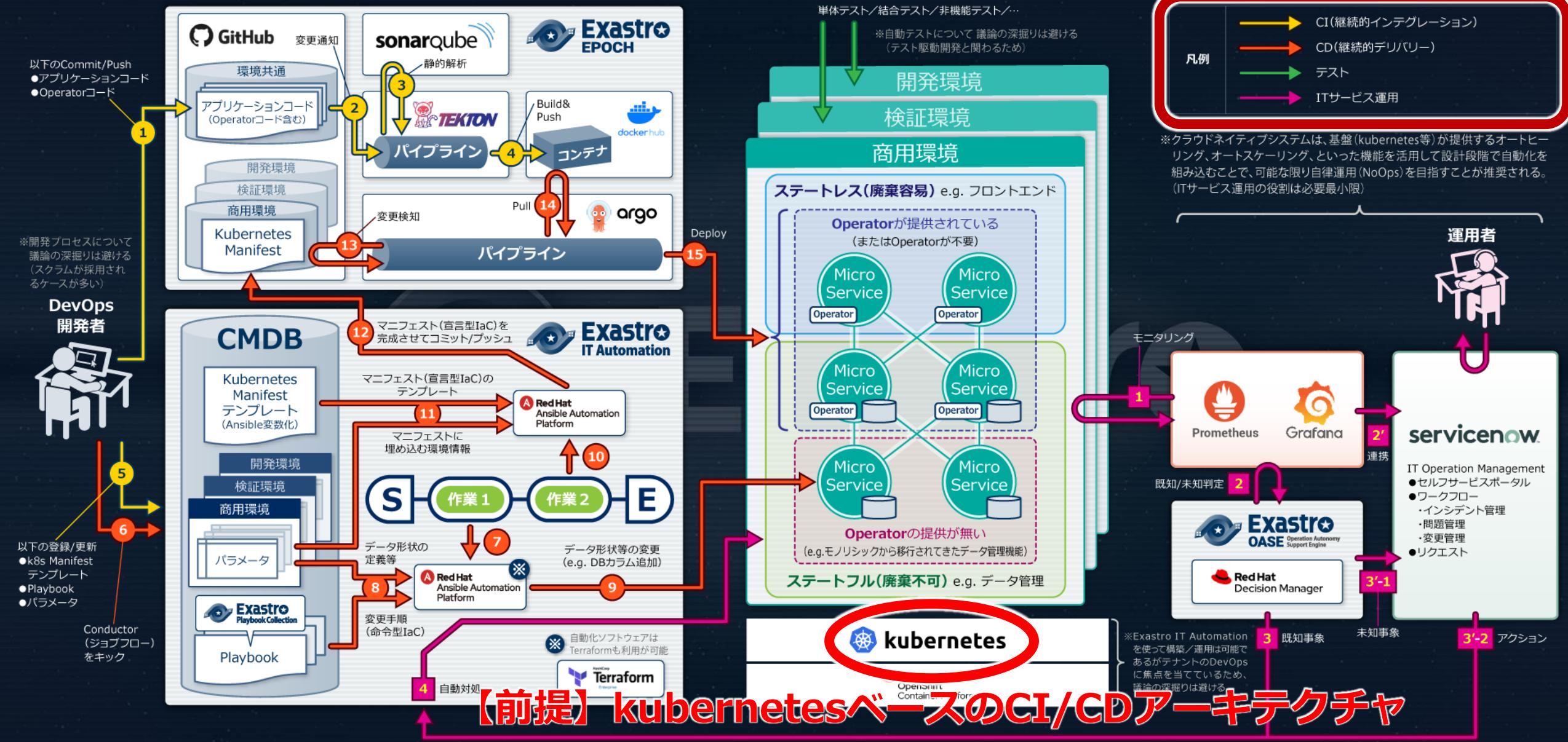
保守費の削減

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)

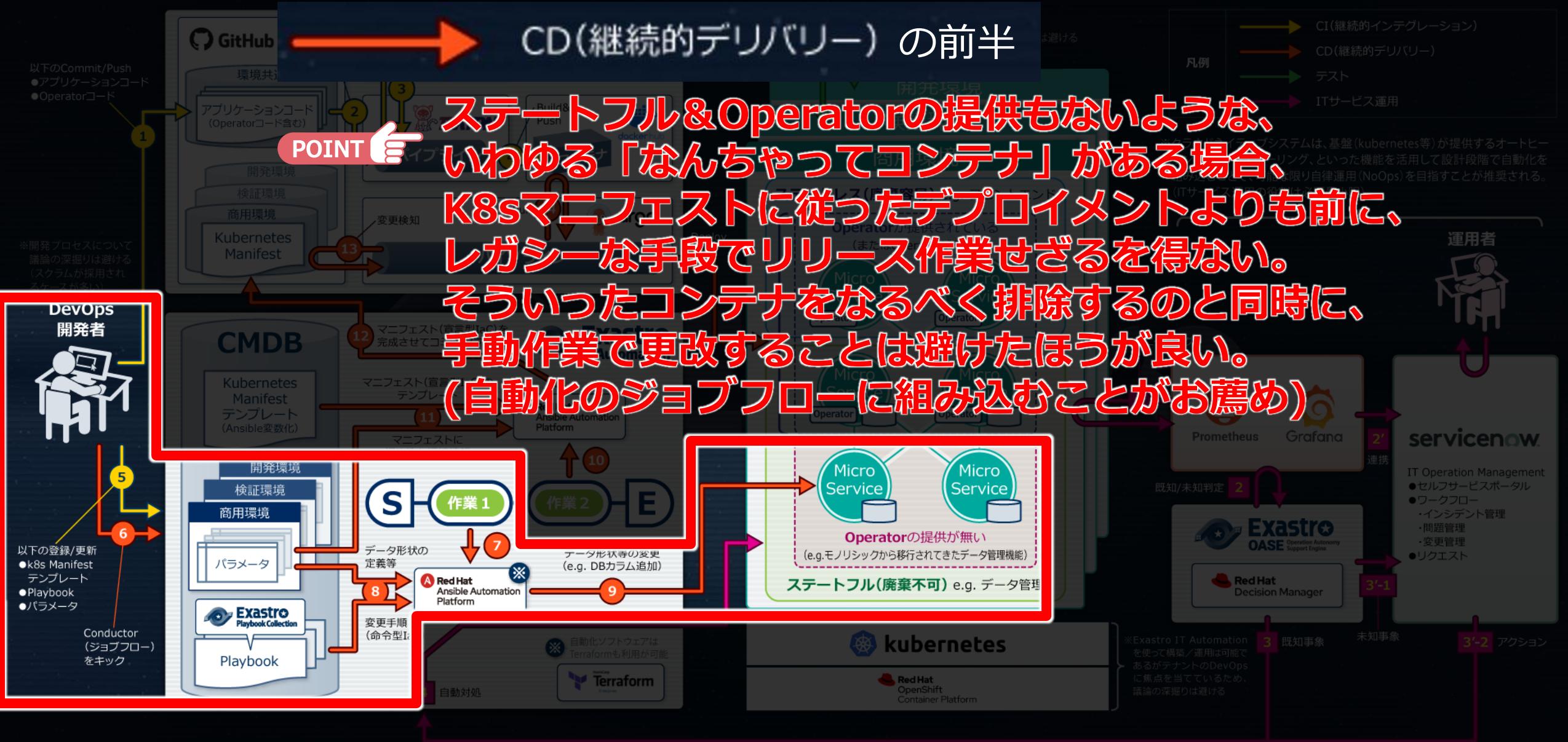
## 「攻めの自動化」の進め方



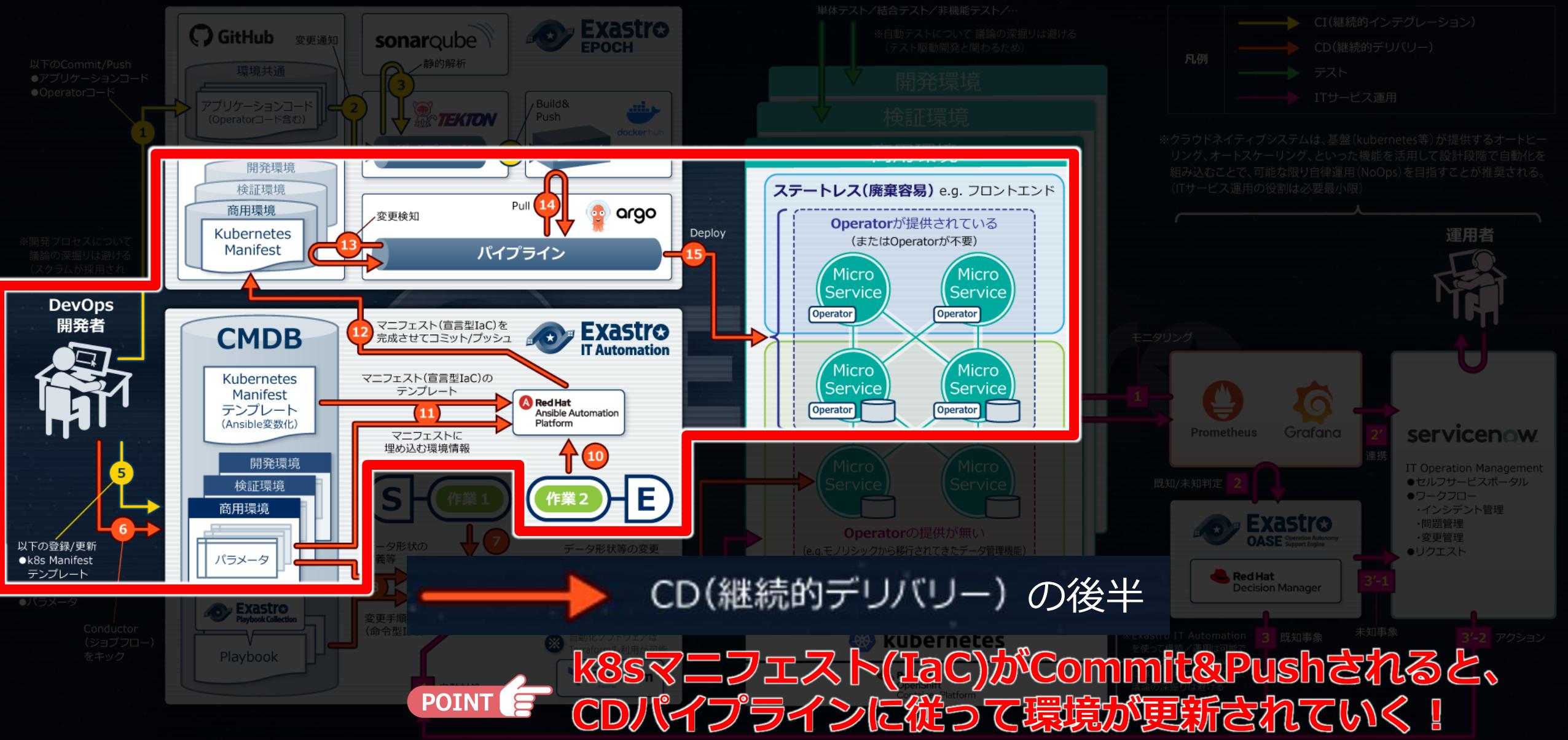
# 「攻めの自動化」の進め方



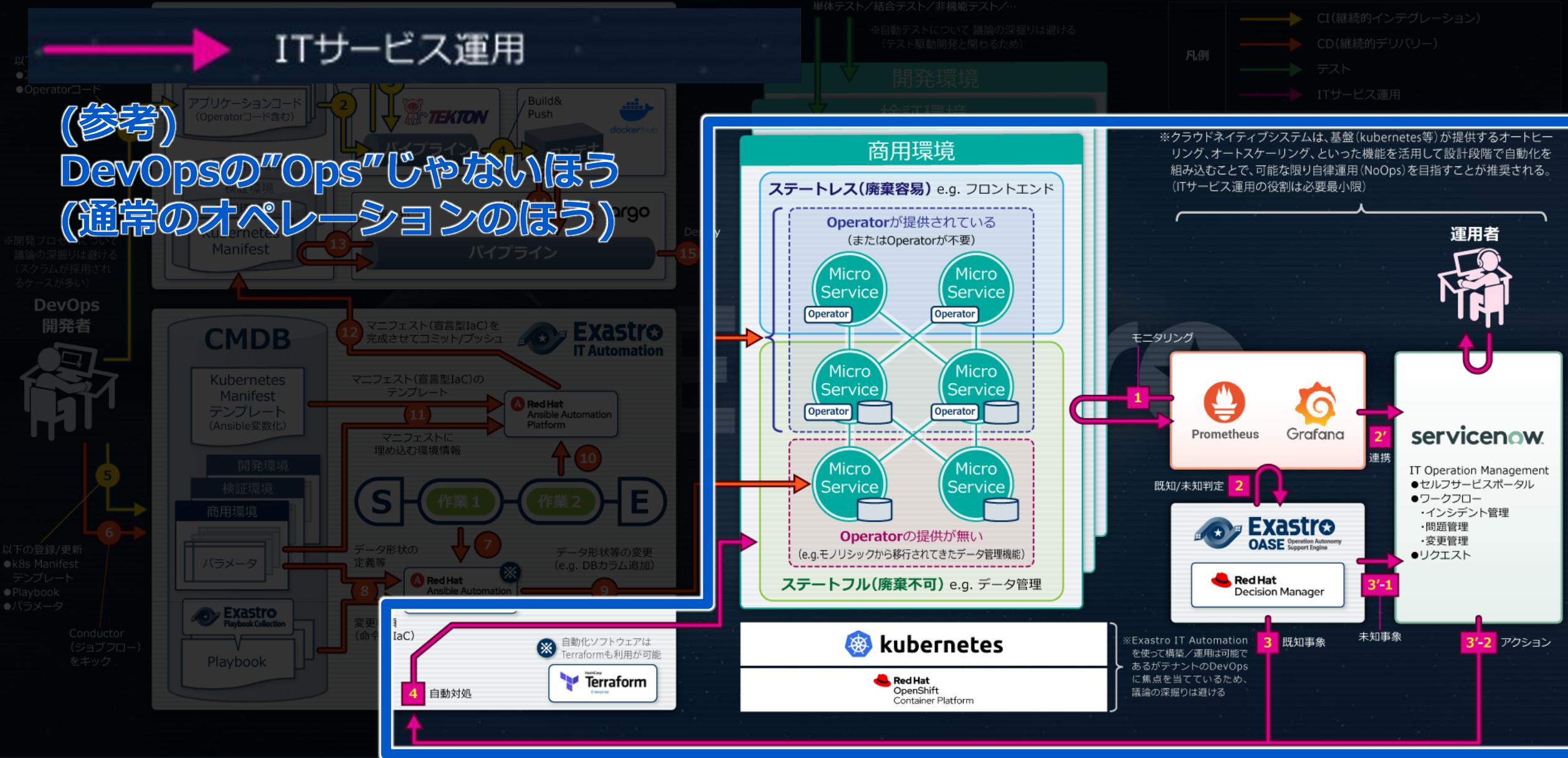
## 「攻めの自動化」の進め方



# 「攻めの自動化」の進め方



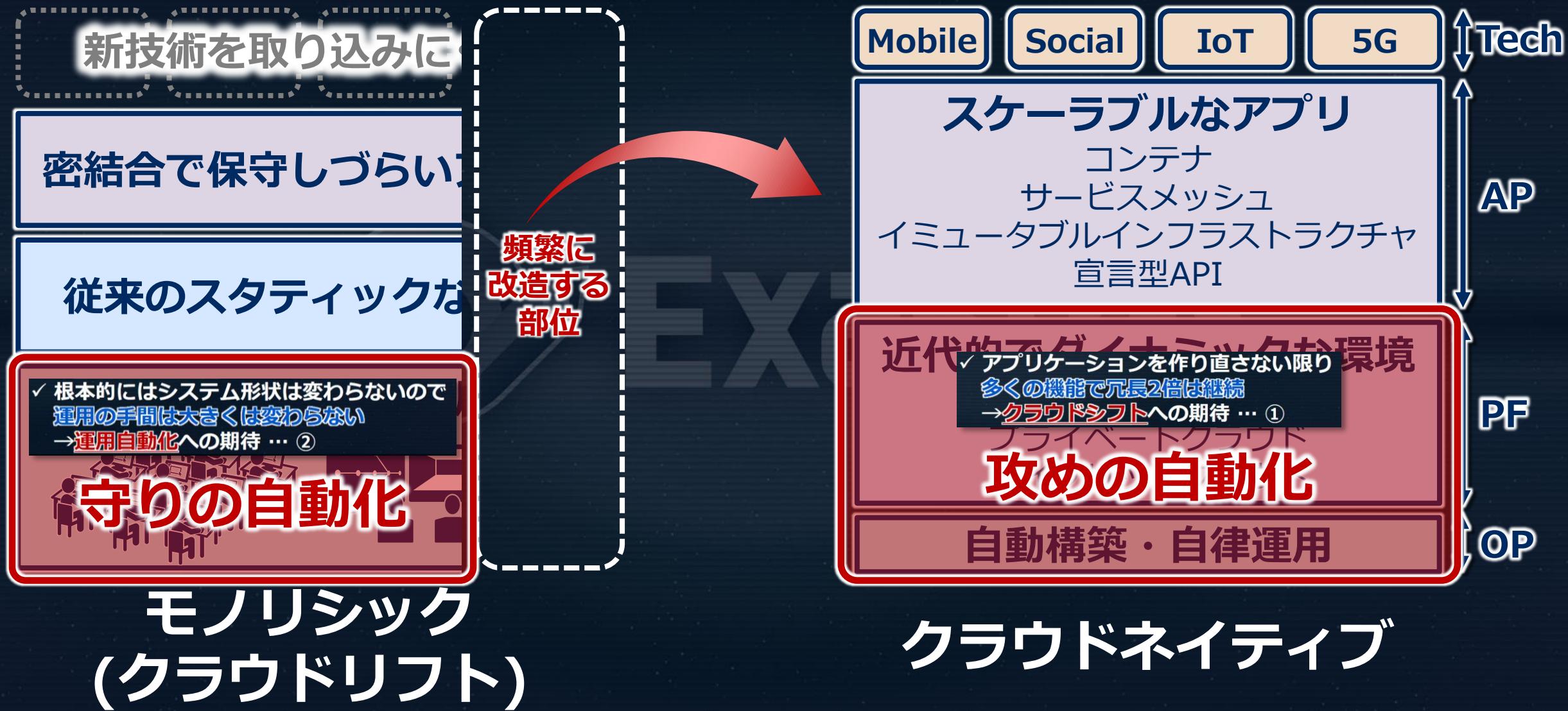
# 「攻めの自動化」の進め方



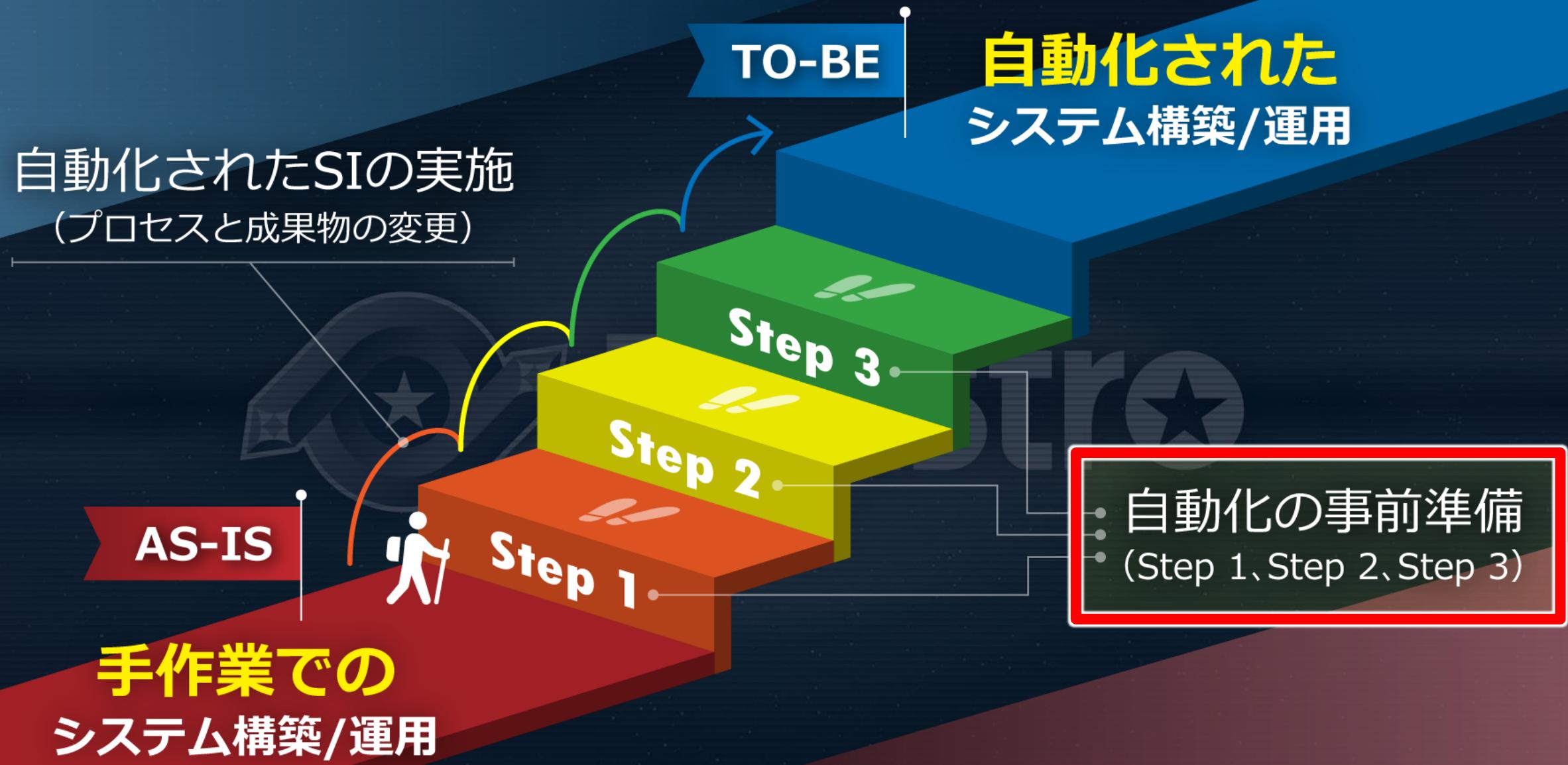
# 「守りの自動化」の進め方



# Introduction 「攻めの自動化」「守りの自動化」とは？



# 「守りの自動化」の進め方



## モノリシックシステムの構築・運用に携わるITエンジニアの現場の声

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

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



設計



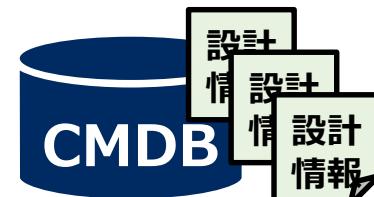
作業準備



作業実施

✓ チーム間の  
情報共有  
結果として  
チーム間の  
作業ごとに  
手順ごとに  
手順ごとに  
人手作業が  
⇒チーム間  
人手作業が  
  
解決策

## Step 1 設計情報の一元管理

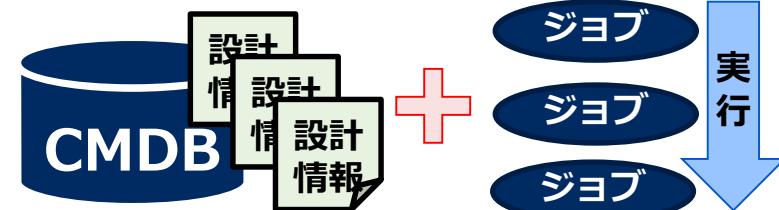


✓ チーム間の  
情報共有  
結果として  
チーム間の  
作業ごとに  
手順ごとに  
手順ごとに  
人手作業が  
⇒チーム間  
人手作業が  
  
解決策

## Step 2 自動実行の実現



## 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

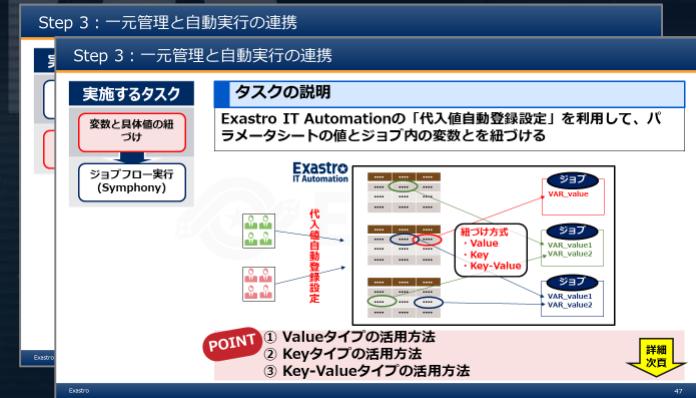
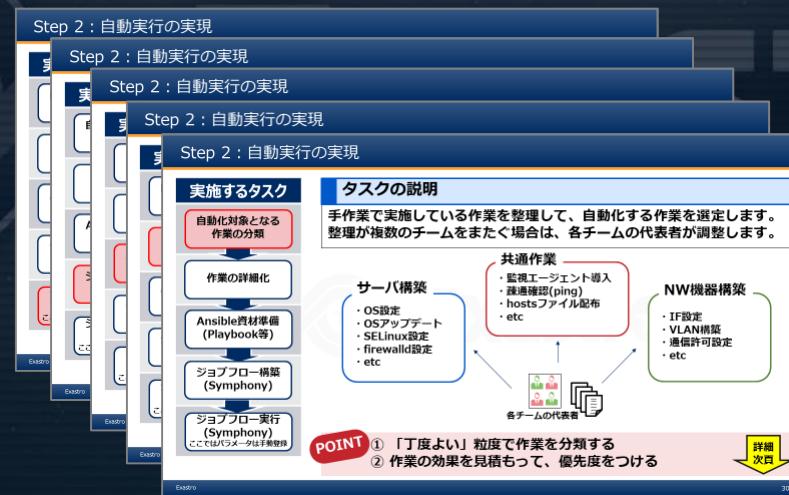
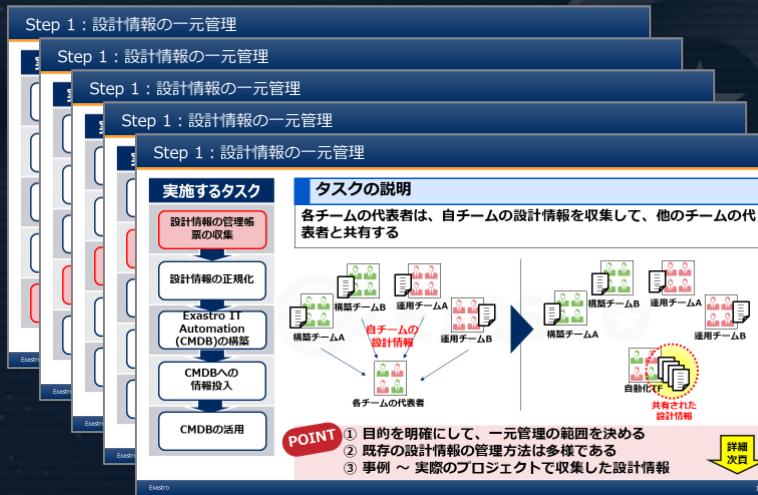
### Step 1



### Step 2



### Step 3

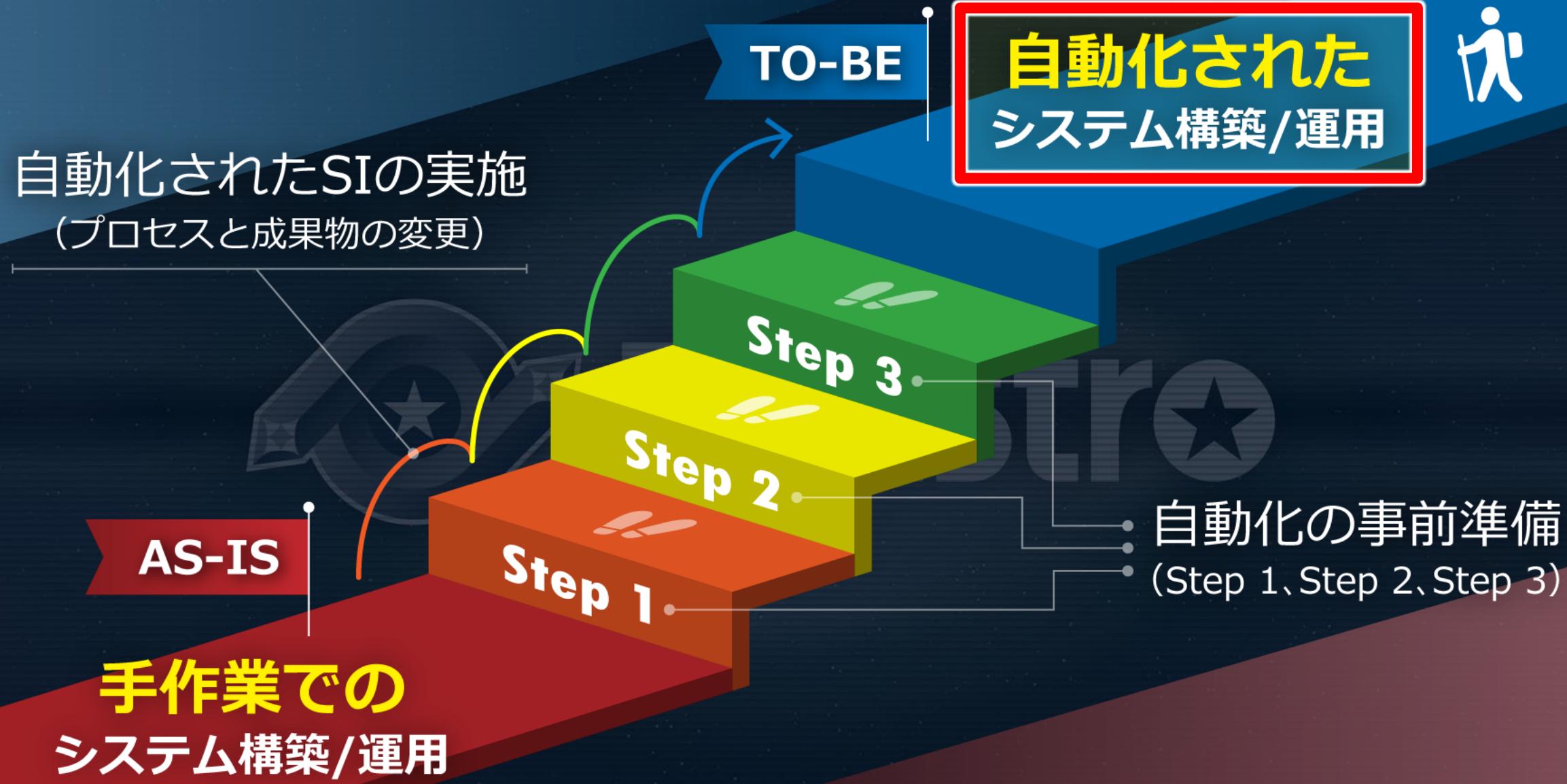


5つのタスク

5つのタスク

2つのタスク

# 「守りの自動化」の進め方



# 「守りの自動化」の進め方

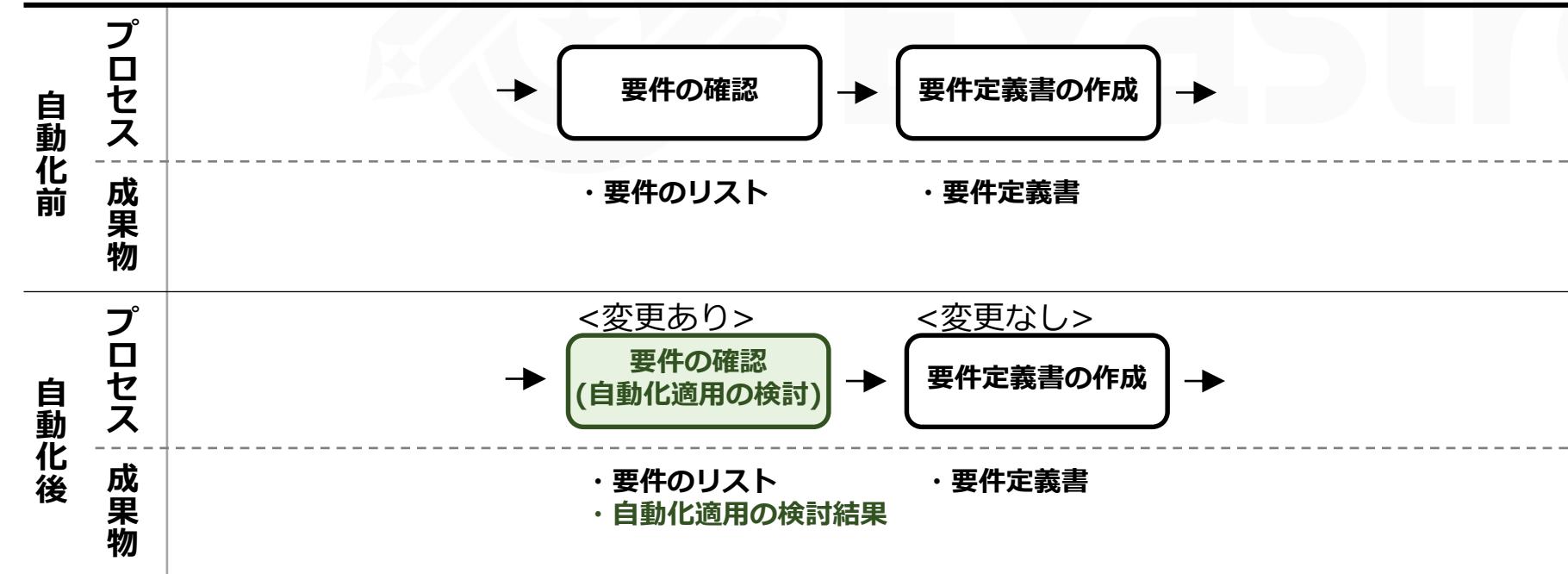
## フェーズにおけるQCDの変化

凡例： ☺ 変化なし ☺ 改善 ☺ 追加の検討項目あり

	要件定義	基本設計	詳細設計	運用設計	製造	テスト	リリース
	Q C D	Q C D	Q C D	Q C D	Q C D	Q C D	Q C D
自動化前	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺
自動化後	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺

## プロセスと成果物の変化

凡例： 変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



## 解説

要件の確認の段階で、自動化の適用範囲等について検討を行い、合意を取る必要がある。その検討分としてCとDが増加する。

# 「守りの自動化」の進め方

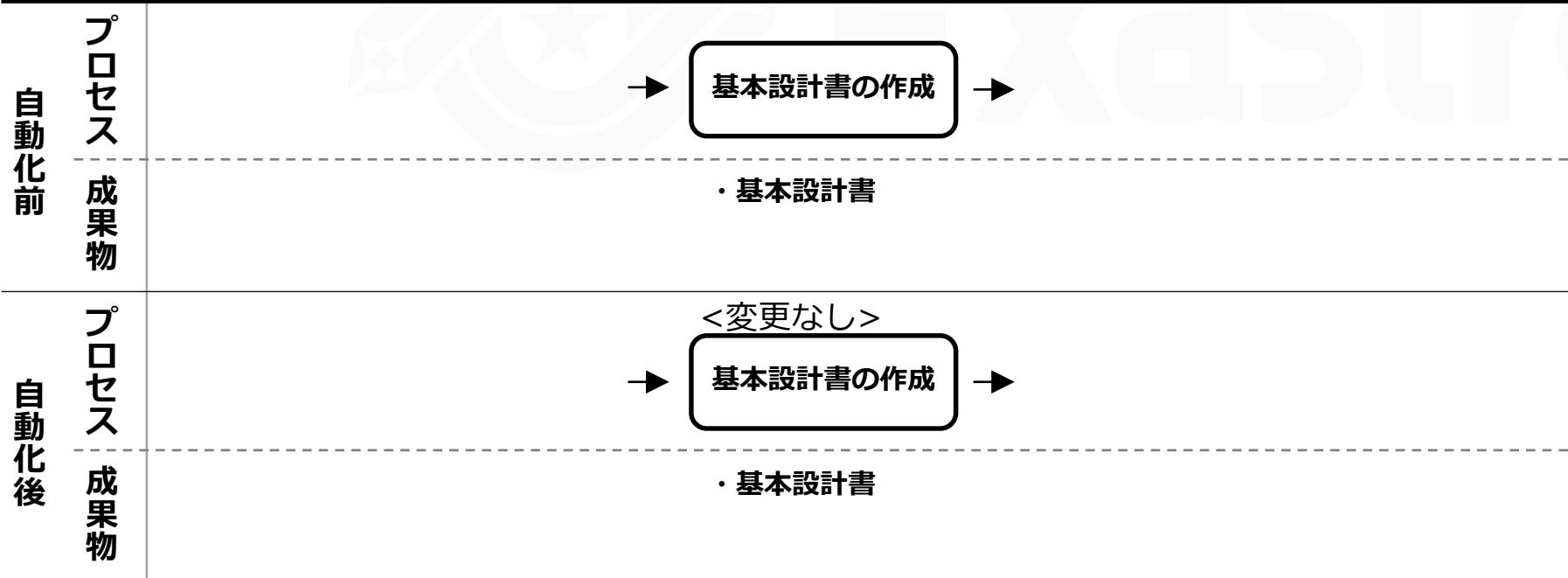
## フェーズにおけるQCDの変化

凡例： ☺ 変化なし ☺ 改善 ☺ 追加の検討項目あり

	要件定義			基本設計			詳細設計			運用設計			製造			テスト			リリース		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
自動化前	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
自動化後	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺

## プロセスと成果物の変化

凡例： 変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



## 解説

基本設計に取り込むべき内容は、自動化の事前準備の段階で検討済みであるため、ここでは追加の作業はない。

# 「守りの自動化」の進め方

## フェーズにおけるQCDの変化

	要件定義			基本設計			詳細設計			運用設計			製造			テスト			リリース		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
自動化前	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊
自動化後	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊

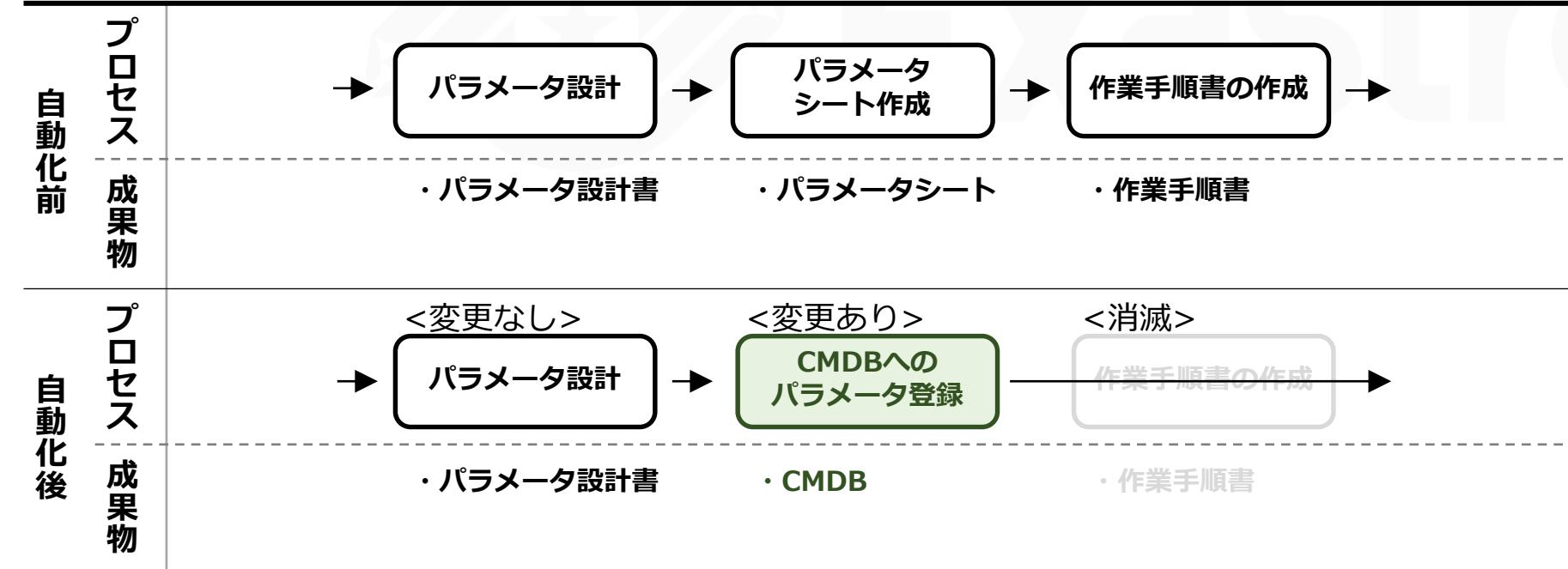
凡例： 😊 変化なし 😃 改善 🧑 追加の検討項目あり

## 解説

パラメータ設計で作成された各パラメータは、CMDBに登録していく。これによりパラメータの形式化が進み、表記ゆれや曖昧さが提言するため、Qが改善する。

## プロセスと成果物の変化

凡例： 変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



また、パラメータの適用順序などの作業手順は、自動化の事前準備の段階で作成したジョブフローに置き換わるため、作業手順書の作成のタスクが消滅する。これにより、CとDも改善する。

# 「守りの自動化」の進め方

## フェーズにおけるQCDの変化

	要件定義	基本設計	詳細設計	運用設計	製造	テスト	リリース
	Q C D	Q C D	Q C D	Q C D	Q C D	Q C D	Q C D
自動化前	😊😊😊	😊😊😊	😊😊😊	😊😊😊	😊😊😊	😊😊😊	😊😊😊
自動化後	😊	😊	😊	😊	😊	😊	😊

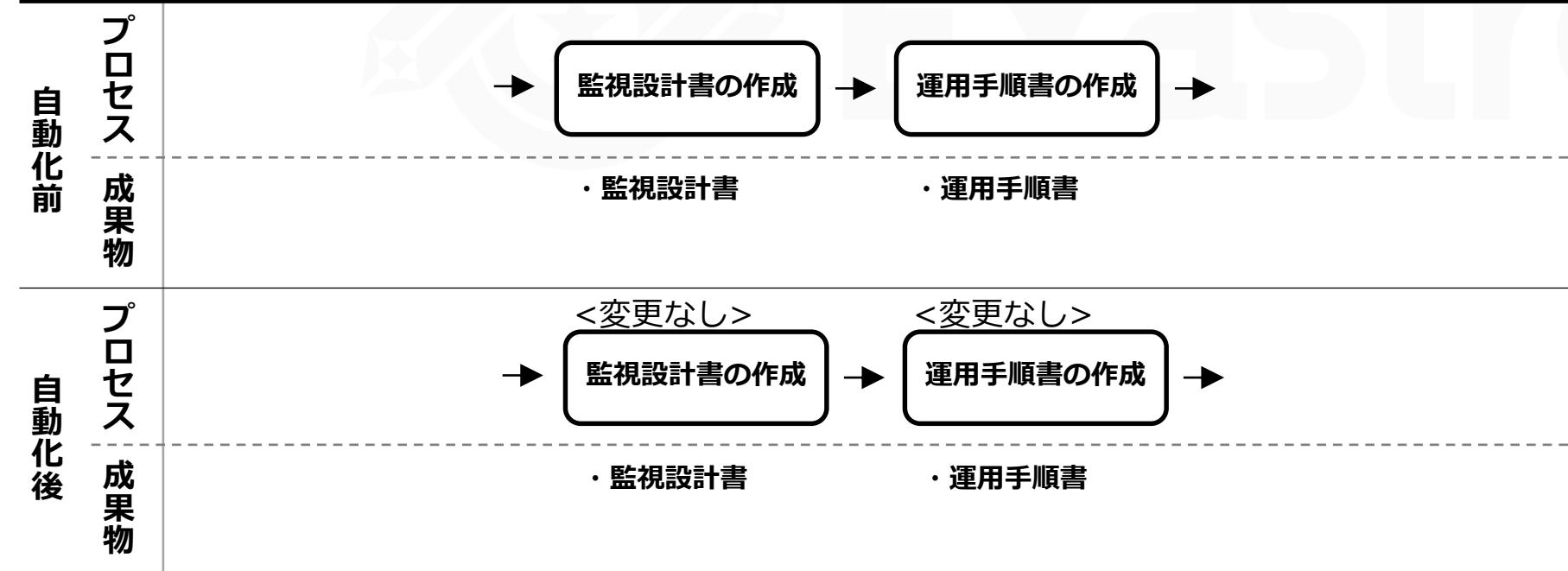
凡例： 😊 変化なし 😃 改善 🤖 追加の検討項目あり

## 解説

本書は構築の自動化に焦点を当てているため、運用の自動化は範囲外とする。

## プロセスと成果物の変化

凡例： 変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



運用の自動化を実施する場合は、QCDおよびプロセスの変化が発生する。

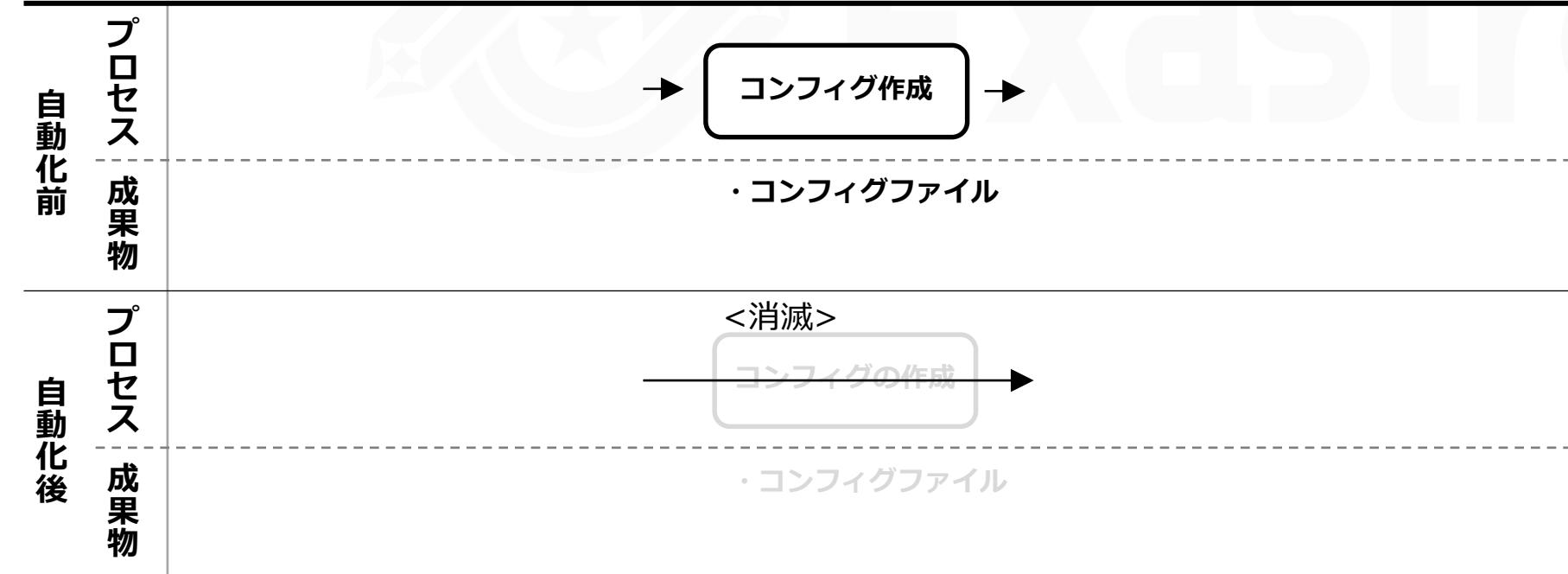
# 「守りの自動化」の進め方

## フェーズにおけるQCDの変化

	要件定義	基本設計	詳細設計	運用設計	製造	テスト	リリース								
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
自動化前	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊
自動化後	😊	😢	😢	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊

## プロセスと成果物の変化

凡例：変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



## 解説

詳細設計に基づき作成されるコンフィグファイルは、IaCとCMDBにより自動生成されるため、コンフィグ作成のタスクは消滅する。

# 「守りの自動化」の進め方

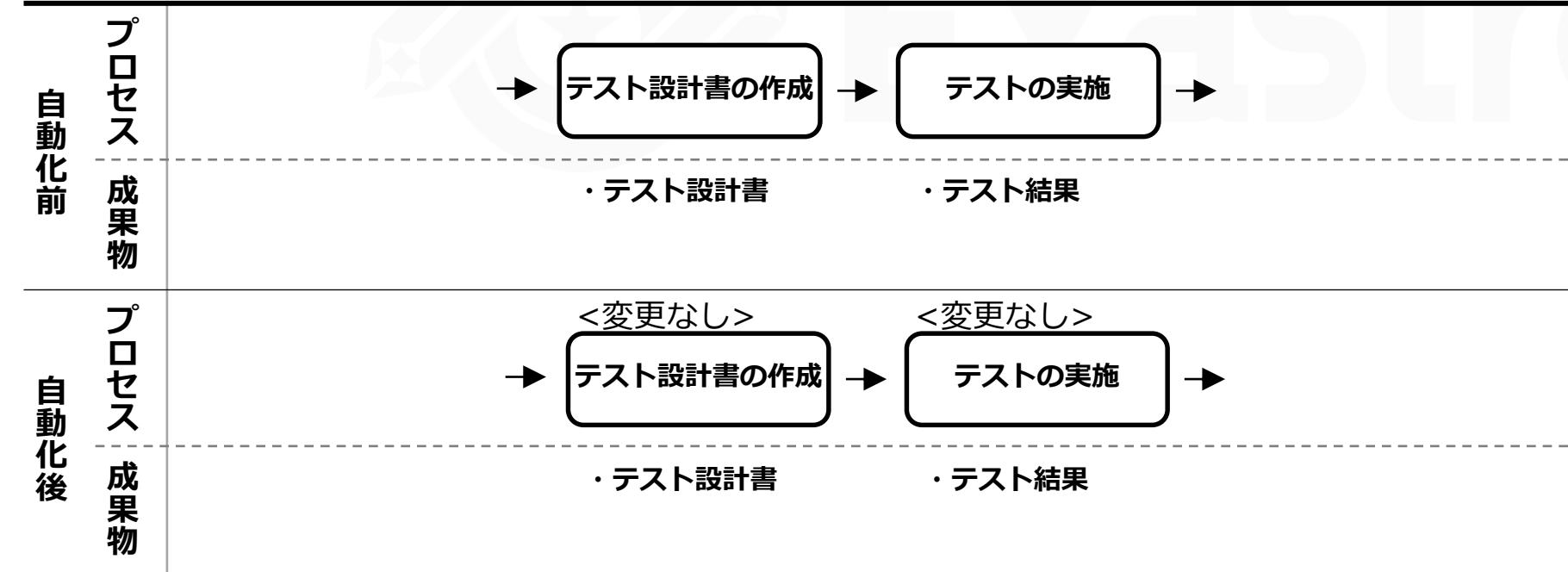
## フェーズにおけるQCDの変化

凡例： ☺ 変化なし ☺ 改善 ☺ 追加の検討項目あり

	要件定義			基本設計			詳細設計			運用設計			製造			テスト			リリース		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
自動化前	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
自動化後	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺

## プロセスと成果物の変化

凡例： 変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



## 解説

本書は構築の自動化に焦点を当てているため、テスト自動化は範囲外とする。

テストの自動化を実施する場合は、QCDおよびプロセスの変化が発生する。

# 「守りの自動化」の進め方

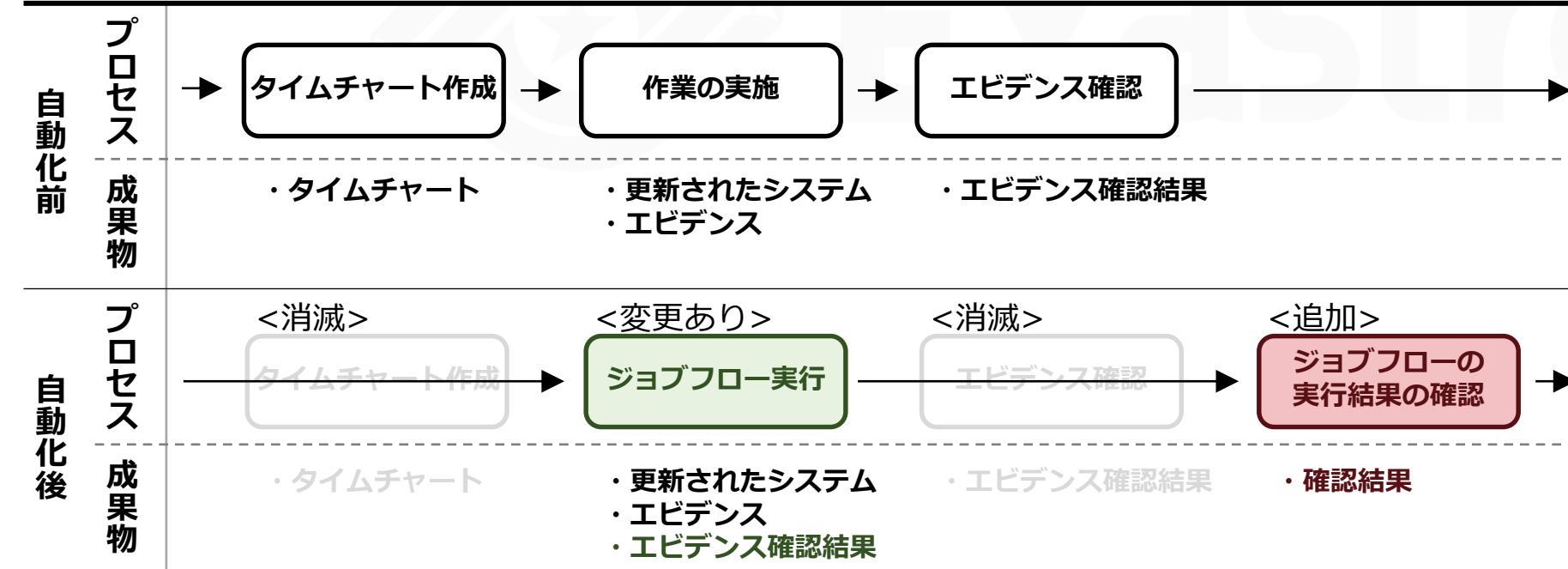
## フェーズにおけるQCDの変化

凡例： ☺ 変化なし ☺ 改善 ☺ 追加の検討項目あり

	要件定義			基本設計			詳細設計			運用設計			製造			テスト			リリース		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
自動化前	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
自動化後	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺

## プロセスと成果物の変化

凡例： 変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



## 解説

詳細設計で作成したジョブフローの実行がメインの作業になる。

タイムチャート作成がジョブフロー実行に置き換わるため消滅。

またジョブフロー中でエビデンスの確認も実施するため、タスクとしてのエビデンス確認も消滅。

よって、ジョブフローの実行と結果の確認のみとなり、QCDとともに改善する

# 「守りの自動化」の進め方

## 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)

The screenshot shows the 'Learn' section of the Exastro IT Automation website. A yellow circle labeled '1' highlights the 'Learn' button in the top navigation bar. A yellow circle labeled '2' highlights the 'Guidebook' link in a dropdown menu. A yellow circle labeled '3' highlights the 'Download PDF' button on the right side of the main content area. To the right, a large QR code is displayed, with arrows pointing from the 'Learn' button and the 'Download PDF' button towards it.

Exastro IT Automation

Learn

Webinar Case Documents Downloads Setting samples FAQ

提供機能 デプロイ例 インストール手順 クイックスタート BASE 機能別

システム構築・運用の効率化ガイドブック

Exastro IT Automation

View PDF

スライドをダウンロード (PDF)

ガイドブック

システム構築・運用の効率化

Learn

ガイドブック

システム構築・運用の効率化

スライドをダウンロード (PDF)

Exastro

Exastro is an open source software suite for digitizing, automating and labor saving the system life cycle.

Why Exastro Exastro Suite Overview News Room Event Asset

IT Automation

Overview Learn Webinar Case Documents Downloads

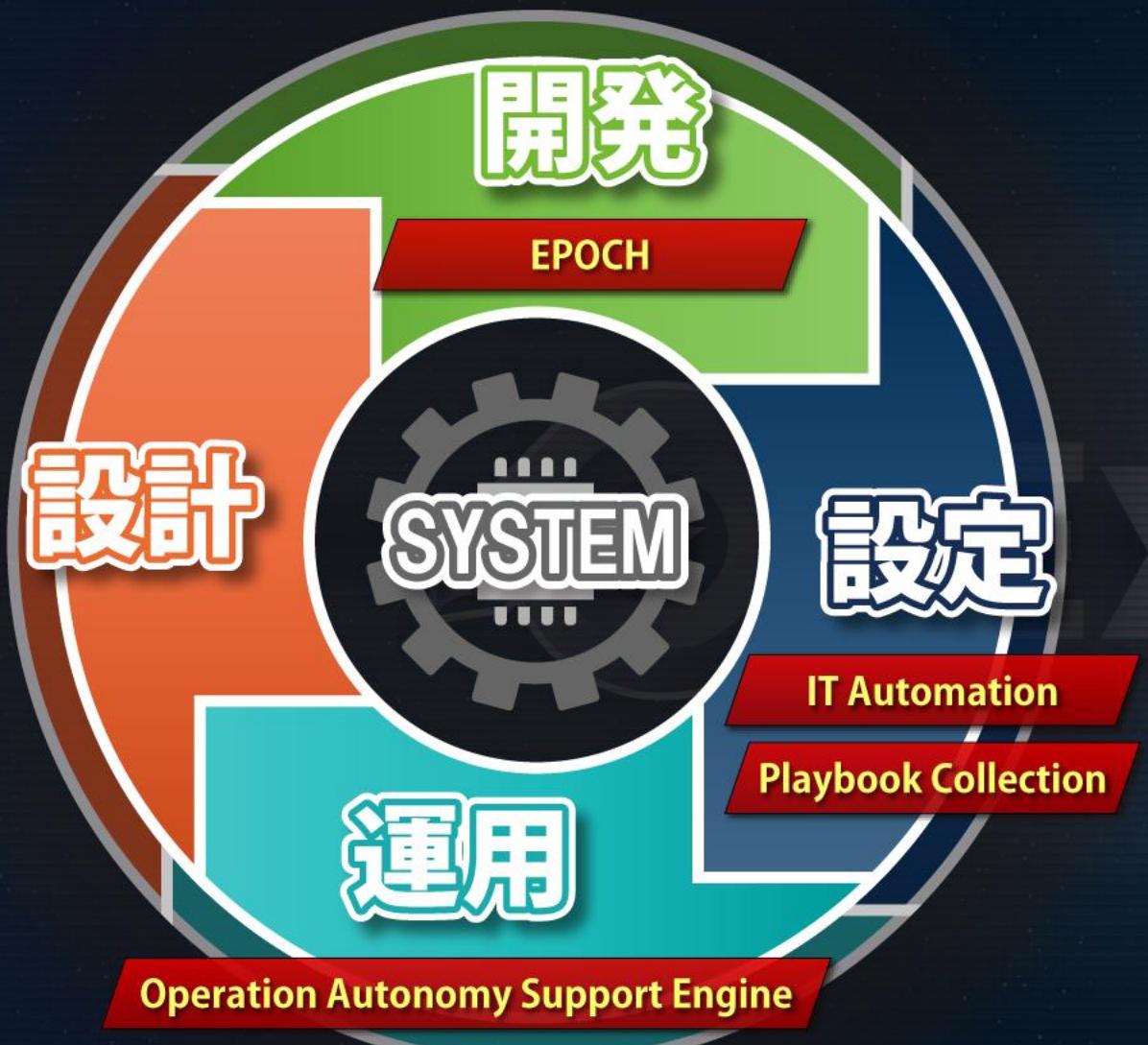
INPUT/RESET SELECT - +

# Exastro Suite

## - 「攻めと守りの自動化」の実現手段 -

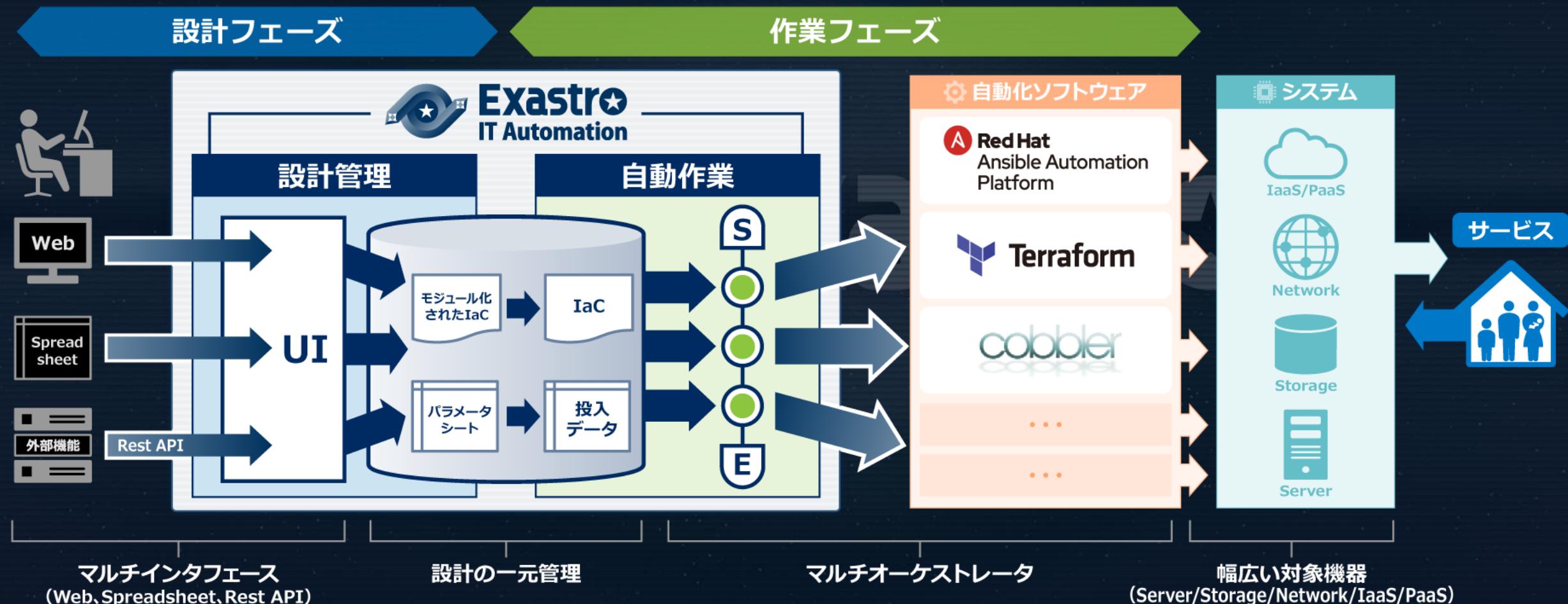


# Exastro Suiteとは？

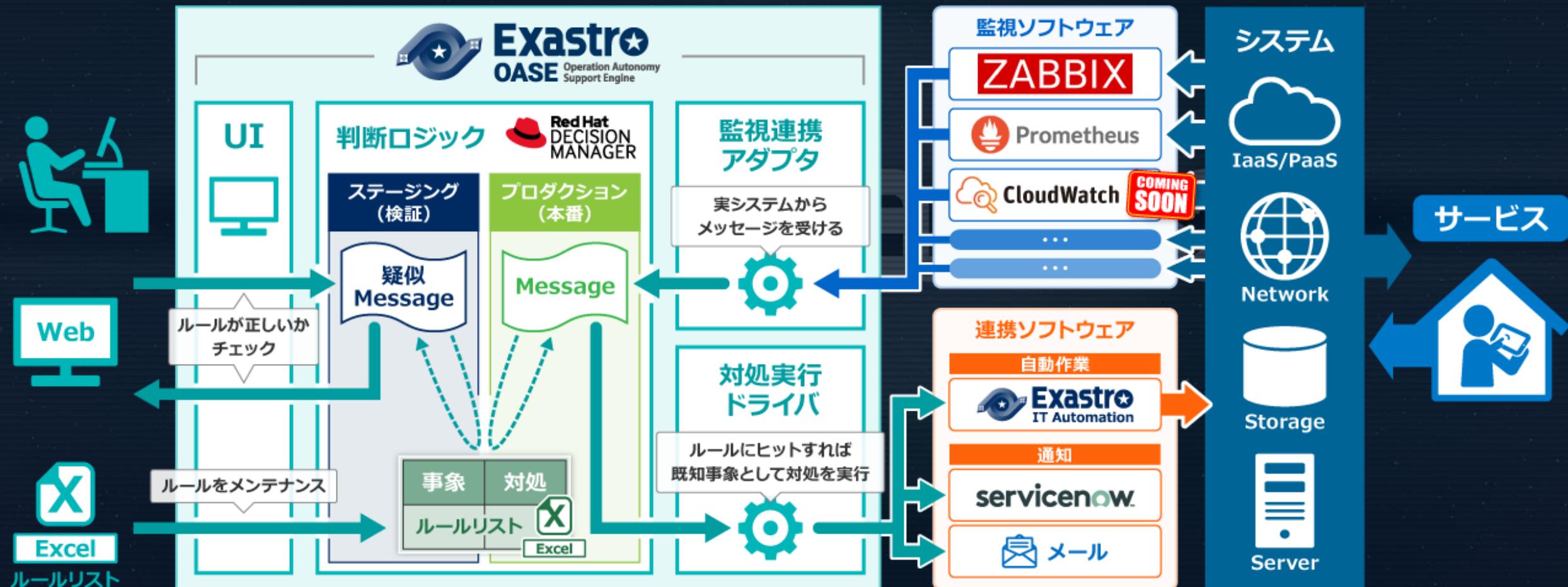


Exastro は システムライフサイクル  
を  
デジタル化・自動化・省力化することを目的とした  
オープンソースのソフトウェアスイートです。

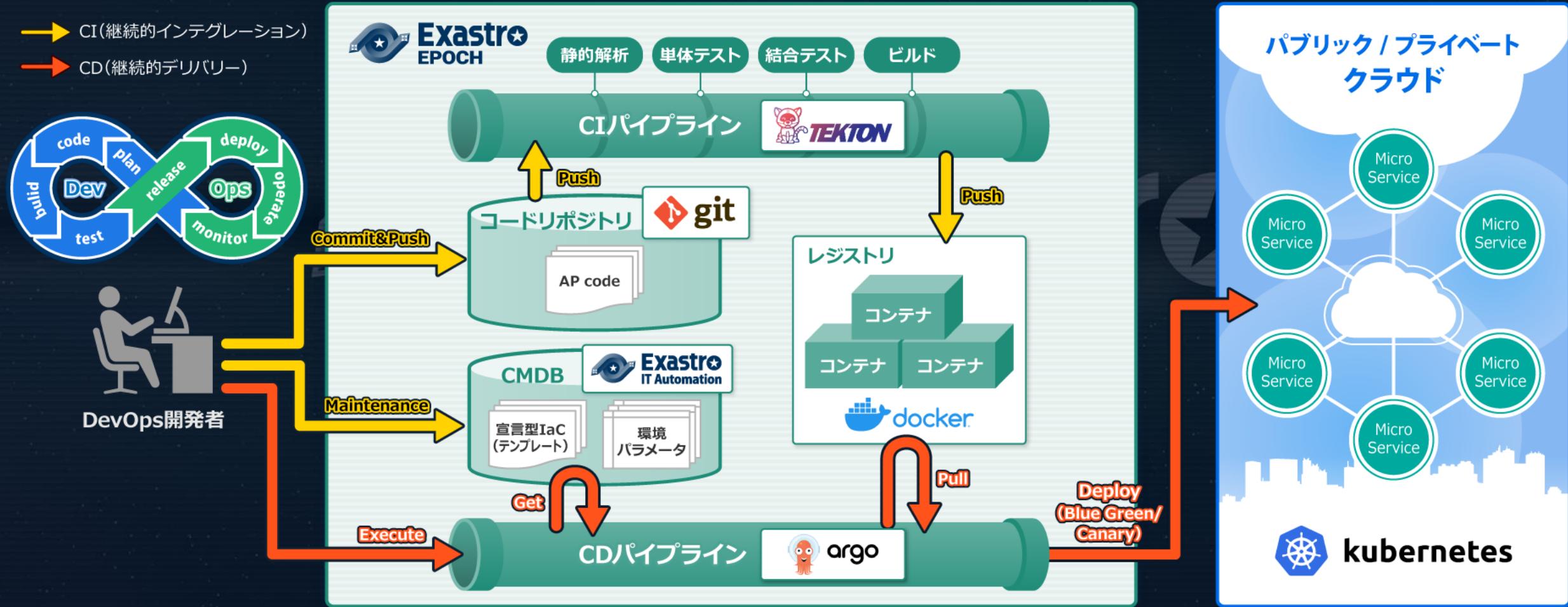
## Exastro IT Automationは 「システム構成(IaC+パラメータ)を管理するためのフレームワーク」です



## Exastro 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



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



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)



お問い合わせはコチラへ

[contact@exastro.jp.nec.com](mailto:contact@exastro.jp.nec.com)

- クラウドネイティブのデモをお見せできます。
- ガイドブックを詳しく解説できます。

お問い合わせいただければと存じます





**Exastro** 