

## UrbanCode DeployによるWASデプロイの自動化

## 目次

---

1. リリース・デプロイ作業のよくある課題
2. 課題解決には何が必要か？
3. UrbanCodeで課題を解決！

当資料内で使用する略語は以下の通りです。

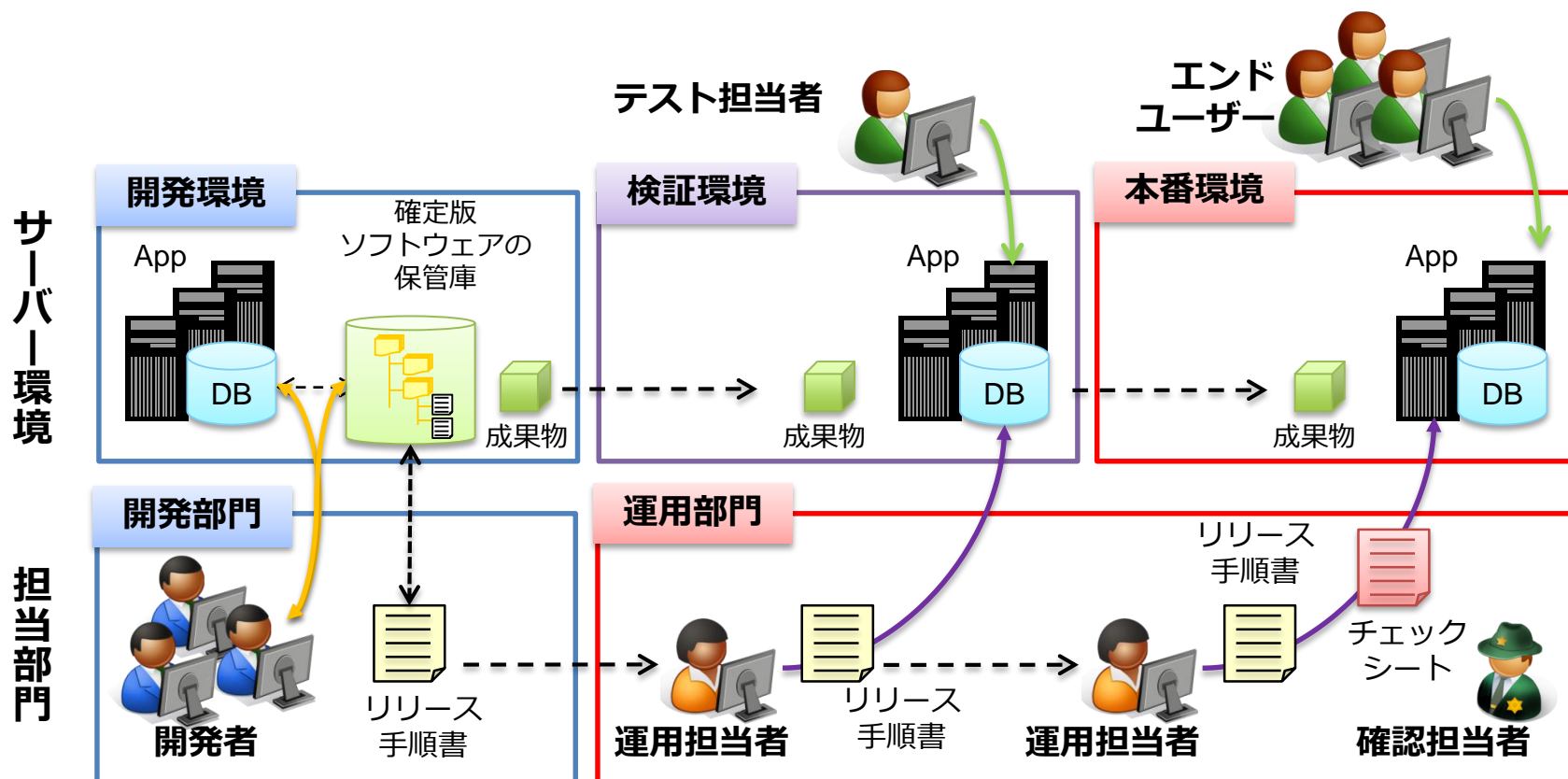
- WAS : WebSphere Application Server
- UCD : UrbanCode Deploy

## 1. リリース・デプロイ作業のよくある課題

---

# 「リリース・デプロイ」作業 全体像

複数のシステム環境、役割が異なる複数の担当者の協働によって、リリース・デプロイは成立します。



# 「リリースとデプロイ」作業でよくある6つの課題



## 「リリースとデプロイ」作業でよくある6つの課題

その1

担当者依存のリリース作業

その2

難解なリリース手順書

その3

成果物引き渡し時のミス

その4

リリース作業の増加/複雑化

その5

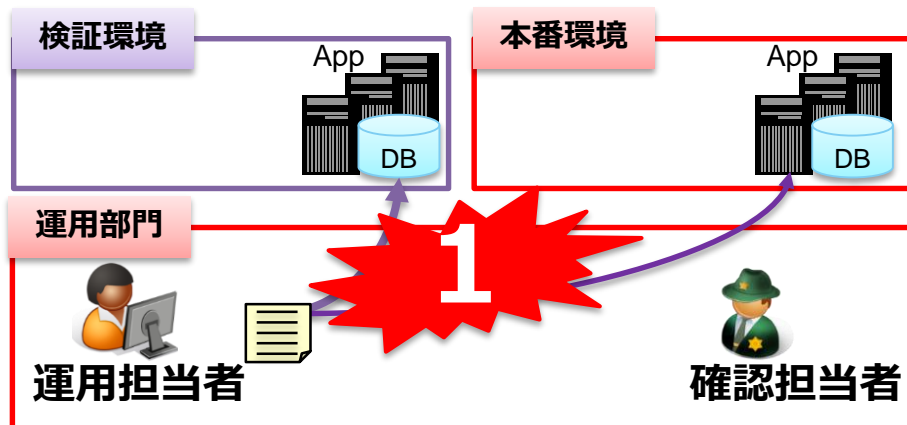
コンプライアンス / セキュリティ

その6

開発と運用、それぞれの思い

## その1

## 担当者依存のリリース作業



## 課題

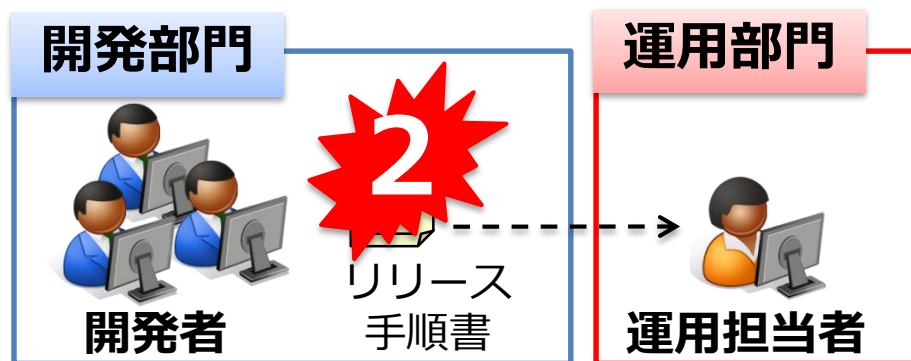
- 週末、月末等に作業が偏る
- 複数人での確認、検証に時間、コストがかかる
- 作業の準備・確認時間が短い
- 特定の担当者がいないと作業ができない

## 原因

- ✓ 特定の熟練した運用担当者への作業依存
- ✓ 熟練した運用担当者でも間違いやすい複雑な手順
- ✓ 締切直前にならないと開発部門から手順書が出てこない

## その2

## 難解なリリース手順書



## 課題

- 開発部門から提示される  
手順書の記載内容がバラバラ
- リリース手順書の修正・変更が、  
適宜反映されていない
- エラー時の対応方法が記載してない

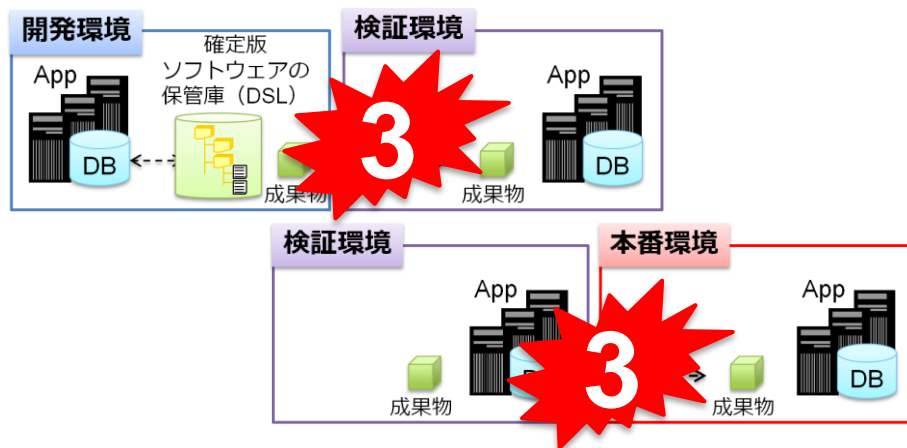
## 原因

- ✓ 運用部門の作業を意識した  
手順書が作成されない
- ✓ リリース手順書が  
標準化されていない
- ✓ リリース手順書の  
作成・修正プロセスがない
- ✓ 何かあれば連絡があるだろう  
という開発部門の意識



## その3

## 成果物引き渡し時のミス



## 課題

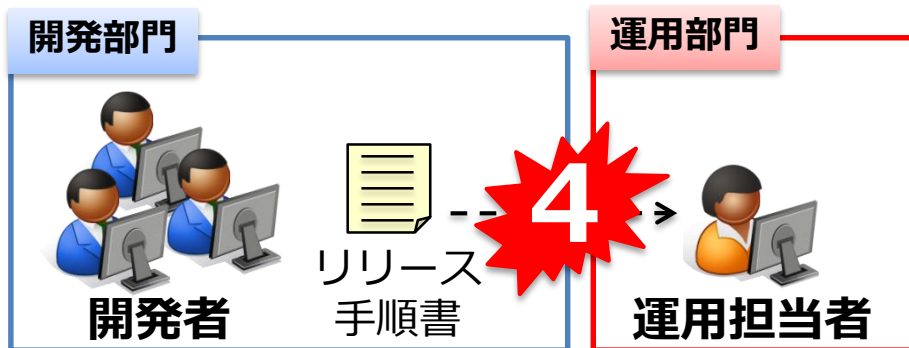
- 開発部門が古いバージョンの成果物を運用部門に引き渡してしまう
- 検証前の成果物を検証環境から本番環境に引き渡してしまう

## 原因

- ✓ そもそも成果物が一元管理されていない
- ✓ 人手・目視での作業により誤った成果物を選択しやすい
- ✓ 引き渡す成果物の状態（テスト前、テスト済み）が確認しにくい

## その4

## リリース作業の増加/複雑化



## 課題

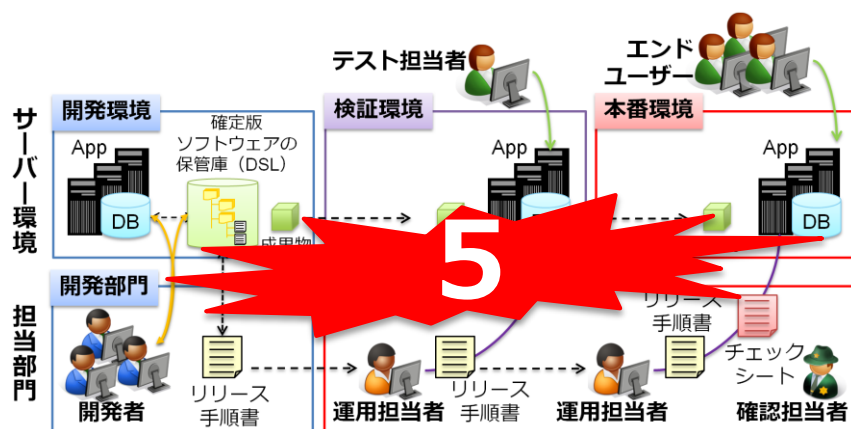
- 数年前に比べ、リリース依頼が増加している
- リリースサイクルが短くなっている
- リリース作業が複雑になっている

## 原因

- ✓ クラウド化によるリリース対象の増加
- ✓ 対応環境の複雑化（OS、モバイル、ブラウザ）
- ✓ 新たな開発手法や開発プロセスの導入によりさらに頻繁なリリースが求められている

## その5

## コンプライアンス / セキュリティ



## 課題

- 作業状況/結果がすぐに確認できない
- いつ、どこで、誰が、何の、作業をしたか、確認できない
- リリース前の承認に時間がかかる
- 特権IDを持つと何でもできてしまう

## 原因

- ✓ 作業状況の更新が作業終了後に実施している
- ✓ 作業ログが個別の仕組みで管理されてしまっている
- ✓ 紙での承認・捺印を回すのに時間がかかる
- ✓ 作業範囲、内容の制御ができない

## その6

## 開発と運用、それぞれの想い



## 課題

- 開発部門の想い：  
できたらすぐでもリリースしたい
- 運用部門の想い：  
手順をきちんと確認、検証してから  
リリースしたい

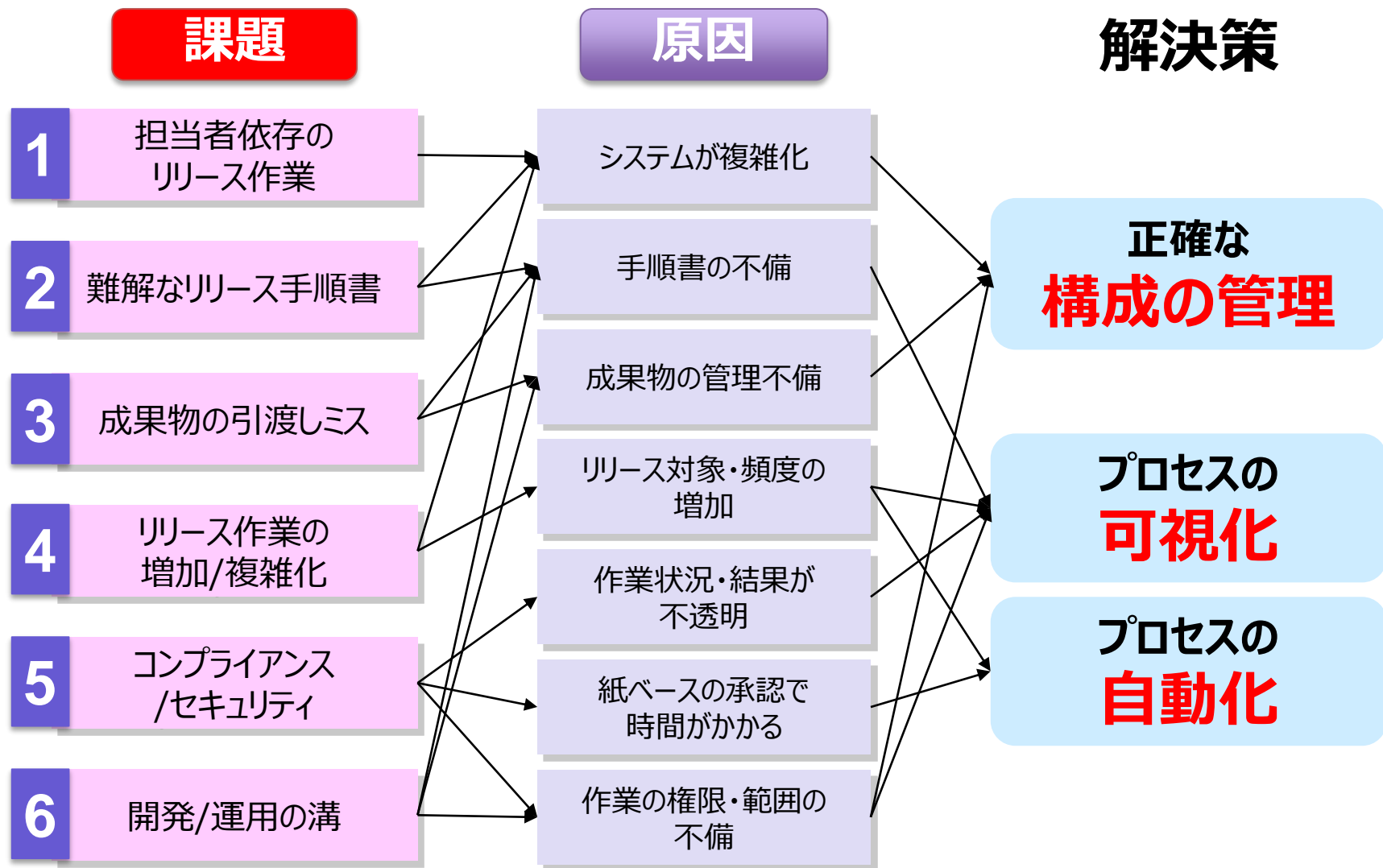
## 原因

- ✓ お互いに何をしているのか  
正しく理解していない
- ✓ 責任範囲があいまい  
もしくは、運用部門に丸投げ
- ✓ 開発部門からの引き継ぎ書は  
作り直すのが前提
- ✓ 互いの信用が無い

## 2. 課題解決には何が必要か？

---

# 課題解決には何が必要か？



まずは、以下を実現してみたいはいいでしょうか？

#### 正確な構成の管理

- 各環境の**WASの構成、パラメータ**を一元的に管理する
- 各環境にデプロイされているアプリケーションの**バージョン**を管理する
- 構成変更の**履歴や権限の管理**を適切に行う

#### プロセスの可視化・自動化

- **アプリケーションのデプロイ**プロセスを**可視化・自動化**する
- **WASの設定変更**プロセスを**可視化・自動化**する



UrbanCode Deployで実現できます

## 3. UrbanCode Deploy で課題を解決

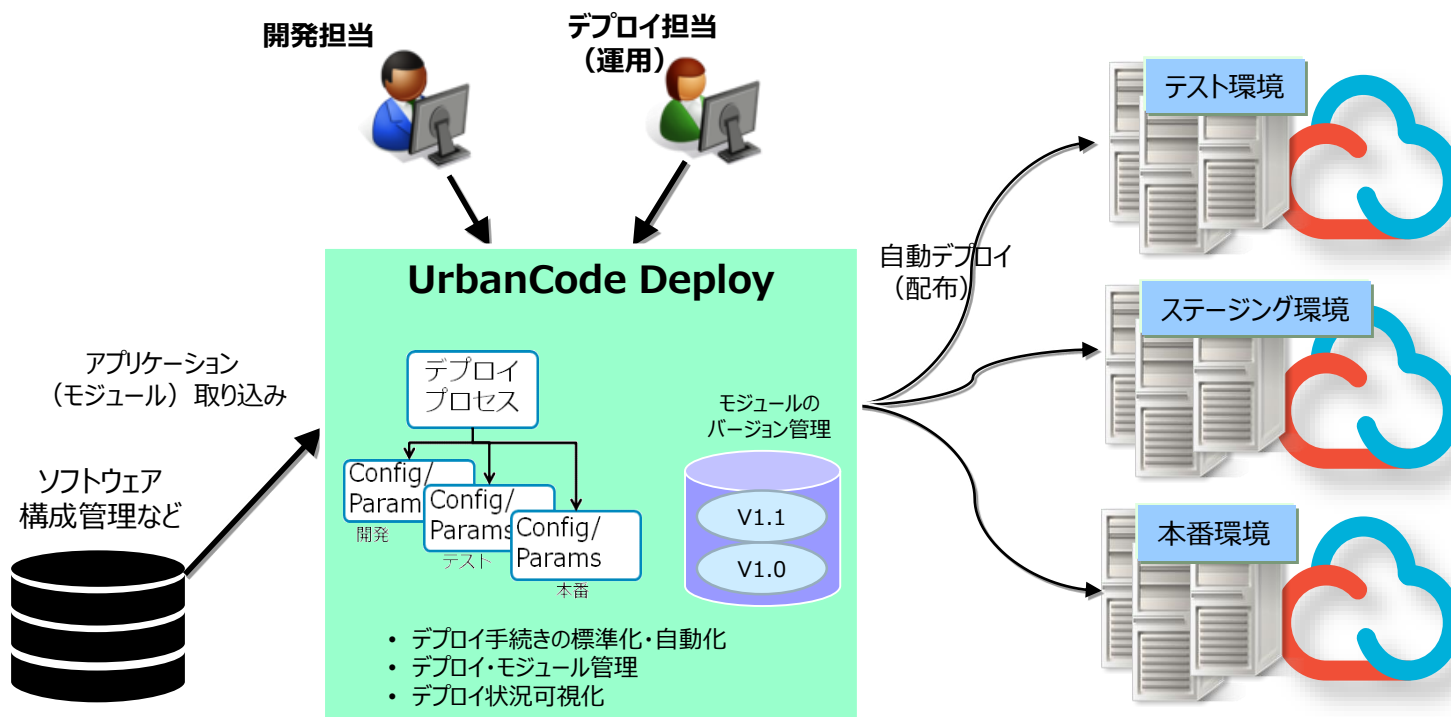
---

- 3.1. UCDとは？
- 3.2. ユースケース
- 3.3. UCD 利用の流れ



# UrbanCode Deployとは？

デプロイ自動化プラットフォーム (IBM UrbanCode Deploy)  
テスト環境～本番環境までのアプリケーション・デプロイの自動化をサポート



頻度・スピード向上

ミスの削減

工数の削減

セキュリティー

開発チーム ～ 運用チームまで共通のプラットフォーム

# UCDの特徴：その1 プロセスの可視化

## ■ デプロイ手順を「プロセス」として定義

- ◆ 事前定義の部品（プラグイン）を用いて、容易にプロセスの定義が可能



事前定義の部品  
(一部)



# UCDの特徴：その2 プロセスの自動実行

## ■ 定義済みプロセスの実行

- ◆ **属人性の排除**：権限が付与されていれば誰でも実行可能
- ◆ スケジュール設定による**自動実行**も可能

stack-tanaka01-3e9d54e6351907b0 上でプロセスを実行

変更済みバージョンのみ ☒

プロセス \* DeployJKE

スナップショットを選択するか、または個々のコンポーネントのバージョンを選択します。

スナップショット

コンポーネント・バージョン

バージョン 1 個を選択 (バージョンの選択)

デプロイメントをスケジュールしますか? ☐

説明

実行依頼 キャンセル

クリックで  
デプロイ開始

各ステップレベルで、  
実行結果、経過時間を  
レポート

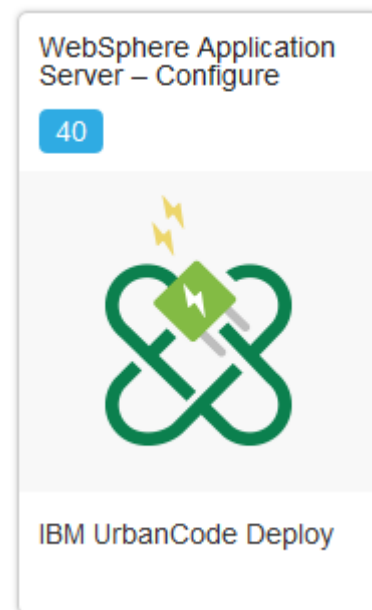
ステップ	進行状況	開始時刻	所要時間	状況
▼ 1. jke.war をインストール	1 / 1	5:32:14	0:00:31	成功
▼ jke.war	1 / 1	5:32:15	0:00:30	成功
▼ deploy (jke.war 1.0)	::	5:32:15	0:00:30	成功
1. Download Artifacts	::	5:32:16	0:00:05	成功
2. Update Property File	::	5:32:22	0:00:02	成功
3. Create WAR archive	::	5:32:25	0:00:03	成功
4. Stop Server	::	5:32:29	0:00:04	成功
5. Create File	::	5:32:33	0:00:03	成功

## UCDの特徴：その3 WASとの親和性が高い（1/3）

- WAS用に定義された部品を多数提供（Pluginにより提供）
  - ◆ **Install**（WASのインストール）
  - ◆ **Configure**（構成情報の管理）
  - ◆ **Deployment**（アプリケーションのデプロイを管理）
- WASの管理作業を視覚化・自動化
  - ◆ 管理コンソールやwsadminで行う作業をUCDから実行
- 様々な構成を管理可能
  - ◆ Base Edition、ND Edition
  - ◆ クラスター構成（静的/動的） / スタンドアロン構成

## UCDの特徴：その3 WASとの親和性が高い（2/3）

- WASの**構成管理**（Configureプラグイン）
  - ◆ 稼動しているWASインスタンスから直接構成情報を取得
    - 「リソース・ツリー」として視覚化
  - ◆ 構成情報はテキスト（JSON形式）で保存
    - UCDが自動でバージョン管理 = 差分比較が可能
  - ◆ 構成の変更履歴を確認可能
    - 誰が、いつ、何を変えたかを記録
  - ◆ WASを稼動させたまま新規環境への移行が可能



## Configure Pluginの主な機能

The screenshot displays the 'Configure Plugin' interface in WebSphere Application Server V9. It includes a sidebar with configuration snippets, a main panel with a resource tree, and a table for version management.

**事前定義の部品** (Pre-defined components): A list of configuration snippets on the left sidebar, including 'WebSphere - Create Configuration Snippet', 'WebSphere - Extract Configuration Data', and various 'Templatize' options for Cell, Cluster, Node, Server, and Cluster Configuration Data.

**リソース・ツリー** (Resource Tree): A tree view on the right showing the hierarchy of resources. It includes 'V8', 'V9', 'agent1 (エージェントの表示)', 'ise043Cell01', 'Cell - ise043Cell01 (コンポーネントの表示)', 'Nodes', 'ise043CellManager01', 'Node - ise043CellManager01 (コンポーネントの表示)', 'Portal Servers', 'Servers', and 'dmgr'.

**コンポーネント: Cell Config (詳細を表示)**: A section showing the 'Cell Config' component with tabs for 'ダッシュボード', '使用法', '構成', 'カレンダー', and 'バージョン'.

**バージョン管理** (Version Management): A table listing versions of the configuration. The table has columns for 'バージョン' (Version) and '状況' (Status). The versions listed are: 2016-09-20T16:48:38.571+0900, 2016-09-14T14:38:29.841+0900, 2016-09-09T17:17:56.222+0900, 2016-09-08T17:11:00.861+0900, 2016-09-08T17:02:32.270+0900, 2016-09-07T12:57:16.195+0900, 2016-09-07T11:42:01.632+0900, 2016-09-06T22:35:26.619+0900, and 2016-09-02T17:17:06.999+0900.

**成果物** (Artifacts): A section showing the total size of the artifacts: '合計: 1.1 MB (2 ファイル)' (Total: 1.1 MB (2 files)). It includes a 'すべてダウンロード' (Download all) button.

**実体はJSONファイル** (Entity is JSON file): A callout pointing to the 'cell' and 'dmgr' folders in the resource tree, indicating that the configuration files are JSON files ('Cell.json' and 'dmgr.json').

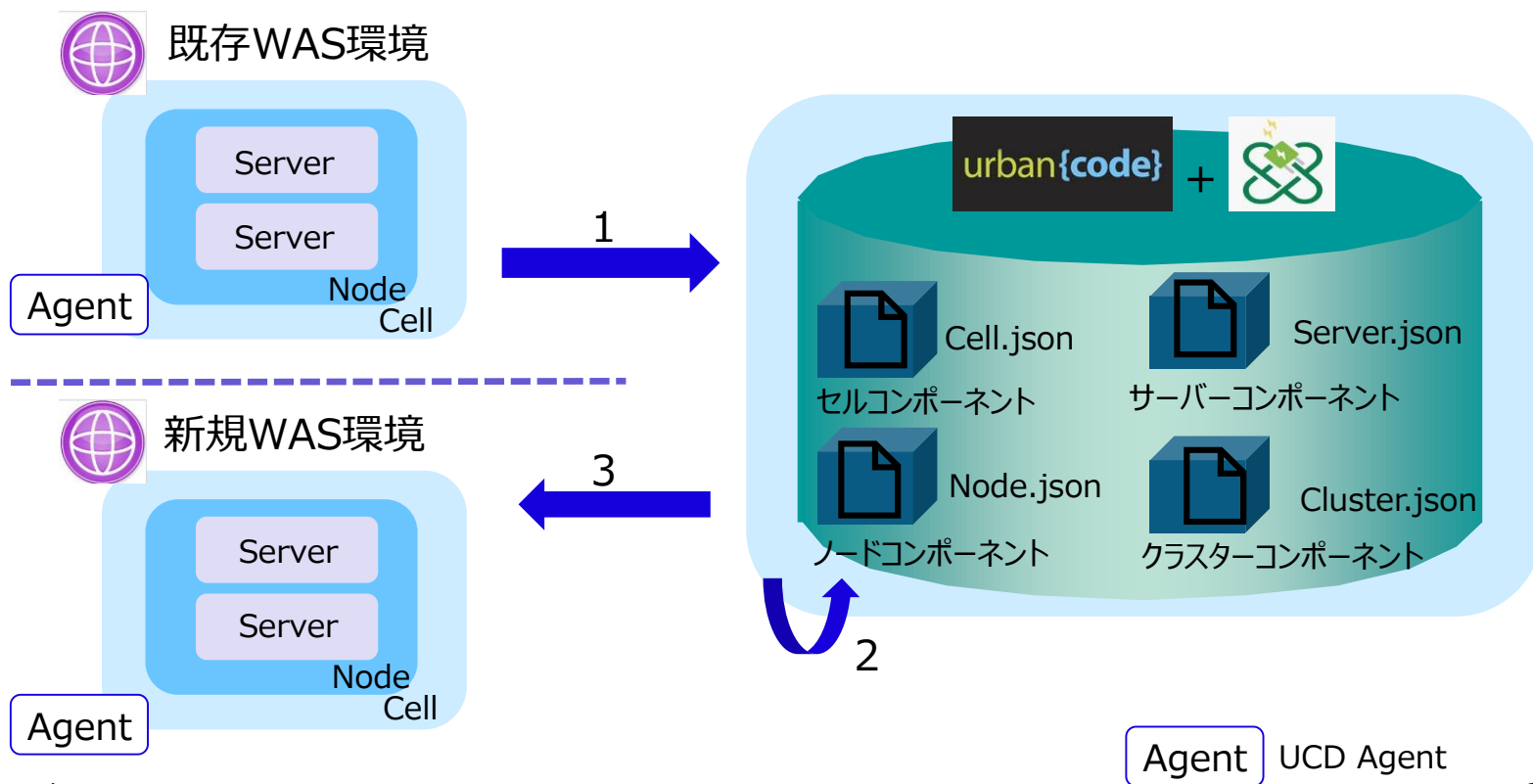
# Configure PluginによるWAS構成管理の流れ

1. 既存環境の  
ディスカバーと  
インポート

2. 構成の  
テンプレート化

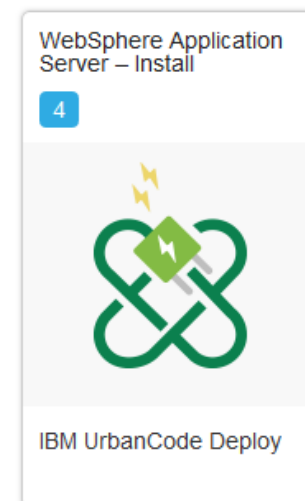
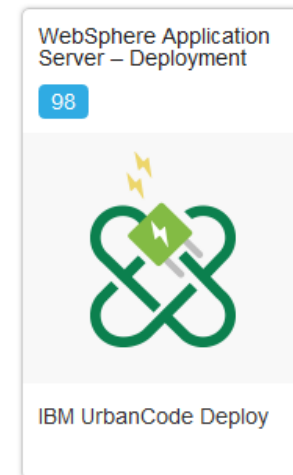
3. テンプレートを  
適用

- ・スコープに分割して管理: Cell/Node/Server/Cluster
- ・各スコープにマップするコンポーネントに保存



## UCDの特徴：その3 WASとの親和性が高い（3/3）

- アプリケーションの**デプロイ**（Deploymentプラグイン）
  - ◆ アプリケーション管理に関連するタスクを実行する多数の部品群を提供
    - アプリケーションのインストール・設定・アンインストールなど
  - ◆ その他、アプリケーションの起動・停止、アプリケーション・サーバーの起動・停止などの基本操作を行うステップも提供
- WASの**インストール**（installプラグイン）
  - ◆ WASおよびIHSのインストール（アンインストール）を自動化
  - ◆ デプロイメント・マネージャー、ノード プロファイルの作成（削除）
  - ◆ その他、ノードの統合、起動、停止、などの基本操作も実行





## 【参考】UrbanCode プラグインの一例

- WAS以外にも多くの製品やツールと連携が可能
- 2016年10月時点で180種のプラグイン

### 【アプリケーション・サーバー】

WebSphere Application Server  
WebLogic  
Apache Tomcat  
Microsoft IIS

### 【データベース】

SQL-JDBC  
Apache Hadoop  
Oracle SQLPlus  
Microsoft SQL Server

### 【品質管理・テスト】

Rational Quality Manager  
Rational Test Workbench Web UI Tester  
HP ALM  
Selenium  
JUnit

### 【ソース管理・ビルド管理】

Rational Team Concert  
Subversion  
Git  
Jenkins  
Maven

### 【クラウド】

Amazon EC2  
Cloud Foundry

### 【スクリプト】

Chef  
Apache Ant  
Groovy

最新情報はこちら

<https://developer.ibm.com/urbancode/plugins/ibm-urbancode-deploy/>

## 3. UrbanCode Deploy で課題を解決

---

- 
- 3.1. UCDとは？
  - 3.2. ユースケース
  - 3.3. UCD 利用の流れ

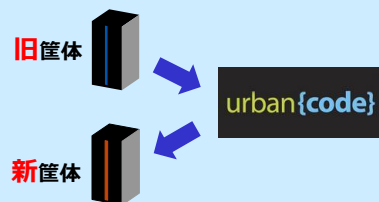
# ユースケース : UCD+WAS 例えばこんなユースケース

## 環境構築系

### Configure Plugin

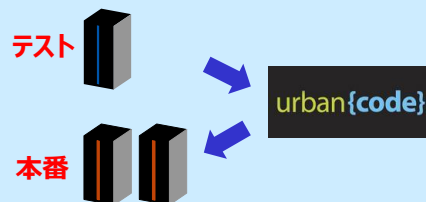
#### 環境の複製

ハードウェアのリプレースに伴う移行



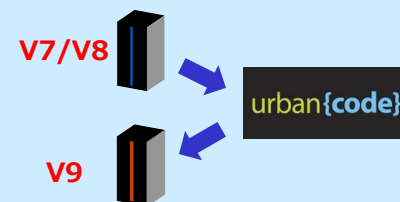
#### 環境の複製&変更

テスト環境をベースにして本番環境を構築



#### バージョンアップ

WAS V7/V8から WAS V9へのマイグレーション



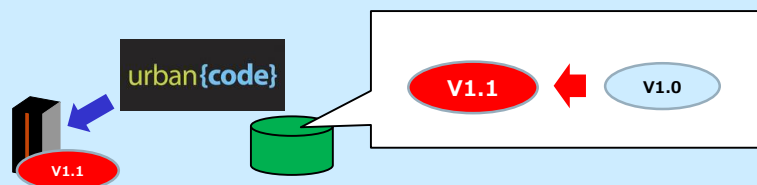
## 通常運用系

### Deployment Plugin

### Configure Plugin

#### アプリケーションのデプロイ

アプリのプロセスを定義、可視化する  
アプリデプロイを自動で実行する



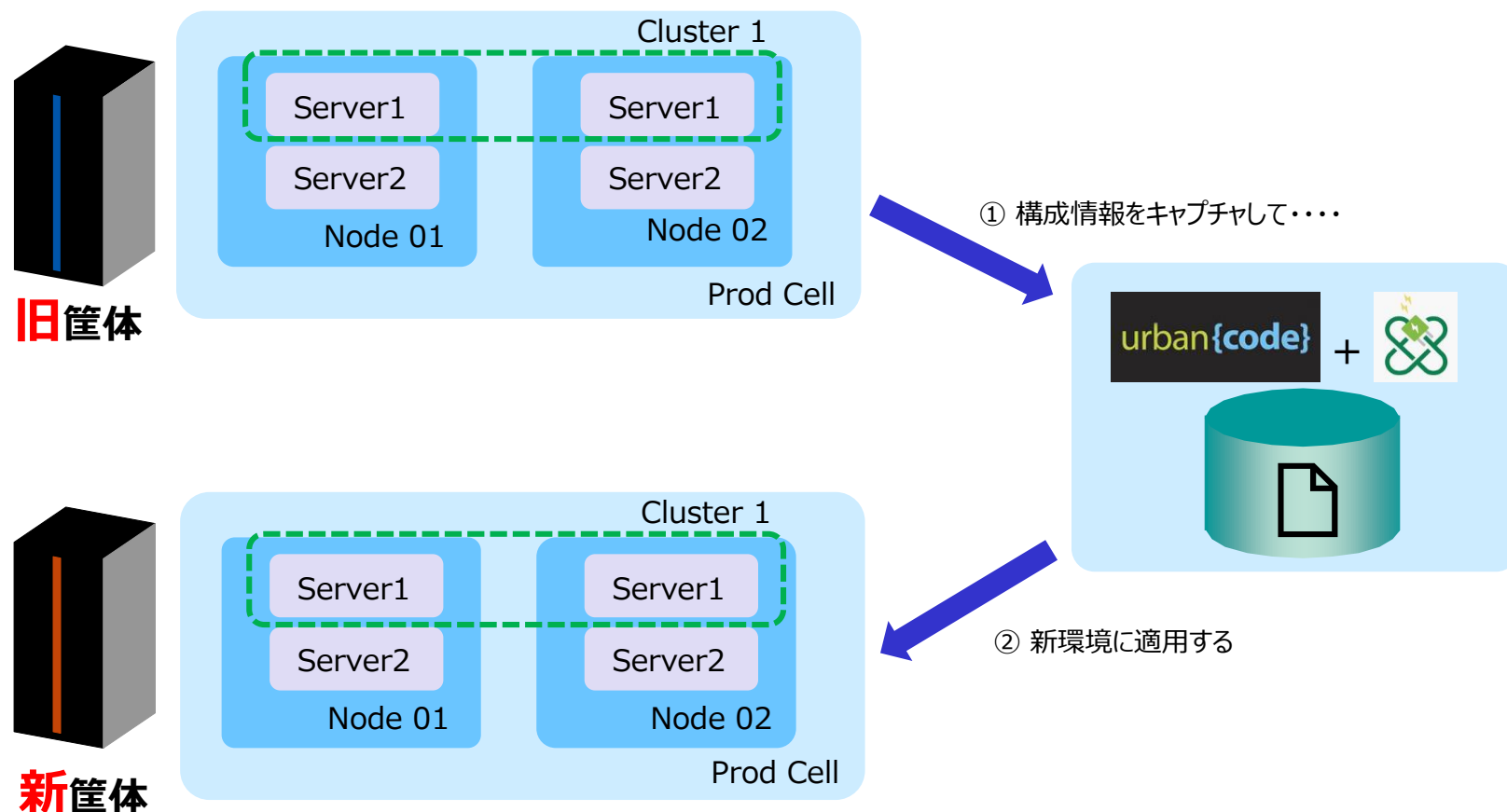
#### WASのパラメータ変更

構成変更のプロセスを定義、可視化する  
構成変更を自動で実行する



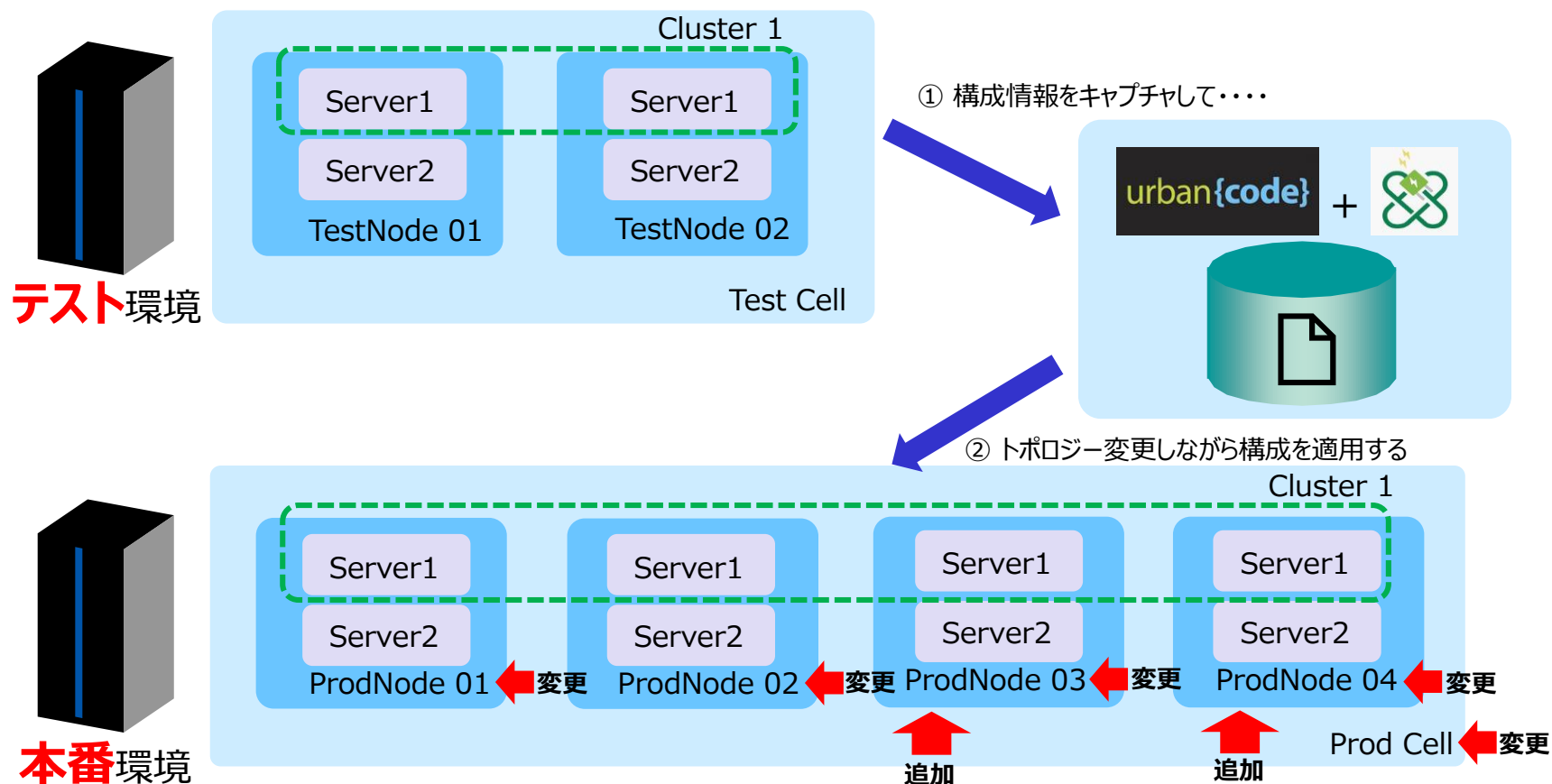
## ユースケース：WASの構成管理 パターン例1

- 移行元/移行先は同じ構成で構築
  - ◆ 例：ハードウェアのリプレイスに伴うWAS**環境の移行**



## ユースケース：WASの構成管理 パターン例2

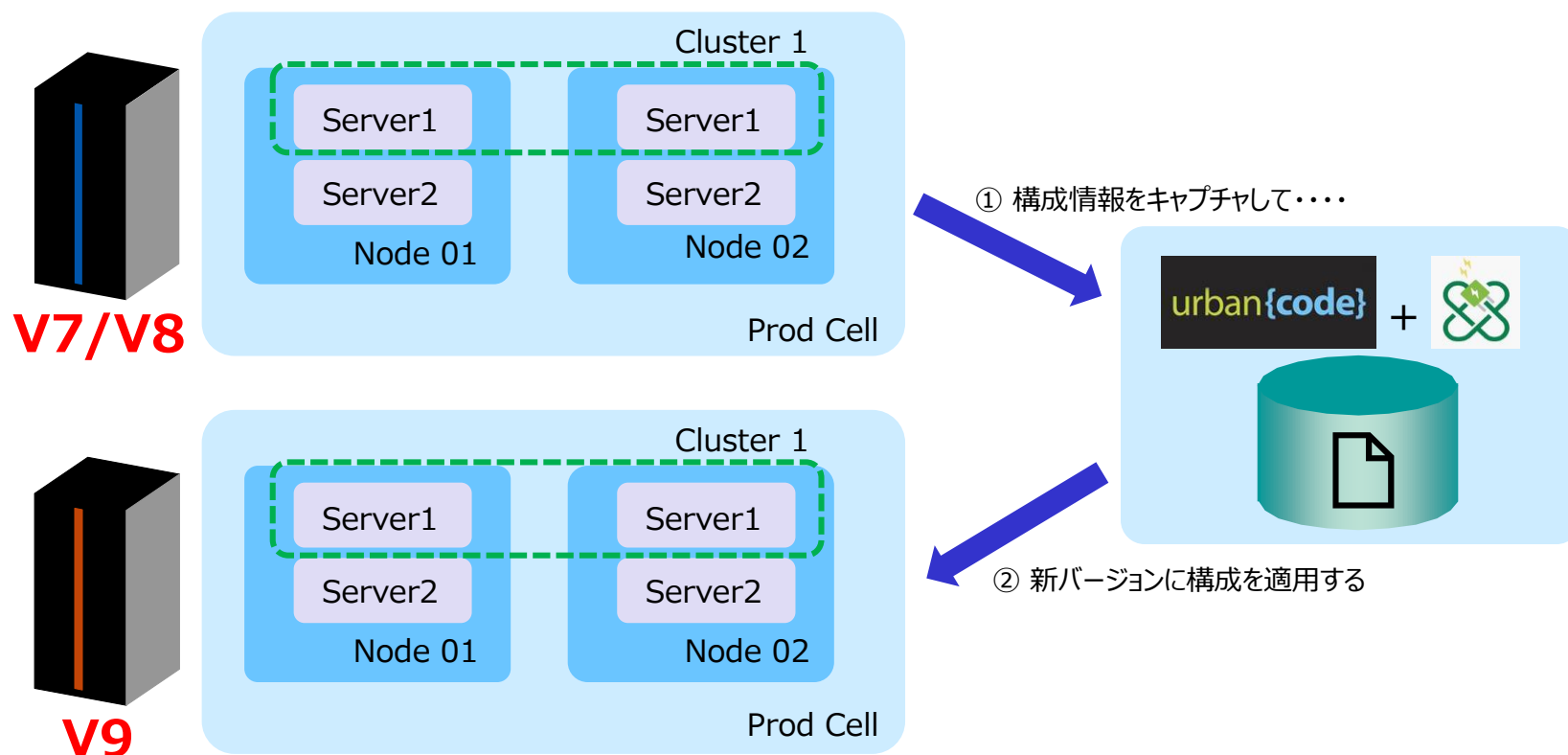
- 移行元/移行先は異なる構成
- 例：**テスト環境**をベースにして**本番環境**を構築
  - ◆ 移行元/移行先でセル名やノード名を変更して構築
  - ◆ 移行先ではノード数やノード上のサーバー数を増やして構築



## ユースケース：WASの構成管理 パターン例3

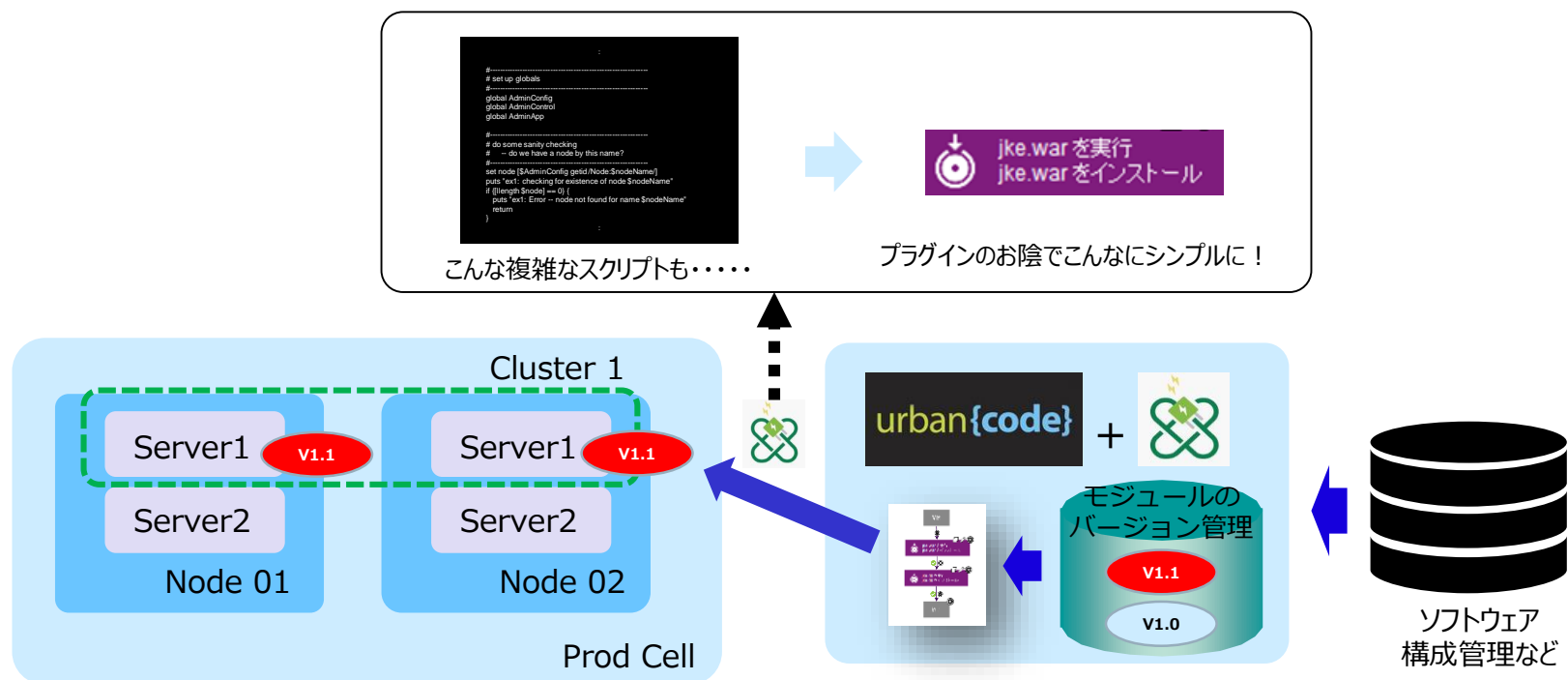
### ■ WAS V7/V8からWAS V9へのマイグレーション

- ◆ 旧バージョンの情報を新バージョンの環境へ適用
- ◆ 手動でコピーが必要なファイルあり
- ◆ アプリケーションの再デプロイは必要



# ユースケース：アプリケーションのデプロイを自動化

- UCDの本分であるアプリケーションの**デプロイ自動化**を、WAS用**Deploymentプラグイン**と組み合わせて利用
  - ◆ アプリケーション導入のために**複雑なスクリプトを用意する必要なし**
  - ◆ Deploymentプラグインが提供する**部品（実行するステップ）を組み合わせてプロセスを作成可能**
  - ◆ UCD内でもアプリケーション・モジュールのバージョン管理が行われる



## 3. UrbanCode Deploy で課題を解決

---

3.1. UCDとは？

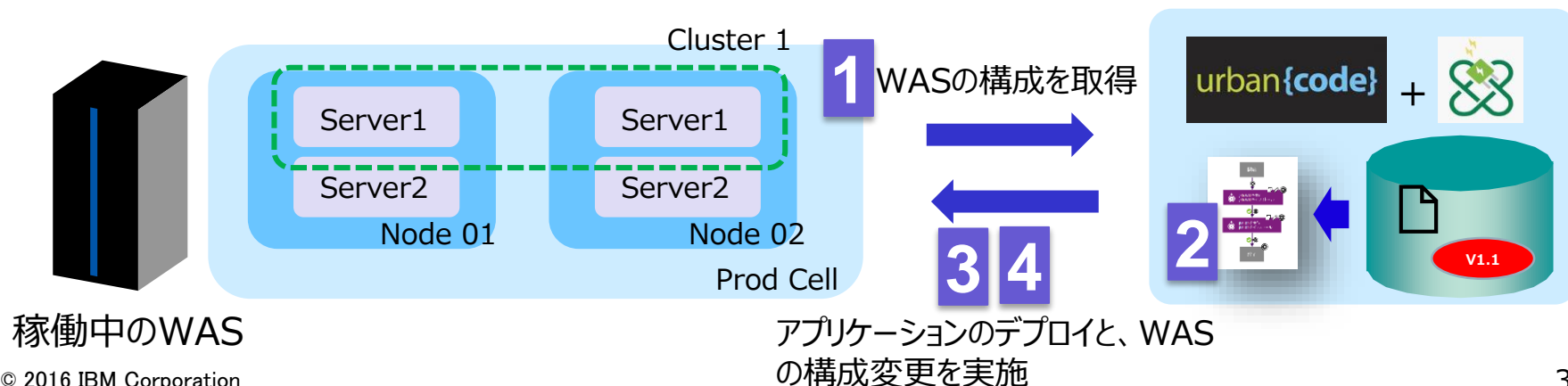
3.2. ユースケース

 3.3. UCD 利用の流れ



## 使用例の紹介

- 前提
  - ◆ 構築済みのWAS環境が存在している
- 実施すること
  - ◆ WAS環境の構成をUCDに取り込む
  - ◆ アプリケーションのデプロイと、WAS構成変更をUCDから行う
- 操作の流れ



## 稼働中のWASの構成を取得



事前準備：UCDのAgentを導入

トポロジーのディスカバーを実行

構成のディスカバーを実行

管理対象のWASが導入されている環境に、UCDのAgentを導入する

# トポロジーのディスカバー 1/3

1 稼働中のWASの  
構成を取得

1. リソース・ツリー上でグループを作成
2. 「エージェントの追加」を実行

The screenshot shows the WebSphere Resource Explorer interface. The top navigation bar includes tabs for 'リソース・ツリー' (Resource Tree), 'リソース・テンプレート' (Resource Template), 'エージェント' (Agent), 'エージェント・リレー' (Agent Relay), and 'エージェント・グループ' (Agent Group). Below the navigation bar, there are buttons for 'トップレベルのグループの作成' (Create top-level group), 'すべて選択...' (Select all...), 'アクション...' (Action...), and '表示' (Display). The main area displays a tree structure with folders 'V8' and 'V9'. The 'V9' folder is selected, and a context menu is open, showing options like '比較または同期化' (Compare or synchronize), '新規テンプレートの定義' (Define new template), 'テンプレートとの同期化' (Synchronize with template), 'テンプレートからの追加' (Add from template), 'グループの追加' (Add group), 'エージェントの追加' (Add agent), 'エージェント・グループの追加' (Add agent group), and '削除' (Delete). The 'エージェントの追加' option is highlighted. A blue arrow points from the 'V9' folder to a detailed view of the 'V9' folder, which shows a sub-folder 'agent1 (エージェントの表示)' and a globe icon labeled 'WebSphereCell - Dmgr01'.

3. Cellオブジェクトがディスカバーされる

## トポロジーのディスカバリー 2/3

1

稼働中のWASの  
構成を取得

### 4.Cellに対して「トポロジー・ディスカバリー」を実行

リソース・ツリー    リソース・テンプレート    エージェント    エージェント・リレー    エージェント・...

トップレベルのグループの作成    すべて選択...    アクション...    表示

名前	リソース名	タグ
...	V8	
...	V9	
...	agent1 (エージェントの表示)	
...	WebSphereCell - Dmgr01	

更新    印刷

- グループの追加
- コンポーネントの追加
- コンポーネント・タグの追加
- WebSphereトポロジー・ディスカバリー を使用した構成**
- WebSphere Configuration Apply を使用した構成
- WebSphere Configuration Discovery を使用した構成
- WebSphereトポロジー・ディスカバリー を使用した比較
- WebSphere Configuration Discovery を使用した比較
- 削除

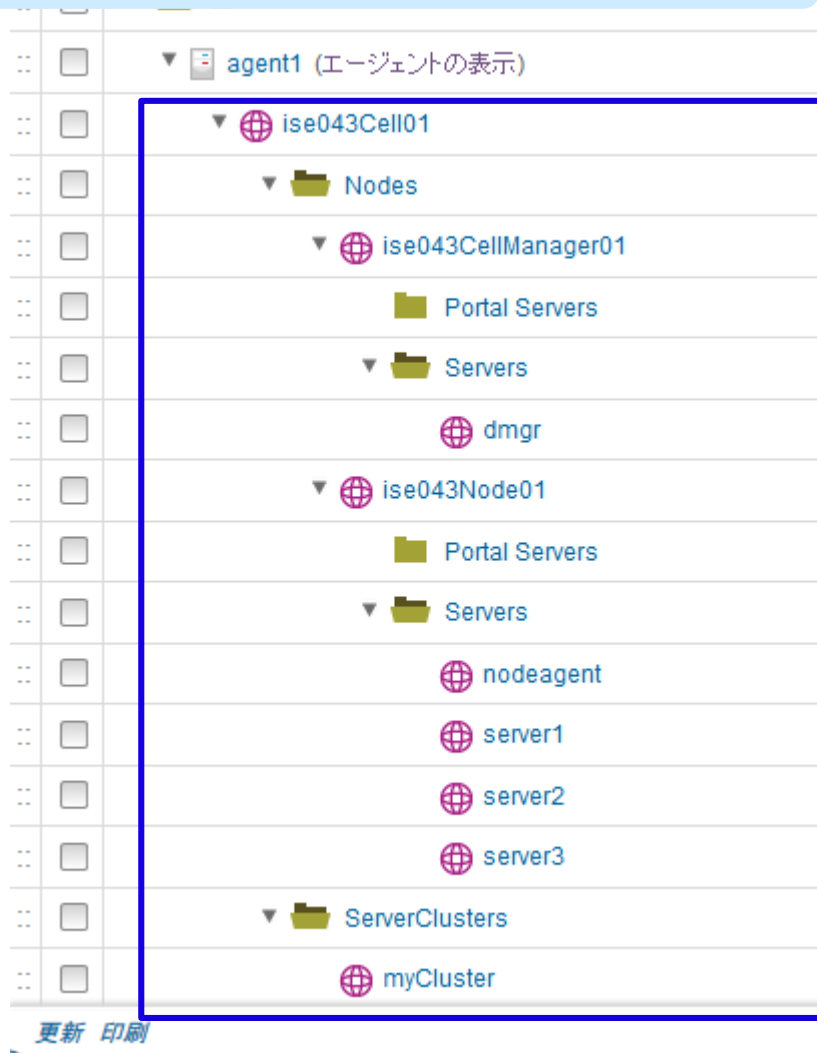
アクション...

## トポロジーのディスカバー 3/3

1

稼働中のWASの  
構成を取得

5.Cell内の情報がディスカバーされ、ツリー形式で表示される



## 構成のディスカバー 1/2

1 稼働中のWASの  
構成を取得

### ■ ディスカバーしたトポロジーに対して実施

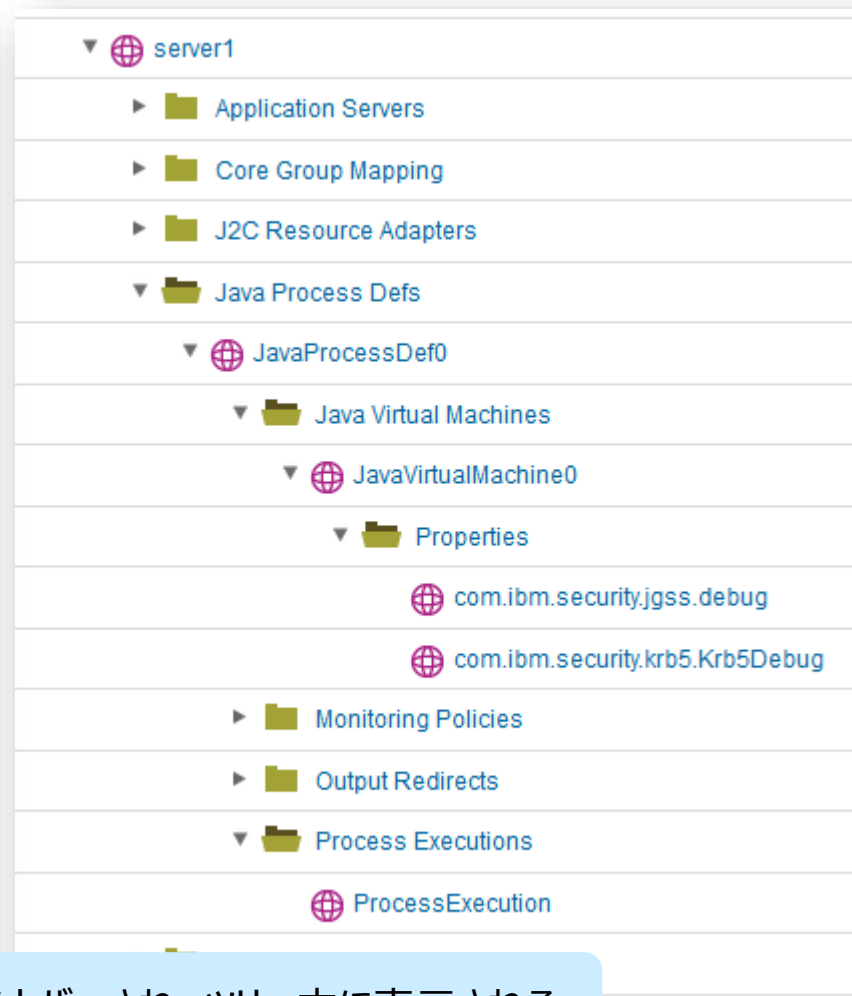
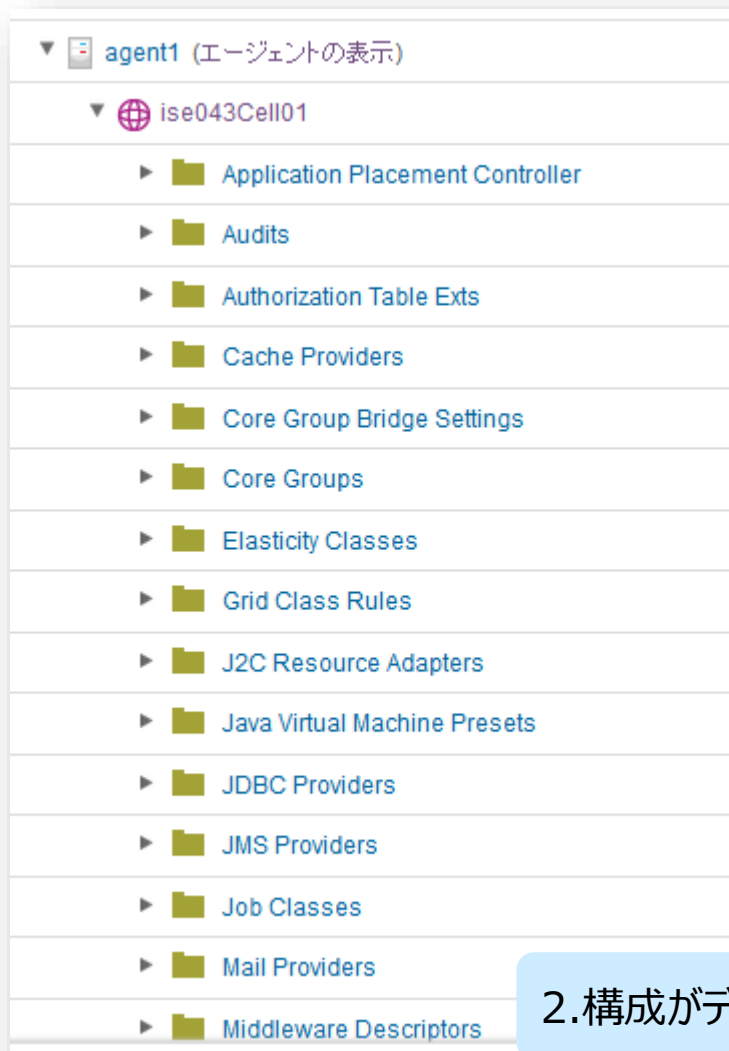
1.「WebSphere Configuration  
Discoveryを使用した構成」を実行

The screenshot displays the WebSphere Configuration Explorer interface. The left pane shows a hierarchical tree structure under the 'V9' environment. The 'agent1 (エージェントの表示)' folder is expanded, showing 'ise043Cell01'. Under 'ise043Cell01', the 'Nodes' folder is expanded, showing 'ise043CellManager01' and 'ise043Node01'. The 'Servers' folder under 'ise043Node01' is expanded, showing 'nodeagent', 'server1', 'server2', and 'server3'. The 'ise043Cell01' folder is highlighted with a blue box. An arrow points from this box to the 'WebSphere Configuration Discovery' option in the 'アクション...' (Actions) menu. The menu is open, showing various options, with 'WebSphere Configuration Discovery' highlighted in blue.

比較または同期化  
新規テンプレートの定義  
テンプレートとの同期化  
テンプレートからの追加  
グループの追加  
コンポーネントの追加  
コンポーネント・タグの追加  
WebSphere トポロジー・ディスカバリー を使用した構成  
WebSphere Configuration Apply を使用した構成  
**WebSphere Configuration Discovery を使用した構成**  
WebSphere トポロジー・ディスカバリー を使用した比較  
WebSphere Configuration Discovery を使用した比較  
削除

## 構成のディスカバー 2/2

1

稼働中のWASの  
構成を取得

2. 構成がディスカバーされ、ツリー内に表示される

## 操作イメージ



デプロイ・プロセスの定義

WebやDBなど、コンポーネント毎にプロセスを定義

アプリケーションの定義

コンポーネントをまとめるアプリケーションを定義

アプリケーションと環境の紐付け

アプリケーションのデプロイ先を定義



## デプロイ・プロセスの定義

- アプリケーションを構成するコンポーネント毎に定義可能
- UCD「コンポーネント」のプロセスとして定義



2. WAR作成とDB作成用の  
コンポーネントを定義した例

The screenshot shows the 'コンポーネント' (Component) tab in the WebSphere Application Server V9 console. The 'コンポーネントの作成' (Create component) button is highlighted. The table below shows the components defined for the WAR and DB creation process.

名前
名前
System Team
Team-A
jke.war
jke.db

2 個のレコード - 更新 印刷

1. WARを作成する  
コンポーネント・プロセスの例

## アプリケーションの定義

2

アプリケーション・  
デプロイプロセスの定義

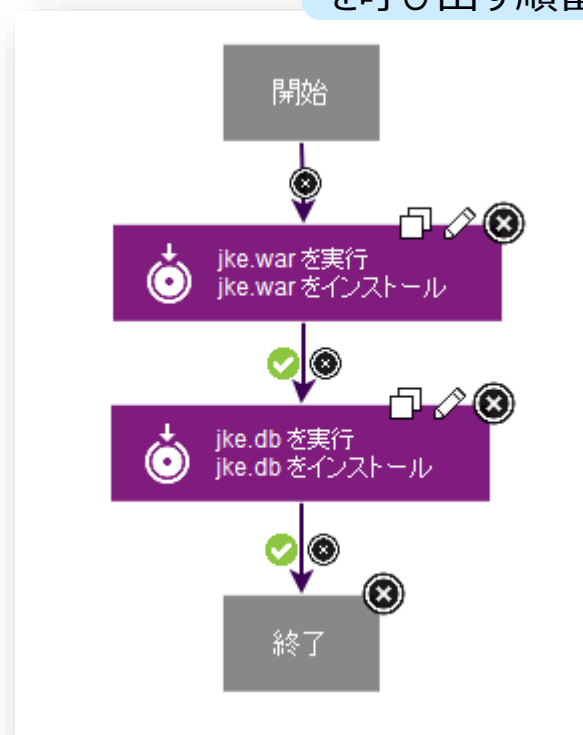
- コンポーネントをまとめる「アプリケーション」を定義
- アプリケーションの単位でデプロイを実行

2. 各コンポーネントのプロセス  
を呼び出す順番を定義

アプリケーション	
アプリケーションの作成	
<input type="checkbox"/>	名前
	名前
	System Team
	Team-A
<input type="checkbox"/>	JKE_APP

2 個のレコード - 更新 印刷

1. アプリケーションを作成



# アプリケーションと環境の紐付け

2

アプリケーション・  
デプロイプロセスの定義

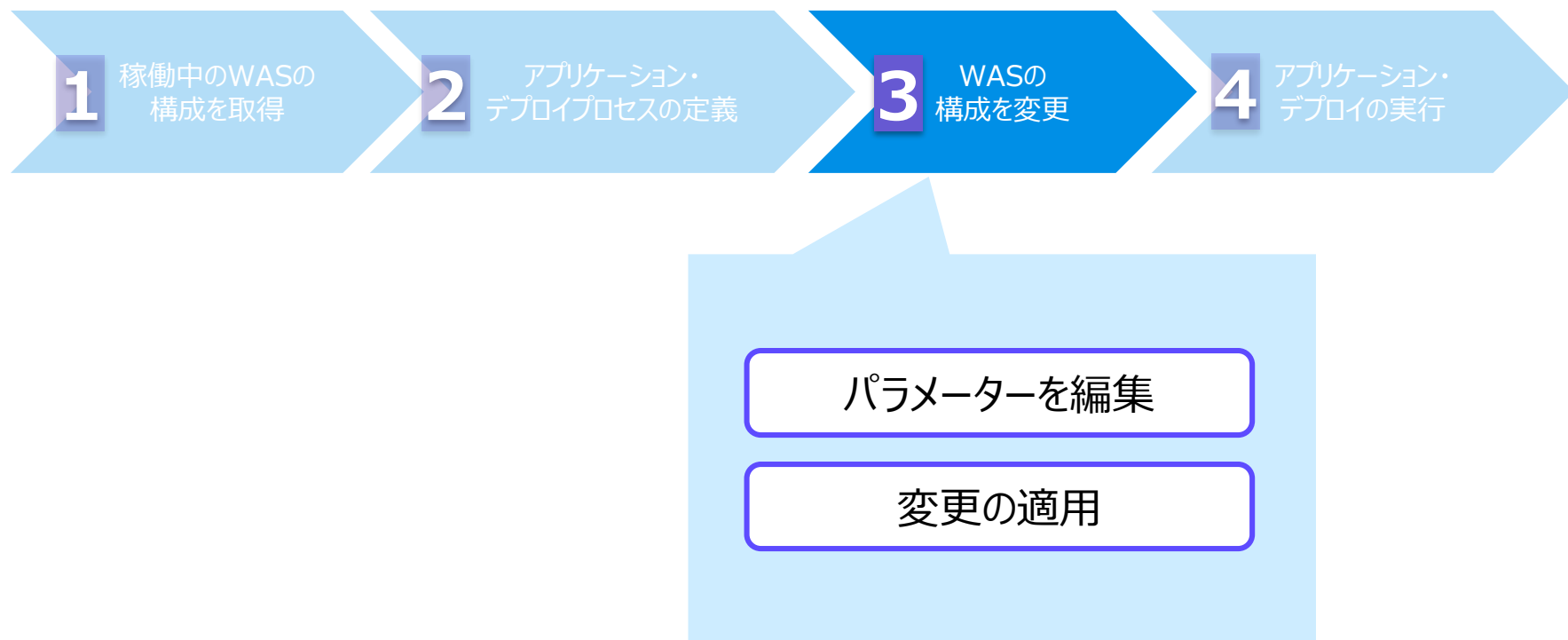
1. アプリケーションのデプロイ  
先を表す「環境」を定義



2. 環境に紐付ける  
リソースを指定



## 操作イメージ



# パラメーターを編集

3 WASの構成を変更



リソースの編集

役割のプロパティ: WebSphere.JavaVirtualMachine

2. Initial Heap Sizeを変更する例

プロパティ	説明	フィルターを表示	
Internal Class Access Mode	The Internal Class Access Mode for this WebSphere JavaVirtualMachine. Acceptable values are: ALLOW, RESTRICT.	ALLOW	デフォルトの使用
Debug Args	The Debug Args for this WebSphere JavaVirtualMachine	-agentlib:jdwp=transport=dt_socket,server=y,suspe	デフォルトにリセット
Classpath	The Classpath for this WebSphere JavaVirtualMachine		デフォルトの使用
Initial Heap Size	The Initial Heap Size for this WebSphere JavaVirtualMachine	0	デフォルトの使用
Run H Prof	The Run H Prof for this WebSphere JavaVirtualMachine	<input type="checkbox"/>	デフォルトの使用
Generic Jvm Arguments	The Generic Jvm Arguments for this WebSphere JavaVirtualMachine		デフォルトの使用
Hprof Arguments	The Hprof Arguments for this WebSphere JavaVirtualMachine		デフォルトの使用
On Name	The On Name for this WebSphere JavaVirtualMachine		デフォルトの使用

Initial Heap Size	The Initial Heap Size for this WebSphere JavaVirtualMachine	2048
-------------------	---	------

# パラメーターを編集

3 WASの構成を変更

リソースの編集

ここでデフォルトの別ユーザー名の使用を構成できます。独自の別ユーザー名使用を指定しないステップはすべて、ここで指定された設定になります。

役割のプロパティ: WebSphereJavaVirtualMachine

プロパティ	説明		
フィルターの表示			
Maximum Heap Size	The Maximum Heap Size for this WebSphere JavaVirtualMachine	4096	デフォルトにリセット
Disable JIT	The Disable JIT for this WebSphere JavaVirtualMachine	<input type="checkbox"/>	デフォルトの使用
Verbose Mode Garbage Collection	The Verbose Mode Garbage Collection for this WebSphere JavaVirtualMachine	<input type="checkbox"/>	デフォルトの使用
Executable Jar File Name	The Executable Jar File Name for this WebSphere JavaVirtualMachine		デフォルトの使用
Verbose Mode Class	The Verbose Mode Class for this WebSphere JavaVirtualMachine	<input type="checkbox"/>	デフォルトの使用
Debug Mode	The Debug Mode for this WebSphere JavaVirtualMachine	<input type="checkbox"/>	デフォルトの使用

16 個のレコード - 更新 印刷

2 / 2

行 10

3.Max Heap Sizeを変更する例

4.「保存」をクリック

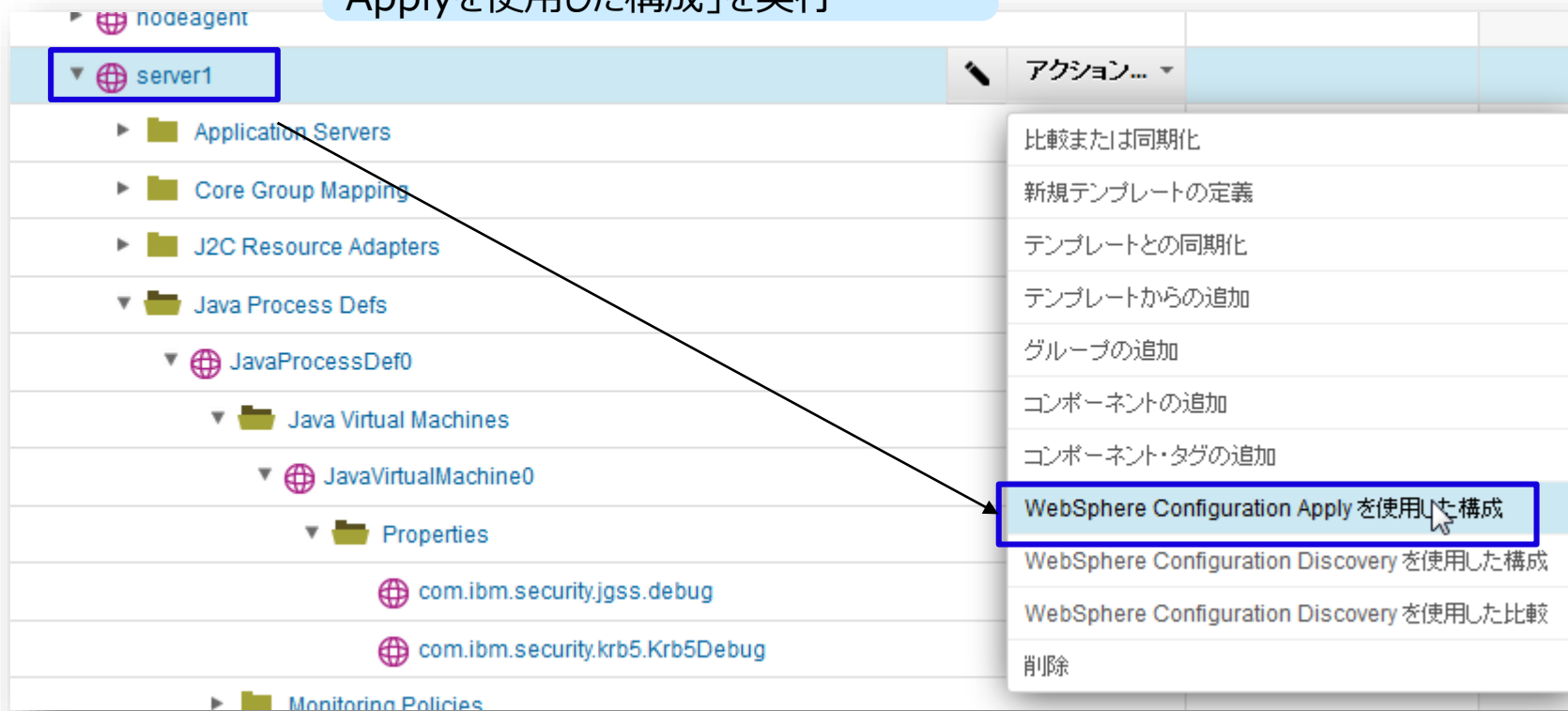
保存

キャンセル

## 変更の適用

3 WASの  
構成を変更

5.「WebSphere Configuration Applyを使用した構成」を実行



## 変更の適用

3 WASの  
構成を変更

ホーム > リソース > JavaVirtualMachine0  
リソース: JavaVirtualMachine0

メイン | インベントリ | 履歴 | 構成

コミット ID	ユーザー	変更日時	変更内容
687	admin	2016/10/26 13:40	WebSphereJavaVirtualMachine v.2 (変更)
686	admin	2016/10/25 21:45	WebSphereJavaVirtualMachine v.1 (変更) アドホック・プロパティ v.1 (変更)

更新 印刷

変更前後の値まで確認可能



誰が、いつ、何を、どのように変更したかトラッキングできます

アプリケーション・サーバー > server1 > プロセス定義 > Java 仮想マシン

このページを使用して、Java(TM) 仮想マシンの詳細設定を構成します。

構成

管理コンソールから変更結果を確認すると……

☐ 冗長 JNI

初期ヒープ・サイズ  
 MB

最大ヒープ・サイズ  
 MB

☐ HProf の

HProf 引数

初期ヒープ・サイズ  
 MB

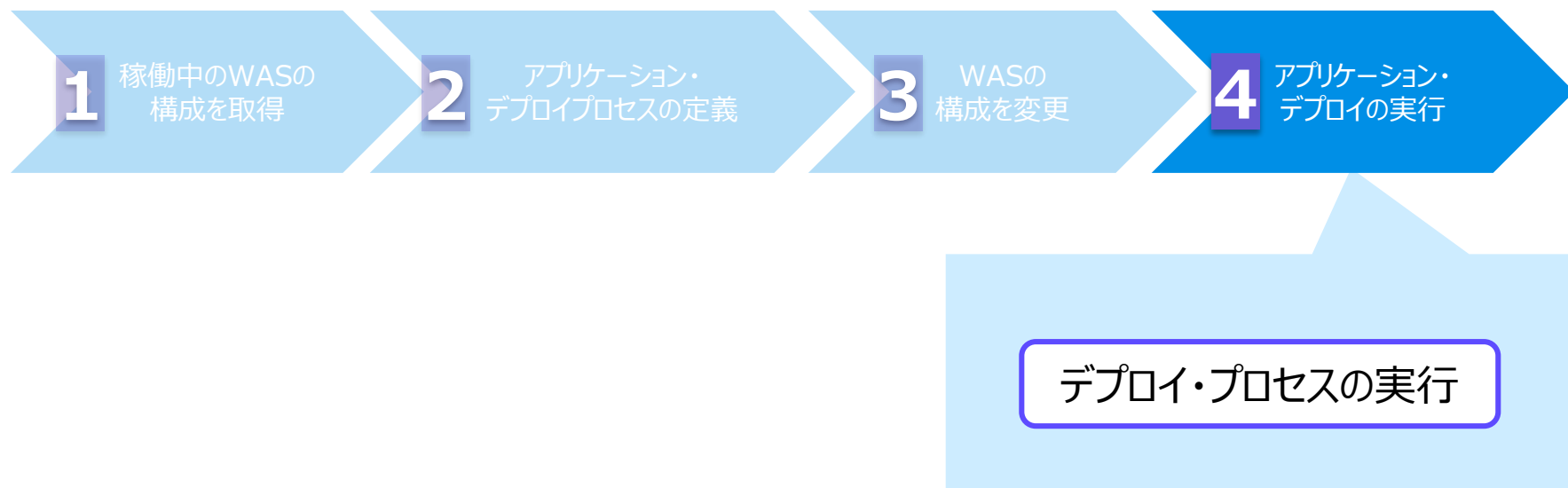
最大ヒープ・サイズ  
 MB

☐ HProf の実行

変更が適用されている事を確認



## 操作イメージ



# デプロイ・プロセスの実行

## 4 アプリケーション・デプロイの実行



1. プロセスの要求をクリック



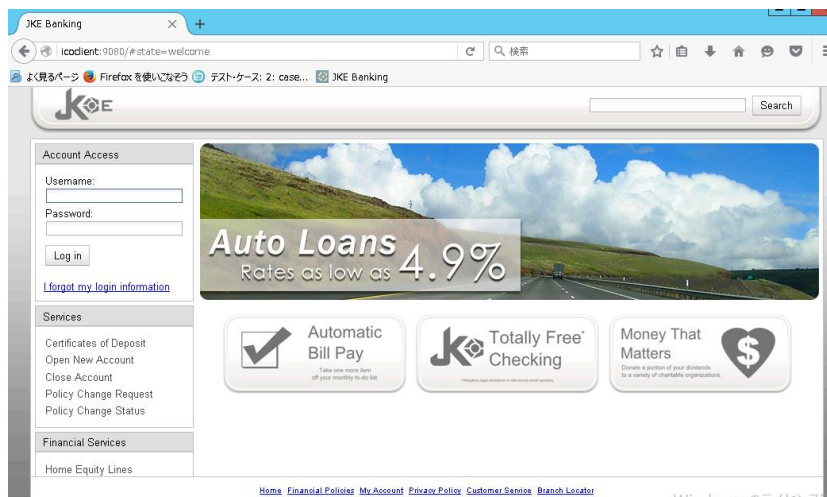
2. 実行対象のプロセスを選択し、「実行依頼」をクリック

# デプロイ・プロセスの実行

## 4 アプリケーション・デプロイの実行

### 3. プロセスが自動実行される

ステップ	進行状況	開始時刻	所要時間	状況
▼ 1. jke.war をインストール	1 / 1	5:32:14	0:00:31	成功
▼ jke.war	1 / 1	5:32:15	0:00:30	成功
▼ deploy (jke.war 1.0)	::	5:32:15	0:00:30	成功
1. Download Artifacts	::	5:32:16	0:00:05	成功
2. Update Property File	::	5:32:22	0:00:02	成功
3. Create WAR archive	::	5:32:25	0:00:03	成功
4. Stop Server	::	5:32:29	0:00:04	成功
5. Create File	::	5:32:33	0:00:03	成功



アプリケーションがデプロイされ、アクセス可能に！

## まとめ

---

- リリース・デプロイ作業によくある課題の解決には、UrbanCode Deployの活用が有効
- UrbanCode Deploy+WASプラグインを利用することで、以下のプロセスの可視化・自動化が可能になる
  - ◆ WASのインストール / アンインストール
  - ◆ WASの構成管理
  - ◆ WASへのアプリケーションのデプロイ

ワークショップ、セッション、および資料は、IBMまたはセッション発表者によって準備され、それぞれ独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる参加者に対しても法律的またはその他の指導や助言を意図したものではなく、またそのような結果を生むものでもありません。本講演資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本講演資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMは責任を負わないものとします。本講演資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引き出すことを意図したもので、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでなく、またそのような結果を生むものでもありません。

本講演資料でIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本講演資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本講演資料に含まれている内容は、参加者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したもので、またそのような結果を生むものでもありません。パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。

IBM、IBM ロゴ、ibm.com、WebSphereは、世界の多くの国で登録されたInternational Business Machines Corporationの商標です。

他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。

現時点での IBM の商標リストについては、[www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)をご覧ください。

Windowsは Microsoft Corporationの米国およびその他の国における商標です。

JavaおよびすべてのJava関連の商標およびロゴは Oracleやその関連会社の米国およびその他の国における商標または登録商標です。