



# IT Automation Ansible Driver 【座学編】

※本書では「Exastro IT Automation」を「ITA」として記載します。

第1.7.2版

Exastro developer

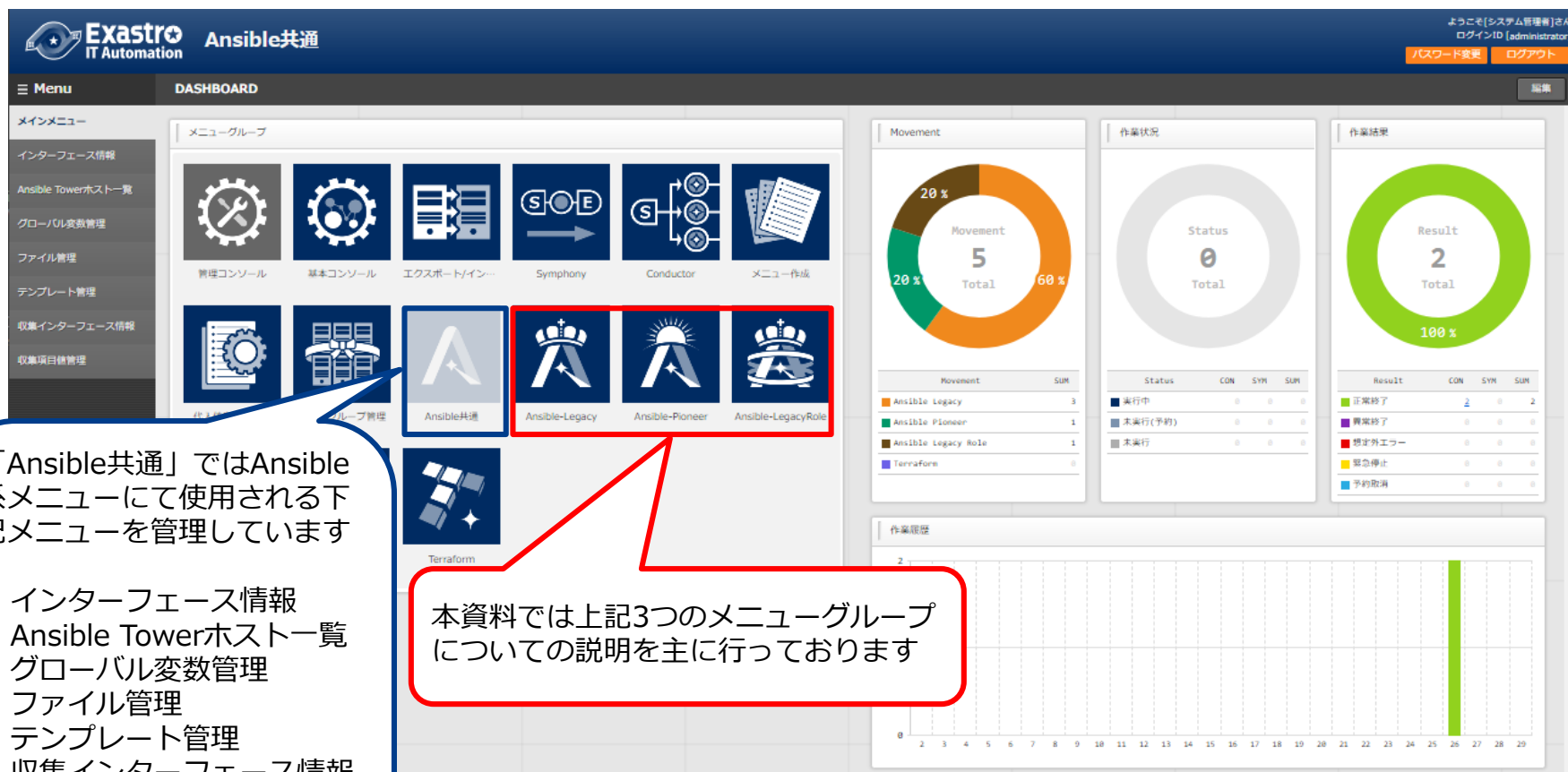
# 目次

1. [はじめに](#)
2. [Ansible Driverとは](#)
3. [Ansible Towerとの連携](#)
4. [3つのモードの説明](#)
5. [各モードの特徴](#)
  1. [Legacyモード](#)
  2. [LegacyRoleモード](#)
  3. [Pioneerモード](#)

# 1. はじめに

## メインメニュー

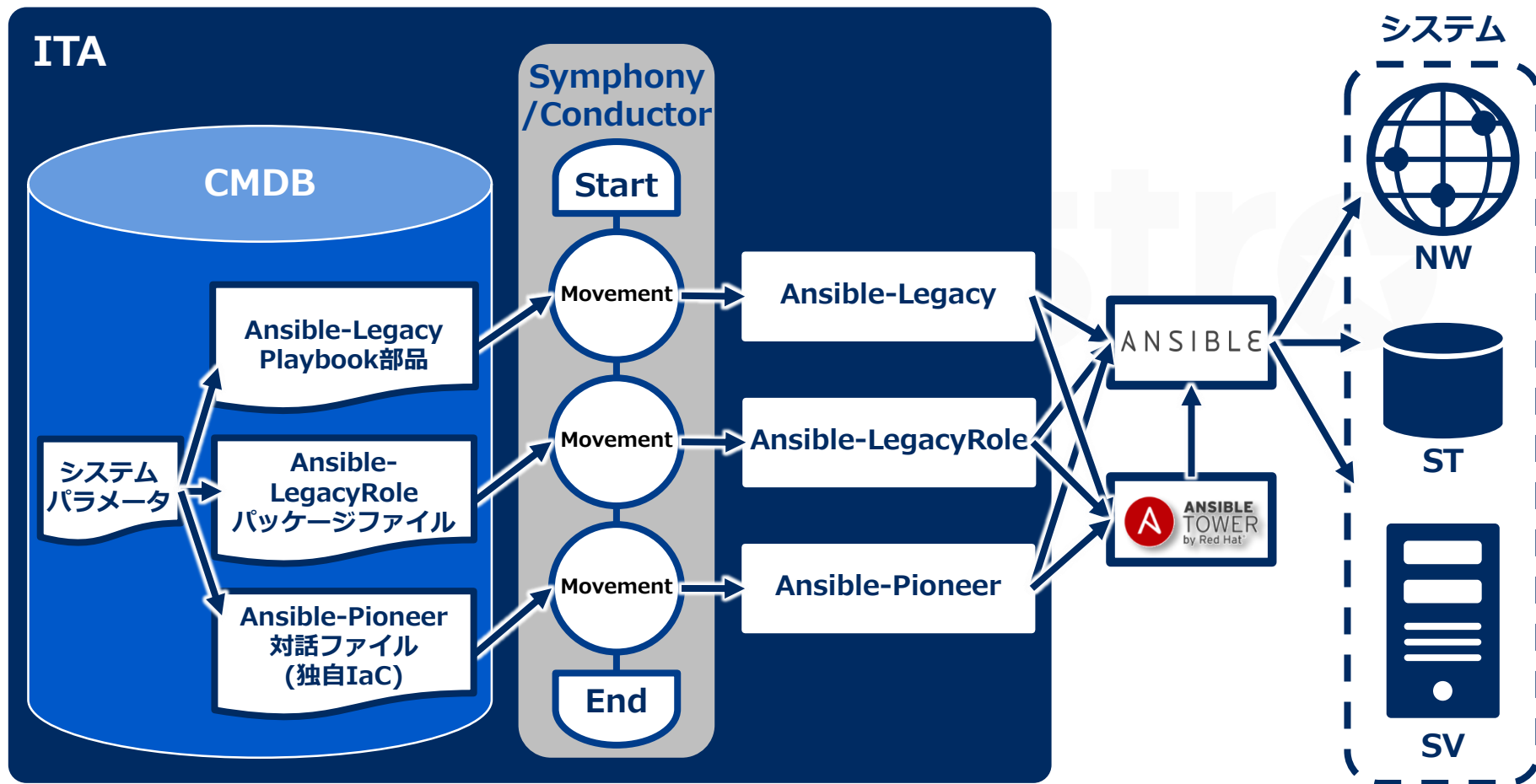
- 本書では、メニューグループの「Ansible-Legacy」「Ansible-LegacyRole」「Ansible-Pioneer」について、概念、機能説明を目的としております。
- 実習編ではITAの画面を用いて説明しておりますので合わせてご覧ください。



## 2. Ansible Driverとは

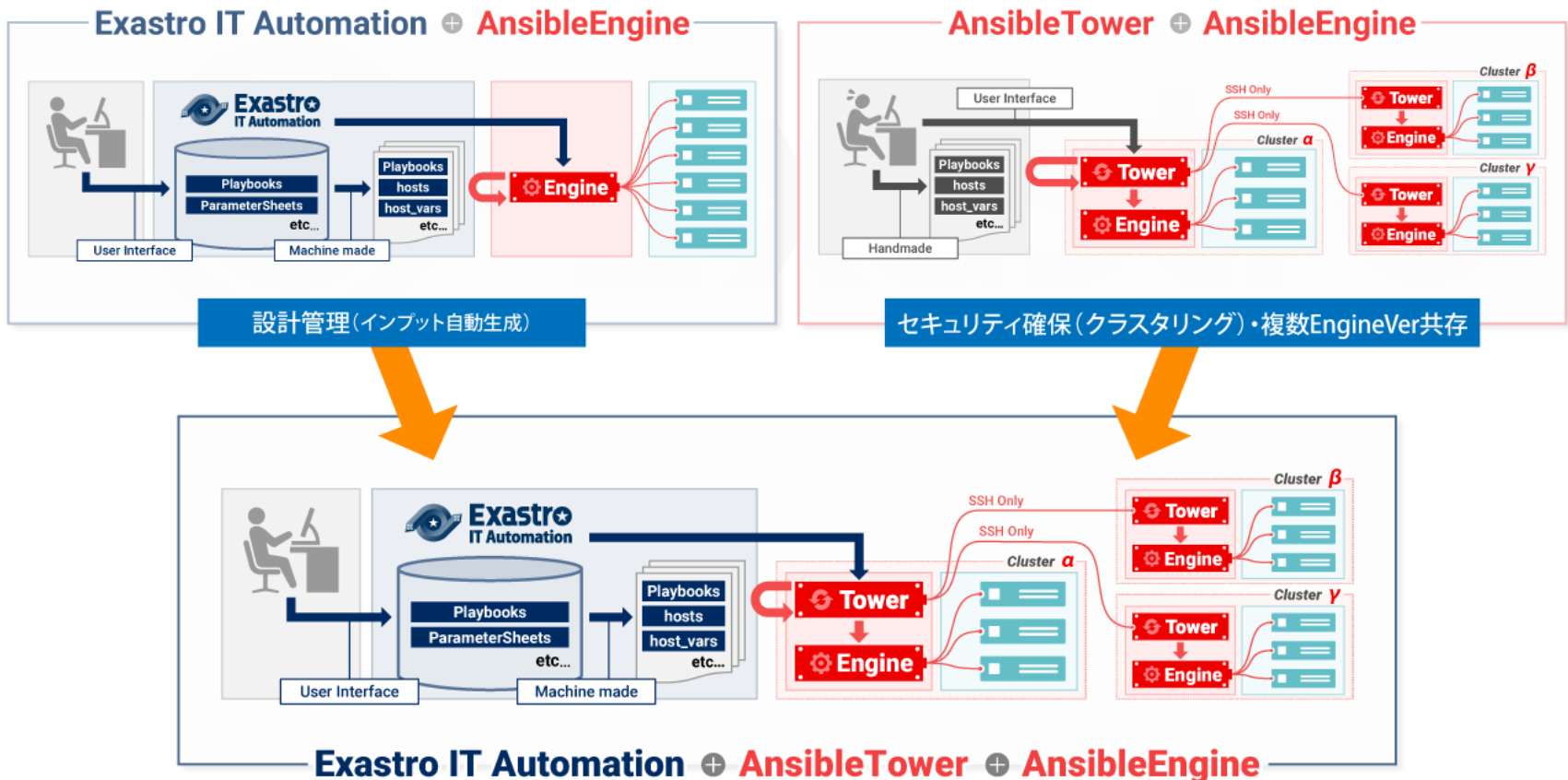
**Ansible DriverはITAが一元管理するシステムパラメータとIaC(Playbook等)の変数を紐づけて、Ansibleに連携実行させることが可能です**

※Ansible Towerを経由するメリットについては、「3. Ansible Towerとの連携」に記載します。



### 3. Ansible Towerとの連携

- IT Automationは設定データを蓄積/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は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。
- 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。



## 4. 3つのモードの説明

**Ansible Driverは用途に応じて特徴のある3つのモードを用意しています**

- Ansible-Legacy、Ansible-LegacyRole、Ansible-Pioneer  
それぞれの特徴の比較を以下に指します。

### ◆ IaCの再利用

1. **Ansible-Legacy**
2. Ansible-Pioneer
3. Ansible-LegacyRole

### ◆ ノウハウの活用

1. **Ansible-LegacyRole**
2. Ansible-Legacy
3. Ansible-Pioneer

### ◆ 適用範囲

1. **Ansible-Pioneer**
2. Ansible-LegacyRole
3. Ansible-Legacy

#### ※凡例

- ◎ : 機能として強みを持っている
- : 機能として使用は可能
- × : 機能として適用は難しい/他モードによる適用

	説明	Ansible-Legacy	Ansible-LegacyRole	Ansible-Pioneer
IaCの再利用	作成したPlaybookをモジュール化し、Exastro内で再利用可能出来ること	◎	×	○
ノウハウの活用	Ansibleが提供する機能を数多く活用でき、 またAnsible-galaxy等で公開されているPlaybookRoleを利用できること	○	◎	×
適用範囲	自動化できる作業手順のバリエーションのこと	○	○	◎

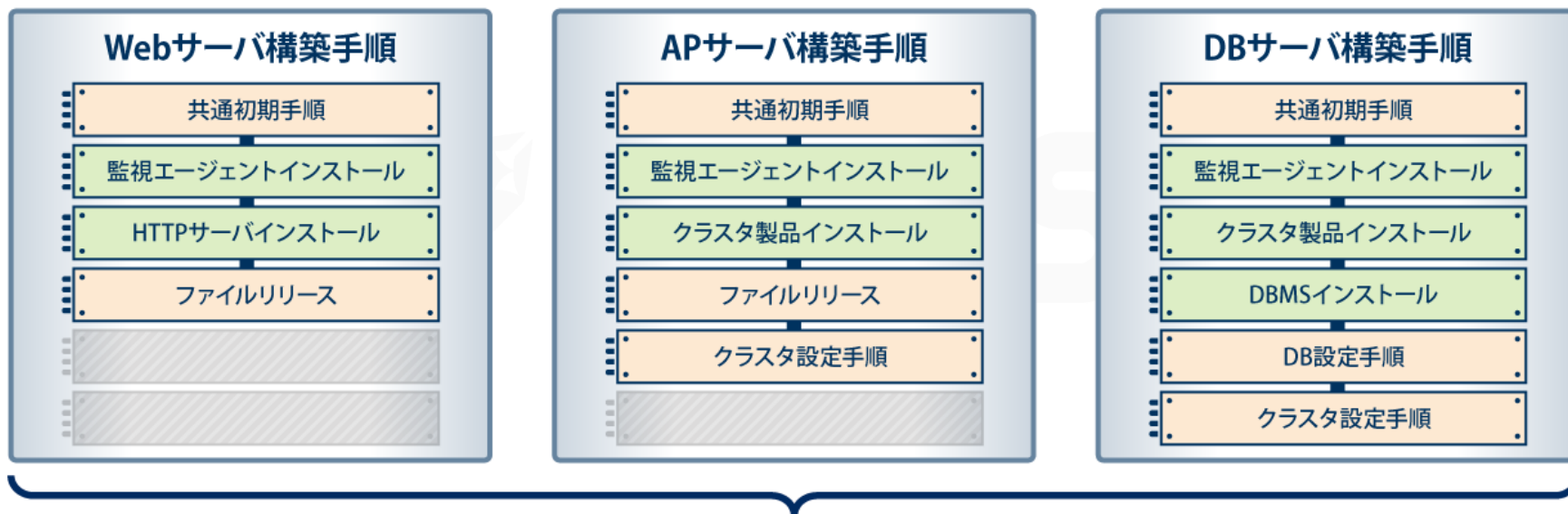
## 5. 各モードの特徴

### 5.1 Ansible-Legacyモード

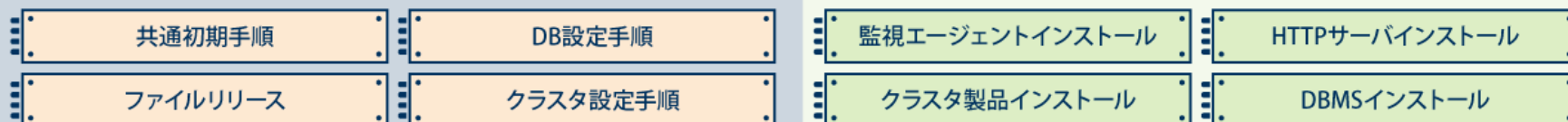
## 5.1 Ansible-Legacyモード (1/5)

### ITAのベースにして醍醐味 - Ansible-Legacyモード

- Ansible-Legacyモードの最たる特徴はIaCのモジュール化による再利用です。
- 登録したIaCを再利用することで効率的なシステム構築が可能です。



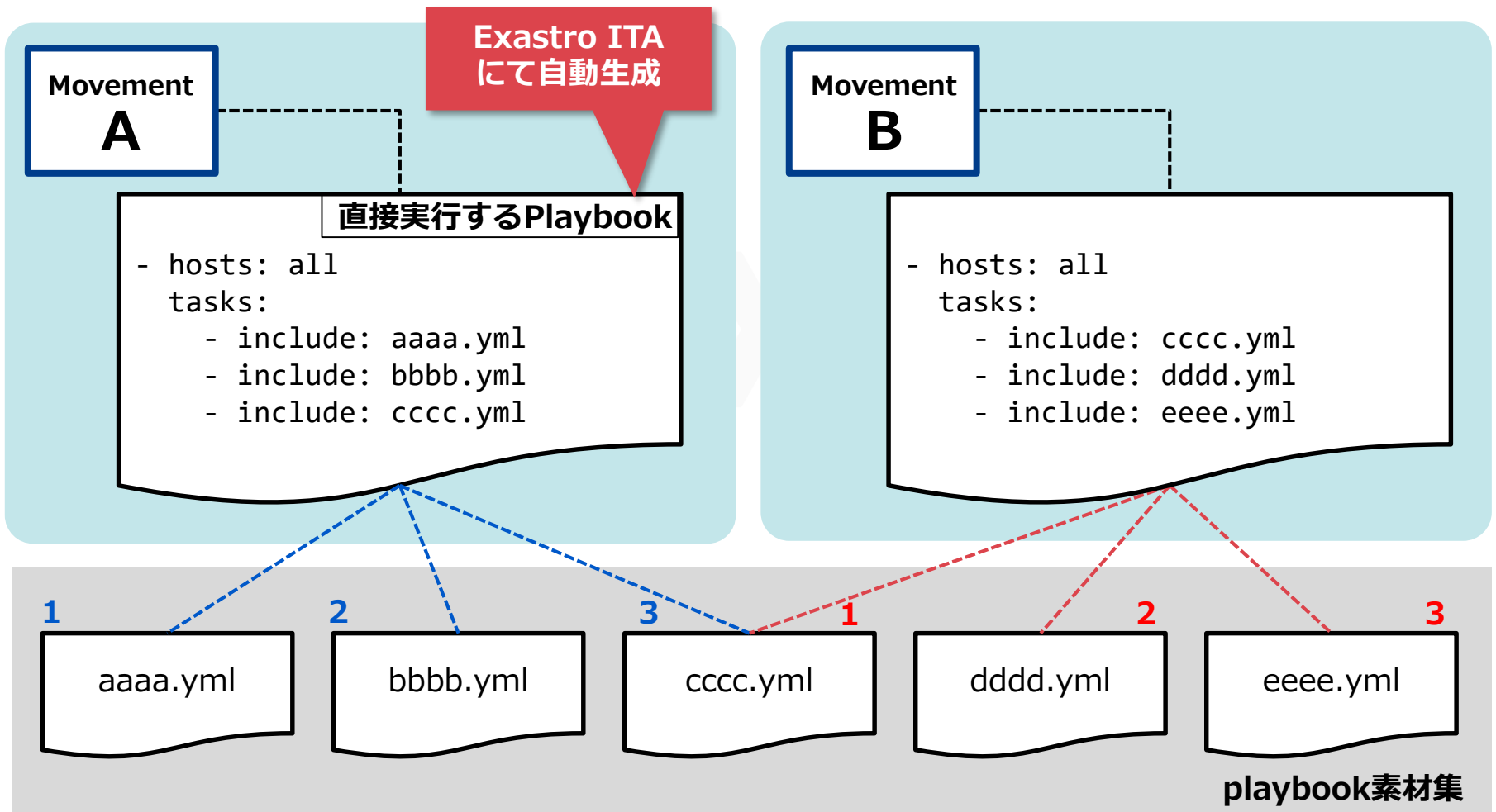
共通の手順はモジュール化し再利用できるように管理する





## 5.1 Ansible-Legacyモード (2/5)

Exastro ITAにおける作業実行単位である「Movement」とPlaybookの関係を2階層で規定しています。



## 5.1 Ansible-Legacyモード (3/5)

作業実行時、変数に与えるパラメータはExastro ITAのパラメータシートにて管理できます。

Movement

A

直接実行するPlaybook

```
- hosts: all
  tasks:
    - include: aaaa.yml
    - include: bbbb.yml
    - include: cccc.yml
```

パラメータシート

hosts	OPERATON_ID	para1	para2	para3	para4	...
host_a	888	AAAA	1	100	1	
host_b	111	BBBB	2	78	0	
host_c	888	CCCC	3	93	0	
⋮						

パラメータシートと  
変数を紐づけ、  
playbookへ設定

代入値管理

Movement	パラメータ	変数
A	para1	VAR_1
	para2	VAR_2
⋮	...	..

1

aaaa.yml

```
- name: "set aaaa"
  debug: msg= "{{ VAR_4 }}"
- command: sleep {{ VAR_2 }}
```

2

bbbb.yml

```
- name: "set bbbb"
  debug: msg= "{{ VAR_5 }}"
- command: sleep {{ VAR_3 }}
```

3

cccc.yml

```
- name: "set cccc"
  debug: msg= "{{ VAR_1 }}"
- command: sleep {{ VAR_2 }}
```

## 5.1 Ansible-Legacyモード (4/5)

ITAを**使用する際に意識する必要はありません**が、バックグラウンドでどのように動作しているかを補足します。

### 作業実行ディレクトリ

```
in
├── hosts
├── playbook.yml
├── child_playbooks
│   ├── aaaa.yml
│   ├── bbbb.yml
│   └── cccc.yml
├── host_vars
│   ├── host_a
│   │   └── main.yml
│   ├── host_c
│   │   └── main.yml
│   └── host_f
│       └── main.yml
├── ...
└── out
```

### パラメータシート

hosts	OPERATION_ID	para1	para2	para3	para4	...
host_a	888	AAAA	1	100	1	
host_b	111	BBBB	2	78	0	
host_c	888	CCCC	3	93	0	
host_d	222	DDDD	10	78	0	
host_e	333	EEEE	5	84	1	
host_f	888	FFFF	4	80	0	
host_g	111	GGGG	3	90	1	
⋮						

- **hosts** : 今回操作対象とするホスト一覧 (OPERATION\_ID=888)
- **playbook.yml** : 直接実行するPlaybook
- **child\_playbooks** : 今回使用するPlaybook素材集を格納
- **host\_vars** : ホストごとに異なる変数を定義したPlaybookを格納

※OPERATION\_IDについてはLearn : [BASE](#)をご確認ください

## 5.1 Ansible-Legacyモード (5/5)

### メニュー機能説明

- **Movement一覧**

Movementの作成、一覧の確認が可能です。

- **プレイブック素材集**

IaCの登録、一覧の確認が可能です。

- **Movement詳細**

Movementにインクルードするプレイブックの管理が可能です。

- **代入値自動登録設定**

登録されているオペレーションとホスト毎の項目の設定値を紐付ける。Movementと変数の管理が可能です。

- **作業対象ホスト**

オペレーションに紐づくMovement、ホストの管理が可能です。

- **代入値管理**

Movementで使用するプレイブックや変数「VAR\_」に代入する値の管理を行えます。

- **作業実行**

作成したMovementの単体実行が可能です。

- **作業状態確認**

実行したMovementの詳細確認が可能です。

- **作業管理**

作成、実行したMovementの作業一覧、履歴の一覧が確認可能です。



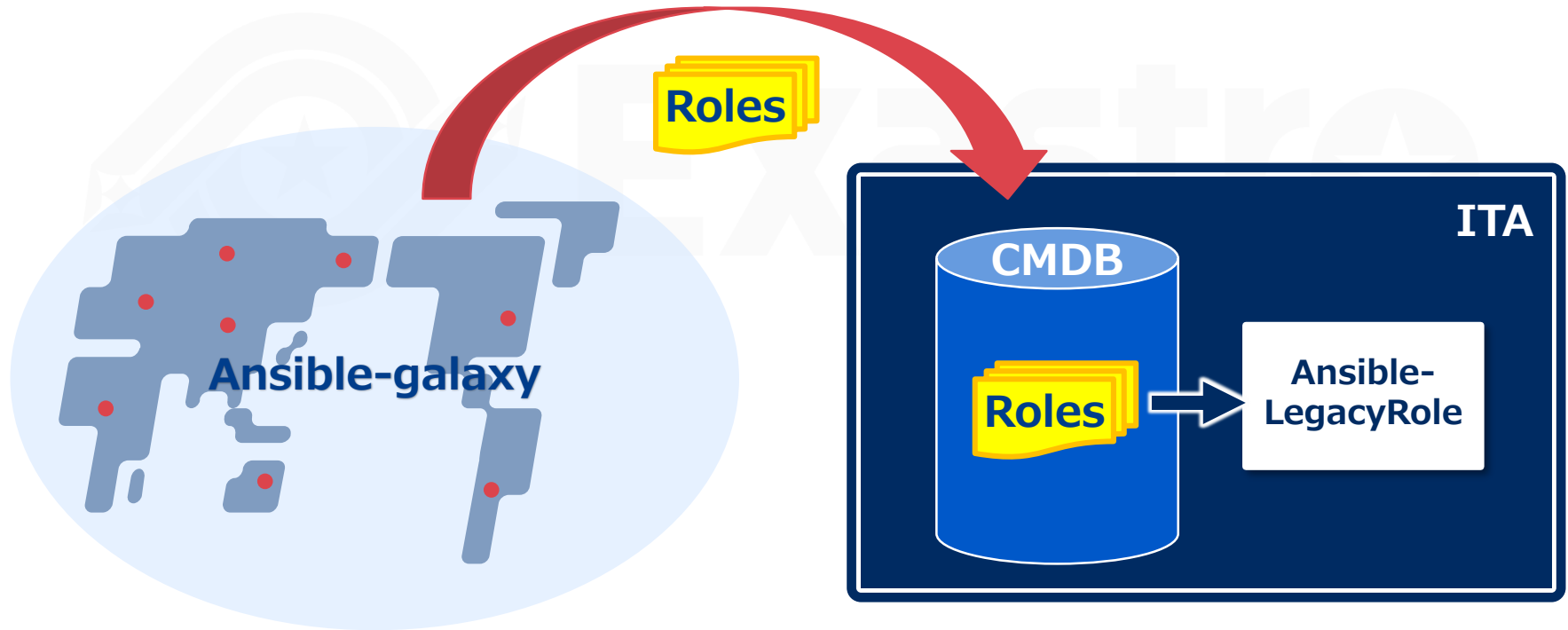
## 5. 各モードの特徴

### 5.2 Ansible-LegacyRoleモード

## 5.2 Ansible-LegacyRoleモード (1/4)

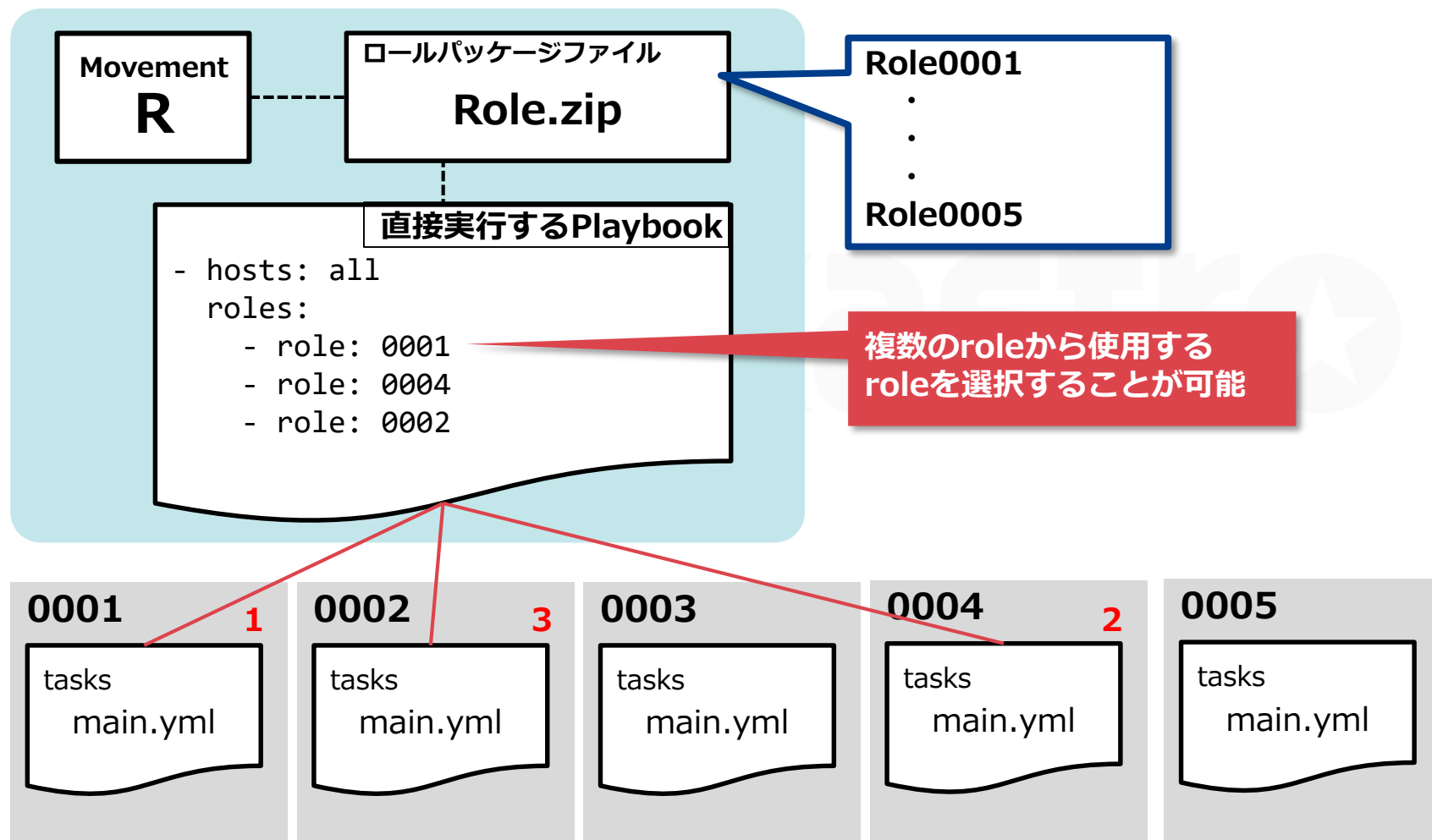
### 世界中の英知をその手に - Ansible-LegacyRoleモード

- Ansible-Legacyモードの最たる特徴はロールパッケージを登録、利用が可能な点です。
- 自身が作成、またはAnsible-galaxy上から取得したRoleを使用することができます。



## 5.2 Ansible-LegacyRoleモード (2/4)

Exastro ITAにおける作業実行単位である「Movement」とロールパッケージ内のroleを紐付けます。



## 5.2 Ansible-LegacyRoleモード (3/4)

- ITAを使用する際に意識する必要はありませんが、背景ではどのように動作をしているかを補足として記載します。

作業実行ディレクトリ

```
in
├── hosts
├── site.yml
├── host_vars
│   ├── host_a
│   └── host_b
├── roles
│   ├── role①
│   │   ├── defaults
│   │   │   └── main.yml
│   │   ├── tasks
│   │   │   └── main.yml
│   │   └── template
│   │       └── xxxx.yml
│   ├── role②
│   │   └── .
│   └── .
└── out
    └──
```

ロールパッケージファイルはrolesフォルダのあるディレクトリを圧縮してzipファイルにすること

実行roleディレクトリ名がそのままsite.yml (直接実行するplaybook)に記載される

- **hosts** : 今回操作対象とするホスト一覧(ITA作成)
- **site.yml** : 直接実行するPlaybook (ITA作成)
- **host\_vars** : ホストごとに異なる変数を定義したPlaybookを格納 (ITA作成)
- **roles** : playbookを実行するrole名ごとに格納

⇒roles配下の各ファイル

- **defaults** : playbook内の可変部に与えるパラメータを記載
- **tasks** : 実行playbook
- **template** : 実行playbook内で使用するテキストファイル

※左図ディレクトリ構成はあくまで一例



## 5.2 Ansible-LegacyRoleモード (4/4)

### メニュー機能説明

(Ansible-Legacyモードとの相違点を説明します)

- **ロールパッケージ管理**

作成したロールパッケージファイルの管理が可能です。

- **変数ネスト管理**

ロールパッケージにて定義されている多段変数のうち、繰返配列されている変数配列の最大繰返数の管理が行えます。



## 5. 各モードの特徴

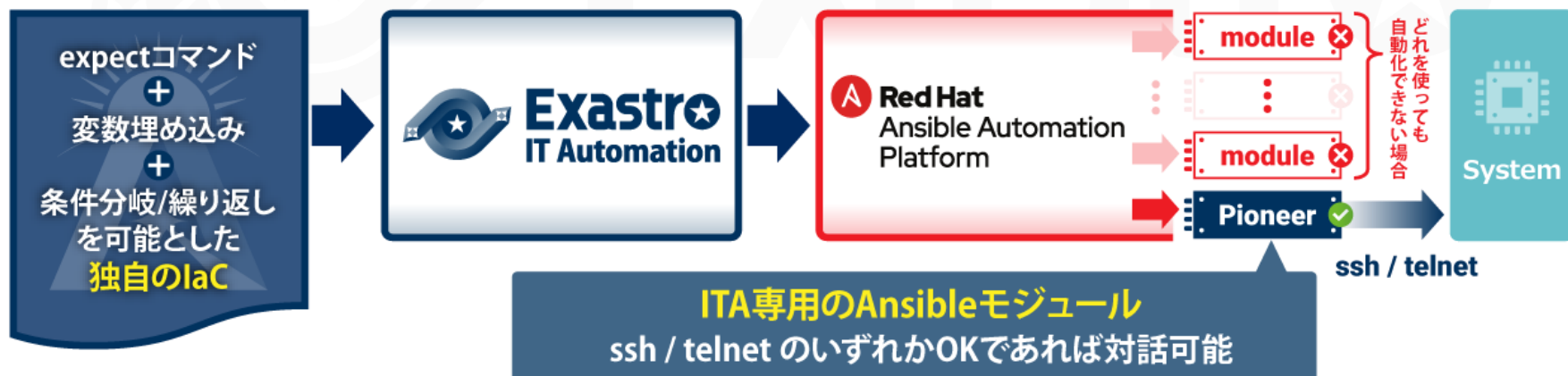
### 5.3 Ansible-Pioneerモード

## 5.3 Ansible-Pioneerモード (1/5)

### 自動化を止めない最後の切り札 - Ansible-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します。  
そこで、自動化を止めない最後の切り札として、ITAではPioneerモードをご用意しています。

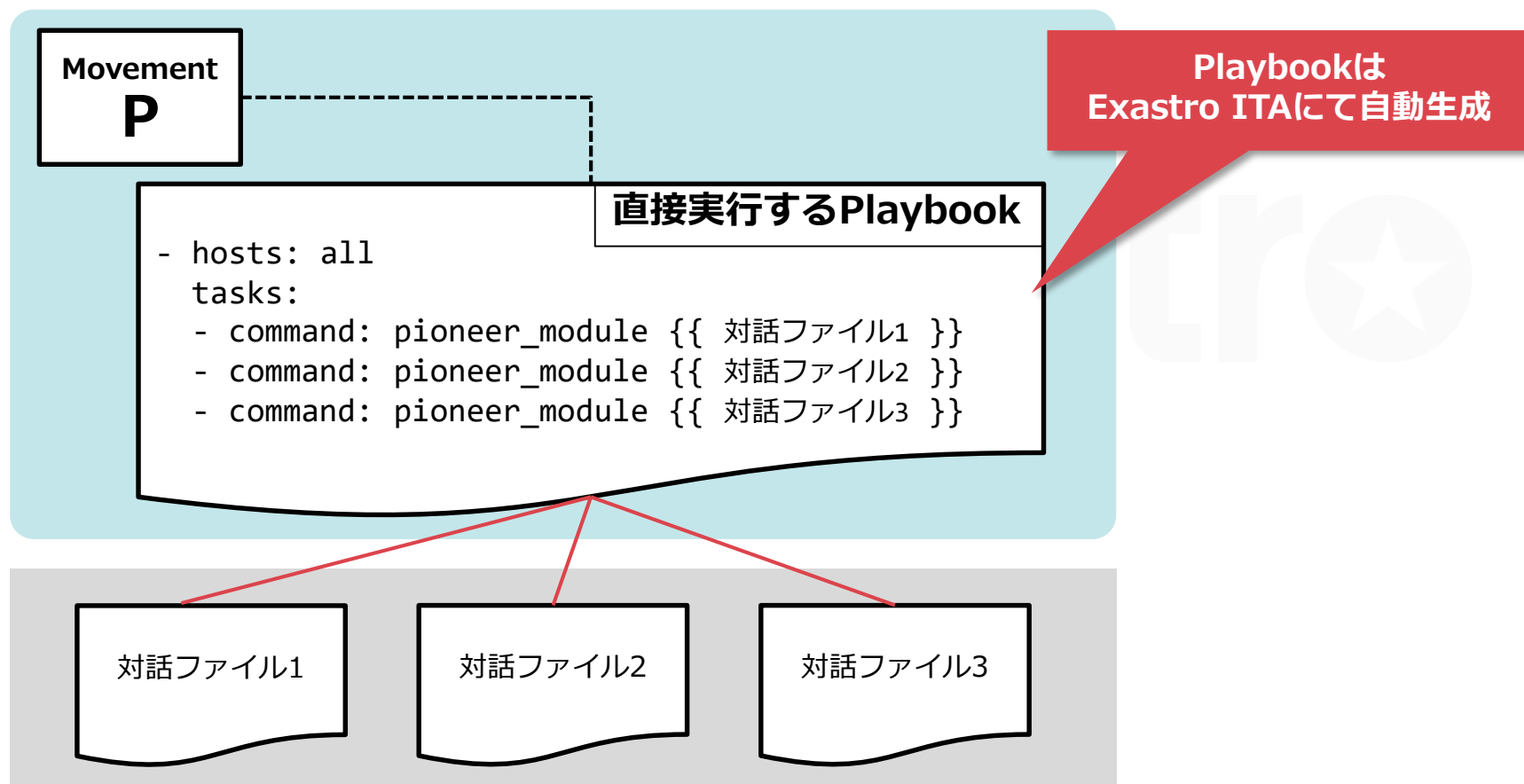
#### ▼ Pioneer専用「対話ファイル」



## 5.3 Ansible-Pioneerモード (2/5)

Pioneerでは直接実行するPlaybookからPioneerモジュール(ITA独自モジュール)を使って対話ファイル(※)を順番で実行します。

※対話ファイルについては次スライドで説明。



## 5.3 Ansible-Pioneerモード (3/5)

Ansible-Pioneer では、ターゲットへの設定を対話形式で記述することができます。また単純なexpectコマンドと比較して繰り返し、条件分岐を使えるなど、より高度な対話を表現することが可能です。

※対話ファイルの詳細はこちらの[マニュアル](#)を参照してください。

### 対話ファイル「テンプレート」記述例

1. 対象のシステムにログインし、変数にて指定したサービスのステータスを確認します。
2. 確認したステータスが「disable」の場合エラー終了処理を行います。ステータスが「disable」以外の場合、プロンプトに「complete!」と出力します。

※対話ファイル内「赤字」はパラメータシートを参照する変数を表現しています。

```
01 - expect: 'password:'
02   exec: "{{ ログインパスワード }}"
03 - command: 'systemctl status {{ item.0 }}'
04   prompt: '{{ ログインユーザ }}@'
05   with_items:
06     - '{{ サービス名1 }}'
07     - '{{ サービス名2 }}'
08   failed_when:
09     - stdout match(disable)
10 - command: 'echo complete!'
```

• 「with\_items」による  
繰り返し処理

• 「failed\_when」による  
分岐処理

## 5.3 Ansible-Pioneerモード (4/5)

Pioneerでは、「OS種別」と「対話種別」を設定することで、OS間の差異を意識しない作業実行が可能です。

- **OS種別**…対話ファイルと対象機器へ設定する。実行する対話ファイルの選択に用いる。
- **対話種別**… 同一目的の対話ファイルと紐づく。



設定時

Movement P

対話種別A

(ロードバランサに実サーバを登録する)

対話ファイルA OS種別: **BigIP**

```
- command: 'create / ltm node {{VAR_host_ip}} up'
prompt: '(tmsh)'
```

対話ファイルB OS種別: **Cisco ACE**

```
- expect: '{{ __loginhostname__ }}/Admin(config)'
exec: 'rserver {{ VAR_group_name }}'
- command: 'ip address {{ VAR_host_ip }}'
prompt: '{{ __loginhostname__ }}/Admin(config-rserver-host)'
```

対話ファイルC OS種別: **A10**

```
- command: 'slb server {{ VAR_server_name }} {{VAR_host_ip}}'
prompt: '(config)#'
```

OS種別: **BigIP**

OS種別: **Cisco ACE**

OS種別: **A10**



実行時

Movement P

対話種別A

(ロードバランサに実サーバを登録する)

ターゲット機器のOS種別に従った対話ファイルを実行するため、実行ユーザは対象機器のOSを意識する必要がない。

## 5.3 Ansible-Pioneerモード (5/5)

### メニュー機能説明

(Ansible-Legacyモードとの相違点を説明します)

- 対話種別リスト

対話種別をメンテナンス(閲覧/登録/更新/廃止)できます。

- OS種別マスタ

OS種別をメンテナンス(閲覧/登録/更新/廃止)できます。

- 対話ファイル素材集

OS種別ごとの対話ファイルの管理が可能です。





**Exastro**