ITA_利用手順マニュアル

Terraform-CLI-driver

ー第1.11版ー

免責事項

本書の内容はすべて日本電気株式会社が所有する著作権に保護されています。
本書の内容の一部または全部を無断で転載および複写することは禁止されています。
本書の内容は将来予告なしに変更することがあります。
日本電気株式会社は、本書の技術的もしくは編集上の間違い、欠落について、一切責任を負いません。
日本電気株式会社は、本書の内容に関し、その正確性、有用性、確実性その他いかなる保証もいたしません。

商標

- LinuxはLinus Torvalds氏の米国およびその他の国における登録商標または商標です。
- Red Hatは、Red Hat, Inc.の米国およびその他の国における登録商標または商標です。
- Apache、Apache Tomcat、Tomcatは、Apache Software Foundationの登録商標または商標です。
- Terraformは、HashiCorpの登録商標または商標です。

その他、本書に記載のシステム名、会社名、製品名は、各社の登録商標もしくは商標です。
なお、® マーク、TM マークは本書に明記しておりません。
※本書では「Exastro IT Automation」を「ITA」として記載します。

目次

はじめに

本書では、ITA の機能および操作方法について説明します。

1 Terraform-CLI-driver 概要

本章では Terraform および Terraform-CLI-driver について説明します。

1.1 Terraform について

Terraform とは HashiCorp 社が提供するインフラストラクチャを効率化するオーケストレーションツールです。

HCL(HashiCorp Configuration Language)という言語でコード化したインフラストラクチャ構成について、実行計画を生成したうえで構築を実行します。

Terraform-CLI-driver では、HashiCorp 社が提供する Terraform CLI パッケージを ITA と同一サーバにインストールし、利用します。
Terraform の詳細情報については、Terraform の製品マニュアルを参照してください。

1.2 Terrform-CLI-driver について

Terraform-CLI-deriver は ITA システムのオプションとして機能し、ITA と同一のサーバにインストールした Terraform に対し、作業の実行（Plan / Apply）および作業ログの取得を行うことができます。
作業の実行（Plan / Apply）に利用する Module ファイルを ITA システム上で部品化し、再利用できるよう管理することができます。

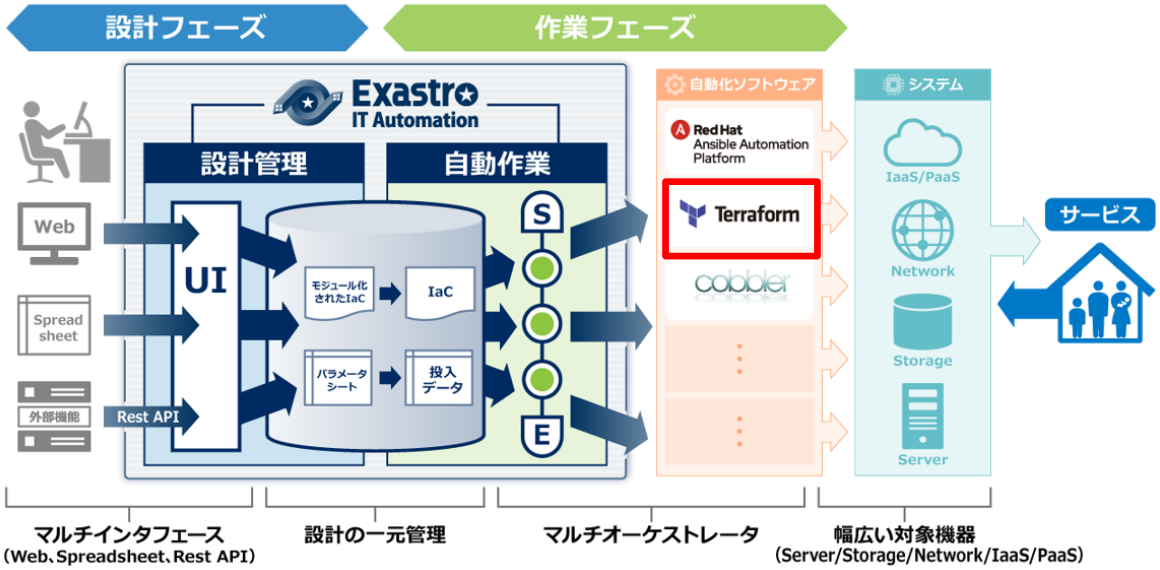


図 1.2-1 ITA システム概要

また、Terraform-CLI-driver は、Module 中の変数を画面から設定することができます。詳細は本書「2 Terraform-CLI-driver での変数取り扱い」をご参照ください。

2　Terraform-CLI-driver での変数取り扱い

2.1　変数の種類

Terraform-CLI-driver では、Module 中の変数の具体値を ITA の設定画面から設定することができます。

※設定方法の詳細は、本書「5.2.8 代入値管理」を参照してください。

Module 中の変数で、ITA の変数として扱える変数は以下の 1 種類があります。

種類	内容

通常変数	変数名に対して具体値を定義できる変数です。
	Module 内の変数は HCL（HashiCorp Configuration Language）の変数ルールに従い以下の形式で記述してください。この場合「xxx」が Module から変数として抜出されます。
	また、type と default 値を設定することができます。
	この場合、「〇〇」が type として、「△△」が default として抜出されます。
	type と default の設定は必須ではありません。
	variable “xxx” {
	type = 〇〇
	default = △△
	～省略~
	}

2.2　変数の抜出および具体値登録

ITA にアップロードされた Module 素材から変数を抜出して具体値を登録できます。

抜出した変数の具体値は「5.2.7 代入値自動登録設定」「5.2.8 代入値管理」で具体値を登録します。
登録された変数と具体値は、作業実行時に生成される terraform.tfvars ファイルに「変数名」が「Key」、「具体値」が「Value」として記載され、作業実行で使用されます。

2.3 変数のタイプについて

変数内で type を設定することができます。

Module 内の変数は HCL（HashiCorp Configuration Language）の変数ルールに従い記述してください。
ITA 内で扱う変数は以下の通りです。
記入例は「8.1 Module 素材記入例」をご参照ください。

type	詳細	代入順序 対象※1	メンバー変数 対象※2	type の 記述例	default の記述例
string	文字列型。	×	×	string	あいう
number	数値型。	×	×	number	2022
bool	true または false	×	×	bool	true
list	配列型。	○	×	list(string)	[“あ”, “い”, “う”]
set	配列型。ユニークな値の設定が求められる。 ITA 上では具体値がユニークか否かは判定されません。	○	×	set(number)	[1, 2, 3]
tuple	配列型。予め n 番目にどの type を設定するか決めておく必要があります。 値の入力数が決められているので ITA 上ではメンバー変数としてプルダウンで選択します。	×	○	tuple([string, number])	[“あいう”, 2022]
map	key-value 型。ITA 上では map 型が一つ以上含まれている type を設定した場合、type 情報から KEY 値を特定できないので、代入値を設定する場合は HCL 設定を ON にしてください。 HCL 設定についての説明は「5.2.7 代入値自動登録設定」または「5.2.8 代入値管理」をご参照ください。	×	×	map(string)	{ “key” = “value” }

object	key-value 型。
	ITA 上では key をメンバー変数として扱います。
	key 名に日本語は含まないでください。
		×	○	object({
				key = number
				})
					{
					“key” = 2022
					}

any	すべてに適合する型。 ITA 上では string 型と同じ扱いとなります。	×	×	any	あいう
記載なし	type を記載しなかった場合、ITA 上では string 型と同じ扱いになります。	×	×		あいう

※１…代入順序
変数に複数具体値を設定する際の先頭から代入する順序。
変数または階層構造の変数の最下層の変数のタイプが list,set の場合、代入値自動登録設定メニュー/代入値管理メニューにて設定可能。

例：変数のタイプが list の場合
・tf ファイルと登録値

```
variable "VAR_hoge" {
      type = list(string)
}
```

1.　代入値例（代入値自動登録設定/代入値管理）

項番	変数名	メンバー変数	代入順序	具体値
1	VAR_hoge	入力不要	1	あいう
2	VAR_hoge	入力不要	2	かきく

2. Terraform に送信される値

```
[“あいう”, “かきく”]
```

例：階層構造の変数の最下層の変数のタイプが set の場合

・tf ファイルと登録値

```
variable "VAR_hoge" {
        type = object({
                key = set(number)
        })
}
```

1.　代入値例（代入値自動登録設定/代入値管理）

項番	変数名	メンバー変数	代入順序	具体値
1	VAR_hoge	key	1	1
2	VAR_hoge	key	2	2

2. Terraform に送信される値

```
{
        key = [1, 2]
}
```

※2…メンバー変数
変数が key-value 型である場合の key 名。変数のタイプが object の場合、<KEY> = <TYPE>の<KEY>をメンバー変数とする。
変数のタイプが tuple の場合、tuple 内に定義した変数を先頭から[0],[1],[2]…と採番してメンバー変数とする。
変数のタイプが変数ネスト管理メニューの登録対象の場合、最大繰返数をもとに[0],[1],[2]…と採番してメンバー変数とする。
変数ネストに関しては「5.2.6 変数ネスト管理」をご参照ください。

例：変数のタイプが object の場合
・tf ファイルと登録値

```
variable "VAR_hoge" {
      type = object({
            NAME = string,
            IP = string
      })
      default = {
            “NAME” = “machine_01”,
            “IP” = “127.0.0.1”
      }
}
```

1． 代入値例（代入値自動登録設定/代入値管理）

項番	変数名	メンバー変数	代入順序	具体値
1	VAR_hoge	NAME	入力不可	my_machine
2	VAR_hoge	IP	入力不可	192.168.0.1

2. Terraform に送信される値

```
{
      NAME = “my_machine”,
      IP = “192.168.0.1”
}
```

例：変数のタイプが tuple の場合

・tf ファイルと登録値

```
variable "VAR_hoge" {
    type = tuple([string, number])
    default = [“aaa”, 2022]
}
```

2.　代入値例（代入値自動登録設定/代入値管理）

項番	変数名	メンバー変数	代入順序	具体値
1	VAR_hoge	[0]	入力不可	bbb
2	VAR_hoge	[1]	入力不可	2023

2. Terraform に送信される値

```
[“bbb”, 2023:
```

例：変数のタイプが変数ネスト管理対象の場合

・tf ファイルと登録値

```
variable "VAR_hoge" {
    type = list(set(string))
    default = [
        [“aaa”, “bbb”],
        [“ccc”, “ddd”]
    ]
}
```

1.　代入値例（代入値自動登録設定/代入値管理）

項番	変数名	メンバー変数	代入順序	具体値
1	VAR_hoge	[0]	1	あああ
2	VAR_hoge	[0]	2	いいい
3	VAR_hoge	[1]	1	ううう
4	VAR_hoge	[1]	2	えええ

2. Terraform に送信される値

```
[
    [“あああ”, “いいい”],
    [ “ううう”, “えええ”]
]
```

3 Terraform-CLI-driver コンソールメニュー構成

本章では、ITA コンソールのメニュー構成について説明します

3.1 メニュー/画面一覧

① ITA **基本コンソールのメニュー**

Terraform-CLI-driver で利用する ITA 基本コンソールのメニュー一覧を以下に記述します。

表 3.1-1 基本コンソール メニュー/画面一覧

No	メニューグループ	メニュー・画面	説明
1	ITA 基本コンソール	投入オペレーション 一覧	オペレーション一覧をメンテナンス(閲覧/登録/更新/廃止)できます。

② **Terraform-CLI-driver コンソールのメニュー**

Terraform-CLI-driver コンソールのメニュー一覧を以下に記述します。

表 3.1-2Terraform-CLI-driver コンソール メニュー/画面一覧

No	メニューグループ	メニュー・画面	非表示メニュー※1	説明
1	Terraform	インターフェース情報		作業実行の情報を管理します。
2		Workspaces 管理		Terraform で利用する Workspace の情報を管理します。
3		Movement 一覧		Symphony に登録する Movement の一覧を管理します。
4		Module 素材集		Module ファイルを管理します。
5		Movement-Module 紐付		Movement と Module 素材の関連付けを管理します。
6		変数ネスト管理		Module 素材集で登録した tf ファイルで定義されている変数のタイプが list,set かつ、その変数の中で list,set,tuple,object が定義されている場合、メンバー変数の最大繰返数を管理します。
7		代入値自動登録設定		パラメータシートのメニューに登録されているオペレーション毎の項目や値を紐付ける Movement と変数を管理します
8		代入値管理		変数の代入値を管理します。
9		作業実行		作業実行する Movement とオペレーションを選択し実行を指示します。
10		作業状態確認		作業実行状態を表示します。
11		作業管理		作業実行履歴を管理します。
12		Module 変数紐付管理	○	Module 変数と Module 素材の紐付を管理します。
13		メンバー変数管理	○	メンバー変数を管理します。
14		Movement 変数紐付管理	○	Movement と変数名の紐付を管理します。

※1 非表示メニューは、バックヤード機能でデータの登録・更新を行うメニューです。

Terraform-CLI-driver 機能をインストールした状態では表示されないメニューに設定されています。

非表示メニューを表示するには、「管理コンソール/ロール・メニュー紐付管理」で各メニューの復活処理を行います。詳細は「利用手順マニュアル_管理コンソール」 を参照してください。

尚、データの更新を行うとバックヤード機能が正しく動作しなくなります。データの更新はしないで下さい。

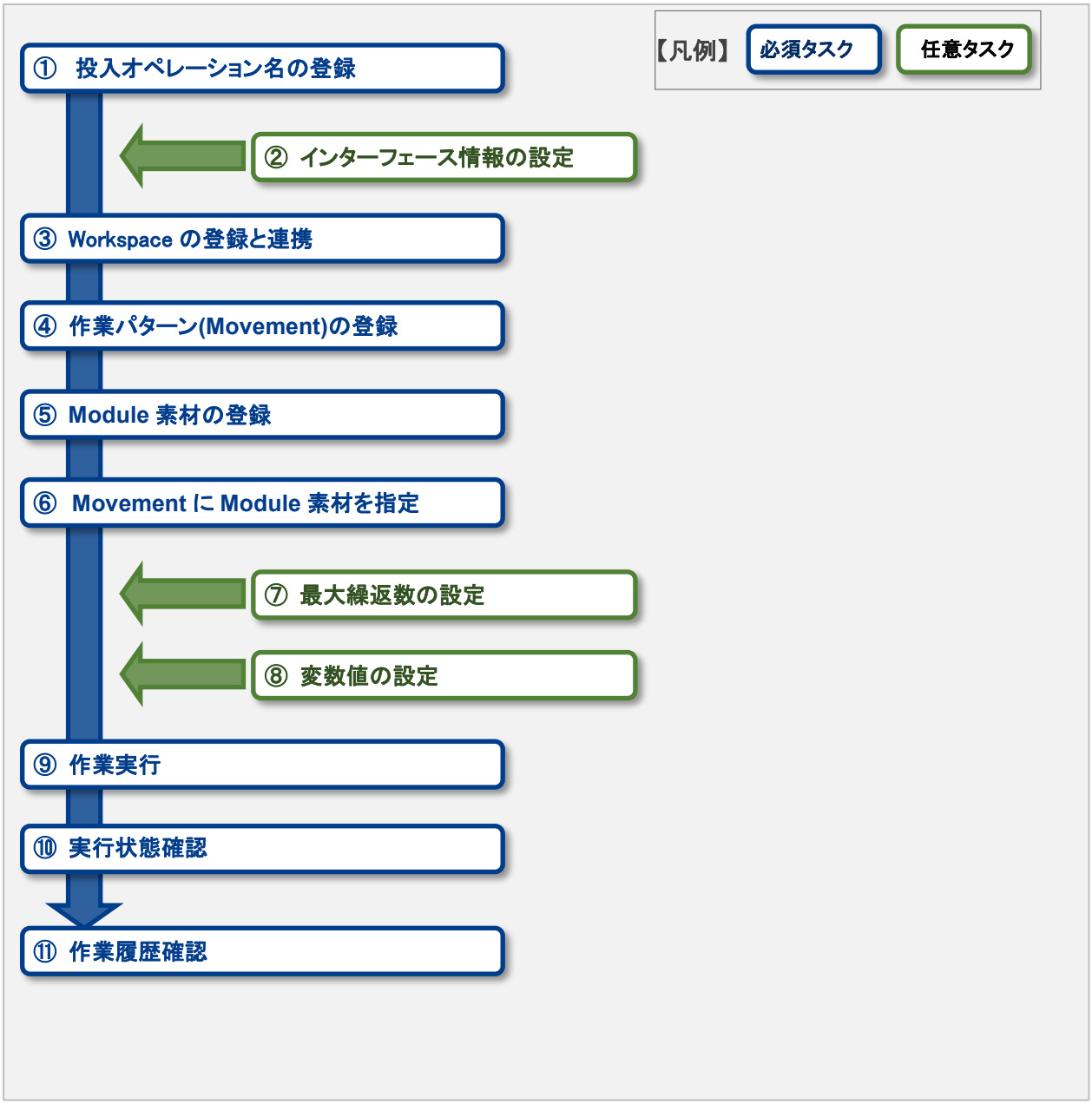
4　Terraform-CLI-driver 利用手順

各 Terraform コンソールの利用手順について説明します

4.1　Terraform 作業フロー

各 Terraform コンソールにおける標準的な作業フローは以下のとおりです。
各作業の詳細は次項に記載しています。
ITA 基本コンソールの利用方法は、「利用手順マニュアル_基本コンソール」を参照してください。

以下は、Terraform で作業を実行するまでの流れです。



- **作業フロー詳細と参照先**

① **投入オペレーション名の登録**
ITA 基本コンソールのオペレーション一覧の画面から、作業用の投入オペレーション名を登録します。
詳細は「5.1.1 オペレーション一覧」を参照してください。

② **インターフェース情報の設定**
作業実行の情報を設定します。
詳細は「5.2.1 インターフェース情報」を参照してください。

③ **Workspace の登録と連携**
Workspace の情報を登録します。
詳細は「5.2.2 Workspaces 管理」を参照してください。

④ **作業パターン(Movement)の登録**
作業用の Movement を登録します。
詳細は「5.2.3 Movement 一覧」を参照してください。

⑤ **Module 素材の登録**
作業で実行する Module ファイルを登録します。
詳細は「5.2.4 Module 素材集」を参照してください。

⑥ **Movement に Module 素材を指定**
登録した Movement に Module 素材を指定します。
詳細は「5.2.5 Movement-Module 紐付」を参照してください。

⑦ **最大繰返数の設定（必要に応じて実施）**
メンバー変数の最大繰返数を設定します。
詳細は「5.2.6 変数ネスト管理」を参照してください。

⑧ **変数値の設定（必要に応じて実施）**
Movement に登録した Module 素材内で定義した変数の値を設定します。変数を利用していない場合、設定は不要です。
詳細は「5.2.7 代入値自動登録設定」「5.2.8 代入値管理」を参照してください。

⑨ **作業実行**
実行日時、投入オペレーションを選択して設定して処理の実行を指示します。
詳細は「5.2.9 作業実行」を参照してください。

⑩ **作業状態確認**
実行した作業の状態がリアルタイムで表示されます。
また、作業の緊急停止や、実行ログ、エラーログを監視することができます。
詳細は「5.2.10 作業状態確認」を参照してください。

⑪ **作業履歴確認**
実行した作業の一覧が表示され履歴が確認できます。
詳細は「5.2.11 作業管理」を参照してください。

5　Terraform-CLI-driver 機能・操作方法説明

本章では、Terraform-CLI-driver で利用する各コンソールの機能について説明します。

5.1　基本コンソール

本節では、ITA 基本コンソールでの操作について記載します。
本作業は ITA 基本コンソールマニュアルを参照して、ITA 基本コンソール画面内で作業を実施してください。

5.1.1 オペレーション一覧

(1) [オペレーション一覧]画面では、オーケストレータで実行する作業対象ホストに対するオペレーションを管理します。作業は ITA 基本コンソール内メニューより選択します。

図　5.1.1- 1　サブメニュー画面（オペレーション一覧）

登録方法の詳細は、関連マニュアルの「利用手順マニュアル_基本コンソール」をご参照下さい。

5.2 Terraform-CLI-driver コンソール

本節では、Terraform-CLI コンソールでの操作について記載します。

5.2.1 インターフェース情報

(1) [インターフェース情報]では、ITA システムと連携する Terraform の作業実行時の情報をメンテナンス（閲覧／更新）することができます。

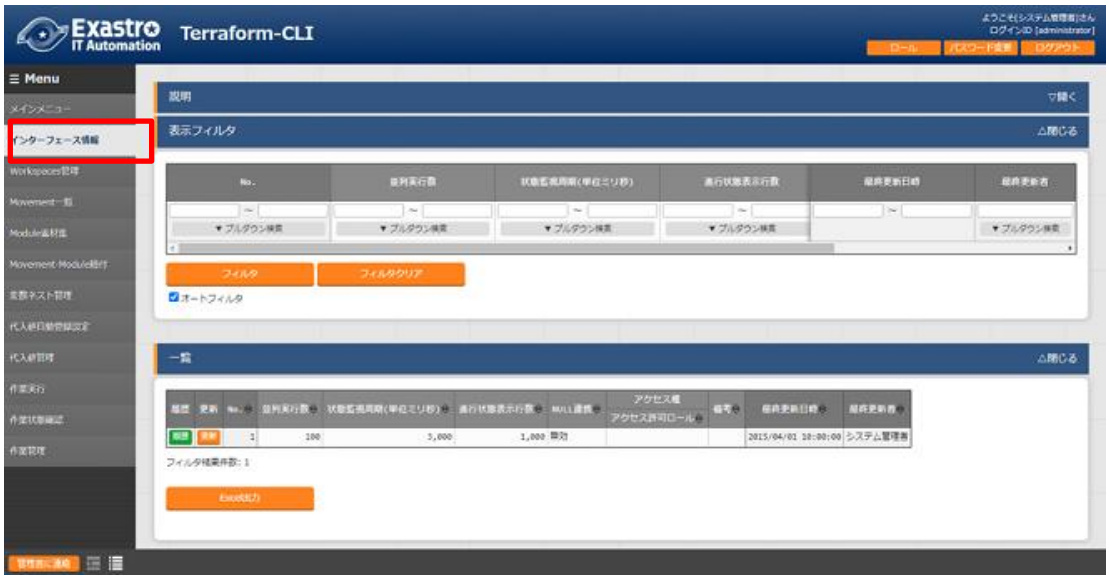


図 5.2.1- 1 サブメニュー画面（インターフェース情報）

(2) 「一覧」-「更新」ボタンより、インターフェース情報の登録を行います。
インターフェース情報が未登録または、複数レコード登録されている状態で作業実行した場合、**作業実行は想定外エラーとなります**。

図 5.2.1- 2 登録画面（インターフェース情報）

(3) インターフェース情報画面の項目一覧は以下のとおりです。

表 5.2.1-1 項目一覧（インタフェース情報）

項目	説明	入力必須	入力形式	制約事項
並列実行数	同時に実行できる Movement(Teraform-CLI)の最大数を入力します。	○	手動入力	
状態監視周期（単位ミリ秒）	「5.2.10 作業状態確認」で表示されるログのリフレッシュ間隔を入力します。通常は 3000 ミリ秒程度が推奨値です。	○	手動入力	最小値 1000 ミリ秒
進行状態表示行数	「5.2.10 作業状態確認」での進行ログ・エラーログの	○	手動入力	-

項目	説明	入力必須	入力形式	制約事項
	最大表示行数を入力します。 ステータスが[未実行]、[準備中]、[実行中]、[実行中(遅延)]の場合、指定した行数でログを出力します。 ステータスが[完了]、[完了(異常)]、[想定外エラー]、[緊急停止]、[未実行(予約)]、[予約取消]の場合、指定した行数ではなくすべてのログを出力します。 環境毎にチューニングを要しますが、通常は 1000 行程度が推奨値です。			
NULL 連携	代入値自動登録設定でパラメータシートの具体値がNULL(空白)の場合に 、代入値管理への登録 をNULL(空白)の値で行うか設定します。代入値自動登録設定メニューの「NULL 連携」が空白の場合この値が適用されます。 ・「有効」の場合、パラメータシートの値がどのような値でも代入値管理への登録が行われます。 ・「無効」の場合、パラメータシートに値が入っている場合のみ代入値管理への登録が行われます。	○	リスト選択	
備考	自由記述欄です。	-	手動入力	最大長 4000 バイト

5.2.2 **Workspaces 管理**

(1) [Workspaces 管理]では、Terraform で利用する Workspace についてのメンテナンス（閲覧／登録／更新／廃止／リソース削除）を行います。
Workspace は Terraform コマンドを実行するためのディレクトリとして利用します。
同一の Workspace を対象とした作業実行を行う場合、Terraform が生成する state ファイルは Workspace 単位で管理され、冪等性が保たれます。

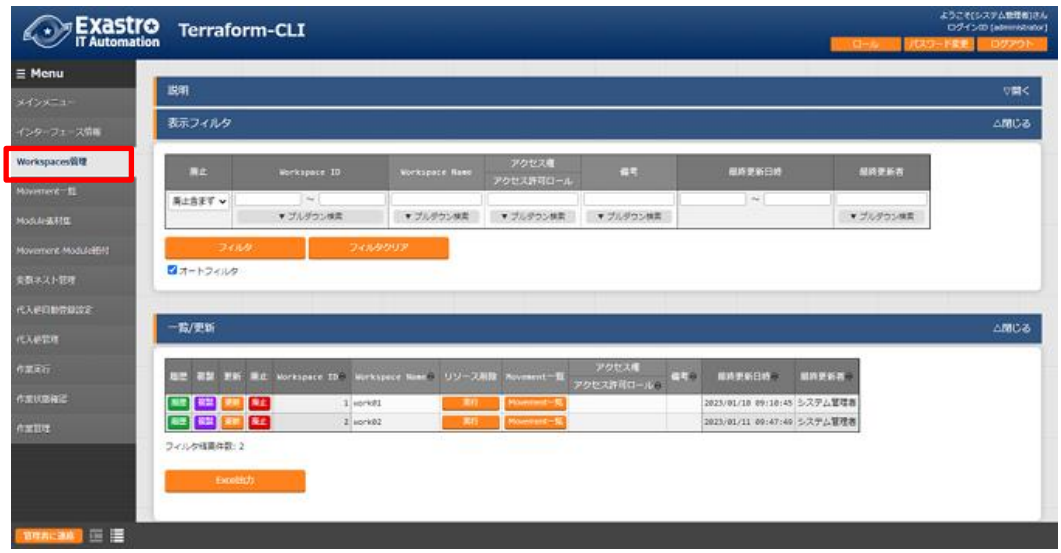


図 5.2.22-1 サブメニュー画面（Workspaces 管理）

(2) 「登録」-「登録開始」ボタンより、Workspace 情報の登録を行います。

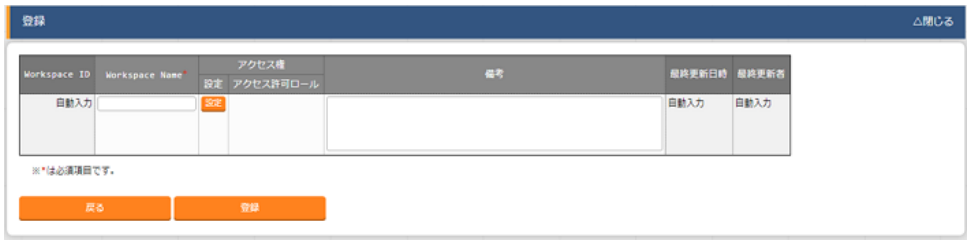


図 5.2.22-2 登録画面（Workspaces 管理）

(3) リソース削除ボタンをクリックすると、対象の Workspace に対してリソース削除(terraform destroy)が実行されます。
Movement 一覧ボタンをクリックすると、対象の Workspace の「5.2.3 Movement 一覧」へ遷移します。

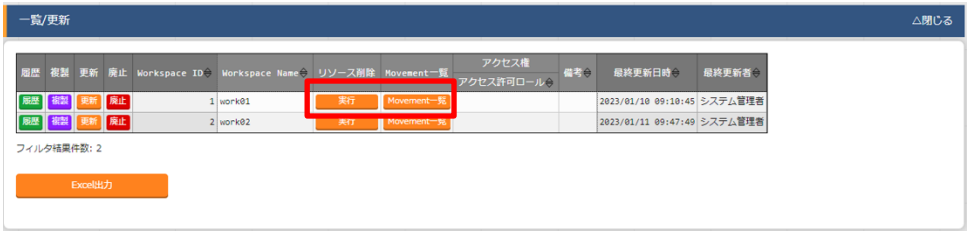


図 5.2.22-3 Workspaces 管理

(4) Workspaces 管理画面の項目一覧は以下のとおりです。

表 5.2.22-1 項目一覧(Workspaces 管理)

項目	説明	入力必須	入力形式	制約事項
Workspace Name	Workspace の名前を入力します。 半角英数字と記号 _ -（アンダーバーとハイフン）のみ利用可能です。	○	手動入力	最大長 90 バイト
リソース削除	Workspace ごとに構成・管理されたリソースの削除を実行するボタンです。 クリックすると確認ダイアログが表示され[OK]をクリックすると「5.2.10 作業状態確認」に遷移し、対象の Workspace ごとに構成・管理されたリソースの削除が実行されます。	-	-	
Movement 一覧	「5.2.3 Movement 一覧」へ遷移するボタンです。	-	-	
備考	自由記述欄です。	-	手動入力	最大長 4000 バイト

5.2.3 **Movement 一覧**

(1) [Movement 一覧]では、Movement 名についてのメンテナンス（閲覧／登録／更新／廃止）を行います。Movement は Terraform 利用情報として Workspace と紐付ける必要があるため、先に「5.2.2 Workspaces 管理」で対象を登録しておく必要があります。

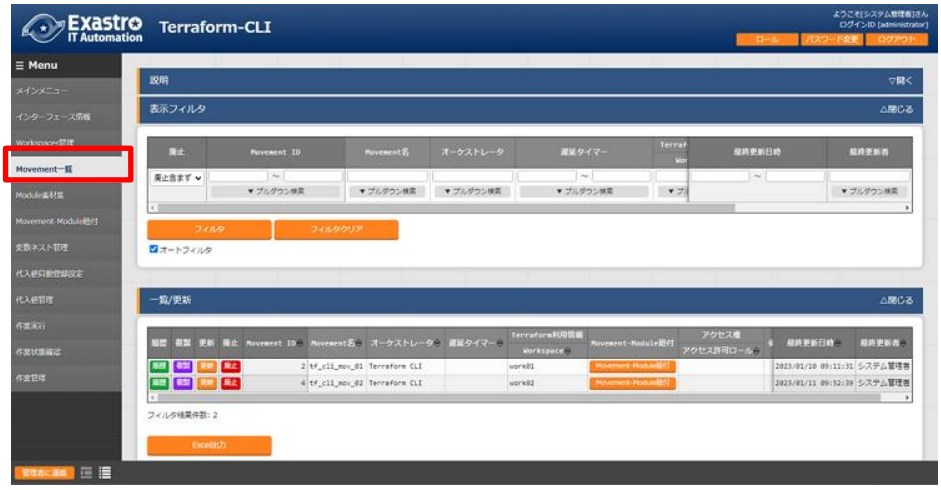


図 5.2.33-1 サブメニュー画面（Movement 一覧）

(2) 「登録」-「登録開始」ボタンより、Movement 情報の登録を行います。

図 5.2.33-2 登録画面（Movement 一覧）

(3) Movement-Module 紐付ボタンをクリックすると対象の Movement の「5.2.5 Movement-Module 紐付」へ遷移します。

図 5.2.33-3 サブメニュー画面（Movement 一覧）

(4) Movement 一覧画面の項目は以下の通りです。

表 5.2.33-1 項目一覧（Movement 一覧）

項目	説明	入力必須	入力形式	制約事項
Movement 名	Movement の名称を入力します。	○	手動入力	最大長 256 バイト

オーケストレータ		『Terraform CLI』が自動で入力されます。	-	-	-
遅延タイマー		Movement が指定期間遅延した場合にステータスを遅延として警告表示したい場合に指定期間(1～)を入力します。(単位:分) 未入力の場合は警告表示しません。	-	手動入力	-
Terraform 利用情報	Workspace	「5.2.2 Workspaces 管理」にて登録した Workspace を選択します。	○	リスト選択	
Movement-Module 紐付		「5.2.5 Movement-Module 紐付」へ遷移するボタンです。	-	-	
備考		自由記述欄です。	-	手動入力	最大長 4000 バイト

5.2.4 **Module 素材集**

(1) [Module 素材集]ではユーザーが作成した Module のメンテナンス（閲覧／登録／更新／廃止）を行います。
Module の記述などに関しては、「 6.1 Module の記述」を参照してください。

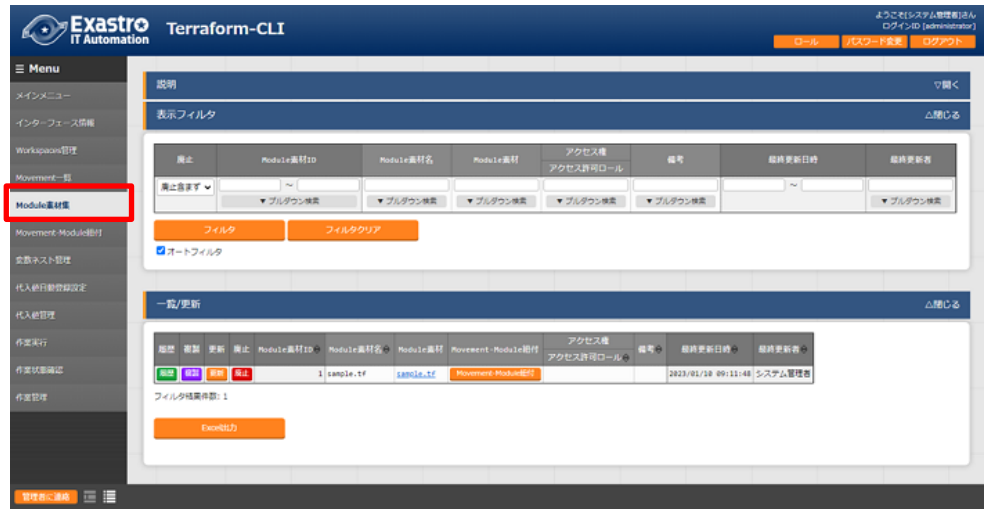
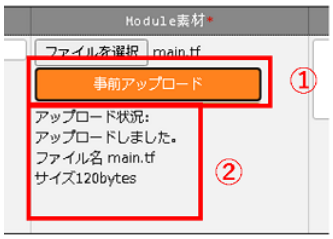


図 5.2.44-1 サブメニュー画面（Module 素材集）

(2) 「登録」-「登録開始」ボタンより、Movement 情報の登録を行います。

図 5.2.44-2 登録画面（Module 素材集）

「登録」の前に、「Module 素材」を「事前アップロード（①）」してください。「アップロード状況（②）」にModule のファイル名が表示されたのを確認してから、「登録」ボタンを押してください。



(3) Movement-Module 紐付ボタンをクリックすると対象の Movement の「5.2.5 Movement-Module 紐付」へ遷移します。

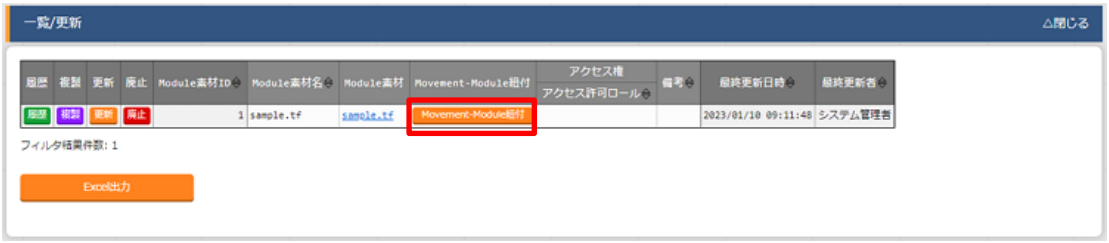


図 5.2.44-3 サブメニュー画面（Module 素材集）

(4) Module 素材集の項目一覧は以下のとおりです。

表 5.2.44-1 項目一覧（Module 素材集）

項目	説明	入力必須	入力形式	制約事項
Module 素材名	ITA で管理する Module 素材名を入力します。	○	手動入力	最大長 256 バイト
Module 素材	作成した Module ファイルをアップロードします。	○	ファイル選択	最大サイズ 4G バイト
Movement-Module 紐付	「5.2.5 Movement-Module 紐付」へ遷移するボタンです。	-	-	
備考	自由記述欄です。	-	手動入力	最大長 4000 バイト

(5) BackYard の処理で Module ファイル内に定義している変数を抜出します。抜出した変数は「5.2.7 代入値自動登録設定」や「5.2.8 代入値管理」で具体値の登録が可能になります。
抜出するタイミングはリアルタイムではありませんので「5.2.7 代入値自動登録設定」や「5.2.8 代入値管理」で変数が扱えるまでに時間がかかる※1場合があります。

※1 抜出のタイミングは「 7.2 メンテナンス方法について」の「③ 起動周期の変更」に記載していますので、そちらをご参照ください。

5.2.5 Movement-Module 紐付

(1) [Movement-Module 紐付]では、Movement で実行する Module 素材のメンテナンス（閲覧／登録／更新／廃止）を行います。
Movement に対して複数の Module 素材を紐付けることが可能です。

図 5.2.55-1 サブメニュー画面（Movement-Module 紐付）

(2) 「登録」-「登録開始」ボタンより、Movement-Module 紐付の登録を行います。

図 5.2.55-2 登録画面（Movement-Module 紐付）

(3) Movement のリンクをクリックすると、「5.2.3 Movement 一覧」へ遷移します。
また、Module 素材のリンクをクリックすると、「5.2.4 Module 素材集」へ遷移します。

図 5.2.55-3 サブメニュー画面（Movement-Module 紐付）

(4) Movement-Module 紐付の項目一覧は以下のとおりです。

表 5.2.55-1 項目一覧（Movement-Module 紐付）

項目	説明	入力必須	入力形式	制約事項
Movement	「5.2.3 Movement 一覧」にて登録した Movement を選択します。	○	リスト選択	-
Module 素材	「5.2.4 Module 素材集」で登録した Module 素材を選択します。	○	リスト選択	-
備考	自由記述欄です。	-	手動入力	最大長 4000 バイト

5.2.6 変数ネスト管理

(1) [変数ネスト管理]では、Module 素材集で登録した tf ファイルで定義されている変数のタイプが list,set かつ、その変数の中で list,set,tuple,object が定義されている場合、メンバー変数の最大繰返数を閲覧及び更新できます。
本メニューは Module 素材集を元に BackYard がレコードを管理するため、登録・廃止・復活はできません。
変数ネストの管理対象については「 8.1Module 素材記入例」をご参照ください。
変数ネストの管理フロー例については「 8.2 変数ネスト管理フロー例」をご参照ください。

図 5.2.66-1 サブメニュー画面（変数ネスト管理）

Module 素材と変数ネスト管理登録例

・Module 素材（tf ファイル）と登録値

```
variable "VAR_hoge" {
    type = list(
        object({
            IP = string,
            NAME = string
        })
    )
    default = [
        { “IP” = “127.0.0.1”, “NAME” = “machine_01” },
        { “IP” = “127.0.0.2”, “NAME” = “machine_02” }
    ]
}
```

・変数ネスト管理

項番	変数名	メンバー変数	最大繰返数
1	VAR_hoge		2

(2) 「一覧」-「更新」ボタンより最大繰返数の更新を行います。

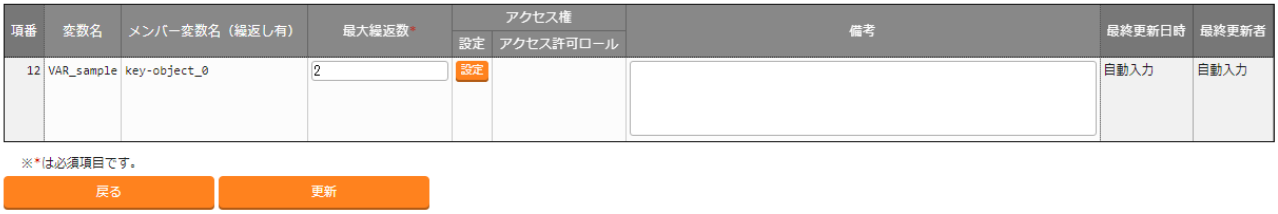


図 5.2.66-2 更新画面（変数ネスト管理）

(3) 変数ネスト管理の項目一覧は以下の通りです。

表 5.2.56-1 項目一覧（変数ネスト管理）

項目	説明	入力必須	入力形式	制約事項
変数名	Movement-Module 紐付で登録した資材で使用している変数が表示されます。	-	入力不可	
メンバー変数名（繰返し有）	変数ネスト管理対象がメンバー変数である場合、メンバー変数名が表示されます。メンバー変数名は各階層の変数を「.」で連結して表示します。	-	入力不可	
最大繰返数	配列の最大繰返数を 1～99,999,999 の範囲で入力します。 初期値は tf ファイルの default に記載されている値から取得した繰返数が設定されます。 tf ファイルに default の記載がない場合、1 が設定されます。 最終更新者が「Terraform 変数更新プロシージャ」でない場合は Module 素材の更新により値が変更されることはありません。	○	手動入力	入力値 1～99,999,999
備考	自由記述欄です。	-	手動入力	最大長 4000 バイト

※初期登録および繰返数の更新はリアルタイムではないので、「5.2.7 代入値自動登録設定」や「5.2.8 代入値管理」で変数が扱えるまでに時間がかかる場合があります。
初期登録および繰返数の更新タイミングは「 7.2 メンテナンス方法について」に記載していますので、そちらをご参照ください。

(4) アクセス許可ロール
変数ネスト管理に設定されるアクセス許可ロールは、該当の変数が定義されている Module 素材集のアクセス許可ロールが設定されます。

5.2.7　代入値自動登録設定

(1) [代入値自動登録設定]では、メニュー作成機能で作成したパラメータシート（オペレーションあり）と、Movement の変数を紐付けます。登録した情報は BackYard の処理により代入値管理に反映されます。
6.2 BackYard コンテンツ(2)代入値自動登録設定に反映ルールを記載しています。

図 5.2.67-1 サブメニュー画面（代入値自動登録）

(2) 「登録」-「登録開始」ボタンより、代入値管理を行います。

図 5.2.67-2 登録画面（代入値自動登録）

(3) メニューID または、メニュー名のリンクをクリックすると、対象のメニューへ遷移します。

図 5.2.67-3 サブメニュー画面（代入値自動登録）

(4) 登録画面の項目一覧は以下のとおりです。

表 5.2.7-1　項目一覧（代入値自動登録）

カラム		説明	入力必須	入力形式	制約事項
メニューグループ:メニュー		メニュー作成機能で作成したパラメータシート(オペレーションあり)が表示 されます。 該当のパラメータシートを選択します。	○	リスト選択	
項目		選択したパラメータシートの項目が表示されます。 対象の項目を選択します。	○	リスト選択	
登録方式		Value 型:項目の設定値を紐付けた変数の具体値とする場合に選択します。 Key 型:項目の名称を紐付けた変数の具体値とする場合に選択します。 項目の設定値が空白の場合は紐付け対象外となり ます。 Key-Value 型:項目の名称(Key)と設定値(Value)を 紐付けた変数の具体値とする場合に選択します。	○	リスト選択	
Movement		Movement 一覧で登録した Movement が表示されます。Movement を選択します。	○	リスト選択	
Key 情報	変数名	Movement-Module 紐付で登録した資材で使用している変数が表示されます。 Key 型で具体値に紐付けたい変数を選択します。	○	リスト選択	登録方式で Key 型 または Key-Value 型を選択した場合は必須
	メンバー変数	メンバー変数を持つ変数名を選択した場合にメンバー変数が表示されます。 メンバー変数を選択します。	変数名により変動	リスト選択	
	代入順序	複数具体値が設定できる変数名の場合のみ必須入力になります。 具体値の代入順序(1～)を入力します。入力値に従い昇順で代入されます。	変数名により変動	手動入力	ブランク または、 正の整数
Value 情報	変数名	Movement-Module 紐付で登録した資材で使用している変数が表示されます。 Value 型で具体値に紐付けたい変数を選択します。	○	リスト選択	登録方式で Value 型 または Key-Value 型を選択した場合は必須
	HCL 設定	「OFF」または「ON」を選択します。 BackYard の処理により代入値管理に反映する際、選択した値が引き継がれます。	○	リスト選択	選択した変数名が map 型の場合は ON で設定する必要があります。 オペレーション、Movement、変数名が一致している場合、HCL 設定は ON または OFF に統一してください。

	メンバー変数	メンバー変数を持つ変数名を選択した場合にメンバー変数が表示されます。 メンバー変数を選択します。	変数名により変動	リスト選択	
	代入順序	複数具体値が設定できる変数名の場合のみ必須入力になります。 具体値の代入順序（1～）を入力します。入力値に従い昇順で代入されます。	変数名により変動	手動入力	ブランク または、 正の整数
NULL 連携		パラメータシートの具体値が NULL(空白)の場合に、代入値管理への登録を NULL(空白)の値で行うか設定します。 ・「有効」の場合、パラメータシートの値がどのような値でも代入値管理への登録が行われます。 ・「無効」の場合、パラメータシートに値が入っている場合のみ代入値管理への登録が行われます。 ・空白の場合、インターフェース情報の「NULL 連携」の値が適用されます。	-	リスト選択	-
備考		自由記述欄です。	-	手動入力	最大長 4000 バイト

※登録方式で Key 型を選択した場合、代入値管理に反映する際に HCL 設定は OFF で設定されます。
※メンバー変数を設定する場合は、同じ変数内のメンバー変数の具体値も全て設定してください。
代入値を設定しなかった他のメンバー変数でもデフォルト値が使用されることはありません。

例
・tf ファイルと登録値

```
variable "VAR_hoge" {
    type = object({
        IP = string
        NAME = string
    })
    default = {
        IP = “127.0.0.1”
        NAME = “machine01”
    }
}
```

・代入値例（代入値自動登録設定/代入値管理）

項番	変数名	メンバー変数	代入順序	具体値
1	VAR_hoge	IP	入力不要	192.168.0.1
2	VAR_hoge	NAME	入力不要	

ここの値を設定する必要がある

5.2.8　代入値管理

(1)　[代入値管理]では、オペレーションごとに、対象の Movement で利用される Module 内の変数に代入する具体値をメンテナンス（閲覧／登録／更新／廃止）できます。

図 5.2.88-1 サブメニュー画面（代入値管理）

(2)　「登録」-「登録開始」ボタンより、代入値管理を行います。

図　5.2.88-2 登録画面（代入値管理）

代入値管理の変数は、「5.2.4 Module 素材集」で登録されたファイルの情報から反映されます。

※　反映のタイミングは「 7.2 メンテナンス方法について」の「③起動周期の変更」に記載していますので、そちらをご参照ください

(3)　代入値管理に登録した変数は、作業実行時に生成される terraform.tfvars ファイルに「変数名」が「Key」、「具体値」が「Value」として記載され、作業実行で使用されます。
「Secure 設定」を「ON」にしていた場合は作業実行時に生成される secure.tfvars ファイルに「変数名」が「Key」、「具体値」が「Value」として記載され、作業実行で使用されます。secure.tfvars ファイルは「5.2.10 作業状態確認」「5.2.11 作業管理」で取得できる投入データの中に格納されません。

(4) 代入値管理の項目一覧は以下のとおりです。

表 5.2.8-1 項目一覧（代入値管理）

項目	説明	入力必須	入力形式	制約事項
オペレーション	対象のオペレーションを選択します。	○	リスト選択	-
Movement	対象の Movement を選択します。	○	リスト選択	-
変数名	Movement-Module 紐付にて登録されている Module 素材の中から、選択された Movement にアタッチしている変数名が表示されます。変数を選択します。	○	リスト選択	-
HCL 設定	「OFF」または「ON」を選択します。 変数に文字列以外の値を設定する際などに「ON」を設定します。	○	リスト選択	選択した変数名が map 型の場合は ON で設定する必要があります。 オペレーション、Movement、変数名が一致している場合、HCL 設定は ON または OFF に統一してください。
メンバー変数	メンバー変数を持つ変数名を選択した場合に メンバー変数が表示されます。 メンバー変数を選択します。	変数名により変動	リスト選択	
代入順序	複数具体値が設定できる変数名の場合のみ必須入力になります。 具体値の代入順序（1～）を入力します。入力値に従い昇順で代入されます。	変数名により変動	手動入力	ブランク または、 正の整数
デフォルト値	default 内で変数に紐づいている具体値が表示されます。	-		
Secure 設定	「OFF」または「ON」を選択します。 「ON」を選択した場合、具体値を暗号化し ITA 上で表示させないようにします。 また「ON」にした対象は「5.2.10 作業状態確認」「5.2.11 作業管理」で取得できる投入データの中に格納されません。	○	リスト選択	Terraform が出力する Plan や Apply のログ上で値を非表示にする場合は、登録した Module 素材の variable ブロックの中で『sensitive = true』を指定する必要があります。
具体値	オペレーション/Movement で使用する変数の具体値を入力します。	○	手動入力	最大長 8192 バイト
備考	自由記述欄です。	-	手動入力	最大長 4000 バイト

5.2.9 作業実行

(1) 作業の実行を指示します。Movement 一覧、オペレーション一覧からそれぞれラジオボタンで選択し、実行ボタンを押すと、「5.2.10 作業状態確認」に遷移し、実行されます。

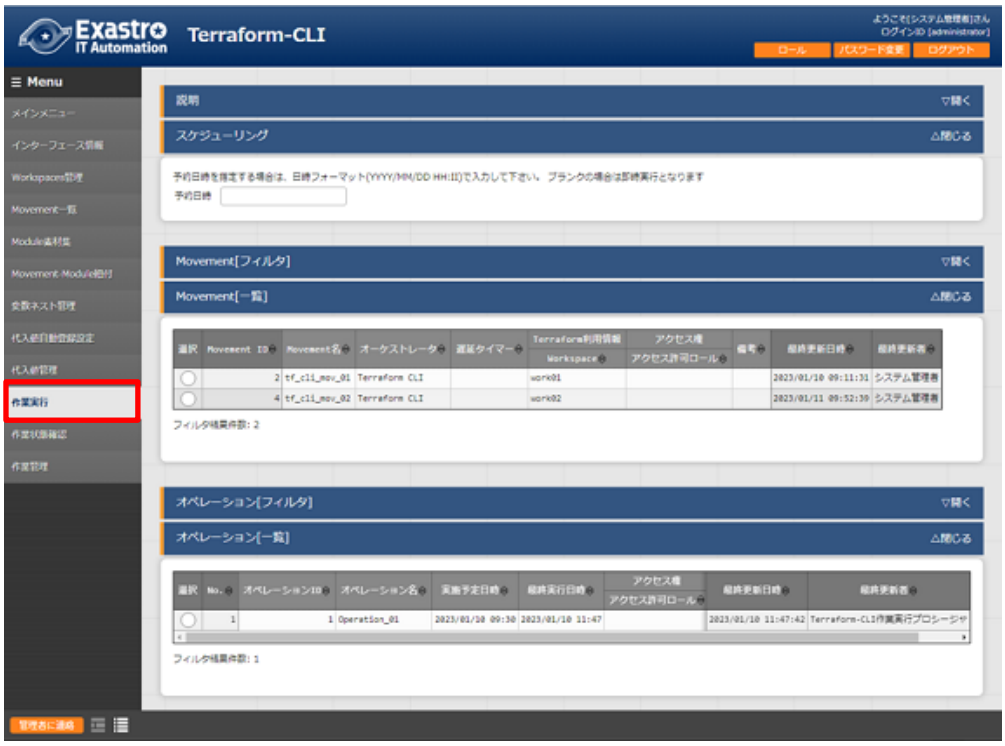


図 5.2.99-1 サブメニュー画面（作業実行）

① 予約日時の指定

「予約日時」を入力することで、実行および Plan 確認を予約することができます。
「予約日時」には、未来の日時のみ登録可能です。

② Movement の指定

「5.2.3 Movement 一覧」で登録した Movement を選択します。

③ オペレーションの指定

「5.1.1 オペレーション一覧」で登録したオペレーションを選択します。

④ 実行

「実行」ボタンをクリックすると、「5.2.10 作業状態確認」に遷移し、作業が実行されます。
Plan 完了後に Apply が自動で実行されます。

⑤ Plan 確認

「Plan 確認」ボタンをクリックすると、「実行」ボタンをクリックした場合同様に作業実行が開始されますが、Plan のみを実行し、Apply は実行されません。

(2) Outputブロックを含むModuleがConductorから実行された場合、Outputブロックに書かれた内容がデータリレイストレージパス（Conductor 実行時、各 Movement で共有するディレクトリを、ITA サーバから見たディレクトリパス）に json 形式ファイルで保存されます。
このファイルを使用することにより、同一 Conductor の別の Movement で Terraform が出力した値を使用することができます。

ファイルパス
[データリレイストレージパス]/[Conductor インスタンス ID]/terraform_output_[作業 No].json

例：/exastro/data_relay_storage/conductor/0000000001/terraform_output_0000000001.json

データリレイストレージパス・・・[Conductor]-[Conductor インターフェース情報]メニューの[データリレイストレージパス]

Conductor インスタンス ID・・・[Conductor]-[Conductor 作業一覧]メニューの[Conductor インスタンス ID]（左 0 埋め 10 桁）

作業 No.・・・[Terraform]-[作業管理]メニューの[作業 No.]（左 0 埋め 10 桁）

記述例

```
variable "VAR_sample" {
   type = string
   default = "sample_string"
}
output "output_sample" {
   value = "${var.VAR_sample}"
}
```

出力例

```
{
   "output_sample": "sample_string"
}
```

5.2.10 作業状態確認

(1) 作業の実行状態を監視します。

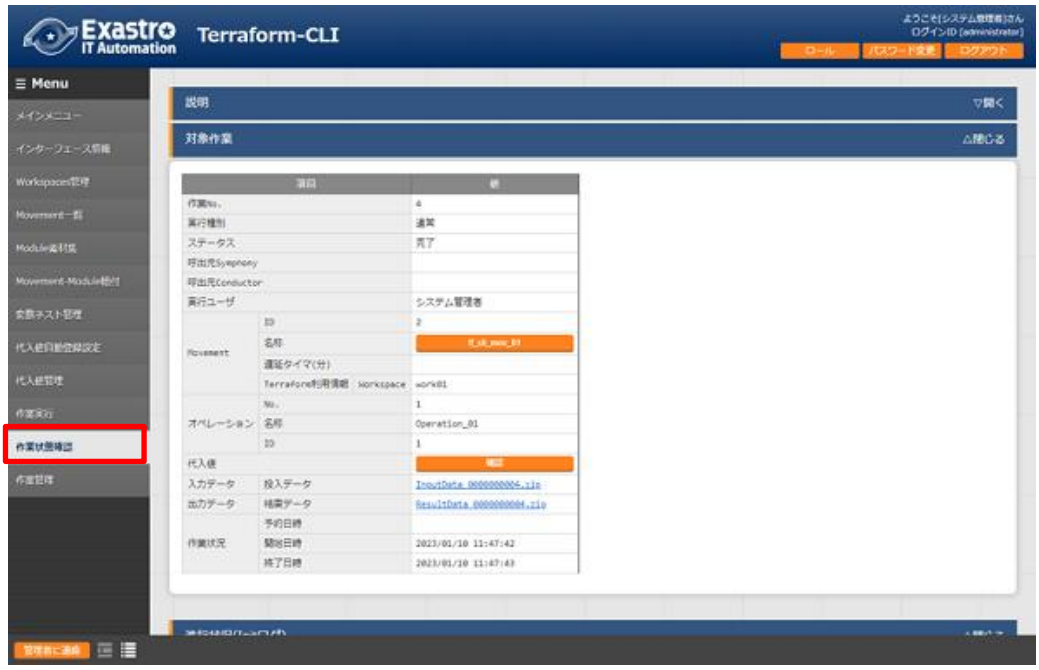


図 5.2.100-1 サブメニュー画面（作業状態確認）

① 実行状態表示

実行状況に即し、ステータスが表示されます。
「実行種別」には Plan 確認の場合には「Plan 確認」、Workspace ごとに構成・管理されたリソースの削除（「5.2.2 Workspaces 管理」から実行されます。）の場合は「リソース削除」、それ以外の場合には「通常」が入ります。

「進行状況(Init ログ)」「進行状況(Plan ログ)」「進行状況(Apply ログ)」には Terraform で実行するコマンド terraform init / terraform plan / terraform apply の各実行状況のログが表示されます。

ステータスが想定外エラーで終了した場合、「5.2.1 インターフェース情報」の登録不備や、その他の Web コンテンツの登録不備が原因であれば、「進行状況(エラーログ)」にメッセージが表示されます。
それ以外のエラーの場合は「進行状況(エラーログ)」にメッセージが表示されません。この場合は、プロセスのログにエラー情報が記録されます。必要に応じてプロセスのログを確認ください。
※プロセスのログは「 7.2 メンテナンス方法について」の「④ログファイル名」に記載していますので、そちらをご参照ください。

「呼出元 Symphony」には、どの Symphony から実行されたかを表示します。Terraform CLI ドライバから直接実行した場合や Conductor から実行した場合は空欄になります。
「呼出元 Conductor」には、どの Conductor から実行されたかを表示します。Terraform CLI ドライバから直接実行した場合や Symphony から実行した場合は空欄になります。

「実行ユーザ」には、作業実行メニューより「実行」ボタンを押下した際のログインユーザが表示されます。

※「実行種別」が「リソース削除」の場合は下記の項目が設定されません。
・呼出元 Symphony
・呼出元 Conductor
・Movement（ID、名称、遅延データ（分））
・オペレーション（No.、名称、ID）
・代入値
・入力データ（投入データ）

② 代入値確認

「確認」ボタンで「5.2.8 代入値管理」が表示され、作業対象のオペレーションと Movement に絞り込んだ代入値が表示されます。

③ 緊急停止/予約取り消し

「緊急停止」ボタンで構築作業を停止させることができます。
また、実行前の「予約実行」の作業の場合は、「予約取消」ボタンが表示されます。「予約取消」ボタンで予約実行が取り消せます。

④ ログ検索

実行ログ、エラーログは、フィルタリングができます。各ログのフィルタのテキストボックスに検索したい文字列を入力し、「該当行のみ表示」のチェックボックスをチェックすることで該当する行だけが表示されます。
実行ログ、エラーログのリフレッシュ表示間隔と最大表示行数を、「5.2.1 インターフェース情報」の「状態監視周期（単位ミリ秒）」と「進行状態表示行数」で設定できます。

⑤ 投入データ

実行した Module 素材および設定した代入値の一覧を json 形式で取得したファイルを格納した zip 形式ファイルをダウンロードすることができます。
格納されているファイルは以下の通りです。

表 5.2.100-1 投入データ格納ファイル

ファイル名	説明
(投入した Module 素材ファイル名)	投入した Module 素材ファイルが zip ファイルの直下にすべて格納されます。
terraform.tfvars	設定した各代入値についての「変数名(key)」「具体値(value)」を記載したファイルです。 Secure 設定が ON の対象は記載されません。

⑥ 結果データ

実行ログ、エラーログおよび、terraform コマンドが生成したファイルを格納した zip 形式ファイルをダウンロードすることができます。
格納されているファイルは以下の通りです。

表 5.2.100-2 結果データ格納ファイル

ファイル名	説明
Init.log	進行状況(Init ログ)に出力された内容を記載した log ファイルです。
plan.log	進行状況(Plan ログ)に出力された内容を記載した log ファイルです。
apply.log	進行状況(Apply ログ)に出力された内容を記載した log ファイルです。

error.log	進行状況(エラーログ)に出力された内容を記載した log ファイルです。
result.txt	作業実行時にバックヤードが利用する進行状況を記録するファイルです。
.terraform.lock.hcl	Terraform が生成したファイルです。provider や module の情報が記載されます。
terraform.tfstate	Terraform が生成した state ファイルです。 暗号化された状態で保存されています。
terraform.tfstate.backup	Terraform が生成した state ファイルのバックアップです。 暗号化された状態で保存されています。

5.2.11 作業管理

(1) 作業の履歴を閲覧できます。
条件を指定し「フィルタ」ボタンをクリックすると、作業一覧テーブルを表示します。

「作業状態確認」ボタンで、「5.2.10 作業状態確認」に遷移し、実行状態の詳細を見ることができます。

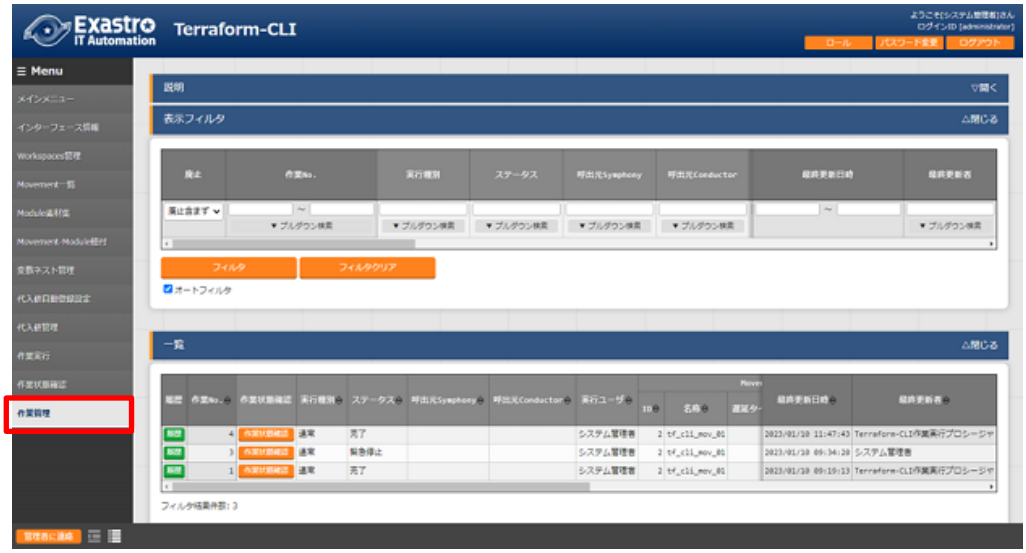


図 5.2.111-1 サブメニュー画面（作業管理）

6 構築コード記述方法

Terraform-CLI-driver で Module の記述について説明します。

6.1 Module の記述

Module ファイルは、HCL（HashiCorp Configuration Language）という HashiCorp 社独自の言語により記述します。

HCL の詳細については、Terraform の製品マニュアルを参照してください。

6.2 BackYard コンテンツ

(1) 変数自動登録

「5.2.4 Module 素材集」に登録した Module 素材ファイルから変数を抽出します。
変数の抽出ルールについては「 2.1 変数の種類」をご参照ください。
なお、取出すタイミングは「Terraform 変数自動登録」プロセスの起動周期に依存します。

(2) 代入値自動登録設定

連携対象としたパラメータシートのオペレーションの項目の設定値と紐付けた Movement と変数の情報を代入値管理に反映します。
なお、取出すタイミングは「Terraform 自動登録設定」プロセスの起動周期に依存します。
代入値管理は複数の操作者が更新を行います。最終更新者が他操作者の場合は反映処理をしません。
代入値自動登録設定のデータを反映したい場合は、代入値管理で該当レコードを廃止にする操作を行ってください。

代入値管理への反映ルールを以下に記載します。

① 代入値自動登録設定に登録されている情報を代入値管理へ反映時

代入値管理の状態	該当レコードなし	該当レコードあり			該当レコード廃止中
		=具体値	≠具体値		
			最終更新者		
			BackYard 処理	他操作者	
代入値管理への反映	新規レコード追加	－	該当レコードの具体値更新	－	廃止レコード復活

※ 該当レコード：オペレーション＋Movement+変数名+HCL 設定+アクセス権が同一のレコードの意

② 代入値自動登録設定に登録されていない情報（代入値管理のみに登録）を代入値管理へ反映

代入値管理の状態	該当レコードあり	
	最終更新者	
	BackYard 処理	他操作者
代入値管理への反映	該当レコード廃止	－

③ HCL 設定について

代入値自動登録に設定した「HCL 設定」の値は代入値管理に反映する際に同じ値が設定されます。

④ **Secure 設定について**

連携対象としたパラメータシートの項目が「パスワード」の場合は代入値管理に反映する際に「Secure 設定」が ON で設定されます。それ以外の場合は OFF で設定されます。

⑤ **アクセス許可ロールについて**

連連携対象としたパラメータシートのレコードに設定されたオペレーションのアクセス許可ロールと、代入値自動登録のレコードに設定された Movement のアクセス許可ロールを参照し、一致しているすべてのアクセス許可ロールが代入値管理に反映する際に設定されます。
どちらもアクセス許可ロールが設定されていない場合（空欄の場合）は、代入値管理に反映する際も空欄が設定されます。
また、一致するアクセス許可ロールが一つも無い場合は代入値管理にレコードが作成されません。

7　運用操作

本機能を活用する操作は、クライアント PC のブラウザ画面からのユーザー利用による入力だけでなく、システム運用・保守による操作もあります。用意している運用・保守の操作は次のとおりです。

7.1　メンテナンス

Terraform-CLI-driver のプロセスの開始/停止/再起動に必要なファイルは以下となります。

説明	対象ファイル名
Terraform 作業実行監視 未実行作業の実行を行う。	ky_terraform_cli_execute-workflow.service
Terraform 変数自動登録 アップロードした Module 素材から変数・メンバー変数・最大繰返数の取出しを行う。 最大繰返数をメンバー変数に反映する。	ky_terraform_cli_varsautolistup-workflow.service
Terraform 自動登録設定 代入値自動登録に設定された情報を代入値管理に反映を行う。	ky_terraform_cli_valautosetup-workflow.service

対象ファイルは「/usr/lib/systemd/system」に格納されています。
プロセス起動/停止/再起動の方法は次の通りです。
root 権限でコマンドを実行してください。

① プロセス起動

```
# systemctl start ky_terraform_cli_execute-workflow.service
```

① プロセス停止

```
# systemctl stop ky_terraform_cli_execute-workflow.service
```

② プロセス再起動

```
# systemctl restart ky_terraform_cli_execute-workflow.service
```

各対象ファイル名に置き換えて起動/停止/再起動を行ってください。

7.2 メンテナンス方法について

① NORMAL レベルへの変更
以下のファイルの 8 行目「DEBUG」を「NORMAL」に書き換えます。
ログレベル設定ファイル：<インストールディレクトリ>/ita-root/confs/backyardconfs/ita_env

② DEBUG レベルへの変更
以下のファイルの 8 行目「NORMAL」を「DEBUG」に書き換えます。
ログレベル設定ファイル：<インストールディレクトリ>/ita-root/confs/backyardconfs/ita_env

③ 起動周期の変更
各対象ファイルの ExecStart の 5 番目のパラメータを変更します。（単位:秒）
例外を除き起動周期はデフォルト値の使用をしてください。

```
ExecStart=/bin/sh ${ITA_ROOT_DIR}/backyards/common/ky_loopcall-php-procedure.sh
/bin/php /bin/php ${ITA_ROOT_DIR}/backyards/terraform_driver/ky_terraform_execute-
workflow.php ${ITA_ROOT_DIR}/logs/backyardlogs 5 ${ITA_LOG_LEVEL} > /dev/null 2>&1
```

書き換え後、**プロセス再起動(restart)後に有効になります。**

④ ログファイル名

プロセス名	ログファイル名
ky_terraform_cli_execute-workflow	ky_terraform_cli_execute-workflow_YYYYMMDD.log
ky_terraform_cli_execute-workflow	ky_terraform_cli_execute-child-workflow_YYYYMMDD.log
ky_terraform_cli_varsautolistup-workflow	ky_terraform_cli_varsautolistup-workflow_YYYYMMDD.log
ky_terraform_cli_valautosetup-workflow	ky_terraform_cli_valautosetup-workflow_YYYYMMDD.log

ログファイルの出力先：<インストールディレクトリ>/ita-root/logs/backyardlogs

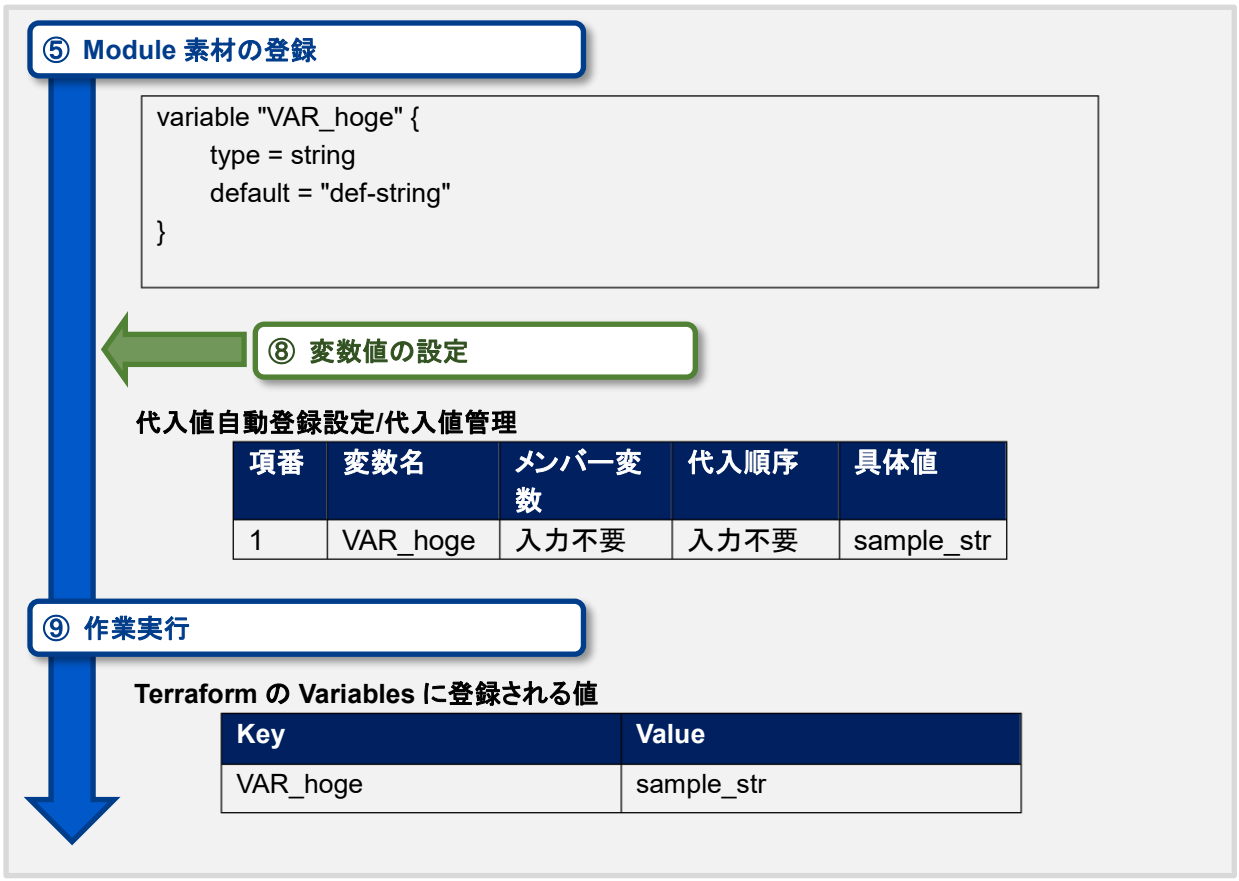
8 付録

8.1 Module 素材記入例・登録例

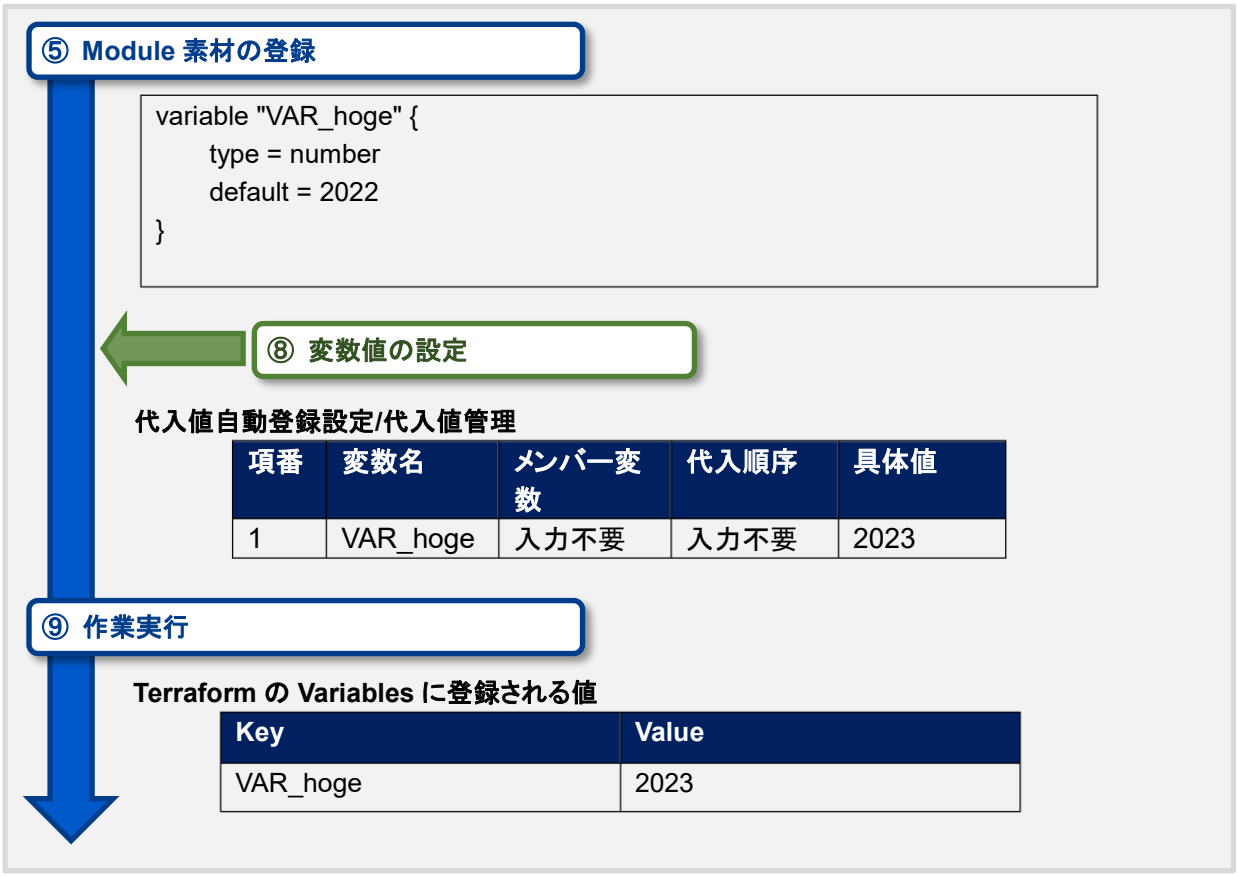
Module 素材の記入例と登録例を「 4.1 Terraform 作業フロー」のフロー番号と照らし合わせて記載します。

(1) シンプルなパターン

① string 型



② number 型


⑤ Module 素材の登録
variable "VAR_hoge" {
type = number
default = 2022
}
⑧ 変数値の設定
代入値自動登録設定/代入値管理
項番 変数名 メンバー変数 代入順序 具体値
1 VAR_hoge 入力不要 入力不要 2023
⑨ 作業実行
Terraform の Variables に登録される値
Key Value
VAR_hoge 2023

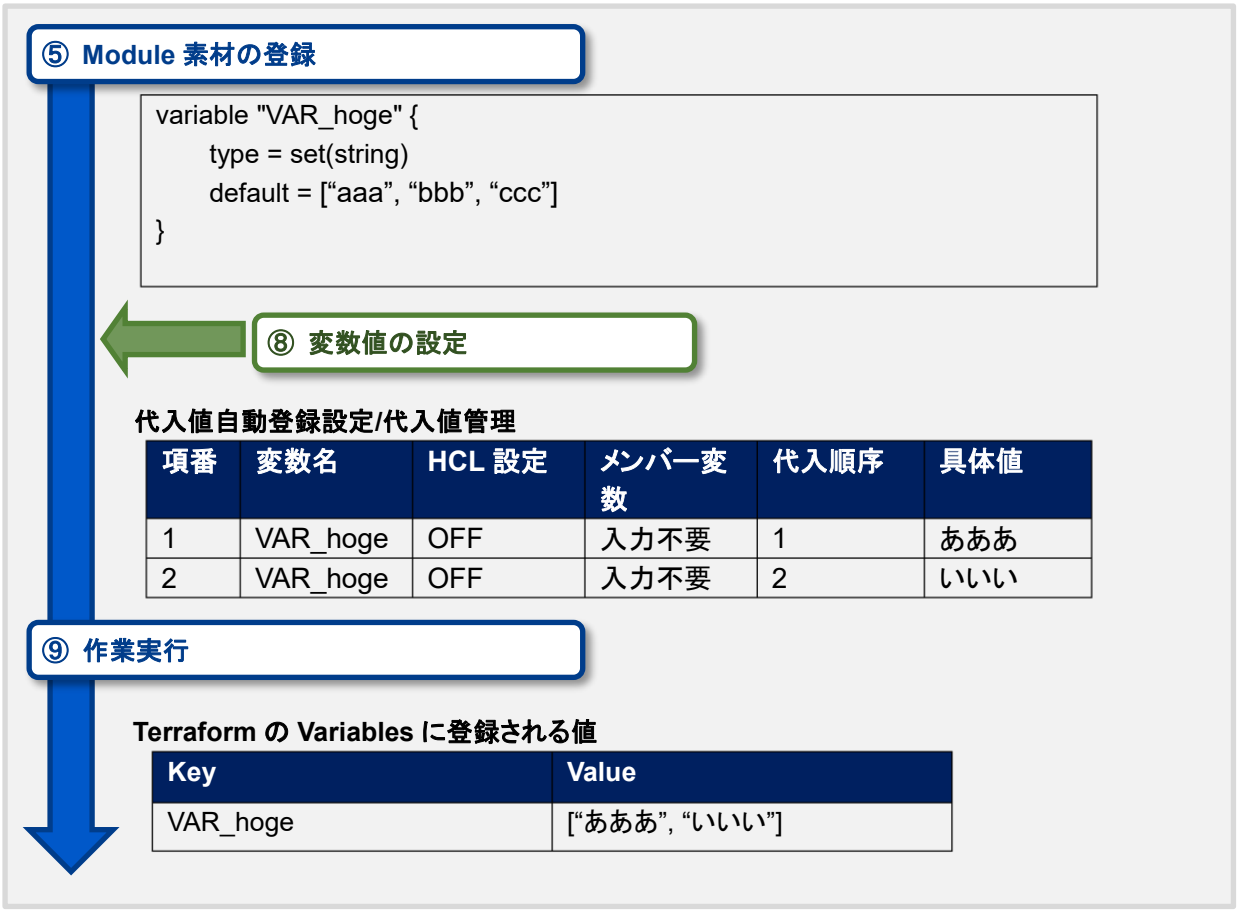
③ bool 型


⑤ Module 素材の登録
variable "VAR_hoge" {
type = bool
default = true
}
⑧ 変数値の設定
代入値自動登録設定/代入値管理
項番 変数名 メンバー変数 代入順序 具体値
1 VAR_hoge 入力不要 入力不要 false
⑨ 作業実行
Terraform の Variables に登録される値
Key Value
VAR_hoge false

④ list 型


⑤ Module 素材の登録
variable "VAR_hoge" {
type = list(string)
default = [“aaa”, “bbb”, “ccc”]
}
⑧ 変数値の設定
代入値自動登録設定/代入値管理
項番 変数名 HCL 設定 メンバー変数 代入順序 具体値
1 VAR_hoge OFF 入力不要 1 あああ
2 VAR_hoge OFF 入力不要 2 いいい
⑨ 作業実行
Terraform の Variables に登録される値
Key Value
VAR_hoge [“あああ”, “いいい”]

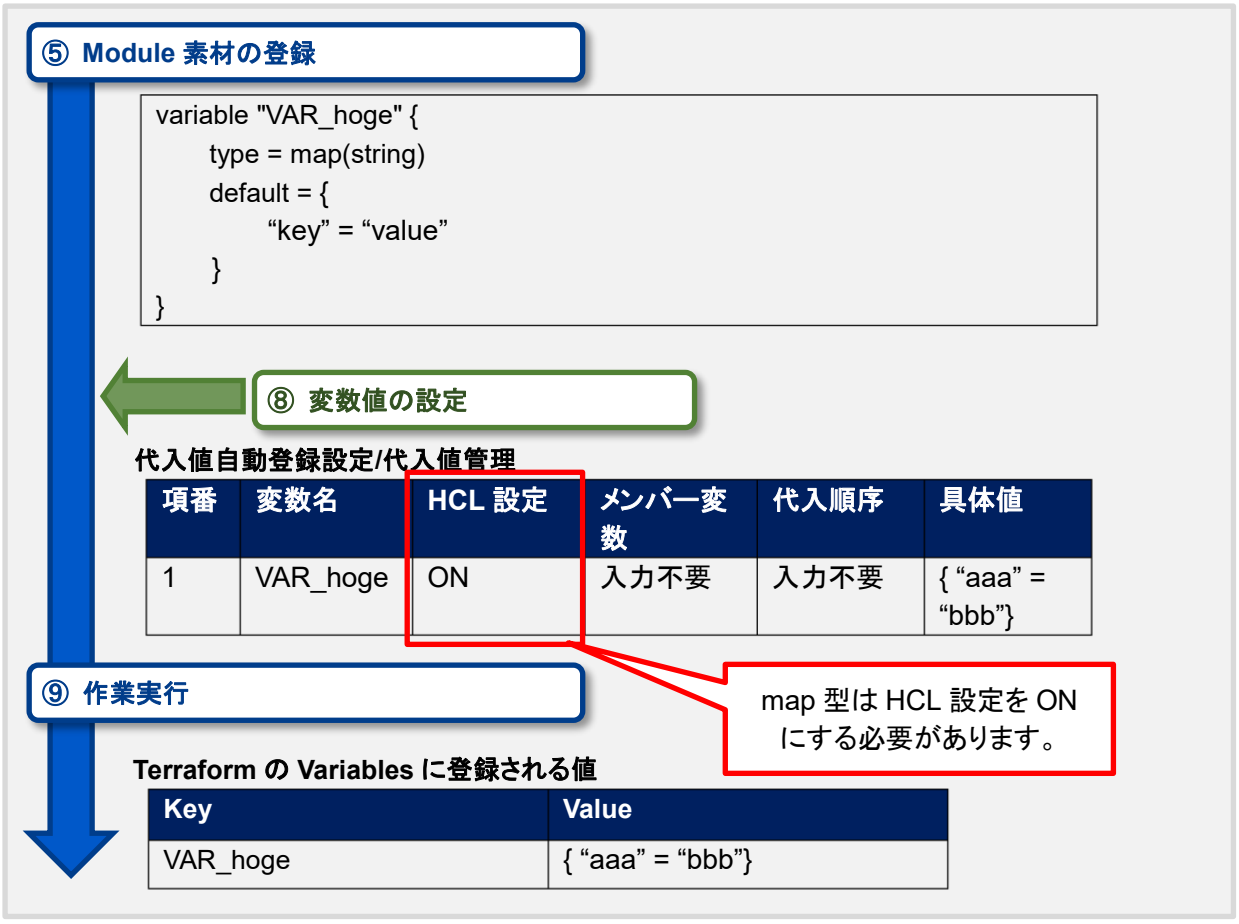
⑤ set 型


⑤ Module 素材の登録
variable "VAR_hoge" {
type = set(string)
default = [“aaa”, “bbb”, “ccc”]
}
⑧ 変数値の設定
代入値自動登録設定/代入値管理
項番 変数名 HCL 設定 メンバー変数 代入順序 具体値
1 VAR_hoge OFF 入力不要 1 あああ
2 VAR_hoge OFF 入力不要 2 いいい
⑨ 作業実行
Terraform の Variables に登録される値
Key Value
VAR_hoge [“あああ”, “いいい”]

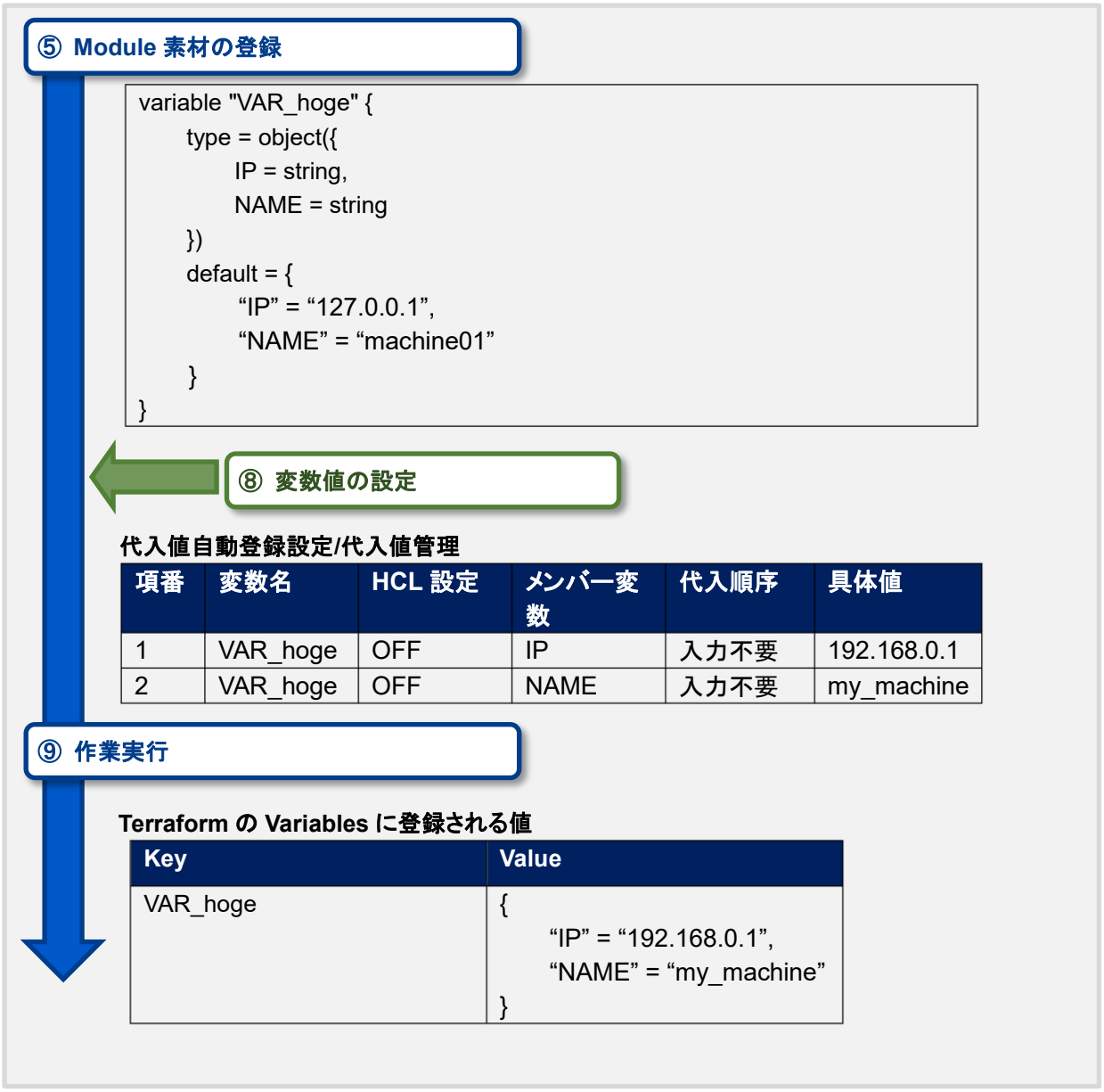
⑥ tuple 型


⑤ Module 素材の登録
variable "VAR_hoge" {
type = tuple([string, number])
default = [“aaa”, 2022]
}
⑧ 変数値の設定
代入値自動登録設定/代入値管理
項番 | 変数名 | HCL 設定 | メンバー変数 | 代入順序 | 具体値
1 | VAR_hoge | OFF | [0] | 入力不要 | あああ
2 | VAR_hoge | OFF | [1] | 入力不要 | 2023
⑨ 作業実行
Terraform の Variables に登録される値
Key | Value
VAR_hoge | [“あああ”, 2023]

⑦ map 型


⑤ Module 素材の登録
variable "VAR_hoge" {
type = map(string)
default = {
"key" = "value"
}
}
⑧ 変数値の設定
代入値自動登録設定/代入値管理
項番
変数名
HCL 設定
メンバー変数
代入順序
具体値
1
VAR_hoge
ON
入力不要
入力不要
{ "aaa" = "bbb"}
map 型は HCL 設定を ON にする必要があります。
⑨ 作業実行
Terraform の Variables に登録される値
Key
Value
VAR_hoge
{ "aaa" = "bbb"}

⑧ object 型


⑤ Module 素材の登録
variable "VAR_hoge" {
type = object({
IP = string,
NAME = string
})
default = {
“IP” = “127.0.0.1”,
“NAME” = “machine01”
}
}
⑧ 変数値の設定
代入値自動登録設定/代入値管理
項番 変数名 HCL 設定 メンバー変数 代入順序 具体値
1 VAR_hoge OFF IP 入力不要 192.168.0.1
2 VAR_hoge OFF NAME 入力不要 my_machine
⑨ 作業実行
Terraform の Variables に登録される値
Key Value
VAR_hoge
{
“IP” = “192.168.0.1”,
“NAME” = “my_machine”
}

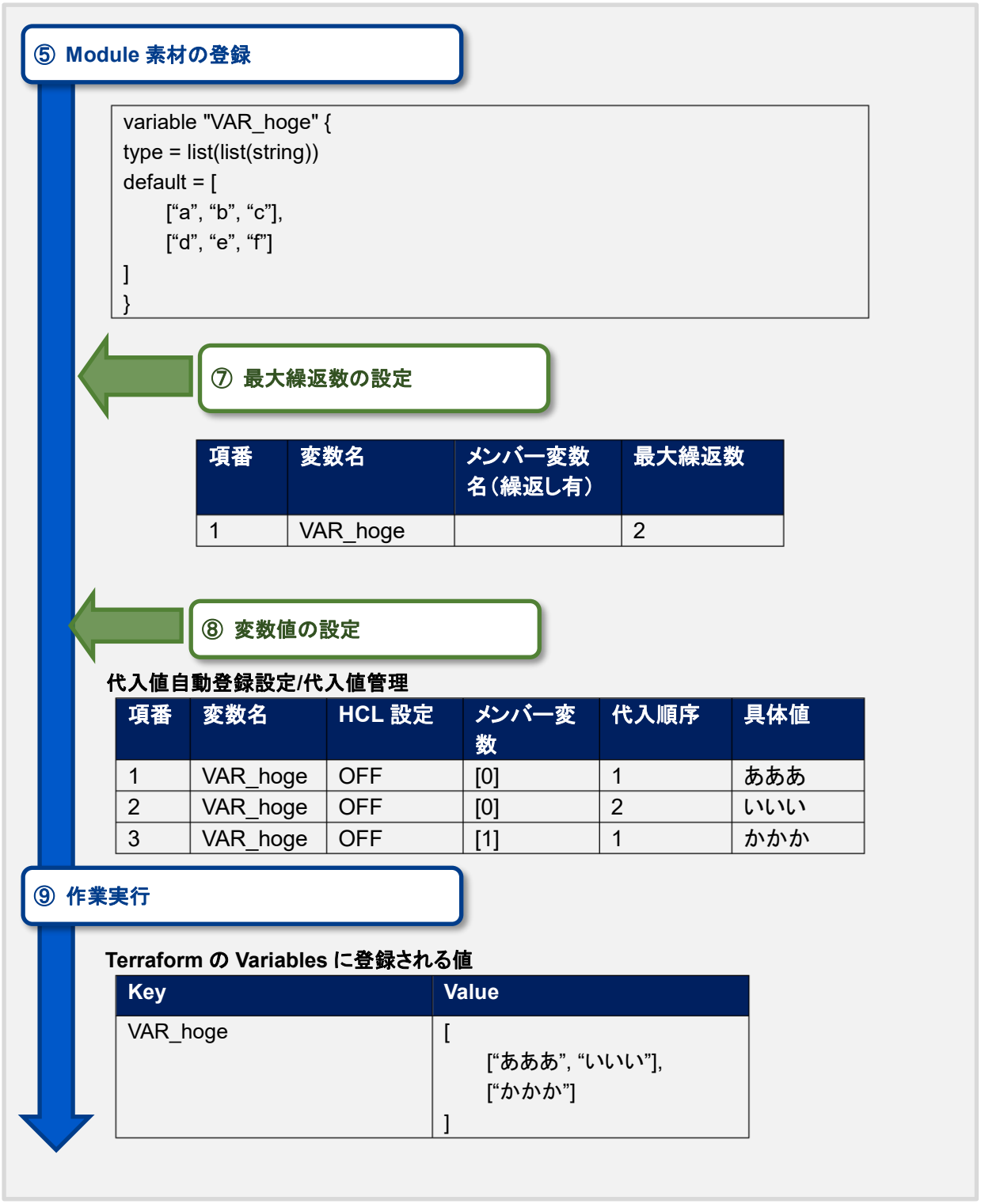
⑨ any 型


⑤ Module 素材の登録
variable "VAR_hoge" {
type = any
default = “def-any”
}
⑧ 変数値の設定
代入値自動登録設定/代入値管理
項番
変数名
HCL 設定
メンバー変数
代入順序
具体値
1
VAR_hoge
OFF
key
入力不要
aaa
⑨ 作業実行
Terraform の Variables に登録される値
Key
Value
VAR_hoge
aaa

⑩ type の記載がない

(2) 複雑なパターン

① list 型配下の list



項番	変数名	メンバー変数名（繰返し有）	最大繰返数
1	VAR_hoge		2

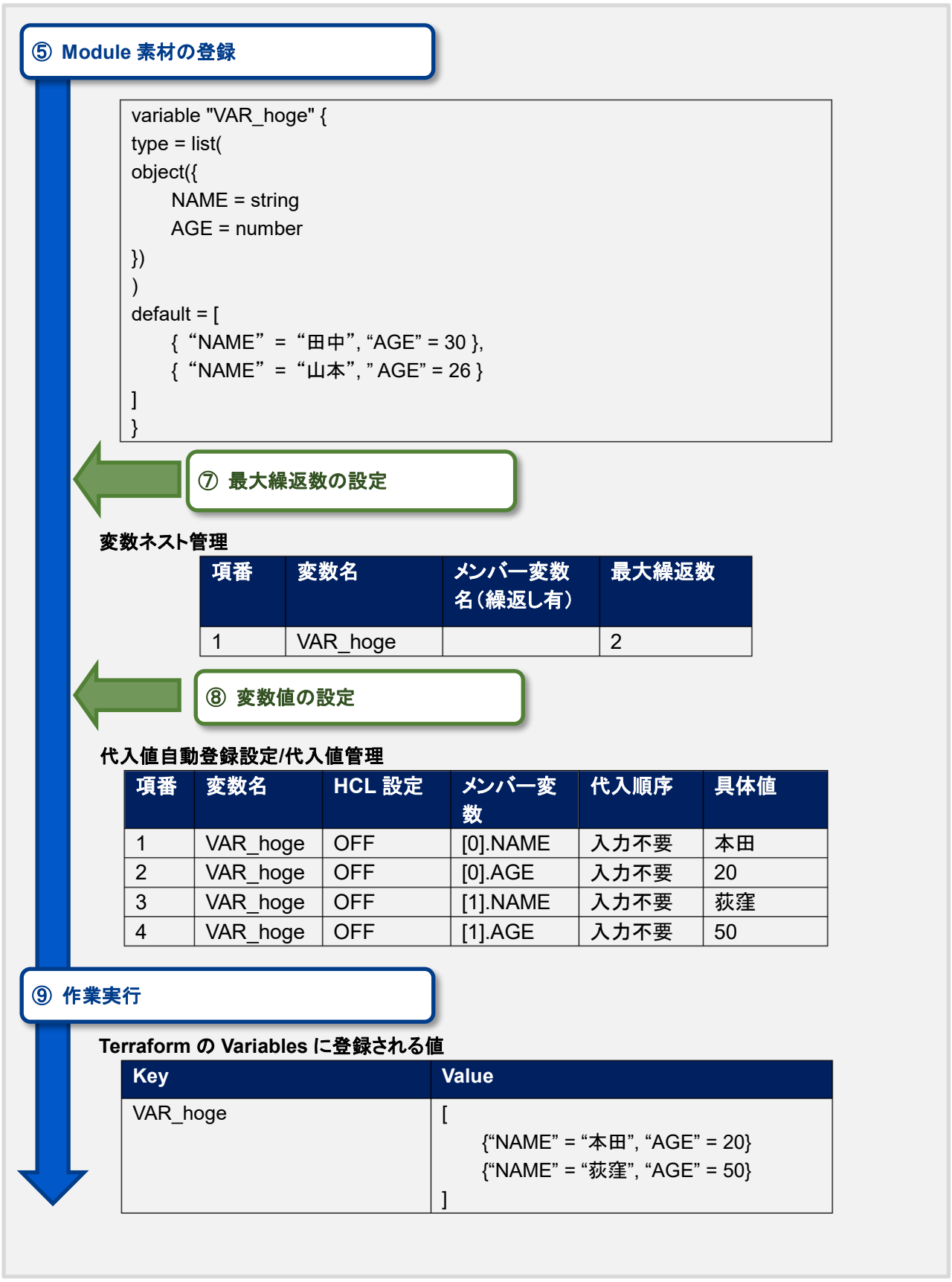
代入値自動登録設定/代入値管理

項番	変数名	HCL 設定	メンバー変数	代入順序	具体値
1	VAR_hoge	OFF	[0]	1	あああ
2	VAR_hoge	OFF	[0]	2	いいい
3	VAR_hoge	OFF	[1]	1	かかか

Terraform の Variables に登録される値

Key	Value
VAR_hoge	[["あああ", "いいい"], ["かかか"]]

② list 型配下の object



③ object 配下の list 配下の object

⑤ Module 素材の登録

```
variable "VAR_hoge" {
    type = object({
        FRUIT = list(object{
            NAME = string, PRICE = number
        }),
        VEGETABLE = l ist(object{
            NAME = string, PRICE = number
        })
    })
    default = {
        FRUIT = [
            { NAME = “りんご”, PRICE = 120 },
            { NAME = “みかん”, PRICE = 80 }
        ],
        VEGETABLE = [
            { NAME = “なす”, PRICE = 100 },
            { NAME = “トマト”, PRICE = 200 }
        ]
    }
}
```

⑦ 最大繰返数の設定

変数ネスト管理

項番	変数名	メンバー変数名（繰返し有）	最大繰返数
1	VAR_hoge	FRUIT	2
2	VAR_hoge	VEGETABLE	2

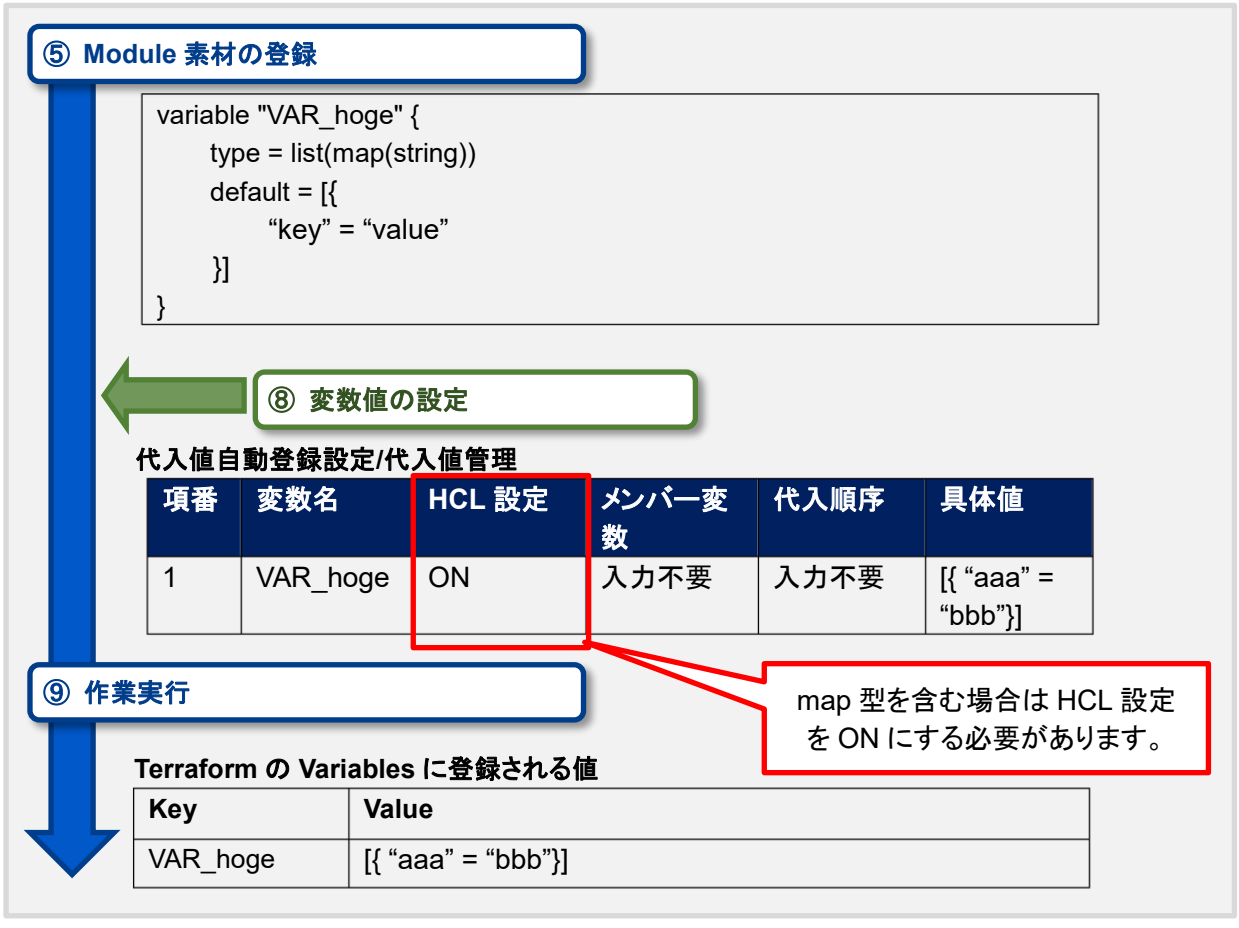
⑧ 変数値の設定

代入値自動登録設定/代入値管理

項番	変数名	HCL 設定	メンバー変数	代入順序	具体値
1	VAR_hoge	OFF	FRUIT.[0].NAME	入力不要	キウイ
2	VAR_hoge	OFF	FRUIT.[0].PRICE	入力不要	200
3	VAR_hoge	OFF	FRUIT.[1].NAME	入力不要	ぶどう
4	VAR_hoge	OFF	FRUIT.[1].PRICE	入力不要	1000
5	VAR_hoge	OFF	VEGETABLE.[0].NAME	入力不要	白菜
6	VAR_hoge	OFF	VEGETABLE.[0].PRICE	入力不要	100
7	VAR_hoge	OFF	VEGETABLE.[1].NAME	入力不要	キャベツ
8	VAR_hoge	OFF	VEGETABLE.[1].PRICE	入力不要	110

(3) 特殊な型

① list 型配下の map 型



8.2 変数ネスト管理フロー例

変数ネスト管理の操作例を「 4.1 Terraform 作業フロー」のフロー番号と照らし合わせて記載します。

(1)　最大繰返数を増加させる

⑤ Module 素材の登録

```
variable "VAR_hoge" {
      type = list(object({ IP = string, NAME = string }))
default = [
            { “IP” = “127.0.0.1”, NAME = “machine01”},
            { “IP” = “127.0.0.2”, NAME = “machine02”}
      ],
}
```

⑦ 最大繰返数の設定

変数ネスト管理（登録時の値）

項番	変数名	メンバー変数名（繰返し有）	最大繰返数
1	VAR_hoge		2

変数ネスト管理（更新）

項番	変数名	メンバー変数名（繰返し有）	最大繰返数
1	VAR_hoge		3

⑧ 変数値の設定

代入値自動登録設定/代入値管理

項番	変数名	HCL 設定	メンバー変数	代入順序	具体値
1	VAR_hoge	OFF	[0].IP	入力不要	192.168.1.1
2	VAR_hoge	OFF	[0].NAME	入力不要	yamamoto
3	VAR_hoge	OFF	[1].IP	入力不要	192.168.1.2
4	VAR_hoge	OFF	[1].NAME	入力不要	suzuki
5	VAR_hoge	OFF	[2].IP	入力不要	192.168.1.3
6	VAR_hoge	OFF	[2].NAME	入力不要	tanaka

変数ネスト管理更新により
追加されたメンバー変数

⑨ 作業実行

Terraform の Variables に登録される値

Key	Value
VAR_hoge	[{ “IP” = “192.168.1.1”, “NAME” = “yamamoto” }, { “IP” = “192.168.1.2”, “NAME” = “suzuki” }, { “IP” = “192.168.1.3”, “NAME” = “tanaka” }]

(2)　最大繰返数を減少させる

⑤ Module 素材の登録

```
variable "VAR_hoge" {
      type = list(object({ IP = string, NAME = string }))
default = [
            { “IP” = “127.0.0.1”, NAME = “machine01”},
            { “IP” = “127.0.0.2”, NAME = “machine02”},
            { “IP” = “127.0.0.3”, NAME = “machine03”}
      ],
}
```

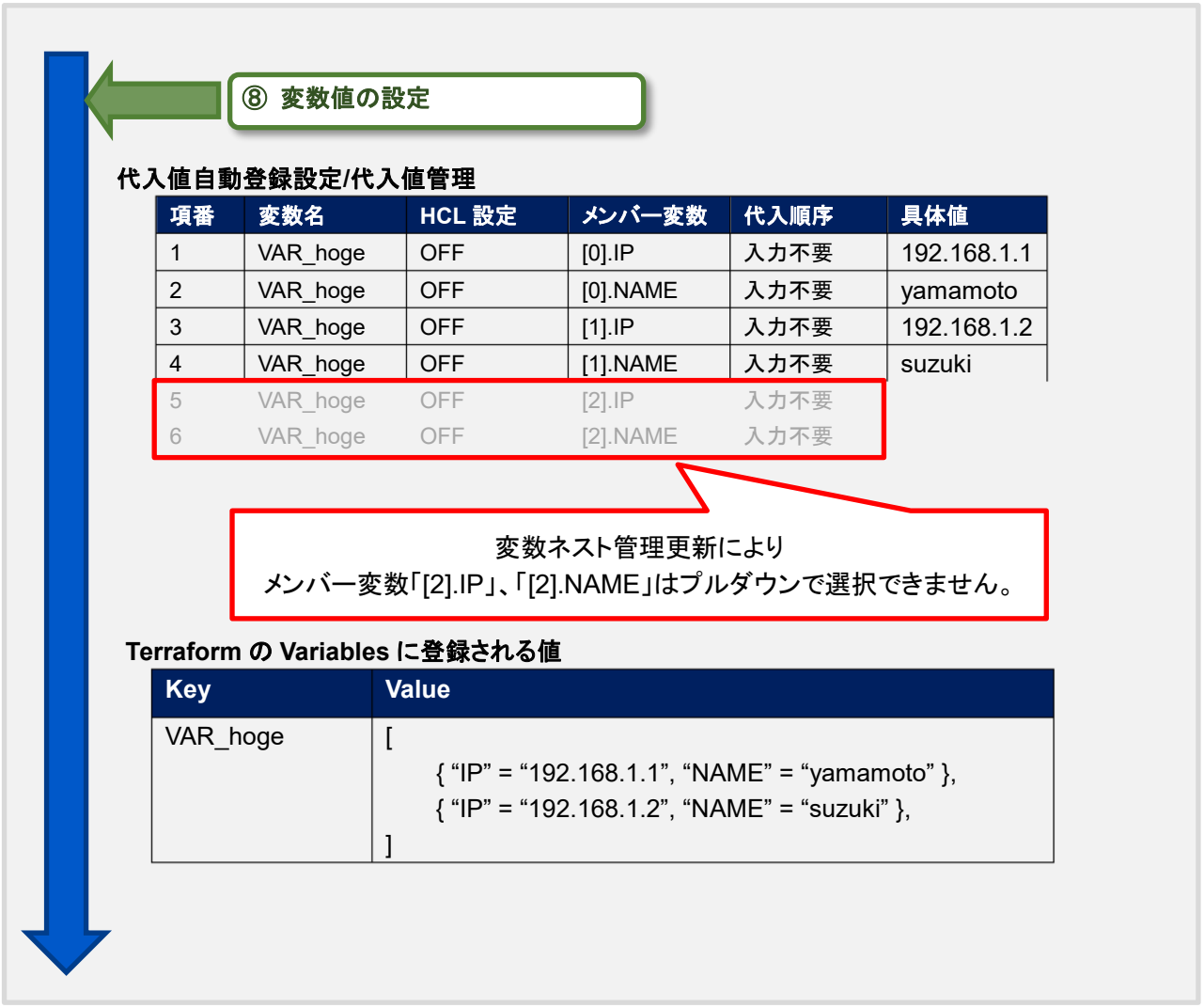
⑦ 最大繰返数の設定

変数ネスト管理（登録時の値）

項番	変数名	メンバー変数名（繰返し有）	最大繰返数
1	VAR_hoge		3

変数ネスト管理（更新）

項番	変数名	メンバー変数名（繰返し有）	最大繰返数
1	VAR_hoge		2


⑧ 変数値の設定
代入値自動登録設定/代入値管理
項番 変数名 HCL 設定 メンバー変数 代入順序 具体値
1 VAR_hoge OFF [0].IP 入力不要 192.168.1.1
2 VAR_hoge OFF [0].NAME 入力不要 yamamoto
3 VAR_hoge OFF [1].IP 入力不要 192.168.1.2
4 VAR_hoge OFF [1].NAME 入力不要 suzuki
5 VAR_hoge OFF [2].IP 入力不要
6 VAR_hoge OFF [2].NAME 入力不要
変数ネスト管理更新により
メンバー変数「[2].IP」、「[2].NAME」はプルダウンで選択できません。
Terraform の Variables に登録される値
Key Value
VAR_hoge
[
{ “IP” = “192.168.1.1”, “NAME” = “yamamoto” },
{ “IP” = “192.168.1.2”, “NAME” = “suzuki” },
]