

Exastro × Terraform × Ansibleで実現する マルチクラウドシステムプロビジョニング



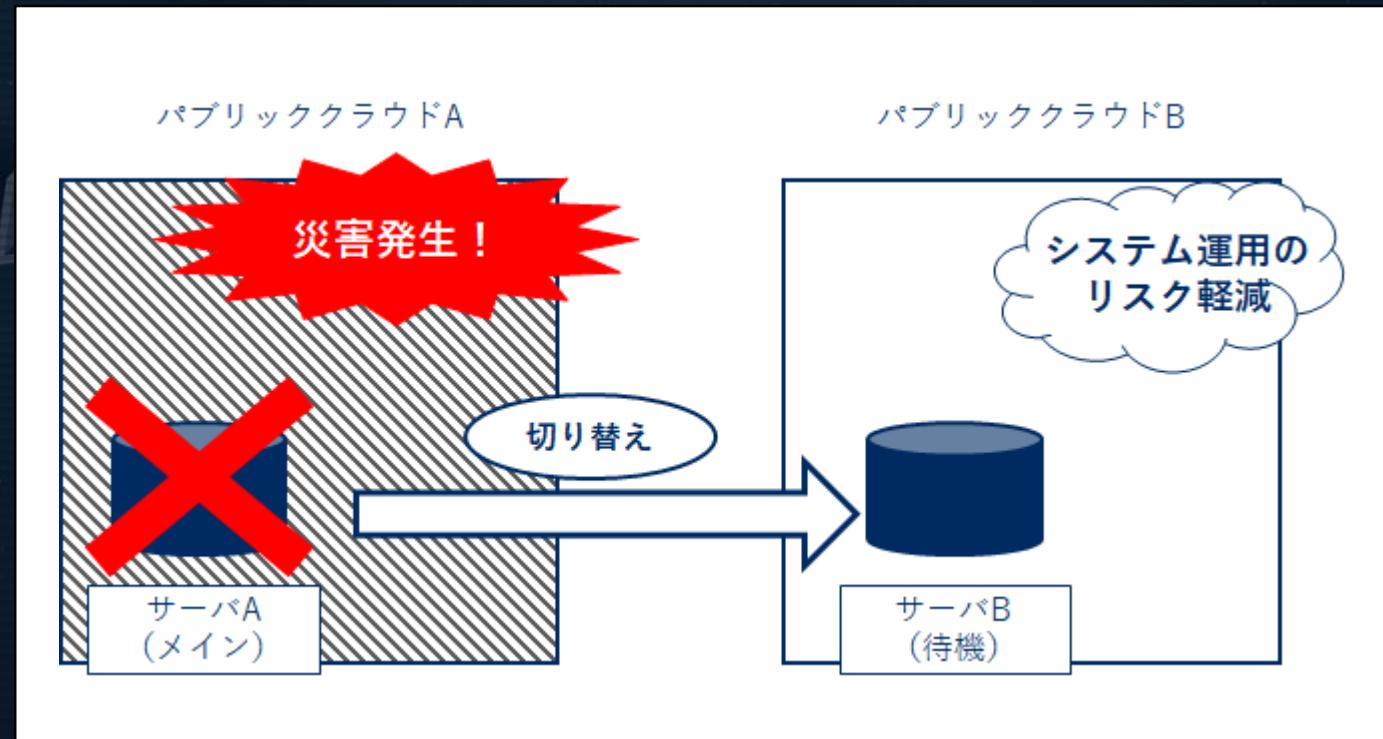
アジェンダ

- 1.マルチクラウドとは？
- 2.Exastro・Terraformについて
- 3.デモ

マルチクラウドへの期待って？

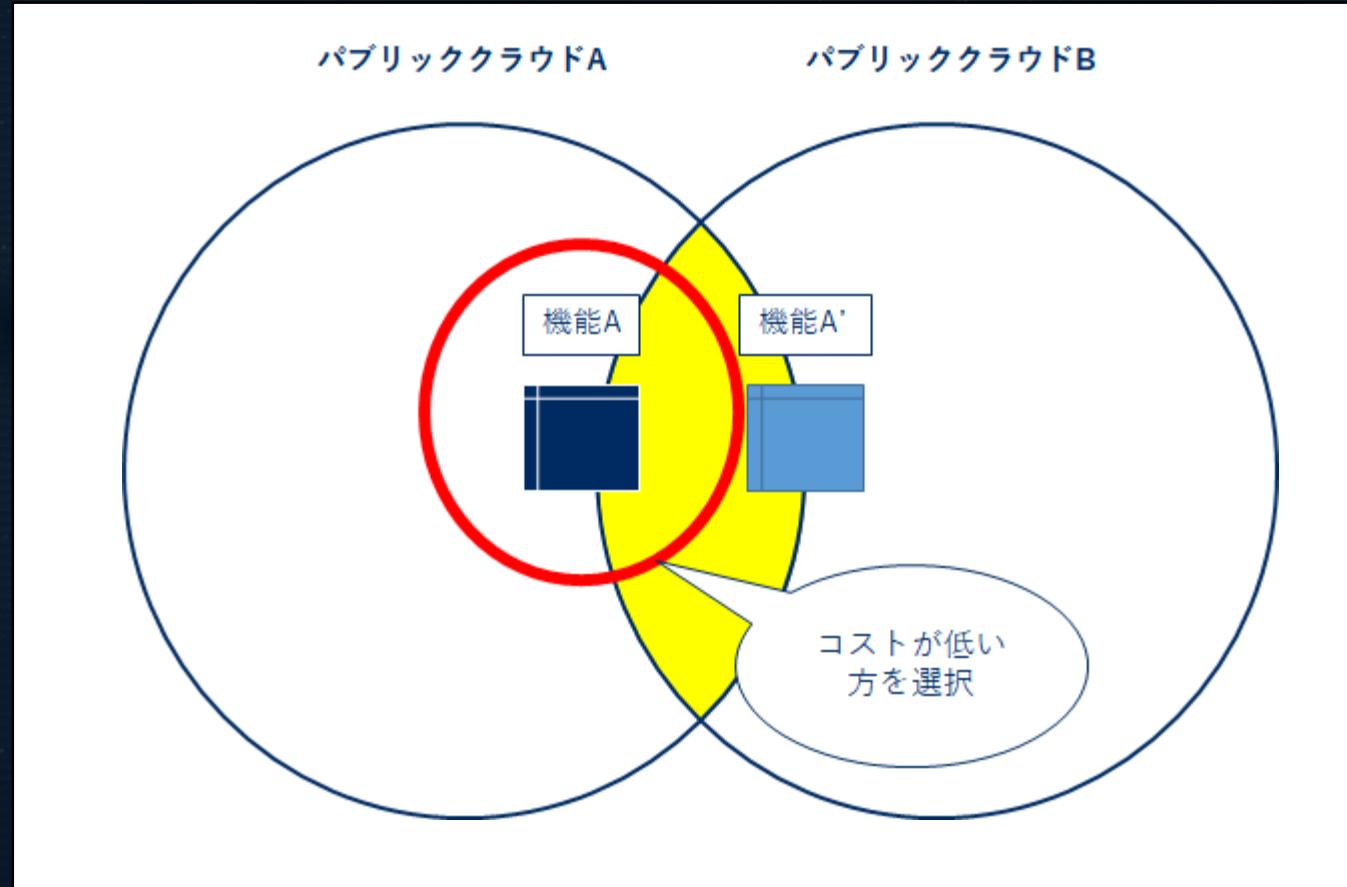
マルチクラウド

“複数のパブリッククラウドを使い分ける利用形態。データ通信量の分散、データ消失・システムダウンのリスク分散などの利点がある。” 引用：「デジタル大辞泉」



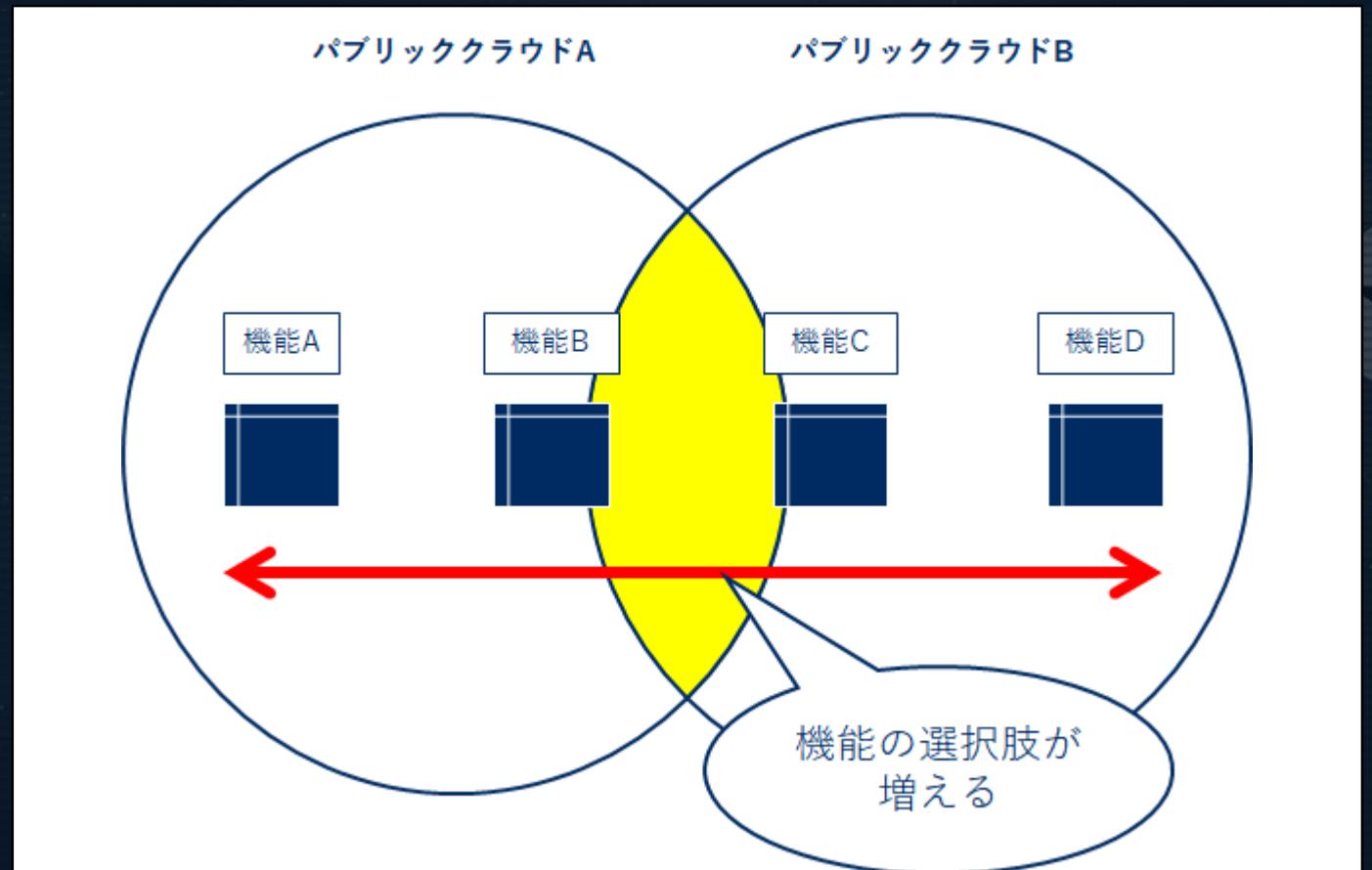
マルチクラウドへのさらなる期待 その1

パブリッククラウドの利用において、“コスト”は非常に重要な観点。
コストが安い機能の選択により、削減効果を見込める可能性あり。



マルチクラウドへのさらなる期待 その2

それぞれのパブリッククラウドごとに用意されている**固有の機能**を利用し、システムの構築を有利・スピーディに進めることができる。



マルチクラウドにおける課題とは？

マルチクラウドは以下のような問題を抱えている

- ✓ エンジニアのスキル問題(※1)
- ✓ 導入までの遅れ(※2)
- ✓ クラウドの管理や可視化(※3)

(※1) <https://www.denodo.com/ja/document/whitepaper/denodo-global-cloud-survey-2020>

(※2) <https://japan.zdnet.com/article/35163287/>

(※3) <https://techttarget.itmedia.co.jp/tt/news/1908/30/news05.html>



マルチクラウドの標準化・自動化・構成管理が必要とされている

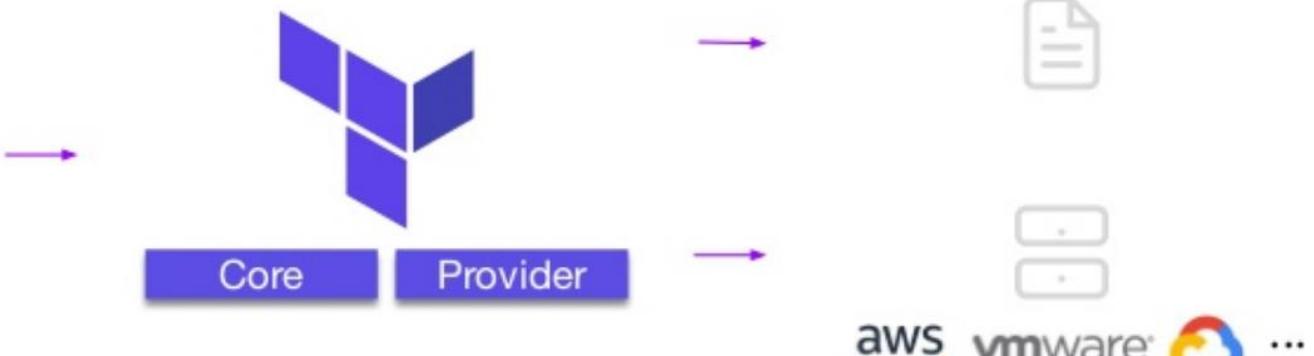
Terraform: Infrastructure as Code

tfconfig

```
resource "vsphere_virtual_machine" "vm" {
  name          = "terraform-test"
  resource_pool_id = "*****"
  datastore_id    = "*****"
  num_cpus       = 2
  memory         = 1024

  network_interface {
    network_id = "*****"
  }

  disk {
    label = "disk0"
    size  = 20
  }
}
```

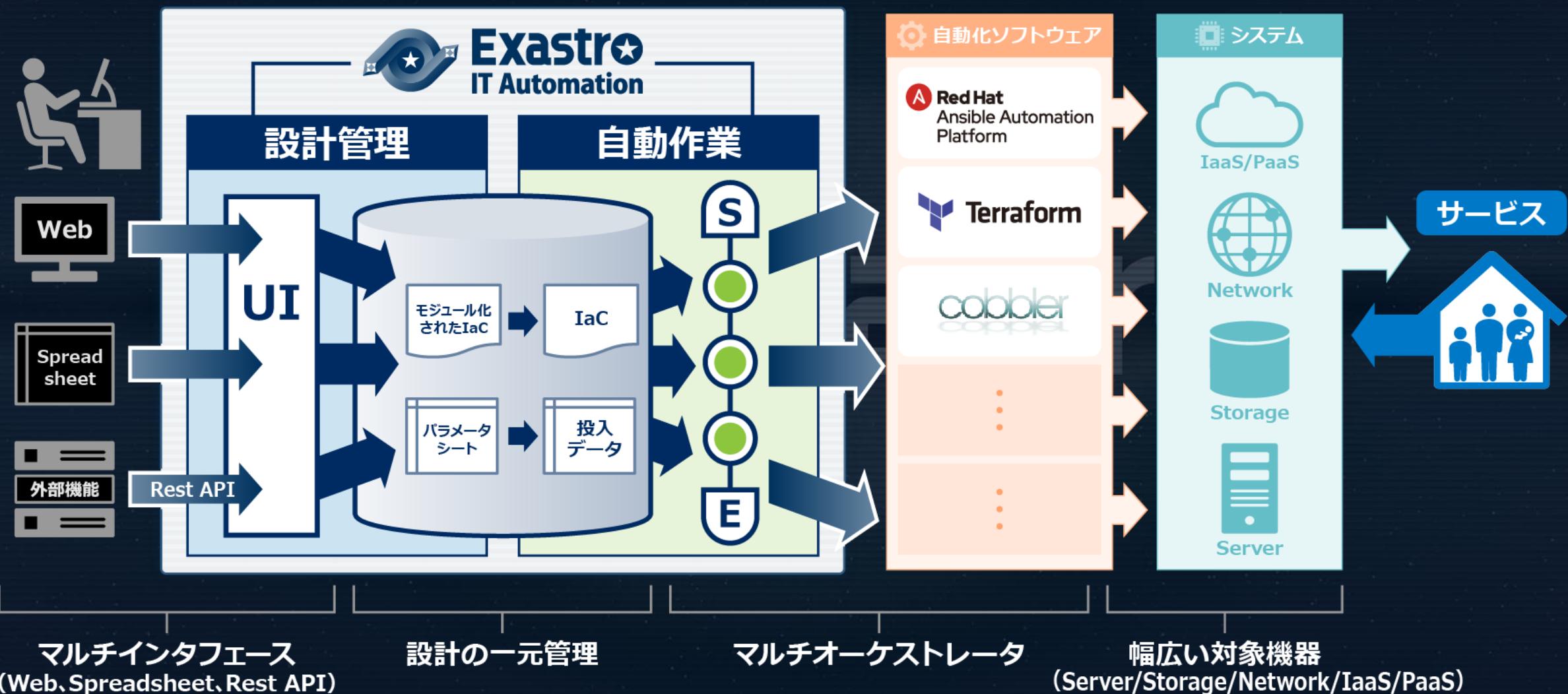


<https://www.slideshare.net/hashicorpjp/terraform-enterprise>
<https://www.terraform.io/>

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

設計フェーズ

作業フェーズ



設計フェーズ

作業フェーズ

Exastro IT Automationの提供価値はココ！

- ✓ IaC/PaCをライブラリ化して再利用可能なように管理
- ✓ 変数に適用するパラメータ値を一元管理

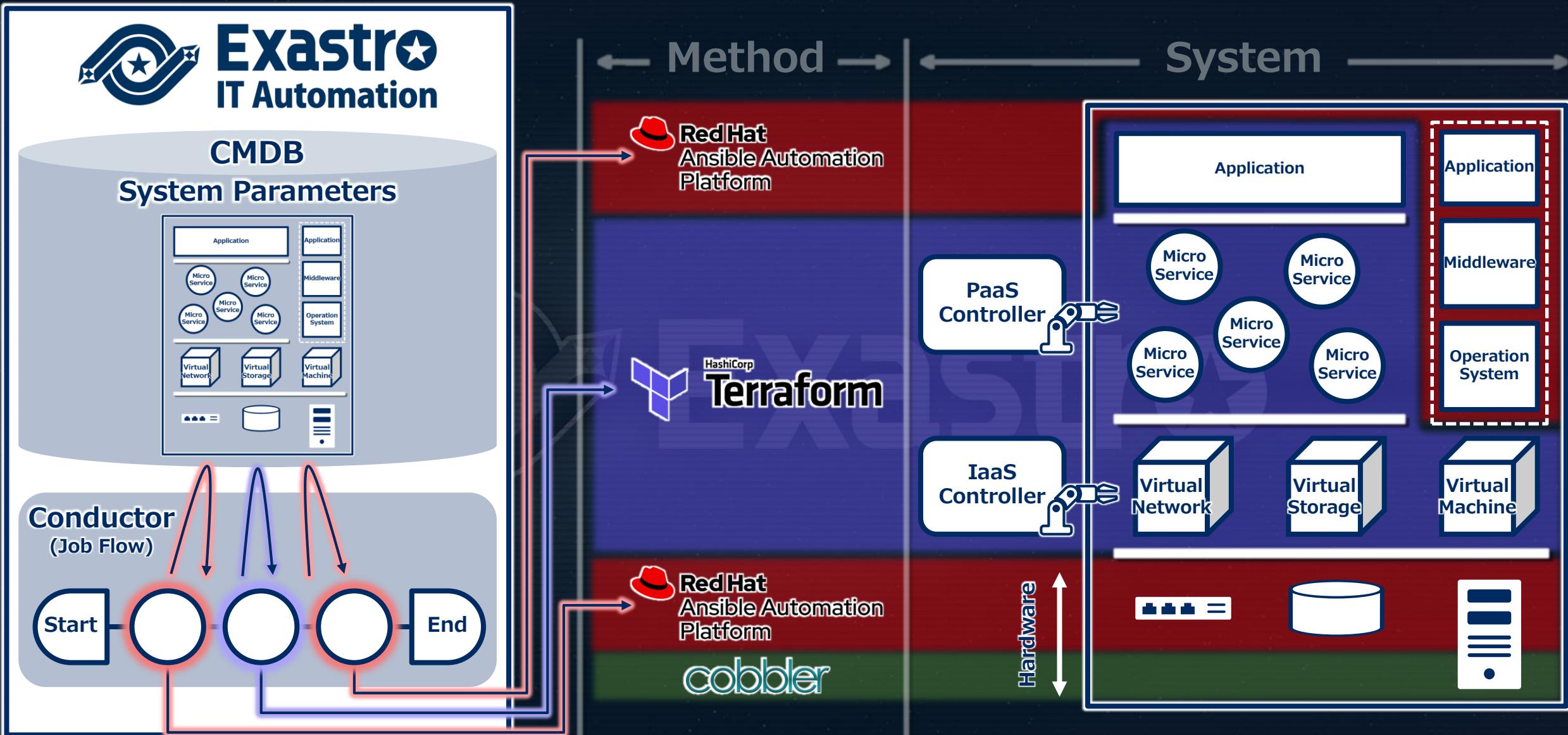


設計フェーズ

作業フェーズ



Exastro IT Automation : システムスタックと使用するメソッド



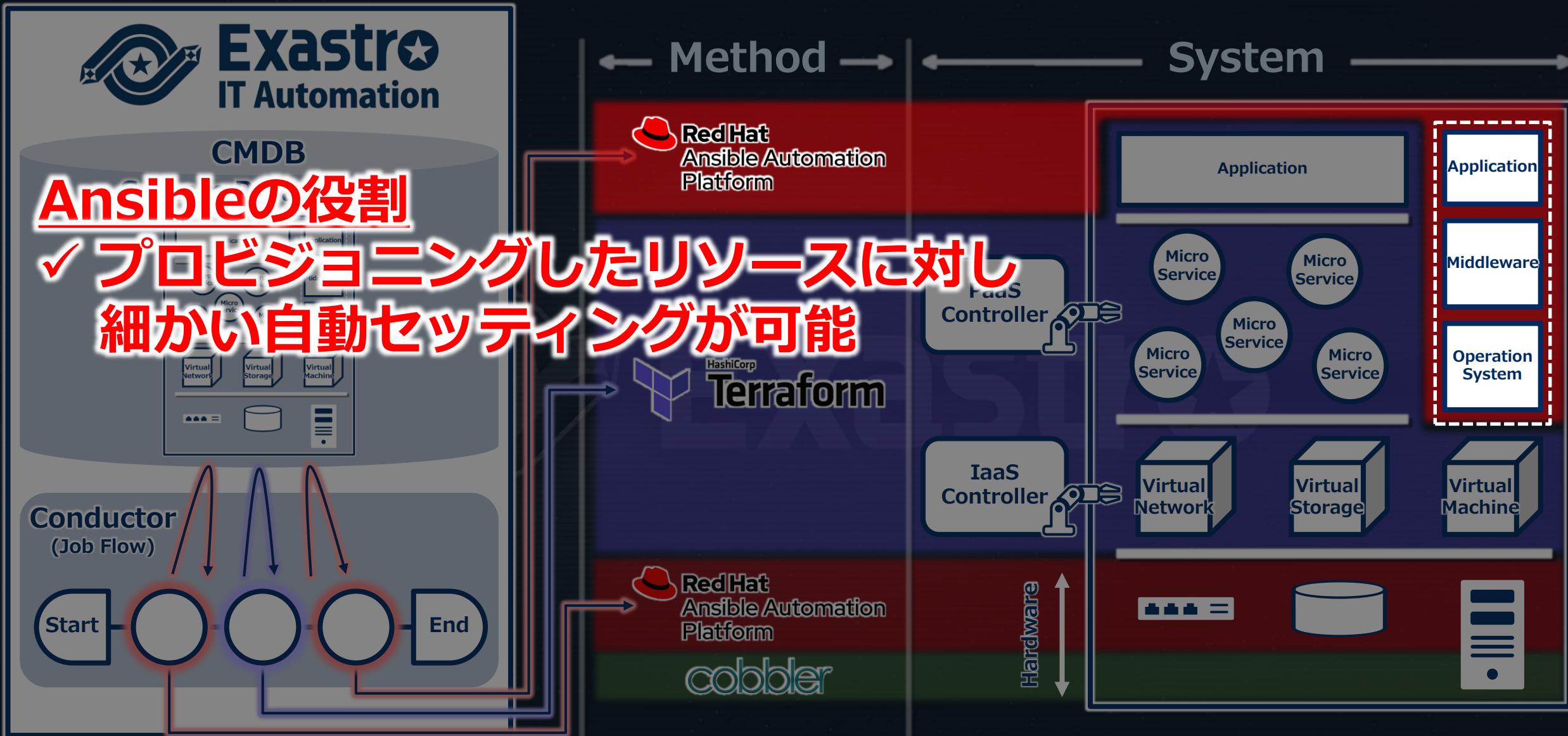
Exastro ITAからAnsibleとTerraformを相互補完して活用することは？

Terraformの役割

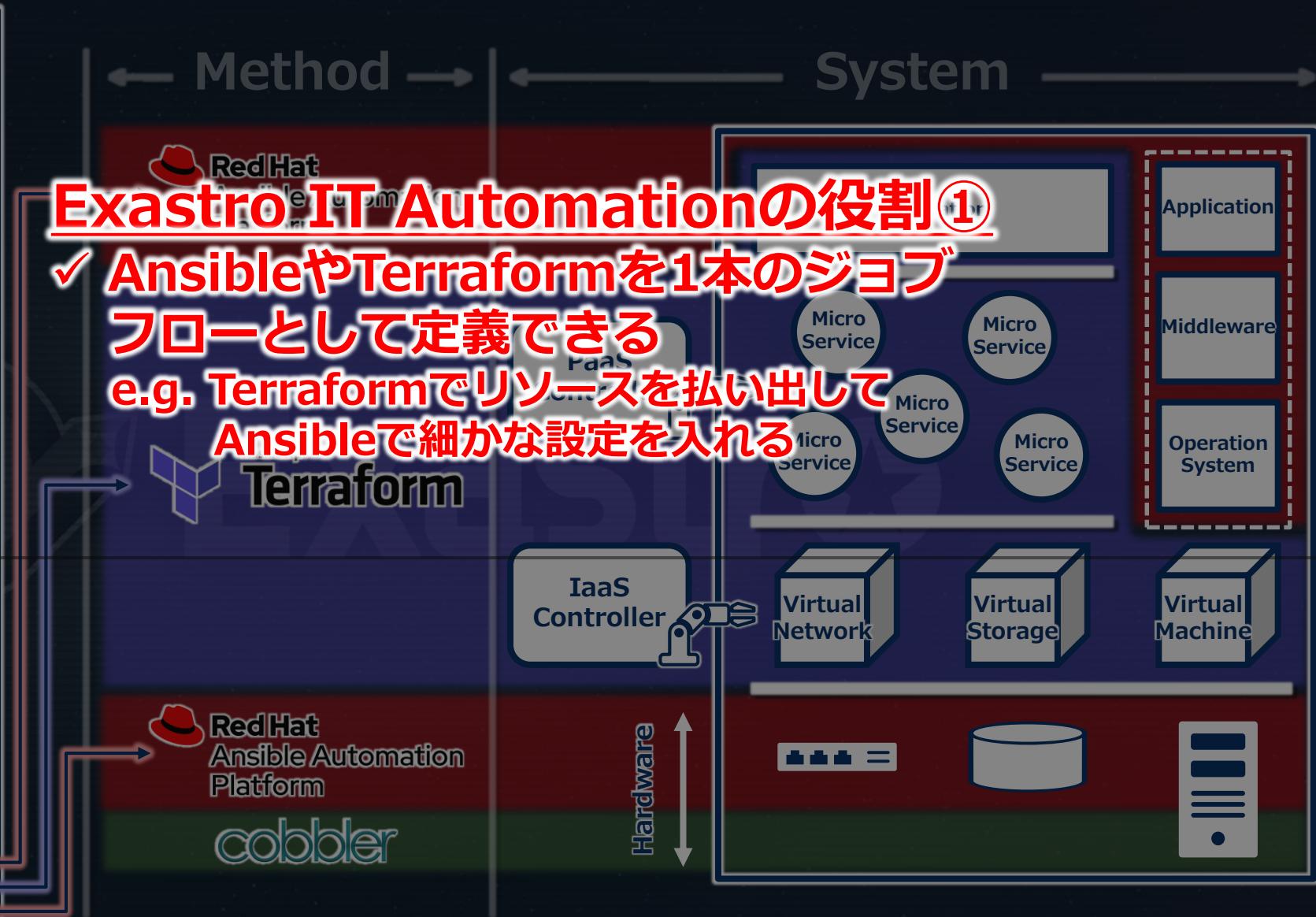
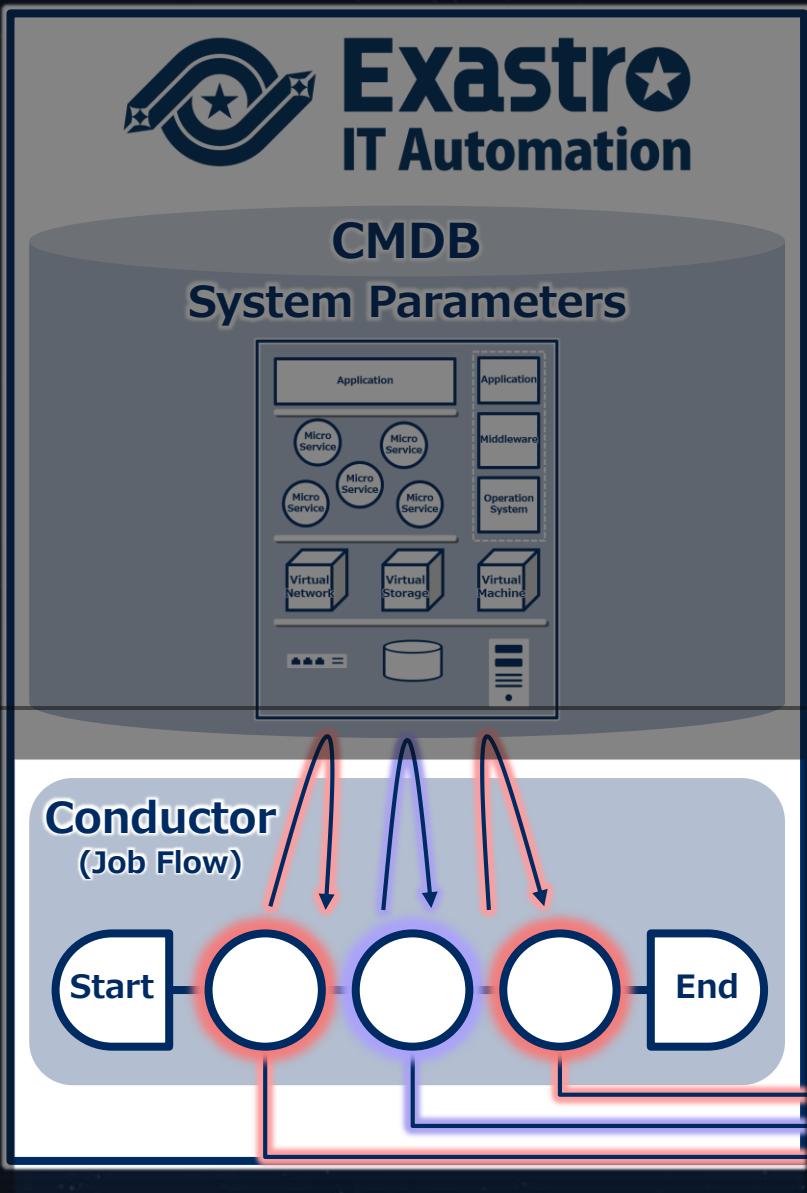
- ✓ パブリッククラウドをまたがるリソースのオーケストレイティングを、共通の言語(HCL)で記述できる



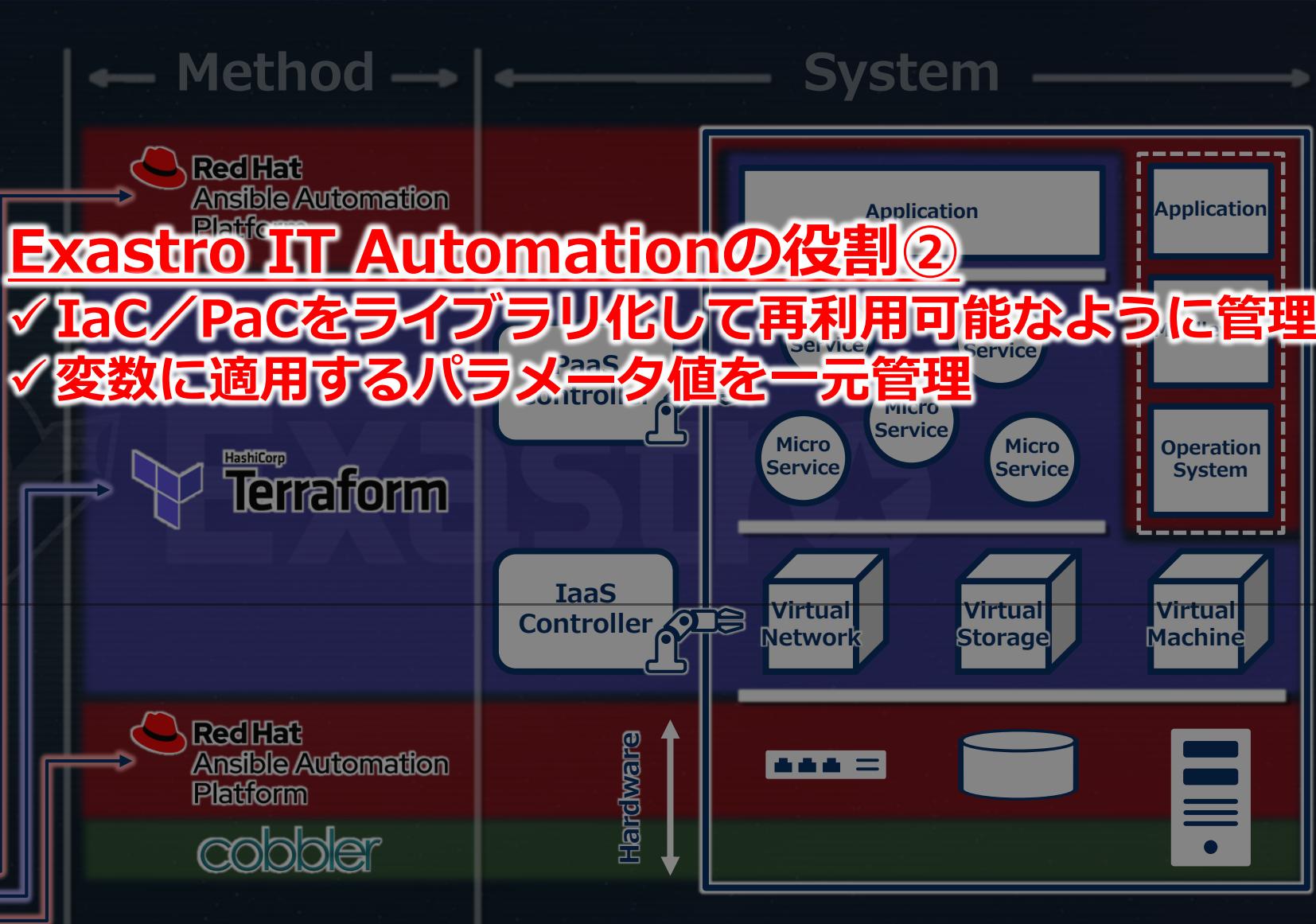
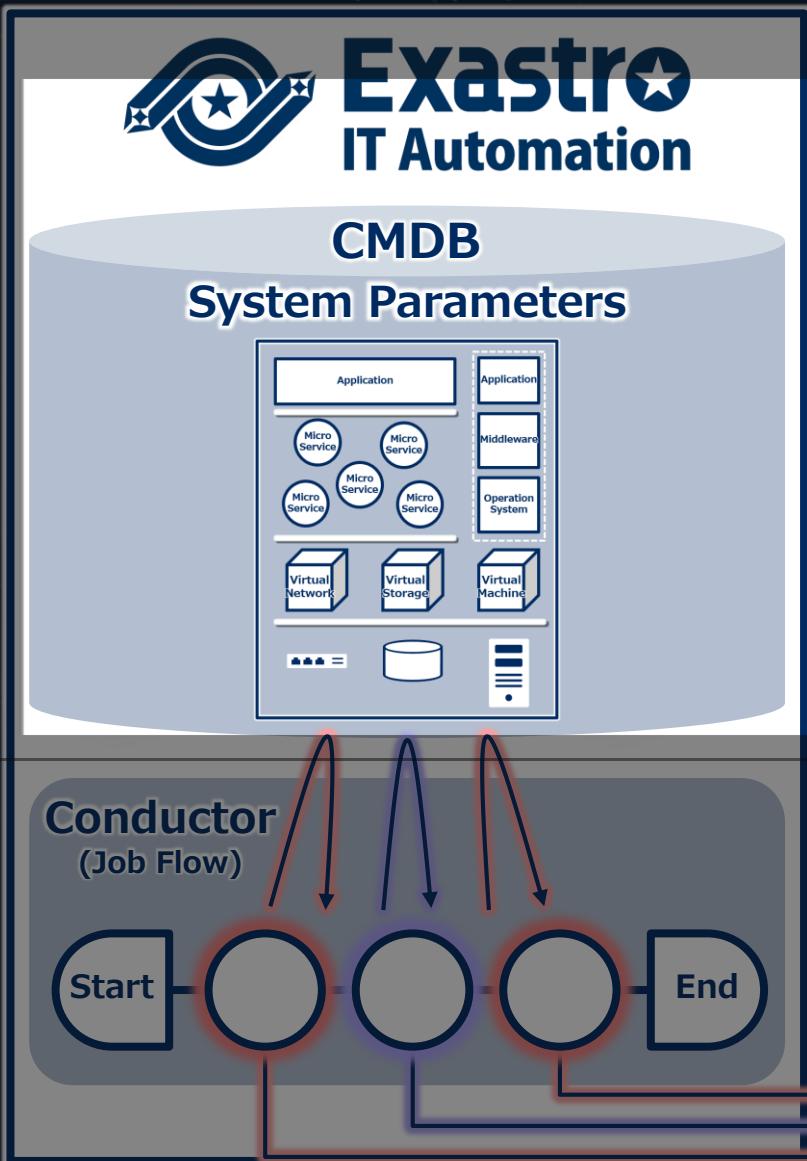
Exastro ITAからAnsibleとTerraformを相互補完して活用することは？



Exastro ITAからAnsibleとTerraformを相互補完して活用することは？



Exastro ITAからAnsibleとTerraformを相互補完して活用することは？



[参考]Exastro IT Automation 関連記事

- [Exastro IT Automationをインストールしてみた（v1.6.0）](#)
- [Exastro IT Automationを実際に動かしてみた（クイックスタート）](#)
- [Exastro IT Automation ver1.6.0のキホンの“キ”](#)
- [Exastro × Ansibleでやってみよう♪ ネットワーク機器の自動設定](#)
- [Exastroコミュニティ（Github）](#)



デモ

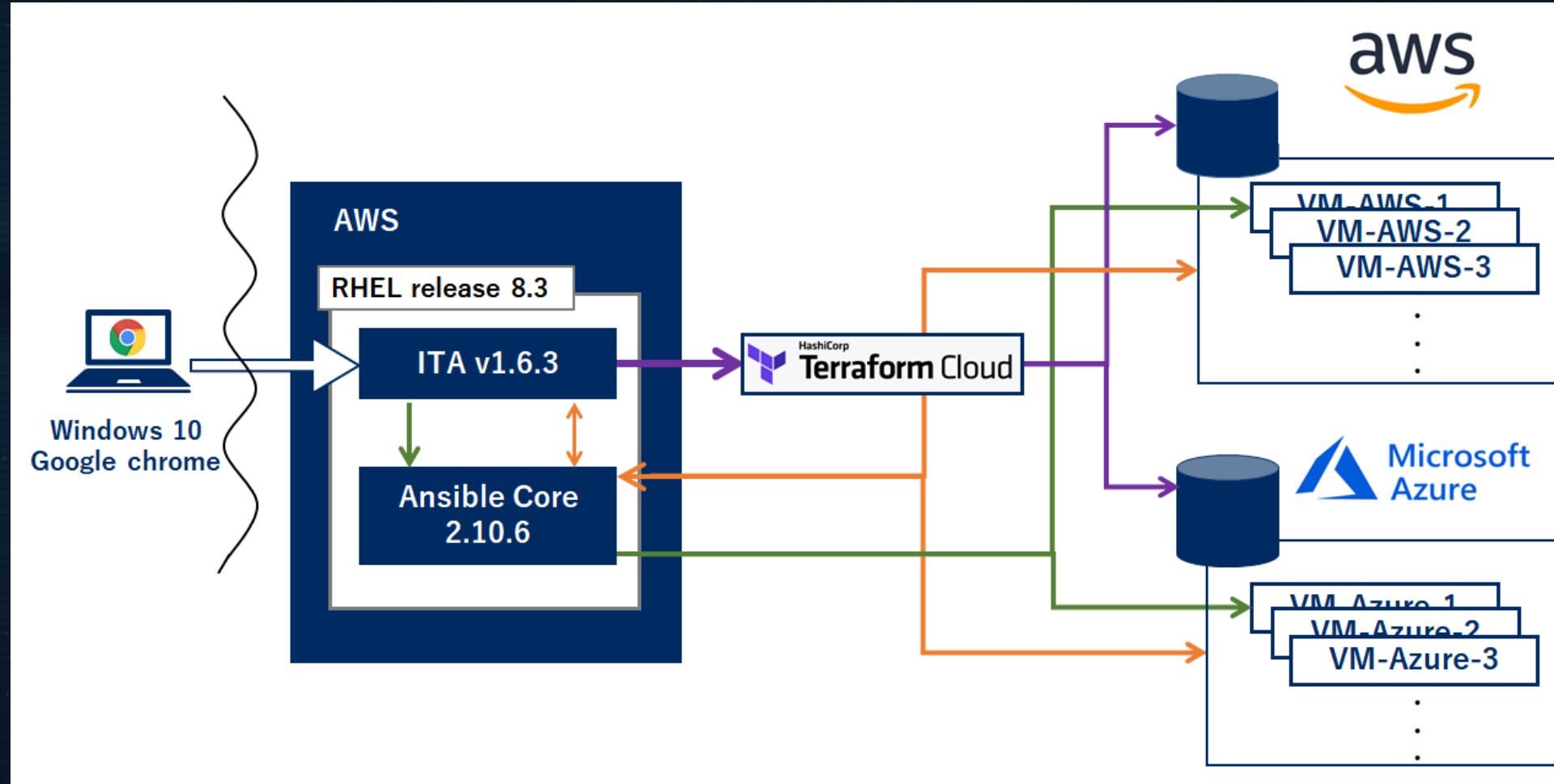


デモ環境の構成イメージ

詳細は以下Qiita記事へ。

「マルチクラウドプロビジョニング(AWS+Azure)を「Exastro×Terraform×Ansible」で実施してみた」

<https://qiita.com/standsetx/items/5e20e9c30bb5efa9b12f>



～デモ～

デモ実施前の状態

- ✓ AWS : 何も起動していない状態
- ✓ Azure : 何も起動していない状態

まずは実行してみよう

～デモ～

設定済みのConductorを実行してみる

- ✓ AWS : 3インスタンスデプロイ→Webサーバセッティング
- ✓ Azure : 3インスタンスデプロイ→Webサーバセッティング

Conductor(ジョブフロー)について

✓3つのサブConductorから構成されるConductor

サブConductor① : AWSプロビ + 機器一覧登録

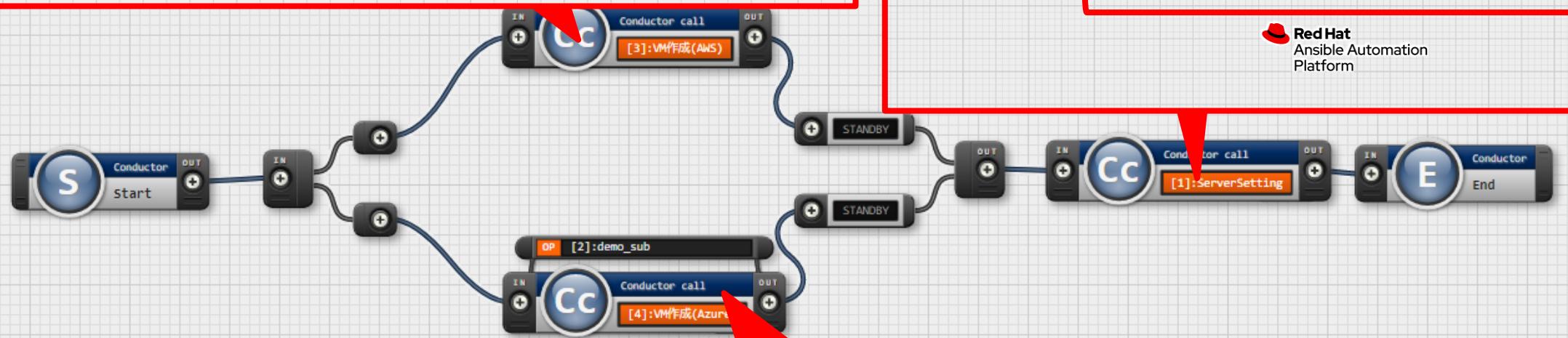


サブConductor③ :

- ・全インスタンス(AWS+Azure)をWebサーバ化
- ・コンテンツの配布



Red Hat Ansible Automation Platform

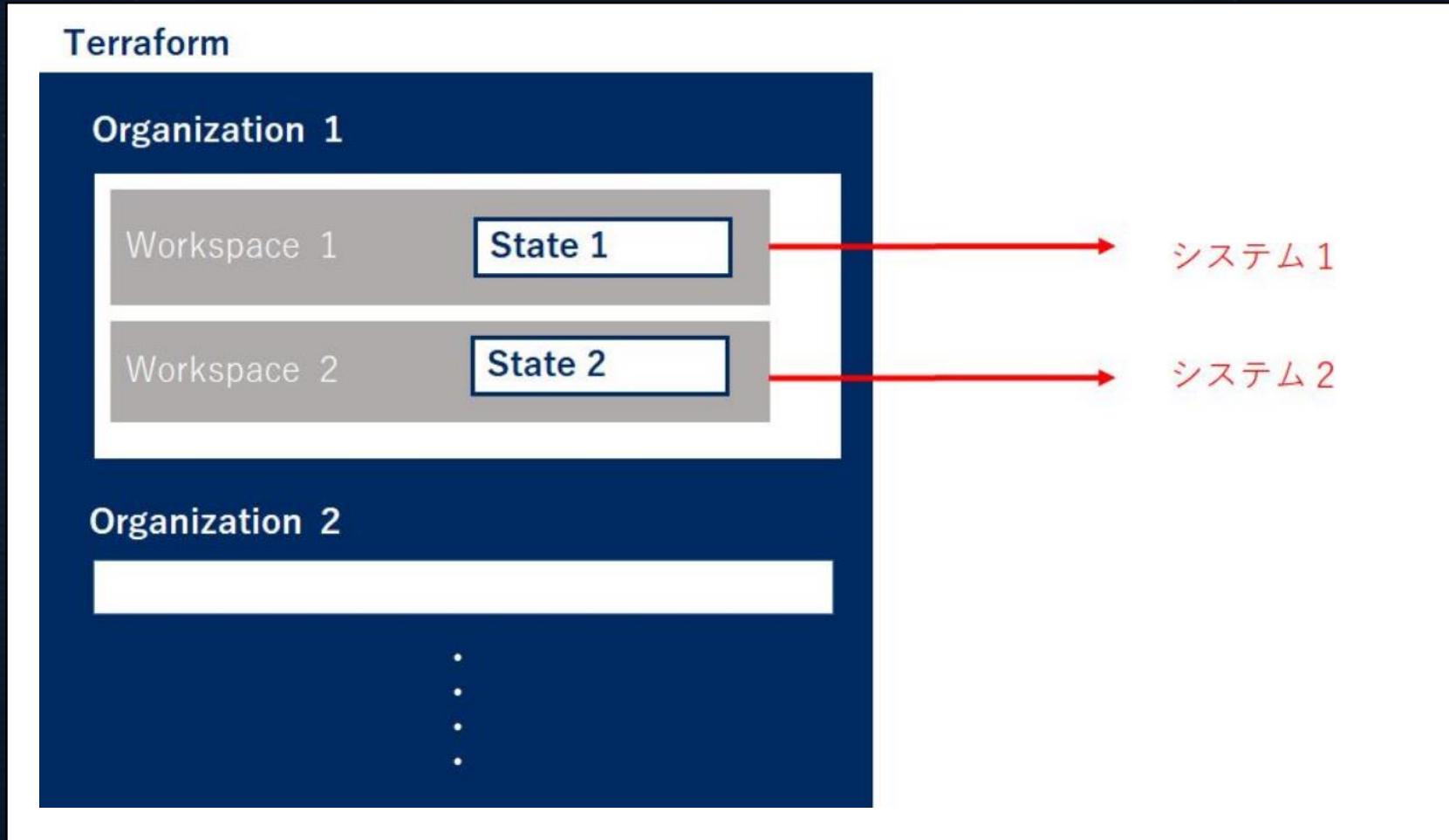


サブConductor② : Azureプロビ + 機器一覧登録



Terraformについて

- ✓リソースのデプロイを行うために“tfstate”と呼ばれるファイルを作成
- ✓stateファイルはOrganization及びWorkSpaceに所属する



～デモ～

Terraform ドライバを利用して以下のセッティング

- ✓ 接続情報 (Token)
- ✓ Organization の登録
- ✓ WorkSpace の作成
- ✓ Movement の設定

Terraform Module素材(IaC)について

AWS用：

- ✓ `aws_create_instance_variables.tf` . . . 変数定義ファイル
- ✓ `aws_create_instance.tf` . . . AWSインスタンス作成用リソース定義ファイル

Azure用：

- ✓ `azure_create_instance_variables.tf` . . . 変数定義ファイル
- ✓ `azure_create_instance.tf` . . . Azureインスタンス作成用リソース定義ファイル

(参考) aws_create_instance_variables.tf (変数定義用)

aws_create_instance_variables.tf

```
variable "access_key" {}

variable "secret_key" {}

variable "region" {}

variable "ami" {}

variable "key_name" {}

variable "security_group" {}

variable "tags_name" {}

variable "hello_tf_instance_count" {
    default = 2
}

variable "hello_tf_instance_type" {
    default = "t2.micro"
}
```

(参考) aws_create_instance.tf (リソース定義ファイル)

aws_create_instance.tf

```
terraform {
  required_version = "~> 0.12"
}

provider "aws" {
  access_key     = var.access_key
  secret_key    = var.secret_key
  region        = var.region
}

resource "aws_instance" "hello-tf-instance" {
  ami           = var.ami
  key_name      = var.key_name
  security_groups = [var.security_group]
  tags = {
    Name = "${var.tags_name}-${count.index+1}"
  }
  count = var.hello_tf_instance_count
  instance_type = var.hello_tf_instance_type
}
```



(参考) azure_create_instance_variables.tf (変数定義用)

```
azure_create_instance_variables.tf

variable "subscription_id" {}
variable "tenant_id" {}
variable "client_id" {}
variable "client_secret" {}
variable "resource_group_name" {}
variable "security_group" {}
variable "location" {}
variable "Vnet_name" {}
variable "Vnet_address_space" {}
variable "subnet_name" {}
variable "address_prefixes" {}
variable "public_ip_name" {}
variable "allocation_method" {}
variable "domain_name_label" {}
variable "network_interface_name" {}
variable "NIC_name" {}
variable "VM_name" {}
variable "VM_size" {}
variable "publisher" {}
variable "offer" {}
variable "sku" {}
variable "source_image_version" {}
variable "admin_username" {}
variable "ssh_public_key" {}
variable "os_disk_name" {}
variable "caching" {}
variable "storage_account_type" {}
```



(参考) azure_create_instance.tf (リソース定義ファイル)

```
azure_create_instance.tf

provider "azurerm" {
    features {}

    subscription_id = var.subscription_id
    client_id       = var.client_id
    client_secret   = var.client_secret
    tenant_id       = var.tenant_id
}

resource "azurerm_resource_group" "hogehoge" {
    name = var.resource_group_name
    location = var.location
}

resource "azurerm_network_security_group" "hogehoge" {
    name = var.security_group
    location = azurerm_resource_group.hogehoge.location
    resource_group_name = azurerm_resource_group.hogehoge.name

    security_rule {
        name          = "SSH"
        priority      = 1001
        direction     = "Inbound"
        access        = "Allow"
        protocol      = "Tcp"
        source_port_range = "*"
        destination_port_range = "22"
        source_address_prefix = "*"
        destination_address_prefix = "*"
    }
    security_rule {

```

～デモ～

Terraformドライバを利用して以下のセッティング

- ✓ パラメータシート
- ✓ 代入値自動登録設定

[Tips] パラメータシートについて

✓ Terraform用パラメータシートとAnsible用パラメータシートは別で作成

The diagram illustrates the separation of parameter sheets between Terraform and Ansible.

Terraform: Represented by its logo. A blue arrow points from the text "「パラメータシート（オペレーションあり）」で作成する" (Created with parameter sheet (operation)) to the "EC2連携(AWS)" button in the main menu. Another blue arrow points from the same text to the "VM連携(Azure)" button.

Ansible: Represented by the Red Hat Ansible Automation Platform logo. A red arrow points from the text "「パラメータシート（ホスト/オペレーションあり）」で作成する" (Created with parameter sheet (host/operation)) to the "Webサーバ設定" button in the main menu.

Dashboard: Shows the main interface with various icons and labels. The "Menu" section lists "インスタンス作成(AWS)", "EC2連携(AWS)", "インスタンス作成(Azure)", "VM連携(Azure)", and "Webサーバ設定". The "DASHBOARD" section includes a "メニュー グループ" section with "管理コンソール", "基本コンソール", "エクスポート…", "Conductor", "メニュー作成", and "代入値自動登…". Below this are sections for "参照用", "ホストグルー…", "Ansible 共通", "Ansible-Lega…", "Ansible-Pion…", and "Ansible-Lega…". At the bottom are icons for "Terraform", "コード管理", and "Webサーバ構築".

～デモ～

パラメータシートを変更して再デプロイしてみる

- ✓ AWS : 5インスタンスデプロイ→Webサーバセッティング
- ✓ Azure : 1インスタンスデプロイ→Webサーバセッティング

～デモ～

Ansibleドライバを利用して以下のセッティング

- ✓ Movement
- ✓ パラメータシート
- ✓ 代入値自動登録

～デモ～

Conductorの準備

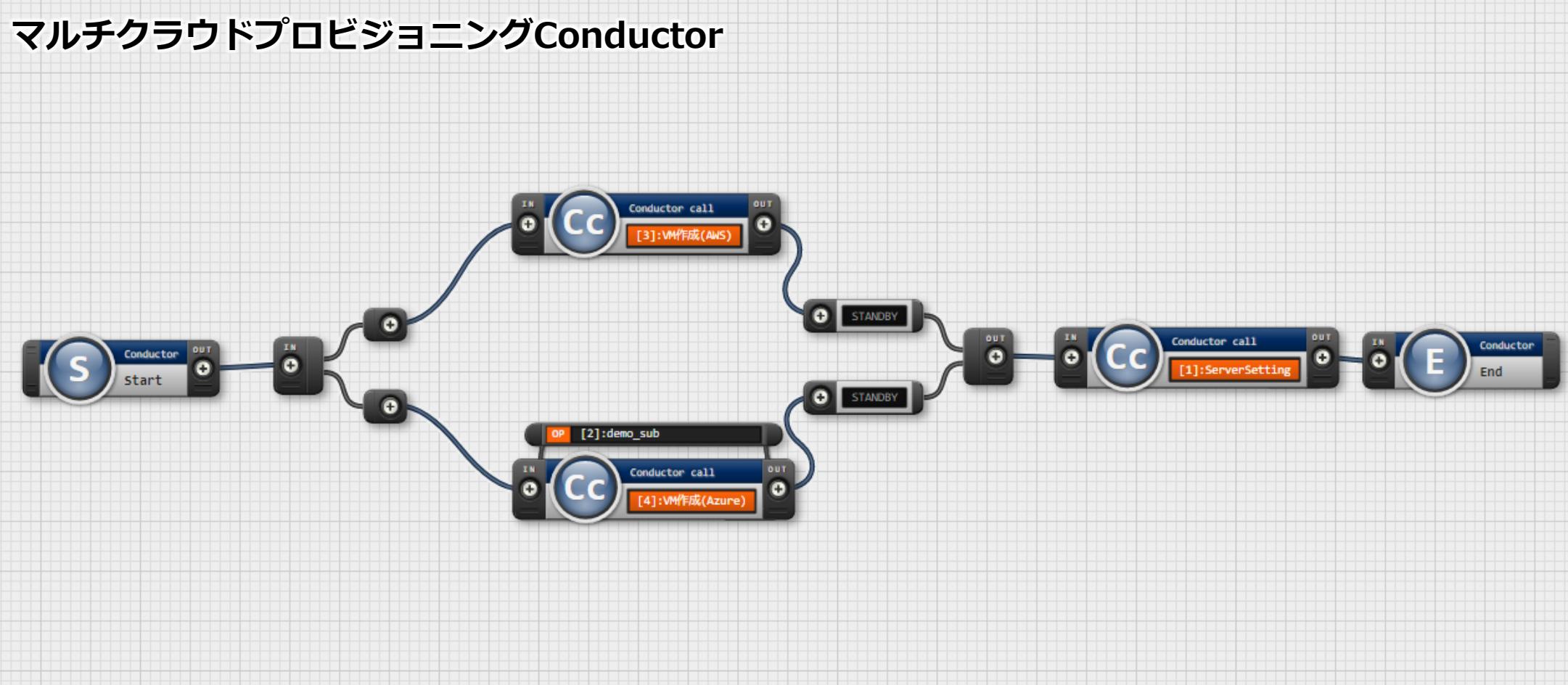
- ✓サブConductor/Movement
- ✓メインConductor



(再掲) Conductor(ジョブフロー)について

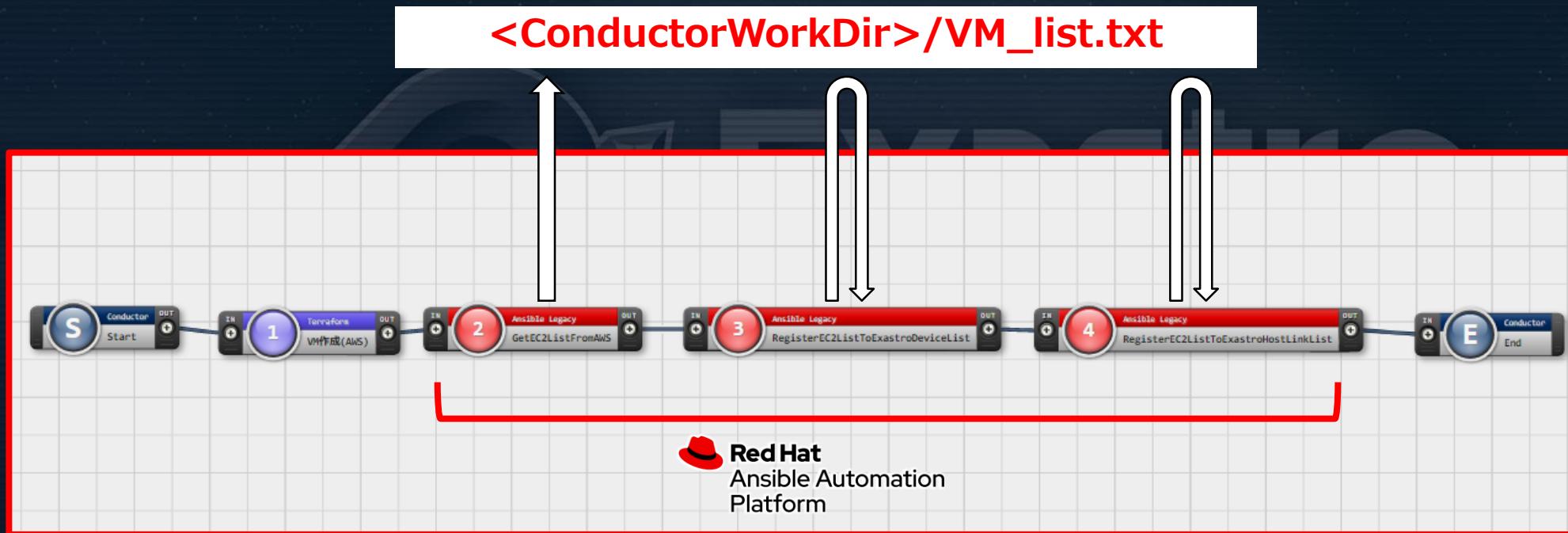
✓3つのサブConductorから構成されるConductor

マルチクラウドプロビジョニングConductor



[Tips] Conductor内でデータを受け渡しする

- ✓ Movement間で情報/ファイルの受け渡し場所に“`__conductor_workflowdir__`”が利用可能
- ✓ Conductor毎に払い出される





Exastro 