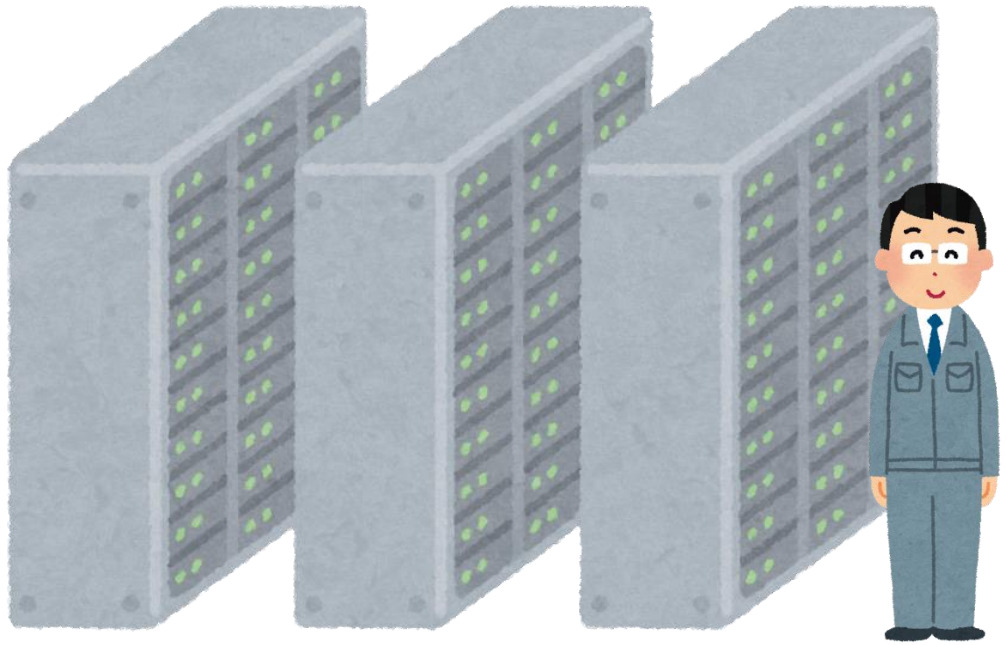
HashiCorp
Terraform Enterprise概要資料

ネットワールド　HashiCorpチーム

もくじ

1. Terraform による Infrastructure as Code とは？
2. Terraform のデモ (動画)
3. Terraform Enterprise とは？
4. Terraform Enterprise のデモ (動画)
5. まとめ



Terraform による Infrastructure as Code とは？

Infrastructure **a**s **C**ode (Iac) とは？

インフラ

CPUやメモリ、ディスクといったリソース、
あるいは仮想マシンやアプライアンスそのもの

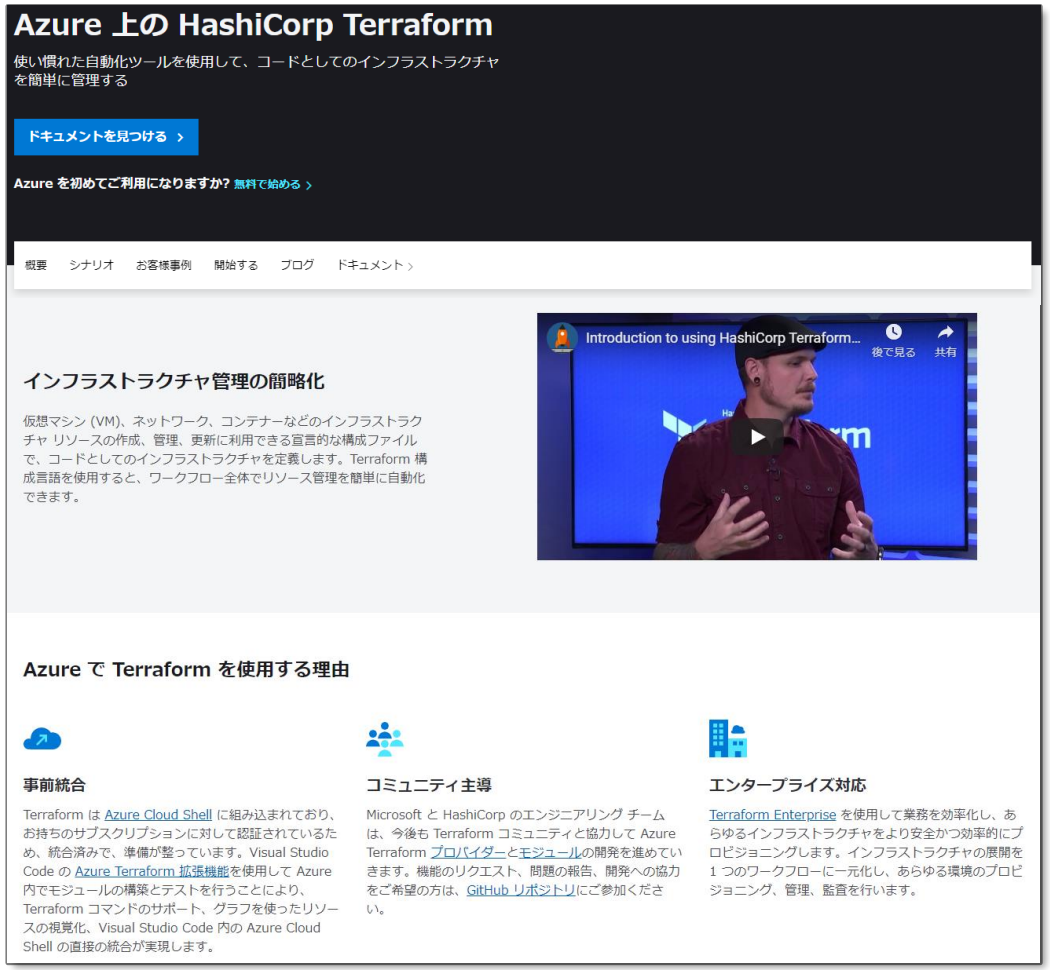
を

コードで表現すること

Terraform で Infrastructure as Code を実現する

再利用可能な
ソースコードとして
インフラの構成を
定義する

多様なインフラを
ワークフローを
変えることなく
導入する



出典：Azure 上の HashiCorp Terraform
https://azure.microsoft.com/ja-jp/solutions/devops/terraform/

出典：Terraformで超サクッとループで
リソースを用意する方法
https://dev.classmethod.jp/cloud/aws/terraform-network-variable/

Terraform による IaC で開発者が自らインフラを構築できる

導入前

ソフトウェア
開発者

インフラ要件
(作業指示書)

インフラ SE が
作業を実施

開発者が
要求した
リソース

導入後

ソフトウェア
開発者

Terraform の
コード (HCL)

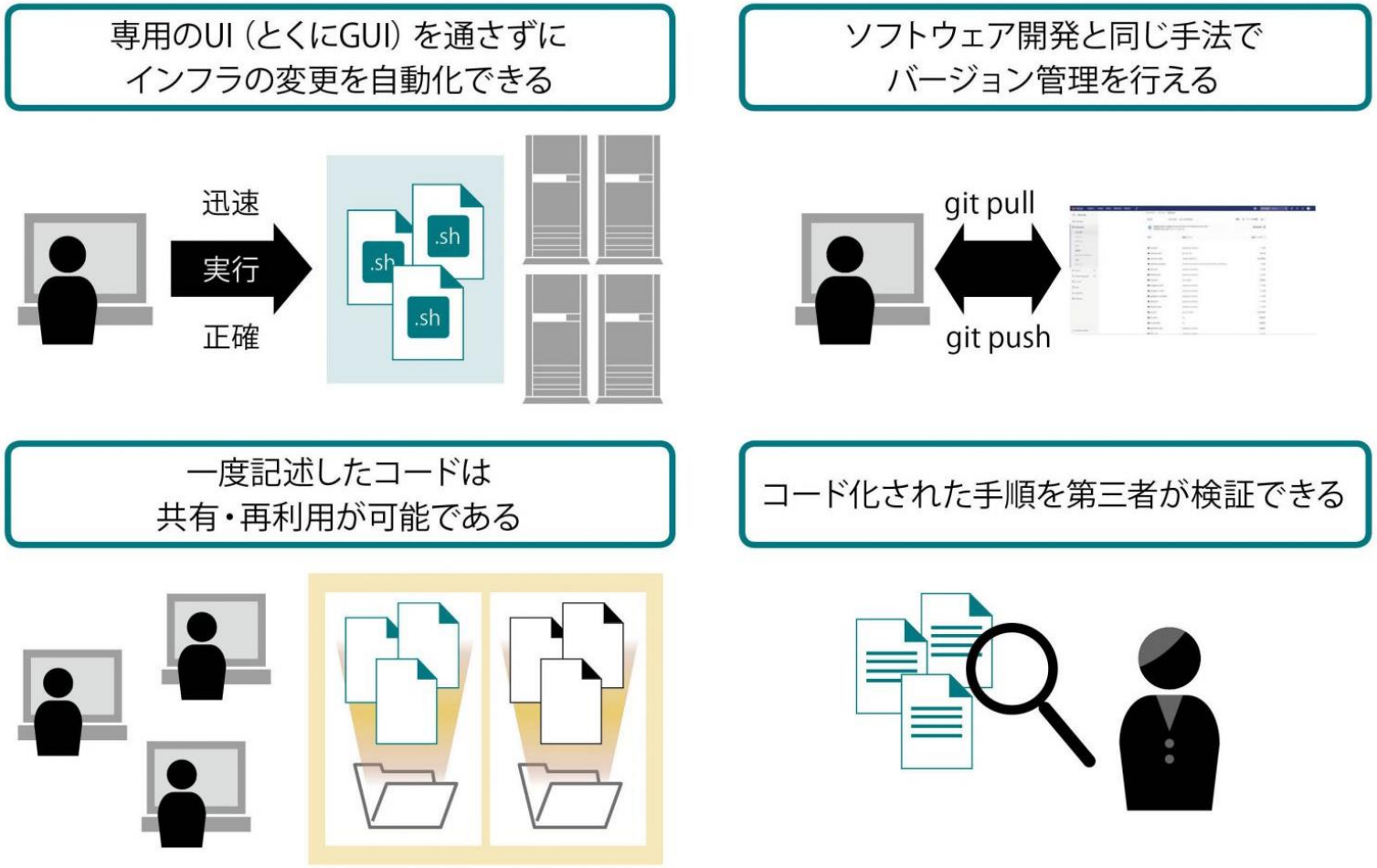
Terraform

Terraform が
プロビジョニング

開発者が
要求した
リソース

Infrastructure as Code のメリットとは

図表3 インフラ構築・運用をコードで記述することのメリット

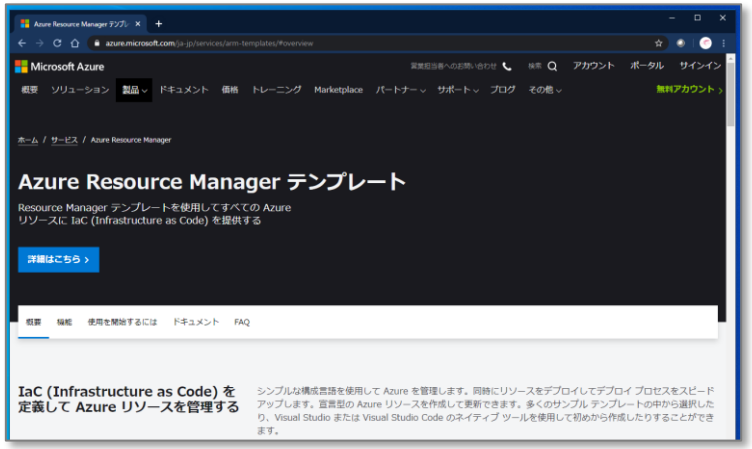


出典：[iMagazine] Infrastructure as Codeの留意点とメリット サーバー更改プロジェクトへの適用で得られた知見・実感
https://www.imagazine.co.jp/infrastructure-as-codeの留意点とメリット%E3%80%80～サーバー更改プロ/

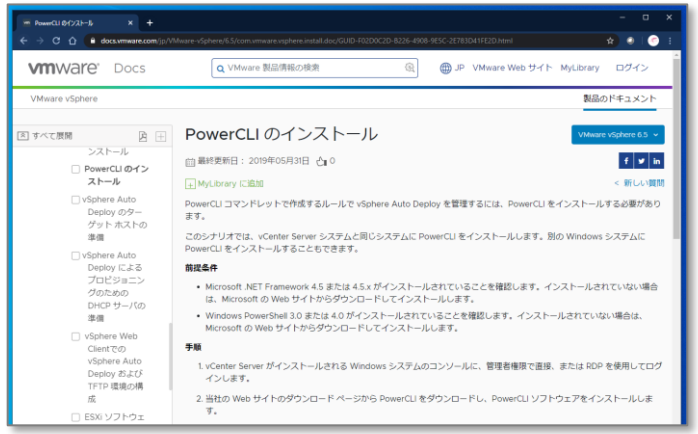
各クラウドやプロダクト専用の IaC ツールはある

AWS なら CloudFormation など

Azure なら ARM テンプレートなど

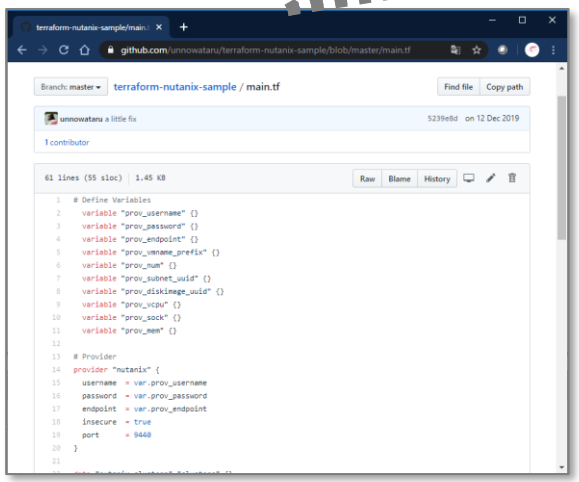


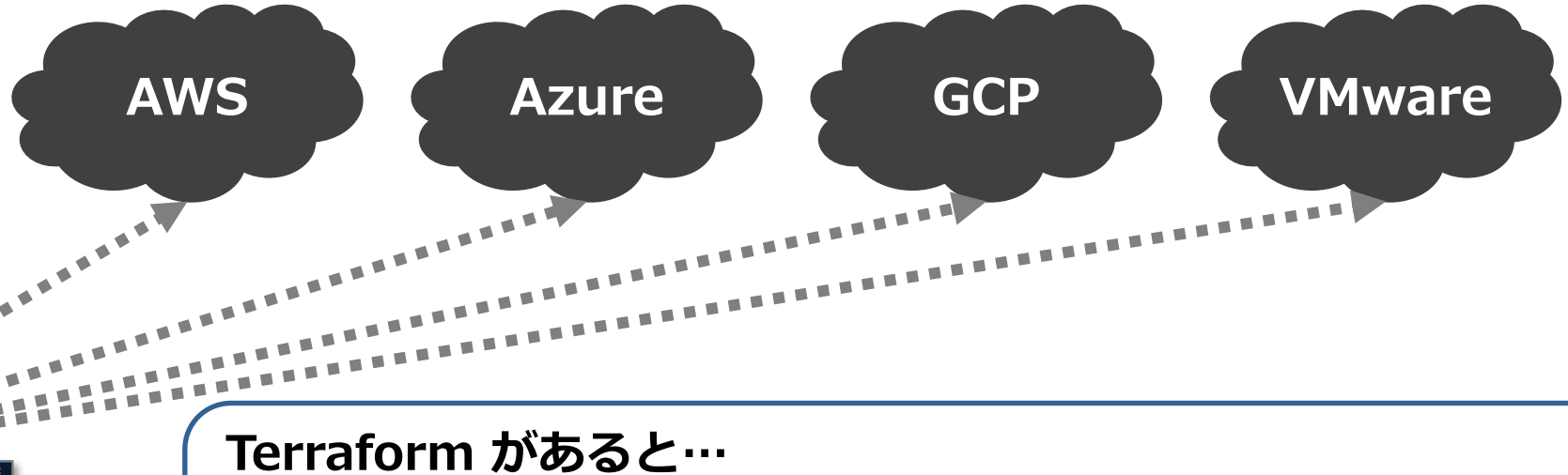
VMware なら PowerCLI など



Ansible や Puppet などは もう少し OS に寄ってるイメージ…

Terraform：ひとつのコード体系 (HCL) で さまざまな環境に対応





Terraform があると…

- **異なるクラウド/オンプレミスでも単一のコードで対応可能**
- **環境変更の前にレビューがしやすい**
- **コードに基づいた自動作業だから作業漏れが発生しない**
- **現在の状態を State ファイルで確認できる**
- **環境の変更履歴が追える**
- **作成したコードを再利用できる**

Terraform だけでこれだけの環境のプロビジョニングが可能

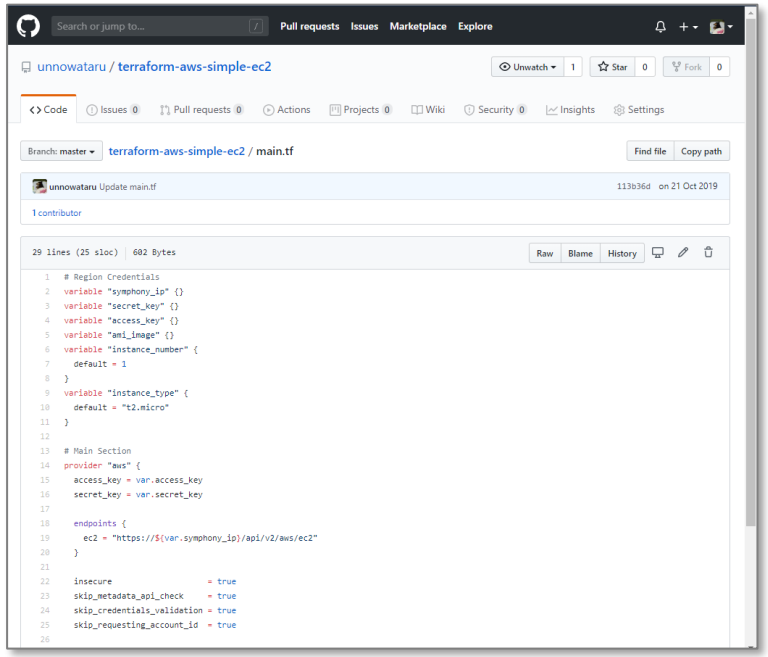
ACME	Cloudflare	GitHub	Local	OVH	Spotinst
Akamai	CloudScale.ch	GitLab	Logentries	Packet	StatusCake
Alibaba Cloud	CloudStack	Google Cloud Platform	LogicMonitor	PagerDuty	TelefonicaOpenCloud
Archive	Cobbler	Grafana	Mailgun	Palo Alto Networks	Template
Arukas	Consul	Gridscale	MongoDB Atlas	PostgreSQL	TencentCloud
Avi Vantage	Datadog	Hedvig	MySQL	PowerDNS	Terraform
Aviatrix	DigitalOcean	Helm	Naver Cloud	ProfitBricks	Terraform Cloud
AWS	DNS	Heroku	Netlify	Pureport	TLS
Azure	DNSimple	Hetzner Cloud	New Relic	RabbitMQ	Triton
Azure Active Directory	DNSMadeEasy	HTTP	Nomad	Rancher	UCloud
Azure Stack	Docker	HuaweiCloud	NS1	Rancher2	UltraDNS
A10 Networks	Dome9	HuaweiCloudStack	Null	Random	Vault
Bitbucket	Dyn	Icinga2	Nutanix	RightScale	Venafi
Brightbox	Exoscale	Ignition	1&1	Rundeck	VMware NSX-T
CenturyLinkCloud	External	InfluxDB	OpenStack	RunScope	VMware vCloud Director
Chef	F5 BIG-IP	JDCloud	OpenTelekomCloud	Scaleway	VMware vRA7
CherryServers	Fastly	Kubernetes	OpsGenie	Selectel	VMware vSphere
Circonus	FlexibleEngine	LaunchDarkly	Oracle Cloud Infrastructure	SignalFx	Vultr
Cisco ASA	FortiOS	Librato	Oracle Cloud Platform	Skytap	Yandex
Cisco ACI	Genymotion	Linode	Oracle Public Cloud	SoftLayer	

出典：HashiCorp Terraform Providers
https://www.terraform.io/docs/providers/index.html

コードで 開発者 (Developer) が必要なリソースを割り当て

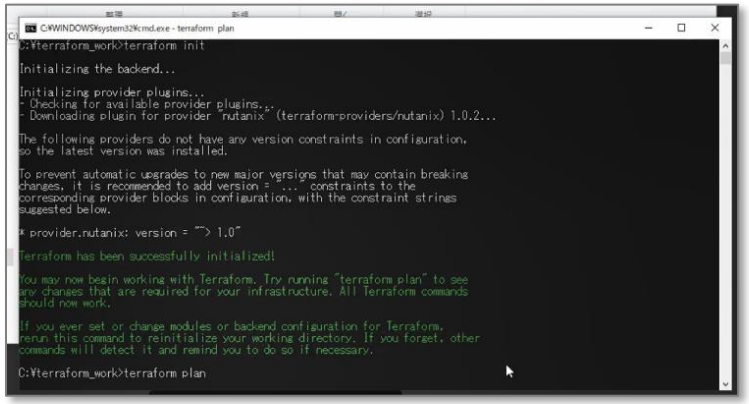
Terraform と AWS の組み合わせ例

コードの作成

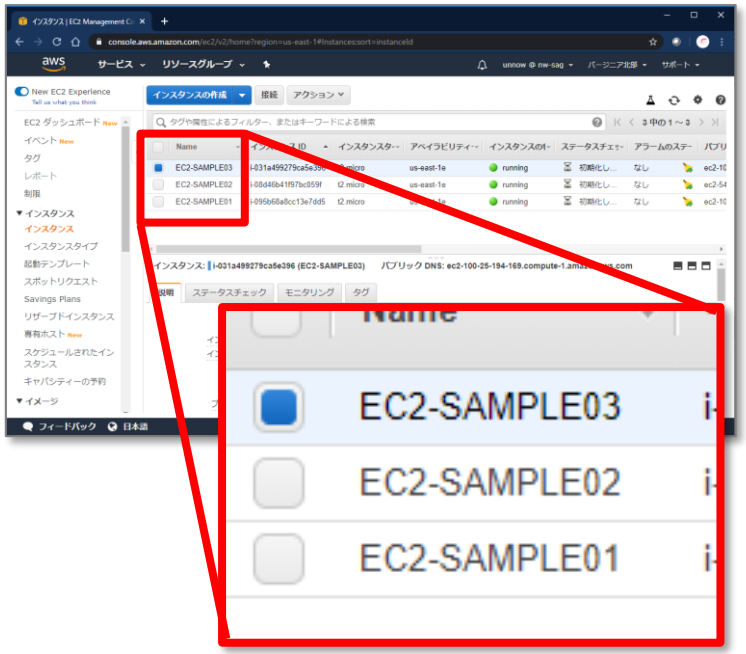


コードの実行

- Terraform INIT
- Terraform PLAN
- Terraform APPLY
- Terraform DESTROY



仮想マシンの作成・削除

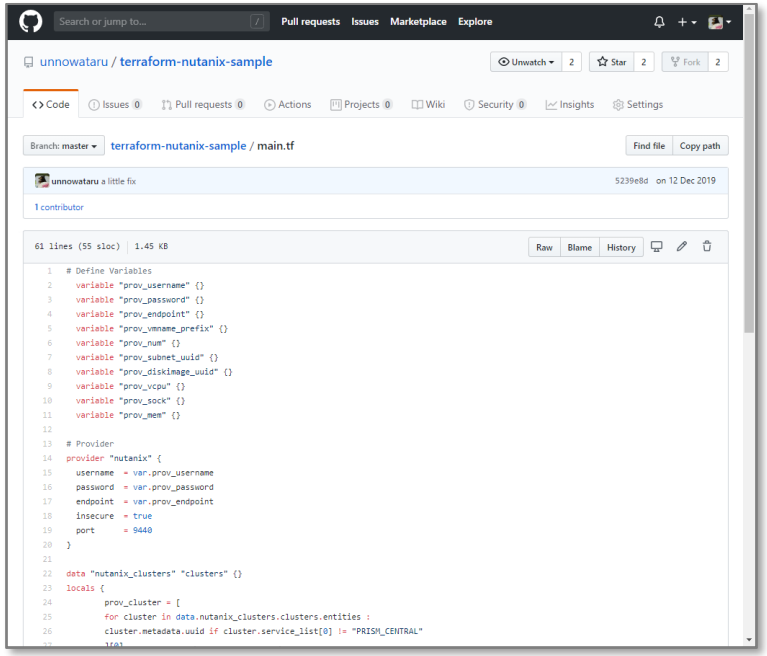


https://github.com/unnowataru/terraform-aws-simple-ec2

コードで 開発者 (Developer) が必要なリソースを割り当て

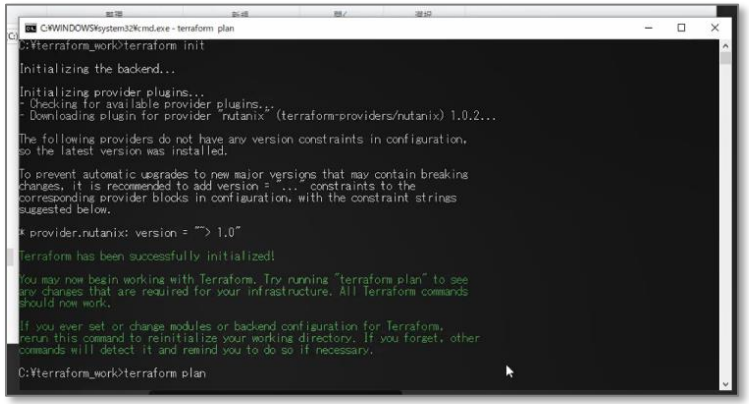
Terraform と Nutanix AHV の組み合わせ例

コードの作成

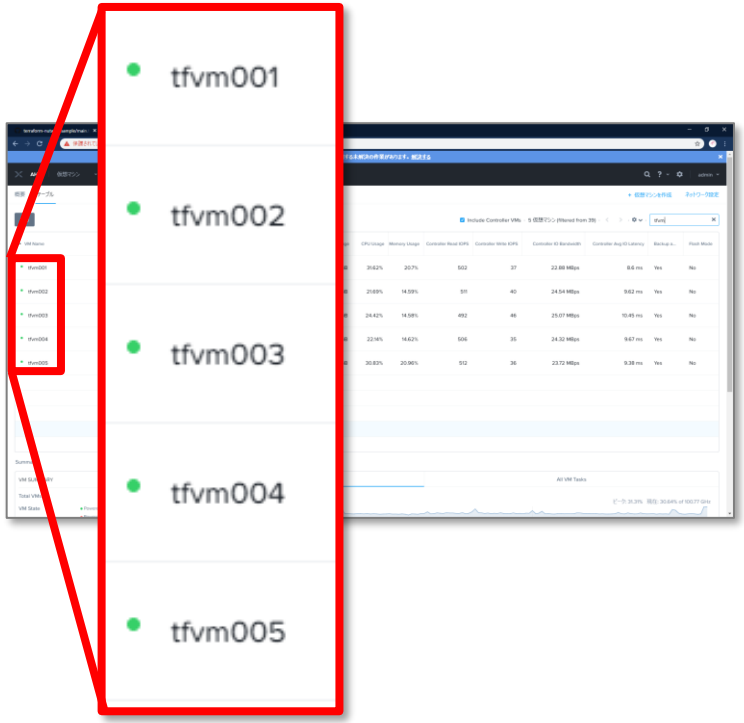


コードの実行

- Terraform INIT
- Terraform PLAN
- Terraform APPLY
- Terraform DESTROY



仮想マシンの作成・削除



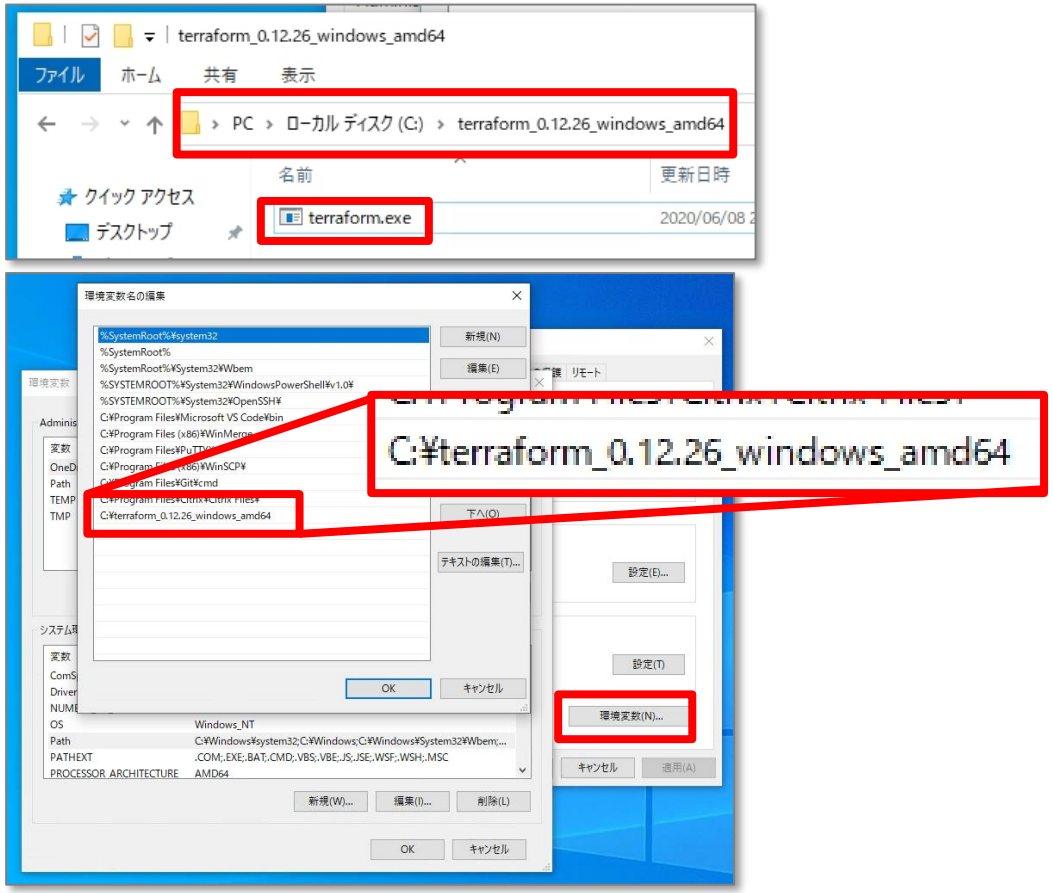
https://github.com/unnowataru/terraform-nutanix-sample

Terraform を実行するためのステップ

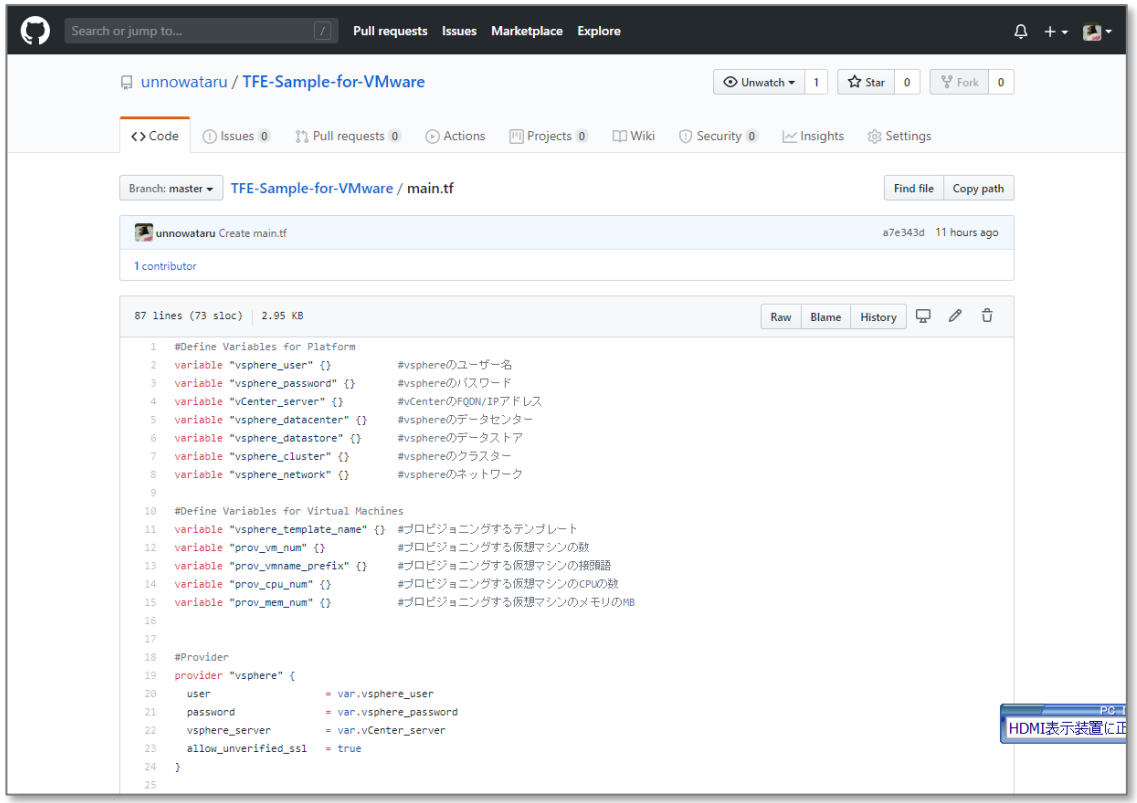
[STEP1] インストールと コードの作成	**Terraform をインストールし、定義ファイルを作成・配置する** (定義ファイルとは 拡張子 .tf のテキストデータ)
[STEP2] Terraform INIT	**Terraform プロバイダーを設定する** (プロバイダーとは、プロビジョニング対象製品のための Terraform のプラグイン)
[STEP3] Terraform PLAN	**Terraform が定義ファイルに基づいて、各製品の API に対応したコードを自動生成、検証(リハーサル/ドライラン)する**
[STEP4] Terraform APPLY	**実際にコードを適用する**

[STEP1] インストール と コードの作成

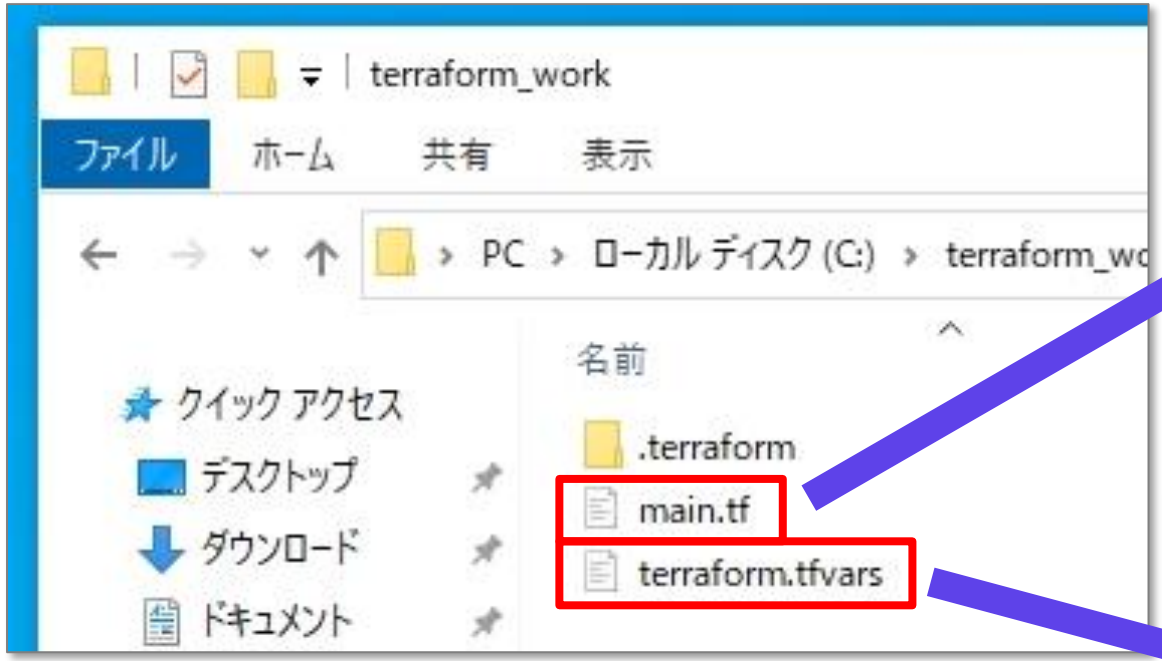
インストール



コードの作成



Terraform のコード：vSphere 向けサンプル



[main.tf]

Terraformのメインコード

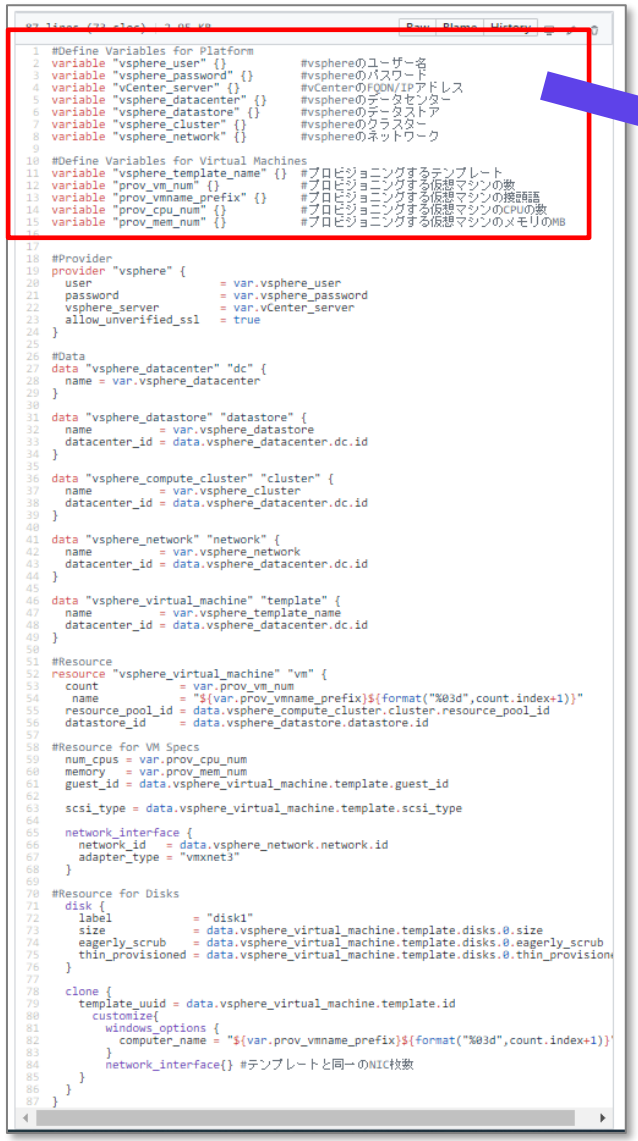
[terraform.tfvars]

変数のみを格納しているファイル

いますぐダウンロードできます!!

https://github.com/unnowataru/TFE-Sample-for-VMware

Terraform のコード解説：main.tf



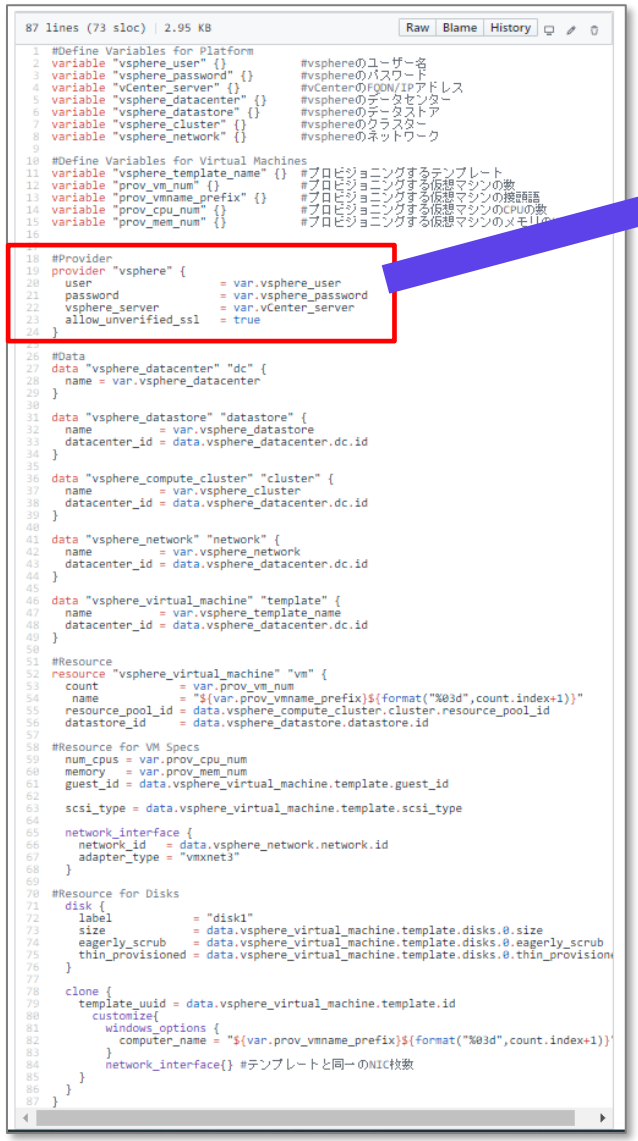
```
#Define Variables for Platform
variable "vsphere_user" {}          #vsphereのユーザー名
variable "vsphere_password" {}      #vsphereのパスワード
variable "vCenter_server" {}        #vCenterのFQDN/IPアドレス
variable "vsphere_datacenter" {}    #vsphereのデータセンター
variable "vsphere_datastore" {}     #vsphereのデータストア
variable "vsphere_cluster" {}       #vsphereのクラスター
variable "vsphere_network" {}       #vsphereのネットワーク

#Define Variables for Virtual Machines
variable "vsphere_template_name" {}  #プロビジョニングするテンプレート
variable "prov_vm_num" {}            #プロビジョニングする仮想マシンの数
variable "prov_vmname_prefix" {}     #プロビジョニングする仮想マシンの接頭語
variable "prov_cpu_num" {}           #プロビジョニングする仮想マシンのCPUの数
variable "prov_mem_num" {}           #プロビジョニングする仮想マシンのメモリのMB
```

variable：変数を定義するセクション

ユーザー名やvCenterのFQDN、プロビジョニングしたい仮想マシンの名前、スペックなどを変数として定義している。

```
1   #Define Variables for Platform
2   variable "vsphere_user" {}          #vsphereのユーザー名
3   variable "vsphere_password" {}      #vsphereのパスワード
4   variable "vCenter_server" {}        #vCenterのFQDN/IPアドレス
5   variable "vsphere_datacenter" {}    #vsphereのデータセンター
6   variable "vsphere_datastore" {}     #vsphereのデータストア
7   variable "vsphere_cluster" {}       #vsphereのクラスター
8   variable "vsphere_network" {}       #vsphereのネットワーク
9
10  #Define Variables for Virtual Machines
11  variable "vsphere_template_name" {}  #プロビジョニングするテンプレート
12  variable "prov_vm_num" {}            #プロビジョニングする仮想マシンの数
13  variable "prov_vmname_prefix" {}     #プロビジョニングする仮想マシンの接頭語
14  variable "prov_cpu_num" {}           #プロビジョニングする仮想マシンのCPUの数
15  variable "prov_mem_num" {}           #プロビジョニングする仮想マシンのメモリのMB
```

Terraform のコード解説：main.tf



provider：インフラを定義するセクション

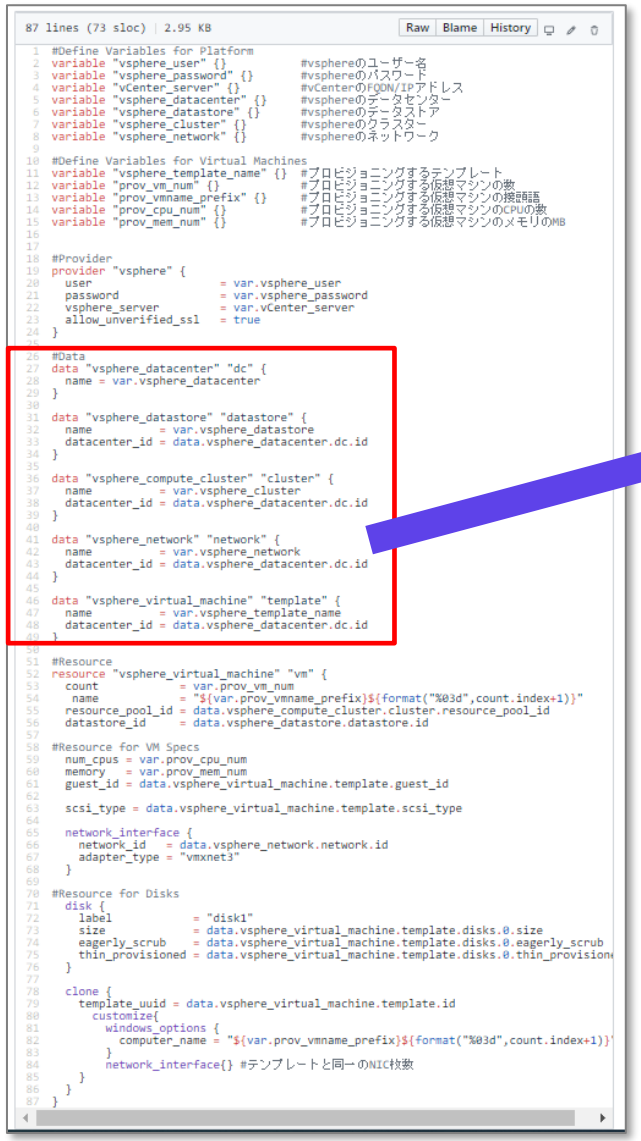
ここでは vsphere としているが、AWS や Azure など
プロビジョニング対象となるインフラをここで定義する。

ちなみに ”vsphere” の場合、ちょっとカッコつけようと思って
“vSphere” とか書いたりすると terraform plan 実行時に怒られる
ことがあるので注意。

これで2時間ぐらいハマりました(;∀;)

```
18  #Provider
19  provider "vsphere" {
20    user                  = var.vsphere_user
21    password              = var.vsphere_password
22    vsphere_server        = var.vCenter_server
23    allow_unverified_ssl  = true
24  }
```

Terraform のコード解説：main.tf



```
87 lines (73 sloc) | 2.95 KB                                    Raw  Blame  History
#Define Variables for Platform
variable "vsphere_user" {}              #vsphereのユーザー名
variable "vsphere_password" {}          #vsphereのパスワード
variable "vCenter_server" {}            #vCenterのFQDN/IPアドレス
variable "vsphere_datacenter" {}        #vsphereのデータセンター
variable "vsphere_datastore" {}         #vsphereのデータストア
variable "vsphere_cluster" {}           #vsphereのクラスター
variable "vsphere_network" {}           #vsphereのネットワーク

#Define Variables for Virtual Machines
variable "vsphere_template_name" {}  #プロビジョニングするテンプレート
variable "prov_vm_num" {}            #プロビジョニングする仮想マシンの数
variable "prov_vmname_prefix" {}     #プロビジョニングする仮想マシンの接頭語
variable "prov_cpu_num" {}           #プロビジョニングする仮想マシンのCPUの数
variable "prov_mem_num" {}           #プロビジョニングする仮想マシンのメモリのMB


#Provider
provider "vsphere" {
  user                 = var.vsphere_user
  password             = var.vsphere_password
  vsphere_server       = var.vCenter_server
  allow_unverified_ssl = true
}

#Data
data "vsphere_datacenter" "dc" {
  name = var.vsphere_datacenter
}

data "vsphere_datastore" "datastore" {
  name          = var.vsphere_datastore
  datacenter_id = data.vsphere_datacenter.dc.id
}

data "vsphere_compute_cluster" "cluster" {
  name          = var.vsphere_cluster
  datacenter_id = data.vsphere_datacenter.dc.id
}

data "vsphere_network" "network" {
  name          = var.vsphere_network
  datacenter_id = data.vsphere_datacenter.dc.id
}

data "vsphere_virtual_machine" "template" {
  name          = var.vsphere_template_name
  datacenter_id = data.vsphere_datacenter.dc.id
}

#Resource
resource "vsphere_virtual_machine" "vm" {
  count            = var.prov_vm_num
   name            = "${var.prov_vmname_prefix}${format("%03d",count.index+1)}"
  resource_pool_id = data.vsphere_compute_cluster.cluster.resource_pool_id
  datastore_id     = data.vsphere_datastore.datastore.id

#Resource for VM Specs
  num_cpus = var.prov_cpu_num
  memory   = var.prov_mem_num
  guest_id = data.vsphere_virtual_machine.template.guest_id

  scsi_type = data.vsphere_virtual_machine.template.scsi_type

  network_interface {
    network_id   = data.vsphere_network.network.id
    adapter_type = "vmxnet3"
  }

#Resource for Disks
  disk {
    label            = "disk1"
    size             = data.vsphere_virtual_machine.template.disks.0.size
    eagerly_scrub    = data.vsphere_virtual_machine.template.disks.0.eagerly_scrub
    thin_provisioned = data.vsphere_virtual_machine.template.disks.0.thin_provision
  }

  clone {
    template_uuid = data.vsphere_virtual_machine.template.id
      customize{
        windows_options {
          computer_name = "${var.prov_vmname_prefix}${format("%03d",count.index+1)}"
        }
        network_interface{} #テンプレートと同一のNIC枚数
    }
  }
}
```

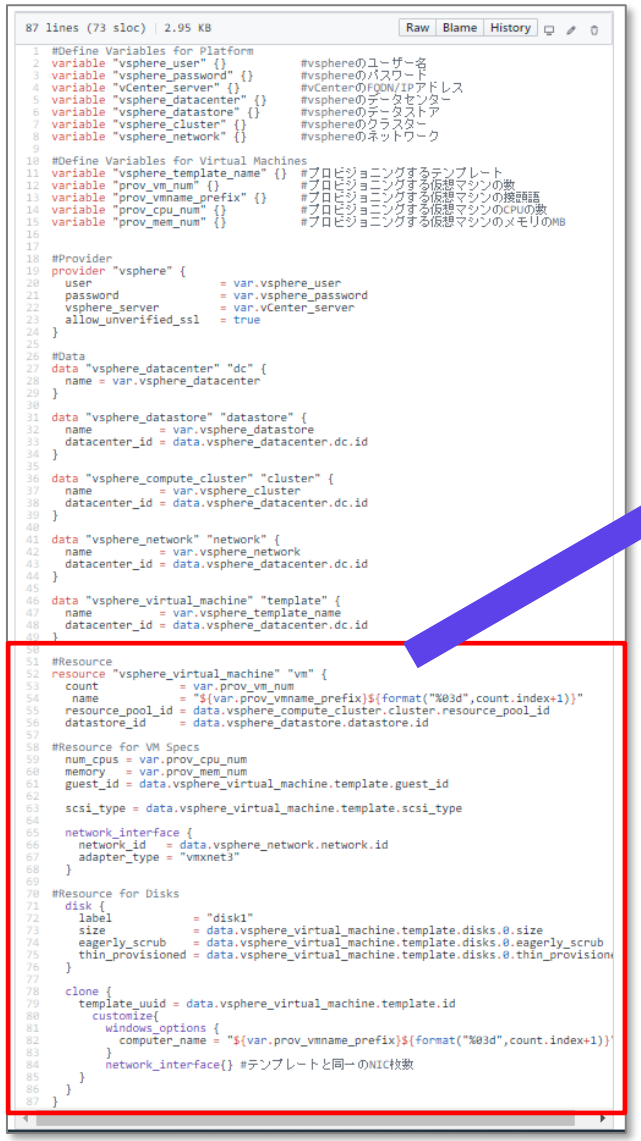
data：既存のリソースから情報を取り込む

変数で指定したオブジェクトから情報を取り込み、プロビジョニングに利用できるようにする。

この例では、指定した仮想マシンテンプレートの情報を取り込み、OSタイプや仮想ハードウェアの情報が Terraform 側で利用できるようになっている。

```
data "vsphere_virtual_machine" "template" {
  name          = var.vsphere_template_name
  datacenter_id = data.vsphere_datacenter.dc.id
}
```

Terraform のコード解説：main.tf




```
#Define Variables for Platform
variable "vsphere_user" {}          #vsphereのユーザー名
variable "vsphere_password" {}      #vsphereのパスワード
variable "vCenter_server" {}        #vCenterのFQDN/IPアドレス
variable "vsphere_datacenter" {}    #vsphereのデータセンター
variable "vsphere_datastore" {}     #vsphereのデータストア
variable "vsphere_cluster" {}       #vsphereのクラスター
variable "vsphere_network" {}       #vsphereのネットワーク

#Define Variables for Virtual Machines
variable "vsphere_template_name" {}  #プロビジョニングするテンプレート
variable "prov_vm_num" {}            #プロビジョニングする仮想マシンの数
variable "prov_vmname_prefix" {}     #プロビジョニングする仮想マシンの接頭語
variable "prov_cpu_num" {}           #プロビジョニングする仮想マシンのCPUの数
variable "prov_mem_num" {}           #プロビジョニングする仮想マシンのメモリのMB


#Provider
provider "vsphere" {
  user                 = var.vsphere_user
  password             = var.vsphere_password
  vsphere_server       = var.vCenter_server
  allow_unverified_ssl = true
}

#Data
data "vsphere_datacenter" "dc" {
  name = var.vsphere_datacenter
}

data "vsphere_datastore" "datastore" {
  name          = var.vsphere_datastore
  datacenter_id = data.vsphere_datacenter.dc.id
}

data "vsphere_compute_cluster" "cluster" {
  name          = var.vsphere_cluster
  datacenter_id = data.vsphere_datacenter.dc.id
}

data "vsphere_network" "network" {
  name          = var.vsphere_network
  datacenter_id = data.vsphere_datacenter.dc.id
}

data "vsphere_virtual_machine" "template" {
  name          = var.vsphere_template_name
  datacenter_id = data.vsphere_datacenter.dc.id
}

#Resource
resource "vsphere_virtual_machine" "vm" {
  count            = var.prov_vm_num
   name            = "${var.prov_vmname_prefix}${format("%03d",count.index+1)}"
  resource_pool_id = data.vsphere_compute_cluster.cluster.resource_pool_id
  datastore_id     = data.vsphere_datastore.datastore.id

#Resource for VM Specs
  num_cpus = var.prov_cpu_num
  memory   = var.prov_mem_num
  guest_id = data.vsphere_virtual_machine.template.guest_id

  scsi_type = data.vsphere_virtual_machine.template.scsi_type

  network_interface {
    network_id   = data.vsphere_network.network.id
    adapter_type = "vmxnet3"
  }

#Resource for Disks
  disk {
    label            = "disk1"
    size             = data.vsphere_virtual_machine.template.disks.0.size
    eagerly_scrub    = data.vsphere_virtual_machine.template.disks.0.eagerly_scrub
    thin_provisioned = data.vsphere_virtual_machine.template.disks.0.thin_provision
  }

  clone {
    template_uuid = data.vsphere_virtual_machine.template.id
      customize{
        windows_options {
          computer_name = "${var.prov_vmname_prefix}${format("%03d",count.index+1)}"
        }
        network_interface{} #テンプレートと同一のNIC枚数
    }
  }
}
```

resource：実際に作成するリソースを定義

変数で指定した情報や data で取り込んだ情報を利用して、対象となるインフラにリソースをプロビジョニング(作成)する。

当然のことながら、HCLはプログラミング言語なので、演算子なども きちんと使える。

ここでは変数 “count” に応じて仮想マシンの名前を1から指定した数字まで、前ゼロ3桁でプロビジョニングするような仕組みになっている。

```
#Resource
resource "vsphere_virtual_machine" "vm" {
  count            = var.prov_vm_num
   name            = "${var.prov_vmname_prefix}${format("%03d",count.index+1)}"
  resource_pool_id = data.vsphere_compute_cluster.cluster.resource_pool_id
  datastore_id     = data.vsphere_datastore.datastore.id
```

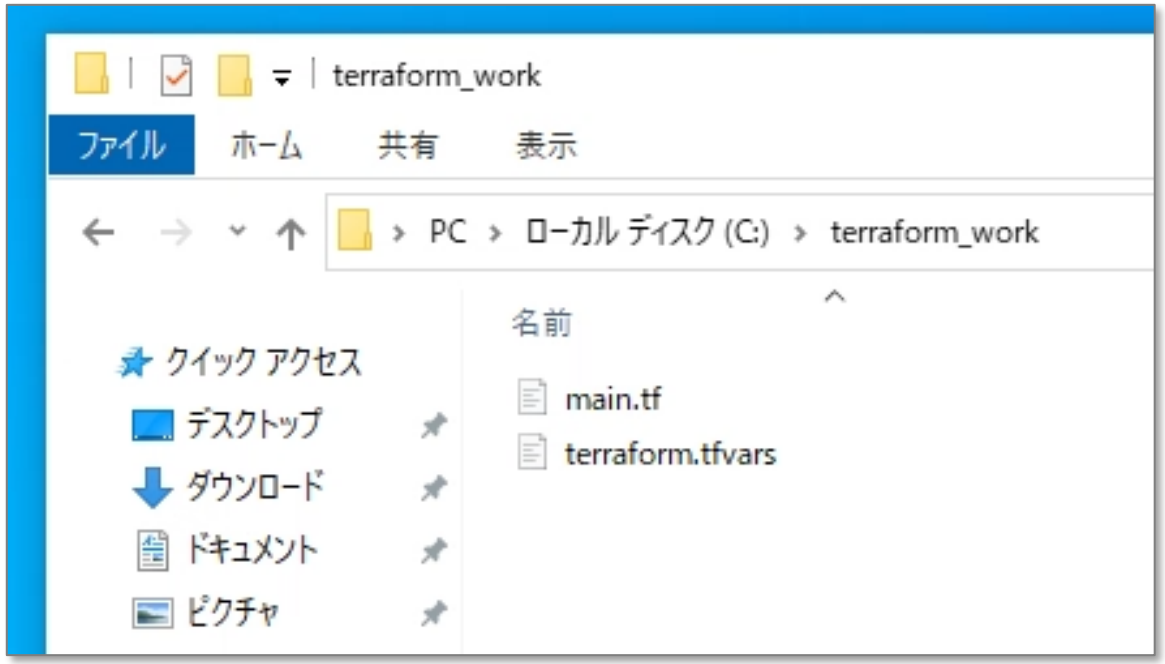
Terraform のコード解説：terraform.tfvars

パスワードやアクセスキーのお漏らしには注意しましょう!!

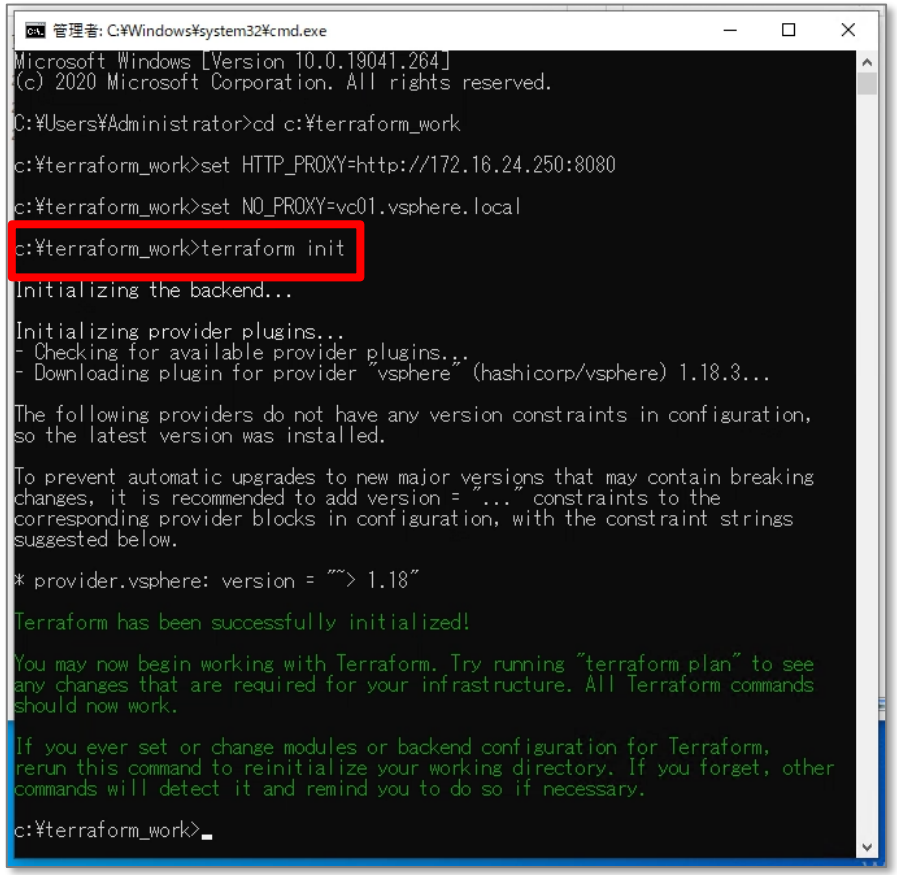
```
#Define Variables for Platform
vsphere_user         = "Administrator@vsphere.local" #vsphereのユーザー名
vsphere_password     = "Netw0rld123!"                #vsphereのパスワード
vCenter_server       = "vc01.vsphere.local"          #vCenterのFQDN/IPアドレス
vsphere_datacenter   = "Datacenter"                  #vsphereのデータセンター
vsphere_datastore    = "datastore1"                  #vsphereのデータストア
vsphere_cluster      = "Cluster"                     #vsphereのクラスター
vsphere_network      = "VM Network"                  #vsphereのネットワーク

#Define Variables for Virtual Machines
vsphere_template_name = "WIN2019"                    #プロビジョニングするテンプレート
prov_vm_num          = 1                             #プロビジョニングする仮想マシンの数
prov_vmname_prefix   = "TFVM"                        #プロビジョニングする仮想マシンの接頭語
prov_cpu_num         = 2                             #プロビジョニングする仮想マシンのCPUの数
prov_mem_num         = 4096                          #プロビジョニングする仮想マシンのメモリのMB
```

[STEP2] Terraform INIT

フォルダーにコードを配置

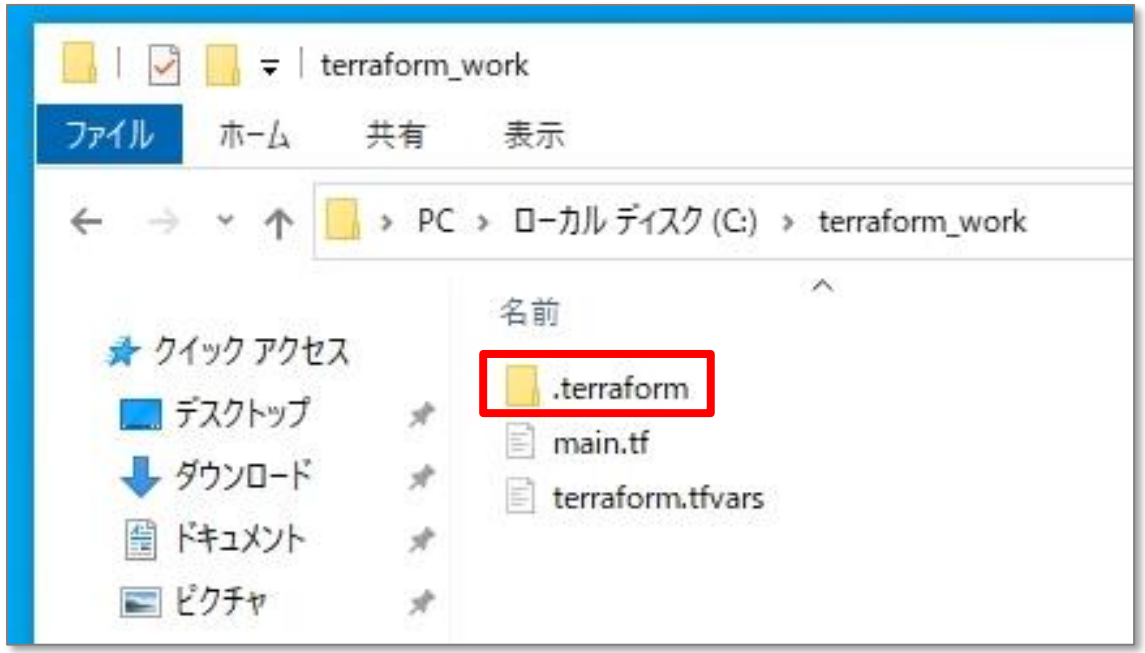


コマンドラインで Terraform INIT

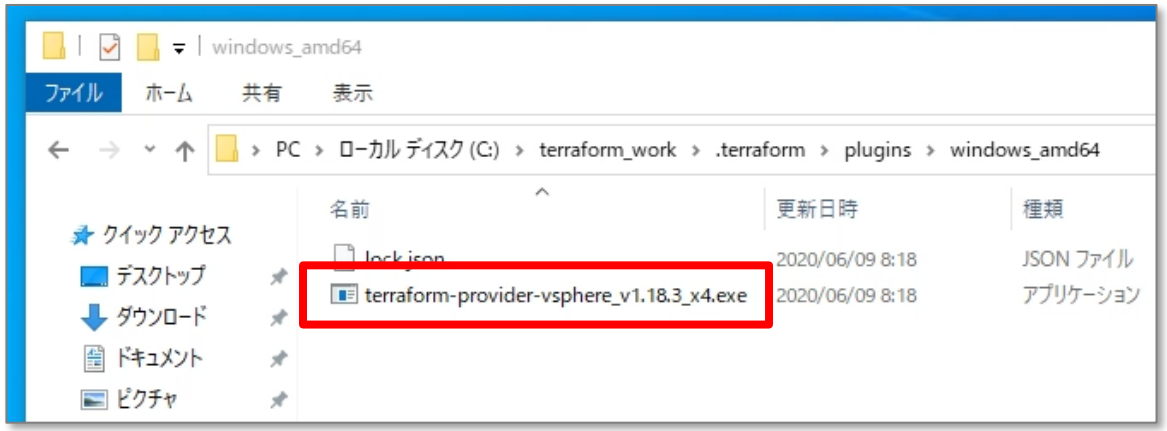


[STEP2] Terraform INIT

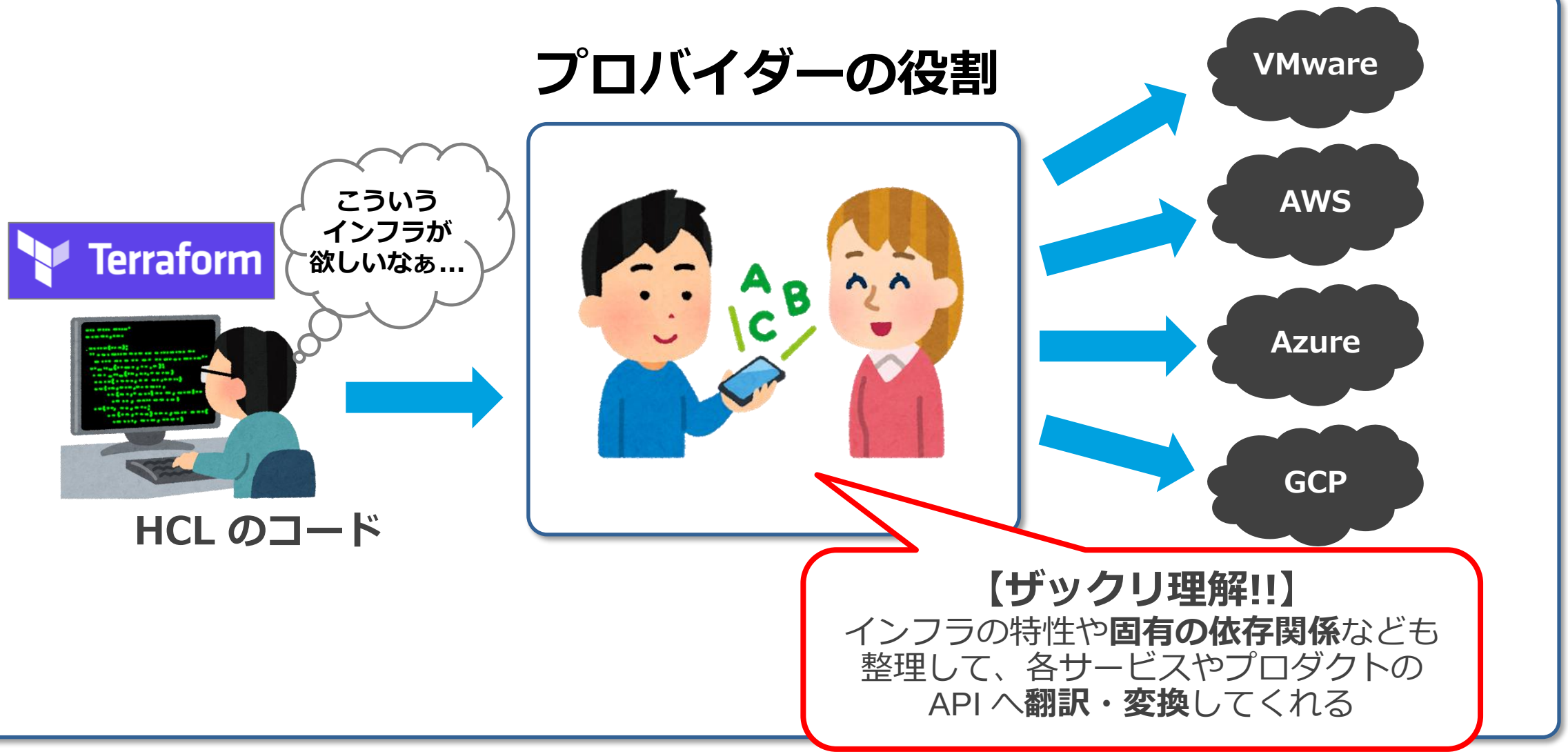
フォルダーにプロバイダーがダウンロードされる



プロバイダーの実体が配置されていることがわかる

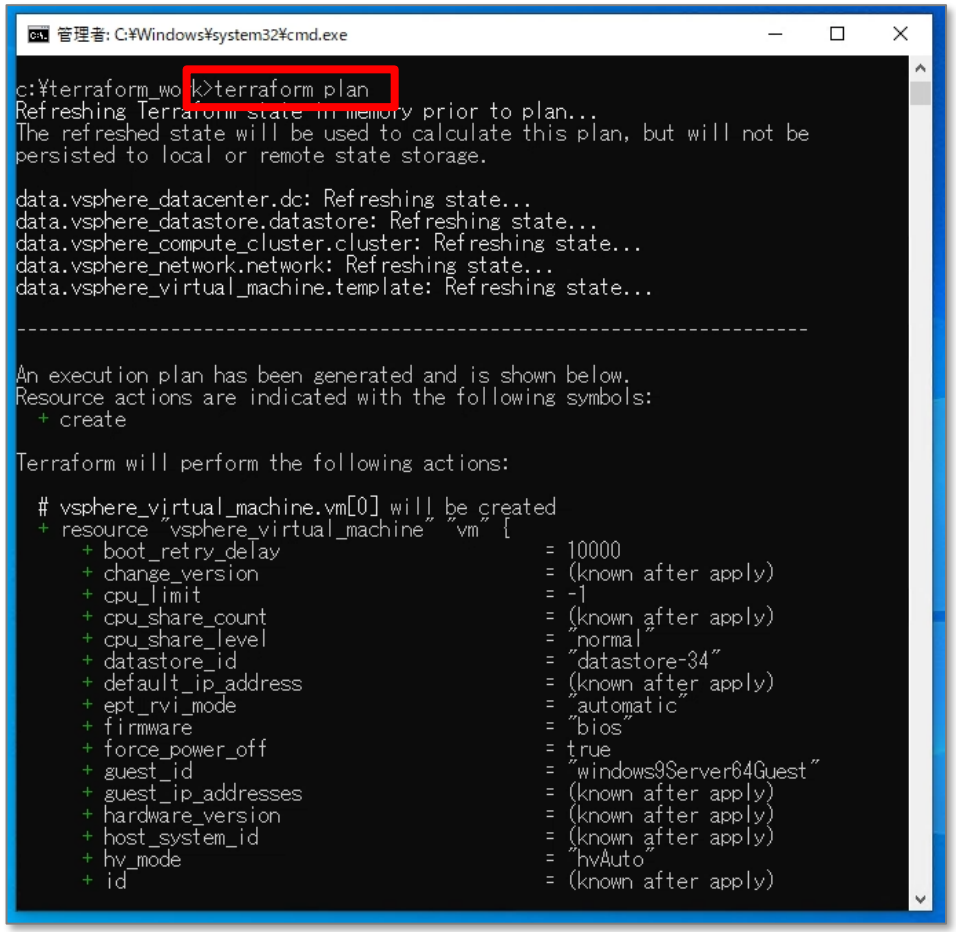


Terraform のプロバイダーとは？

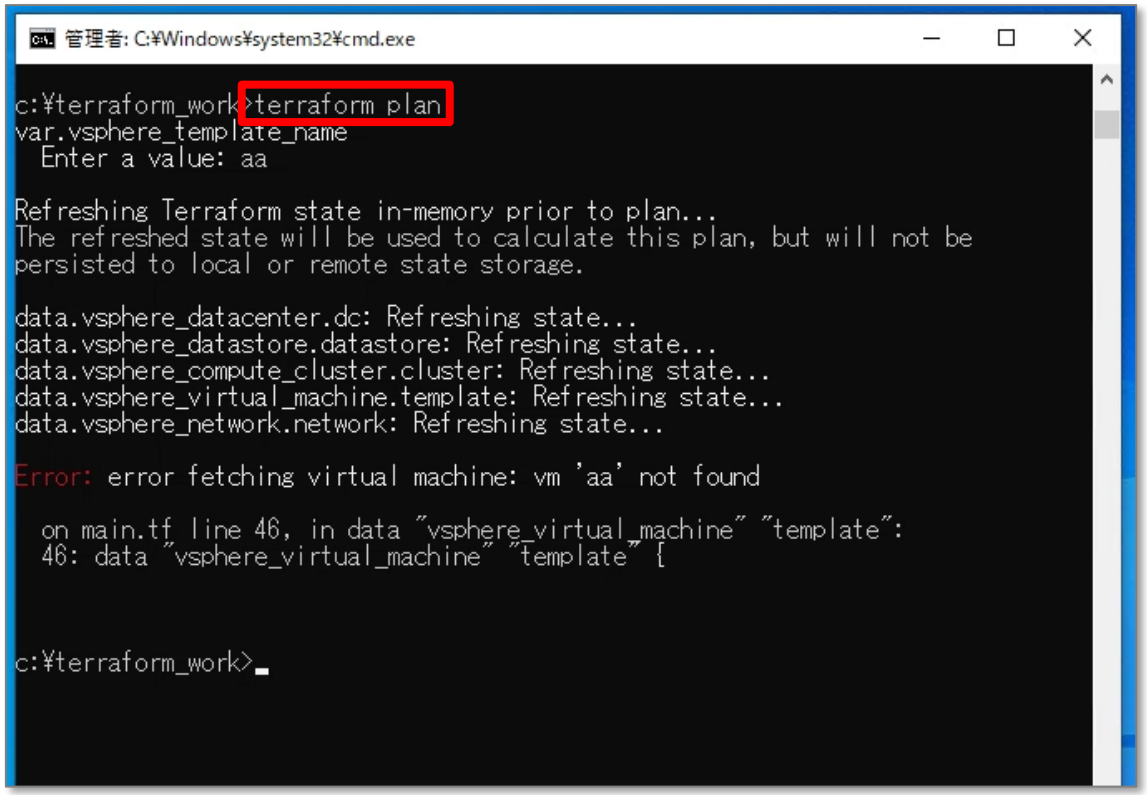


[STEP3] Terraform PLAN

Terraform PLAN でコードの検証を実施する

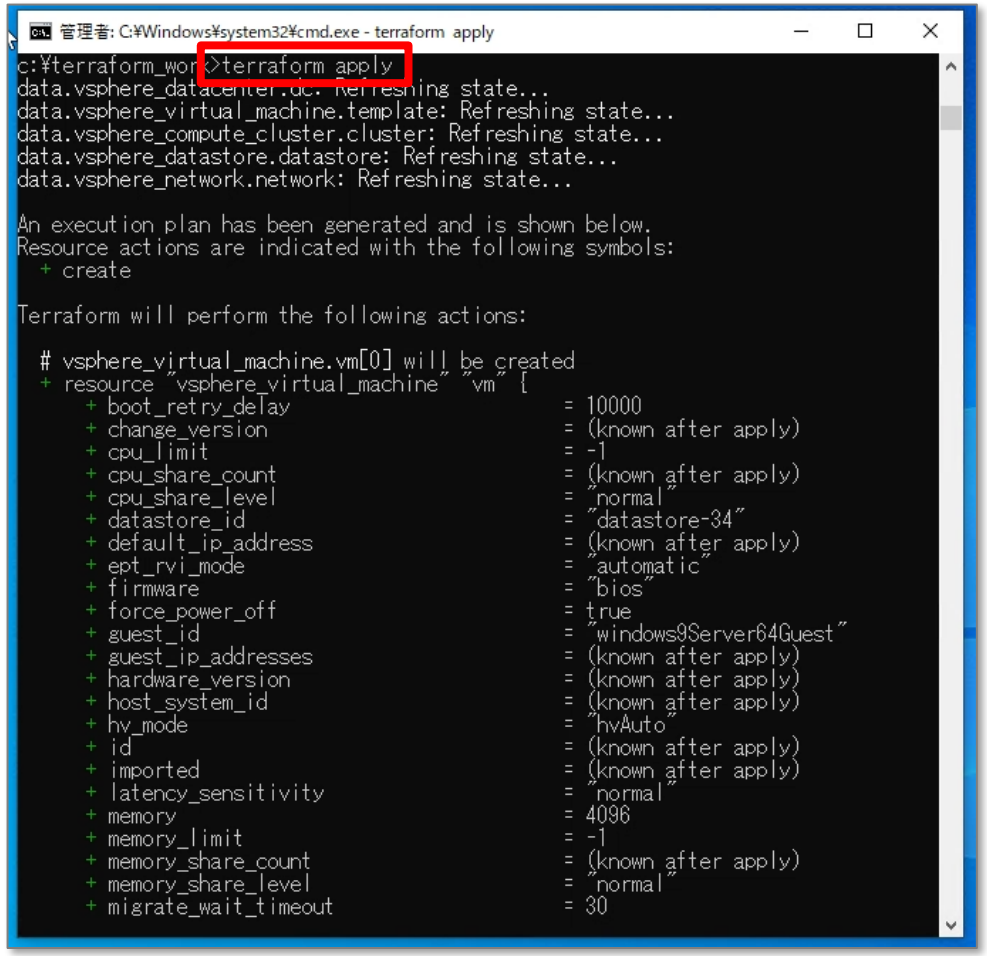


コードに不備があるとエラーが表示される

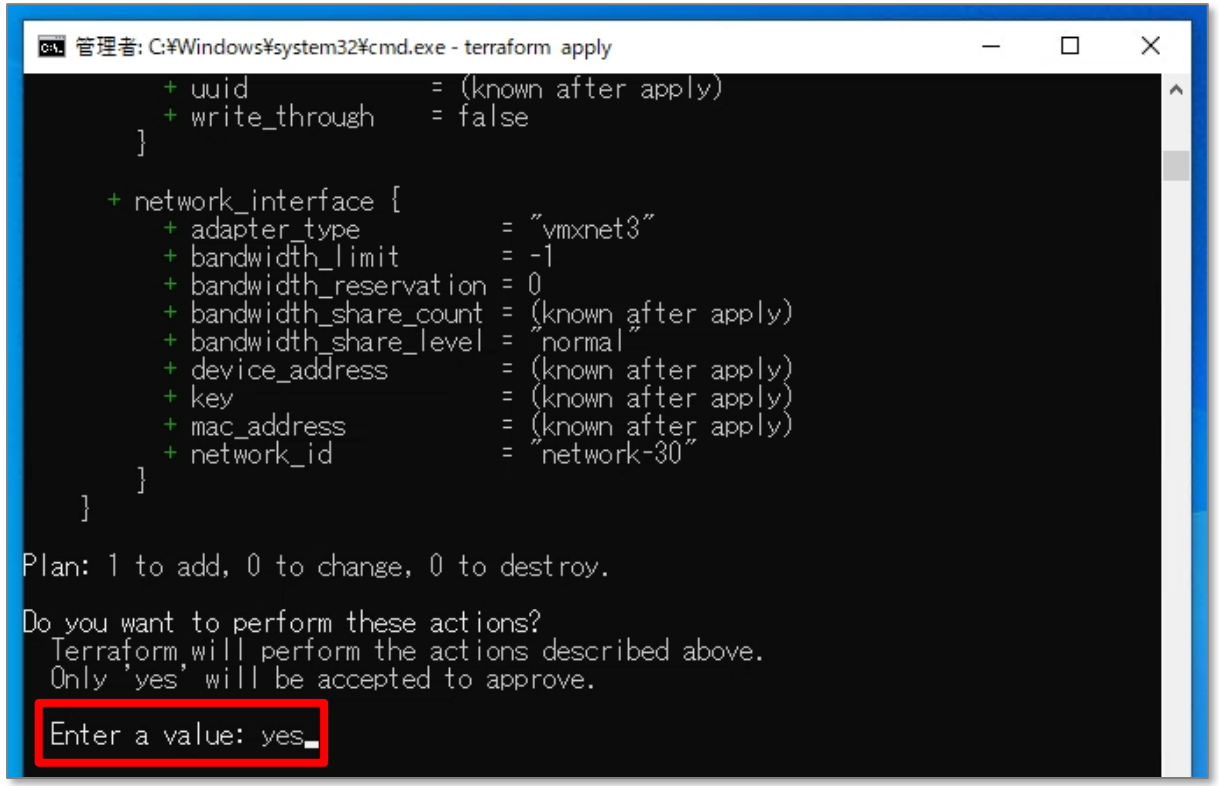


[STEP4] Terraform APPLY

Terraform APPLY でコードをインフラに適用する



プロビジョニングされるリソースを確認し、
問題がなければ “YES” で適用を開始する



Terraform のデモ (動画)

デモのお品書き

Terraform (OSS版) をダウンロードして、Windows 10 に初期設定する

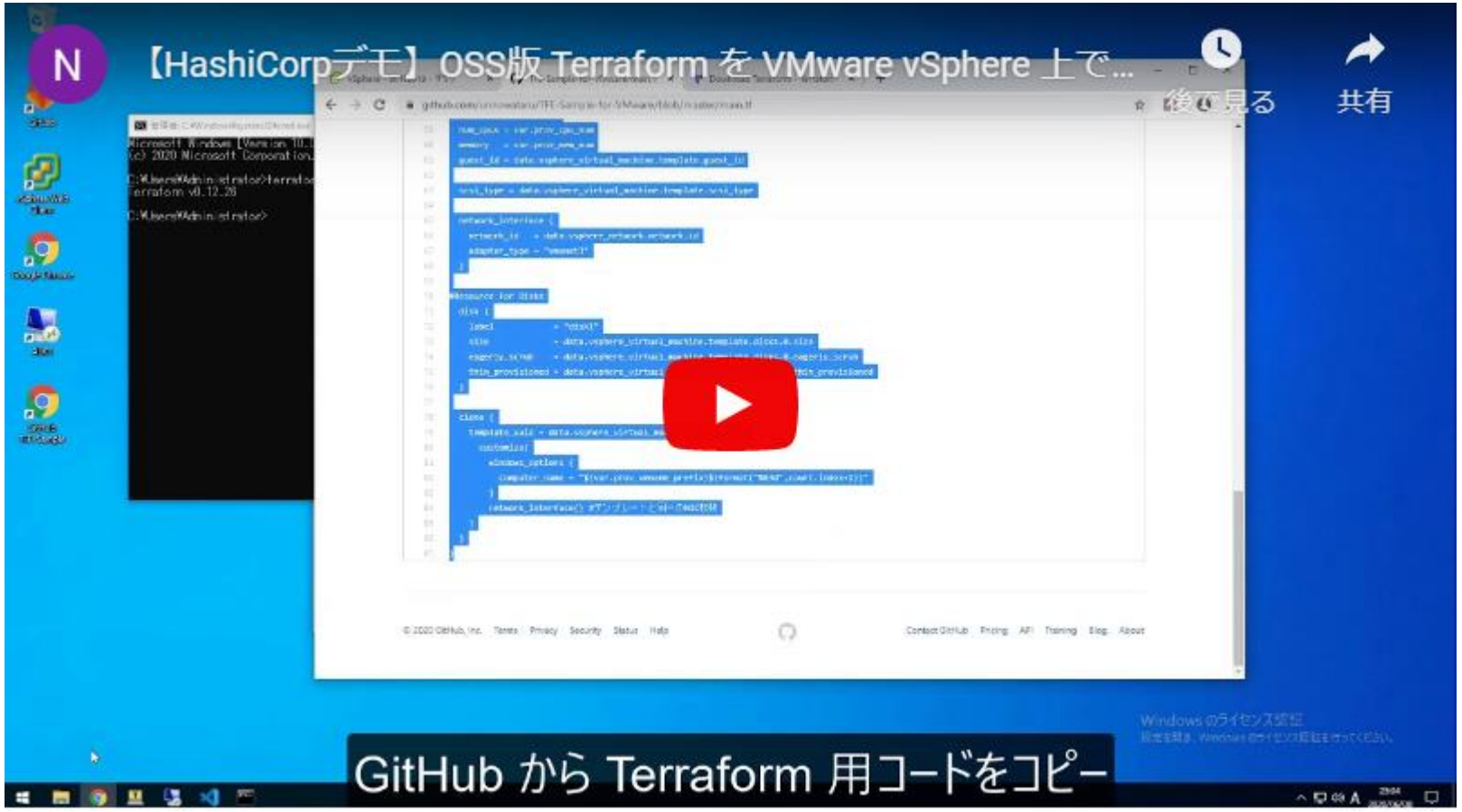
GitHub からコードをコピペしてくる

vSphere 6.7u1 の環境で テンプレートを使って Windows Server を 1台 プロビジョニングする

(仮想マシンの名前は TFVM001 とする)

デモ：OSS版 Terraform と vSphere の組み合わせ

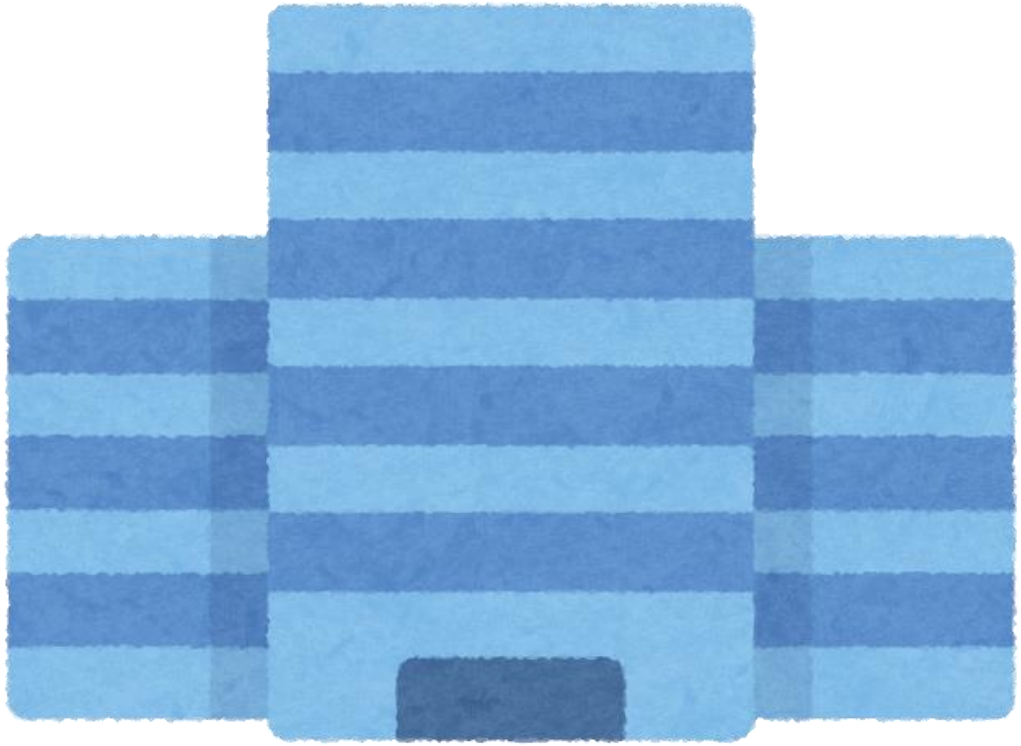
- https://www.youtube.com/watch?v=8EhO1ocX2zw



Terraform Enterprise とは？

Terraform Enterprise とは？

有償版の Terraform のひとつです!!

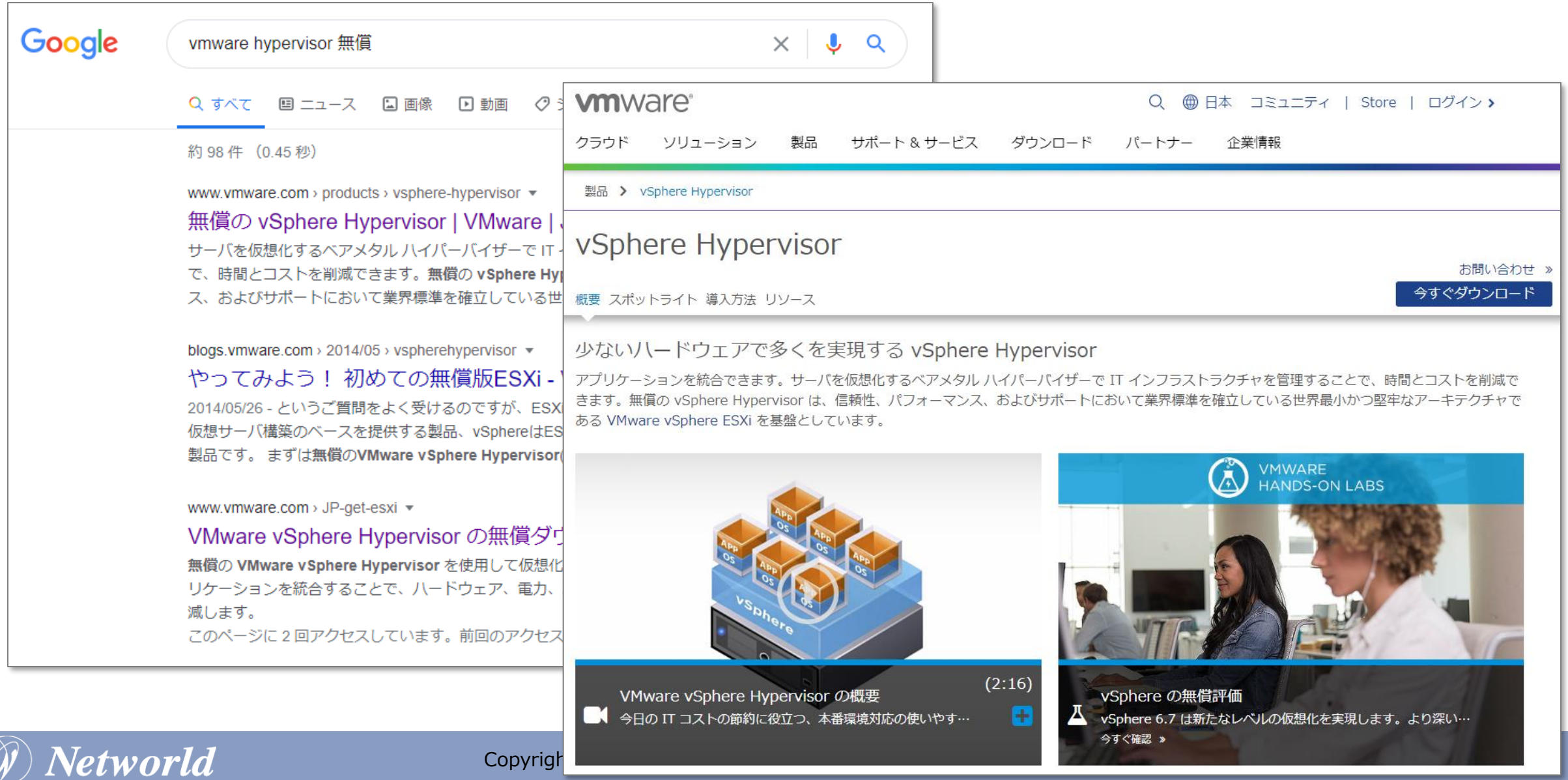


こんなお客様へ!!

Terraform を導入したいけど…

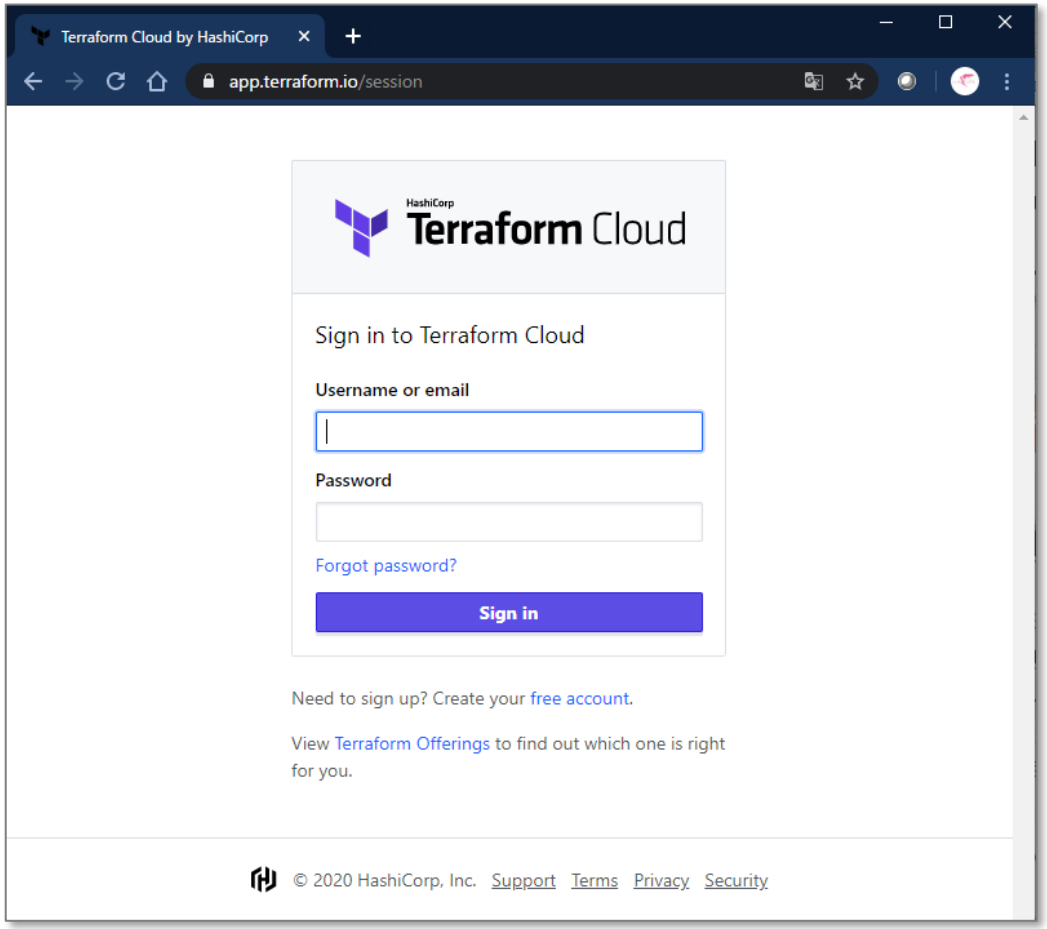
- サポートがほしい
- チームで取り組みたい
  (属人性をできる限りなくしたい)
- 管理・監査が必要
  (ヒヤリハットを防ぎたい)
- オンプレミスでも使いたい

VMware も無償版があるけど、一般的には本番では使われない



その前に…

Terraform Cloud ってのもある



その前に…

Terraform Cloud ってのもある

Terraform Cloud Get started with our Cloud Infrastructure Automation as a Service	**Free** Collaborate on infrastructure as code for Terraform configurations.	**Team & Governance** Collaborate on infrastructure as code, manage users, and enforce provisioning policies.	**Business** Everything organizations need to use Terraform at scale.
Cloud Pricing	$0 up to 5 users	Starting at $20 user/month	Contact Sales

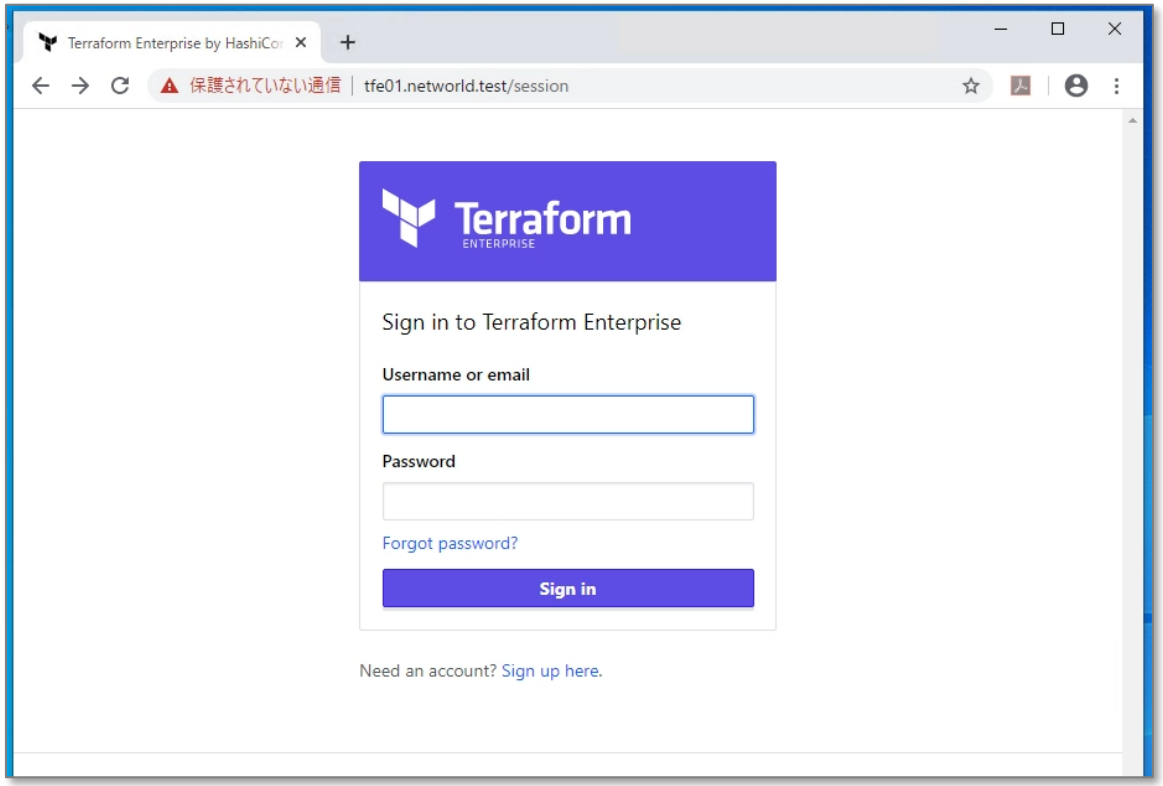
https://www.hashicorp.com/products/terraform/pricing/

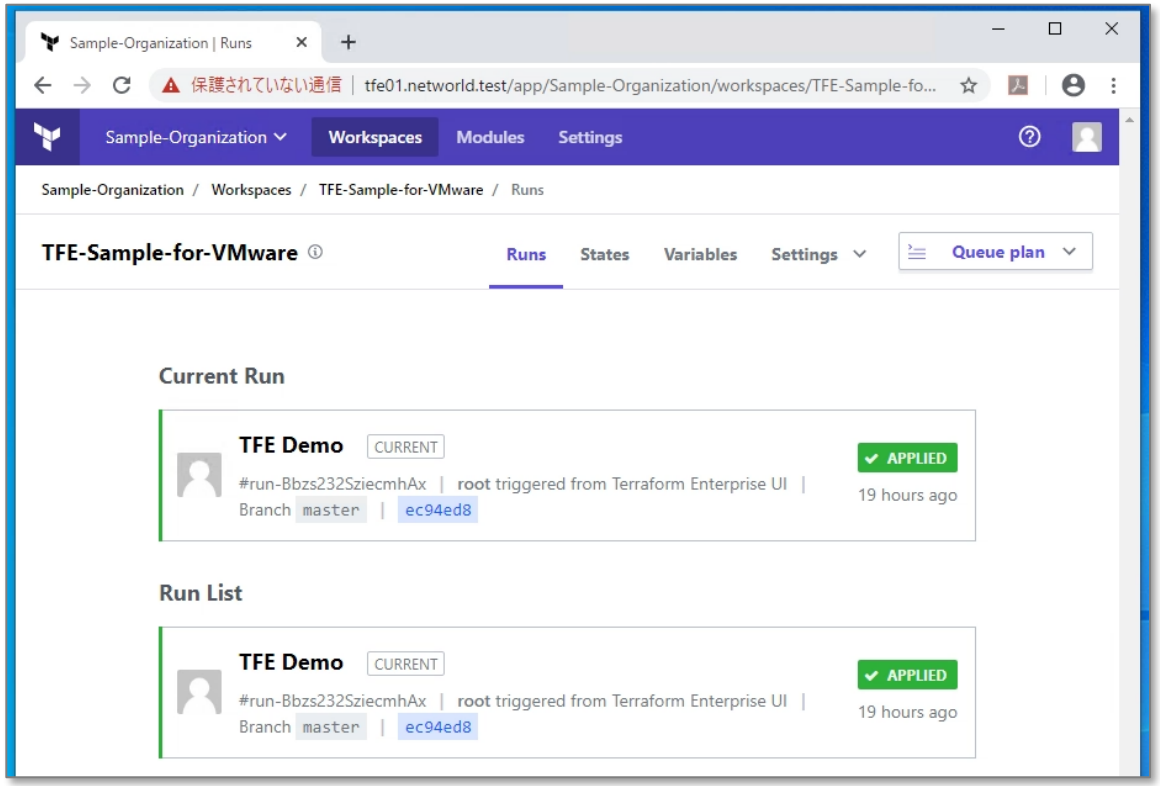
【これが難しい!!】無償・有償 と オンプレミス・クラウド

	オンプレミス	クラウド
無償	**Terraform (OSS)** • CUI のみ • ローカルの端末で実行 • オンプレでもパブリックでもOK	**Terraform Cloud (Free)** • GUI あり • HashiCorp による SaaS • オンプレミス利用NG
有償	**Terraform Enterprise** • GUI あり • **お客様自身によるホスティング** • オンプレでもパブリックでもOK • チーム管理&ポリシー機能あり • エンタープライズ向け機能あり (SAMLや監査ログなど)	**Terraform Cloud Business** • GUI あり • **HashiCorp による SaaS** • オンプレでもパブリックでもOK • チーム管理&ポリシー機能あり • エンタープライズ向け機能あり (SAMLや監査ログなど)

この資料はコレについての説明です!!

Terraform Enterprise (TFE) の GUI





OSS と TFE の機能の違い (弊社製品紹介ページより)

カテゴリ	機能	OSS	Enterprise
Infrastructure as Code	HCL による Infrastructure as Code (HashiCorp Configuration Language)	✓	✓
	ワークスペース	✓	✓
	変数の設定	✓	✓
	実行 (PlanとApply)	✓	✓
	依存関係の可視化	✓	✓
	プロバイダー	✓	✓
	モジュール	✓	✓
	Public Module Registry	✓	✓

OSS と TFE の機能の違い (弊社製品紹介ページより)

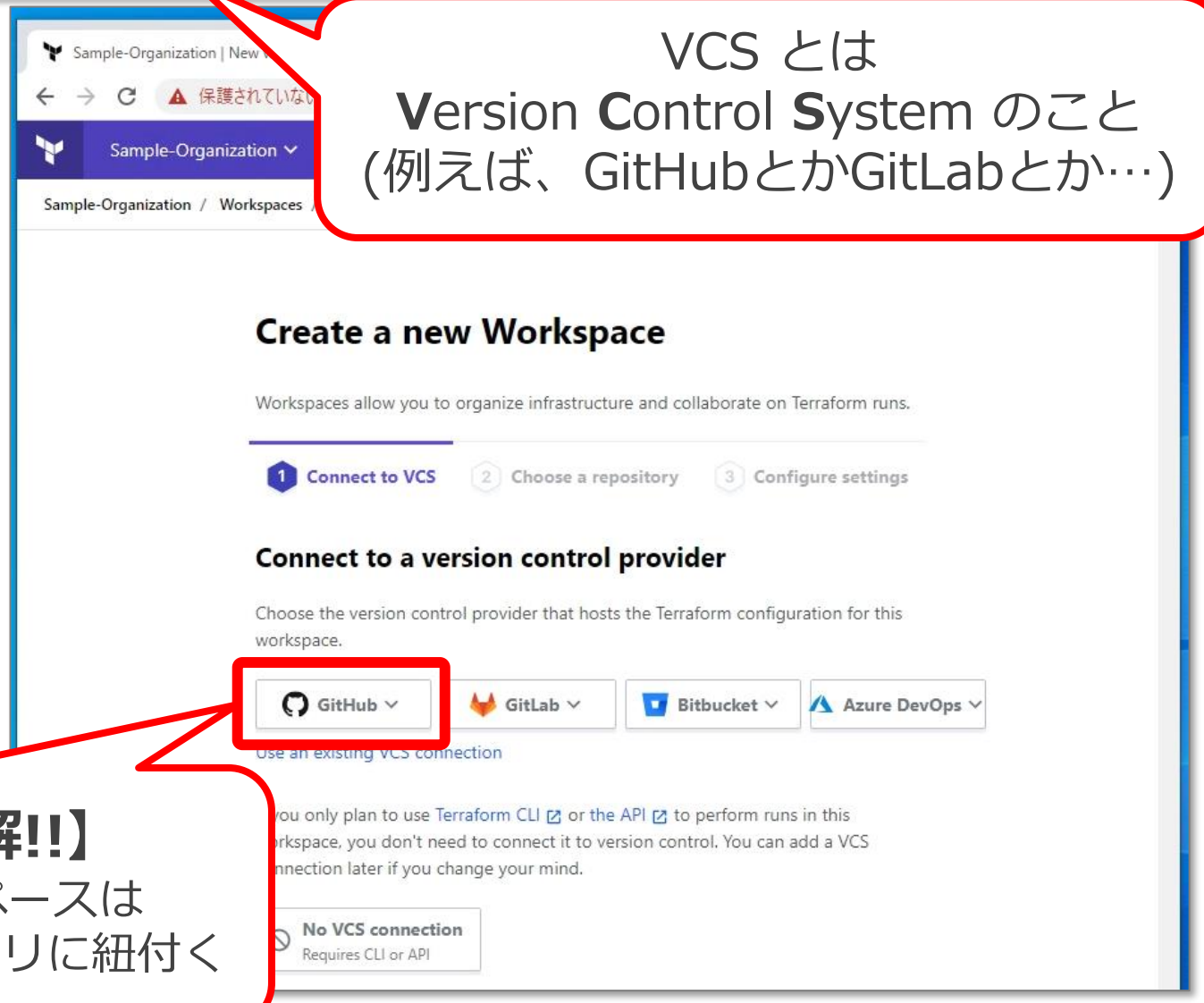
カテゴリ	機能	OSS	Enterprise
コラボレーション	インフラ状態をリモートに保存	—	✓
	VCS連携	—	✓
	ワークスペース管理	—	✓
	セキュアな変数管理	—	✓
	リモート実行	—	✓
	Private Module Registry	—	✓
	Full API Coverage	—	✓
	チーム管理	—	✓
	SAMLによるSSO	—	✓

OSS と TFE の機能の違い (弊社製品紹介ページより)

カテゴリ	機能	OSS	Enterprise
ガバナンスとポリシー	Sentinel によるポリシー管理	ー	✓
	費用予測	ー	✓
	監査ログ	ー	✓
インフラのセルフサービス化	Configuration Designer	ー	✓
	ServiceNow	ー	✓
運用	高可用性クラスタアーキテクチャ	ー	✓
	パフォーマンススケーリング	ー	✓
	オンプレミスネットワーク接続	ー	✓
	オンプレミス構築	ー	✓
サポート	英語(メーカー)サポート	ー	✓
	日本語サポート ※オプション	ー	✓

TFE注目機能：VCS と連携したワークスペース

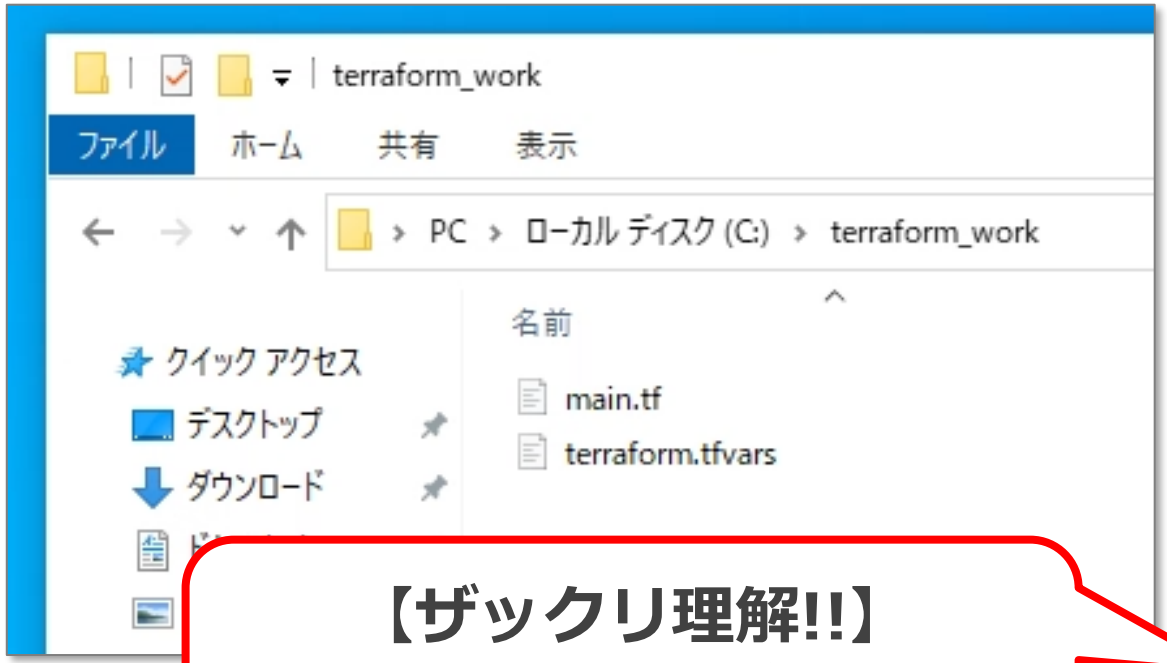
VCS とは
Version **C**ontrol **S**ystem のこと
(例えば、GitHubとかGitLabとか…)



【ザックリ理解!!】
ひとつのワークスペースは
ひとつの VCS リポジトリに紐付く

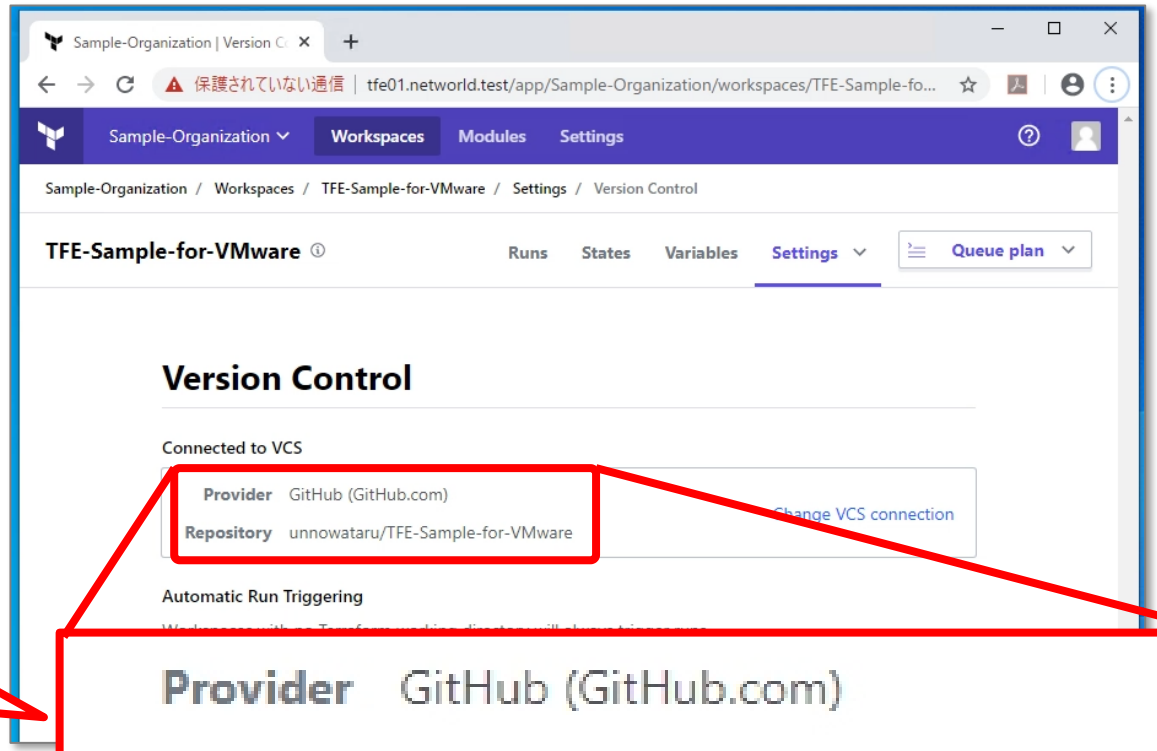
TFE注目機能：VCS と連携したワークスペース

ローカルにコードを配置する代わりに...

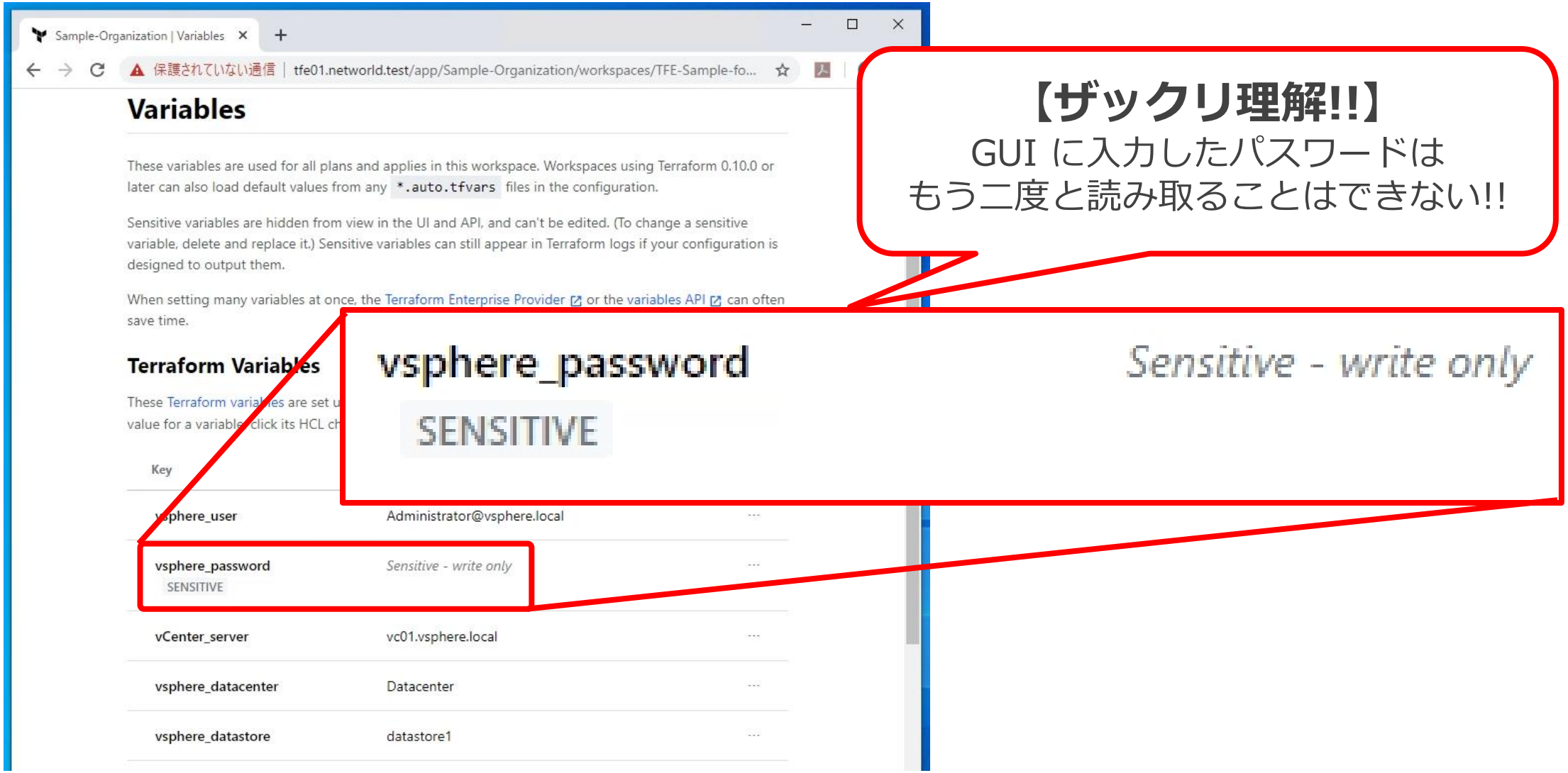


【ザックリ理解!!】
ここからコードを取得して、
Terraform のサーバーに格納する

ワークスペースに VCS のリポジトリを紐付ける

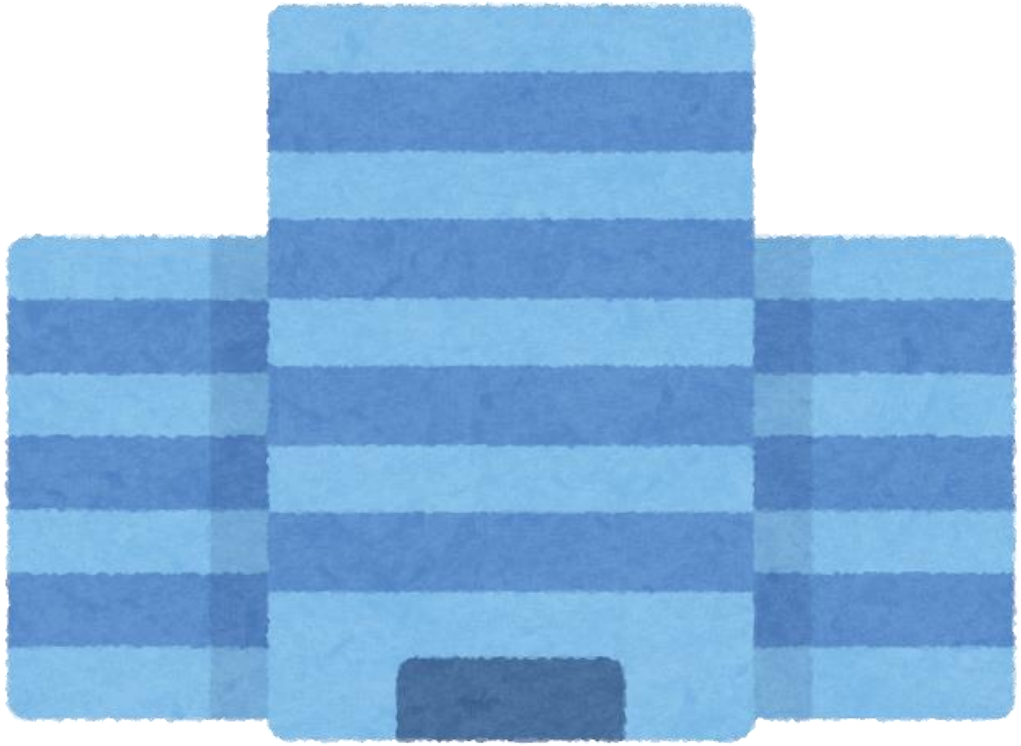


TFE注目機能：Secure Variables



TFE注目機能：Sentinel とは？

HashiCorp 製品のひとつで Terraform に統合されています



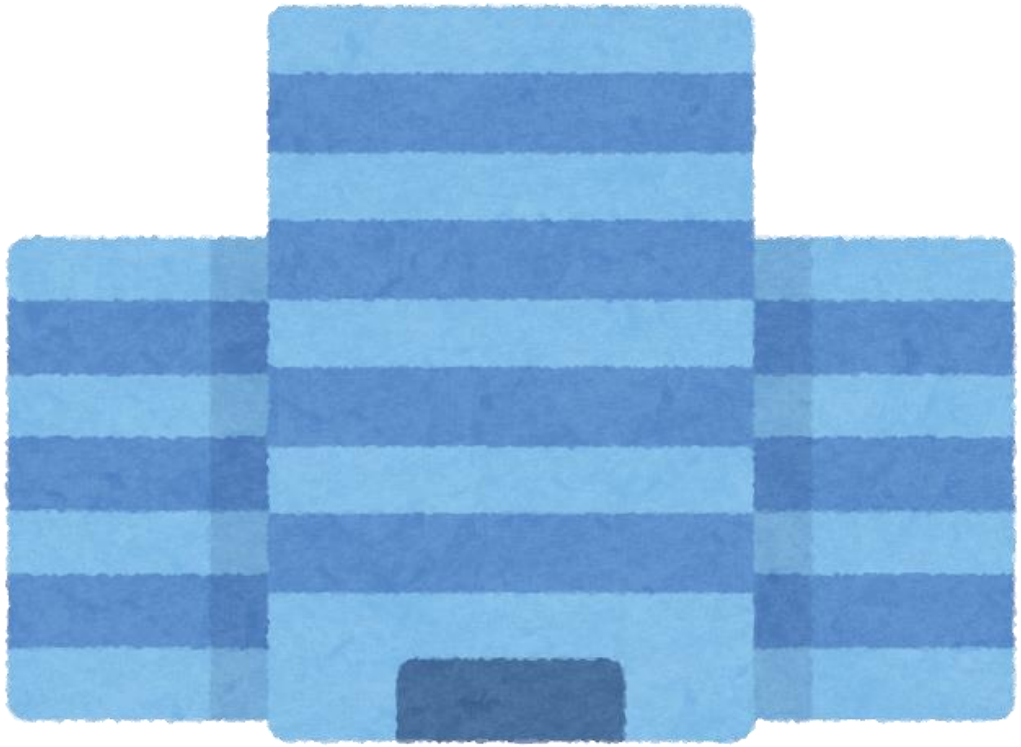
こんなお客様へ!!

Terraform を導入したいけど…

- **サポートがほしい**
- **チームで取り組みたい**
  (属人性をできる限りなくしたい)
- **管理・監査が必要**
  (ヒヤリハットを防ぎたい)

TFE注目機能：Sentinel とは？

HashiCorp 製品のひとつで Terraform に統合されています



こんなお客様へ!!

Terraform を導入したいけど…

- **サポ**
- **チー**
  (属
- **管理・監査が必要**
  (ヒヤリハットを防ぎたい)

【ザックリ理解!!】
特にこの部分に有効で、
いろいろできちゃう Terraform で
事故が起きないよう**ポリシー設定**できる

TFE注目機能：Sentinel

Policy as Code： ポリシーをコード化し、プロビジョニングするときに自動的にルールが適用されているか確認。

Version Control： ポリシー作成やバージョン管理、コードのレビューなどを既存の VCS ベースの手法で実現。

Policy Sets： 複数のポリシーをグループ化し、複数の異なる環境に対し、独自のルールの設定が可能。

Testing： Sentinel Simulator により、カンタンにポリシーのテストや開発が可能。

Sentinel にはどんなポリシーがあるのか？

VMware 向けのサンプルだと…

VMware

- Require Storage DRS to be enabled
- Restrict virtual disk size
- Restrict VM CPU count and memory
- Enforce NFS 4.1 and Kerberos

Azure 向けのサンプルだと…

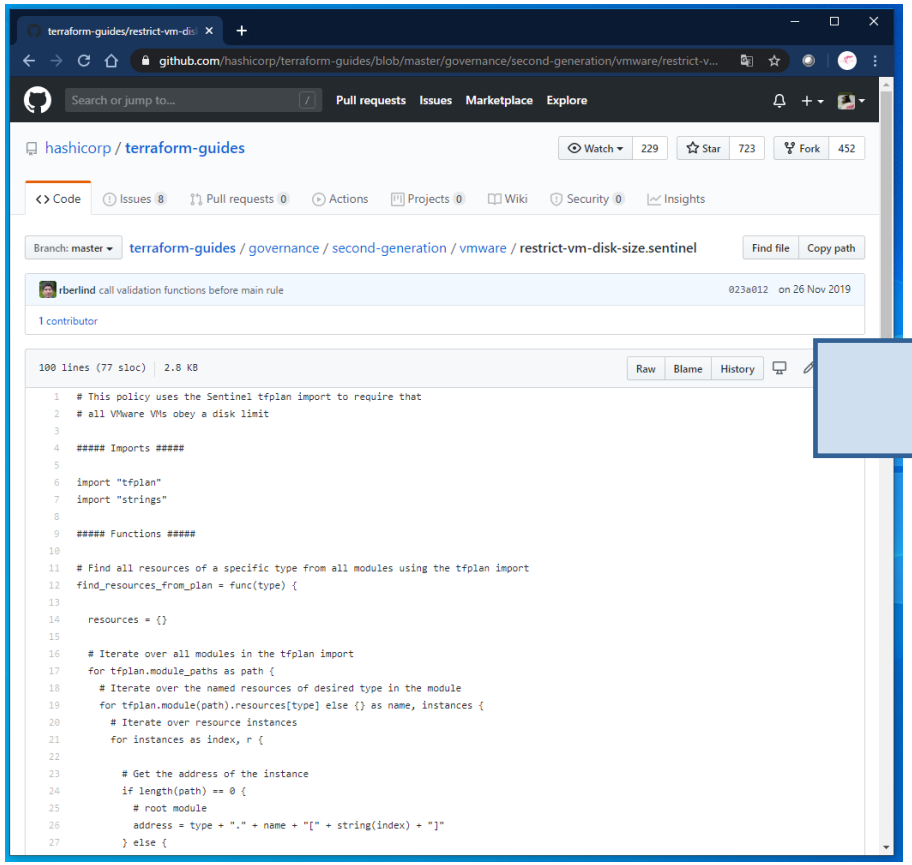
Microsoft Azure

- Restrict VM images
- Restrict the size of Azure VMs
- Enforce limits on AKS clusters

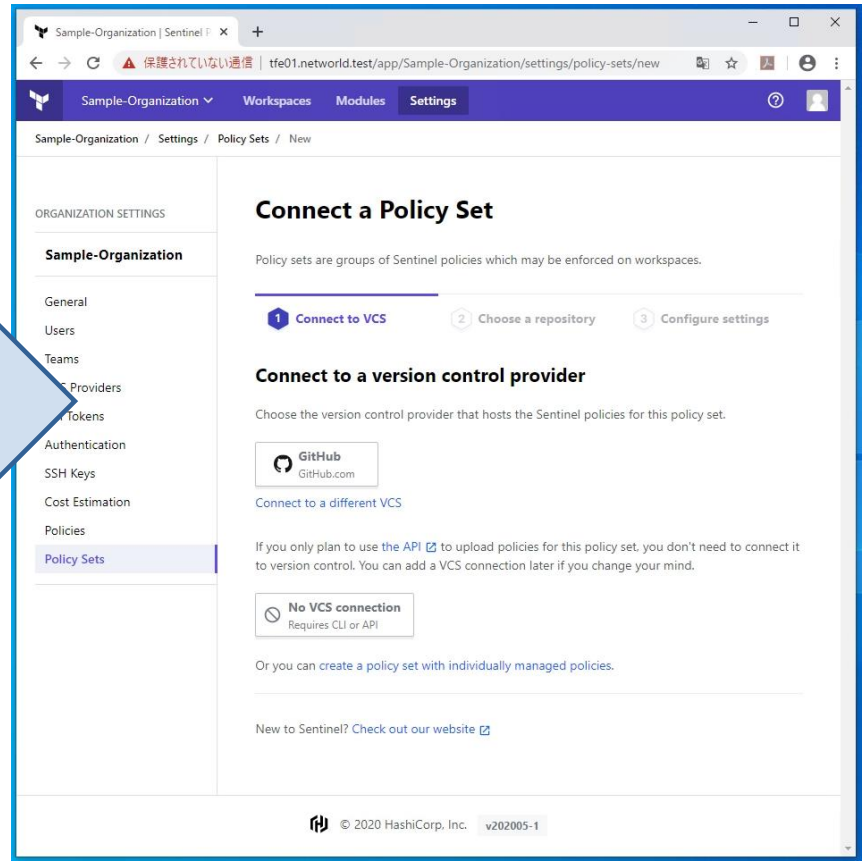
https://www.terraform.io/docs/cloud/sentinel/examples.html

Sentinel はどうやって設定するのか？

VCS にコードを作成して…



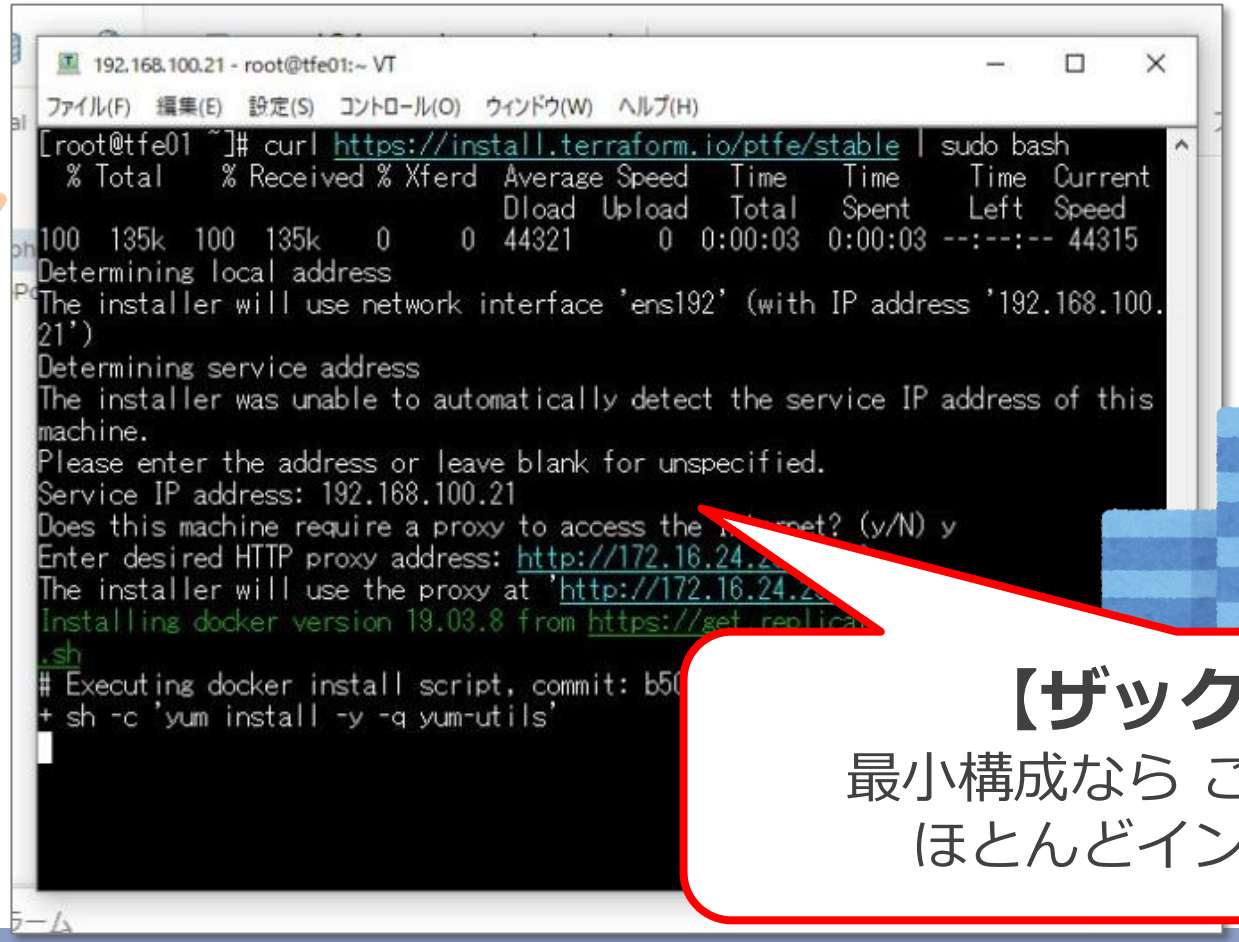
リポジトリからポリシーを設定



https://github.com/hashicorp/terraform-guides/blob/master/governance/second-generation/vmware/restrict-vm-disk-size.sentinel

TFE注目機能：オンプレミスでの構築・利用

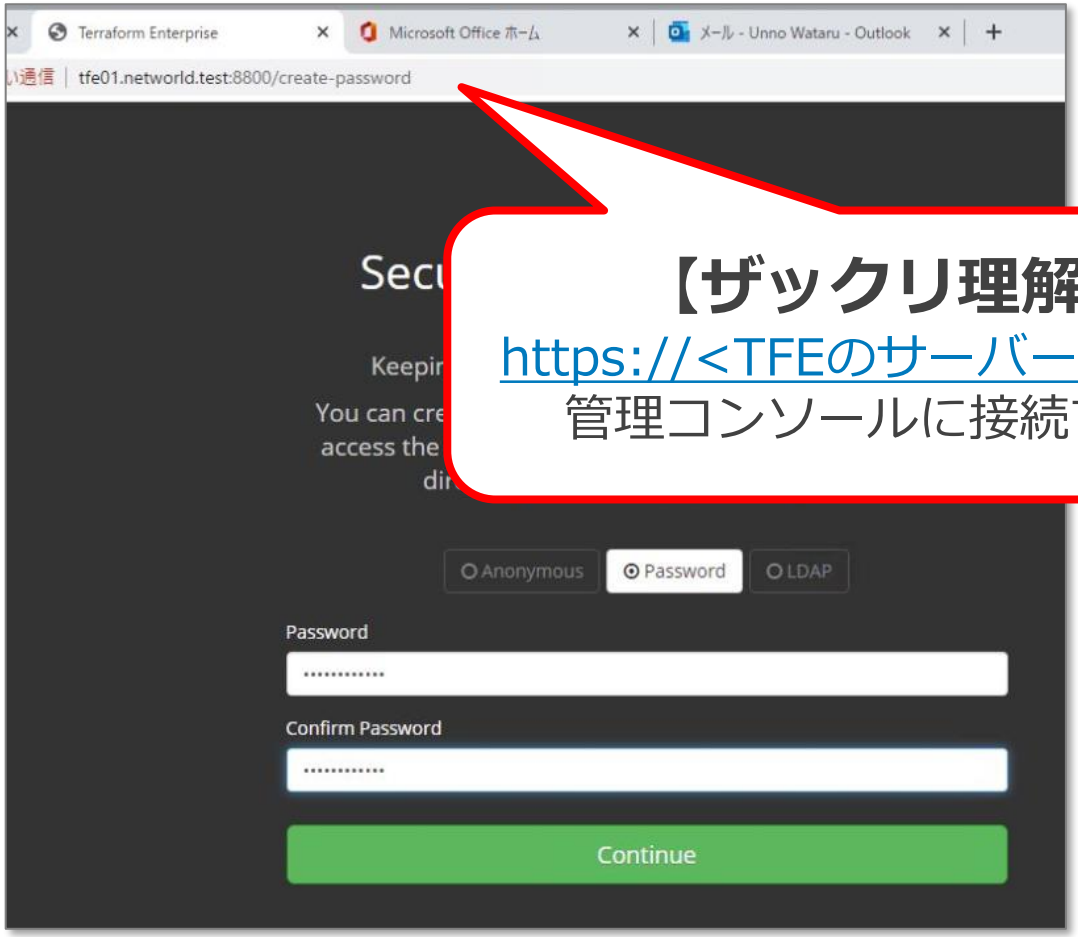
curl https://install.terraform.io/pfte/stable | sudo bash



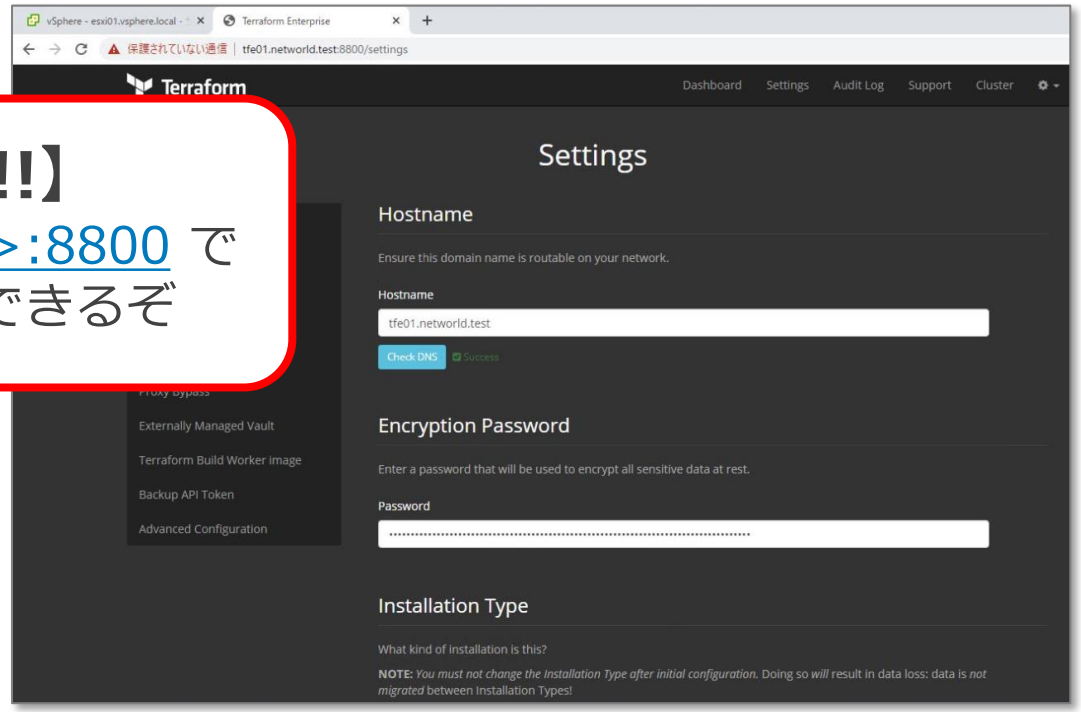
【ザックリ理解!!】
最小構成なら このコマンド一発で
ほとんどインストールできる

TFE注目機能：オンプレミスでの構築・利用

初期セットアップ と 管理コンソールはこんな感じ



【ザックリ理解!!】
https://<TFEのサーバー>:8800 で
管理コンソールに接続できるぞ



TFE注目機能：オンプレミスでの構築・利用

“docker ps” で Terraform Enterprise のプロセスを確認

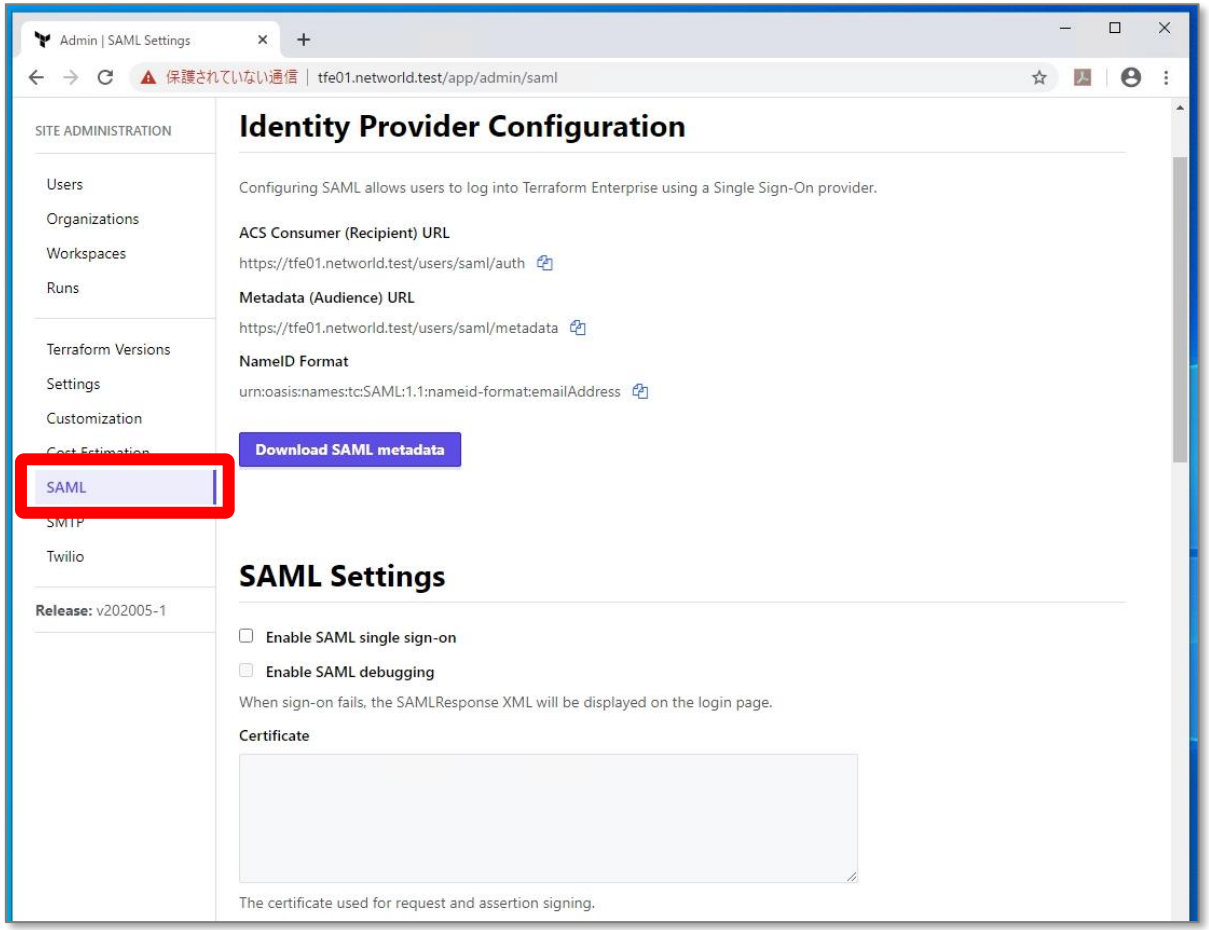
192.168.100.21 - root@tfe01:~ VT

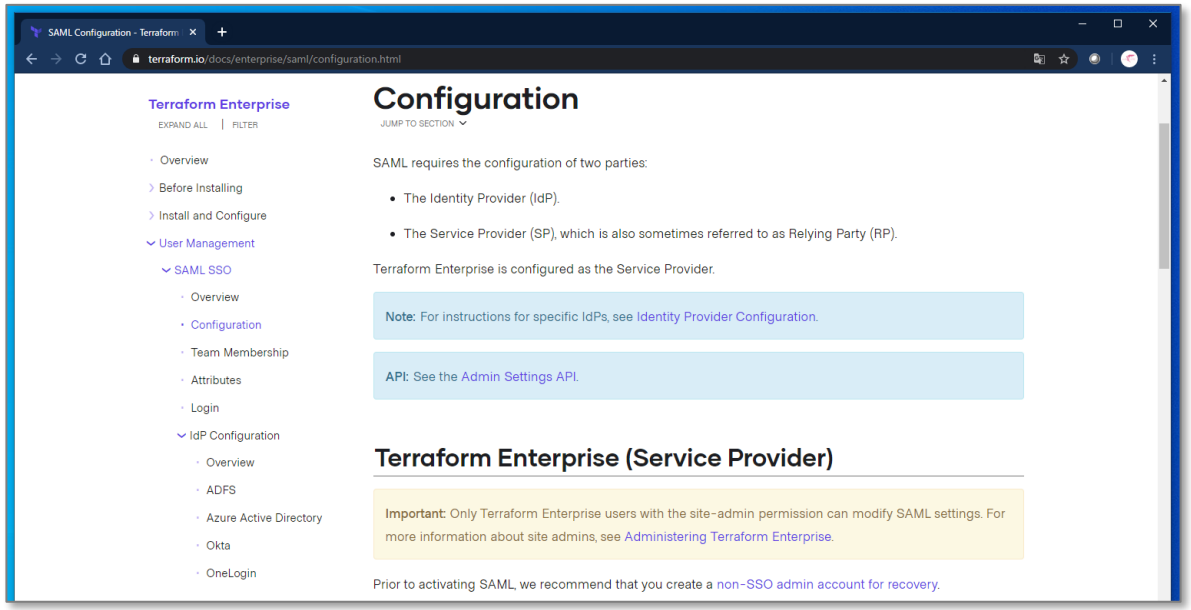
ファイル(F)　編集(E)　設定(S)　コントロール(O)　ウィンドウ(W)　ヘルプ(H)

```
[root@tfe01 ~]# docker ps
CONTAINER ID        IMAGE                                                          COMMAND                  CREATED             STATUS              PORTS                                                                                  NAMES
d410a913addf        192.168.100.21:9874/hashicorp-atlas:CIRC-74535-514208127d      "/usr/bin/init.sh ba…"   22 hours ago        Up 22 hours                                                                                ptfe_sidekiq
c8148c7636ca        192.168.100.21:9874/hashicorp-tf-registry:CIRC-3974-de1be025   "setup-ssl /usr/bin/…"   22 hours ago        Up 22 hours                                                                                ptfe_registry_api
e48d910459c0        192.168.100.21:9874/hashicorp-atlas:CIRC-74535-514208127d      "/usr/bin/init.sh ba…"   22 hours ago        Up 22 hours         0.0.0.0:9292->9292/tcp                                                 ptfe_atlas
7f8c9e41e8d6        192.168.100.21:9874/hashicorp-tf-registry:CIRC-3974-de1be025   "setup-ssl /usr/bin/…"   22 hours ago        Up 22 hours                                                                                ptfe_registry_worker
f6de88ad3a2d        192.168.100.21:9874/hashicorp-terraform-build-worker:CIRC-500-7d346cb   "start-tbw /terrafor…"   22 hours ago        Up 22 hours                                                                            goofy_shamir
a2ae1c34ca0c        192.168.100.21:9874/hashicorp-archivist:CIRC-128-9398793       "/usr/bin/wait-for-t…"   22 hours ago        Up 22 hours         0.0.0.0:7675->7675/tcp                                                 ptfe_archivist
7bc4ca81678e        192.168.100.21:9874/hashicorp-terraform-build-manager:CIRC-50-431cd32   "/usr/bin/tbm-start"     22 hours ago        Up 22 hours                                                                            ptfe_build_manager
0238974b9f89        192.168.100.21:9874/hashicorp-ptfe-vault:CIRC-53-b485c55       "vault-start"            22 hours ago        Up 22 hours         0.0.0.0:8200->8200/tcp                                                 ptfe_vault
aa740c0c826c        192.168.100.21:9874/hashicorp-ptfe-postgres:86ba725            "docker-entrypoint.s…"   22 hours ago        Up 22 hours         0.0.0.0:5432->5432/tcp                                                 ptfe_postgres
5ca156539474        192.168.100.21:9874/hashicorp-ptfe-rabbitmq:df253ff            "/start.sh rabbitmq-…"   22 hours ago        Up 22 hours         0.0.0.0:5672->5672/tcp, 0.0.0.0:32791->4369/tcp, 0.0.0.0:32790->5671/tcp, 0.0.0.0:32789->25672/tcp   rabbitmq
```

TFE注目機能：SAML 認証

Single Sign-On 連携が可能

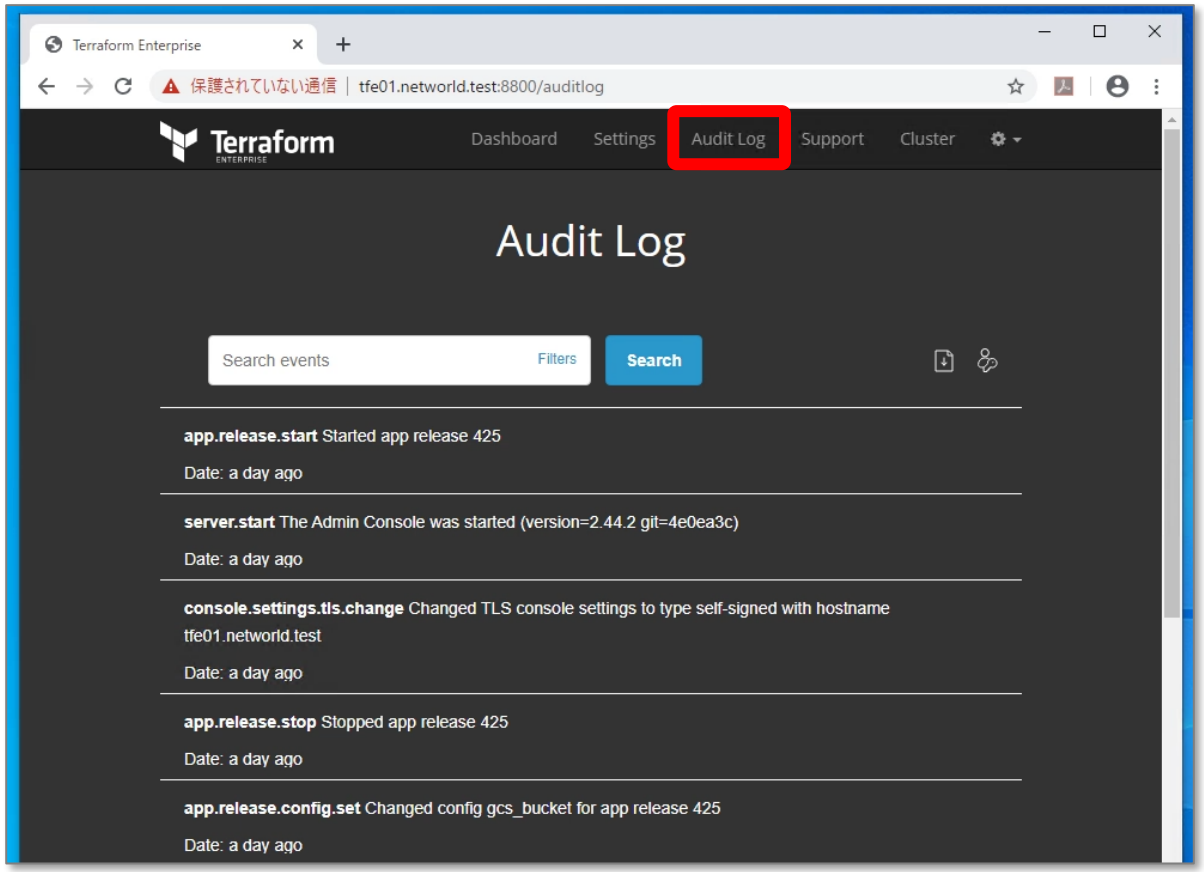


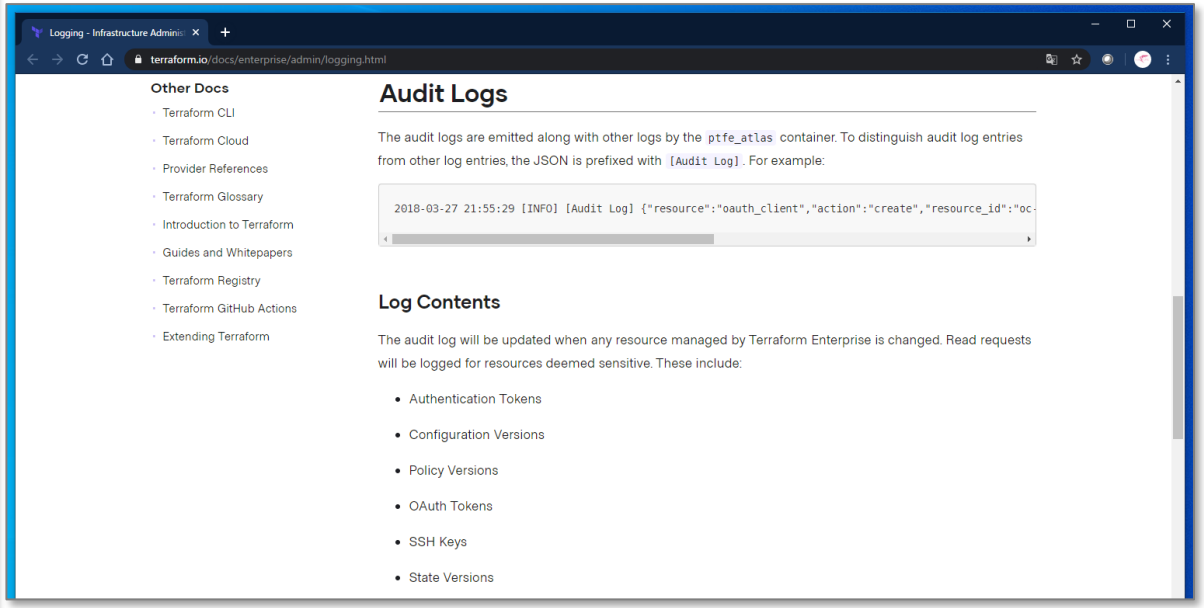


https://www.terraform.io/docs/enterprise/saml/configuration.html

TFE注目機能：監査ログ

「Terraform で いつどこで誰が何をしたのか」をロギング





https://www.terraform.io/docs/enterprise/admin/logging.html

Terraform Enterprise のデモ (動画)

おさらい：Terraform を実行するためのステップ

[STEP1] インストールと コードの作成	Terraformをインストールし、定義ファイルを作成・配置する (定義ファイルとは 拡張子 .tf のテキストデータ)
[STEP2] Terraform INIT	Terraform プロバイダーを設定する (プロバイダーとは、プロビジョニング対象製品のためのプラグイン)
[STEP3] Terraform PLAN	Terraform が定義ファイルに基づいて、各製品の API に 対応したコードを自動生成、検証(リハーサル/ドライラン)する
[STEP4] Terraform APPLY	実際にコードを適用する

では、Terraform Enterprise ではどうなるのか？

デモのお品書き

Terraform Enterprise にログインして
GitHub と連携させたワークスペースを作成する

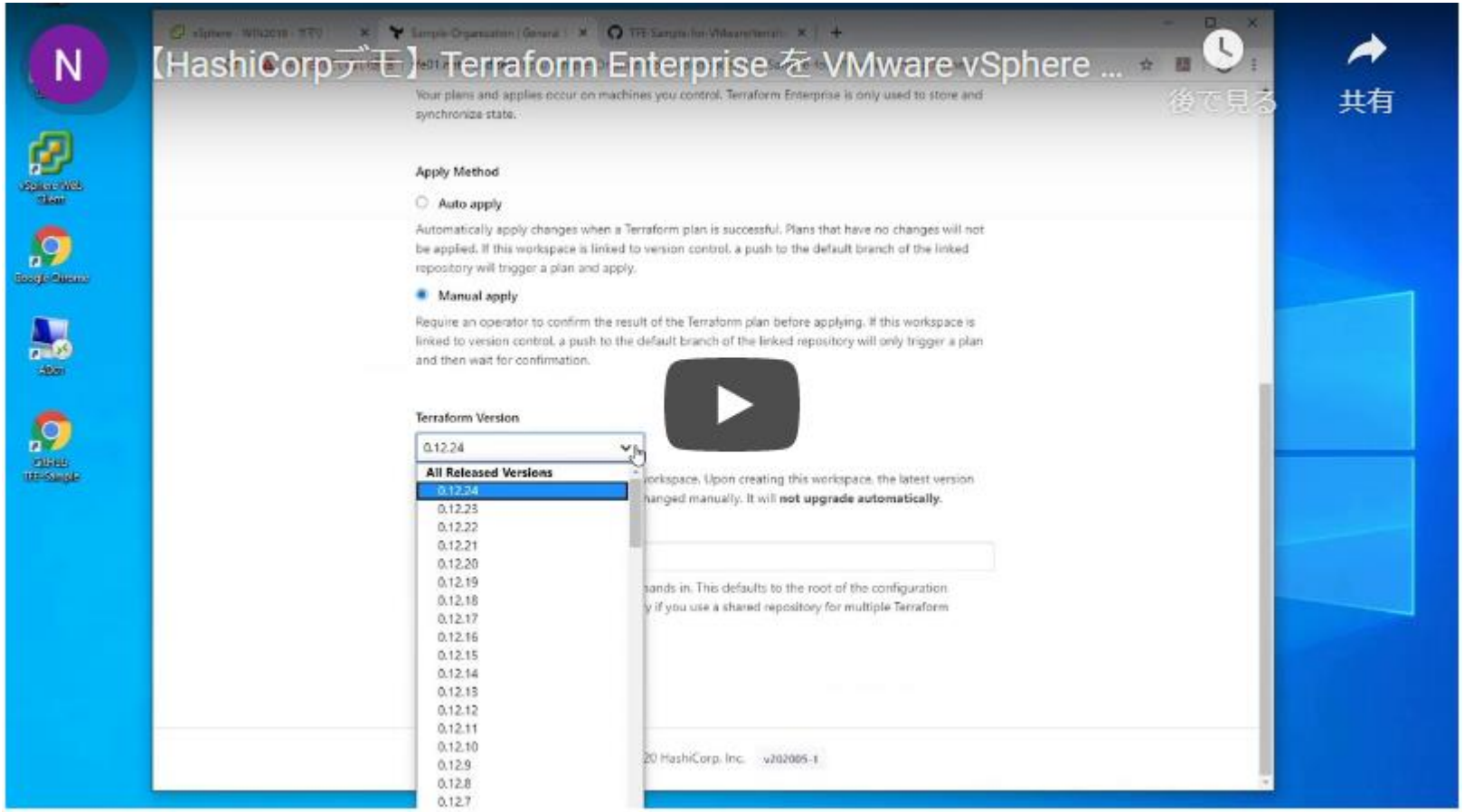
Terraform (OSS版) と完全に同じコードを使う

vSphere 6.7u1 の環境で テンプレートを使って
Windows Server を 1台 プロビジョニングする

(仮想マシンの名前は TFEVM001 とする)

デモ：Terraform Enterprise と vSphere の組み合わせ

- https://www.youtube.com/watch?v=C8k285e4TUs



3スライドで まとめ

まとめ：**I**nfrastructure **a**s **C**ode (Iac) とは

インフラ

CPUやメモリ、ディスクといったリソース、
あるいは仮想マシンやアプライアンスそのもの

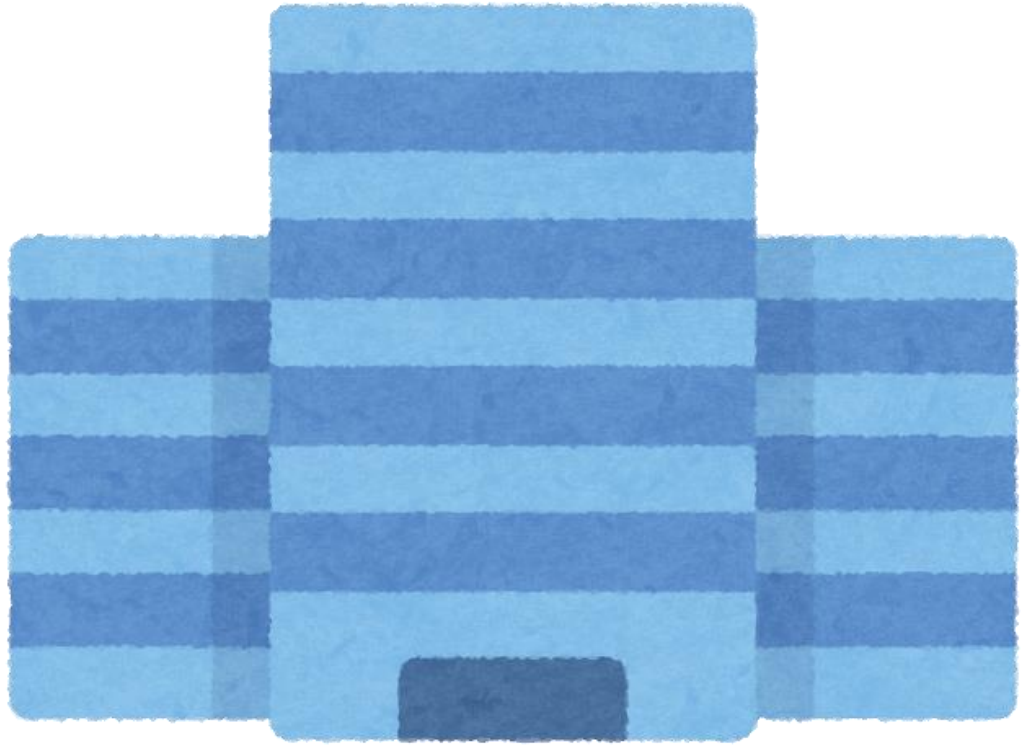
を

コードで表現すること

まとめ：Terraform Enterprise 実行ステップ

まとめ：Terraform Enterprise とは

ハイブリッドクラウドで
安心・安全 に使うなら **Terraform Enterprise**



こんなお客様へ!!

- サポートがほしい
- チームで取り組みたい
  (属人性をできる限りなくしたい)
- 管理・監査が必要
  (ヒヤリハットを防ぎたい)
- オンプレミスでも使いたい

Networld