HashiCorp

クラウドオペレーティングモデルは ネットワークの世界でどこまでできるか？

How can the cloud operating model be applied to current network operations?

Shogo Katsurada
Partner Solutions Engineer at HashiCorp
He/him

January 28, 2022

Shogo Katsurada / 桂田 祥吾

Partner Solutions Engineer at HashiCorp

2005年 4月 - Cisco Systems.
2021年 8月 - HashiCorp.

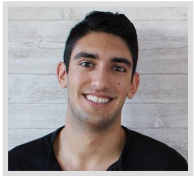
Janog との関わり

- Janog41.5
  - パブリッククラウドでの仮想ルーター利用と自動化
- Janog42
  - To disaggregate or not to disaggregate – そして良いディスアグリゲーション．悪いディスアグリゲーションとは

Twitter: @shogokatsurada

HashiCorpについて

Unlocking Cloud Operating Model


HachiCorpは、すべてのビジネスにおけるクラウドオペレーションモデルの制約を解放し、お客様のデジタル変革を成功に導きます。

HashiCorpのプロダクトを利用することにより、アプリケーションとそれを支えるインフラストラクチャの「プロビジョニング」「セキュリティ保護」「実行」「接続と連携」が可能になります。

設立	従業員	資金調達	評価額
2012	1,700+	$34	$5B +

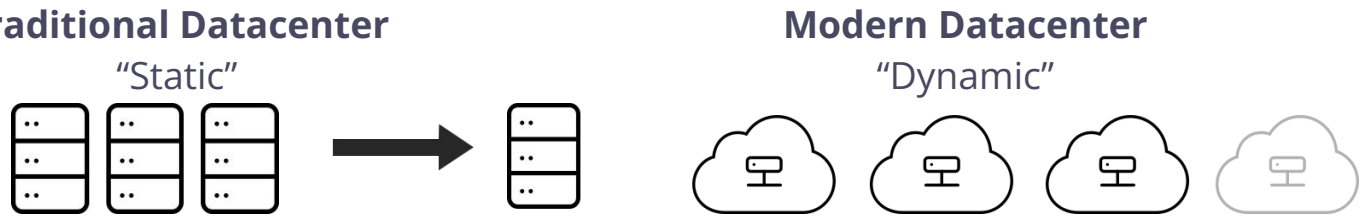
2021/12/9 IPO🎉

直近の受賞:
- Forbes Cloud 100の4位に3年連続ランクイン
- Enterprise Tech 30にて2年連続 Late Stage部門で1位

キー製品

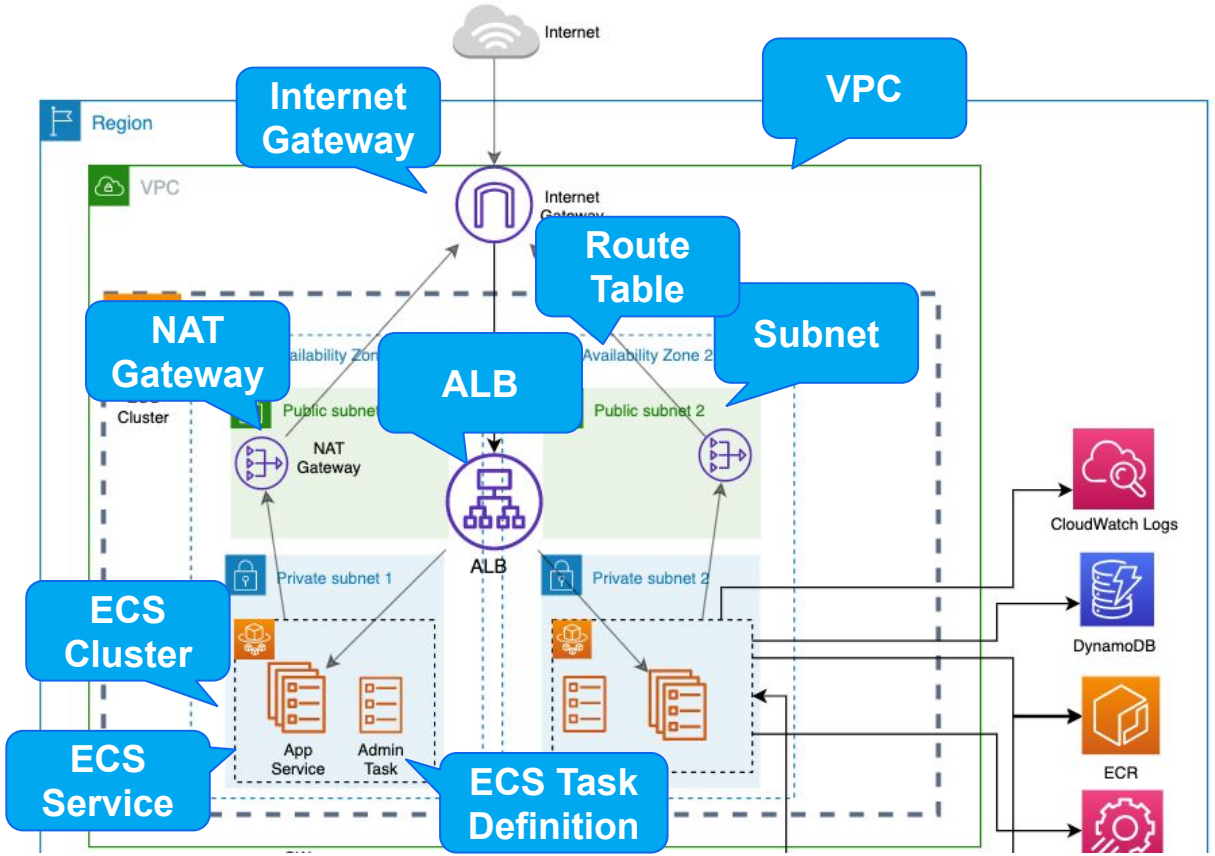
Terraform	Vault	Nomad	Consul	Vagrant	Packer	Boundary	Waypoint
Infrastructure as Code	シークレット管理と暗号化	Application ワークロード オーケストレータ	サービスベースのネットワーク通信	仮想マシン構築	サーバーイメージ作成	IDベースアクセス管理	Applicationのビルド~リリースの抽象化
Enterprise版+OSS版				OSS版			

ハイブリッドなインフラへのシフト



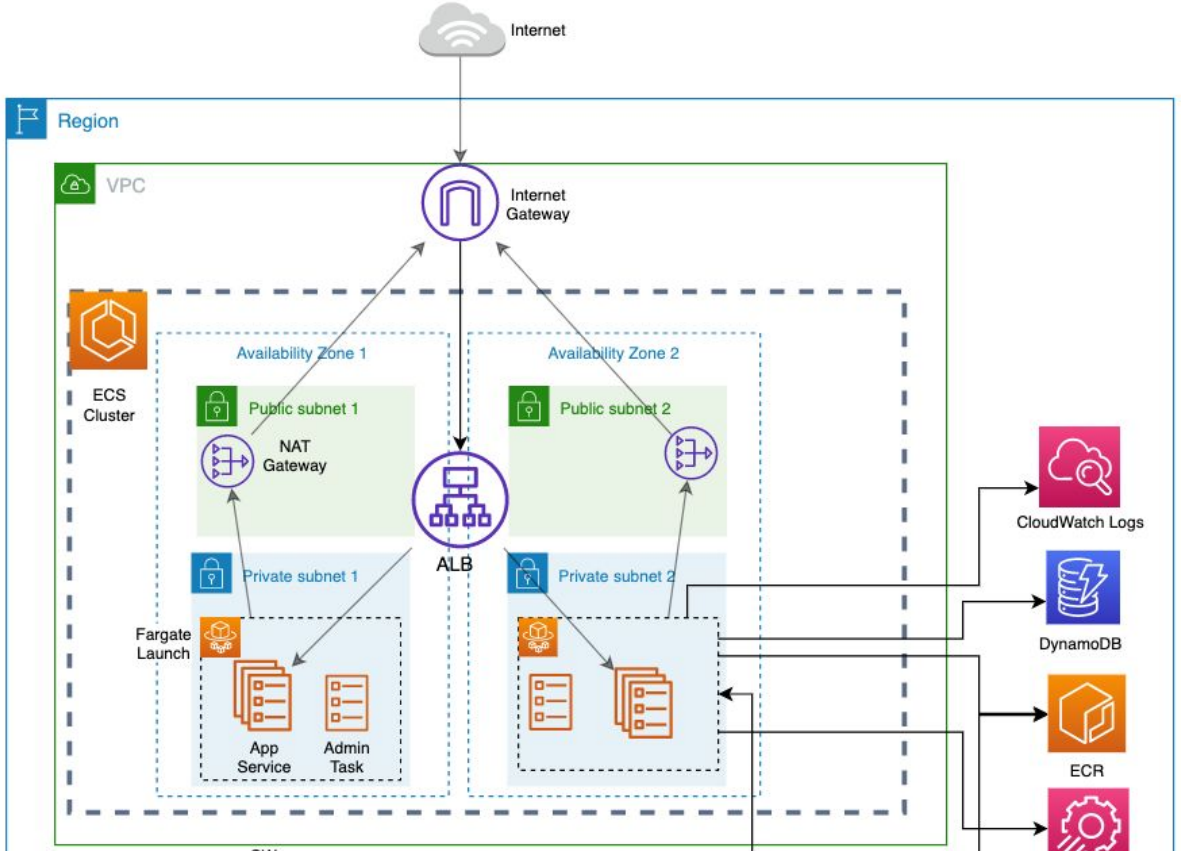
Run Development	専用プラットフォーム	→	様々なプラットフォームへの自動デプロイ
Connect Networking	ホストベース	→	サービスベース
Secure Security	高信頼性 IP-based	→	低信頼性 Identity-based
Provision Operations	専用マシン	→	オンデマンド

クラウド上で動くアプリの構成例



クラウド上で動くアプリの構成例

- VPC
- Internet Gateway
- EIP
- Subnet
- Route
- Security Group
- IAM Group
- IAM Policy
- IAM User
- CloudWatch
- RDS
- DynamoDB
- ECR
- ECS Cluster
- ECS Task Definition
- ECS Service



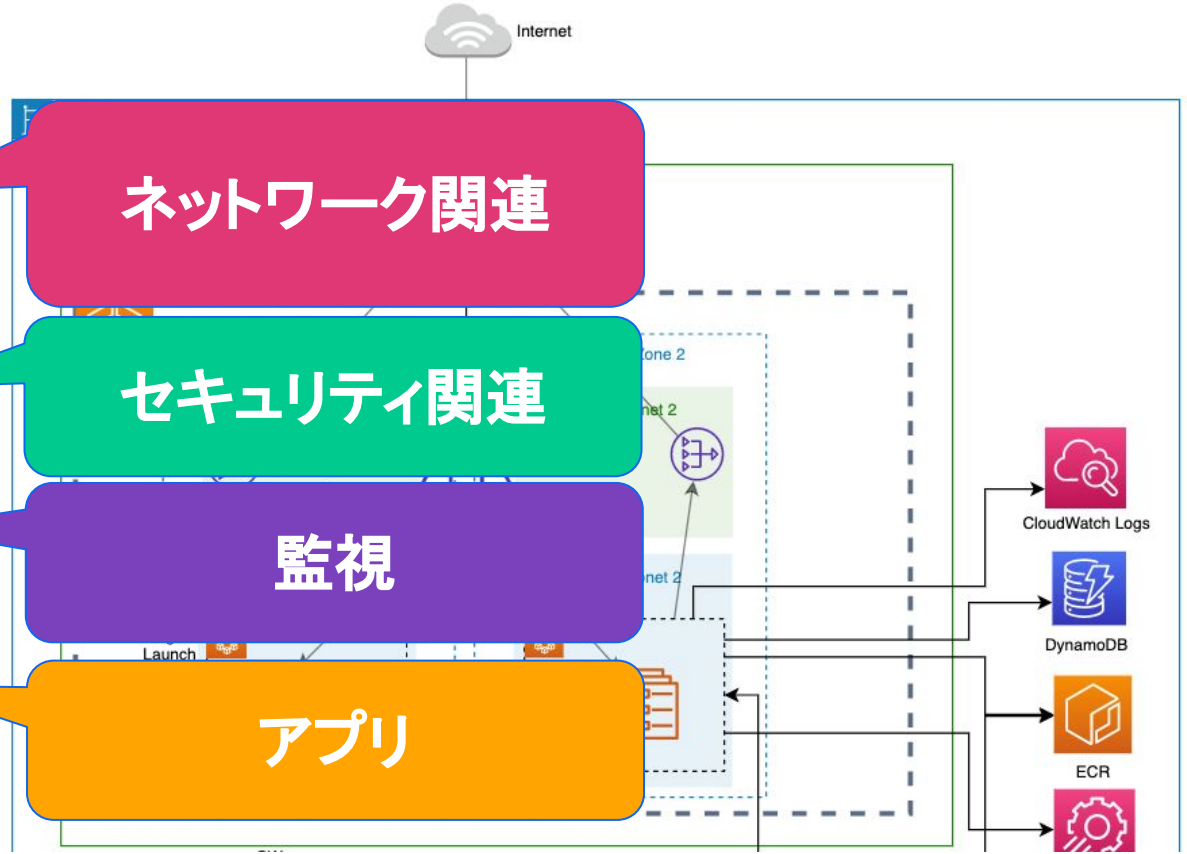
クラウド上で動くアプリの構成例

- VPC
- Internet Gateway
- EIP
- Subnet
- Route

- Security Group
- IAM Group
- IAM Policy
- IAM User

- CloudWatch

- RDS
- DynamoDB
- ECR
- ECS Cluster
- ECS Task Definition
- ECS Service



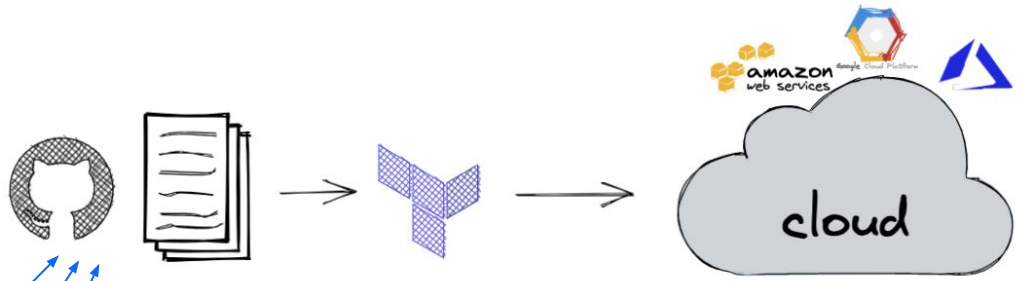
クラウド上で動くアプリの構成例

- VPC
- Internet Gateway
- EIP
- Subnet
- Route

- Security Group
- IAM Group
- IAM Policy
- IAM User

- CloudWatch

- RDS
- DynamoDB
- ECR
- ECS Cluster
- ECS Task Definition
- ECS Service

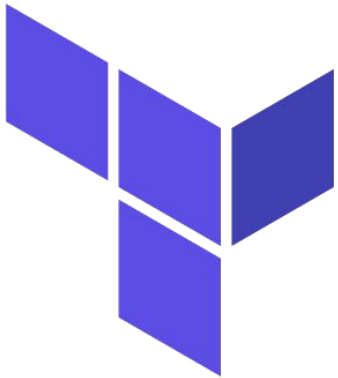


クラウドの自動化では、

- ネットワーク
- セキュリティ
- 監視設定
- アプリケーション

ほとんどのコンポーネントが、一貫したワークフローで運用されている (DevOps)
Terraform はマルチクラウドに対応したオープンソースの IaC のツールとしてデファクトスタンダードな存在

Terraform is 何?



HashiCorp
Terraform

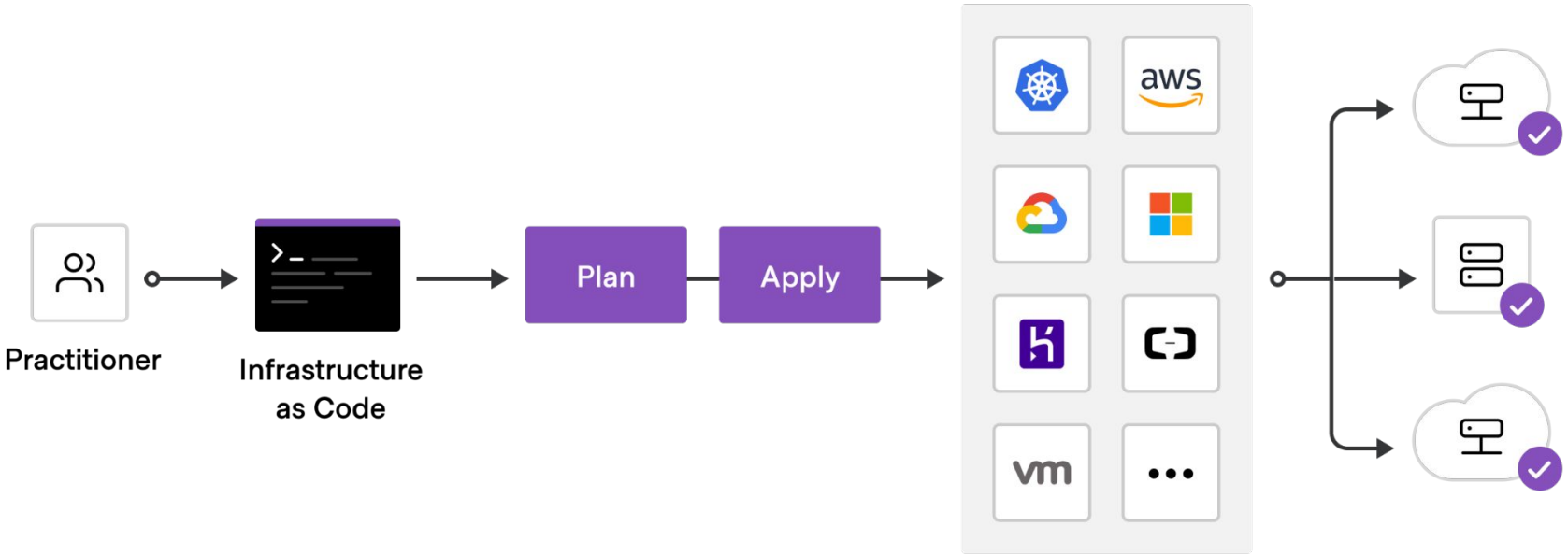
Terraform はオープンソースのプロビジョニングツール
(2014年～、2021/6 に version 1.0 GA, 現在latest 1.1.4)

Go で書かれたシングルバイナリ。Terraform はLinux、
Windows、MacOS などクロスプラットフォームで動作。

Terraform では

- プロビジョニング先を**provider**、
- 設定するインフラを**resource** として扱い
- **宣言的**にプロビジョニングを実行します
- また **state** ファイルにてライフサイクルの管理を行います（ドリフト検出が可能）

様々なインフラへのプロビジョニング



- 150 を超える公式プロバイダ
- 1,300 を超えるCommunity プロバイダ

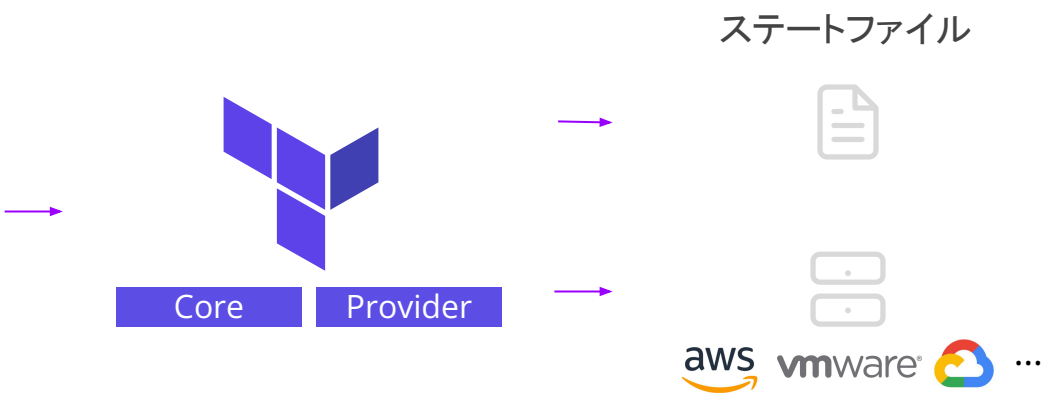
Terraform によるプロビジョニングイメージ

HashiCorp Configuration Language

```
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
     }}
```



実行可能なパラメータシート
(のようなもの)

Terraformによるプロビジョニングイメージ


```
resource "aws_instance" "web_ec2" {
  ami    = var.ami
  count = 1
  tags = merge(var.tags, map(
    "Name", "instance-a",
    "ENV", "DEV"
  ))
  instance_type = "t2-small"
  vpc_security_group_ids =
"my-sec-group"
  associate_public_ip_address = true
}
```

```
resource "aws_vpc" "main" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "main"
  }
}
```

```
resource "aws_subnet" "main" {
  vpc_id     = aws_vpc.main.id
  cidr_block = "10.0.1.0/24"

  tags = {
    Name = "Main"
  }
}
```

『あるべき形』をコードとして記述

Terraform のプロバイダが増加中

Cisco, Citrix, and Fortinet Among New Verified Terraform Providers

Cloud, SaaS だけでなく

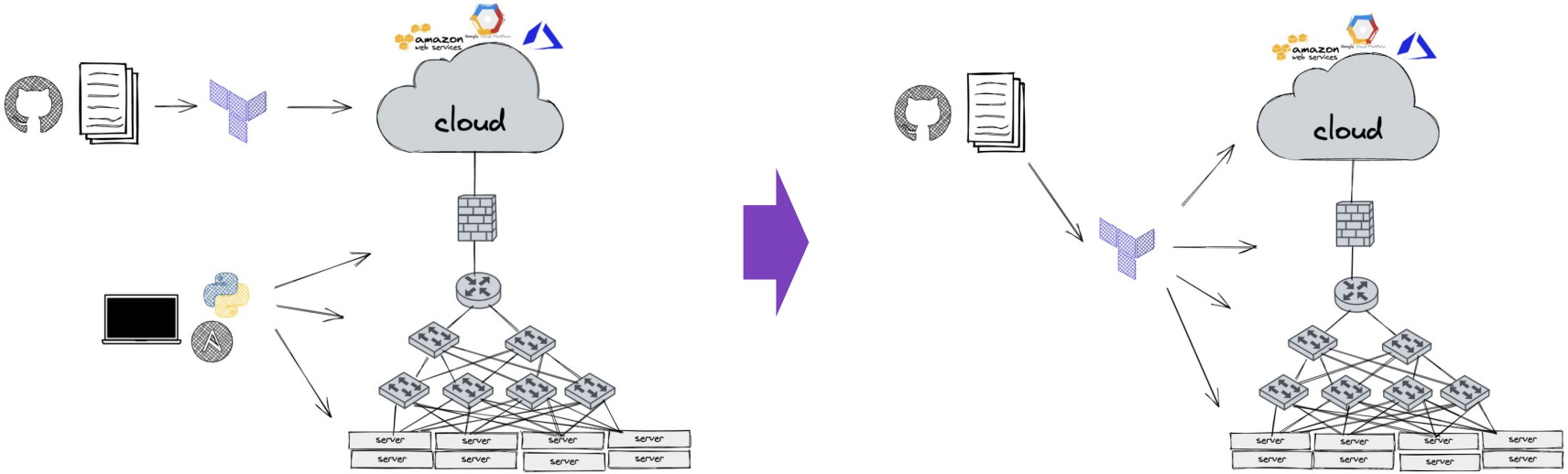
対応ネットワークベンダーも増加中

- Cisco
- Citrix
- Fortinet
- Purestorage
- F5
- Arista
- Vmware
- Paloalto
- A10
- Infoblox
- Bluecat
- Cloudflare
- Fastly

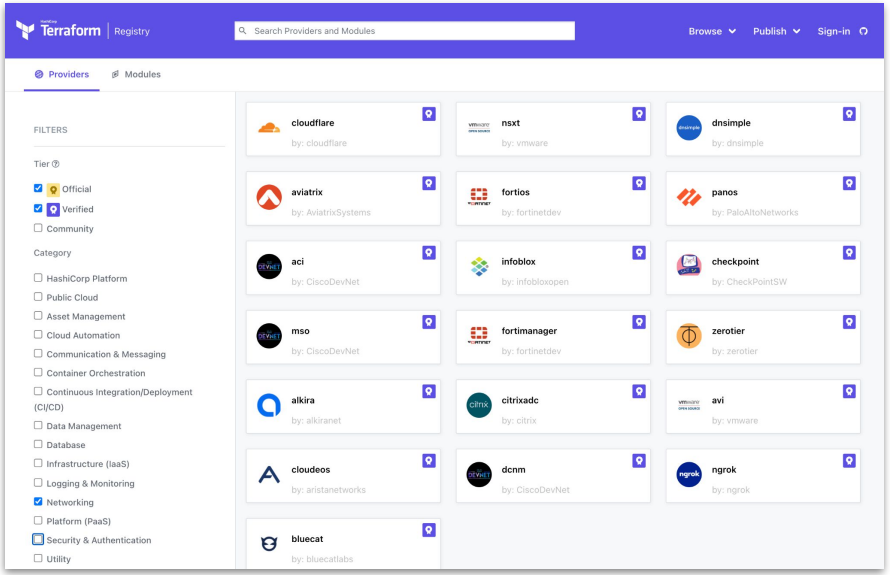
https://www.hashicorp.com/blog/cisco-citrix-fortinet-among-new-verified-terraform-providers

てことは...

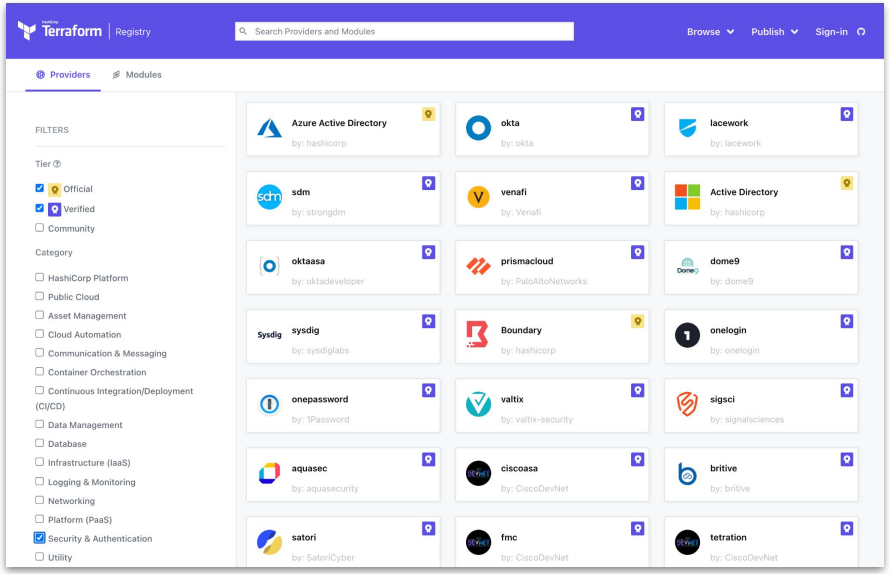
クラウド運用と同様の 運用パイプラインに組み込むこともできそう？



registry.terraform.io でプロバイダを検索



https://registry.terraform.io/search/providers?category=networking



https://registry.terraform.io/search/providers?category=security-authentication

プロバイダを開発するための SDKもあるので、プロバイダを自作することも可能

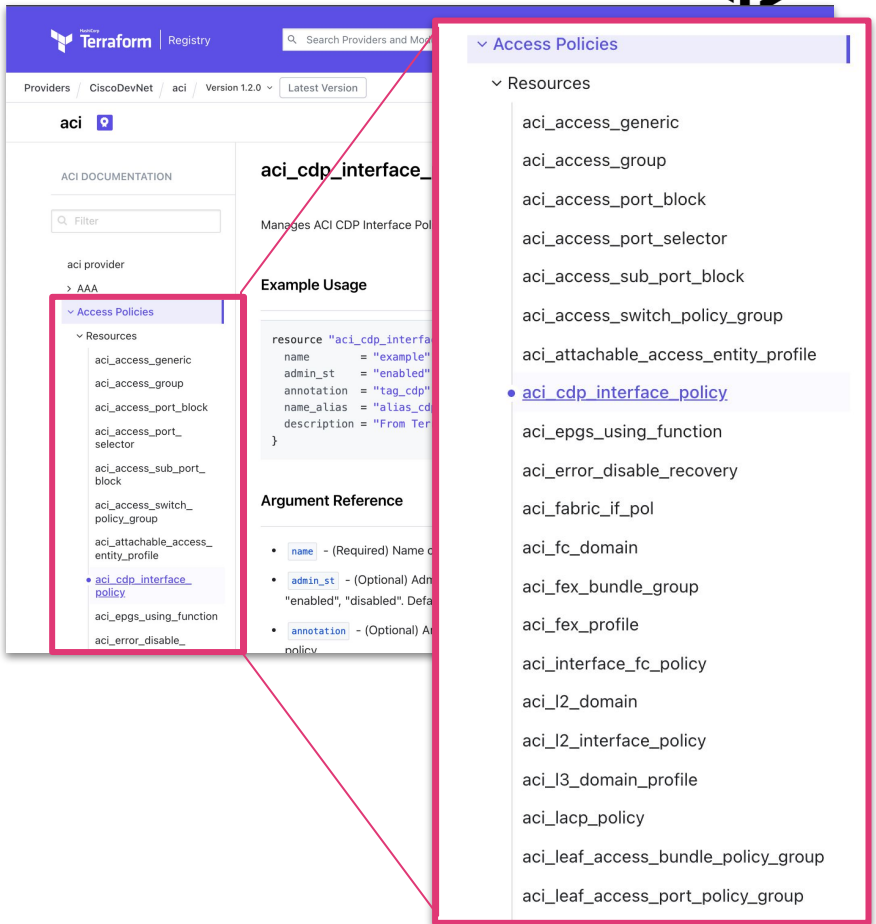
対応リソースを確認する

Terraform では設定対象をリソースとみなす

- 追加したいリソースをコードに書く（宣言的）

```
resource "fortios_router_static" "test" {
 blackhole           = "disable"
 device              = "port1"
 distance            = 10
 dst                 = "120.2.2.122/32"
 dynamic_gateway     = "enable"
 status              = "enable"
 weight              = 2
}
```

- リソースとして定義されていないものは設定できない
  - 1行毎にコマンドを入力という設定はない
  - REST API を叩くリソースを提供している プロバイダも存在



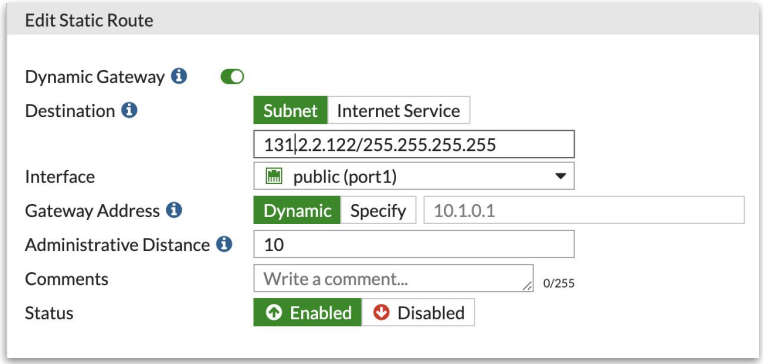
NW関連 Terraform プロバイダの対応状況

2022年1月14日 現在での傾向

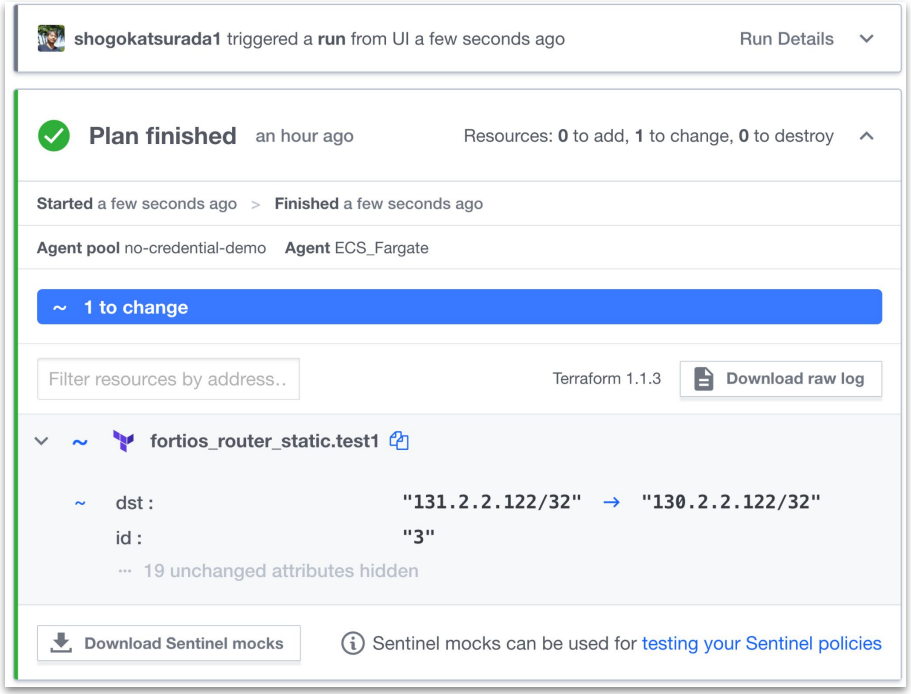
- プロバイダ数
  - Networking カテゴリ: 103
  - Security & Authentication カテゴリ: 155
- 幾つかの主要ベンダーでの対応が進んでいる
- NW製品は、クラウド親和性の高い製品、SDN製品が多い
  - APIが叩ける製品, 仮想アプライアンスなど
- FW, LB の対応は、そこそこ進んでいる印象
  - ACL/ポリシーの追加/削除など、宣言型の需要は多そう
- ほとんどがここ1-2年以内リリースされたもの
  - 頻繁にアップデートされている

Demo

ドリフト検出の例



Terraform で設定したアドレスを
手動で変えてしまった！



Terraform plan を実行すると、差分が表示

まとめと議論に向けて

クラウドオペレーティングモデルで、運用の壁を ”乗り越える”

Terraform のネットワークソリューション対応プロバイダも増えて来ました

- クラウド基盤、ネットワーク基盤も併せた自動化が捗りそう
- 宣言型、ドリフト検出など、手続き型とは違うアプローチの自動化
- 若手の育成、クラウド人材の採用に良いかも？

ただ、課題も多そう...

- 組織の問題、Cloud CoE/自動化チームがあるか？
- 対応製品の数（時間が解決しそうではある）

皆さんに聞いてみたいこと

- クラウド運用チームとネットワーク運用チームの壁
  - クラウド運用とネットワーク運用の壁 “乗り越える” べき？
- Terraformをネットワークに適用するアプローチはいかがですか？
- Terraformをネットワークで使われてる方いらっしゃいますか？
- こういう事が出来たら使ってみたい

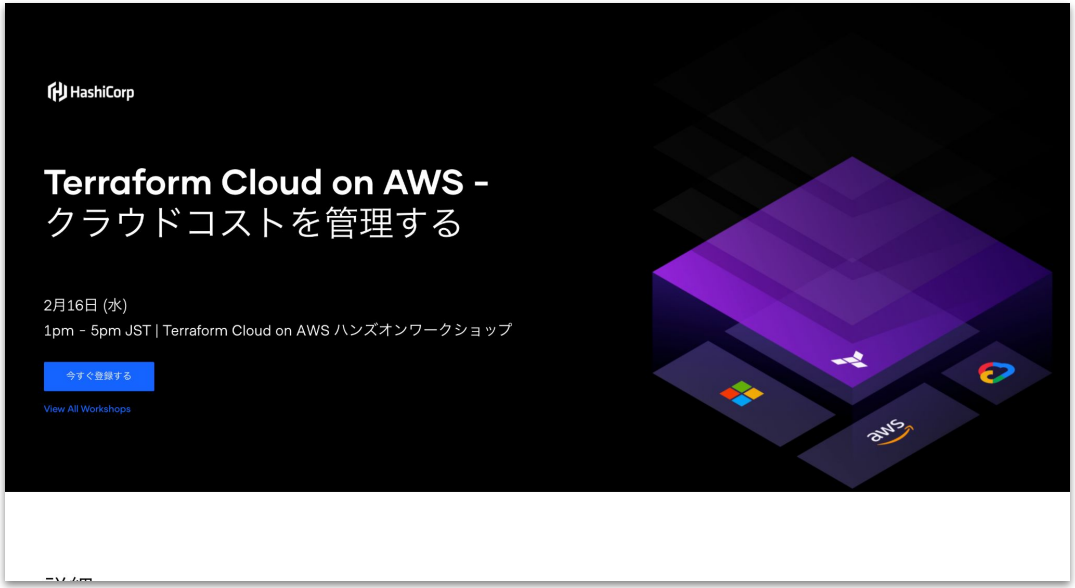
など、その他ご質問

Terraform ネットワークインフラ用のプロバイダは、数が増えてきましたが、まだまだリリースから日も浅いものが多いので、みなさんと一緒に知見をためていけると嬉しいです！

ハンズオン 企画しました

Terraformについて、勉強してみたい方、ぜひご参加ください〜

https://events.hashicorp.com/workshops/terraform-2/16/JANOG



その他イベント https://events.hashicorp.com/jpworkshops

Thank You

shogo.katsurada@hashicorp.com
www.hashicorp.com

宣言型 vs 手続き型

Terraform（宣言モデル）

```
resource "aws_vpc" "default" {
  cidr_block           = var.vpc_cidr_block
  enable_dns_hostnames = true

  tags = {
    Name = "dev"
  }
}

resource "aws_subnet" "default" {
  availability_zone       = var.availability_zone_a
  cidr_block              = "10.0.0.0/24"
  map_public_ip_on_launch = false
  vpc_id                  = aws_vpc.default.id

  tags = {
    Name = "dev"
  }
}
```

7

Ansible（命令実行モデル）

```
tasks:
  - name: "VPC"
    ec2_vpc_net:
      name: "vpc"
      cidr_block: 10.0.0.0/24
      dns_hostnames: yes
    register: _vpc
  - name: "Public subnet"
    ec2_vpc_subnet:
      vpc_id: "{{ _vpc.vpc.id }}"
      az: "{{ item.az }}"
      cidr: "{{ item.cidr }}"
      resource_tags:
        Name: "dev"
    with_items: "{{ subnet }}"
```

Ansibleは、リソースの作成順序をユーザが定義するため、依存関係を考慮してコードを定義する必要がある。

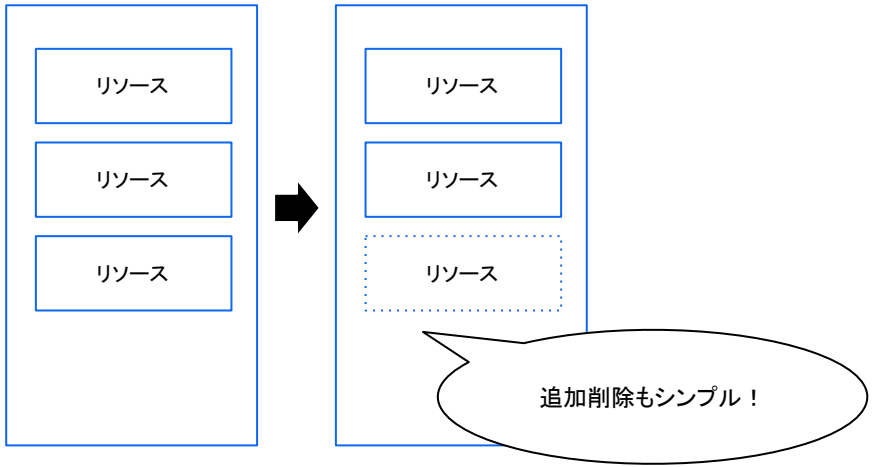
他方Terraformは、リソースの依存関係を自動で解決するため、作成順序をコードで定義する必要がない。図のAnsibleで、VPCよりサブネットを先に記述した場合には、実行時にエラーとなる。

宣言型 vs 手続き型 - Cont.

Network自動化に関連して

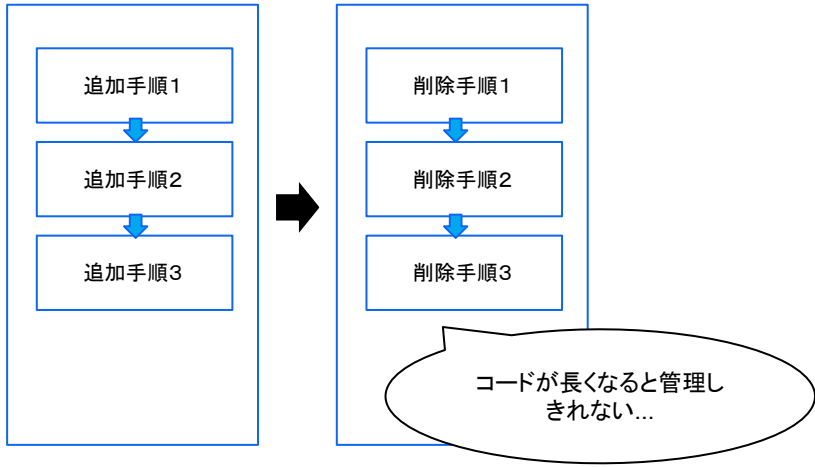
アクセスリストや、テナントを追加・削除するケース

宣言型の設定ファイル(状態を記述)



宣言型の場合は、あるべき状態を記載すればインフラに反映されるので、コードから消せば、削除される（VCS一つで状態の管理がしやすい）

手続き型の設定ファイル(手順を記述)



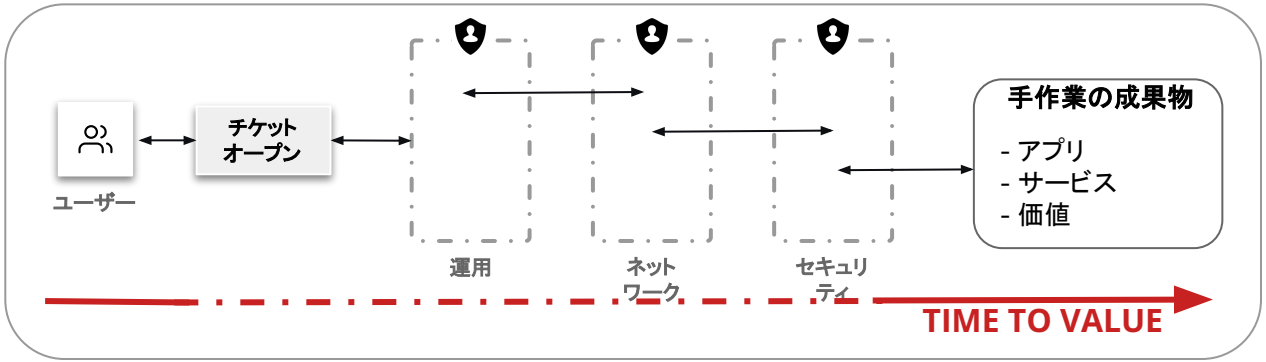
手続き型の場合は、手順書をコード化しているため、現在の状態は管理しない（別途管理が必要）
削除のためのコードを、順序通り書く必要があり、管理が複雑化しやすい

クラウドの世界でのワークフローは？

従来のワークフロー:

運用者がインフラに直接設定投入する

- **Manufacturing Snowflakes**
- **Mutable インフラ**
- **変更に時間とコストがかかる**



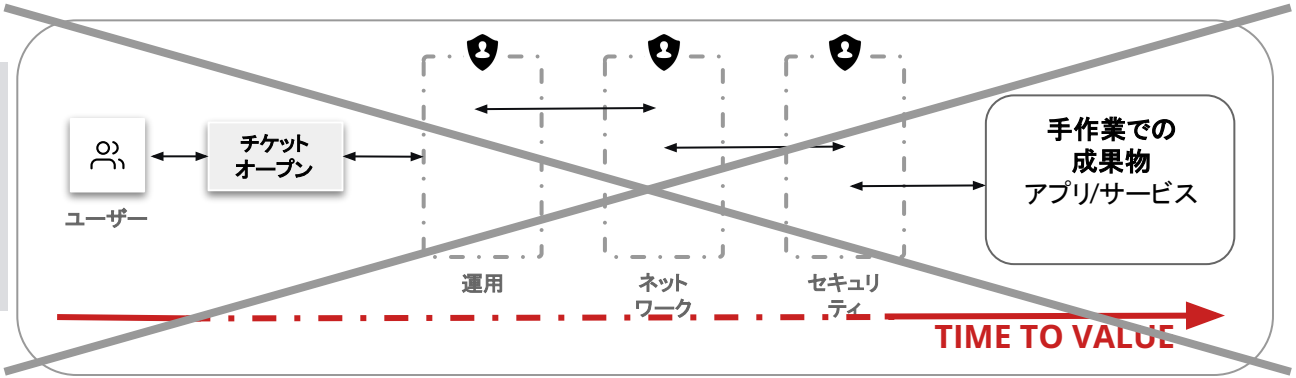
手動で行う時代には理にかなっていたが…
こういったワークフローは、
今日のビジネス要求を満たさないケースも…

クラウドの世界でのワークフローは？

従来のワークフロー:

運用者がインフラに直接設定投入する

- Manufacturing Snowflakes
- Mutable インフラ
- 変更に時間とコストがかかる



クラウド時代のワークフロー:

運用者が、IaCパイプラインの権限を持つ

- データドリブン -> IaC -> 自動化
- Immutable インフラ
- 変更は反復的、迅速かつ容易に