# 【Terraform入門コース】

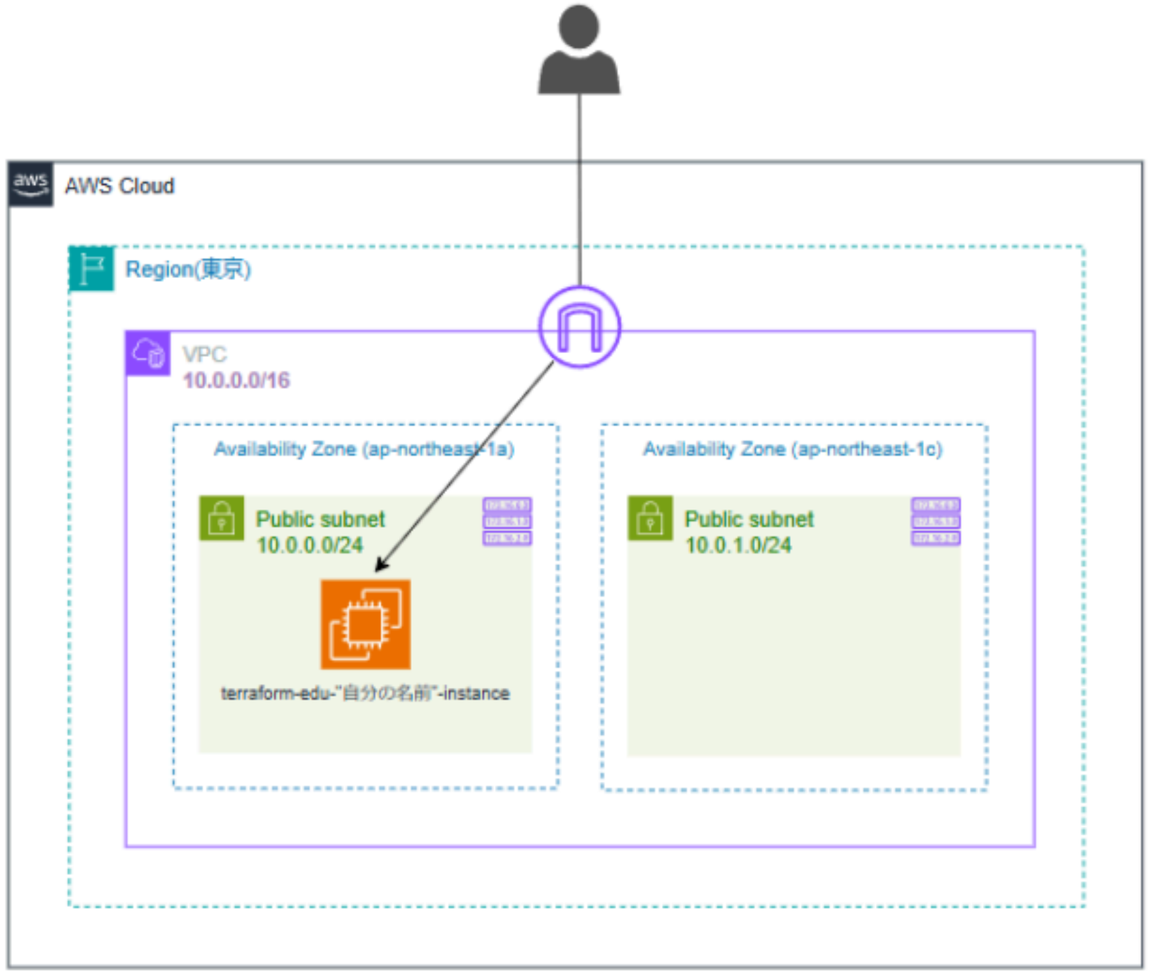
## IaC と Terraform の概要

by @kitasan_chi

# 本研修の対象者と目標

- **実施時間**
  Terraform 入門研修 (1時間 ~ 2時間) ※内休憩10分
- **対象者**
  インフラ構築の最低限の基礎知識がある方（VPC,EC2等）
  IaC に興味があり、Terraform を初めて触る方
- **目標**
  Terraform の基本的な概念を理解する
  Terraform を使ってAWS上に簡単なリソースを
  作成・変更・削除できるようになる

# 今回作成する構成図




## よくある構成

★VPC - 10.0.0/16

★パブリックサブネット×2

1a - 10.0.0/24 , 1c - 10.0.1/24

★IGW,RT,SG

★EC2

インスタンスタイプ：t3.micro

EBS：4GB

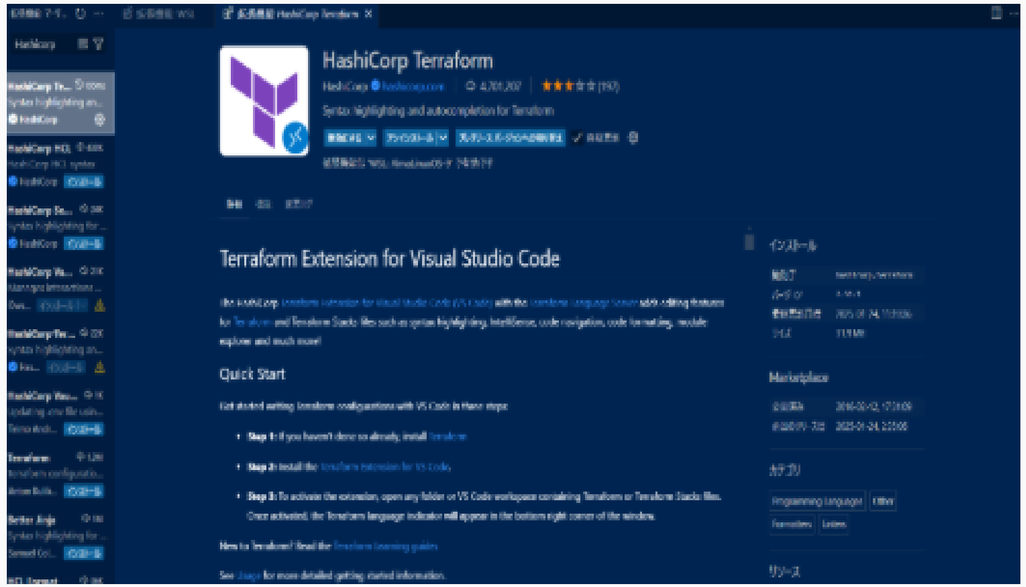
# タイムテーブル

| 時間 | 内容 | 資料リンク |
| --- | --- | --- |
| 15分くらい | IaCとTerraformの概要<br>(座学) | https://www.canva.com/design/DAGjRbyjsfg/af5cwpHJQReqzM1zir8viQ/edit?utm_content=DAGjRbyjsfg&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton |
| それ以降の時間(21時まで) | 基本操作解説・実践<br>(座学+ハンズオン) | https://www.canva.com/design/DAGjSNDhW90/JT9x1Lg4xGTrjm_lG6wbew/edit?utm_content=DAGjSNDhW90&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton |

# 前提条件

1. VS Code が入っていること
2. VS Code 拡張機能の「Remote - SSH」と「HashiCorp Terraform」が入っていること

※未導入の方は座学中に導入資料からインストールをお願いします


導入資料はこちら

# IaC とは

- Infrastructure as Code の略

  ソフトウェア開発の手法をインフラのシステム管理に応用するための方法論
  コードによる自動化 / バージョン管理 / テスト / デプロイが可能
  システム(インフラ)管理は今までは手作業でやっていた
  ソフトウェア開発と同様にインフラを「コードを書いて管理しよう」という話

- Infrastructure as Code の略

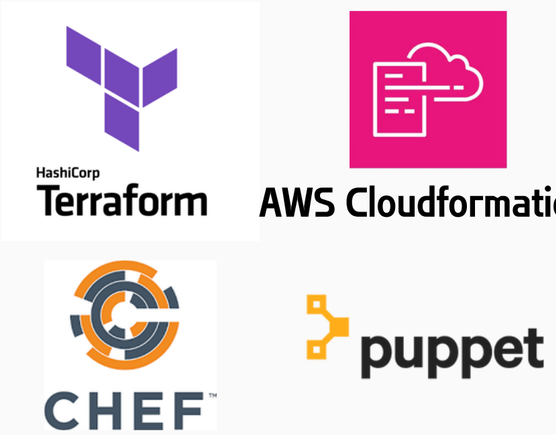
  2006年、当時の主要技術を覆すほどに革新的な技術だった
  特にWEBアプリケーション開発はオンプレミスからクラウドが主流になっていった
  その過程で IaC もより強く求められるようになる
  最近注目されるようになっただけで、IaC という方法論自体は以前から存在する

# IaC とは

OSやミドルウェアの
設定自動化


インフラ構築の
自動化


プログラミング言語(複数対応)
を抽象化して扱う




定義型コードで
低レイヤーのまま扱う

# Terraform とは

- Hashicorp社により開発されているIaCを実現するツールの1つ
- 2025/01 時点で v1.10.5が最新

  初回リリースが2014年のv0.1.0なのでツールとしてはそれほど新しいものではない

# Terraform の特徴

- **Go言語で開発されている**

  ビルド後のバイナリファイルがあれば、Windows/Linux/Mac どこでも動く


- **マルチプロバイダー**

  様々なクラウドプラットフォームに対応している

  AWS / GCP / Azure はもちろん Alibaba Cloud や Oracle Cloud 等も対応

  クラウドプラットフォーム以外の構築を自動化することも可能

  Kubernetesや Datadog の設定にも対応している

- **コードはHCLと呼ばれる独自言語で記述する**

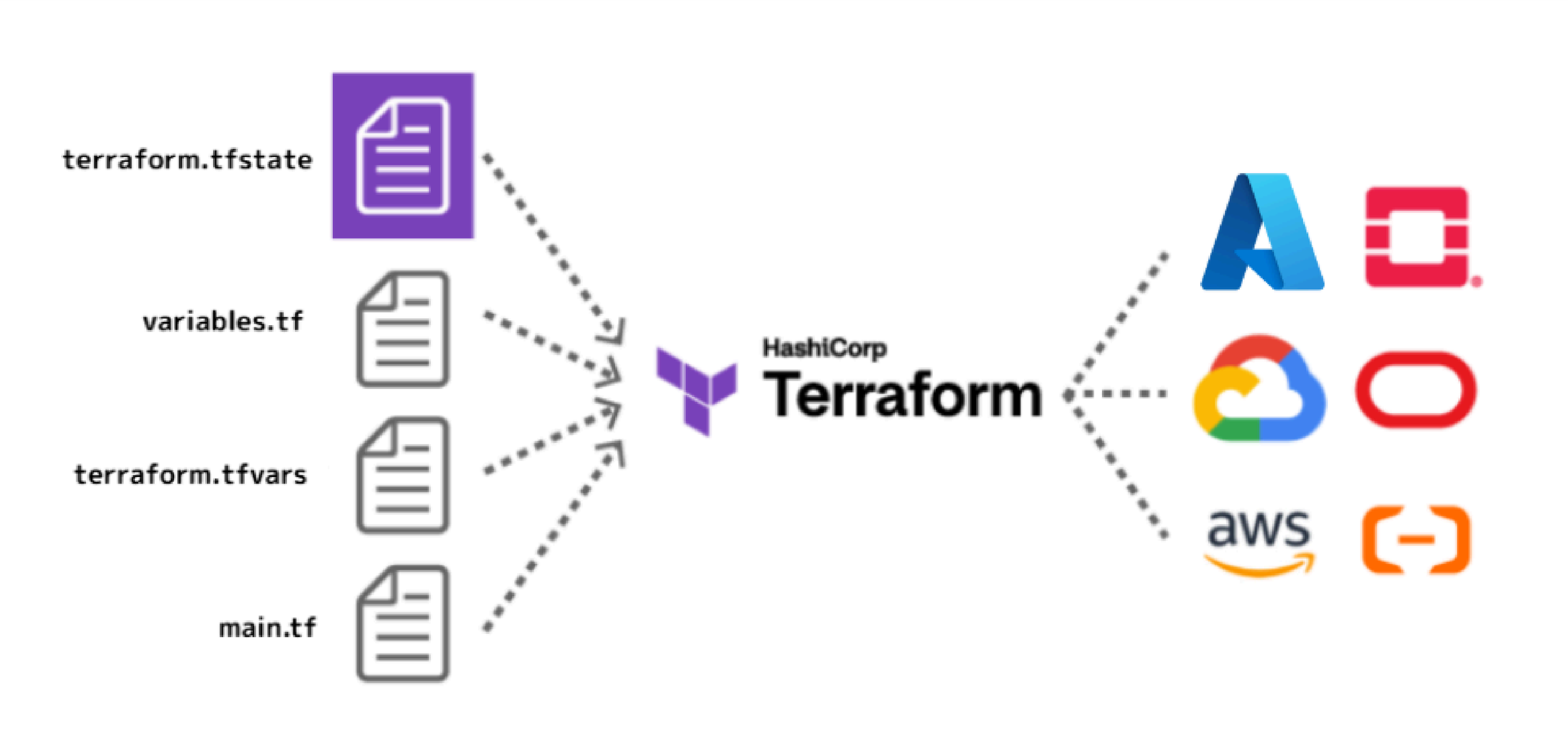
  **(Hashicorp Configuration Language)　※後述**

  HCLを覚える必要はあるが、それさえ覚えれば複数のクラウドを管理できる

  クラウドごとにある、プロビジョニングサービスの使い方を覚える必要がない。

  例：AWS CloudFormation / GCP Cloud Development Manager

# Terraform の特徴



# Terraform と Ansible の違い

- **Terraform はインフラ層、Ansible はOS・ミドルウェア層の管理に最適**

  技術的に言えば、どちらでもお互いの領域の自動化は可能

  例：AWS の EC2 を構築したい

  ⇒ Terraform は HCL、Ansible は yaml でコードを書いてデプロイすればいい

- **それぞれ IaC のモデルが異なる**

  Terraform は宣言モデル、Ansible は命令実行モデルを採用している

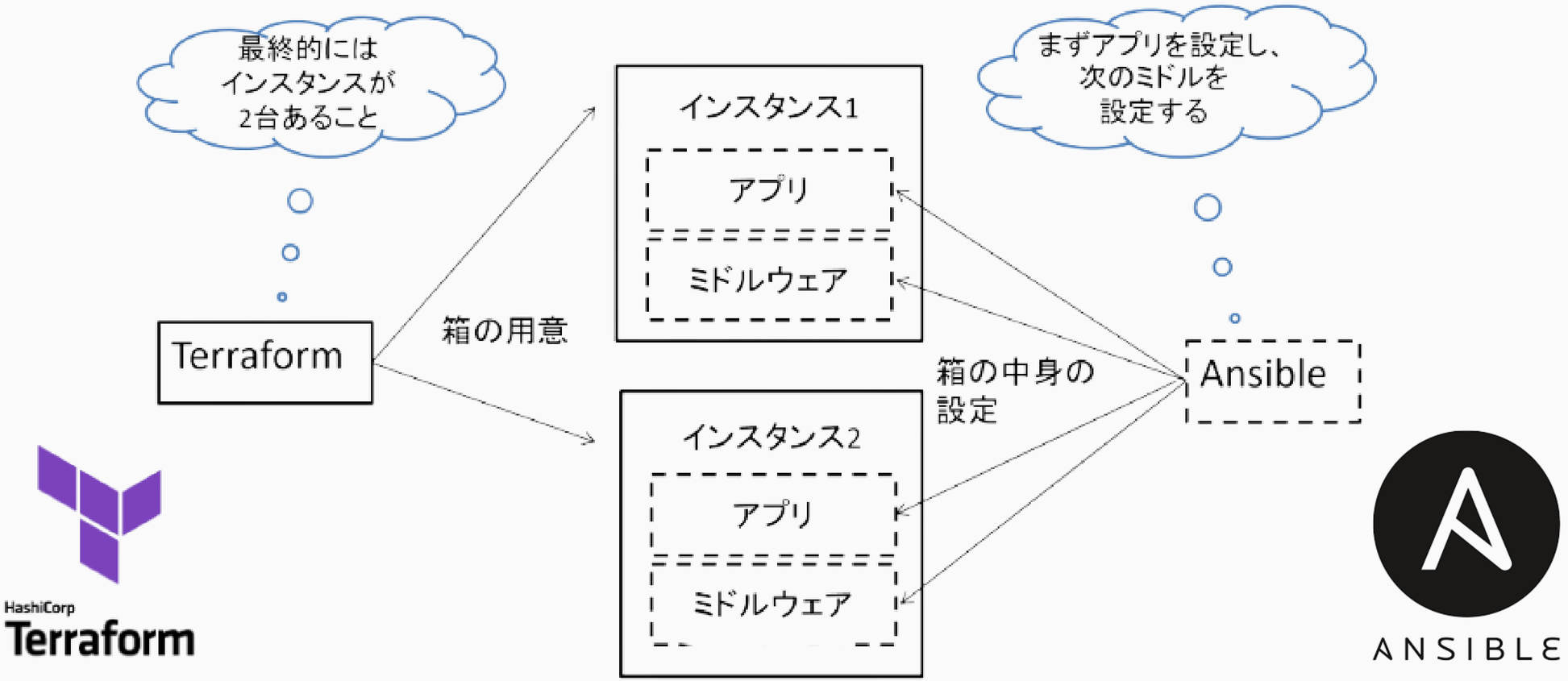
  宣言モデル　　→ インフラのあるべき状態をコードで表現する

  命令実行モデル → 必要なリソースを作成する手順をコードで記述する

  この辺は派閥や信者がいて怖いので、あんまり触れたくないです

  Ansible 講座やってくれる人お待ちしております

# Terraform と Ansible の違い



# Terraform の優位性

- **インフラ層の構築冪等性は Terraform の方が上手く担保できる**
  - 冪等性 → ある操作を1回行っても複数回行っても結果が同じであることをいう概念
  - 例：AWS EC2を10台作成したい
    - Terraform → 宣言モデルのため、何度実行しても10台しか作られない (tfstateファイルで管理)
    - Ansible → 命令実行モデルのため、10 * 実行回数分作成されてしまう
- **リソースの依存関係を上手く解釈してくれる**
  - AWS サブネットを作成するためには、VPC を先に構築しなければならない
  - コード上サブネットの記述を先に書いたとしても、自動で依存関係を解釈してくれる
  - VPC の後にサブネットを作成してくれる
  - Ansible はサブネットを先に記述すると VPC がないためエラーになる
- **故にマルチクラウドの管理を行うのであれば Terraform が最適**
  - 実際の使い方はこの後説明していきます
  - 逆にOS・ミドルウェアの管理は可能だが不向き
  - Ansible に任せた方がいい、上手く使い分ける

# HCL の書き方

- HCL では

Terraformのコードは、HCL(HashiCorp Configuration Language)という言語を利用して記述していきます。
構造は非常にシンプルで、初心者でも分かりやすい言語となっています。
また、インフラ構成の定義は、「.tf」という拡張子のファイルに、各インフラのリソースの定義をコードで記述します。

```
resource “<プロバイダ>_<タイプ>” “<名前(リソース名)>” {
    項目1 = 項目のオプション
    項目2 = 項目のオプション
    項目3 = 項目のオプション
}
```

**例:**

```
resource “aws_instance” “example” {
    ami           = ”ami-XXXXXXXXXXXXXXXX”
    instance_type = ”t2.micro”
}
```

**ドキュメント：https://registry.terraform.io/providers/hashicorp/aws/latest/docs**

では、実際に触っていきましょう！！