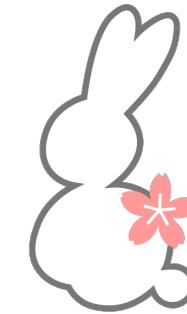




HashiCorp

**Terraform**



**usacloud**

ハンズオン for さくらのクラウド  
～Terraform編～

2017/9/26

@yamamoto\_febc

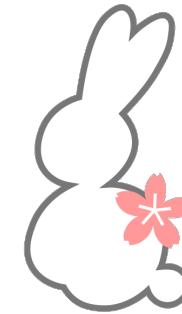
# タイムテーブル

開始時刻	概要	内容
19:10～	概要説明(座学)	<ul style="list-style-type: none"><li>• Infrastructure as Codeとは</li><li>• usacloudの紹介</li><li>• Terraformの紹介</li></ul>
19:40～	環境構築 (休憩含む)	<ul style="list-style-type: none"><li>• クーポン適用</li><li>• さくらのクラウドAPIキー取得/設定</li></ul>
20:00～	usacloud編	<ul style="list-style-type: none"><li>• ハンズオン</li></ul>
21:00～	Terraform編	<ul style="list-style-type: none"><li>• ハンズオン</li></ul>



HashiCorp

**Terraform**



**usacloud**

# 環境構築編

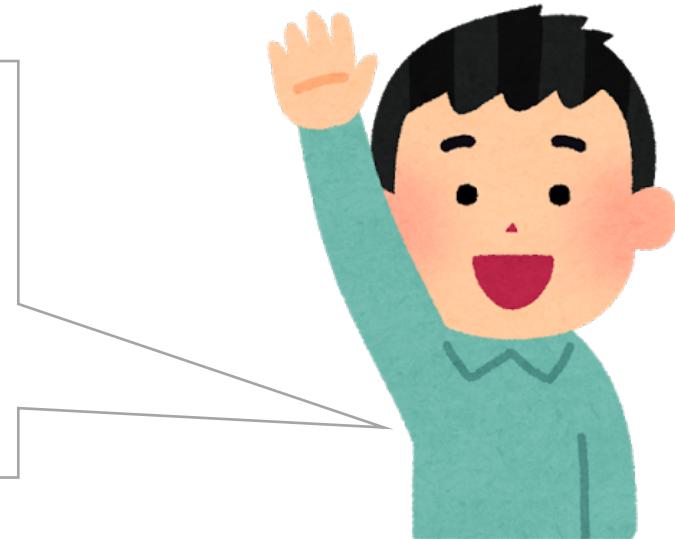


# クーポンの適用

## クーポン適用マニュアル

<https://manual.sakura.ad.jp/cloud/payment/coupon.html#coupon-apply>

クーポン適用方法がわからない場合  
声をかけてください！！





# APIキーの取得

## usacloud 導入ガイド

[https://sacloud.github.io/usacloud/start\\_guide/#api](https://sacloud.github.io/usacloud/start_guide/#api)



「他サービスへのアクセス権」は  
請求情報にチェックを！！



# APIキーの取得

APIキーを追加

名前\* 任意の値を入力

説明 「作成・削除」を選択

アクセスレベル \* 無効 リソース閲覧 電源操作 設定集 作成・削除

他サービスへのアクセス権  請求情報  ウェブアクセラレータ

「他サービスへのアクセス権」は  
請求情報にチェックを入れる！！



# APIキーの取得

webaccel  
APIキー (詳細表示)

リソース ID	[REDACTED]
名前	webaccel
説明	webアクセラレーター関連テスト用
ACCESS TOKEN	b[REDACTED]21
ACCESS TOKEN SECRET	NUL[REDACTED]oPQib[REDACTED]i84g
アクセスレベル	リソース閲覧
他サービスへのアクセス権	ウェブアクセラレータ
作成日時	[REDACTED]

アクセストークン  
アクセスシークレット



この画面を開いたままにしておいてください  
(後で使います)



# APIキーの設定



いくつか設定方法がありますが、  
今日は 「環境変数」 を使います！！



# APIキーの設定

## Unix系OSなどの場合

```
# アクセストークン  
$ export SAKURACLOUD_ACCESS_TOKEN=入力
```

```
# アクセスシークレット  
$ export SAKURACLOUD_ACCESS_TOKEN_SECRET=入力
```

```
# デフォルトゾーン(石狩第2を指定)  
$ export SAKURACLOUD_ZONE=is1b
```



# APIキーの設定

## Windowsの場合

```
# アクセストークン  
$ setx SAKURACLOUD_ACCESS_TOKEN 入力
```

```
# アクセスシークレット  
$ setx SAKURACLOUD_ACCESS_TOKEN_SECRET 入力
```

```
# デフォルトゾーン(石狩第2を指定)  
$ setx SAKURACLOUD_ZONE is1b
```

※設定後、コマンドプロンプトの再起動を行ってください。



HashiCorp

**Terraform**

インストール



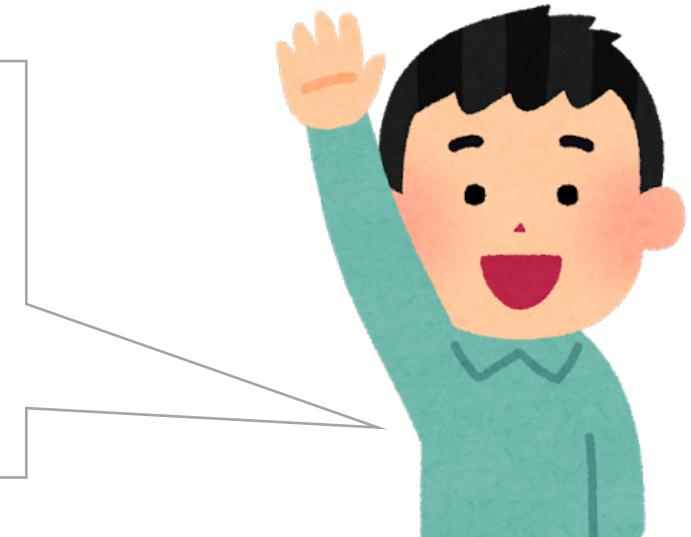
## インストール

Terraform for さくらのクラウド 導入ガイド

<https://sacloud.github.io/terraform-provider-sakuracloud/installation/>

インストール方法に不安のある方は  
声をかけてください！

注: Terraform本体とプラグイン両方の  
ダウンロードが必要です！！





# インストールの確認

```
# ヘルプを表示  
$ terraform --help
```



インストールできていれば  
ヘルプが表示できるはず



# ログイン情報(APIキー)の設定

TerraformではAPIキーを様々な方法で指定可能です。

ハンズオンでは「環境変数」を利用します。

その他の指定方法についてはTerraformのドキュメントなどを参照してください。



参考 : Terraform for さくらのクラウド スタートガイド(全5回)  
<http://knowledge.sakura.ad.jp/knowledge/7230/>



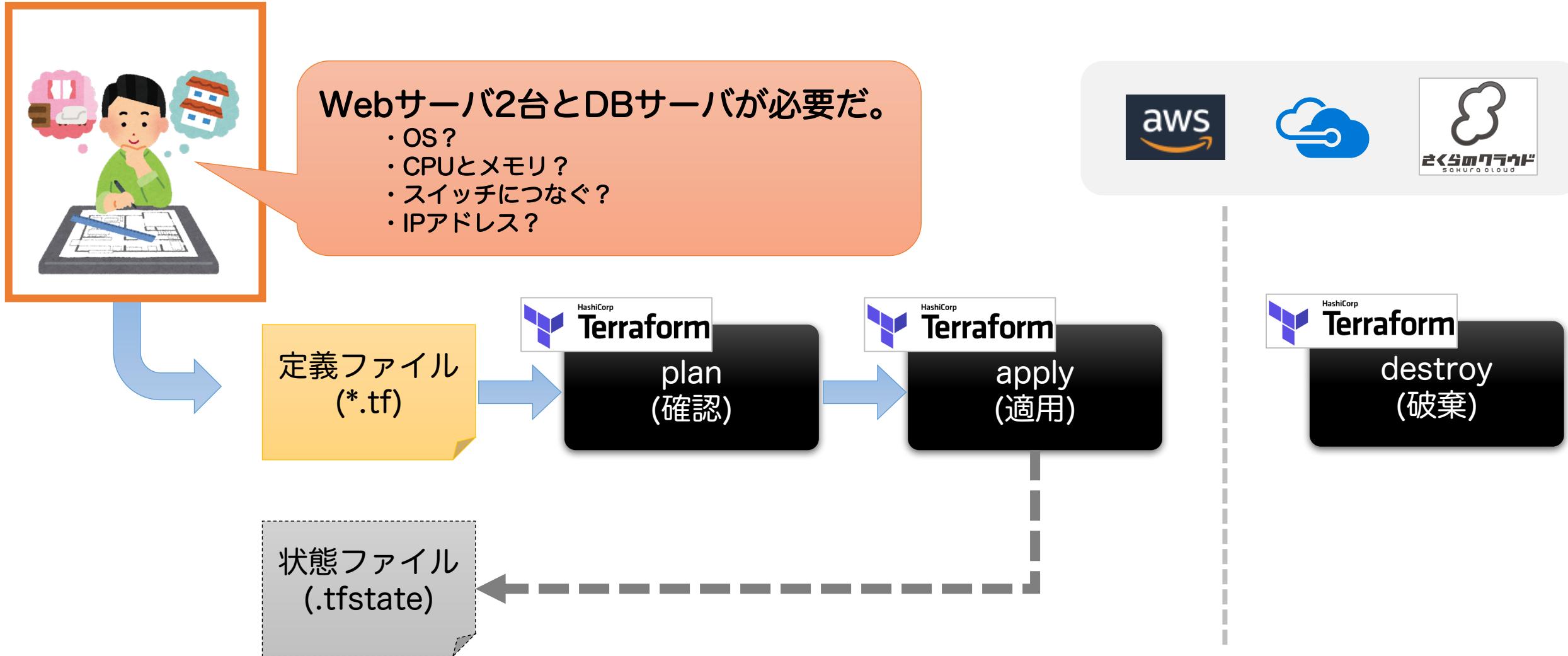
HashiCorp

**Terraform**

**インフラ構築の流れ**



# インフラ構築の流れ





# インフラ構築の流れ



HCLで定義を作成

定義ファイル  
(\*.tf)

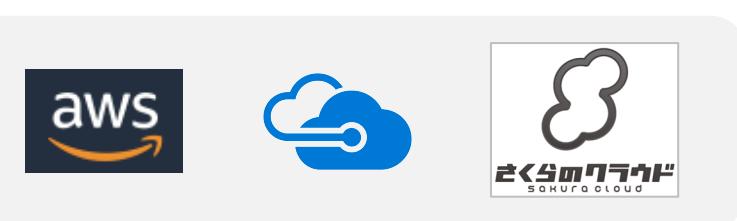


plan  
(確認)



apply  
(適用)

状態ファイル  
(.tfstate)

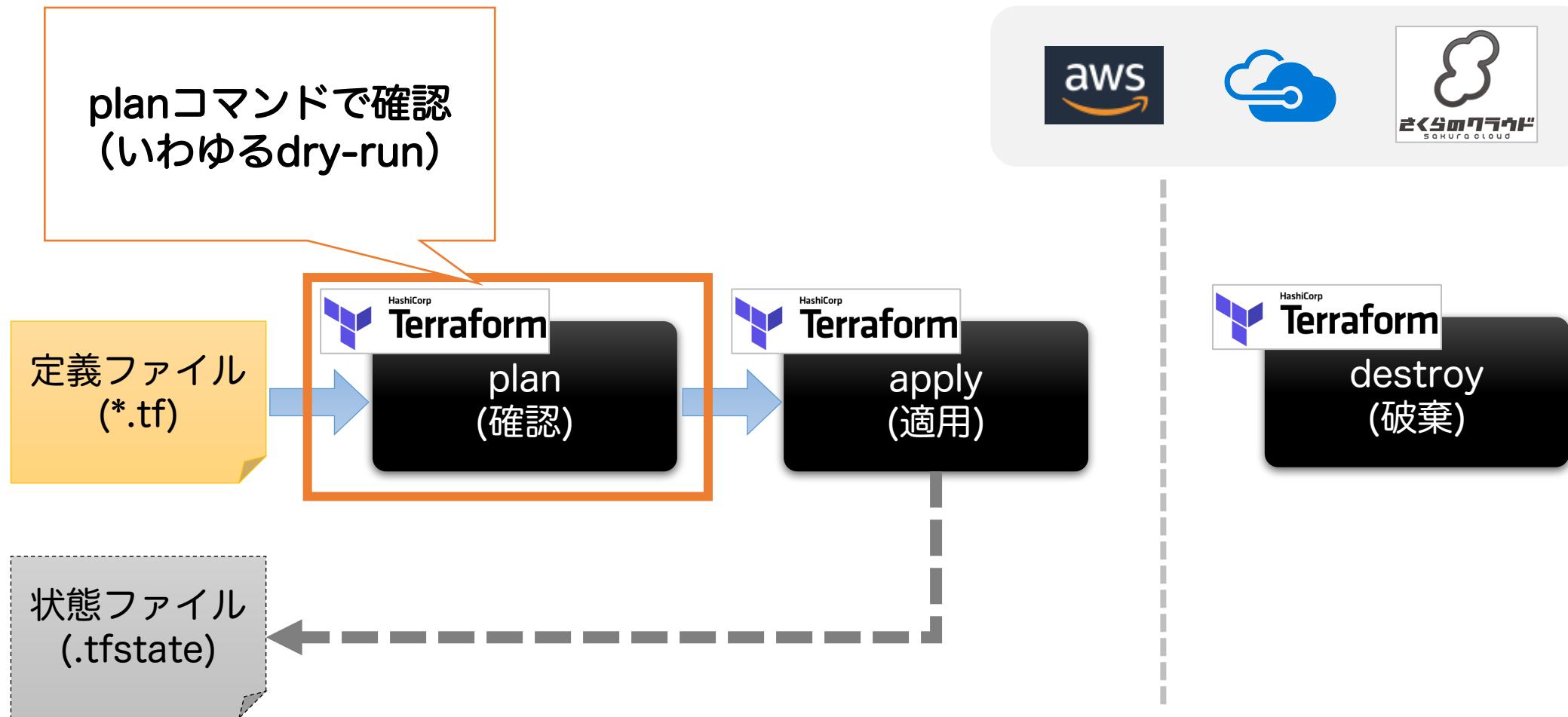
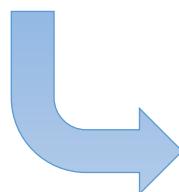


HashiCorp  
**Terraform**  
destroy  
(破棄)



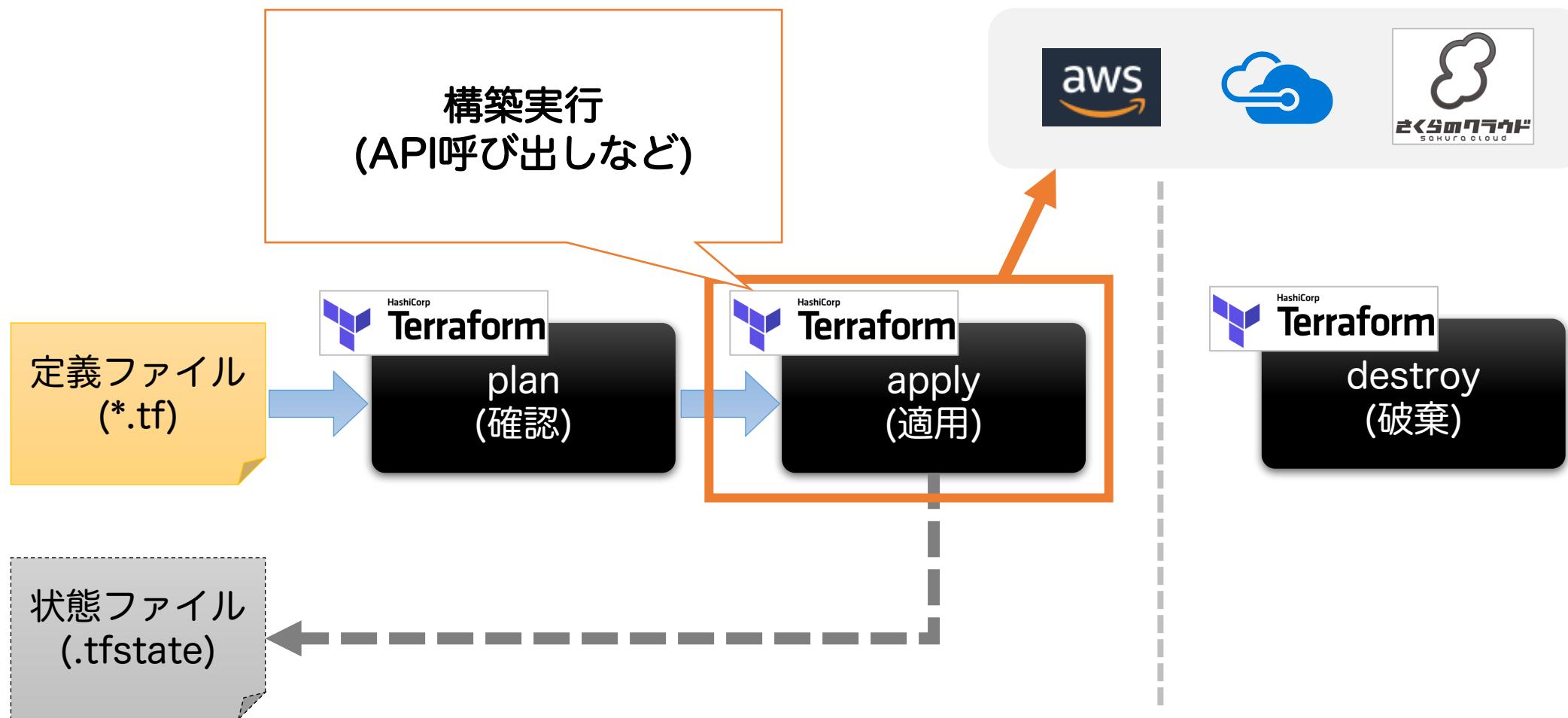
HashiCorp  
**Terraform**

# インフラ構築の流れ





# インフラ構築の流れ





HashiCorp  
**Terraform**

# インフラ構築の流れ



今のインフラの状態を  
ファイル(など)で保持しておく

定義  
(\*.tf)



HashiCorp  
**Terraform**

apply  
(適用)



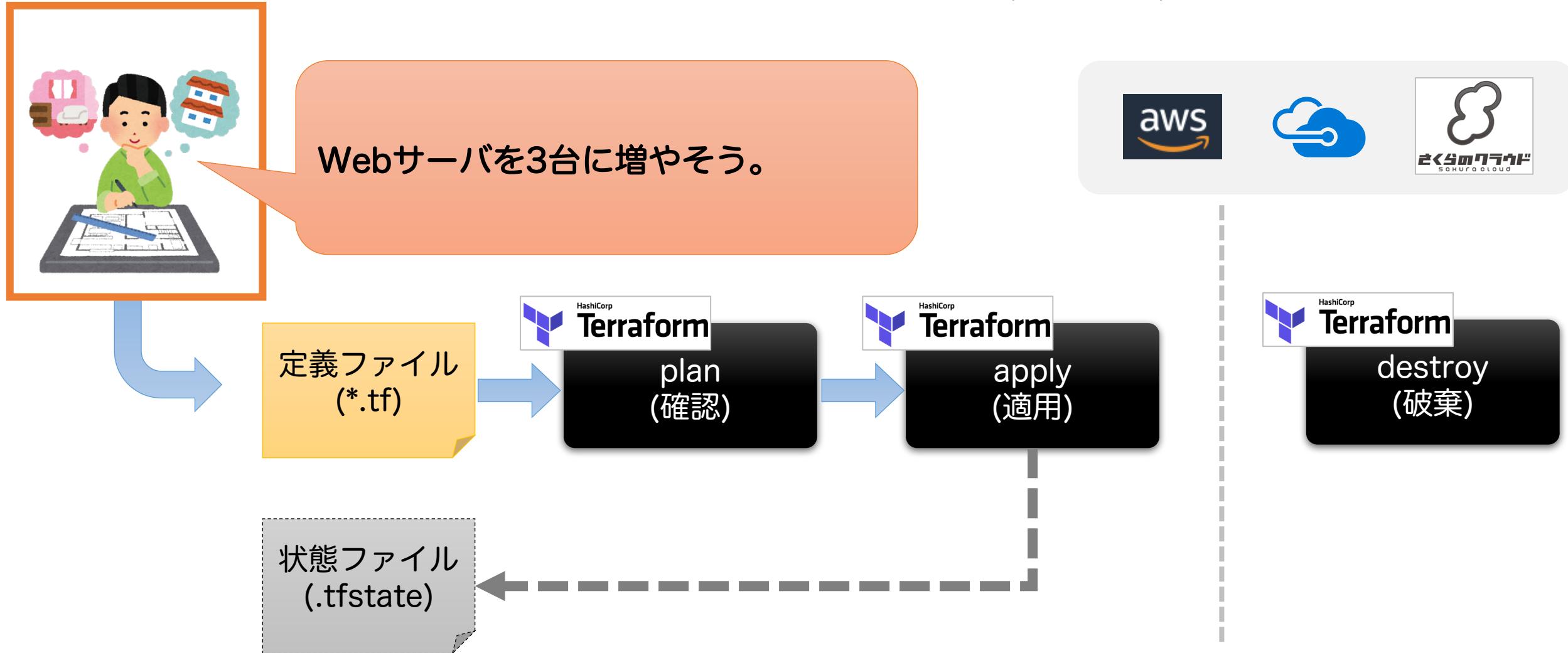
HashiCorp  
**Terraform**

destroy  
(破棄)

状態ファイル  
(.tfstate)

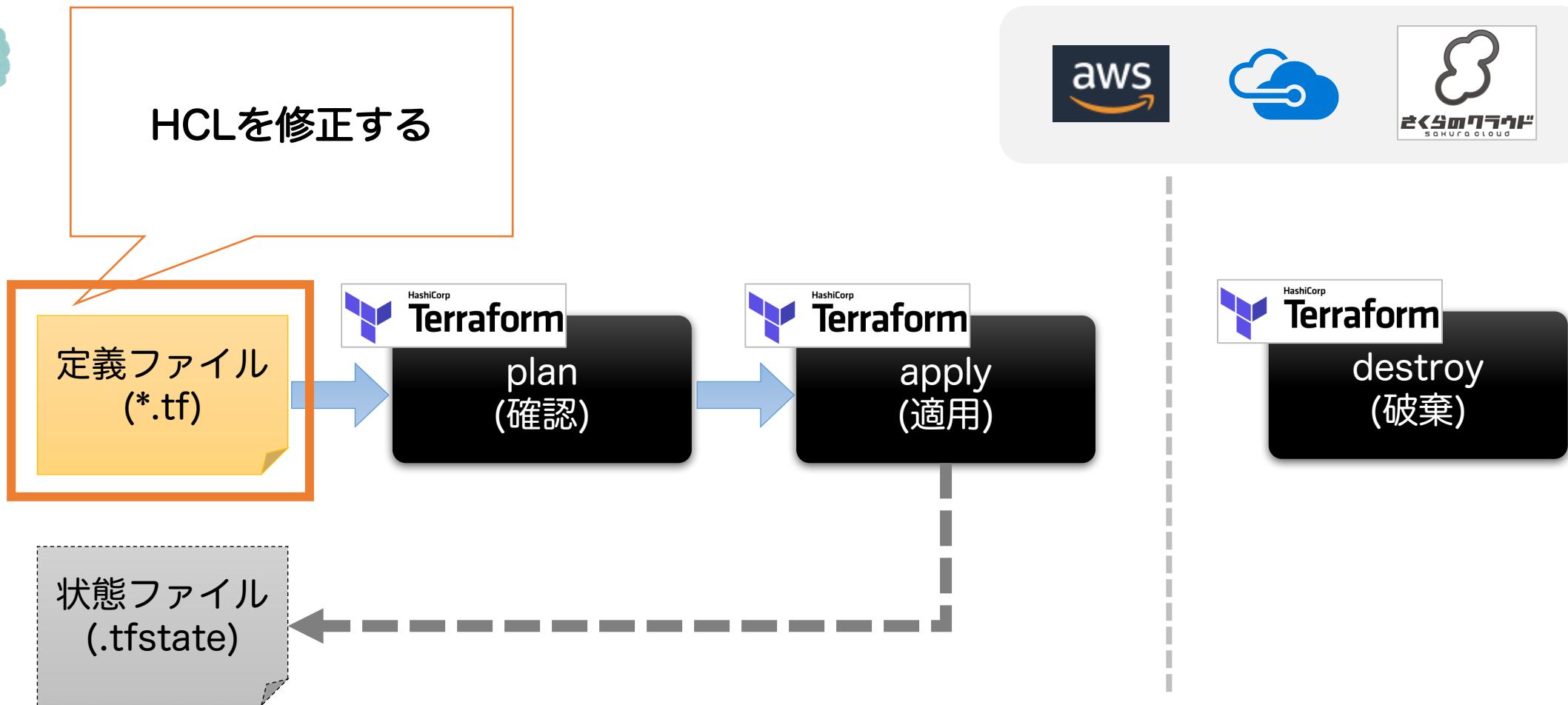


# インフラ構築の流れ(更新)



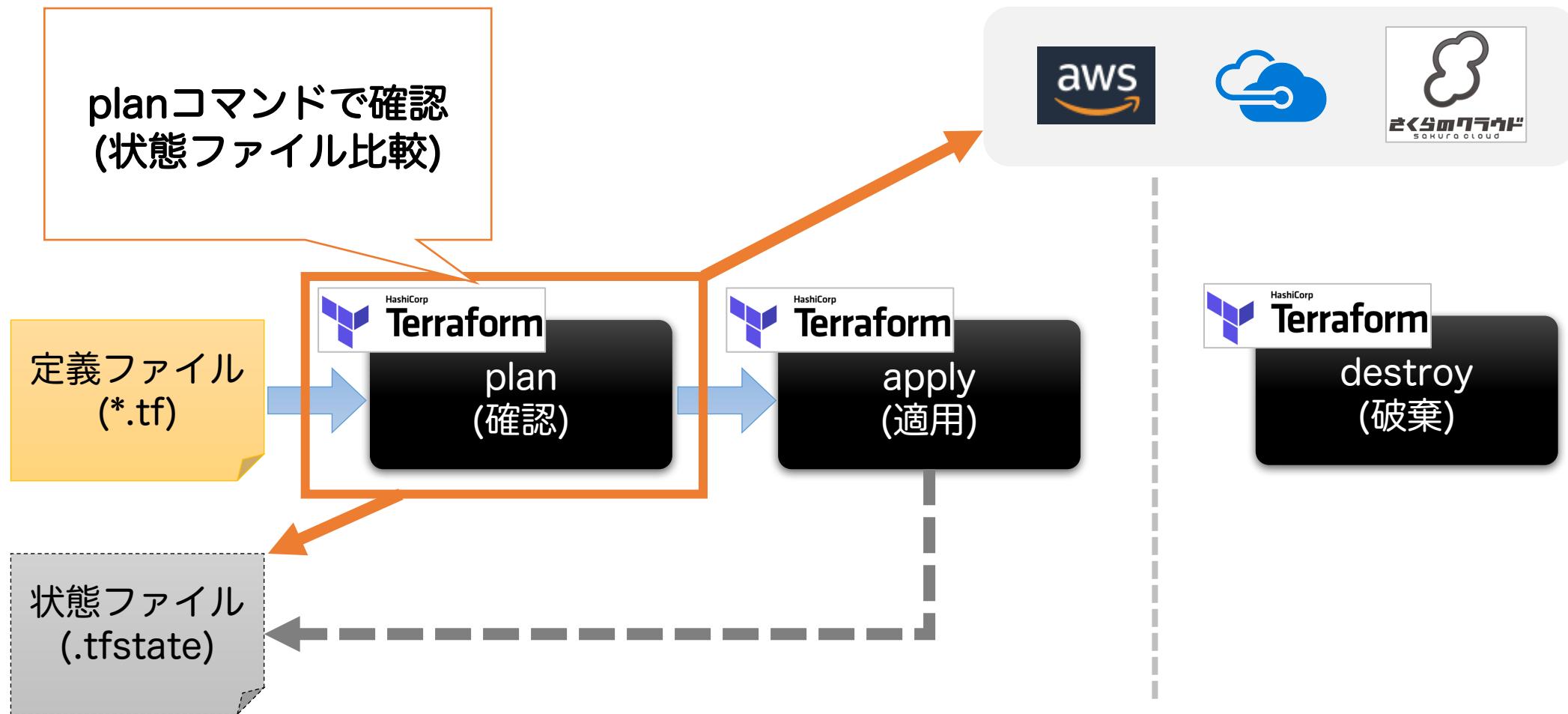


# インフラ構築の流れ(更新)



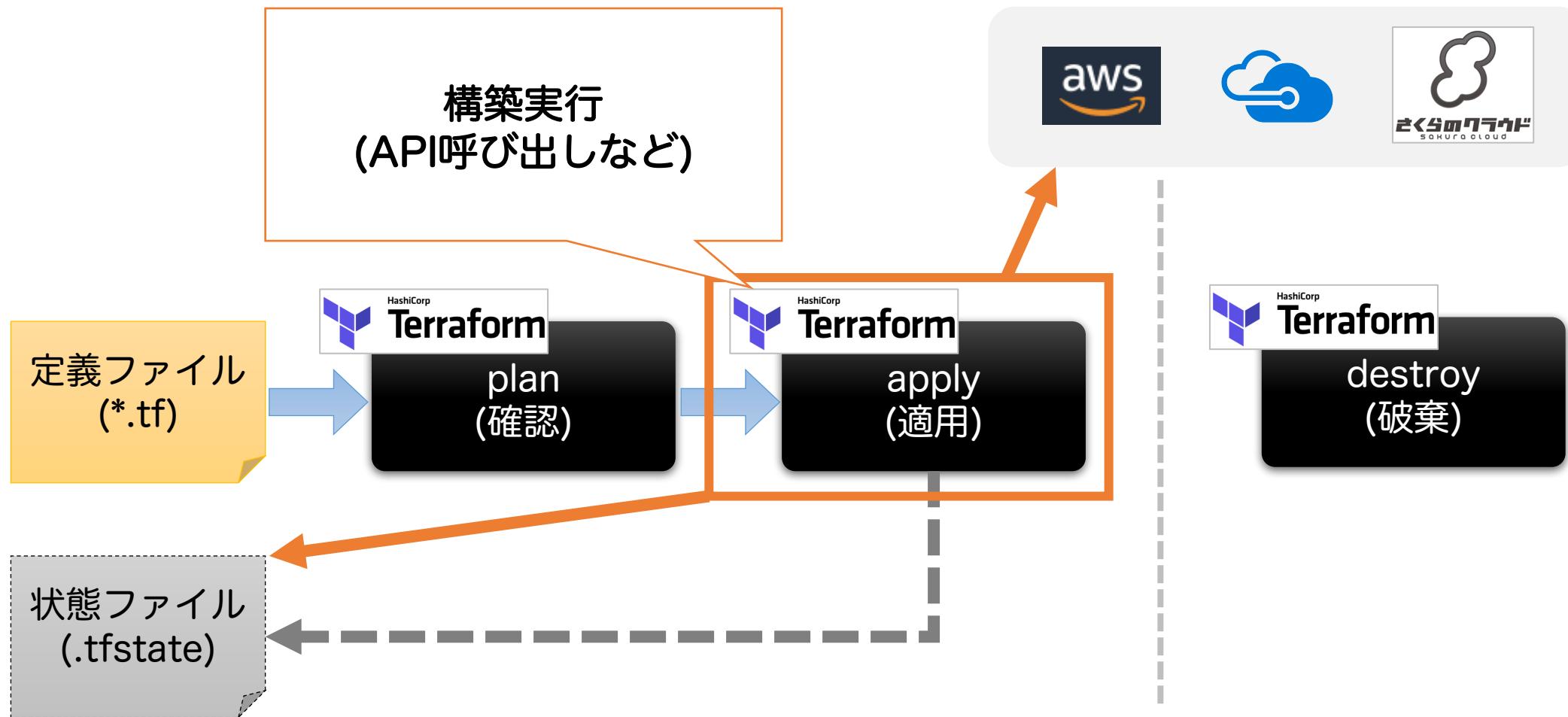
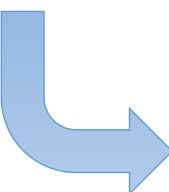


# インフラ構築の流れ(更新)



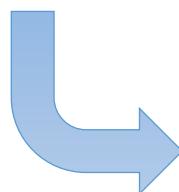


# インフラ構築の流れ(更新)





# インフラ構築の流れ(更新)



今のインフラの状態を  
ファイル(など)で保持しておく

定義  
(\*.tf)



apply  
(適用)

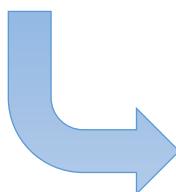
状態ファイル  
(.tfstate)



destroy  
(破棄)



# インフラ構築の流れ(破棄)

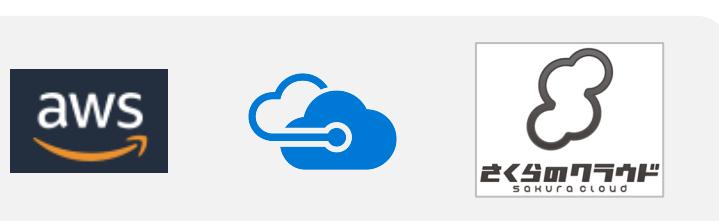


インフラの破棄  
(状態ファイルのクリア)

定義ファイル  
(\*.tf)

HashiCorp  
**Terraform**  
plan  
(確認)

HashiCorp  
**Terraform**  
apply  
(適用)



HashiCorp  
**Terraform**  
destroy  
(破棄)

状態ファイル  
(.tfstate)



HashiCorp  
**Terraform**

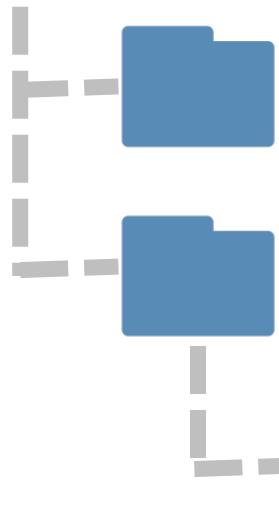
**基本的な使い方**



# 定義ファイルのディレクトリ構成



定義ファイルは環境ごとに  
ディレクトリが分かれます



プロジェクトA

プロジェクトB

プロジェクトC



サーバの定義.tf



スイッチの定義.tf



xxxの定義.tf



.terraform  
(管理用メタデータ)



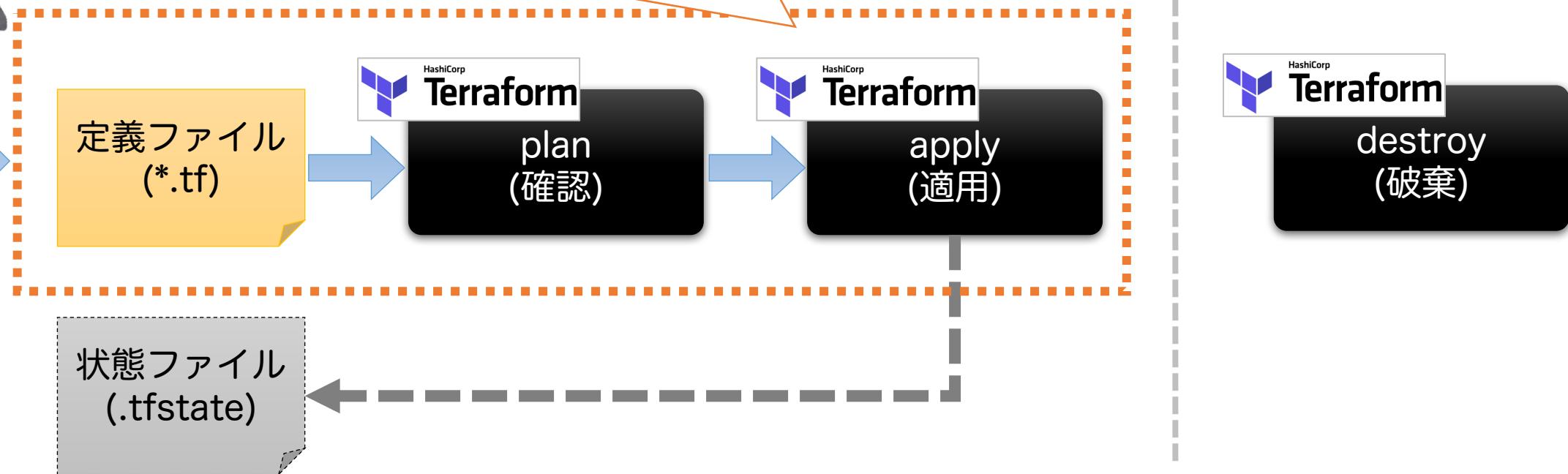
terraform.tfstate  
(状態ファイル)



# 定義ファイルの作成～適用まで



もう少し詳しく見ると、、





# 定義ファイルの作成～適用まで



定義ファイル  
(\*.tf)

HashiCorp  
**Terraform**  
init  
(初期化)

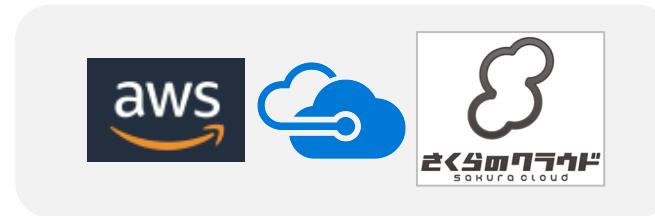
HashiCorp  
**Terraform**  
validate  
(検証)

HashiCorp  
**Terraform**  
plan  
(確認)

HashiCorp  
**Terraform**  
apply  
(適用)

※基本的に初回のみでOK

HashiCorp  
**Terraform**  
fmt  
(整形)





# 定義ファイルの作成～適用まで



- ・tfファイルを作成
- ・必要に応じてfmtで整形する

定義ファイル  
(\*.tf)

※基本的に初回のみでOK

HashiCorp  
**Terraform**  
init  
(初期化)

HashiCorp  
**Terraform**  
validate  
(検証)

HashiCorp  
**Terraform**  
plan  
(確認)

HashiCorp  
**Terraform**  
apply  
(適用)

fmt  
(整形)

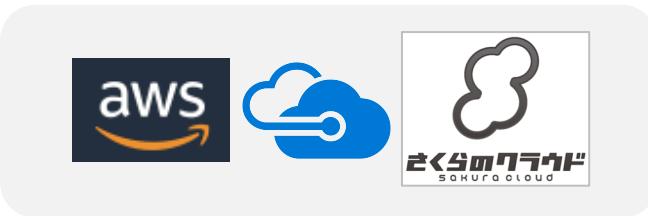




# 定義ファイルの作成～適用まで



定義ファイル  
(\*.tf)



- tfファイルを解析
- 必要なプラグインをダウンロード
- 管理用メタデータ(.terraform)作成



# 定義ファイルの作成～適用まで



定義ファイル (\*.tf)

HashiCorp  
**Terraform**  
init  
(初期化)

HashiCorp  
**Terraform**  
validate  
(検証)

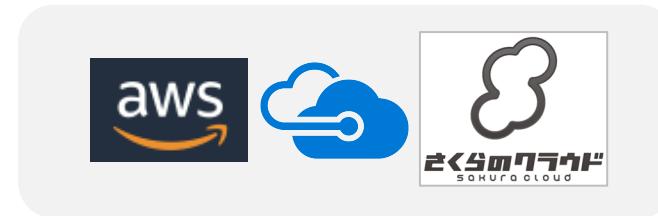
HashiCorp  
**Terraform**  
plan  
(確認)

HashiCorp  
**Terraform**  
apply  
(適用)

※基本的に初回のみでOK

- tfファイルの構文を検証
- 省略してもOK(planでも検証される)
- planより素早く実行可能

HashiCorp  
**Terraform**  
fmt  
(整形)





# 定義ファイルの作成～適用まで



定義ファイル  
(\*.tf)

HashiCorp  
**Terraform**  
init  
(初期化)

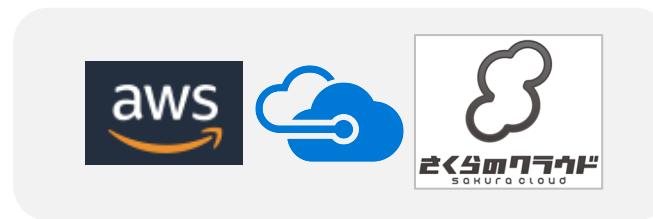
HashiCorp  
**Terraform**  
validate  
(検証)

HashiCorp  
**Terraform**  
plan  
(確認)

HashiCorp  
**Terraform**  
apply  
(適用)

※基本的に初回のみでOK

HashiCorp  
**Terraform**  
fmt  
(整形)





HashiCorp

**Terraform**

ハンズオン



# サーバ1台のシンプルな構成



まずコントロールパネルからサーバを作成してみます。

その後、同じ構成をTerraformで構築します。



# [実習]コントロールパネルからサーバ作成



サーバ作成画面から以下のオーダーで作成



## <オーダー>

- 1CPU/1GBメモリ
- ディスクは20GB(SSD)
- OSはCentOS7
- インターネットに接続する
- パスワードは各自で設定
- ホスト名は「handson-00」
- 名前は「handson-00」



## [実習]Terraformからサーバ作成



- ① 作業用ディレクトリ「handson01」を作成
- ② シェルで①のディレクトリへ移動しておく
- ③ 次ページを参考にtfファイルを作成  
(パスワードは各自で変更してください)
- ④ terraformコマンド実行  
( fmt / init / validate / plan / apply )



# [実習]コピペ用tfファイル

サーバ

```
# サーバの定義
resource sakuracloud_server "server" {
  name = "handson01"

  # ディスクとの接続
  disks = ["${sakuracloud_disk.disk.id}"]
}
```

ディスク

```
# ディスクの定義
resource sakuracloud_disk "disk" {
  name = "handson01"

  # パブリックアーカイブ(OSテンプレート)を指定
  source_archive_id = "${data.sakuracloud_archive.centos.id}"

  # パスワード
  password = "PUT_YOUR_PASSWORD_HERE"

  # ホスト名
  hostname = "handson01"

  # size = 20 #サイズ
  # plan = ssd #プラン
}
```

アーカイブ(データソース)

```
# データソース(アーカイブ)の定義
data sakuracloud_archive "centos" {
  os_type = "centos"
}
```



# [解説]Terraformからサーバ作成

HCLではコメントが使えます(# や /\* \*/)



サーバ

```
# サーバの定義(ここがコメント)
resource sakuracloud_server "server" {
  name = "handson01"

# ディスクとの接続(ここがコメント)
  disks = ["${sakuracloud_disk.disk.id}"]
}
```



# [解説]Terraformからサーバ作成

リソースは以下のように定義します



```
resource リソースタイプ リソース名 {  
    属性名1 = 属性の値  
    属性名2 = ["配列も", "使えるよ"]  
}
```

リソース名 = IDです。変数名みたいなものです。  
”\${}”記法でリソースの値を参照する際に使います



# [解説]Terraformからサーバ作成

“\${}”という記法で値の参照や関数の利用



サーバ

```
resource sakuracloud_server "server" {
  name = "handson01"

  # ディスクとの接続
  disks = "${sakuracloud_disk.disk.id}"
}
```

- `sakuracloud_disk`というリソースの中の
- `disk`という名前のリソースの
- 属性”`id`”の値を参照



# [解説]Terraformからサーバ作成

データソース = 読み取り専用リソース



アーカイブ(データソース)

```
# データソース(アーカイブ)の定義  
data sakuracloud_archive "centos" {  
    os_type = "centos" ← この条件で検索する  
}
```

- クラウド上にリソースを作るのでなく、読み取る(検索する)ためのリソース
- 読み取った値は"\${}"記法で参照することが可能



## [解説]Terraformからサーバ作成

利用できるリソースはドキュメントに記載されています。

<https://sacloud.github.io/terraform-provider-sakuracloud/>



## スケールアップ



Terraformで作成したサーバのスペックを上げます  
(スケールアップ)



## [実習]スケールアップ

サーバ定義を編集し、2CPU、4GBメモリにしてください



どう変えれば良いかは  
以下のドキュメントを参考に考えてみてください。

サーバリソースのドキュメント:

<https://sacloud.github.io/terraform-provider-sakuracloud/configuration/resources/server/>



# 次のハンズオンに進む前に



次のハンズオン「プロビジョニング」の実施前に  
「**terraform destroy**」を実行しておいてください。



# プロビジョニング



Terraformで作成したサーバをプロビジョニングします。

今回はApache(httppd)をインストールします。



## [実習] プロビジョニング

サーバの定義の中に以下のブロックを追加してください。



サーバ

```
connection {  
    host    = "${self.ipaddress}"  
    user    = "root"  
    password = "${sakuracloud_disk.disk.password}"  
}  
  
provisioner "remote-exec" {  
    inline = [  
        "", #ここに処理を書く(実行したいスクリプト1行をリストの1要素として記述)  
    ]  
}
```



## [実習] プロビジョニング

### Apacheをインストールするスクリプト

```
# httpdのインストール  
yum install -y httpd  
  
# 確認用ページ  
echo 'This is a TestPage!!' >> /var/www/html/index.html  
  
# サービス起動設定  
systemctl enable httpd.service  
systemctl start httpd.service  
  
# ファイアウォール設定  
firewall-cmd --add-service=http --zone=public --permanent  
firewall-cmd --reload
```



## 動作確認



<http://サーバのIPアドレス/>  
をブラウザで開いてみましょう

```
# サーバのIPアドレスを確認  
$ usacloud server list
```



## [付録]スイッチでのローカルネット構築



Terraformで作成したサーバにNICを追加し、  
スイッチにつないでみましょう。



# [実習]スイッチでのローカルネット構築



NICの追加 + スイッチとの接続は以下のように書きます。

```
#追加NIC  
additional_nics = ["スイッチのID"]
```

スイッチの定義は  
以下のドキュメントを参照に考えてみてください。

スイッチリソースのドキュメント:

<https://sacloud.github.io/terraform-provider-sakuracloud/configuration/resources/switch/>



HashiCorp  
**Terraform**

應用編



# 大量のWebサイトをシンプル監視で監視



例えば100サイト監視する場合、  
100サイト分の定義を書かなきゃいけないの？



# 大量のWebサイトをシンプル監視で監視



URLを変数に切り出し、  
監視先URLを柔軟に追加/削除可能にします。



# 大量のWebサイトをシンプル監視で監視

シンプル監視

```
resource sakuracloud_simple_monitor "monitor" {
  # 監視先は変数に切り出し
  target = "${var.monitor_targets[count.index]}"

  health_check = {
    protocol = "http"
    status   = "200"
    path     = "/"
  }

  # 変数に定義された監視先の数だけシンプル監視を定義
  count = "${length(var.monitor_targets)}"
}

# 変数の宣言
variable "monitor_targets" {
  type = "list"
}
```

変数ファイル  
(`terraform.tfvars`)

```
monitor_targets = [
  "google.co.jp",
  "sakura.ad.jp",
]
```

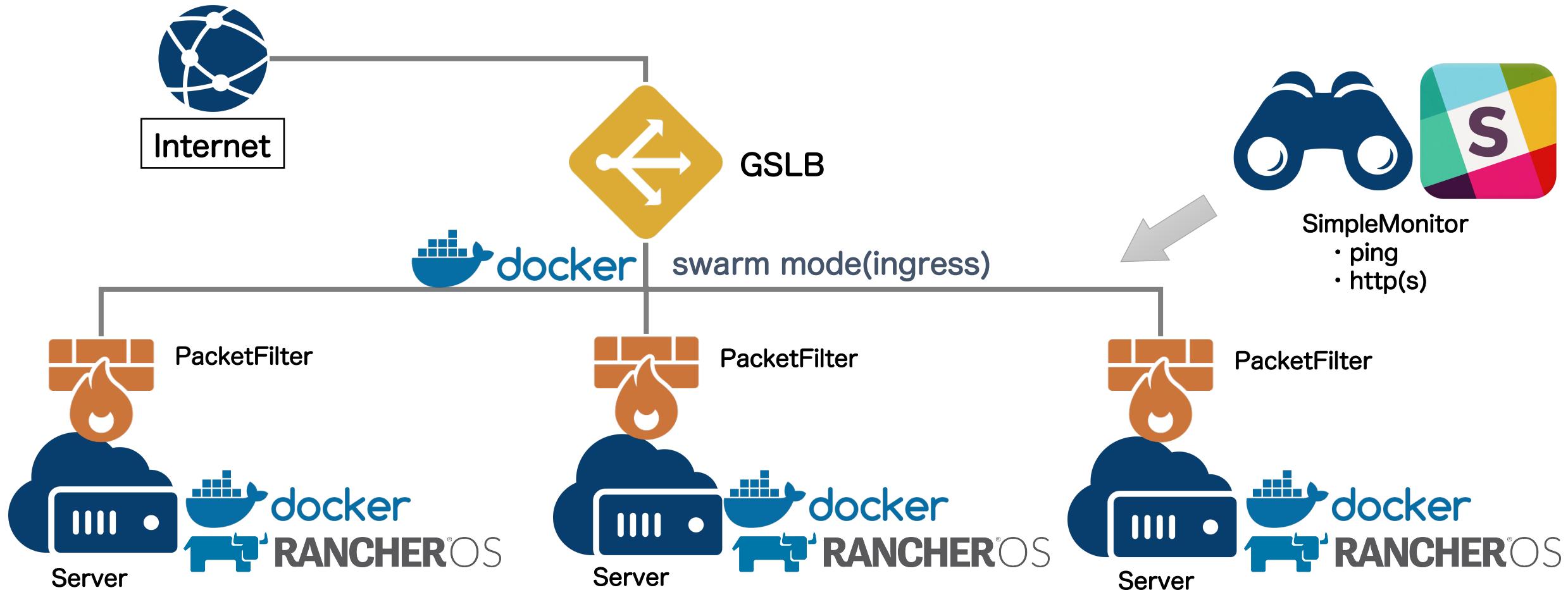
監視先を増やす時は  
このファイルに追加するだけでOK



HashiCorp  
**Terraform**



# usacloud配布サイト(usacloud.jp)



近日中にtfファイル一式を公開予定

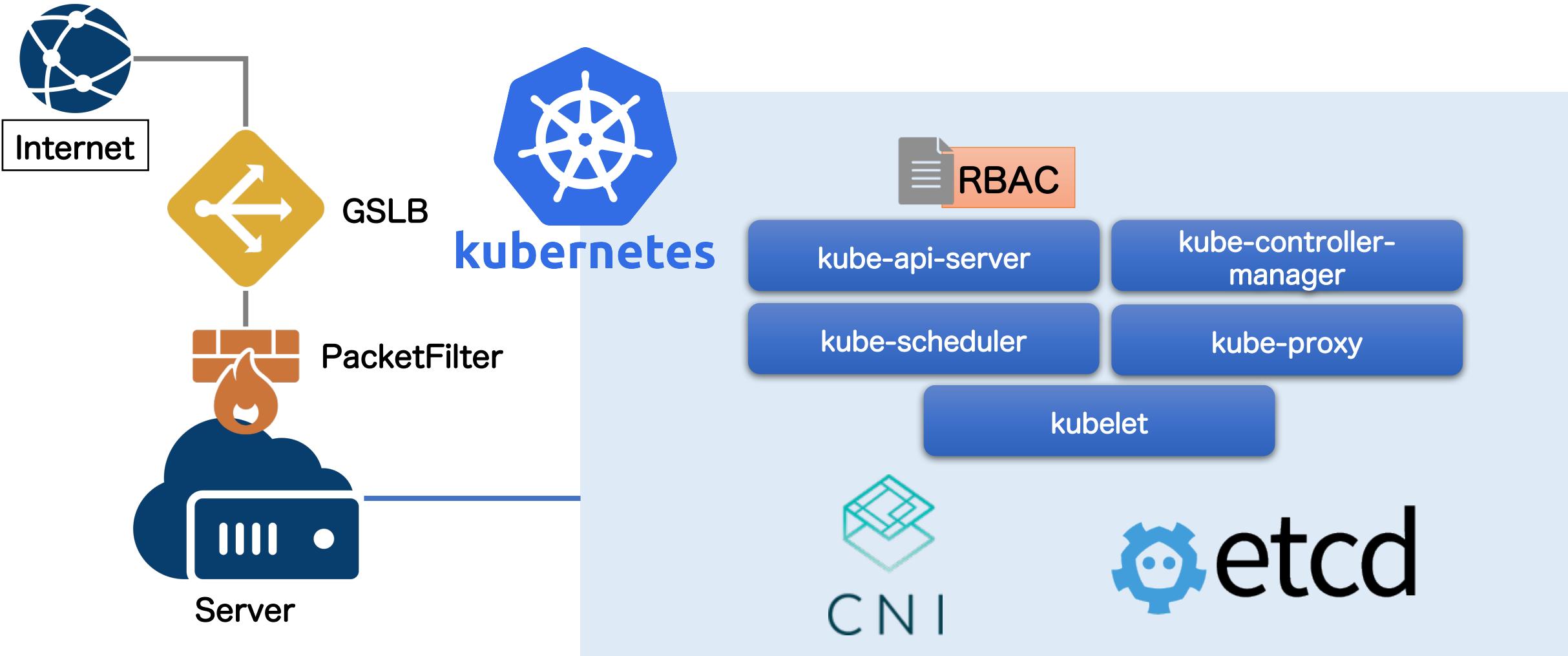


HashiCorp  
**Terraform**



さくらのクラウド  
SAKURA CLOUD

# kubernetes環境を1発構築



<https://github.com/yamamoto-febc/kube-sacloud-template>



HashiCorp

**Terraform** for



<https://github.com/sacloud/terraform-provider-sakuracloud/>

**Enjoy!!**