

僕は Terraform と Serverless  
Framework の相互連携したいんだ



基本は Terraform でインフラの構築

ですが.....

サーバーレスアーキテクチャの場合は  
Serverless Framework で構築

こんなことってありますよね？

そもそもなんだけど全部 Terraform  
で良くない？

# Terraform と Serverless Framework を比べる

API Gateway, AWS Lambda 構成の簡単なサーバーレスアーキテクチャで考えます 🤔

## Terraform のドキュメントより

- Deploy Serverless Applications with AWS Lambda and API Gateway

## Serverless Framework のドキュメントより

- [Hello World Node.js Example](#)
- [Serverless Framework - AWS Lambda Events - API Gateway](#)

# 圧倒的記述量の差



# Serverless Framework で嬉しい事

まったく、ローカルエミュレートは最高だぜ！！





# Serverless Framework で嫌な事

- 短い記述量でいろいろと勝手に作ってくれる（何が作られるかわからない）
- お前、最終的に AWS CloudFormation やんけ！
- Serverless Framework が作った AWS CloudFormation を読む必要が出てくる

# 意外に出来るぞ Serverless Framework !

The screenshot shows the 'AWS Provider Documentation' page on the Serverless Framework website. The page has a dark header with the 'serverless' logo and navigation links for 'Products', 'Docs', 'Pricing', and 'Company'. A 'Contact' link is also visible in the top right. On the left, a sidebar lists navigation options: 'Overview', 'Get Started', 'Provider References', and a list of providers including AWS, Azure, Tencent, Google, Knative, Alibaba Cloud, Cloudflare Workers, Fn, Kubeless, OpenWhisk, and Spotinst. The 'AWS' provider is selected, and its 'Overview' sub-page is active. The main content area is titled 'AWS Provider Documentation' and contains three sections: 'Guides', 'CLI references', and 'Events'. Each section lists various topics and links in red text.

serverless Products Docs Pricing Company Contact

Overview  
Get Started  
Provider References

Overview  
AWS  
CLI Reference  
AWS Events  
Examples  
User Guide  
Azure  
Tencent  
Google  
Knative  
Alibaba Cloud  
Cloudflare Workers  
Fn  
Kubeless  
OpenWhisk  
Spotinst

## AWS Provider Documentation

### Guides

- Intro
- Functions
- Deploying
- Packaging
- Workflow
- Credentials
- Events
- Testing
- IAM
- Serverless.yml
- Services
- Resources
- Variables
- Plugins

### CLI references

- Config Credentials
- Package
- Deploy List
- Logs
- Info
- Remove
- Plugin Install
- Serverless Stats
- Create
- Deploy
- Invoke
- Login
- Rollback
- Plugin List
- Plugin Uninstall
- Install
- Deploy Function
- Invoke Local
- Metrics
- Rollback Function
- Plugin Search
- Print

### Events

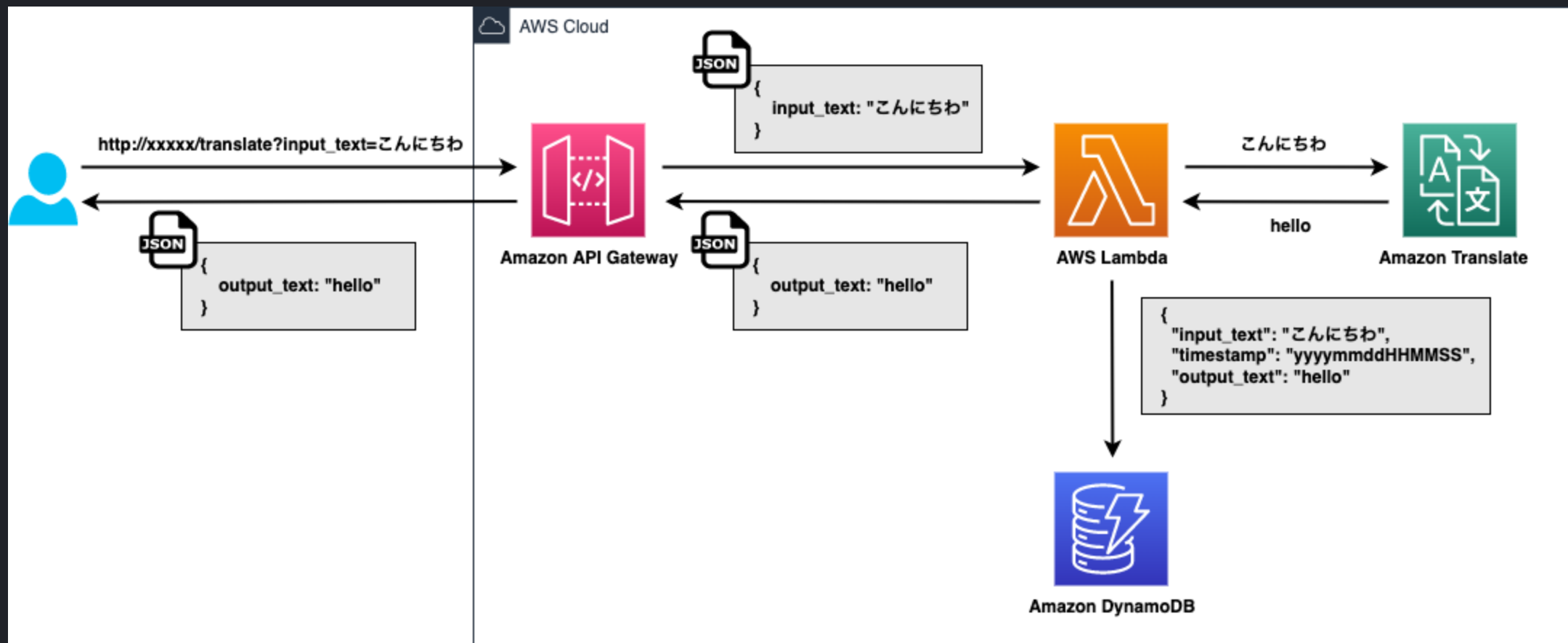
- API Gateway
- Streams
- SNS
- Alexa Skill
- CloudWatch Event
- CloudFront
- MSK
- HTTP API
- S3
- SQS
- Alexa Smart Home
- CloudWatch Log
- Cognito User Pool
- IoT Fleet Provisioning
- Websocket
- Schedule
- ALB
- IoT
- EventBridge
- Self Managed Apache Kafka

皆さんはどんな時、Serverless Framework を使いますか？  
もしよければ教えてください！

# ここから今日の本題です！

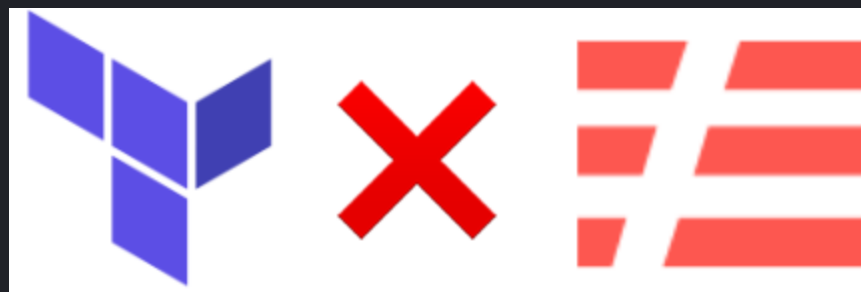
- サーバーレスアーキテクチャ作成した Web API の紹介
- Terraform と Serverless Framework でそれぞれ何を管理させるのか？
- Terraform と Serverless Framework の連携
- Terraform と Serverless Framework を相互連携して思ったこと

# Terraform と Serverless Framework を使って翻訳 Web API を作成してみました



# Web API の実演です

# Terraform と Serverless Framework でそれぞれ何を管理させるのか？



業務でちゃんと使って開発したことがないのでイマイチイメージがわからないんですが、これから紹介することは何も考えずに分けました。

**何をどう分けたらいいか皆さんの意見をもらえたら嬉しいです！**



# Terraform 管理化

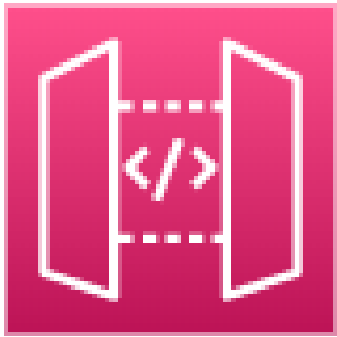


IAM ロール



AWS Systems Manager  
Parameter Store

# Serverless Framework 管理化



**Amazon API Gateway**



**AWS Lambda**



**Amazon DynamoDB**



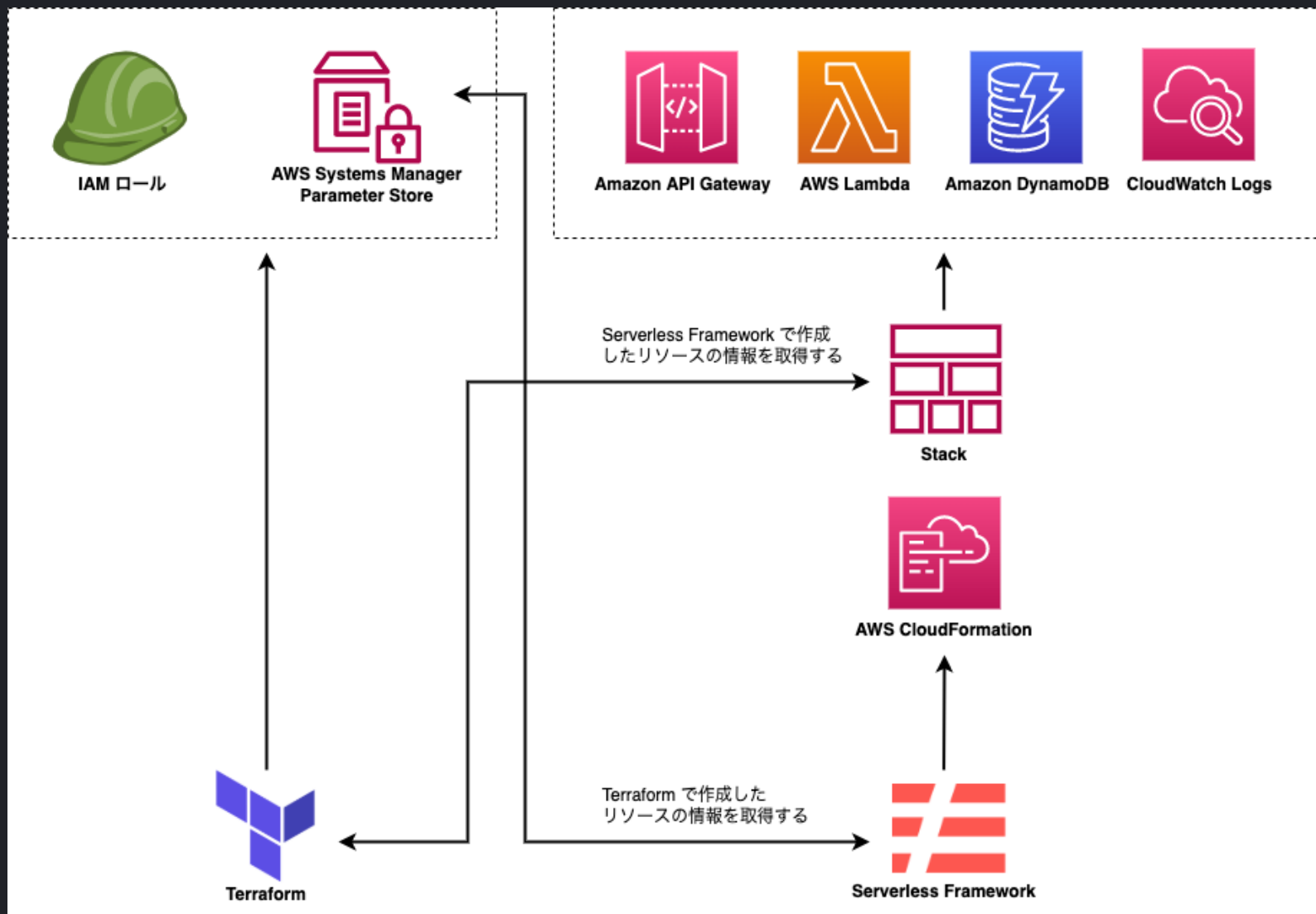
**CloudWatch Logs**

# どうやって連携するの？

連携という以下2つが考えられると思います。

- **Serverless Framework → Terraform**
- **Terraform → Serverless Framework**

# とりあえず全体図



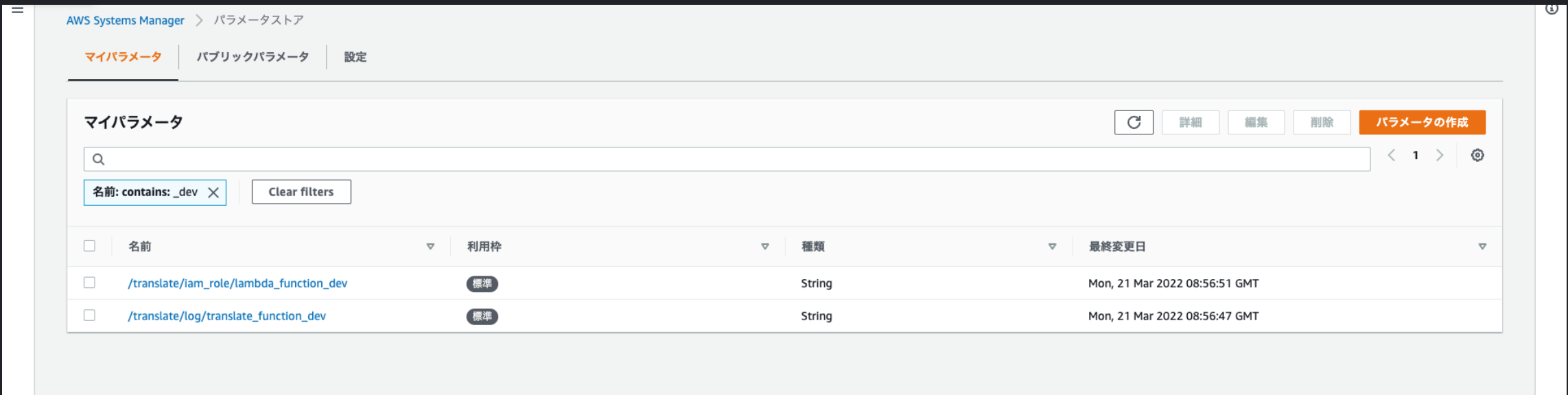
# Serverless Framework → Terraform

Serverless Framework から Terraform で作成されたリソースにアクセスする方法ですが、Serverless Framework の公式のブログに紹介されています。

- ・ [The definitive guide to using Terraform with the Serverless Framework](#)

# Serverless Framework → Terraform

arnなどをパラメータストアにセットしてそれを Serverless Framework で読み込むようにする。



AWS Systems Manager > パラメータストア

マイパラメータ | パブリックパラメータ | 設定

マイパラメータ

検索

名前: contains: \_dev X Clear filters

<input type="checkbox"/>	名前	利用枠	種類	最終変更日
<input type="checkbox"/>	/translate/iam_role/lambda_function_dev	標準	String	Mon, 21 Mar 2022 08:56:51 GMT
<input type="checkbox"/>	/translate/log/translate_function_dev	標準	String	Mon, 21 Mar 2022 08:56:47 GMT

```
custom:  
  iamRoleName: ${ssm:/translate/iam_role/lambda_function_${sls:stage}}
```

# Serverless Framework → Terraform

IAM ポリシーを作る時、リソースで絞り込みたいため Terraform で CloudWatch Logs を作り参照しようと思ったが.....

translate-function-dev-policy の編集

ポリシーにより、ユーザー、グループ、またはロールに割り当てることができる AWS アクセス権限が定義されます。ビジュアルエディタで JSON を使用してポリシーを作成または編集できます。詳細はこちら

ビジュアルエディタ JSON 管理ポリシーのインポート

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "",
6       "Effect": "Allow",
7       "Action": "logs:CreateLogGroup",
8       "Resource": "arn:aws:logs:ap-northeast-1:300367504550:*"
9     },
10    {
11      "Sid": "",
12      "Effect": "Allow",
13      "Action": [
14        "logs:PutLogEvents",
15        "logs:CreateLogStream"
16      ],
17      "Resource": "arn:aws:logs:ap-northeast-1:[REDACTED]:log-group:/aws/lambda/translate-function-dev:*"
18    }
19  ]
20 }
```

セキュリティ: 0 エラー: 0 警告: 0 提案: 1

# Serverless Framework → Terraform

Serverless Framework で使用する CloudWatch Logs を arn で指定する方法がなかったため、同じ名前にしたらエラーになってしまったりと **CloudFormation** 力が足りなくて 結局やめてしまった。

Terraform で locals で LogGroupName を管理させ、パラメータストア経由で LogGroupName を Serverless Framework では参照するようにした。



# Serverless Framework → Terraform

## Terraform

```
locals {  
  log_group_name = format("/aws/lambda/translate-function-%s", terraform.workspace)  
}
```

## Serverless Framework

```
custom:  
  logGroupName: ${ssm:/translate/log/translate_function_${sls:stage}}  
  
resources:  
  extensions:  
    TranslateLogGroup:  
      Properties:  
        LogGroupName: ${self:custom.logGroupName}  
        RetentionInDays: 30
```

# Terraform → Serverless Framework

CloudFormation の Stack が出力した値を Terraform 側で読み込むことで Serverless Framework が作成したリソースにアクセスすることができるようになります。

# Terraform → Serverless Framework

CloudFormation > スタック > translate-function-dev

translate-function-dev

削除 更新 スタックアクション ▼ スタックの作成 ▼

スタックの情報 イベント リソース 出力 パラメータ テンプレート 変更セット

出力 (6)

検索結果の出力

キー	値	説明	エクスポート名
LambdaFunctionArn	arn:aws:lambda:ap-northeast-1:██████████:function:translate-function-dev	-	lambda-function-arn-dev
LambdaFunctionName	translate-function-dev	-	lambda-function-name-dev
ServerlessDeploymentBucketName	translate-sls	-	sls-translate-function-dev-ServerlessDeploymentBucketName
ServiceEndpoint	<a href="https://██████████.ap-northeast-1.amazonaws.com/dev">https://██████████.ap-northeast-1.amazonaws.com/dev</a>	URL of the service endpoint	sls-translate-function-dev-ServiceEndpoint
TranslateDashfunctionDashdevDashpythonDashdefaultLambdaLayerQualifiedArn	arn:aws:lambda:ap-northeast-1:██████████:layer:translate-function-dev-python-default:8	-	TranslateDashfunctionDashdevDashpythonDashdefaultLambdaLayerQualifiedArn
TranslateLambdaFunctionQualifiedArn	arn:aws:lambda:ap-northeast-1:██████████:function:translate-function-dev:28	Current Lambda function version	sls-translate-function-dev-TranslateLambdaFunctionQualifiedArn

# Terraform → Serverless Framework

## CloudFormation Stack

TranslateLambdaFunctionQualifiedArn	arn:aws:lambda:ap-northeast-1: [REDACTED] :function:translate-function-dev:28	Current Lambda function version	sls-translate-function-dev-TranslateLambdaFunctionQualifiedArn
-------------------------------------	---	---------------------------------	--

## Terraform

```
data "aws_cloudformation_export" "lambda_function_qualified_arn" {
  name = format("sls-translate-function-%s-TranslateLambdaFunctionQualifiedArn", terraform.workspace)
}
```

相互連携をしてみても思ったこと 🤔

相互連携させる時に

重要なのは **デプロイ順** である！



現状だと Serverless Framework は Terraform に必ず依存している状態になっています。

バッチなどにおいては冪等性（べきとうせい）の考え方はすごく重要だと思っていますが、インフラにおいてはどうなのでしょう？

終わり