

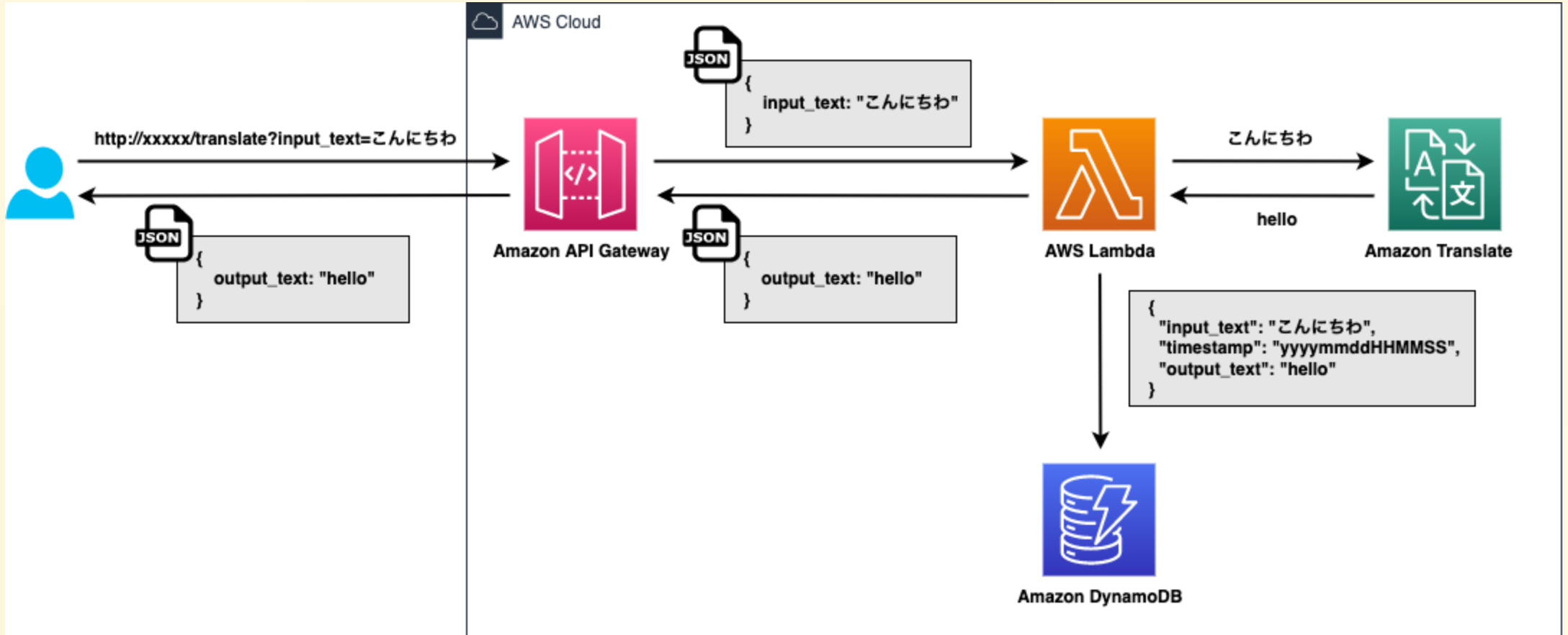
サーバーレスアーキテクチャで翻訳 Web API を作成した

今日話すこと

- Web API の全体構成
- AWS ハンズオン資料
- Serverless Framework について
- ローカル環境について
- 作成 Web API の実演
- 開発方法について
- なぜ Docker で開発をするのか？

全部でスライドが38枚です

こんな感じの構成です



実はこれ.....

AWS ハンズオン資料をパクって作成しました（笑）

サーバーレスアーキテクチャで翻訳 Web API を構築する »

AWS Hands-on for Beginners

～オリジナルAPIの作成を通してサーバーレスの基本を学ぼう～



翻訳 Web API の構築を通し、サーバーレスアーキテクチャの基本が学びます。主に扱う AWS サービスは AWS Lambda、Amazon API Gateway、Amazon DynamoDB です。

[AWS Hands-on for Beginners ～ Serverless 編 ～](#)

ハンズオン資料の良いところ

- 10個の動画しかない
- 1つの動画が11分以内で終わる
- サービスごとの説明から作成まで行う
- 一般的なサーバーレスアーキテクチャである
- 自分が実際にやった時は2時間もかからず終わった

サーバーレスアーキテクチャの触りとしてはとても良い！！！！

ハンズオン資料の悪いところ

- 本当に触りしかやらない
- Infrastructure as Code (IaC) ではない

残念ながらそのまま業務に使えるわけではない！！！！

今日の発表の本題はここからです

このハンズオンだけだとイケてないよね 🤔

僕らはカッチョ良く開発したいんだ！！！！

とりあえず目的をはっきりとさせます！

今回の目的です

- ローカルにカッチョ良くエミュレートして開発できる環境を整える
- DynamoDB をローカルで使えるようにする
- Serverless Framework を 1 から作成して使ってみる
- 既に作成してあるインフラをインポートしてコードに落とし込む (IaC)
- Terraform と Serverless Framework の連携を可能にする
- Terraform で workspace を使用して環境ごとインフラを作成してみる

今回の目的です

- ローカルにカッチョ良くエミュレートして開発できる環境を整える
- DynamoDB をローカルで使えるようにする
- Serverless Framework を 1 から作成して使ってみる
- 既に作成してあるインフラをインポートしてコードに落とし込む (IaC)
- Terraform と Serverless Framework の連携を可能にする
- Terraform で workspace を使用して環境ごとインフラを作成してみる

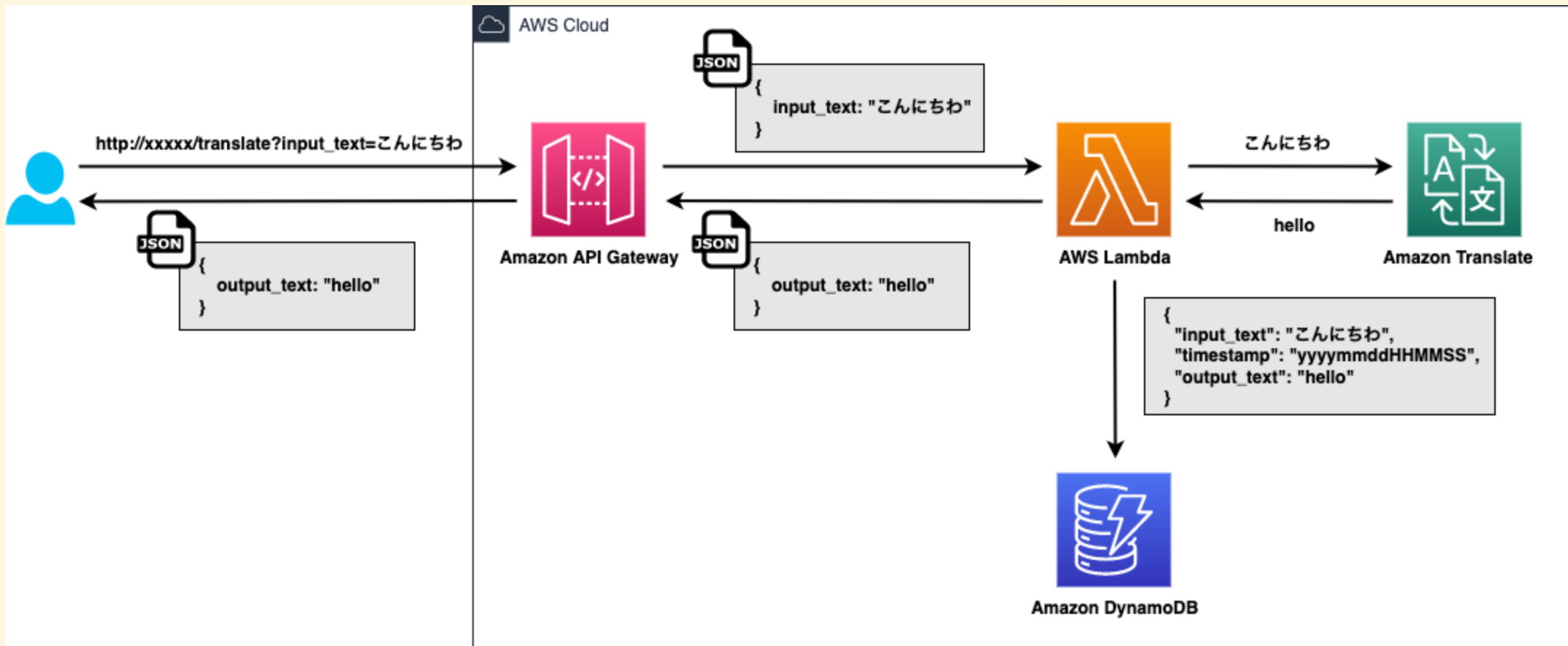
**話の内容の大半は Serverless
Framework のことが中心になります**

Serverless Framework とは？

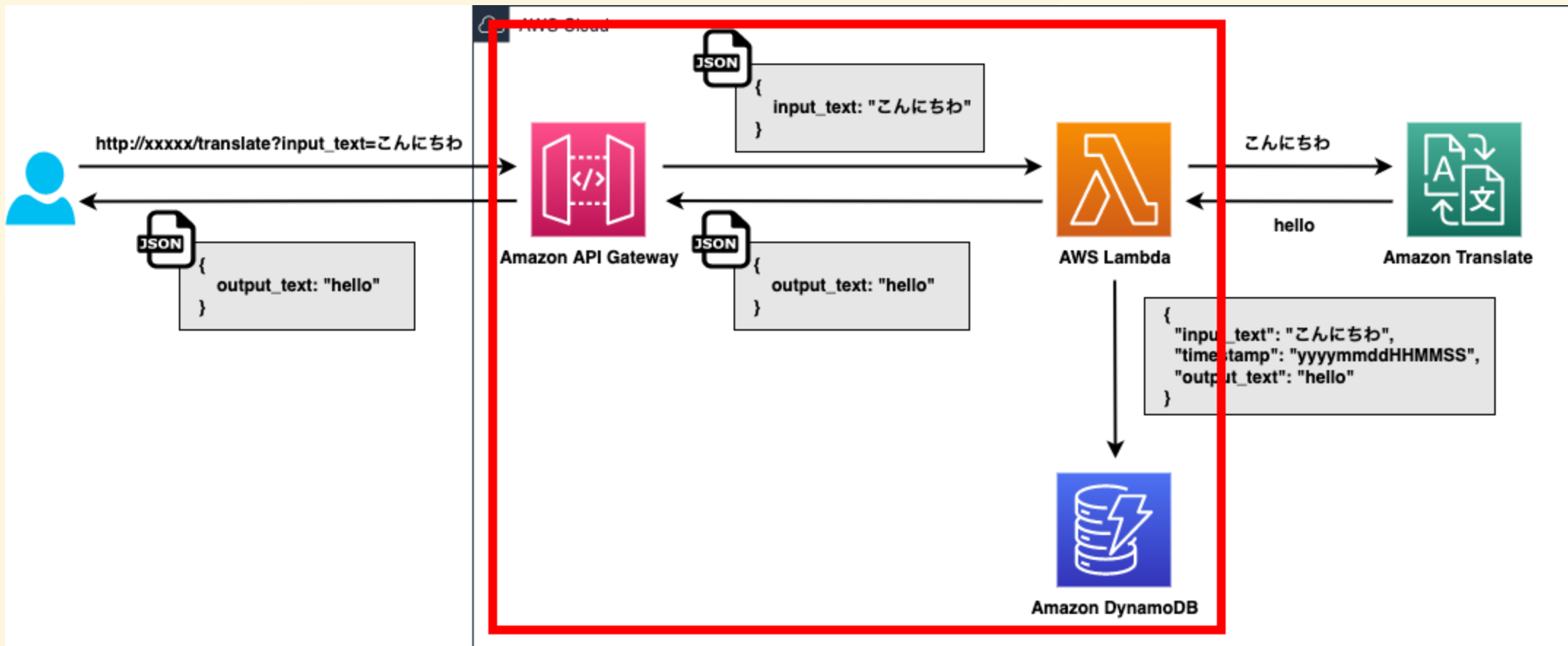
“ All-in-one development & monitoring of auto-scaling apps on AWS Lambda ”

- Lambda の開発に特化している
- Infrastructure as Code (IaC)
- ローカルでサーバーレスアーキテクチャのエミュレートが可能
- デプロイがコマンド 1 つで可能 (CI/CDも完結)
- 環境別のインフラを簡単に作成可能
- 豊富な plugin によって拡張できる

改めておさらい

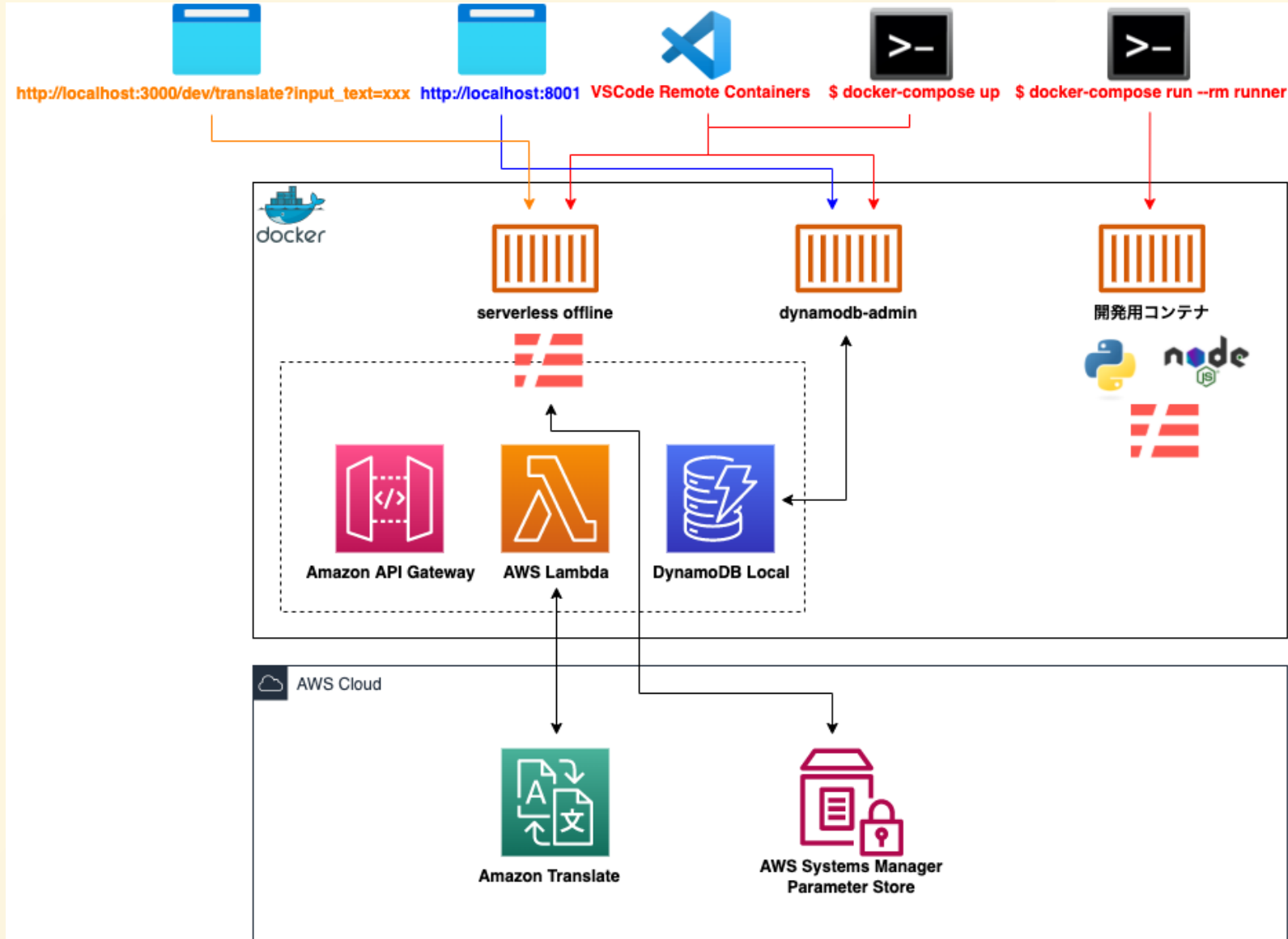


ローカルでエミュレートした箇所

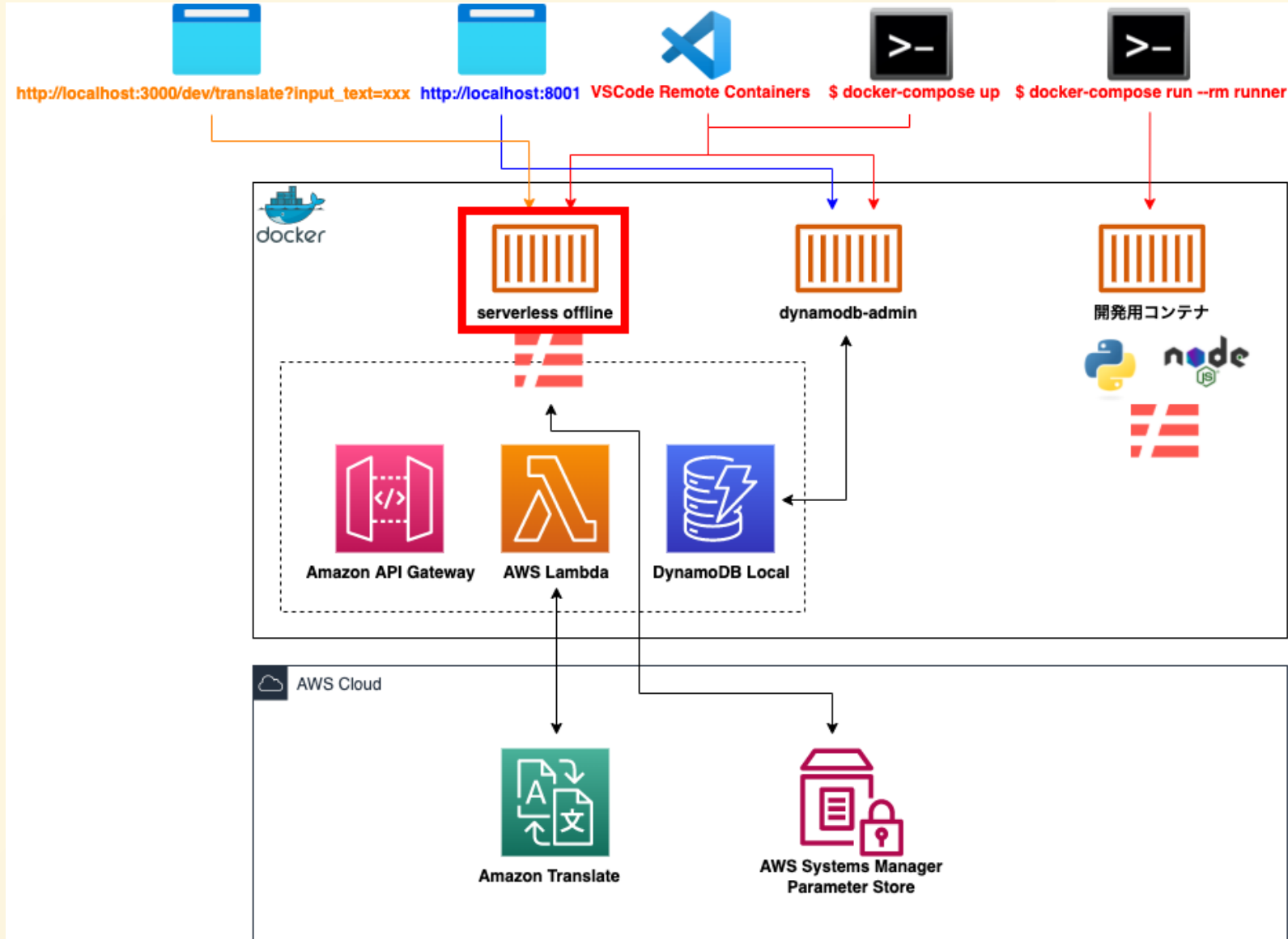


**では、実際にローカル環境を見ていき
ましょう！**


ローカル環境の全体図



serverless offline コンテナ



serverless offline コンテナ

 README.md

Serverless Offline

serverless ⚡

npm v8.5.0

build failing

node >=12.0.0

serverless ^1.60.0 || 2 || 3

code style prettier

license MIT

PRs welcome

gitter join chat

This [Serverless](#) plugin emulates [AWS λ](#) and [API Gateway](#) on your local machine to speed up your development cycles. To do so, it starts an HTTP server that handles the request's lifecycle like APIG does and invokes your handlers.

Features:

- [Node.js](#), [Python](#), [Ruby](#) and [Go](#) λ runtimes.
- Velocity templates support.
- Lazy loading of your handler files.
- And more: integrations, authorizers, proxies, timeouts, responseParameters, HTTPS, CORS, etc...

This plugin is updated by its users, I just do maintenance and ensure that PRs are relevant to the community. In other words, if you [find a bug or want a new feature](#), please help us by becoming one of the [contributors](#) 🙌 ! See the [contributing section](#).

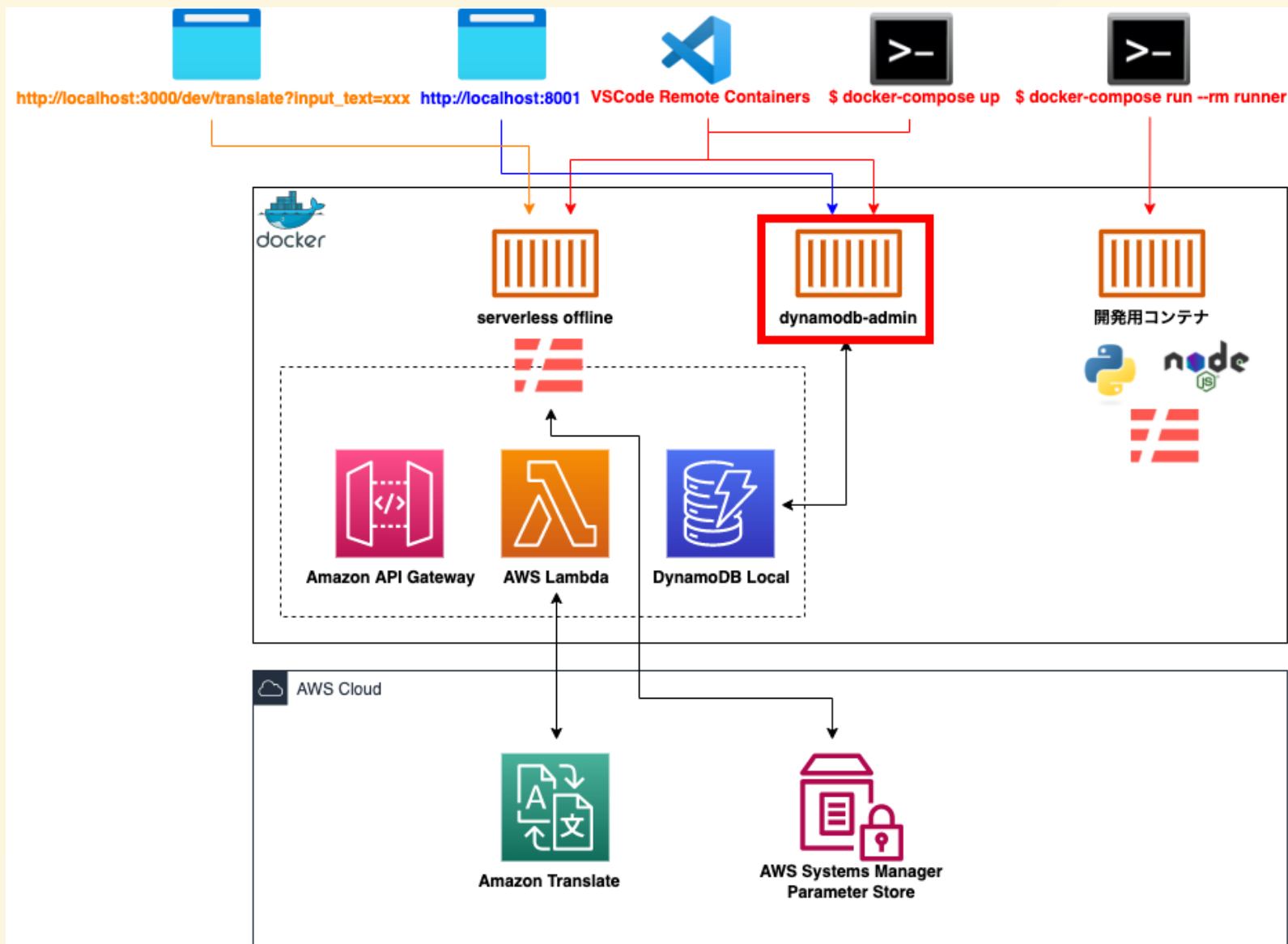
serverless offline コンテナ

- Serverless Framework のプラグインで AWS Lambda と API Gateway をローカルでエミュレートするツール
- 更にプラグインを追加することで AWS Lambda と API Gateway 以外もローカルでエミュレートすることができる

代表的なエミュレート例

- DynamoDB
- AppSync

dynamodb-admin コンテナ



dynamodb-admin コンテナ

Tables		
utahdata-entities	145	Delete
utahdata-entity-payees	122	Delete
utahdata-payee-year-compensation	156	Delete
utahdata-scheduled-imports	780	Delete

Create table

実際に見てみましょう！

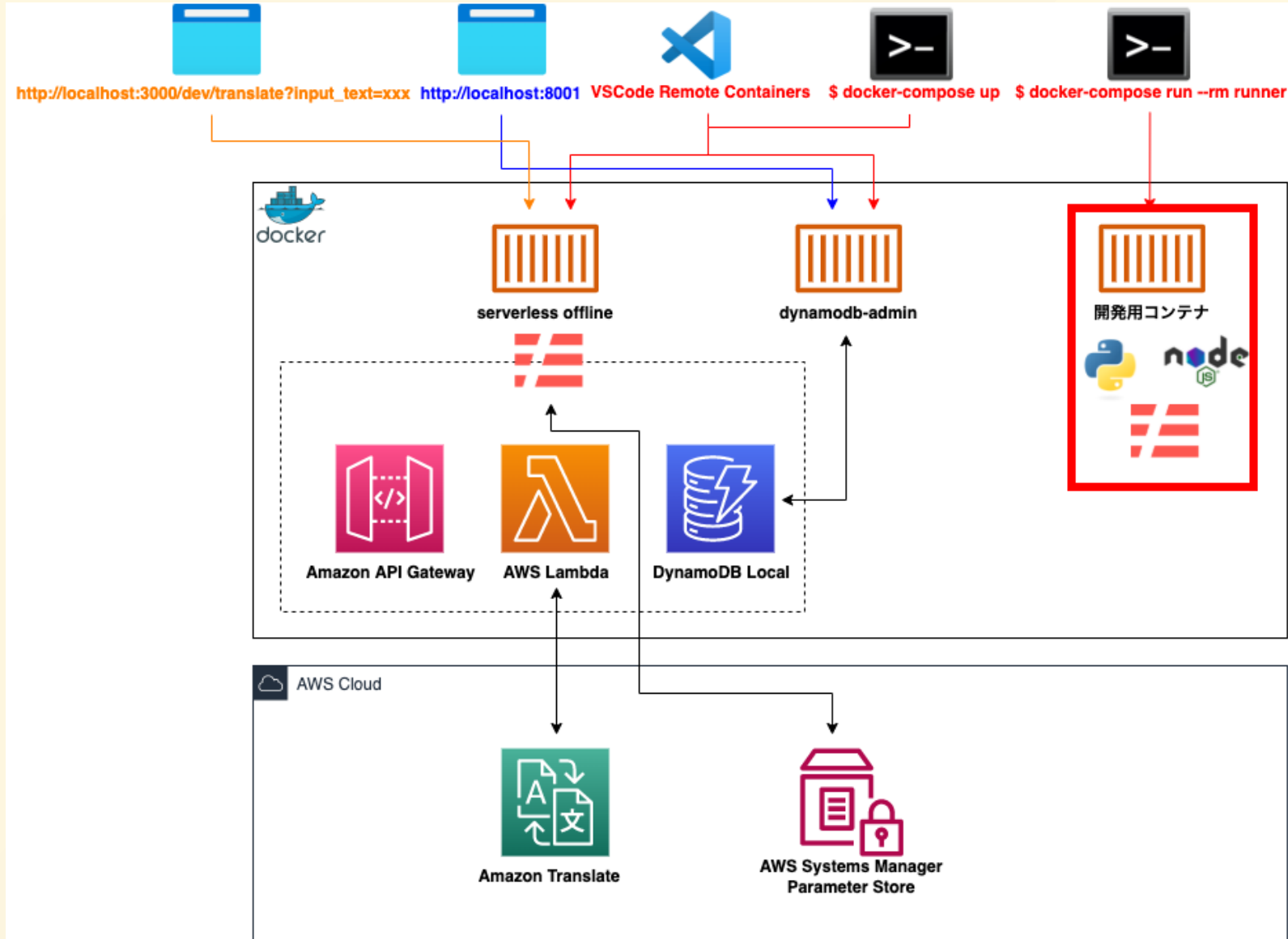
**ここからは
開発方法について紹介していきます**

開発方法

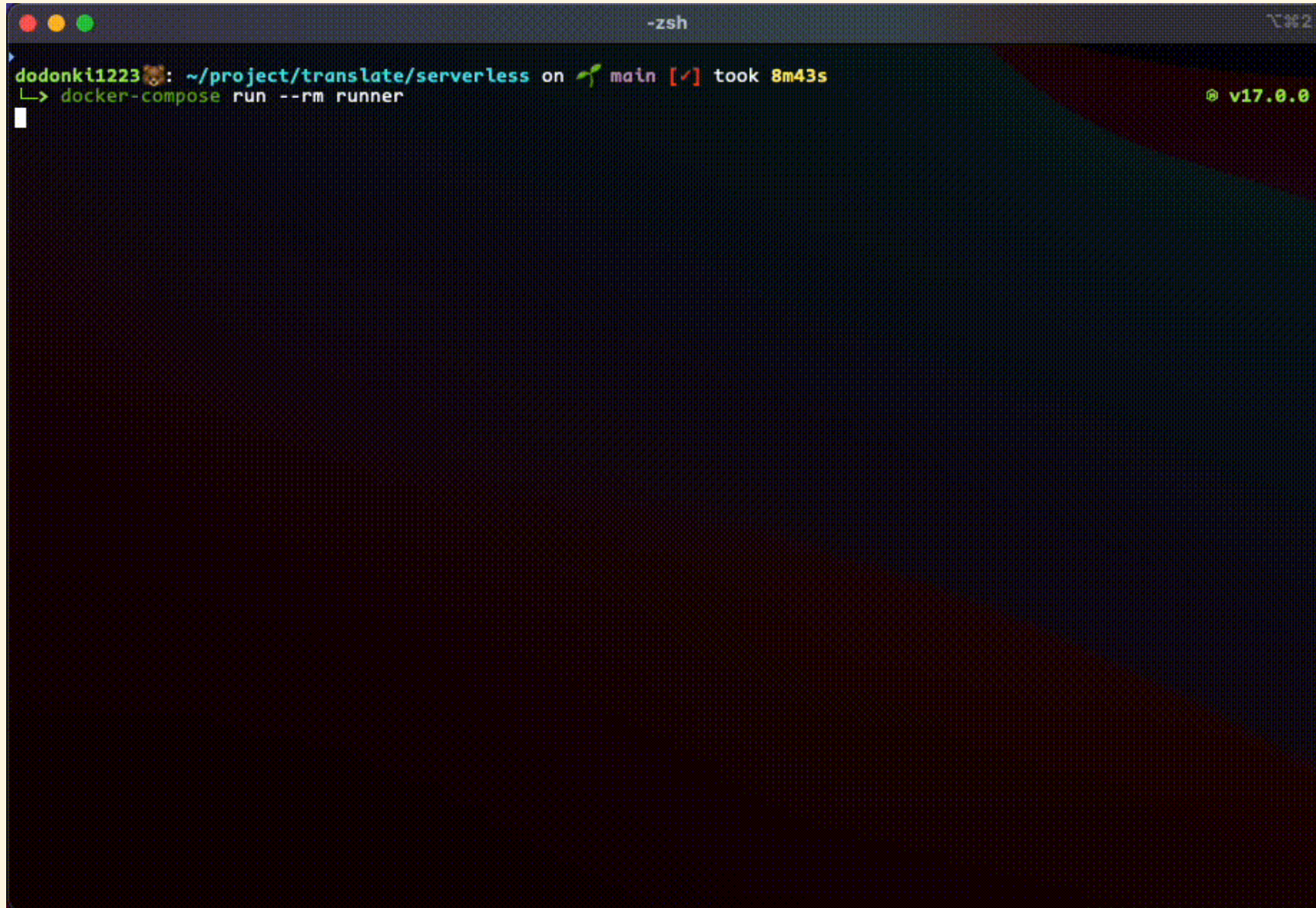
2つの開発方法があり、好みで選べるようにしてあります

- 開発用のコンテナにログインして開発を行う
- VSCode の Remote Containers を使用して offline コンテナにログインして開発を行う

開発用のコンテナ



開発用のコンテナ

A terminal window with a dark background and light-colored text. The window title bar shows three colored circles (red, yellow, green) on the left, '-zsh' in the center, and 'TM2' on the right. The terminal content shows a prompt 'dodonki1223' followed by a status bar indicating the current directory is '~/project/translate/serverless', the branch is 'main', and the commit took '8m43s'. Below this, the command 'docker-compose run --rm runner' is entered. A cursor is visible at the end of the command line. In the top right corner of the terminal area, there is a small green icon and the text '@ v17.0.0'.

```
dodonki1223: ~/project/translate/serverless on main [✓] took 8m43s
↳ docker-compose run --rm runner
```

開発用のコンテナ

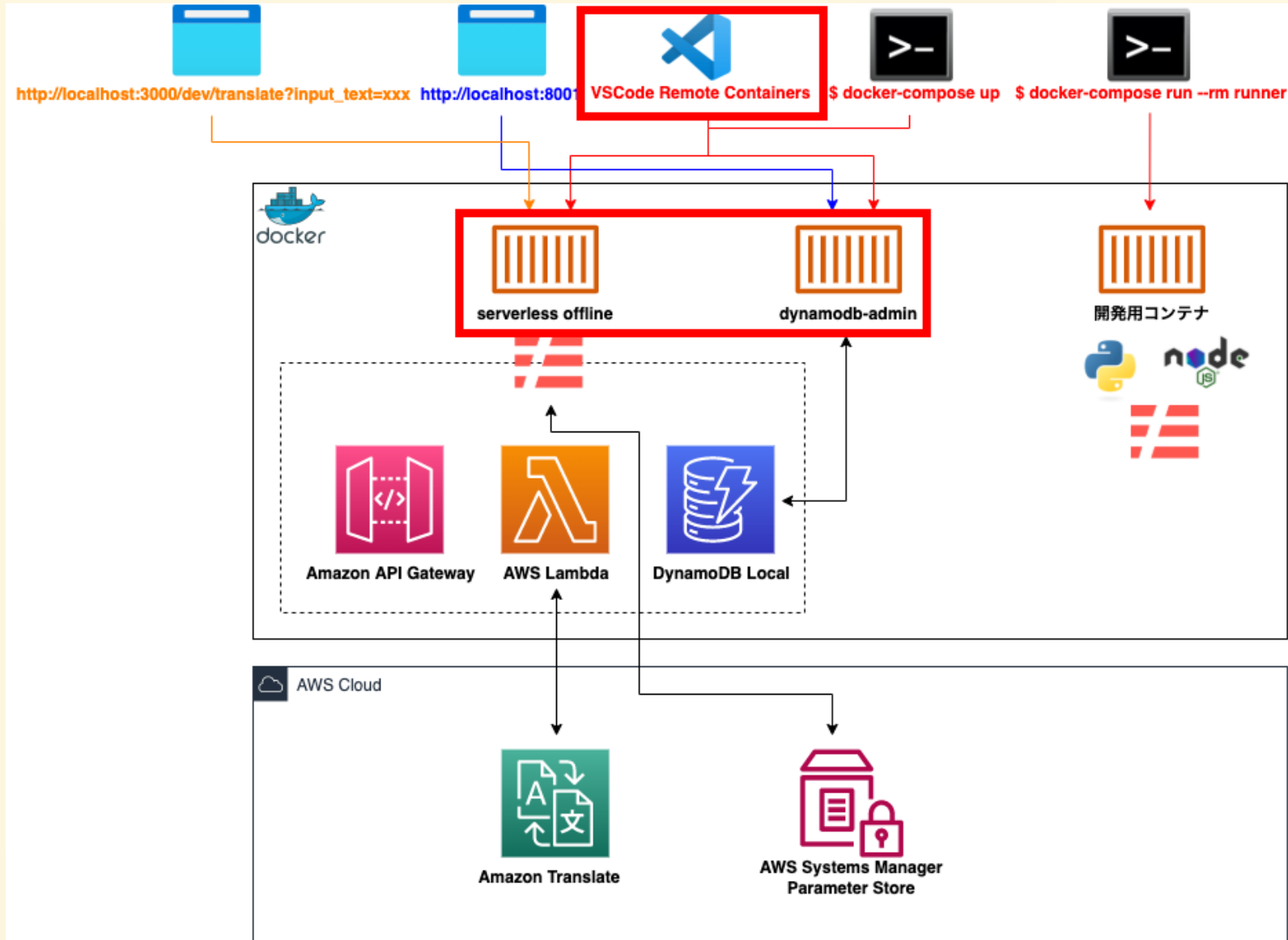
起動コマンド

```
$ docker-compose run --rm runner
```

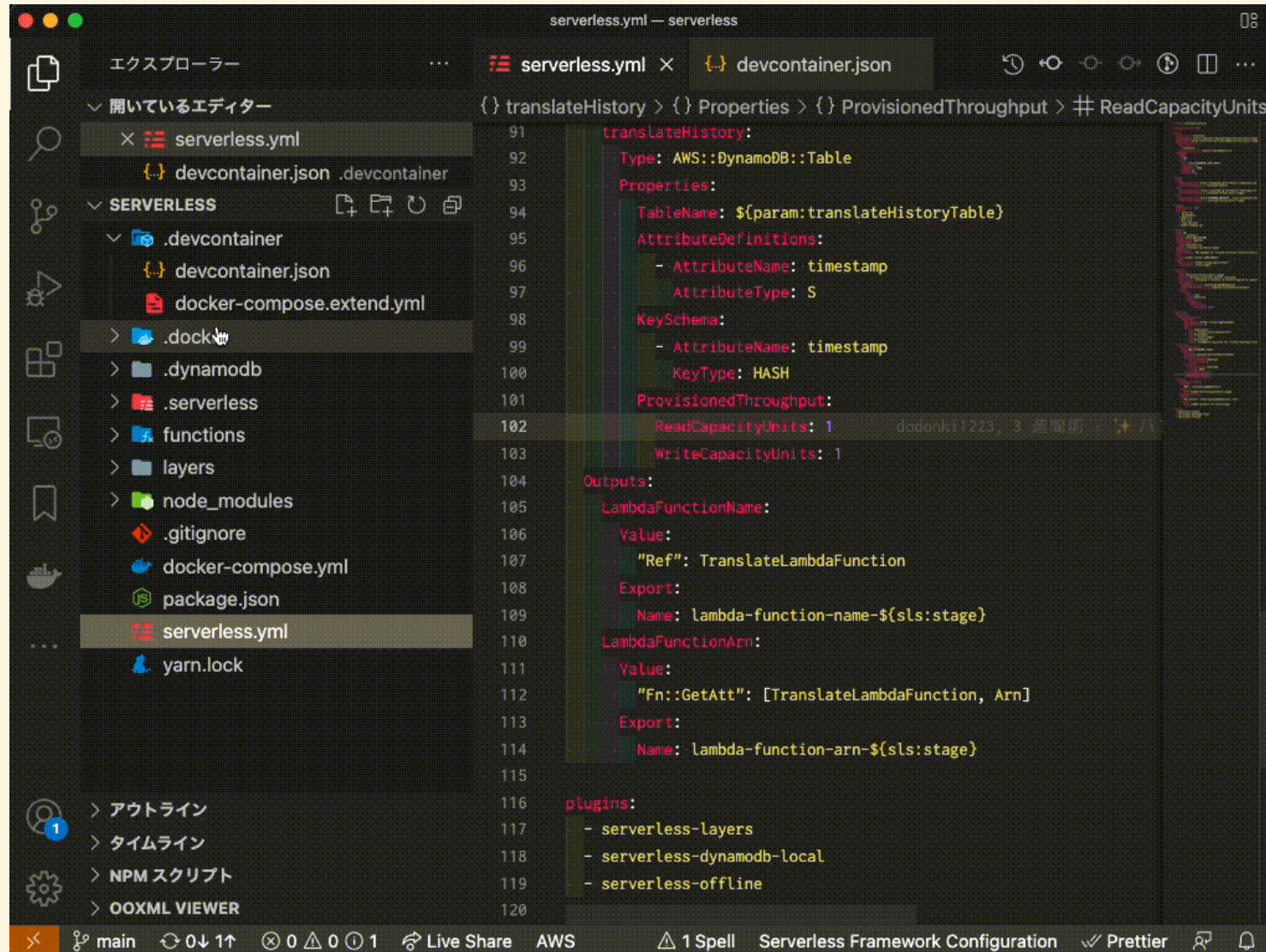
用途

- Serverless Framework 関連のコマンドの実行
- AWS CLI からの実行
- テストの実行

Remote Containers



Remote Containers



Remote Containers

とりあえず使ってみたかったので試してみただけです。
また機会があれば発表しようと思います。

なぜ Docker を使って開発しているのか？

DynamoDB Local が M1 Mac に対応していない！

DynamoDB Local が M1 Mac に対応していない！！

DynamoDB Local が M1 Mac に対応していない！！！！

**DynamoDB Local が M1 Mac に対応し
ていない ! ! ! !**

だから仕方なく Docker 化した 😇

最近、DynamoDB を扱うプラグインが Docker に対応されたため
Docker 化しなくても使えるかも 🤔

詳しくは [こちら - Support dockerized environment](#)

今日の発表を終わります

興味がある人はこちらの [リポジトリ](#) を覗いて下さい！