



SUMMIT  
TOKYO

# 時計会社がつくるUX共有プラットフォームとは？ ユーザーが開発したコードをLambdaにデプロイする 開発者参加型アプリケーションの仕組み

松王 大輔

時計開発本部 時計開発部  
コネクテッド開発課  
シチズン時計株式会社

# Agenda

Riiiverとは？ - なぜ時計会社がAWSプラットフォームが必要になったか

Riiiverの構想 誰でも簡単に自分の理想の体験を創造し、  
世界中の誰もがクリエイターになれるサービス

Riiiverの仕組み Pieceとiiidea

Piece開発者はサーバレスでデプロイ

Pieceのデプロイとデプロイ後の実行について

まとめと課題

# Riiiver とは？

なぜ時計会社がAWSプラットフォームが必要になったか

# New concept of WATCH

これからの腕時計って何だろう？

## Hackする

≠

## プログラミングする

今まで



これから

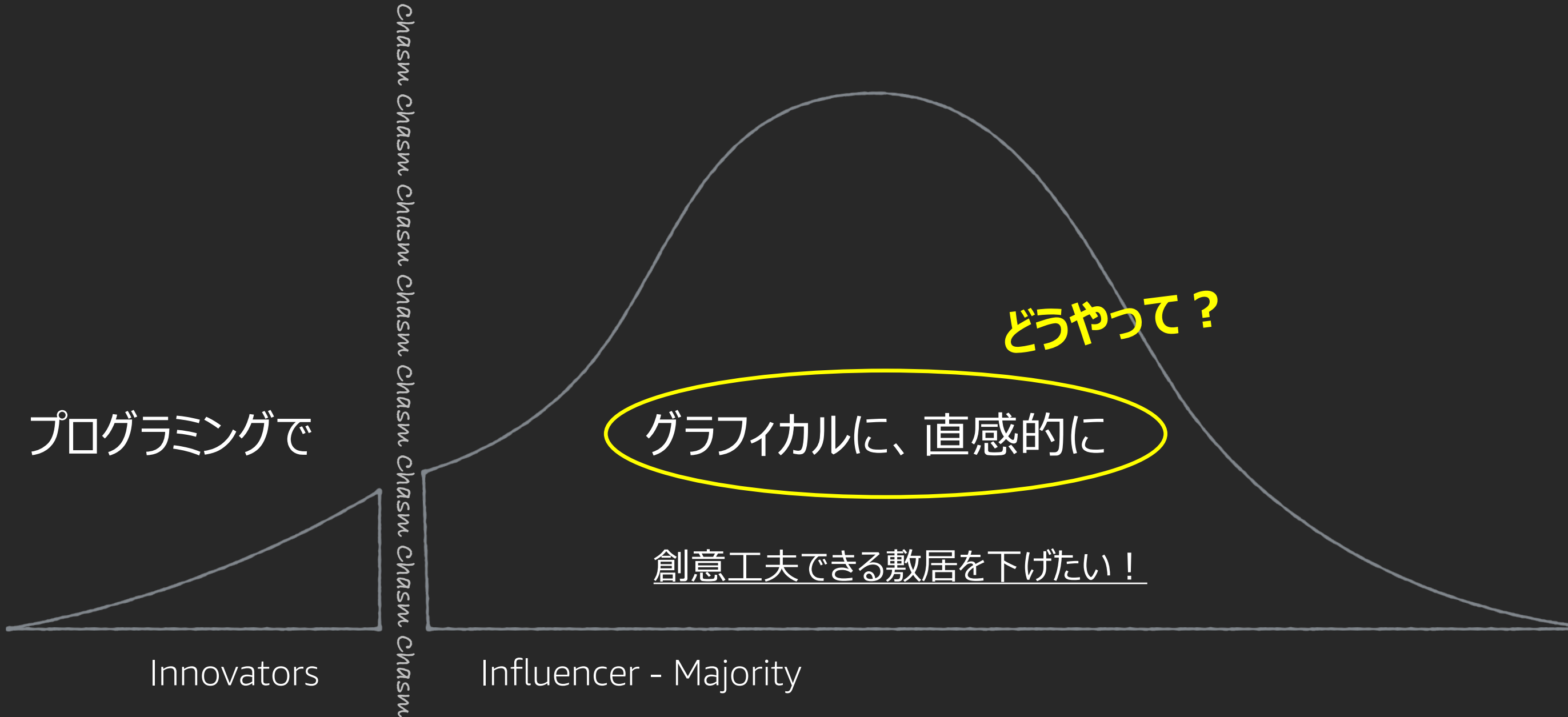


特定の機能を持ったこれまでの時計

ユーザー自身の手でこの機能をつくれるようにできないか？  
→ ハッカブルな腕時計にならないだろうか？

# 誰でも機能を作れる様に！

自分で機能を創造する環境を2種類用意

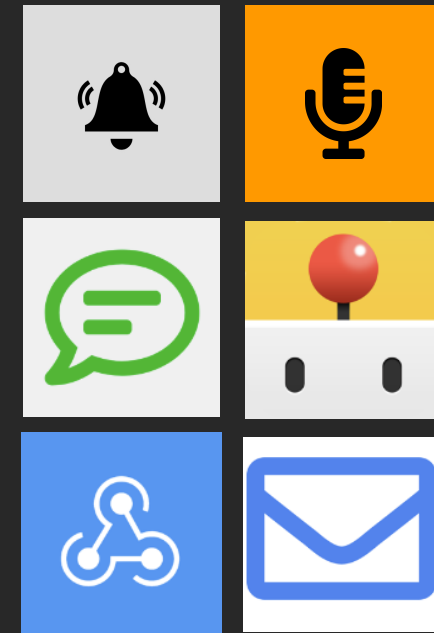


# Riiver の仕組み

## Piece と iiidea



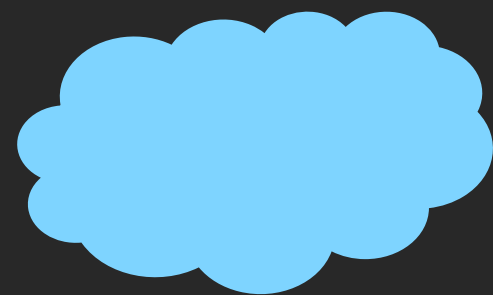
機能を  
分解!



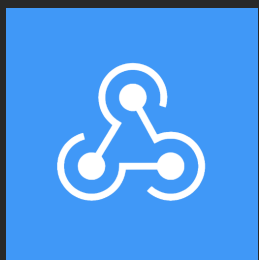
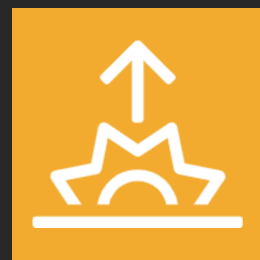
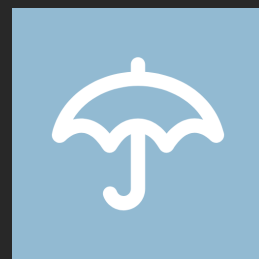
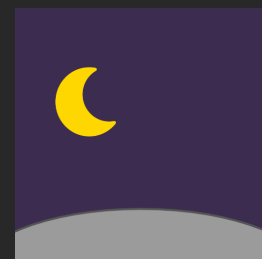
機能をモジュール化/シンボル化する

# Riiver の仕組み

Piece と iiidea



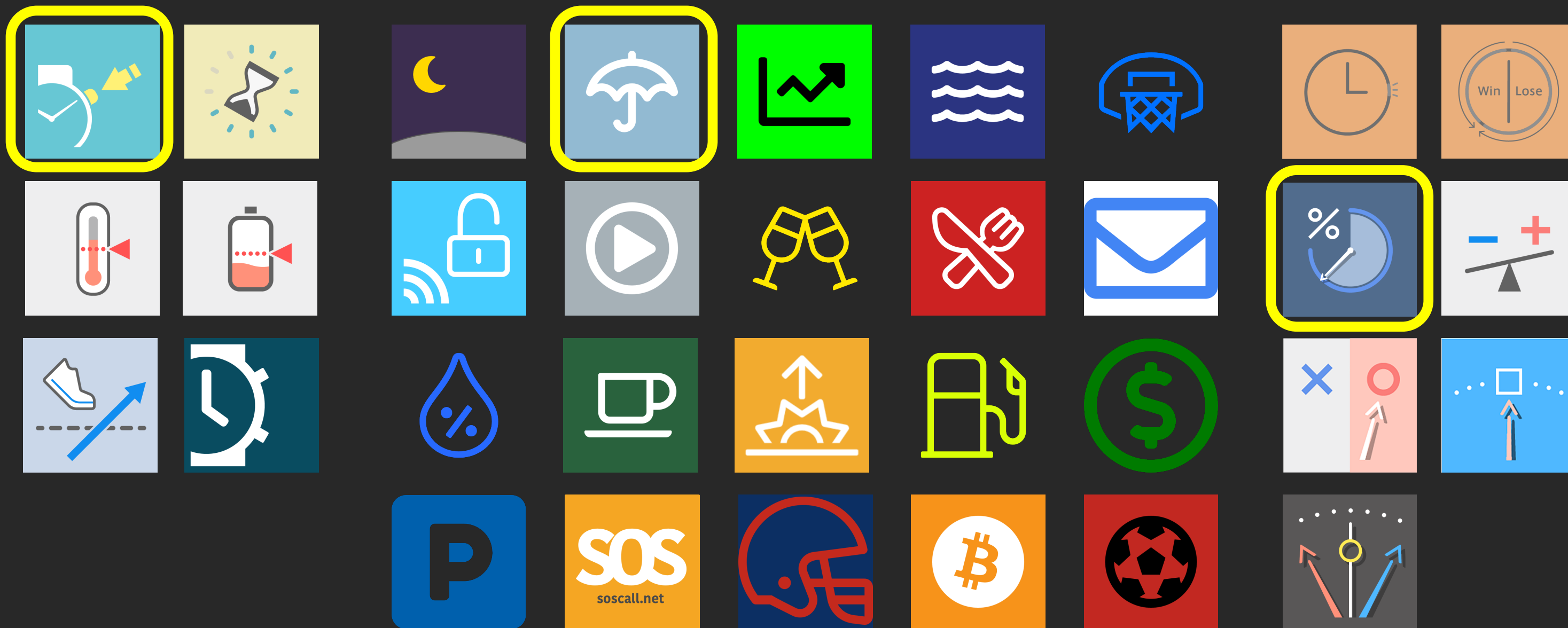
Web  
サービス





# Riiver の仕組み

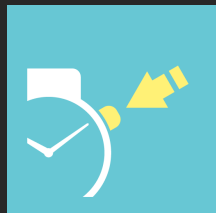
Piece と iiidea



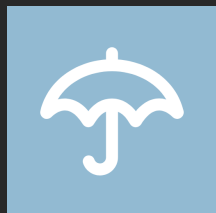
# Riiver の仕組み

## Piece と iiidea

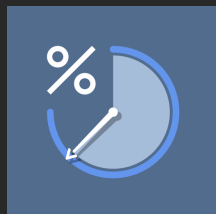
Piece -ユーザーが組み合わせる機能パーツ



Trigger Piece - きっかけ



Service Piece - サービスを提供

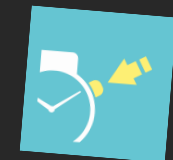


Action Piece - サービスの結果を出力



iiidea

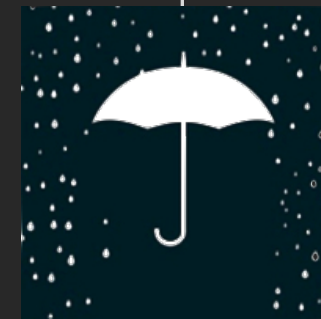
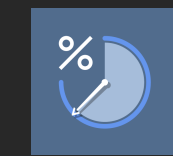
T



S

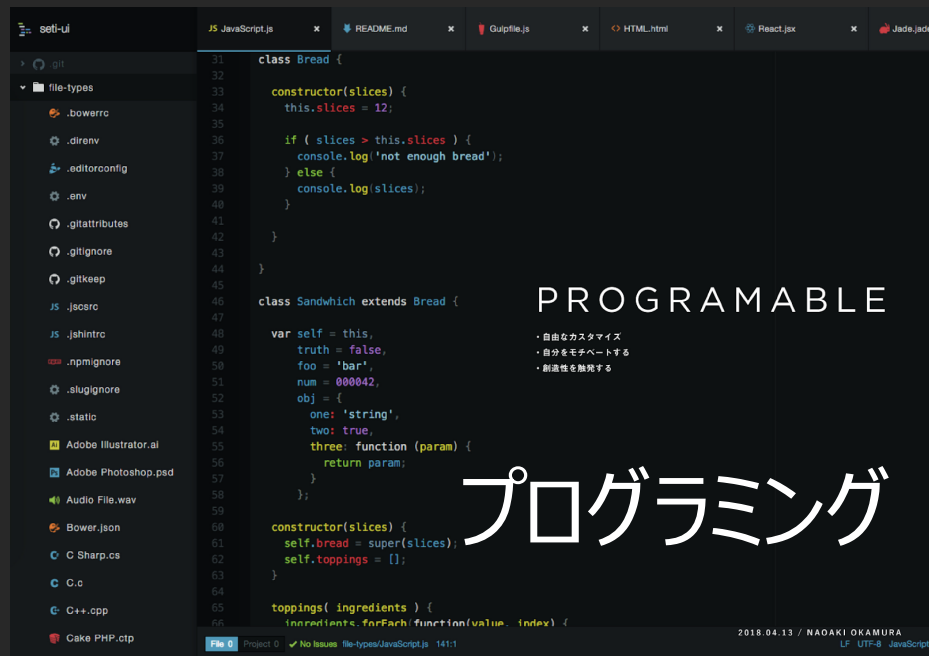


A



# Riiiver の仕組み

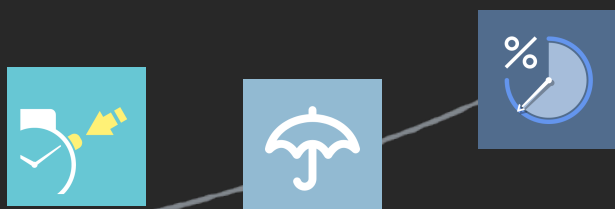
## Piece と iiidea



PROGRAMMABLE

プログラミング

Innovators

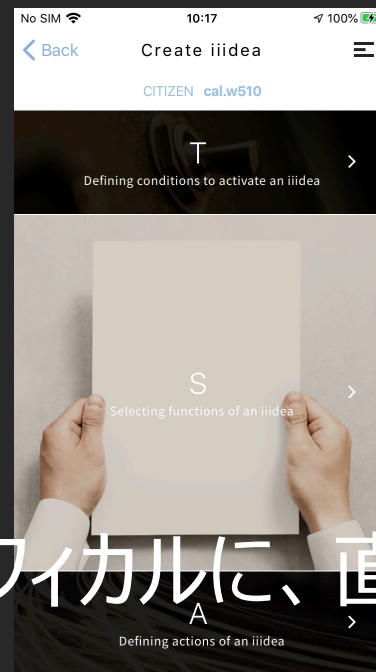


Chasm Chasm

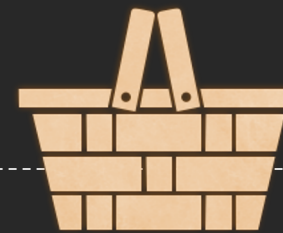
Influencer - Majority

誰でもiiideaリリース可能!

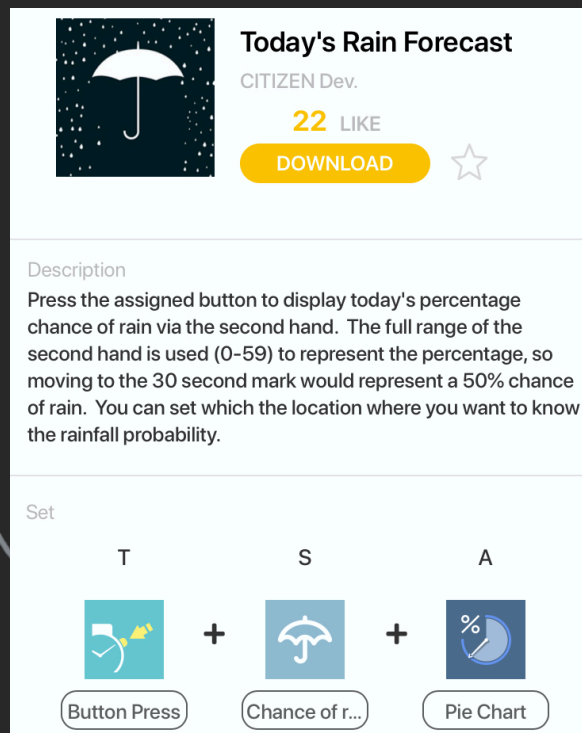
グラフィカルに、直的に



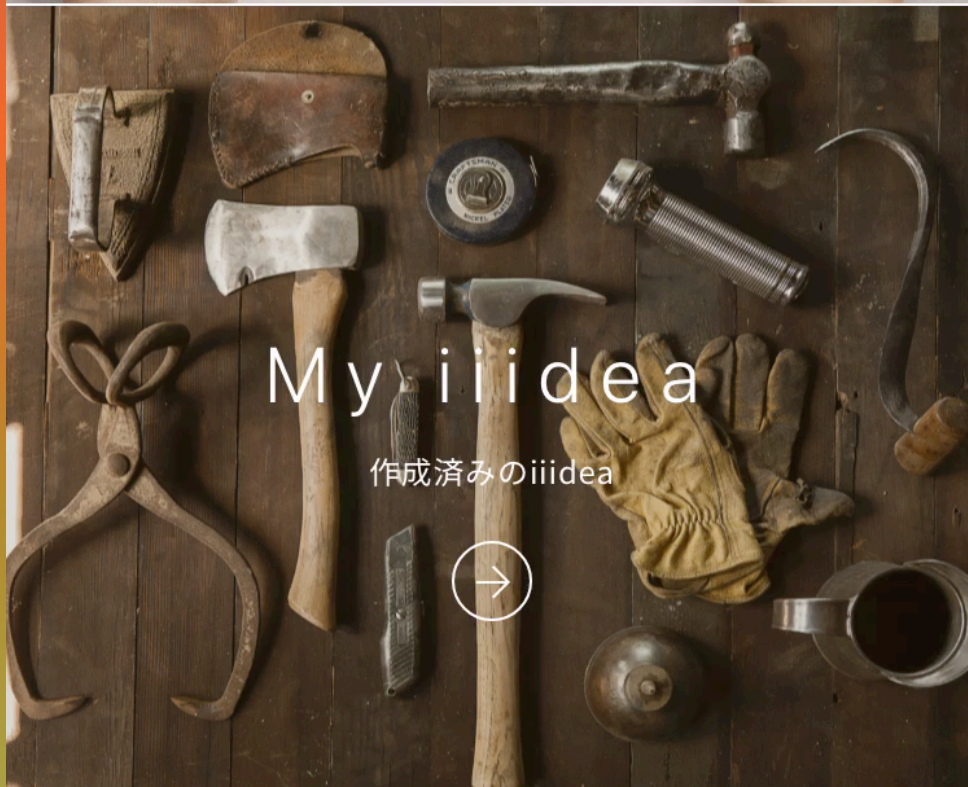
スマホアプリを使ってiiideaをリリース



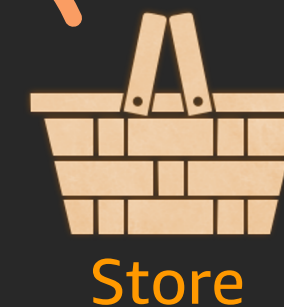
Store



# Demo



# 誰でもがクリエイターに！



自分が作った機能を利用できる、  
リリースして共有できる

# New concept of WATCH + Riiver Platform



+

*Riiver*

**Imagine. Inspire. Innovate.**

# 時計会社がプラットフォーム開発

グローバルを見据えつつ、スモールスタート。

## 構築時のポイント

- 小さく始めてまずは試してみたい

*AWS Lambda*

- Piece (たくさんの開発者の成果物) を抱える

- グローバル展開

*Amazon VPC /  
Amazon CloudFront*

- 個人情報dataの扱い



# Piece開発者はサーバレスでデプロイ

体験を創造するためのPieceに

# Piece開発の枠組み

ユーザーが組み合わせる部品であるPieceを増やす



```
class Bread {
  constructor(slices) {
    this.slices = slices;
  }
  if (slices > this.slices) {
    console.log('not enough bread');
  } else {
    console.log(slices);
  }
}

class Sandwich extends Bread {
  var self = this;
  truth = 'false',
  foo = 'bar',
  num = 000042,
  obj = {
    one: 'string',
    two: true,
    three: function (param) {
      return param;
    }
  };
}

constructor(slices) {
  self.bread = super(slices);
  self.toppings = [];
}

toppings(ingredients) {
  ingredients.forEach(function(value, index) {

```

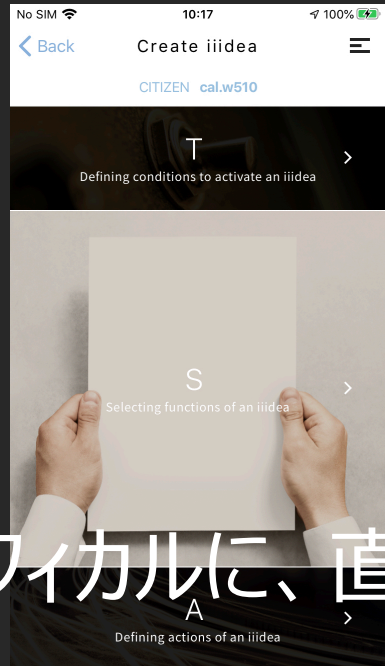
PROGRAMMABLE  
プログラミング



Innovators

Chasm Chasm Chasm Chasm Chasm Chasm Chasm Chasm Chasm Chasm Chasm

誰でもiiideaリリース可能！



グラフィカルに、直的に

Influencer - Majority

スマホアプリを使ってiiideaをリリース

**Today's Rain Forecast**  
CITIZEN Dev.  
22 LIKE  
[DOWNLOAD](#) ☆

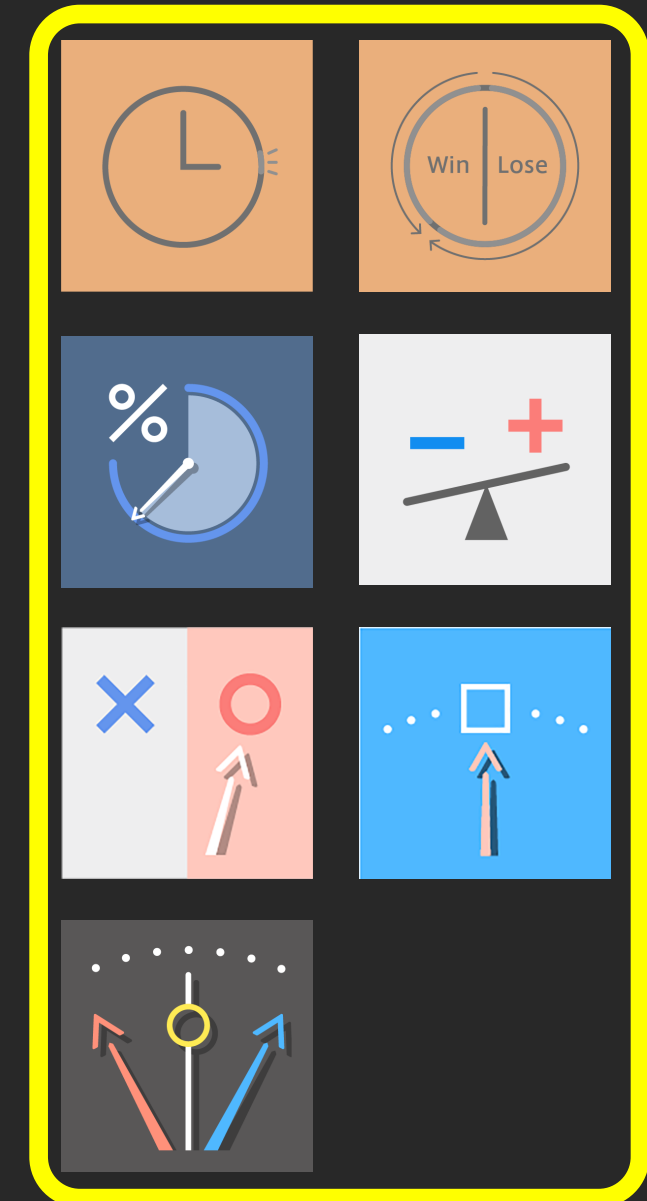
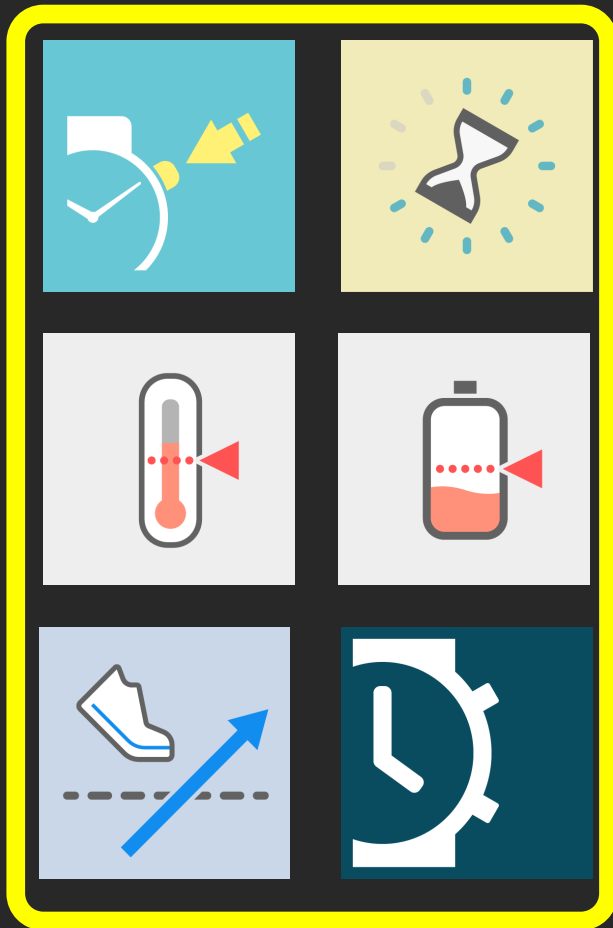
Description  
Press the assigned button to display today's percentage chance of rain via the second hand. The full range of the second hand is used (0-59) to represent the percentage, so moving to the 30 second mark would represent a 50% chance of rain. You can set which the location where you want to know the rainfall probability.

Set

T	S	A
Button Press	Chance of r...	Pie Chart

# Piece 開発方法は 2 種類

エッジで処理するPieceの開発 と Web上の処理で完結するPiece



# Piece 開発の種類

エッジで処理するPieceの開発 と Web上の処理で完結するPiece



Riiver SDK  
(Android/iOS向け)

スマホ上にPieceの  
振る舞いや前後Pieceとの  
データやりとりを実装

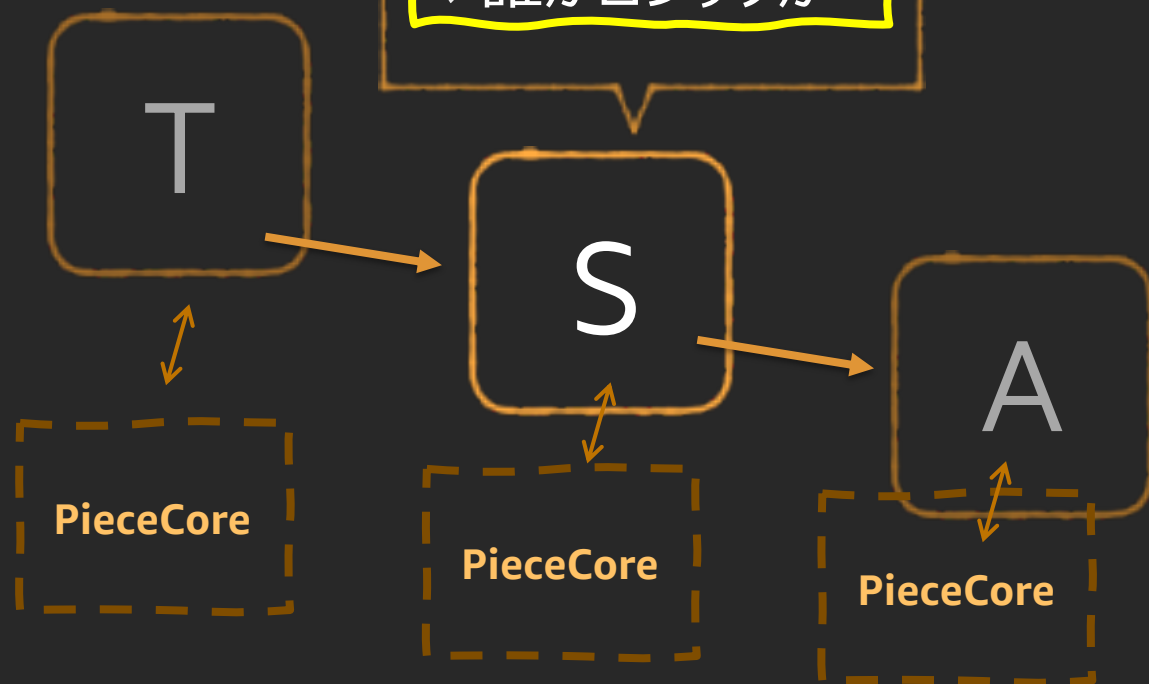


Riiverが準備する  
AWS Lambda 上で  
動作する。

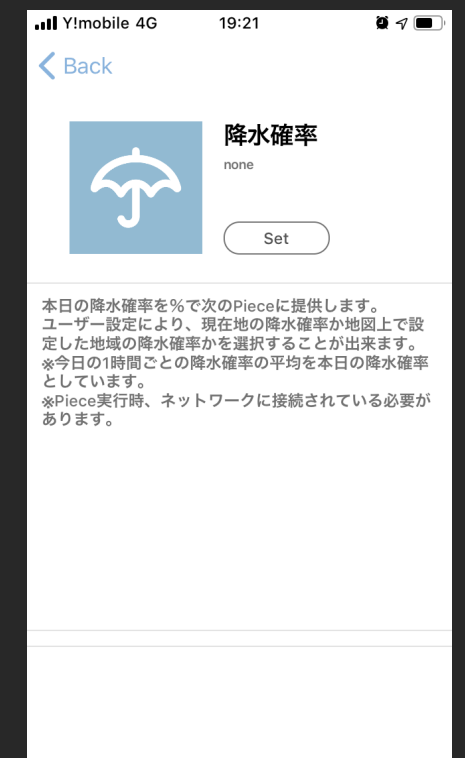
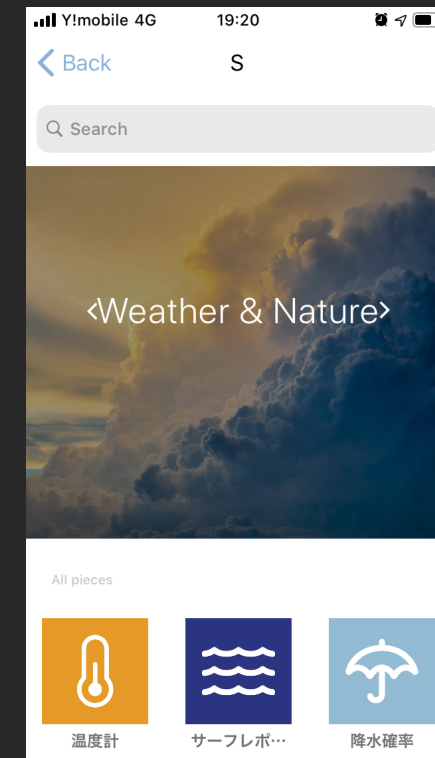
# Pieceの構成

## Pieceの仕様書と実装部分

- PieceJSON
- ✓ 名前
  - ✓ 説明文
  - ✓ カテゴリ
  - ✓ 入力データ
  - ✓ 出力データ
  - ✓ 設定可能項目
  - ✓ 誰がロジックか



Create



Use



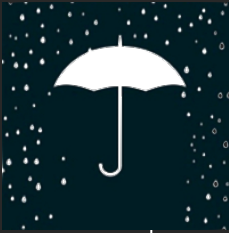
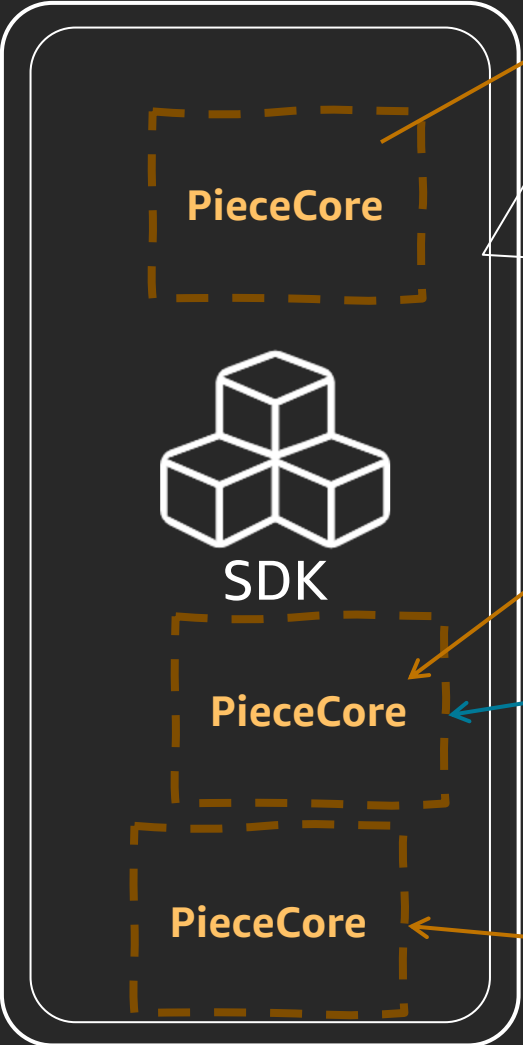
# iiidea実行時の動作

PieceCoreはSDKがハンドル



iiidea実行

針を3時の位置に！



ボタンを押す



外部サービスを利用



針で指す



現在位置

30% !

# iiidea実行時の動作

PieceCoreはSDKがハンドル



## 気にしないでいいです!

iiidea実行  
針を3時の位置に!



ボタンを押す

外部サービスを利用

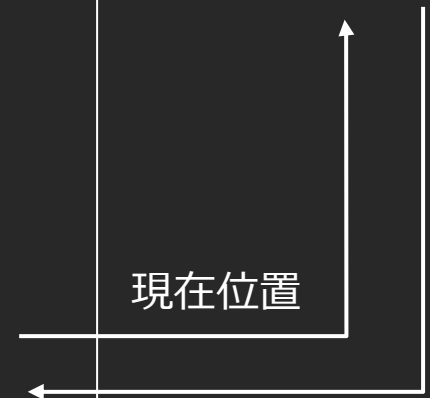


現在位置

30%!



針で指す



# iiidea実行時の動作

PieceCoreはSDKがハンドリング

## Lambdaのみで完結

```
5 const request = require('request');
6
7 exports.handler = async event => {
8   /* Code for "warm start" The Riiver system might call your Lambda function
9   > if (event.call === 'lambda') { ...
17  }
18
19   /* Start coding here - これ以降にコードを記入してください。*/
20
21   // 前のpieceからの入力データ
22   const inputdata = event.properties.input;
23
24   const apiHost = 'https://api.example.com';
25   const apiPath = '/data/2.2/target';
26   const apikey = '98dxxxxxxxxxxxxxxxxxxxxxxxxxxxx'; // ← insert your API key
27
28   /// 処理！
29
30   // 次のPieceにわたすデータ
31   let response = {
32     status: 200,
33     body: {
34       chanceOfRain: 20 // ← これが次のPieceに渡る
35     }
36   };
37
38   return response;
39
```



位置

30% !



iiidea実行

針を3時の位置に！



# Piece をデプロイし、公開する

Riiver側で用意しています！

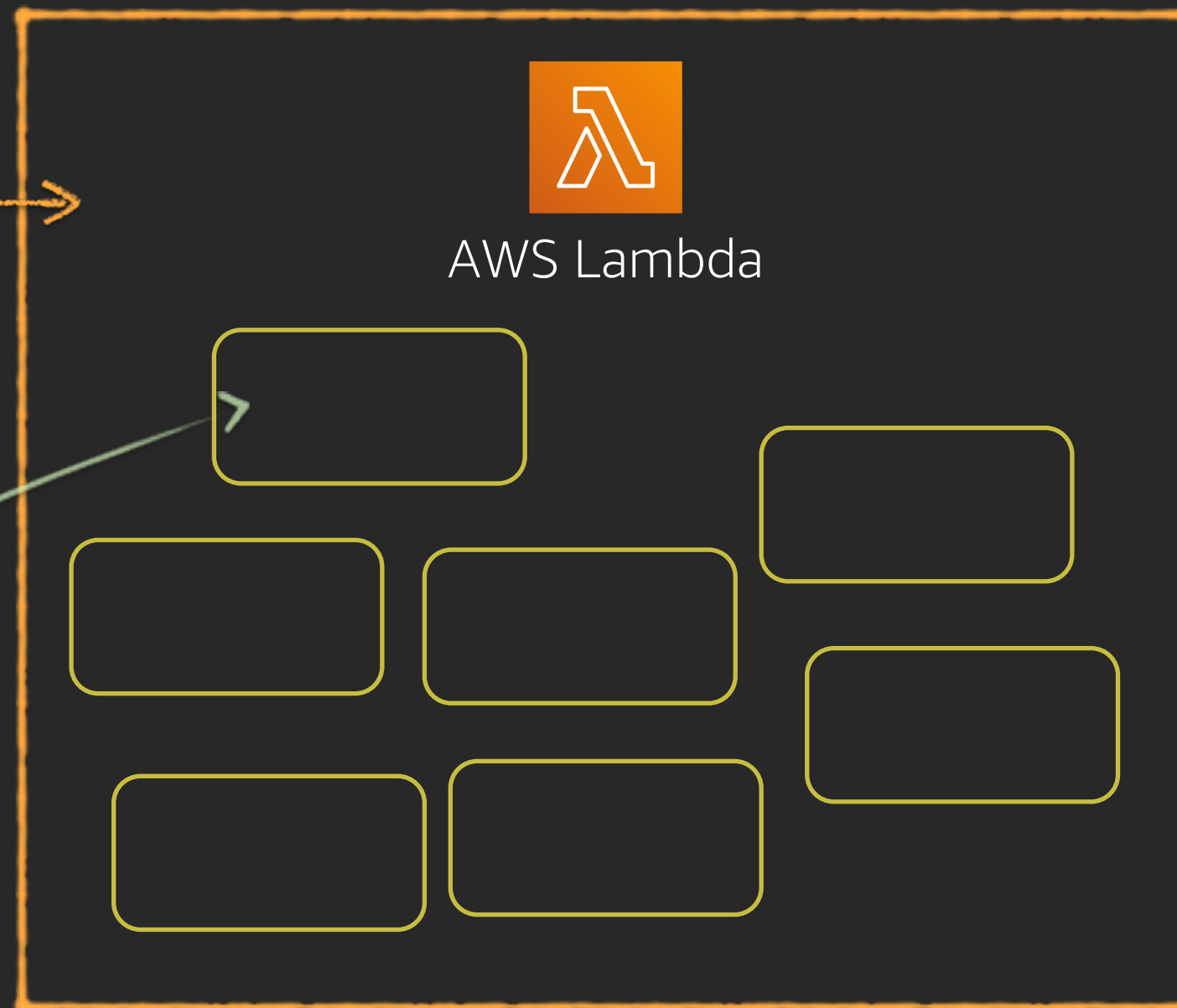
WebPortal <Piece Builder>



Javascript

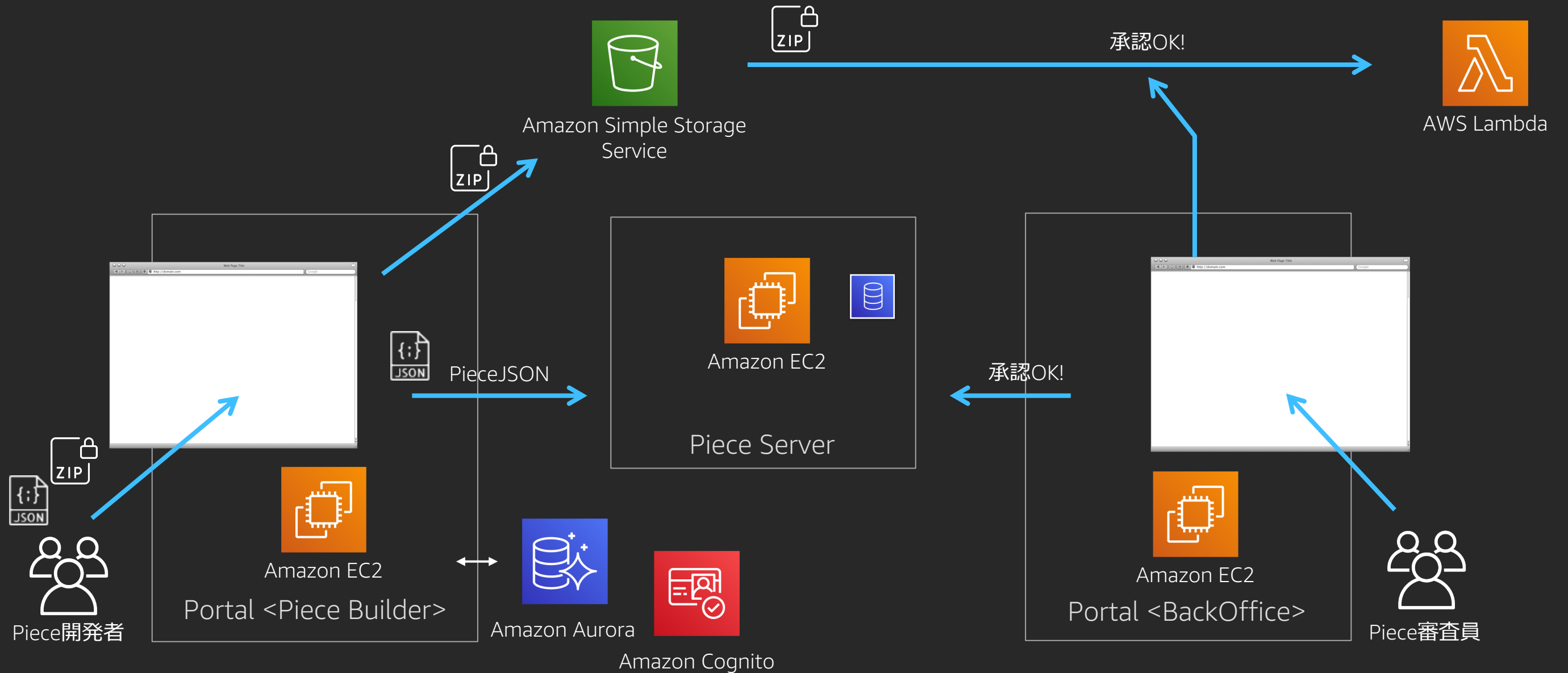


AWS Lambda



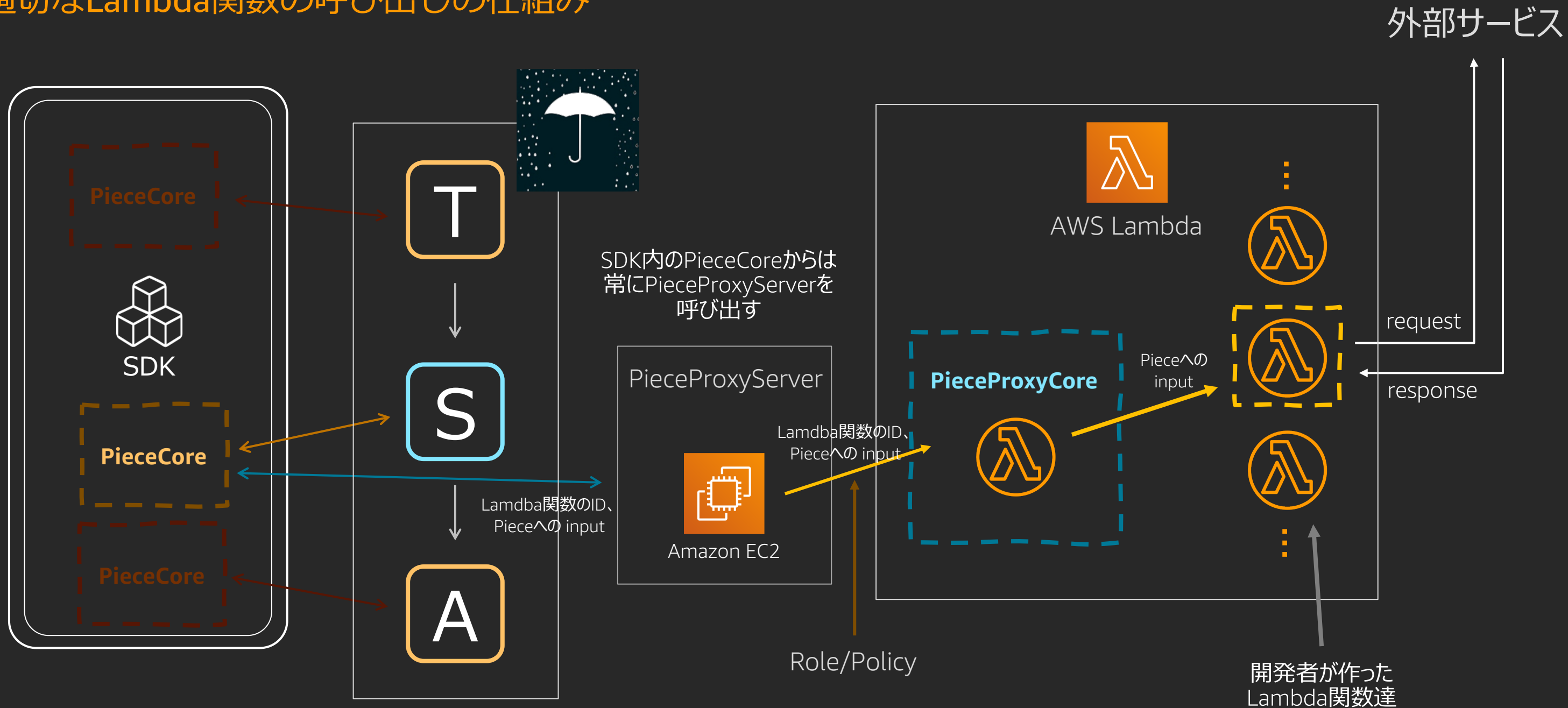
# Piece開発からリリースまでのフロー

実際のオペレーションと構成 (一部抜粋)



# Piece の実行

## 適切なLambda関数の呼び出しの仕組み



# Lambdaを利用したPiece開発からリリースまで

## Lambdaの特徴を活用できたか？

- Pros

- 開発者はサーバを準備しなくても良い。
- スマートフォン側の処理や開発環境について意識する必要がない。
- 一つのLambda関数だけに集中できる。
- 独立性
  - 開発者の区切りが関数の区切り。
  - 障害が起きても他のPieceに影響しない。

- Cons

- 現状、本番環境でテスト・デバッグができない。（今後の課題です。）

# Riiver Platformの開発環境として

Lambdaを利用することで得られたこと

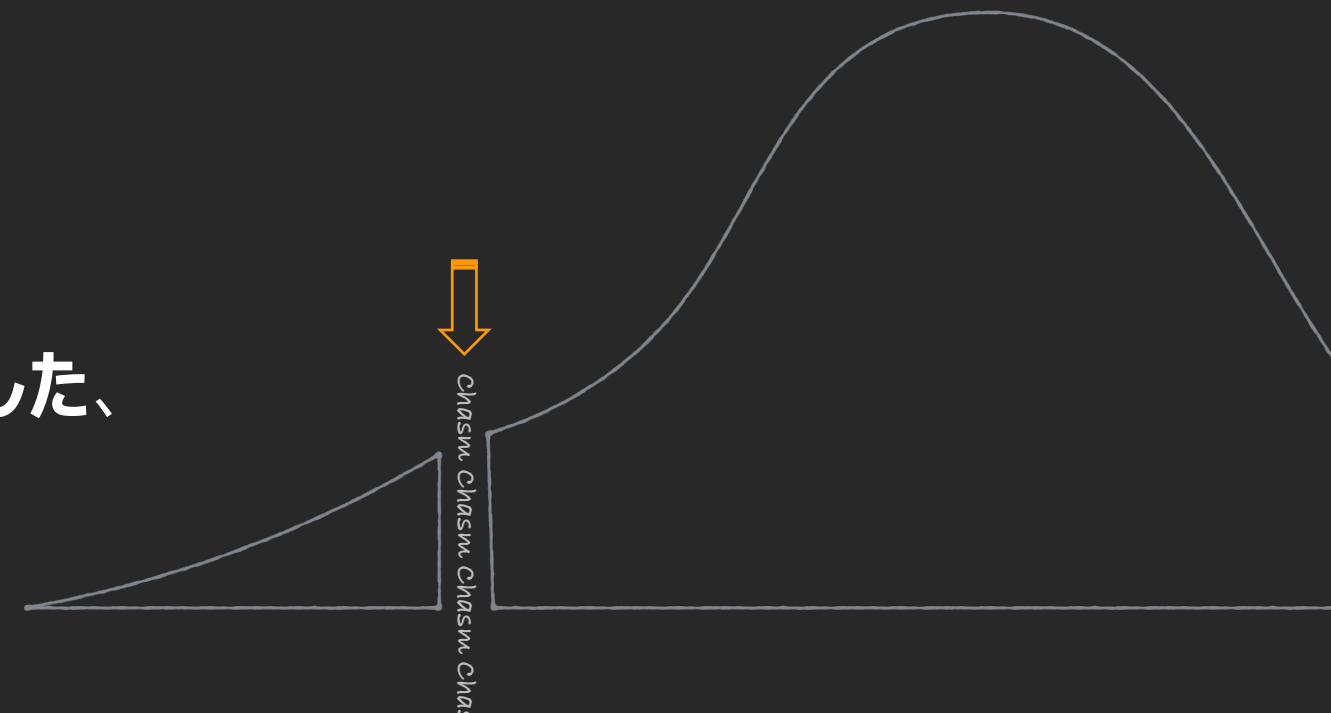
- Pros

- 開発者はサーバを準備しなくても良い。
- スマートフォン側の処理や開発環境について意識する必要がない。
- 一つのLambda関数だけに集中できる。



創意工夫できる敷居を下げることができた！

世にリリースする初めてのプログラムが  
**Lambdaを利用したRiiverのPiece開発でした、**  
という人が増えれば！



# ぜひ、皆様のPieceをLambda上にリリース下さい！

デベロッパーサイトにチュートリアルも準備

Riiiver Developers

Riiiverについて | ドキュメント | チュートリアル | ダウンロード | ニュース | English

Search Login

## 人類、皆クリエイター。

Riiiverは、老若男女誰でも簡単に自分の理想の体験を創造し、世界中の誰もがクリエイターになれるサービスを目指しています。

Riiiverについて >

<https://developer.riiiver.com/>



# Riiiver

Imagine. Inspire. Innovate.

Riiiver is IoT platform, produced by CITIZEN.

# Thank you!

松王 大輔

[matsuohd@citizen.co.jp](mailto:matsuohd@citizen.co.jp)