

AWS Cloud9 を使ってみた



目次

1. AWS Cloud9 とは
2. 設定方法
3. 使い方



1. AWS Cloud9 とは

1. AWS Cloud9 とは

- ▶ <https://aws.amazon.com/jp/cloud9/>
- ▶ ブラウザのみでコードを記述、実行、デバッグできるクラウドベースの統合開発環境 (IDE)



歴史

- ▶ 2016年7月にAWSがCloud9を買収
- ▶ 2017年のre:inventで発表
- ▶ 2017年12月にリリース



特徴

- ▶ ブラウザのみでコードを記述出来る
- ▶ リアルタイムで共同コーディング出来る
- ▶ 40以上の言語をサポート
 - ▶ Node.js (JavaScript)、Python、PHP、Ruby、Go、C++・・・など
 - ▶ https://docs.aws.amazon.com/ja_jp/cloud9/latest/user-guide/language-support.html
- ▶ 2018年7月現在、東京リージョンには未対応



気になったのが

▶ サーバーレスアプリケーションを簡単に構築

- ▶ AWS Lambda 関数をローカルでテストおよびデバッグするための環境を利用できるらしい
- ▶ Lambdaのテストはめんどくさいので簡単に出来たら嬉しそう
- ▶ AWS自身の製品なので、AWSとの親和性が高そうなのと、今後の改良にも期待が持てるのではないか

2. 設定方法

サービスからCloud9を選択

Developer Tools

AWS Cloud9

a cloud IDE for writing,
running, and debugging
code

AWS Cloud9 allows you to write, run, and debug your code with just a browser. With AWS Cloud9, you have immediate access to a rich code editor, integrated debugger, and built-in terminal with preconfigured AWS CLI. You can get started in minutes and no longer have to spend the time to install local applications or configure your development machine.

New AWS Cloud9 environment

Create environment



とりあえず環境作成

Step 1
Name environment

Step 2
Configure settings

Step 3
Review

Name environment

Environment name and description

Name
The name needs to be unique per user. You can update it at any time in your environment settings.

Limit: 60 characters

Description - *Optional*
This will appear on your environment's card in your dashboard. You can update it at any time in your environment settings.

Limit: 200 characters

Cancel **Next step**



Cloud9用のEC2インスタンスを設定

Configure settings

Environment settings

Environment type [Info](#)

Choose between creating a new EC2 instance for your new environment or connecting directly to your server over SSH.

- Create a new instance for environment (EC2)**
Launch a new instance in this region to run your new environment.
- Connect and run in remote server (SSH)**
Display instructions to connect remotely over SSH and run your new environment.

Instance type


- t2.micro (1 GiB RAM + 1 vCPU)**
Free-tier eligible. Ideal for educational users and exploration.
- t2.small (2 GiB RAM + 1 vCPU)**
Recommended for small-sized web projects.
- m4.large (8 GiB RAM + 2 vCPU)**
Recommended for production and general-purpose development.
- Other instance type**
Select an instance type.

t2.nano

自動シャットダウンの設定とか

Cost-saving setting
Choose a predetermined amount of time to auto-hibernate your environment and prevent unnecessary charges. We recommend a hibernation settings of half an hour of no activity to maximize savings.



After 30 minutes (default) ▼

IAM role
AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf. You can delete the role from the AWS IAM console once you no longer have any AWS Cloud9 environments. [Learn more](#) 



AWSServiceRoleForAWScloud9

▼ **Network settings (advanced)**

Network (VPC)
Launch your EC2 instance into an existing Amazon Private Cloud (VPC) or create a new one.

vpc-4322a724 (default) ▼   Create new VPC

Subnet
Select a range of IP addresses in your VPC to isolate EC2 resources from each other.

No preference (default subnet in any Availability Zone) ▼   Create new subnet

Cancel Previous step Next step

確認画面

Subnet

subnet-d6ced9a0

Cost-saving settings



After 30 minutes (default)

IAM role

AWSServiceRoleForAWSCloud9 (generated)



We recommend the following best practices for using your AWS Cloud9 environment

- Use **source control and backup** your environment frequently. AWS Cloud9 does not perform automatic backups.
- Perform regular **updates of software** on your environment. AWS Cloud9 does not perform automatic updates on your behalf.
- **Turn on AWS CloudTrail in your AWS account** to track activity in your environment. [Learn more](#) 
- Only share your environment with **trusted users**. Sharing your environment may put your AWS access credentials at risk. [Learn more](#) 

Cancel

Previous step

Create environment

作成できました

The screenshot displays the AWS Cloud9 IDE interface. At the top, a menu bar includes 'AWS Cloud9', 'File', 'Edit', 'Find', 'View', 'Goto', 'Run', 'Tools', 'Window', and 'Support'. A 'Preview' button and a green 'Run' button are also visible. On the right side of the top bar, there is a user icon 'K' and a 'Share' button.

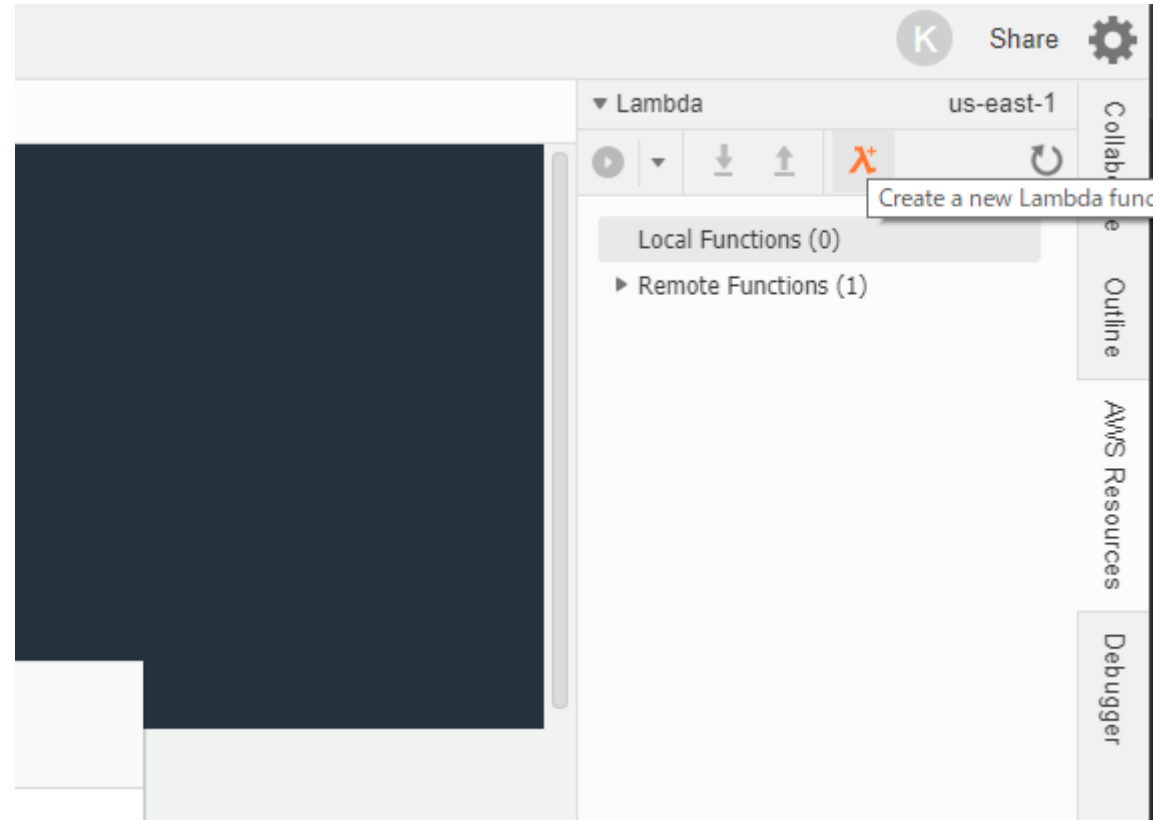
The main workspace is divided into several sections:

- Environment:** Shows a folder named 'kumakura-dev' containing a 'README.md' file.
- Developer Tools:** A dark-themed area with the text 'Developer Tools' and a large 'Welcome' message. The message reads: 'AWS Cloud9 Welcome to your development environment'. Below this, it states: 'AWS Cloud9 allows you to write, run, and debug your code with just a browser. You can [tour the IDE](#), write code for [AWS Lambda](#) and [Amazon API Gateway](#), share your [IDE](#) with others in real time, and much more.'
- Getting started:** A white box on the right side of the workspace containing the title 'Getting started' and three links: 'Create File', 'Open File...', and 'Upload Files...'.
- AWS Cloud9 for AWS Lambda:** A section with the title 'AWS Cloud9 for AWS Lambda' and a text box that says: 'AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda executes your code'.
- Terminal:** A terminal window at the bottom with two tabs: 'bash - "ip-172-31' and 'Immediate'. The terminal output shows: '[detached (from session cloud9_terminal_623)]' followed by a cursor.

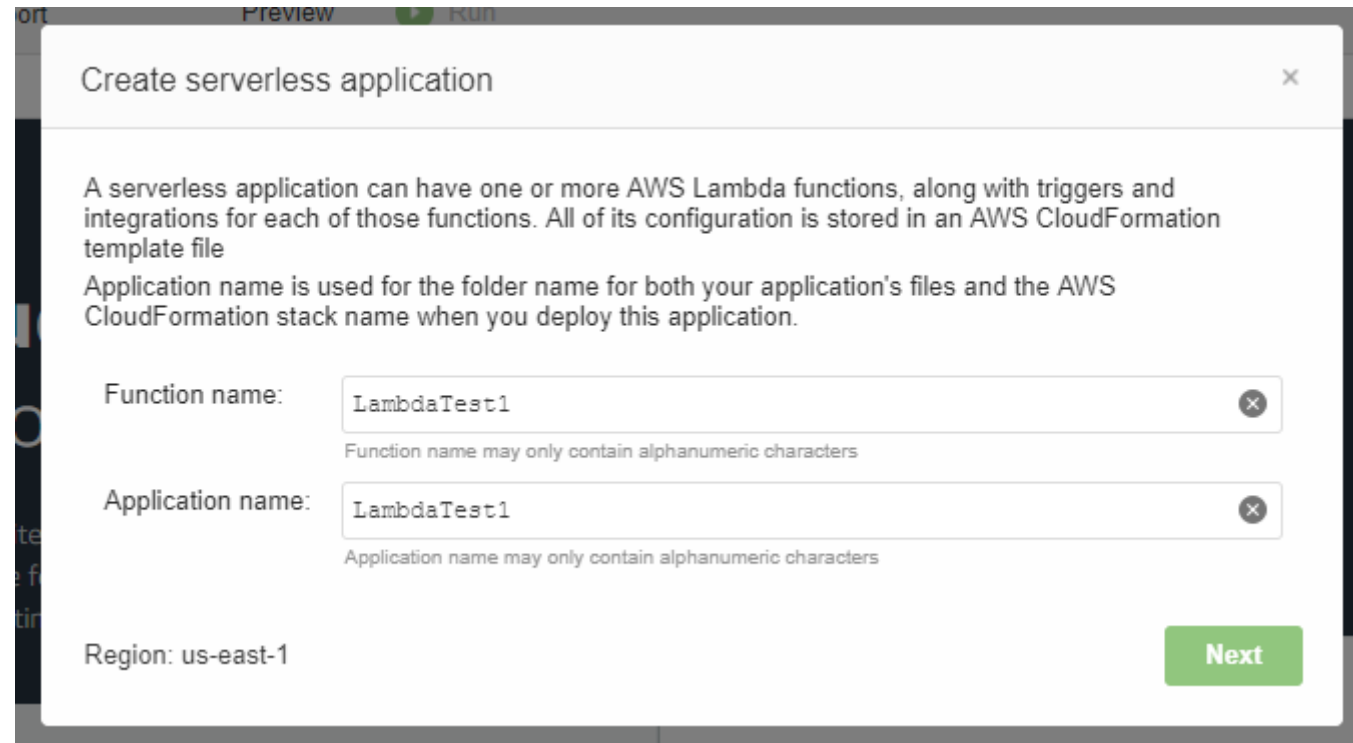
On the left side, there are vertical navigation bars for 'Environment', 'Navigate', 'Commands', and 'Collaborate'. On the right side, there are vertical navigation bars for 'Collaborate', 'Outline', 'AWS Resources', and 'Debugger'.

3. 使い方

Lambdaを作成する



Lambdaを作成する



Preview Run

Create serverless application ✕

A serverless application can have one or more AWS Lambda functions, along with triggers and integrations for each of those functions. All of its configuration is stored in an AWS CloudFormation template file

Application name is used for the folder name for both your application's files and the AWS CloudFormation stack name when you deploy this application.

Function name: ✕
Function name may only contain alphanumeric characters

Application name: ✕
Application name may only contain alphanumeric characters

Region: us-east-1

[Next](#)



Runtimeやblueprintを選択

port Preview Run

Create serverless application

Select runtime

Python 3.6

Select blueprint

empty-python
An empty Python function
python · python3.6

cloudwatch-alarm-to-slack
An Amazon SNS trigger that sends CloudWatch alarm notifications to Slack.
python3.6 · cloudwatch · slack

Region: us-east-1

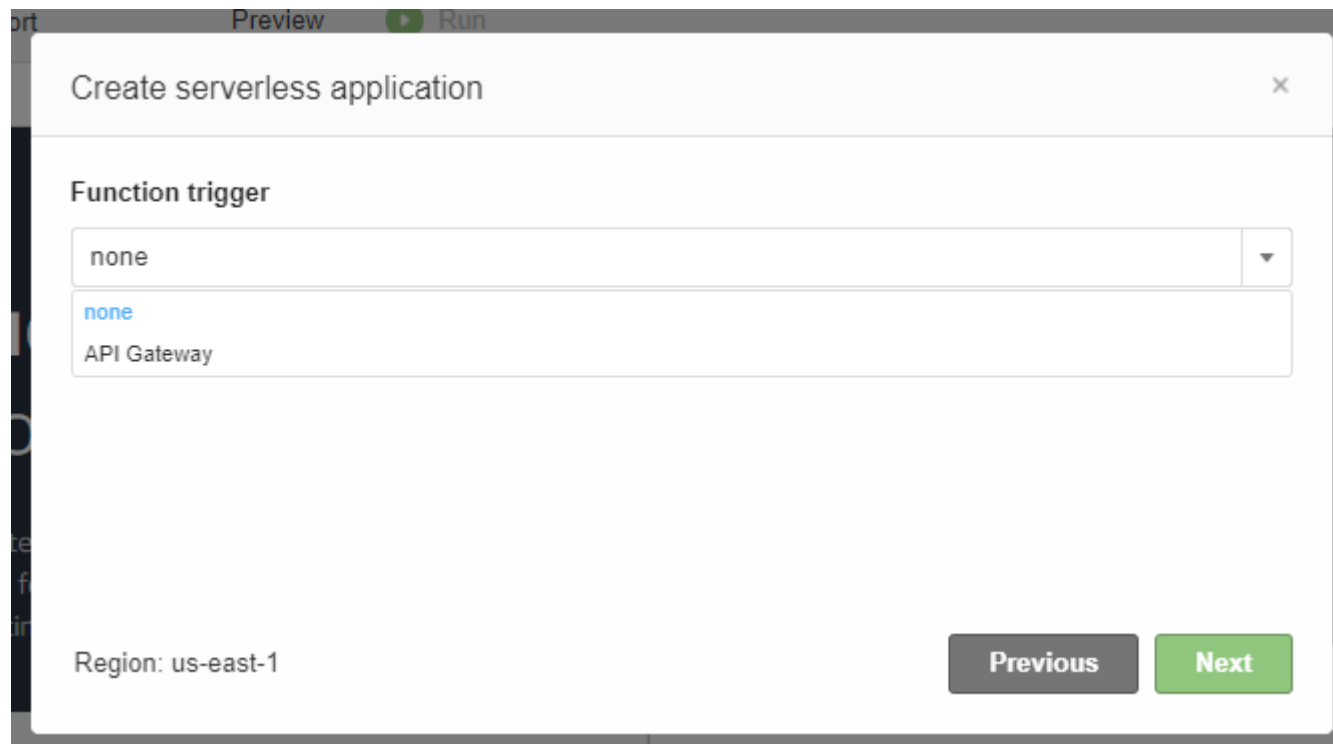
dynamodb-process-stream

hello-world-python3

Previous Next



Triggerを設定



Preview ▶ Run

Create serverless application ✕

Function trigger

none ▼

none
API Gateway

Region: us-east-1

Previous Next



MemoryとRoleを選択

Create serverless application ×

Memory (MB): ▼

Your function is allocated CPU proportional to the memory configured.

Role: ▼

Region: us-east-1

[Previous](#) [Next](#)



少し待つと作成完了



The screenshot shows the AWS Cloud9 IDE interface. The top menu bar includes 'AWS Cloud9', 'File', 'Edit', 'Find', 'View', 'Goto', 'Run', 'Tools', 'Window', 'Support', 'Preview', and a 'Run' button. The left sidebar has three sections: 'Environment' showing a project tree with folders like 'kumakura-dev', '.c9', and 'LambdaTest1', and a file 'lambda_function.py'; 'Navigate' showing a sub-tree with 'venv', '__init__.py', 'requirements.txt', and 'template.yaml'; and 'Commands' showing 'README.md', 's3_test.py', and 'test.sh'. The main editor window has tabs for 'Welcome', 'lambda_function', 'requirements.txt', and 'template.yaml'. The 'lambda_function' tab is active, displaying the following Python code:

```
1 import json
2
3 print('Loading function')
4
5
6 def lambda_handler(event, context):
7     #print("Received event: " + json.dumps(event, indent=2))
8     print("value1 = " + event['key1'])
9     print("value2 = " + event['key2'])
10    print("value3 = " + event['key3'])
11    return event['key1'] # Echo back the first key value
12    #raise Exception('Something went wrong')
13
```



Lambdaのコンソール

Lambda > 関数 > cloud9-LambdaTest1-LambdaTest1-R7WX7OZPF9WY

ARN - arn:aws:lambda:us-east-1:516977277448:function:cloud9-LambdaTest1-LambdaTest1-R7WX7OZPF9WY


cloud9-LambdaTest1-LambdaTest1-R7WX7OZPF9WY

スロットリング

限定条件 ▼

アクション ▼

テスト

 この関数は CloudFormation スタック **cloud9-LambdaTest1** に属しています。CloudFormation コンソールに移動して、このスタックを管理します。

設定

モニタリング

▼ Designer

トリガーの追加

下のリストのトリガーをクリックして、トリガーを関数に追加します。

API Gateway

AWS IoT



左側のリストからトリガーを追加します



cloud9-LambdaTest1-LambdaTest1-R7WX7OZPF9WY



AWS Lambda

IDEからローカルとリモートのLambda関数が見れる

The screenshot displays the AWS Cloud9 IDE interface. The main editor shows a Python file named `lambda_function.py` with the following code:

```
1 import json
2 import requests
3
4 print('Loading function')
5
6
7 def lambda_handler(event, context):
8     #print("Received event: " + json.dumps(event, indent=2))
9     print("value1 = " + event['key1'])
10    print("value2 = " + event['key2'])
11    print("value3 = " + event['key3'])
12    # result = requests.get('https://www.google.co.jp')
13    # print(result)
14    return event['key1'] # Echo back the first key value
15    #raise Exception('Something went wrong')
16
```

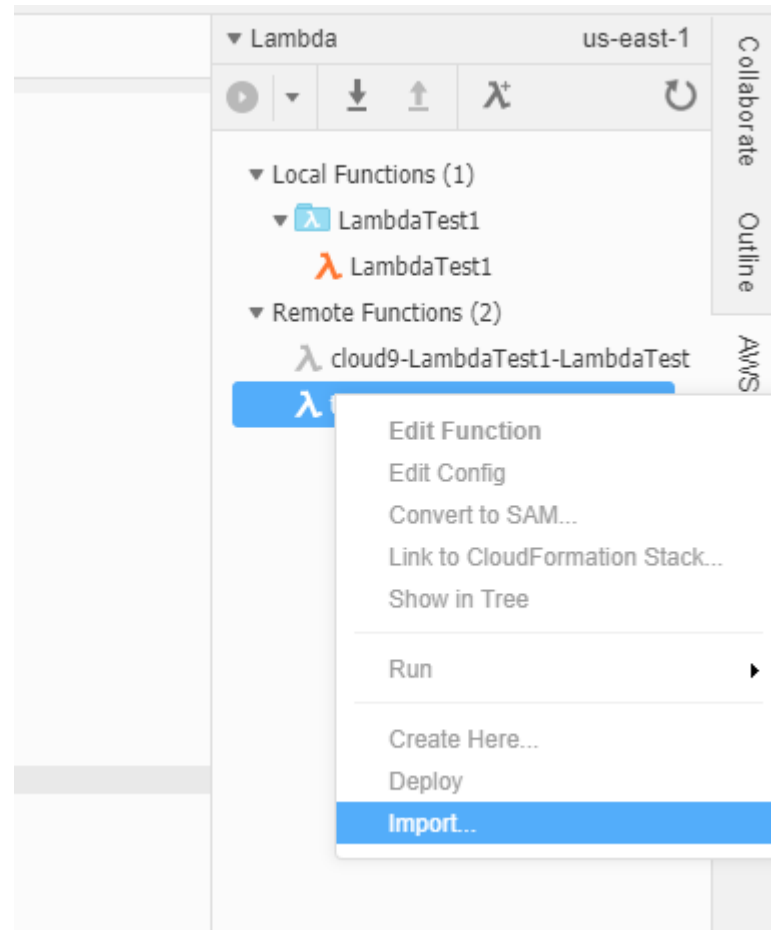
The terminal window at the bottom shows the output of a `ls -la` command:

```
total 20
-rw-r--r-- 1 ec2-user ec2-user  0 Jul 4 05:59 __init__.py
drwxr-xr-x 2 ec2-user ec2-user 4096 Jul 4 06:58 LambdaTest1
drwxr-xr-x 2 ec2-user ec2-user 4096 Jul 4 06:21 LambdaTest2
-rw-r--r-- 1 ec2-user ec2-user  45 Jul 4 06:52 requirements.txt
-rw-r--r-- 1 ec2-user ec2-user  475 Jul 4 05:59 template.yaml
drwxr-xr-x 6 ec2-user ec2-user 4096 Jul 4 05:49 venv
```

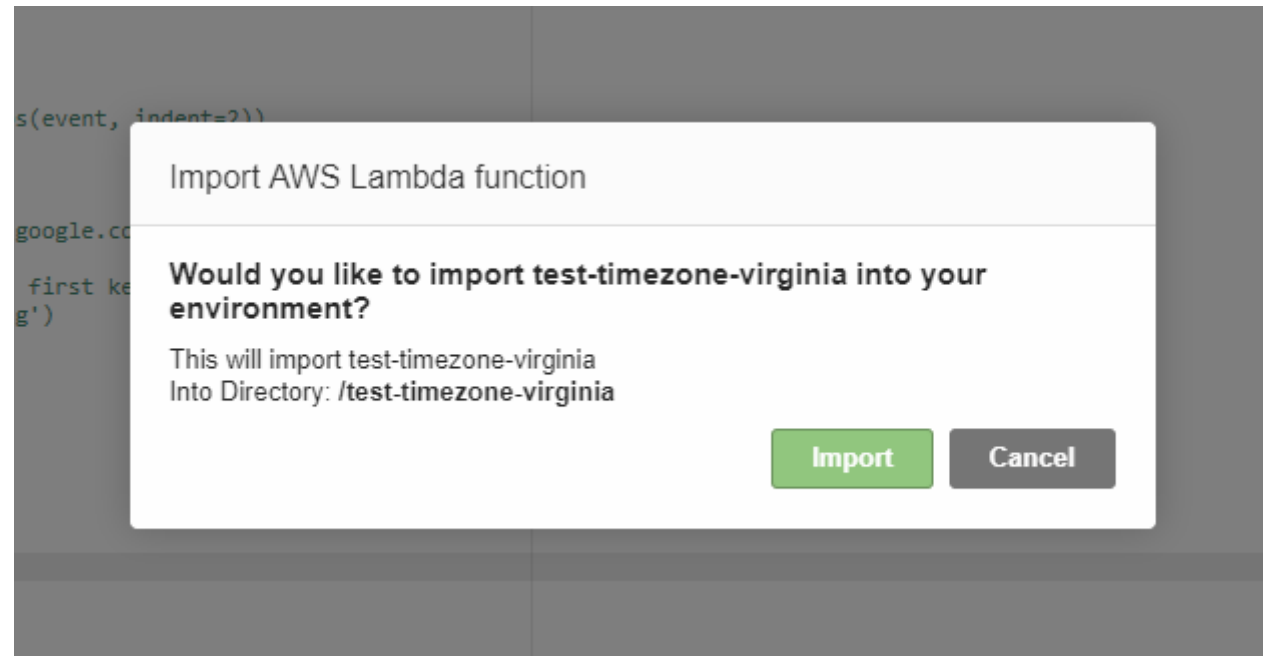
The right sidebar shows the AWS Lambda console view for the `us-east-1` region, displaying a list of functions:

- Local Functions (2)
 - test-timezone-virginia
 - LambdaTest1
- Remote Functions (2)
 - cloud9-LambdaTest1-LambdaTest
 - test-timezone-virginia

リモートのLambda関数もインポート出来たり



リモートのLambda関数もインポート出来たり



AWS Cloud9 File Edit Find View Goto Run Tools Window Support Preview Run

Environment

- kumakura-dev
 - LambdaTest1
 - LambdaTest1
 - lambda_function.py
 - LambdaTest2
 - lambda_function.py
 - venv
 - __init__.py
 - requirements.txt
 - template.yaml
 - test-timezone-virginia
 - README.md
 - s3_test.py
 - test.sh

Commands

lambda_function

```
1 from datetime import datetime
2 import logging
3
4 def lambda_handler(event, context):
5     # TODO implement
6     logging.info("datetime.now() = {}".format(datetime.now()))
7     logging.info("datetime.utcnow() = {}".format(datetime.utcnow()))
8     logging.info("datetime.fromtimestamp() = {}".format(datetime.fromtimestamp(1504591821)))
9     print("datetime.now() = {}".format(datetime.now()))
10    print("datetime.utcnow() = {}".format(datetime.utcnow()))
11    print("datetime.fromtimestamp() = {}".format(datetime.fromtimestamp(1504591821)))
12    return 'Hello from Lambda'
```

1:1 Python Spaces: 4

bash - "ip-172-31" Immediate (Java) LambdaTest1/La

```
total 20
drwxr-xr-x 6 ec2-user ec2-user 4096 Jul  4 05:59 LambdaTest1
-rw-r--r-- 1 ec2-user ec2-user  579 Jul  3 11:33 README.md
-rw-r--r-- 1 ec2-user ec2-user   14 Jul  3 12:32 s3_test.py
-rw-r--r-- 1 ec2-user ec2-user   51 Jul  3 12:35 test.sh
drwxr-xr-x 2 ec2-user ec2-user 4096 Jul  4 06:58 test-timezone-virginia
kumakura:~/environment $
```

Collaborate Outline AWS Resources Debugger

us-east-1

- Local Functions (2)
 - test-timezone-virginia
 - LambdaTest1
 - LambdaTest1
- Remote Functions (2)
 - cloud9-LambdaTest1-LambdaTest
 - test-timezone-virginia

ローカル/リモートでの実行とか

The screenshot displays the AWS Cloud9 IDE interface. The top menu bar includes options like File, Edit, Find, View, Goto, Run, Tools, Window, Support, Preview, Run Last, and Share. The main workspace is divided into several panels:

- Environment:** Shows a project structure with folders like 'kumakura-dev', 'LambdaTest1', and 'LambdaTest2', and files like 'lambda_function.py', 'requirements.txt', and 'template.yaml'.
- Editor:** Displays the code for 'LambdaTest1' with a 'Run' button circled in red. The code is a Python lambda function that returns 'bbb'.
- Test payload:** Shows a test payload: `1 [{"key1": "bbb", "key2": "ddd", "key3": "eee"}]`.
- Execution results:** Shows the response 'bbb' and function logs: 'Loading function', 'value1 = bbb', 'value2 = ddd', 'value3 = eee'. The request ID is '8d3d29ba-5fd4-479b-8be1-4c82f0e11b26'.
- Terminal:** Shows the output of 'which pip' command: '/usr/bin/pip'.
- Right sidebar:** Lists 'Local Functions (2)' including 'test-timezone-virginia' and 'LambdaTest1', and 'Remote Functions (2)' including 'cloud9-LambdaTest1-LambdaTest' and 'test-timezone-virginia'.

共同コーディング

- ▶ ※実際の画面参照
- ▶ 強調表示や選択箇所などは共有ユーザーに表示される
- ▶ Runの結果は共有ユーザーには表示されなかった
- ▶ Terminalも連動される
- ▶ チャットも使えるからIDEの画面から切り替えずにコミュニケーション可能



良かった点

- ▶ Lambda関数の作成・デプロイが簡単
- ▶ 教育の際などの、ペアプログラミングで使えるかも
- ▶ ローカルに環境を用意する手間が無い
- ▶ AWSの各種リソースとの親和性が高い
- ▶ EC2の料金以外は無料



いまいちな点

- ▶ IDEとしての補完機能などがIntelliJ IDEAなどに比べるとやはり弱い
 - ▶ Pythonでしか試していないが、docstringすら補完してくれず
- ▶ pythonでのrequirements.txtから自動でライブラリインストールしてパッケージングしてデプロイ、とかまでしてくれるなら楽だったかも
- ▶ 別AWSアカウントユーザーを招待するのが手間がかかった
- ▶ AWSアカウントを持つユーザーしか招待できない
 - ▶ AWSの製品の1種なので仕方ないが



まとめ

現状では普段使いのIDEとして使用したい！とまでは思えなかったが、AWS自身で提供するIDEということもあり、今後のバージョンアップに期待したい

