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

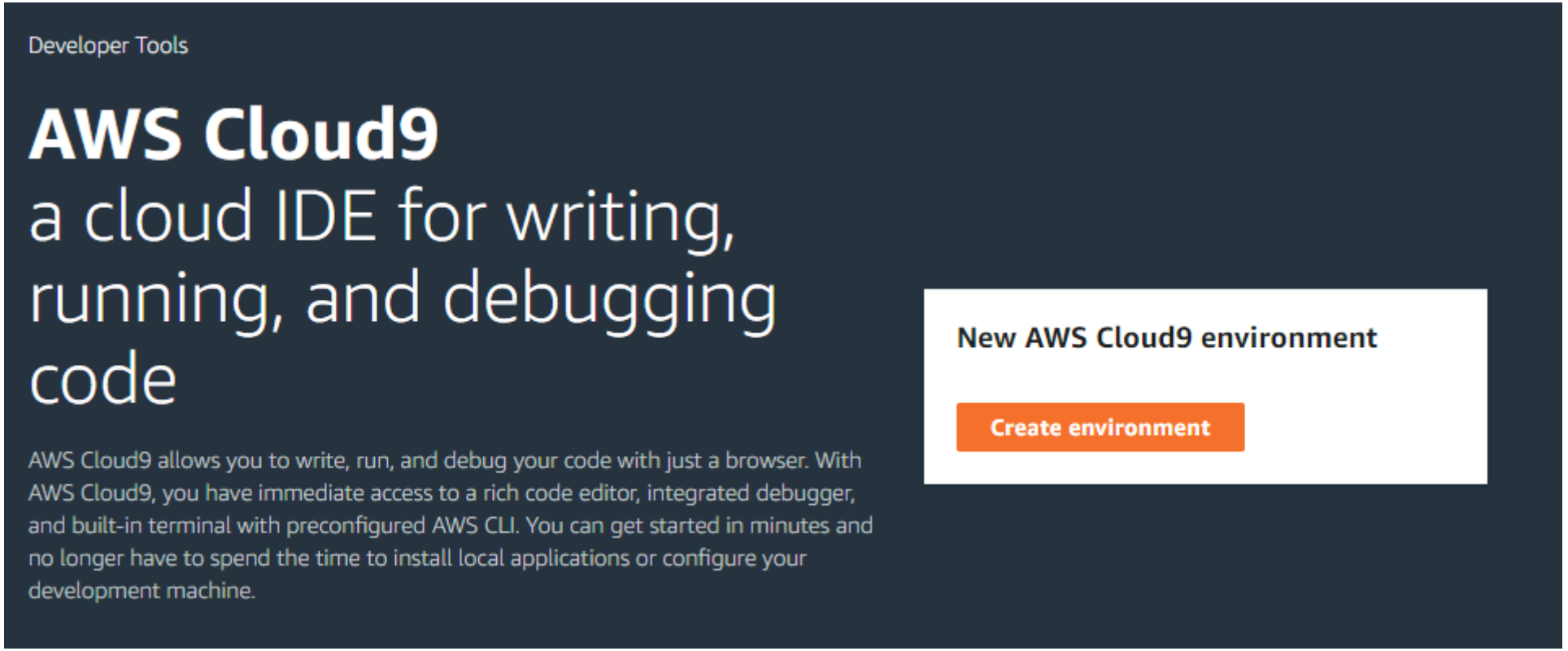
- ブラウザのみでコードを記述出来る
- リアルタイムで共同コーディング出来る
- 40以上の言語をサポート
  - Node.js (JavaScript)、Python、PHP、Ruby、Go、C++・・・など
  - https://docs.aws.amazon.com/ja_jp/cloud9/latest/user-guide/language-support.html
- 2018年7月現在、東京リージョンには未対応

気になったのが

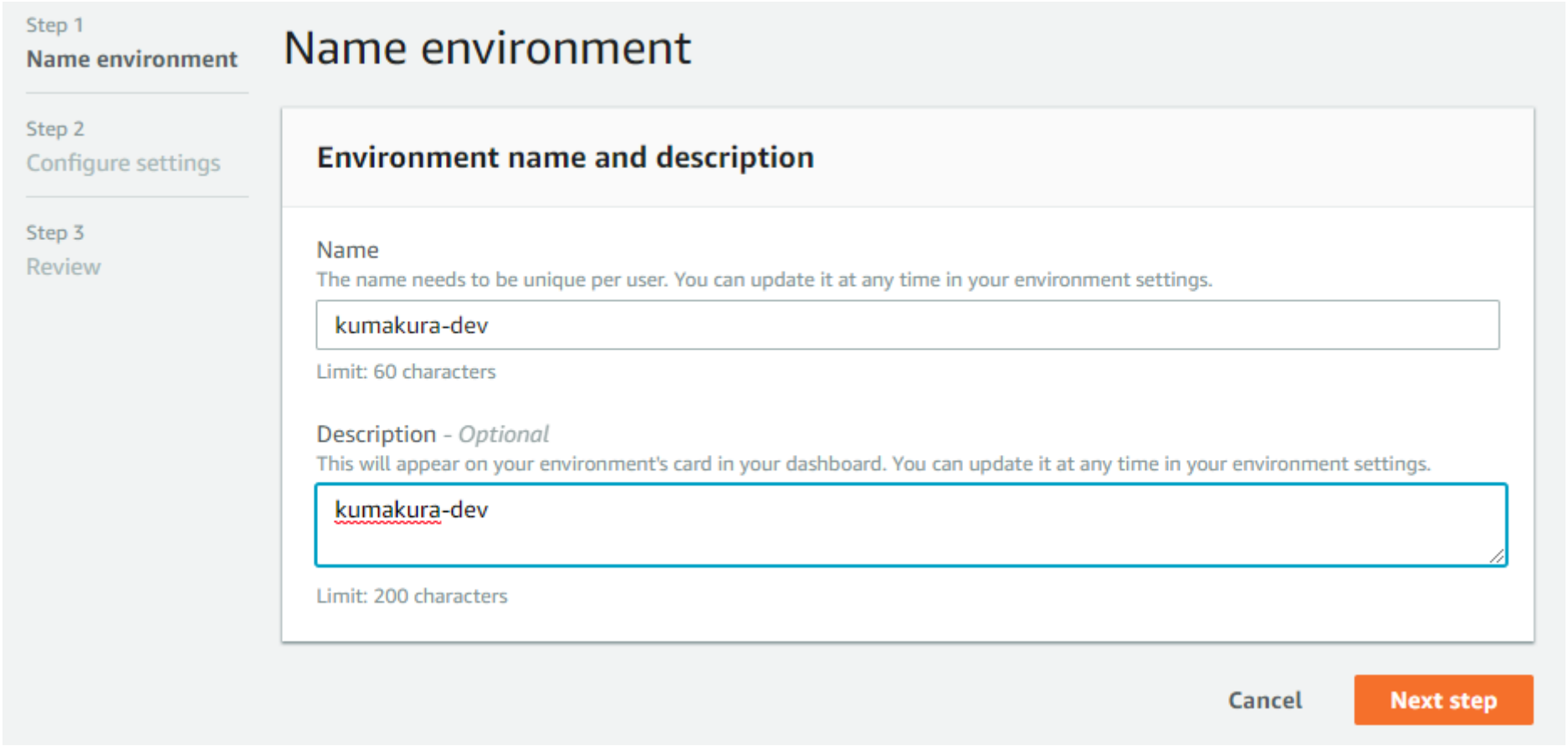
- **サーバーレスアプリケーションを簡単に構築**
  - AWS Lambda 関数をローカルでテストおよびデバッグするための環境を利用できるらしい
  - Lambdaのテストはめんどくさいので簡単に出来たら嬉しそう
- AWS自身の製品なので、AWSとの親和性が高そうなのと、今後の改良にも期待が持てるのではないか

2. 設定方法

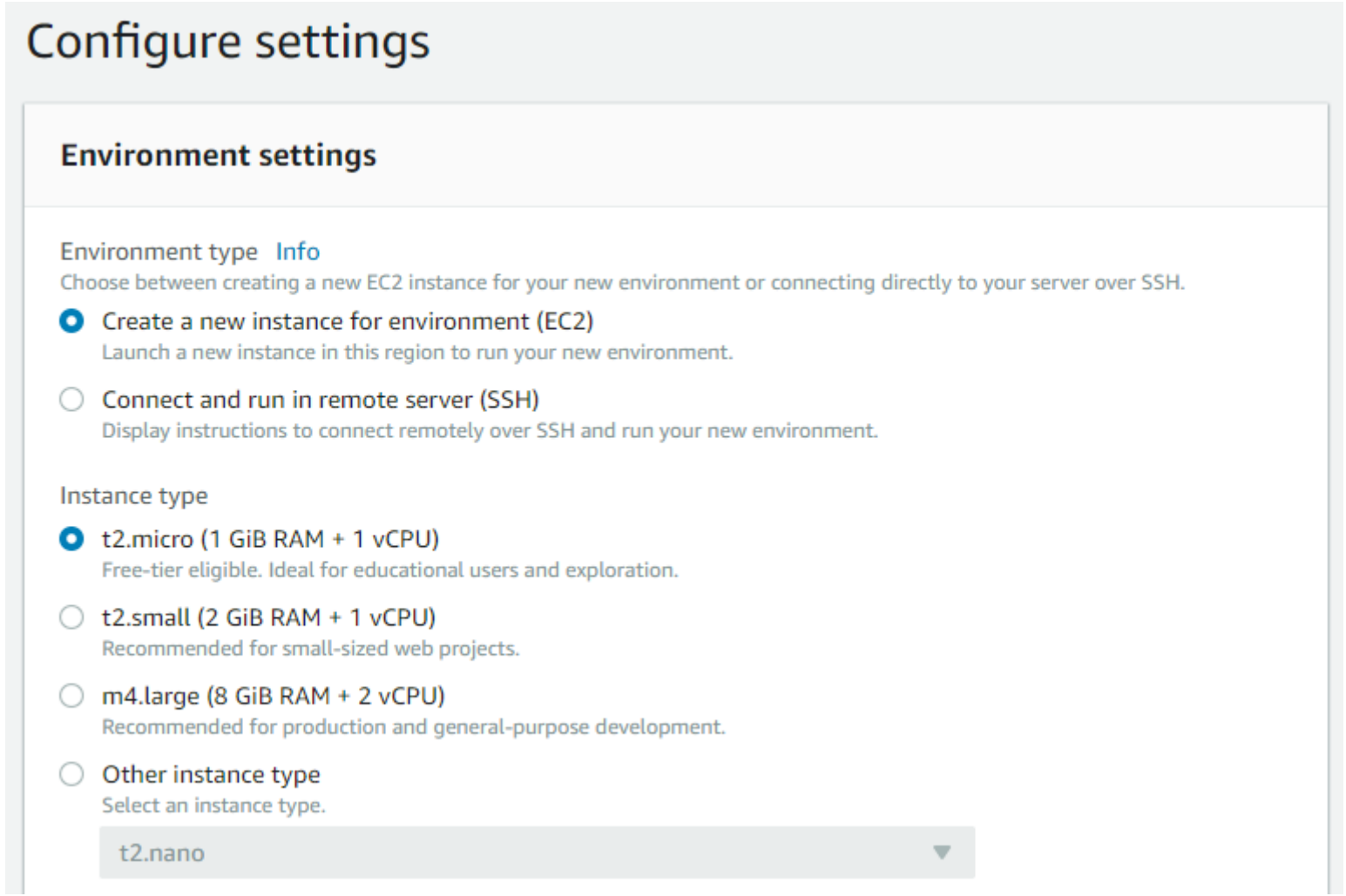
サービスからCloud9を選択



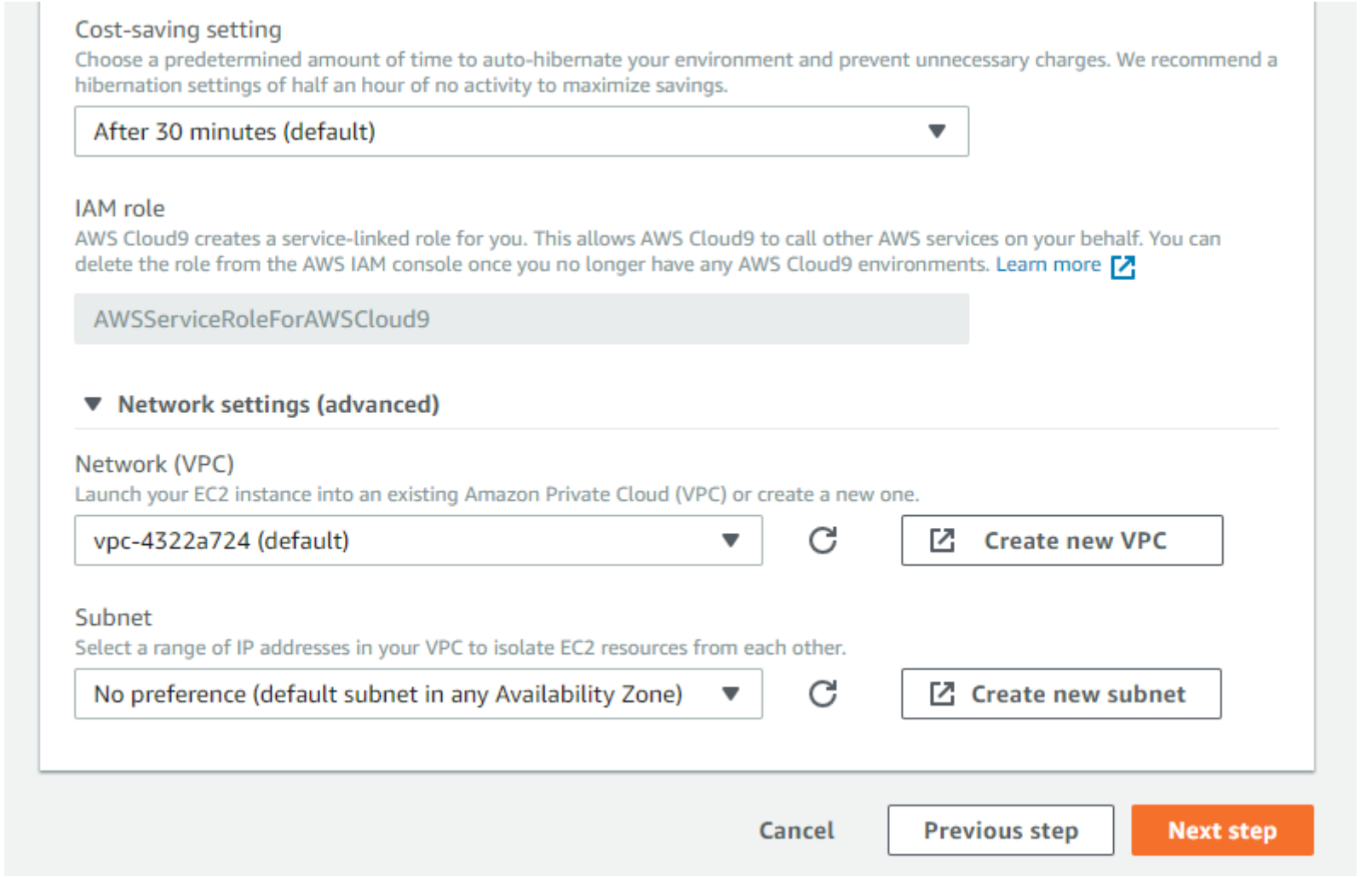
とりあえず環境作成


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.
kumakura-dev
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.
kumakura-dev
Limit: 200 characters
Cancel
Next step

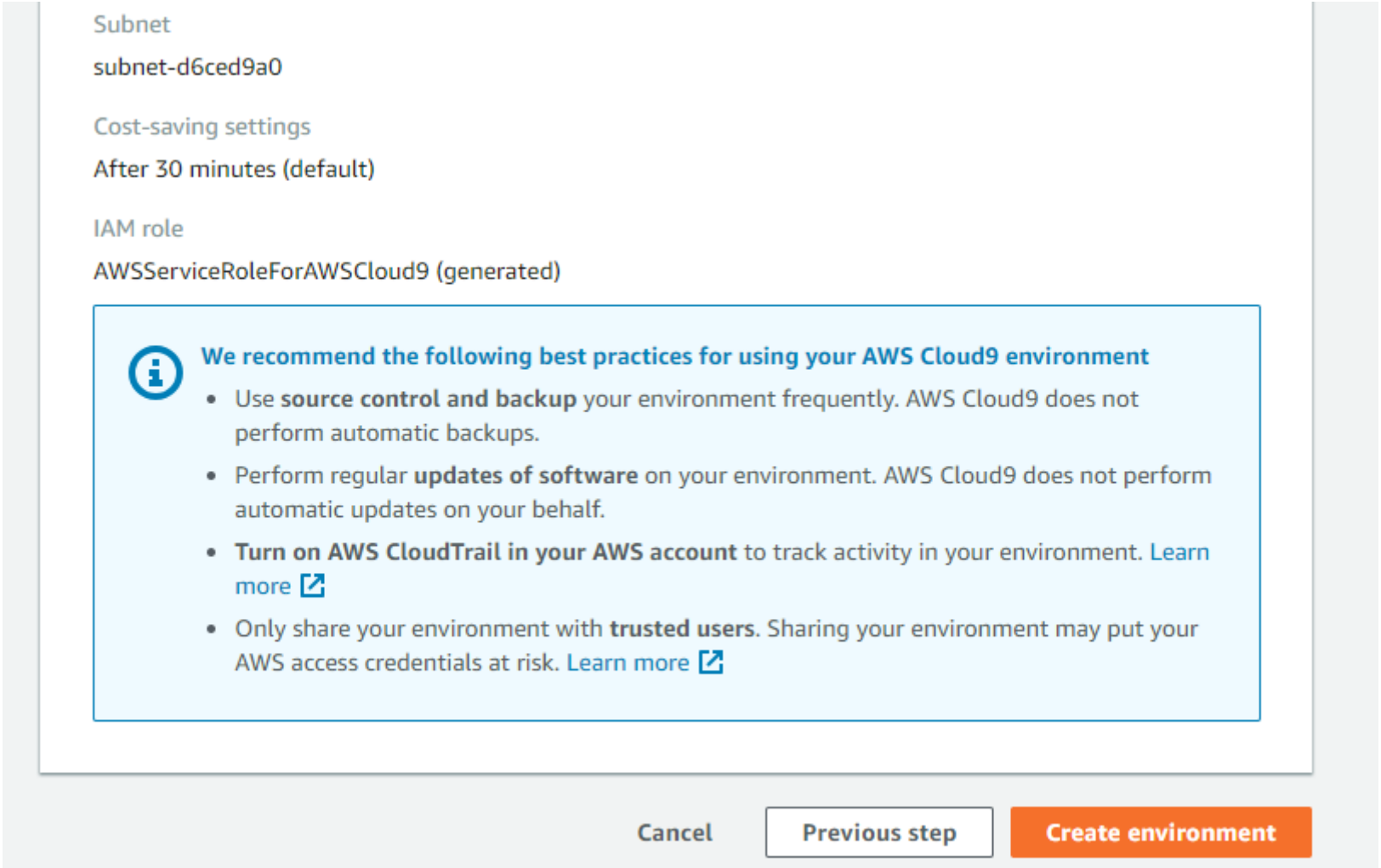
Cloud9用のEC2インスタンスを設定



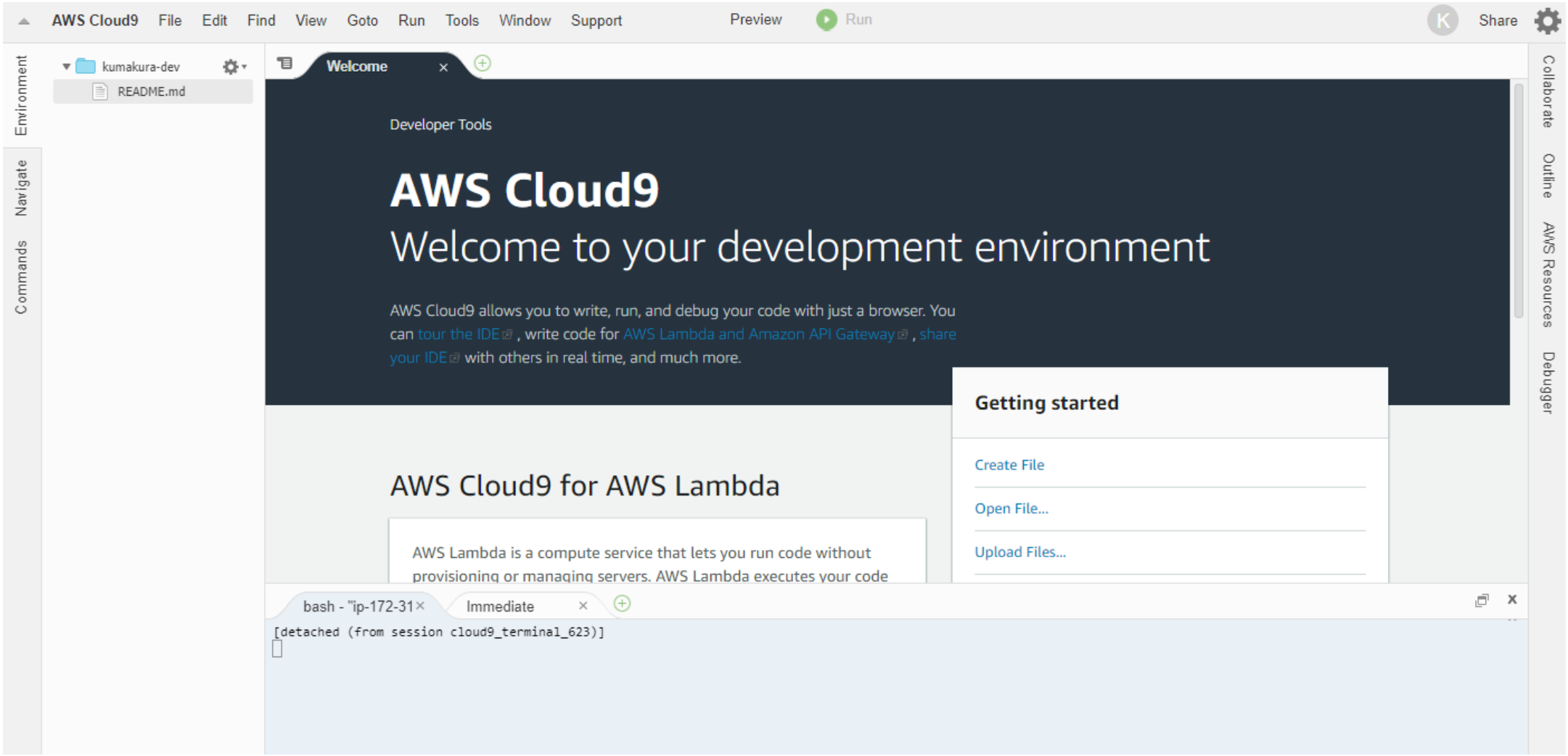
自動シャットダウンの設定とか



確認画面

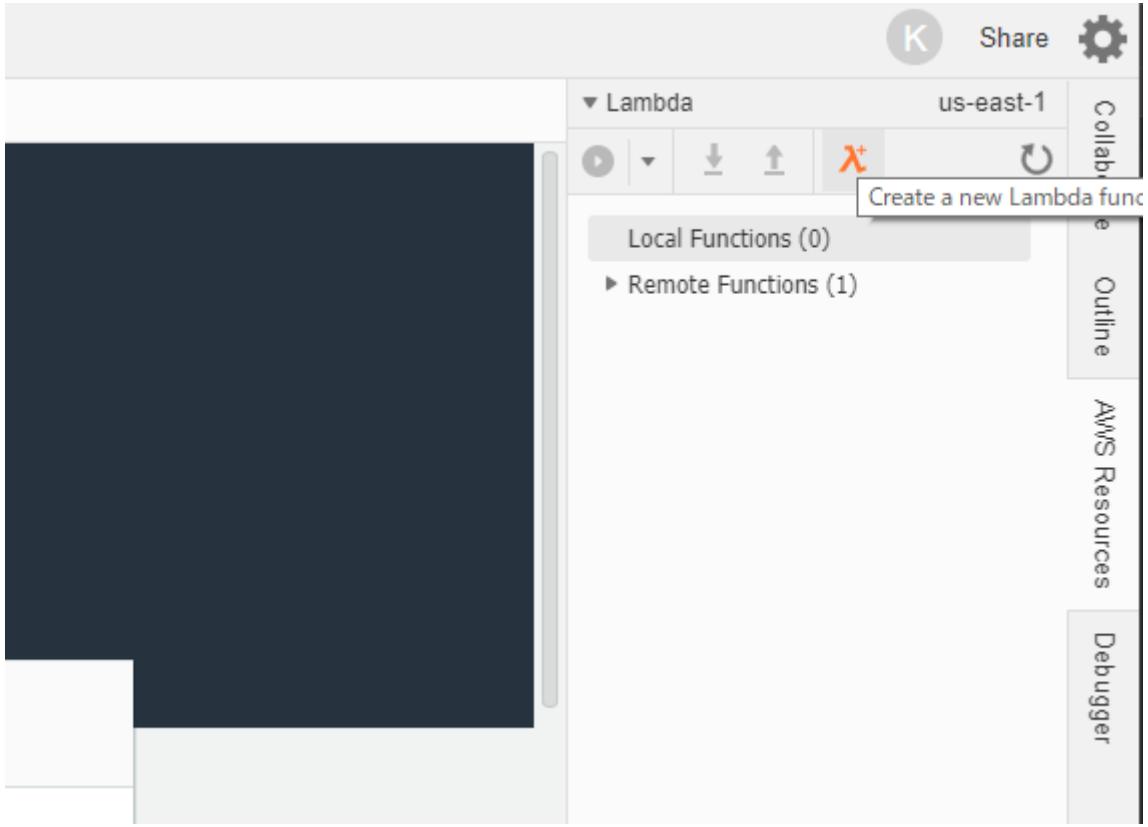


作成できました

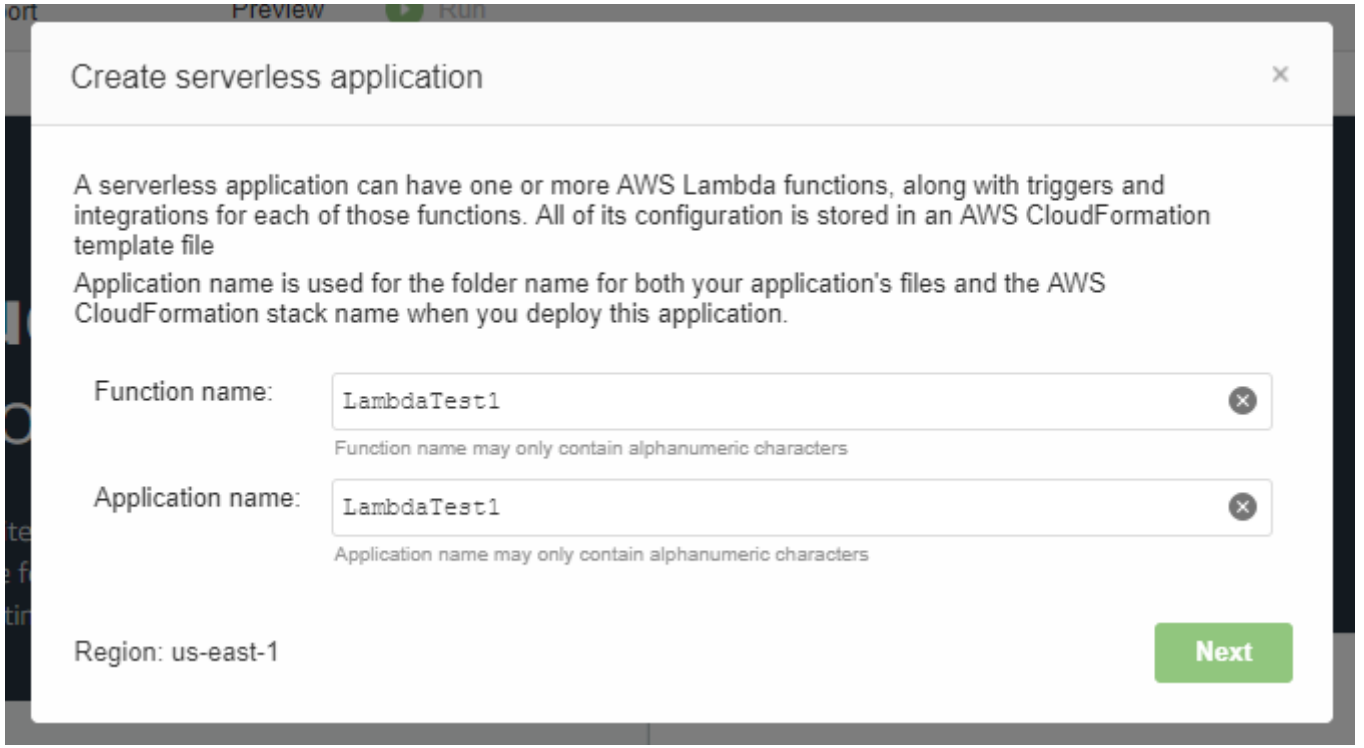


3. 使い方

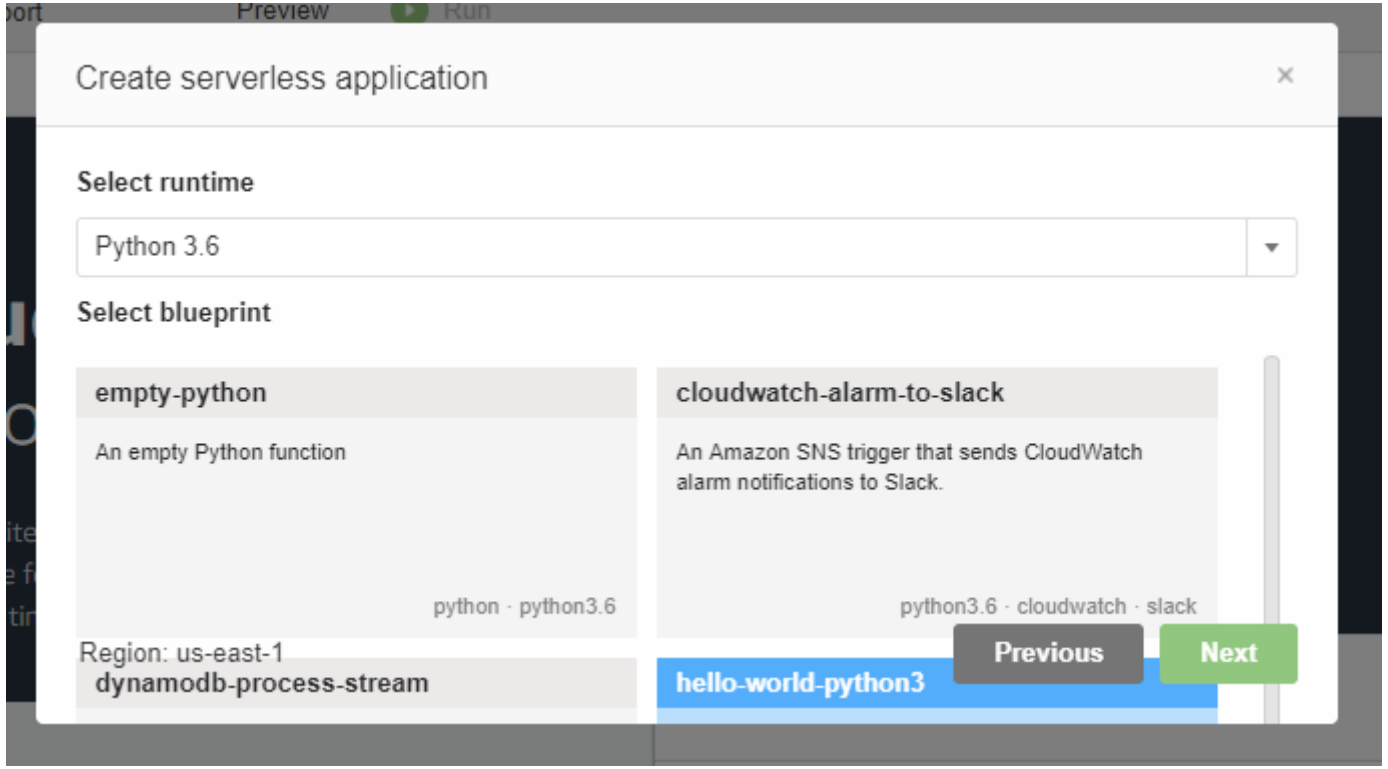
Lambdaを作成する



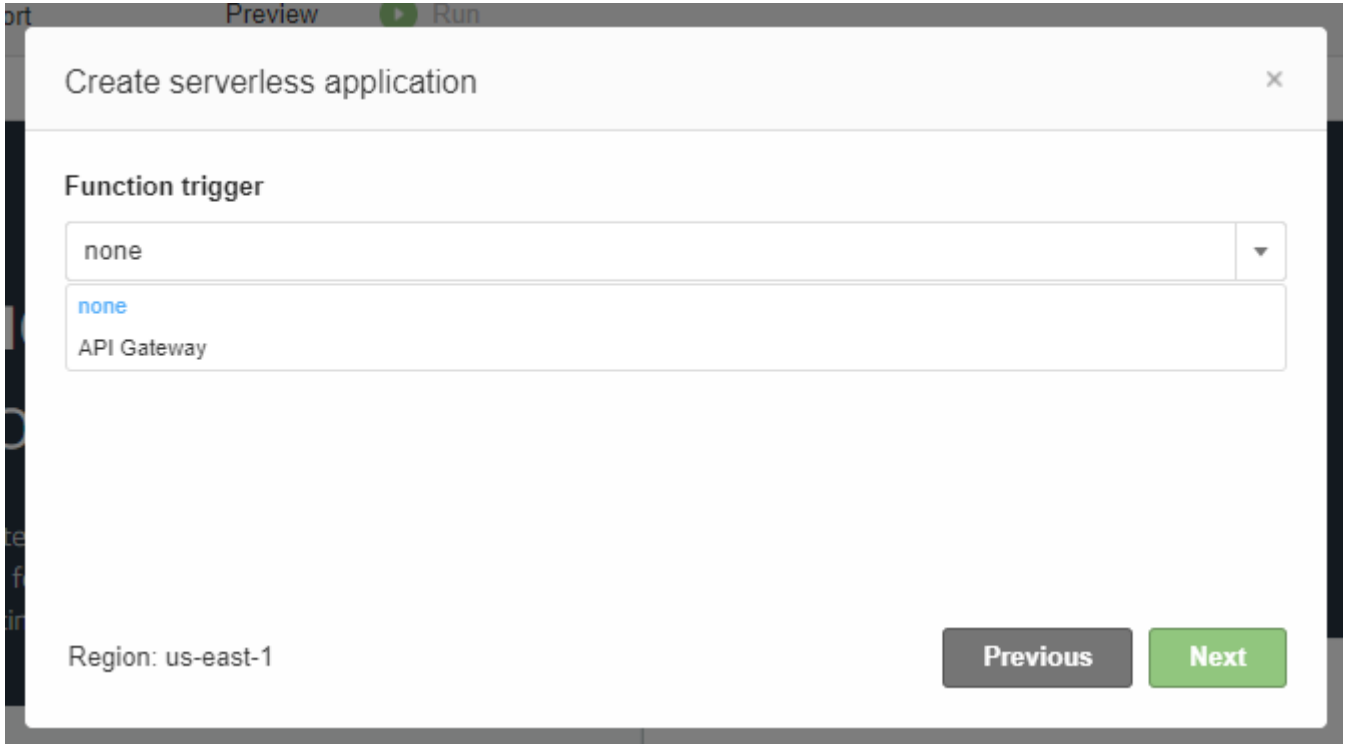
Lambdaを作成する



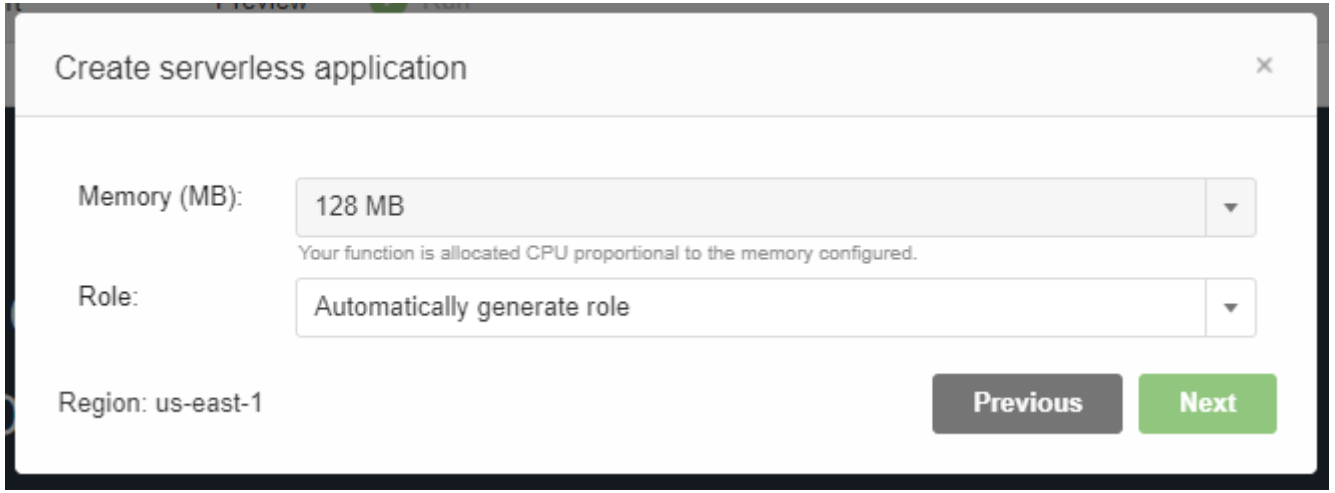
RuntimeやblueprintをSelect



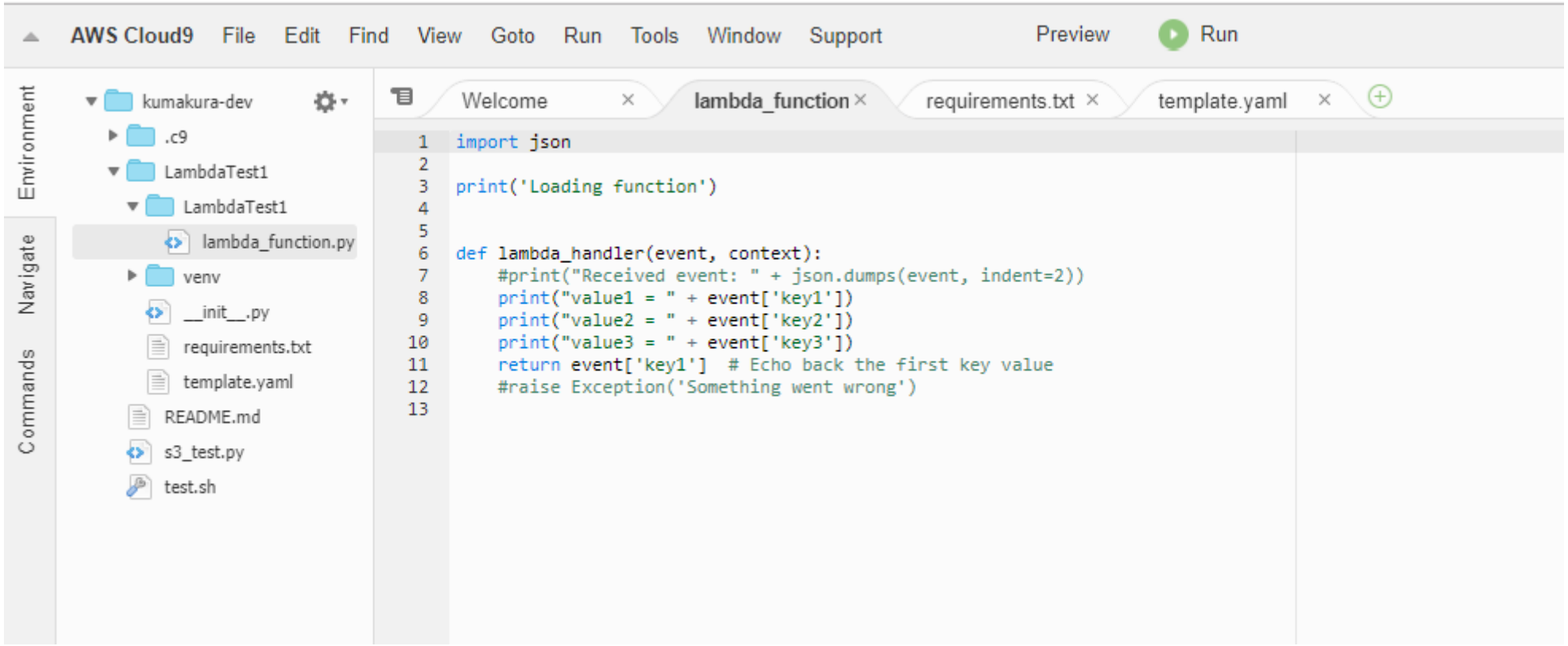
Triggerを設定


Preview
Run
Create serverless application
Function trigger
none
none
API Gateway
Region: us-east-1
Previous
Next

MemoryとRoleを選択

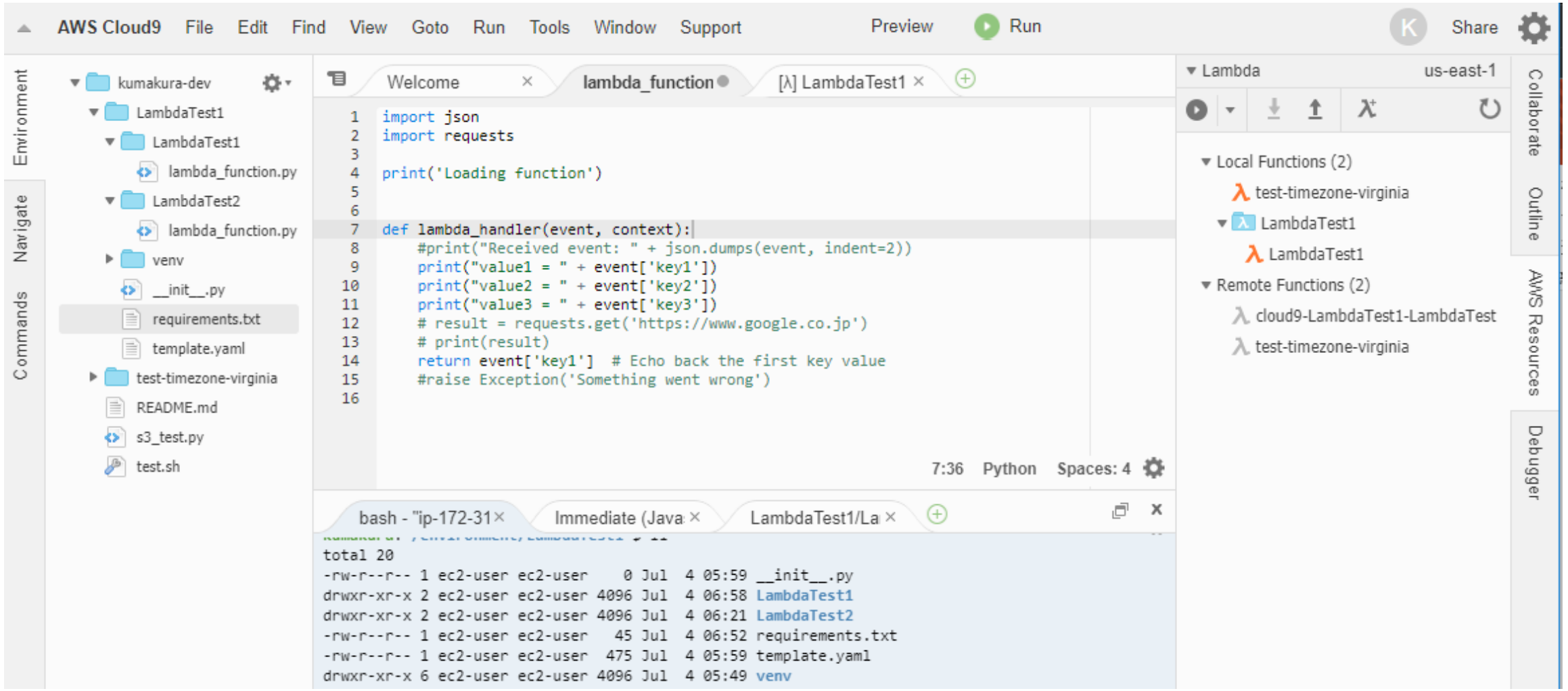


少し待つと作成完了

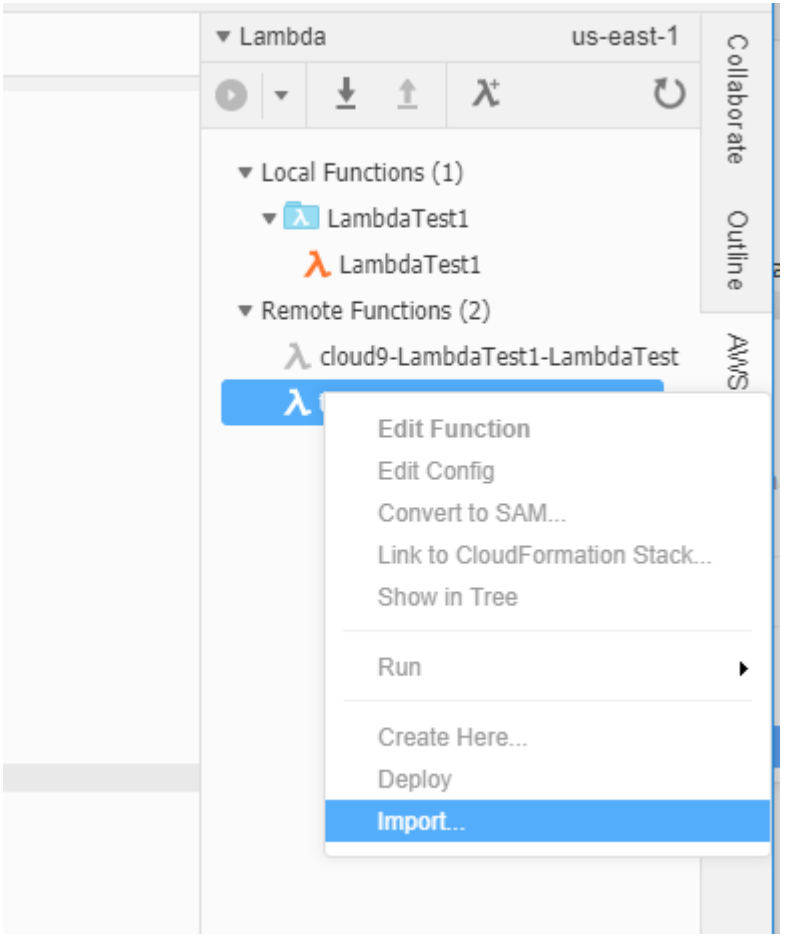

AWS Cloud9
File
Edit
Find
View
Goto
Run
Tools
Window
Support
Preview
Run
Environment
Navigate
Commands
kumakura-dev
.c9
LambdaTest1
LambdaTest1
lambda_function.py
venv
__init__.py
requirements.txt
template.yaml
README.md
s3_test.py
test.sh
Welcome
lambda_function
requirements.txt
template.yaml
1 import 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

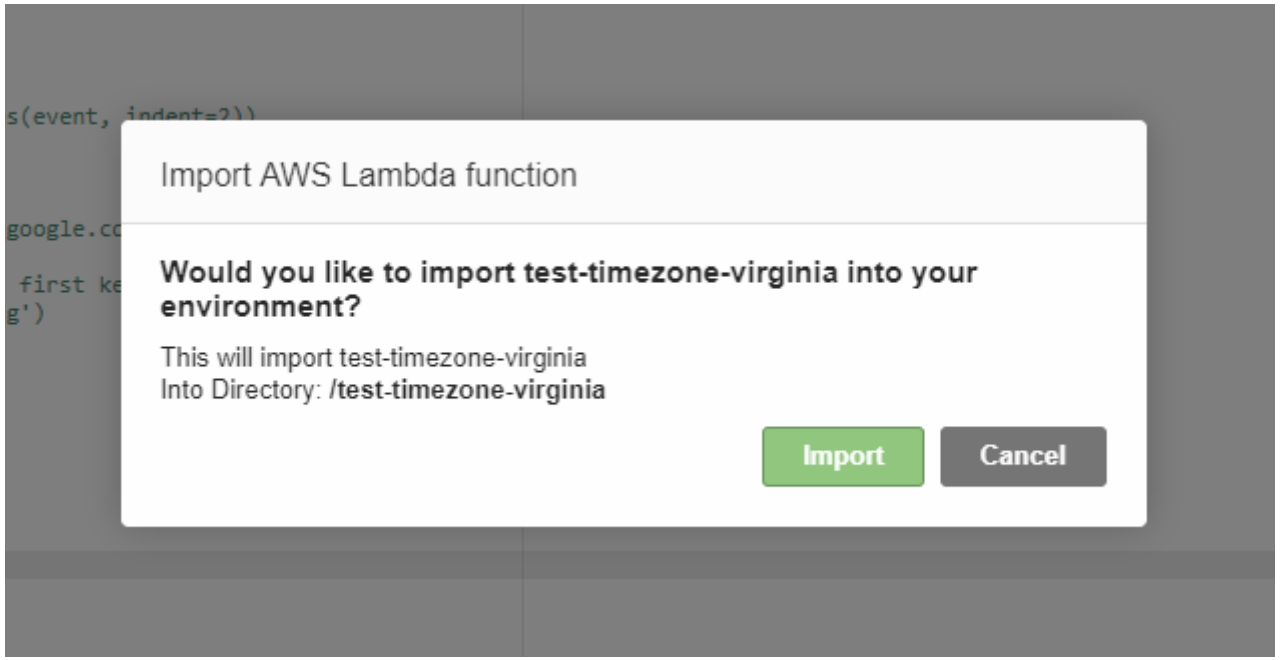
IDEからローカルとリモートのLambda関数が見れる

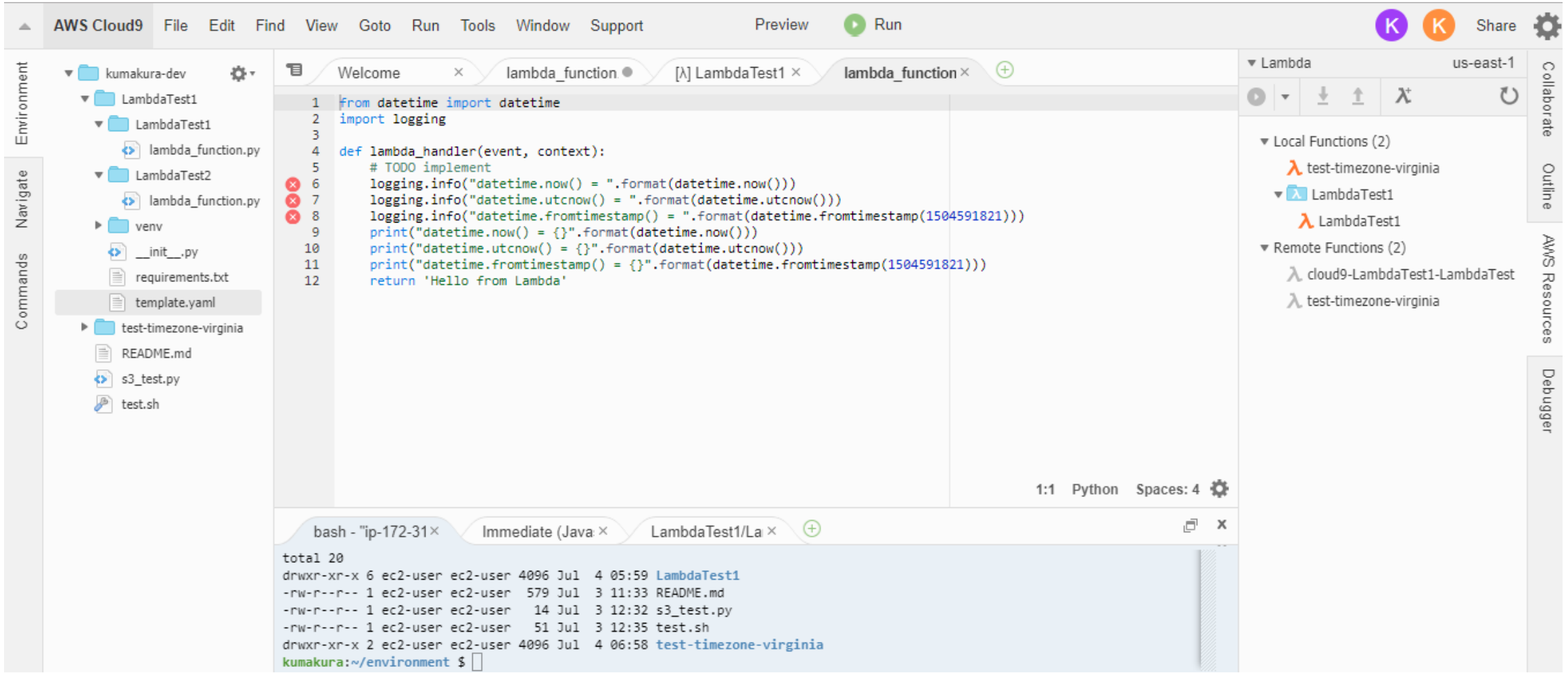


リモートのLambda関数もインポート出来たり

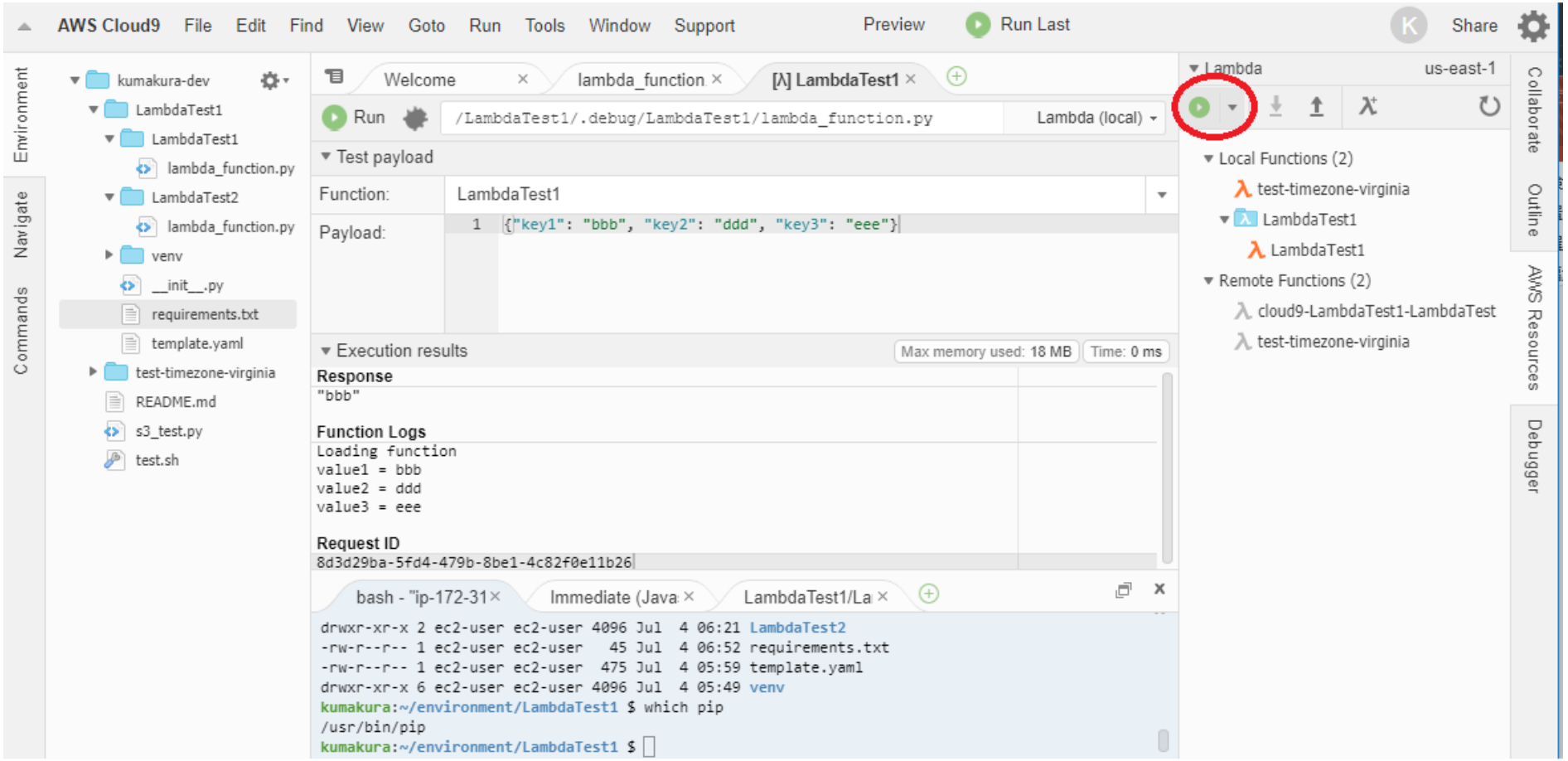


リモートのLambda関数もインポート出来たり




AWS Cloud9
File
Edit
Find
View
Goto
Run
Tools
Window
Support
Preview
Run
Share
Environment
Navigate
Commands
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
Welcome
lambda_function
[λ] LambdaTest1
lambda_function
from datetime import datetime
import logging
def lambda_handler(event, context):
# TODO implement
logging.info("datetime.now() = ".format(datetime.now()))
logging.info("datetime.utcnow() = ".format(datetime.utcnow()))
logging.info("datetime.fromtimestamp() = ".format(datetime.fromtimestamp(1504591821)))
print("datetime.now() = {}".format(datetime.now()))
print("datetime.utcnow() = {}".format(datetime.utcnow()))
print("datetime.fromtimestamp() = {}".format(datetime.fromtimestamp(1504591821)))
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 $
Lambda
us-east-1
Local Functions (2)
test-timezone-virginia
LambdaTest1
LambdaTest1
Remote Functions (2)
cloud9-LambdaTest1-LambdaTest
test-timezone-virginia
Collaborate
Outline
AWS Resources
Debugger

ローカル/リモートでの実行とか



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