

サーバーレス開発：環境準備・CI/CD

Beyond Beginner



@_kensh



Who am I?

- Name
 - Kensuke Shimokawa
- Company
 - Amazon Web Services Japan K.K.
- Role
 - Serverless Specialist Solutions Architect



@_kensh

Agenda

- Serverlessの開発環境を整える
- Serverlessのデプロイ環境を整える
- ServerlessのCI/CDパイプラインを整える
- まとめ



A group of people in a meeting looking at a laptop screen. The scene is brightly lit, suggesting a modern office environment. The text is overlaid on a semi-transparent dark blue band across the middle of the image.

Serverlessの開発環境を整える

AWS提供の総合的なツールセット

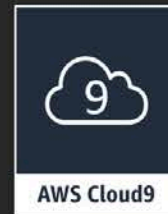
CI/CD Tools



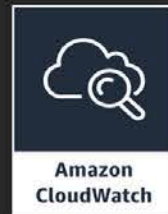
Infrastructure as Code



IDE



Monitoring & Tracing



Web Apps



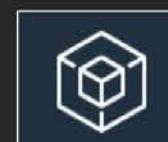
IDE and DevOps Toolkits



CLI and Scripting Tools



Languages



Mobile



SDKs



AWS提供の総合的なツールセット

統合開発環境(IDE)

IDE and DevOps Toolkits



Visual Studio Code



IntelliJ



PyCharm



Visual Studio



Eclipse



AWS Toolkit

AWS Toolkit

- オープンソースプラグイン
- AWS上でのアプリケーションの作成、デバッグ、デプロイを容易にします。
- ステップ実行によるデバッグ、および IDE からのデプロイを含む、サーバーレスアプリケーションの統合開発環境を提供します。



AWS Toolkit

AWS提供の総合的なツールセット

AWS Toolkit 対応 IDE

IDE and DevOps Toolkits



Visual Studio Code



IntelliJ



PyCharm



Visual Studio



Eclipse

- 雛形からプロジェクトを作成
- ステップ実行などのデバッグ支援
- IDE/エディタからのデプロイ

Visual Studio Code
Software

Japan Past 12 months

Jan 19 – 25, 2020

Visual Studio Code

100

Eclipse

78

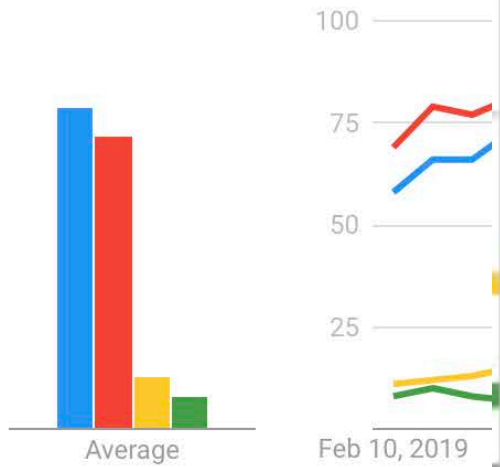
IntelliJ IDEA

16

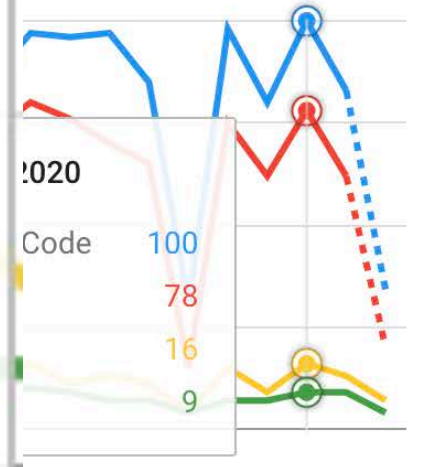
PyCharm

9

Interest over time



Download Refresh Share



How to install Visual Studio Code

<https://code.visualstudio.com/docs/setup/mac>



Visual Studio Code

Docs

Updates

Blog

API

Extensions

FAQ



Search Docs

Version 1.41 is now available! Read about the new features and fixes from November.

Overview

SETUP

Overview

Linux

macOS

Windows

Network

Additional
Components

GET STARTED

Visual Studio Code on macOS

Edit

Installation

1. [Download Visual Studio Code](#) for macOS.
2. Double-click on the downloaded archive to expand the contents.
3. Drag **Visual Studio Code.app** to the **Applications** folder, making it available in the **Launchpad**.
4. Add VS Code to your Dock by right-clicking on the icon to bring up the context menu and choosing **Options, Keep in Dock**.

「AWS Toolkit for Visual Studio Code」を検索してInstall

The screenshot shows the Visual Studio Code interface with the Extensions Marketplace open. The search bar contains the text "AWS Toolkit for Visual Studio Code". The search results list several extensions, with the top result being the "AWS Toolkit for Visual Studio Code" extension by Amazon Web Services. The extension details show version 1.5.0, 93K downloads, and a 3-star rating. Below it are other extensions like "Chinese (Simplified) Language Pack for Visual Studio Code" and "Visual Studio IntelliCode". The "Install" button is visible for the AWS Toolkit extension. The terminal window at the bottom shows a bash prompt and the command "38f9d36982e7:sample shimokk\$".

EXTENSIONS: MARKETPLACE

AWS Toolkit for Visual Studio Code

AWS Toolkit for Visual Studio Code 1.5.0 93K ★ 3
An extension for working with Amazon Web Services in Visu...
Amazon Web Services

Chinese (Simplified) Language Pack for Visual Studio Code 1.42.1 3.8M ★ 5
中文(简体)
Microsoft **Install**

Visual Studio IntelliCode 1.2.4 3.9M ★ 4.5
AI-assisted development
Microsoft

Japanese Language Pack for Visual Studio Code 1.42.1 893K ★ 4.5
日本語
Microsoft **Install**

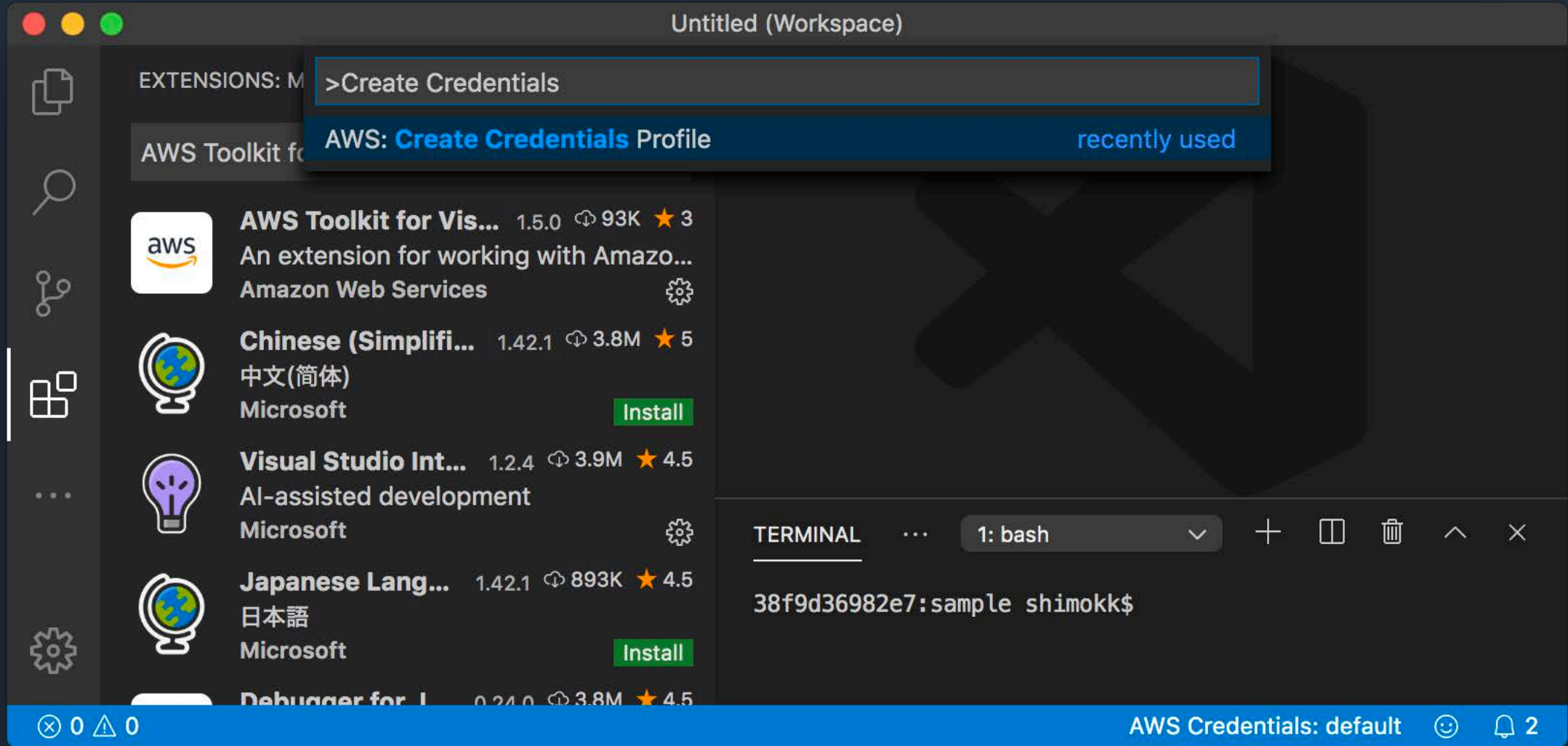
Debugger for Java 0.24.0 3.8M ★ 4.5

TERMINAL 1: bash
38f9d36982e7:sample shimokk\$

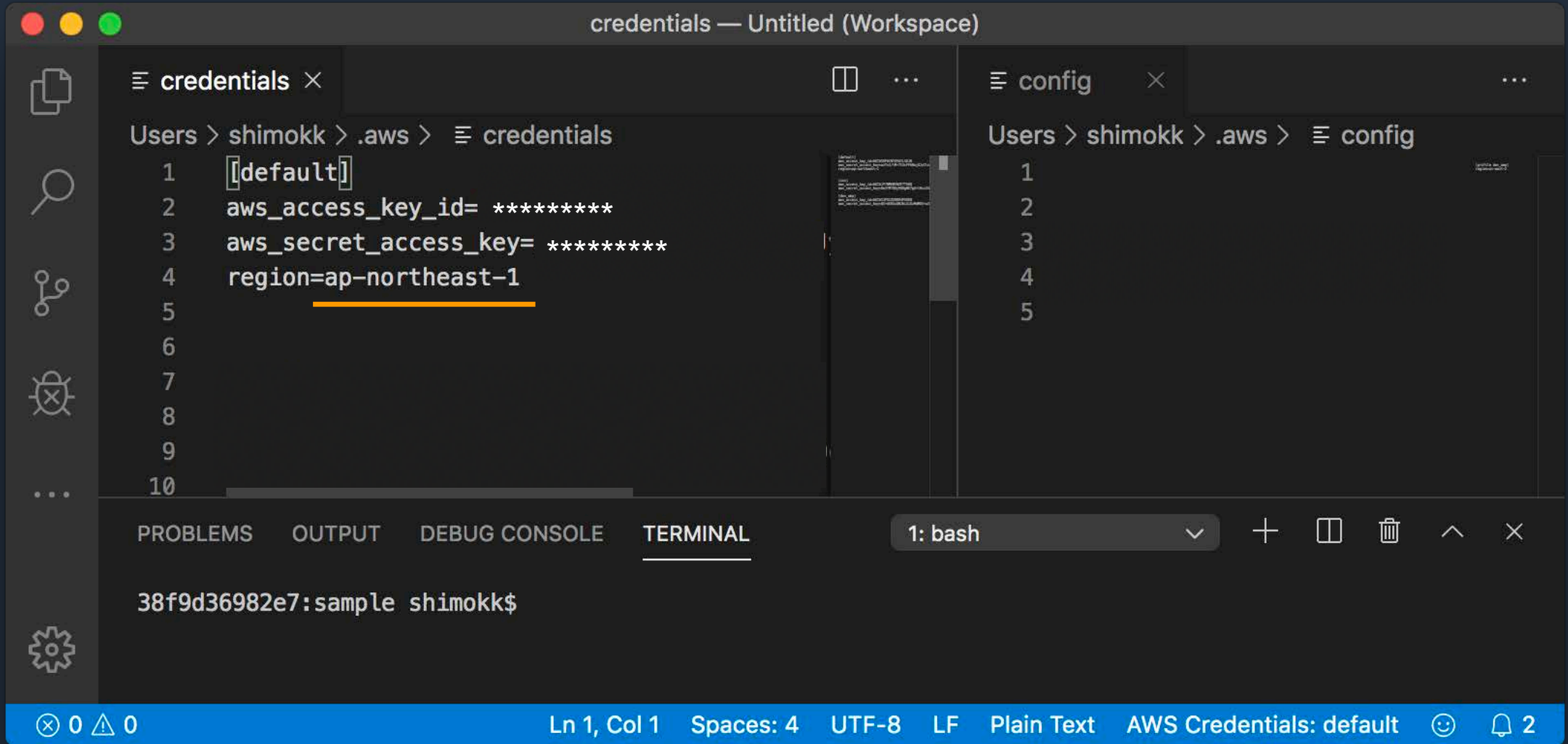
AWS Credentials: default

Credentials の設定

Command+ Shift + P



Credentials の設定



The screenshot shows a code editor window titled "credentials — Untitled (Workspace)". The editor is split into two panes. The left pane shows the file "credentials" with the following content:

```
Users > shimokk > .aws > credentials
1  [[default]]
2  aws_access_key_id= *****
3  aws_secret_access_key= *****
4  region=ap-northeast-1
5
6
7
8
9
10
```

The right pane shows the file "config" with the following content:

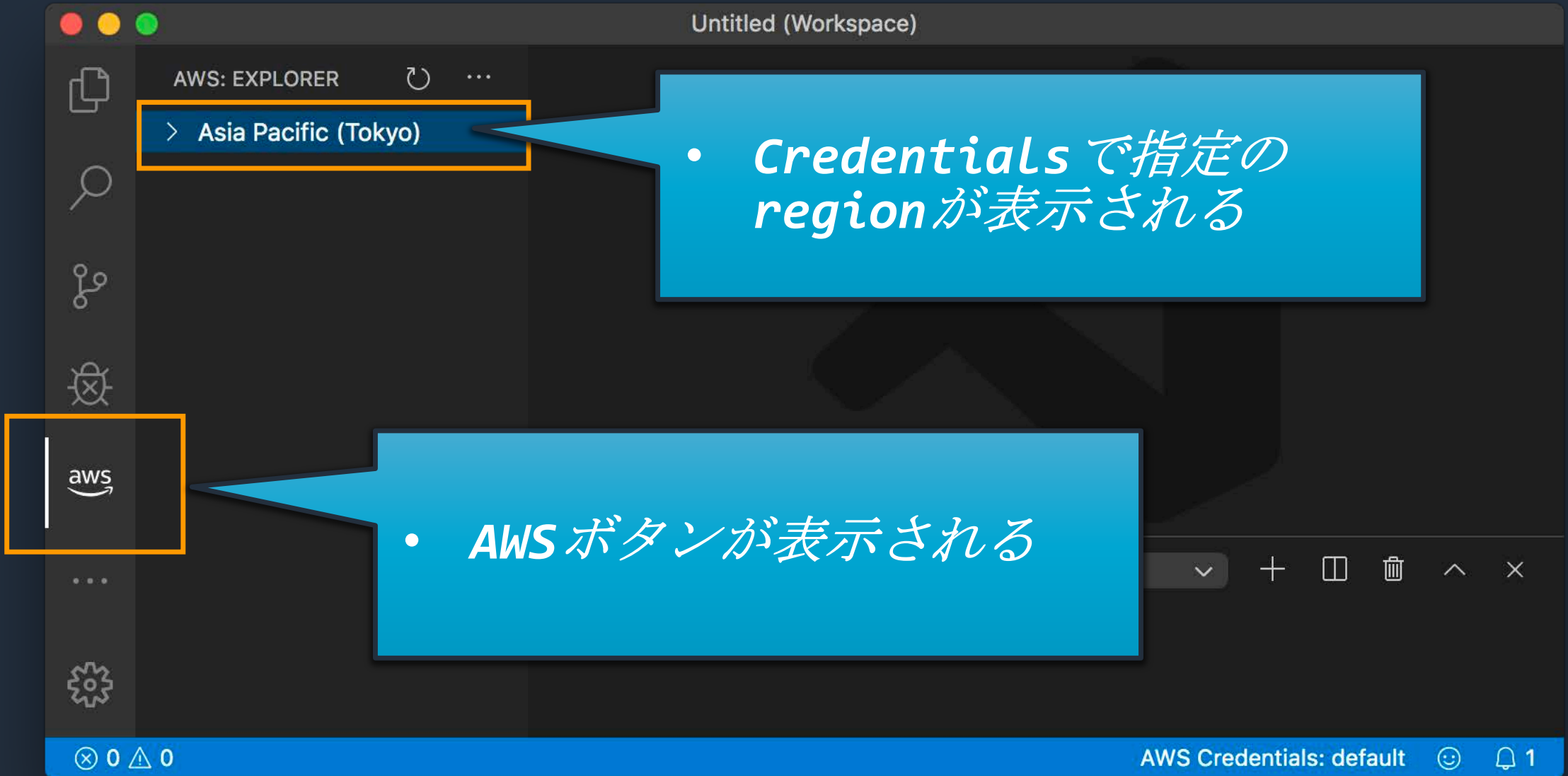
```
Users > shimokk > .aws > config
1
2
3
4
5
```

At the bottom of the editor, there is a terminal window with the following content:

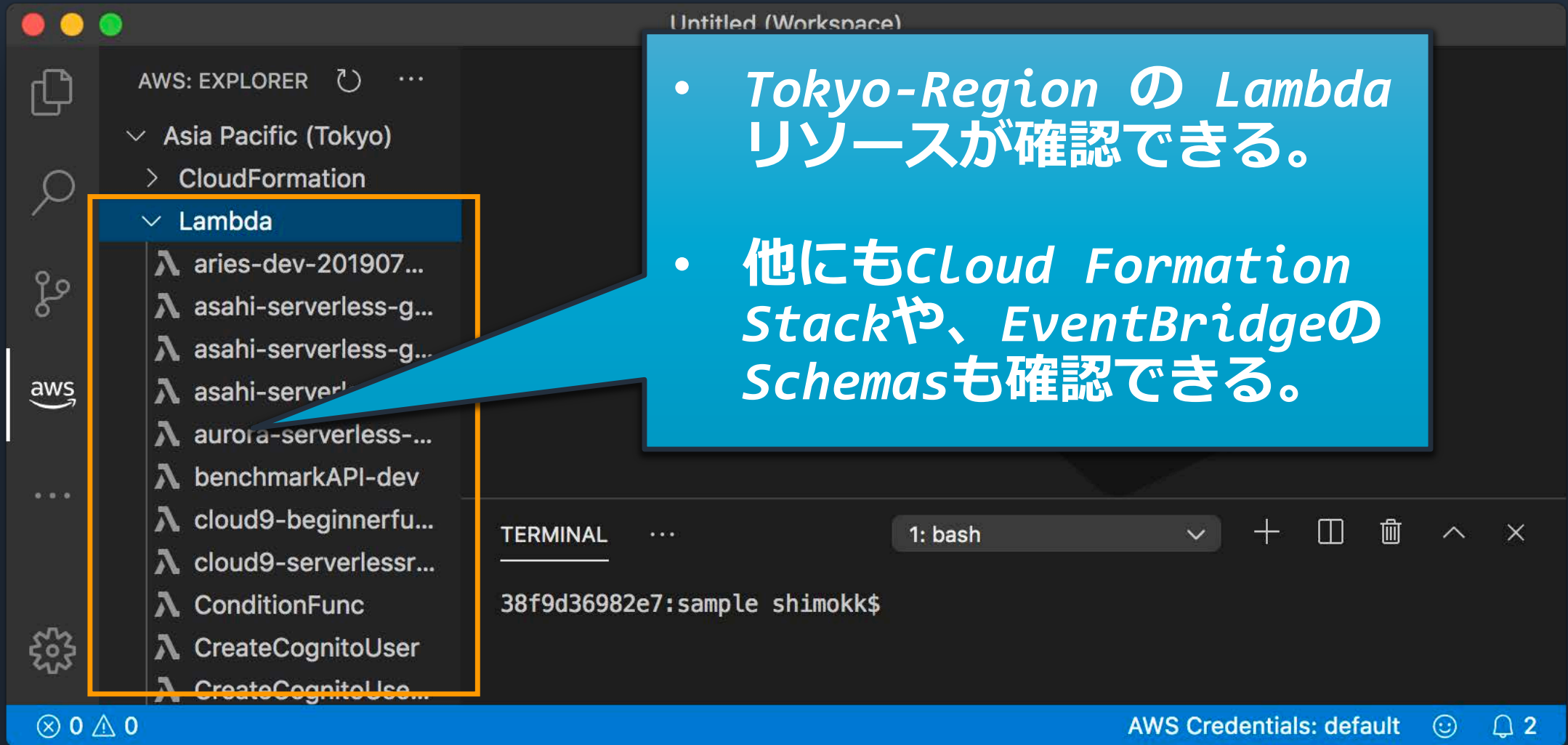
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
38f9d36982e7:sample shimokk$
```

The status bar at the bottom of the IDE shows the following information: 0 errors, 0 warnings, Ln 1, Col 1, Spaces: 4, UTF-8, LF, Plain Text, AWS Credentials: default, and 2 notifications.

AWS Toolkit 対応 IDE/エディタ



アカウント上のリソースを確認



The screenshot shows the AWS Explorer interface with the following structure:

- AWS: EXPLORER
- Asia Pacific (Tokyo)
- CloudFormation
- Lambda (highlighted with an orange box)
- Resources under Lambda:
 - aries-dev-201907...
 - asahi-serverless-g...
 - asahi-serverless-g...
 - asahi-serverl...
 - aurora-serverless-...
 - benchmarkAPI-dev
 - cloud9-beginnerfu...
 - cloud9-serverlessr...
 - ConditionFunc
 - CreateCognitoUser
 - CreateCognitoUse...

A blue callout box contains the following text:

- *Tokyo-Region* の *Lambda* リソースが確認できる。
- 他にも *Cloud Formation Stack* や、 *EventBridge* の *Schemas* も確認できる。

The terminal window at the bottom shows the following command and output:

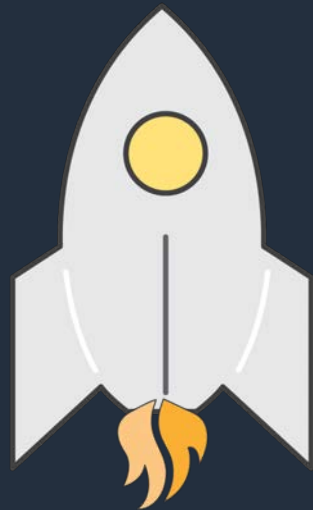
```
1: bash
38f9d36982e7:sample shimokk$
```

The bottom status bar shows "AWS Credentials: default" and a notification icon with the number 2.

A group of people in a meeting looking at a laptop screen. The scene is brightly lit, suggesting a modern office environment. The text is overlaid on a semi-transparent dark blue band across the middle of the image.

Serverlessのデプロイ環境を整える

従来のインフラ構築/管理の課題



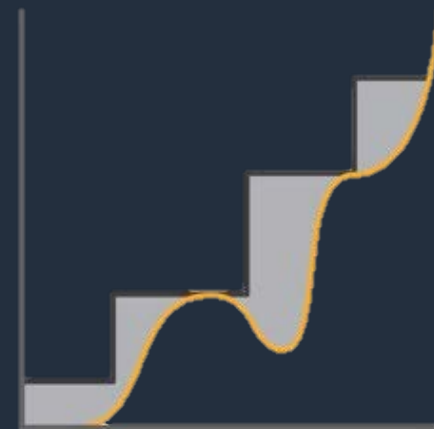
スピード

調達、構築に時間がかかる



安全性

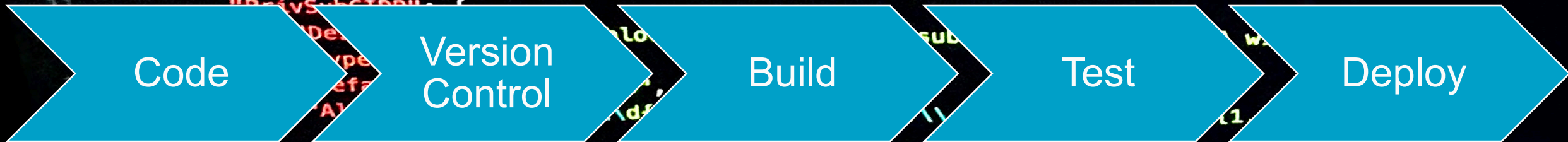
ヒューマンエラーによる構成の不一致



効率性

必要に応じた設定変更ができない

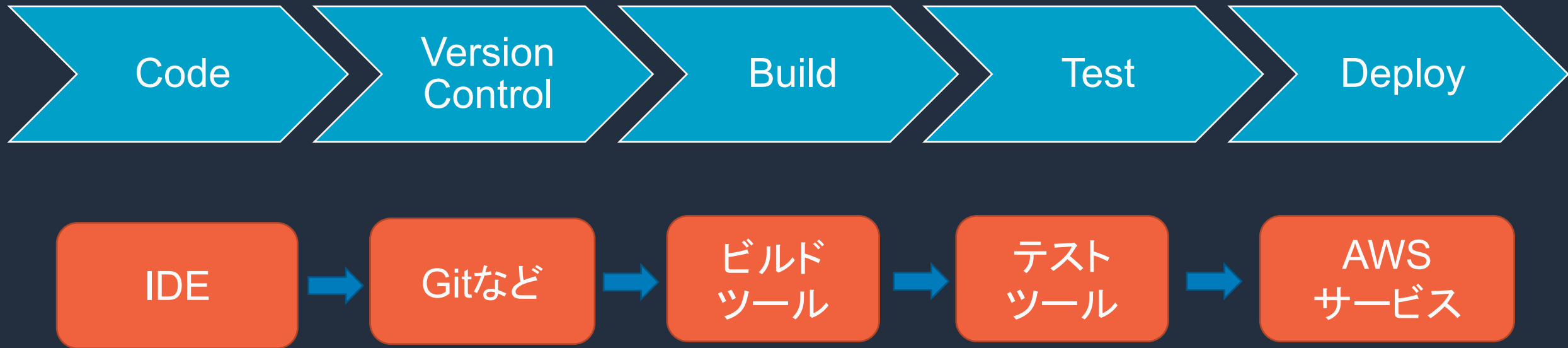
Infrastructure-as-Code



“インフラをコード化、ソフトウェアのように管理”

```
1 {
2   "AWSTemplateFormatVersion": "2010-09-09",
3   "Description": "SAP HANA on AWS",
4   "Parameters": {
5     "VPCID": {
6       "Type": "AWS::EC2::VPC::Id",
7       "Description": "The existing Amazon VPC where you want to deploy SAP HANA.",
8       "Default": "vpc-xxxxxxx",
9       "AllowedPattern": "vpc-[0-9a-z]{8}"
10    },
11    "PrivateSubCIDR": {
12      "Type": "AWS::EC2::Subnet::Id",
13      "Description": "The existing private subnet in your VPC where you want to deploy SAP HANA.",
14      "Default": "subnet-xxxxxxx",
15      "AllowedPattern": "subnet-[0-9a-z]{8}"
16    },
17    "DMZCIDR": {
18      "Description": "CIDR block of the public DMZ subnet where BASTION Host / NAT Gateway exists",
19      "Type": "String",
20      "Default": "10.0.2.0/24",
21      "AllowedPattern": "(\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3}) / (\\d{1,2})"
22    },
23    "RemoteAccessCIDR": {
24      "Description": "CIDR block from where you want to access your RDP instance.",
25      "Type": "String",
26      "Default": "0.0.0.0/0",
27      "AllowedPattern": "(\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3})\\. (\\d{1,3}) / (\\d{1,2})",
28      "ConstraintDescription": "This must be a valid CIDR range in the format x.x.x.x/y."
29    },
30    "PublicSubnet": {
31      "Type": "AWS::EC2::Subnet::Id",
32      "Description": "The existing private subnet in your VPC where you want to deploy SAP HANA.",
33      "Default": "subnet-xxxxxxx",
34      "AllowedPattern": "subnet-[0-9a-z]{8}"
35    }
36  }
37 }
```

Infrastructure-as-Code ワークフロー



“It’s all software”

AWS提供の総合的なツールセット

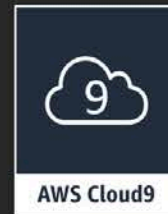
CI/CD Tools



Infrastructure as Code



IDE



Monitoring & Tracing



Web Apps



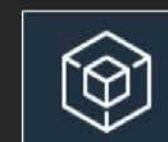
IDE and DevOps Toolkits



CLI and Scripting Tools



Languages



Mobile



SDKs



AWS Serverless Application Model



- AWS Serverless Application Model は、サーバーレス構築用のInfrastructure as Codeフレームワークです。

略して SAM と呼ばれます

AWSのIaCって、AWS CloudFormationがあったよね？

おさらい



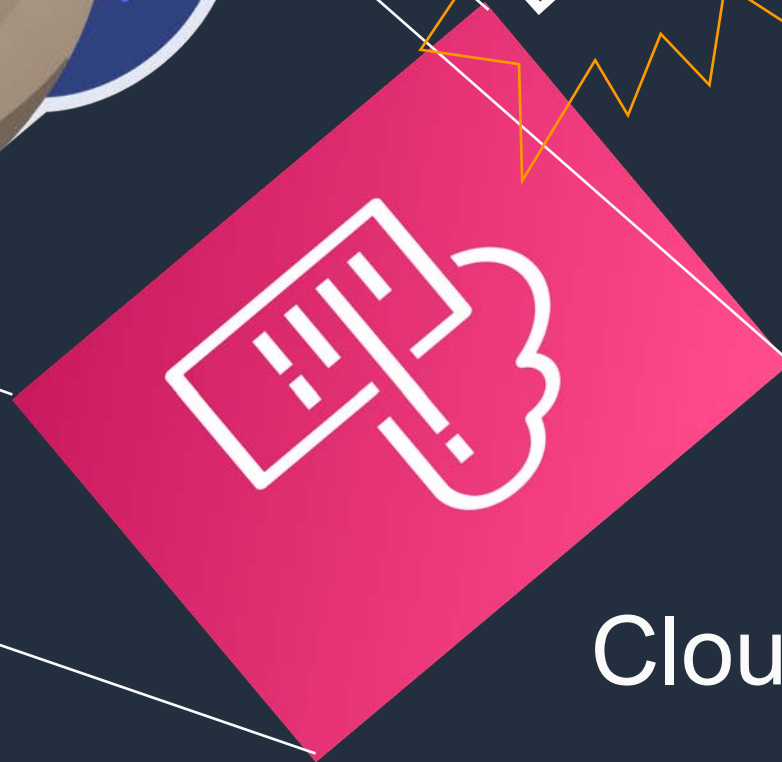
AWS CloudFormation

- AWSの環境構築を、設定ファイル(テンプレート)で自動化するサービス
- システム構成をJSONやYAMLフォーマットのテキストで記述
- ベストプラクティスが盛り込まれた各種サンプルテンプレートを利用可能

AWS SAM



Extended



CloudFormation

SAMで、より簡潔にサーバーレス アプリを定義できる

AWS CloudFormation

```
AWSTemplateFormatVersion: '2010-09-09'  
Transform: 'AWS::Serverless-2016-10-31'  
Description: HelloWorld  
Resources:  
  HelloWorld  
    Type: 'AWS::Serverless::Function'  
    Properties:
```

```
      Handler: index.handler  
      Runtime: python3.8  
      CodeUri: src/handlers/func1  
      Description: helloworld  
      MemorySize: 128  
      Timeout: 3  
      Events:
```

```
        GetResource:  
          Type: Api  
          Properties:  
            Path: /hello  
            Method: get
```



```
AWSTemplateFormatVersion: "2010-09-09"  
Description: HelloWorld  
Resources:  
  HelloWorldRole:  
    Type: "AWS::IAM::Role"  
    Properties:  
      ManagedPolicyArns:  
        - "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"  
      AssumeRolePolicyDocument:  
        Version: "2012-10-17"  
        Statement:  
          - Action:  
              - "sts:AssumeRole"  
            Effect: Allow  
            Principal:  
              Services:  
                - "lambda.amazonaws.com"  
  HelloWorld:  
    Type: "AWS::Lambda::Function"  
    Properties:  
      Code:  
        S3Bucket: xxx-bucket  
        S3Key: xxx.zip  
      Description: HelloWorld  
      Tags:  
        - Value: SAM  
          Key: "lambda:createdBy"  
      MemorySize: 128  
      Handler: "index.handler"  
      Role:  
        "Fn::GetAtt":  
          - HelloWorldRole  
      Arn:  
        Fn::Join:  
          - Arn  
          - "lambda:  
            Runtime: "nodejs10.x"  
  ServerlessRestApiProdStages:  
    Type: "AWS::ApiGateway::Stage"  
    Properties:  
      DeploymentId:  
        Ref: ServerlessRestApiDeploymentcd3ad6578f  
      RestApiId:  
        Ref: ServerlessRestApi  
      StageName: Prod  
  ServerlessRestApiDeploymentcd3ad6578f:  
    Type: "AWS::ApiGateway::Deployment"  
    Properties:  
      RestApiId:  
        Ref: ServerlessRestApi  
      Description: "RestApi deployment id: cd3ad6578f645e5e5c5d2c73458893b594684"  
    StageName: Stage  
  HelloWorldGetResourcePermissionProd:  
    Type: "AWS::Lambda::Permission"  
    Properties:  
      Action: "lambda:invokeFunction"  
      Principal: "apigateway.amazonaws.com"  
      FunctionName:  
        Ref: HelloWorld  
      SourceArn:  
        "Fn::Sub":  
          - "arn:aws:execute-api:${AWS:Region}:${AWS:AccountId}:${_ApiId_}/${_Stage_}/GET/hello"  
          - "_Stage_" : Prod  
          - "_ApiId_" :  
              Ref: ServerlessRestApi  
  HelloWorldGetResourcePermissionTest:  
    Type: "AWS::Lambda::Permission"  
    Properties:  
      Action: "lambda:invokeFunction"  
      Principal: "apigateway.amazonaws.com"  
      FunctionName:  
        Ref: HelloWorld  
      SourceArn:  
        "Fn::Sub":  
          - "arn:aws:execute-api:${AWS:Region}:${AWS:AccountId}:${_ApiId_}/${_Stage_}/GET/hello"  
          - "_Stage_" : "a"  
          - "_ApiId_" :  
              Ref: ServerlessRestApi  
  ServerlessRestApi:  
    Type: "AWS::ApiGateway::RestApi"  
    Properties:  
      Body:  
        info:  
          version: "1.0"  
          title:  
            Ref: "AWS::StackName"  
          paths:  
            "/hello":  
              get:  
                - "x-amazon-apigateway-integration":  
                    httpMethod: POST  
                    type: aws_proxy  
                    uri:  
                      "Fn::Sub": "arn:aws:apigateway:${AWS:Region}:lambda:path/2015-03-31/functions/${HelloWorldArn}/invocations"  
                    responses:  
                      swagge: "2.0"
```


SAMで、より簡潔にサーバーレス アプリを定義できる

```
AWSTemplateFormatVersion: '2010-09-09'  
Transform: 'AWS::Serverless-2016-10-31'  
Description: HelloWorld  
Resources:
```

```
  HelloWorld
```

```
    Type: 'AWS::Serverless::Function'
```

```
    Properties:
```

```
      Handler: index.handler  
      Runtime: python3.8  
      CodeUri: src/handlers/func1  
      Description: HelloWorld  
      MemorySize: 128  
      Timeout: 3
```

```
    Events:
```

```
      GetResource  
        Type: Api  
        Properties:  
          Path: /hello  
          Method: get
```



CloudFormation Stack

API Gateway



Permissions

GET /hello

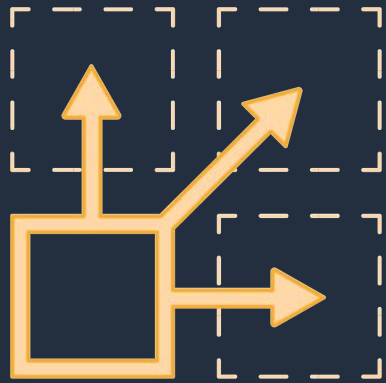


AWS Lambda



Role

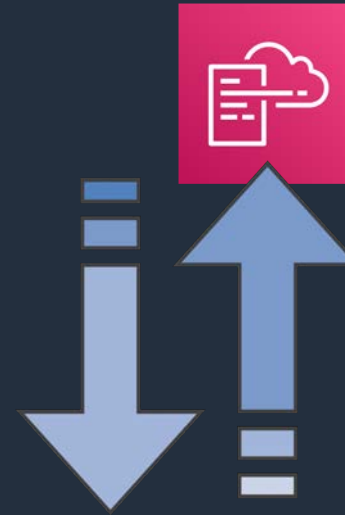
AWS SAM



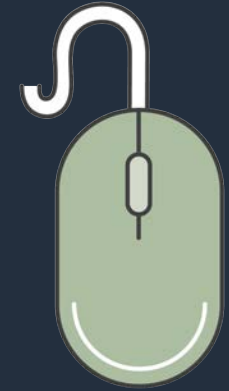
リソース参照による
効率良い記述



ローカルでのテスト
およびデバッグ



CloudFormation
文法での構築も可能



組み込みの
ベストプラクティス
での簡単な構築

[AWS Black Belt Online Seminar] AWS Serverless Application Model 資料及び QA 公開

by AWS Japan Staff | on 23 AUG 2019 | in AWS Lambda, AWS Serverless Application Model, Webinars | [Permalink](#) | [Share](#)

先日 (2019/08/14) 開催しました AWS Black Belt Online Seminar 「AWS Serverless Application Model」 の資料を公開しました。当日、参加者の皆様から頂いた QA の一部についても共有しております。



【AWS Black Belt Online Seminar】 AWS Serverless Application Model (AWS SAM)

サービスカットシリーズ

Solutions Architect
今村 優太
2019/8/14

AWS 公式 Webinar
<https://amzn.to/JPWebinar>

過去資料
<https://amzn.to/JPArchive>



SAMによるServerless構築の流れ

SAM
template



ロード

SAM



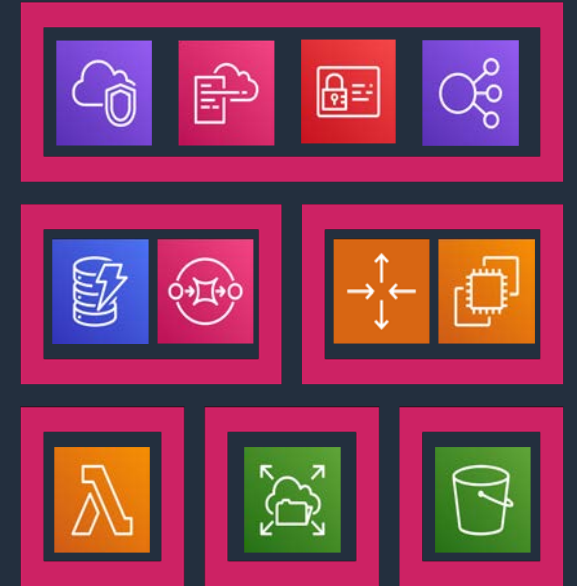
トランスパイル

CloudFormation



作成/変更/削除

CloudFormation
Stack



リソースの定義
パラメータの定義

スタックの作成/変更/削除
エラー検知とロールバック

AWSリソース

SAMによるServerless構築の流れ

SAM
template



ロード

SAM



トランスパイル

CloudFormation



作成/変更/削除

CloudFormation
Stack



リソースの定義
パラメータの定義

SAMを効率よく操作したい！

AWSリソース

SAMを使いやすくしよう！



IDE + AWS Toolkit



SAM CLI

SAMを使いやすくしよう！



IDE + AWS Toolkit



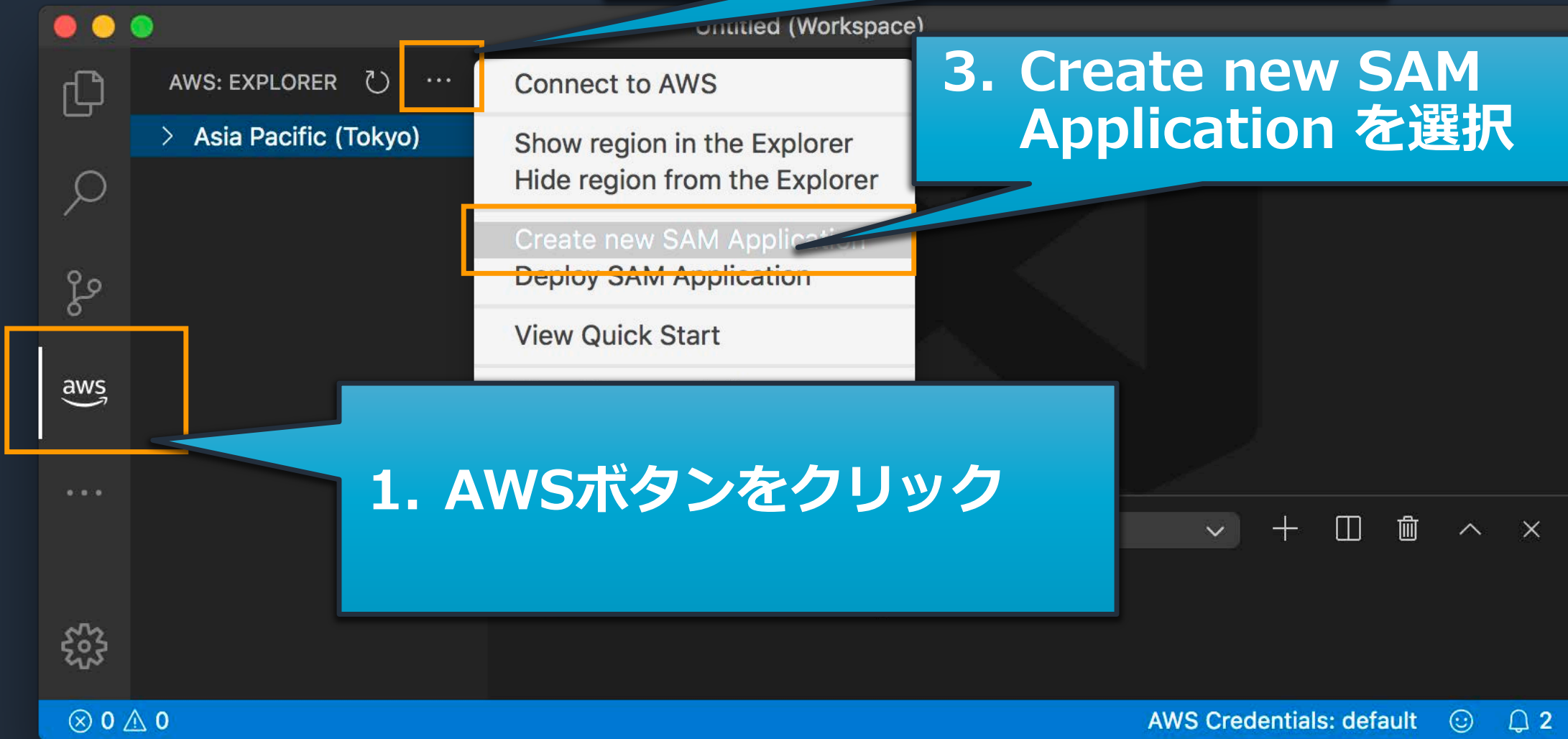
SAM CLI

IDE + AWS Toolkit

2. [...]をクリック

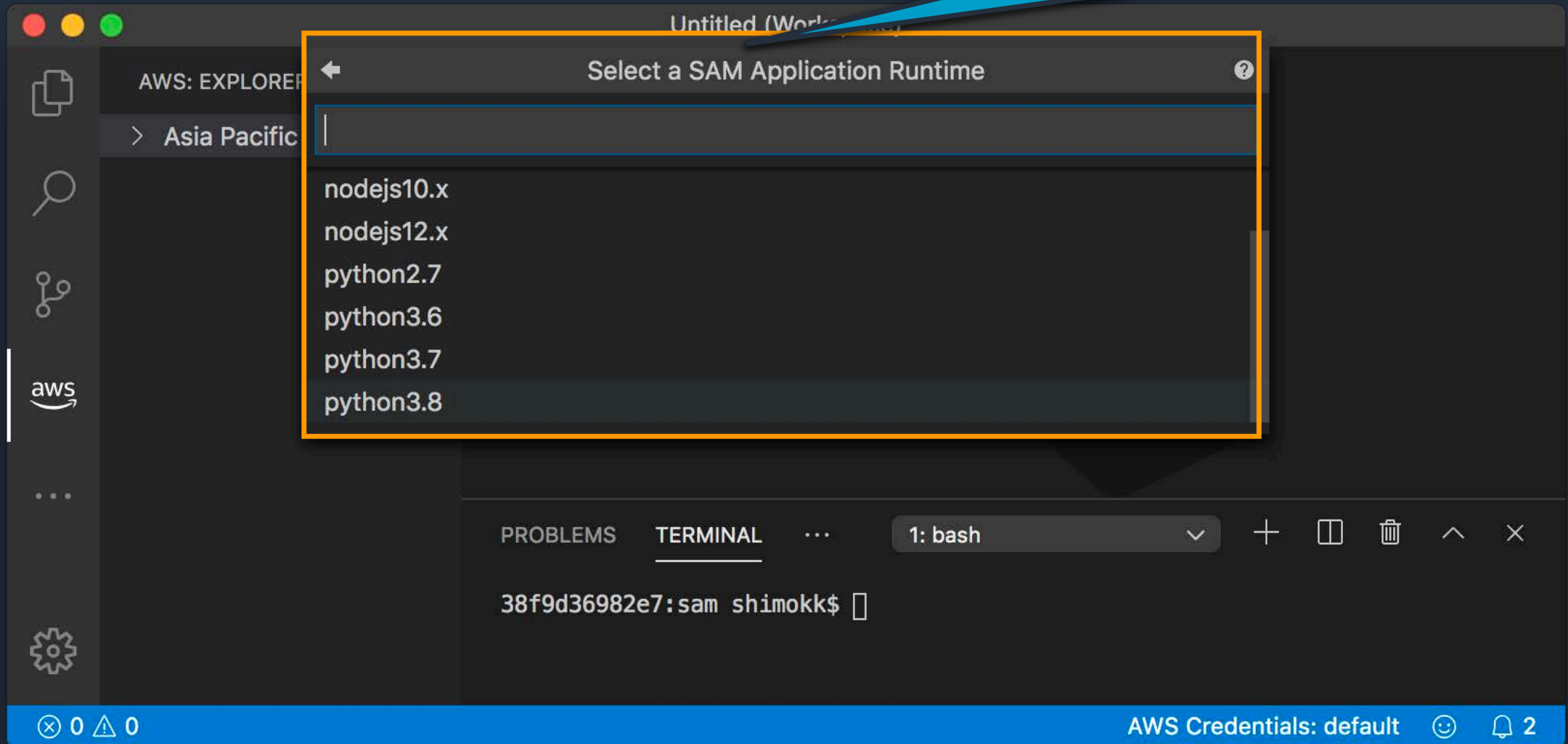
3. Create new SAM Application を選択

1. AWSボタンをクリック



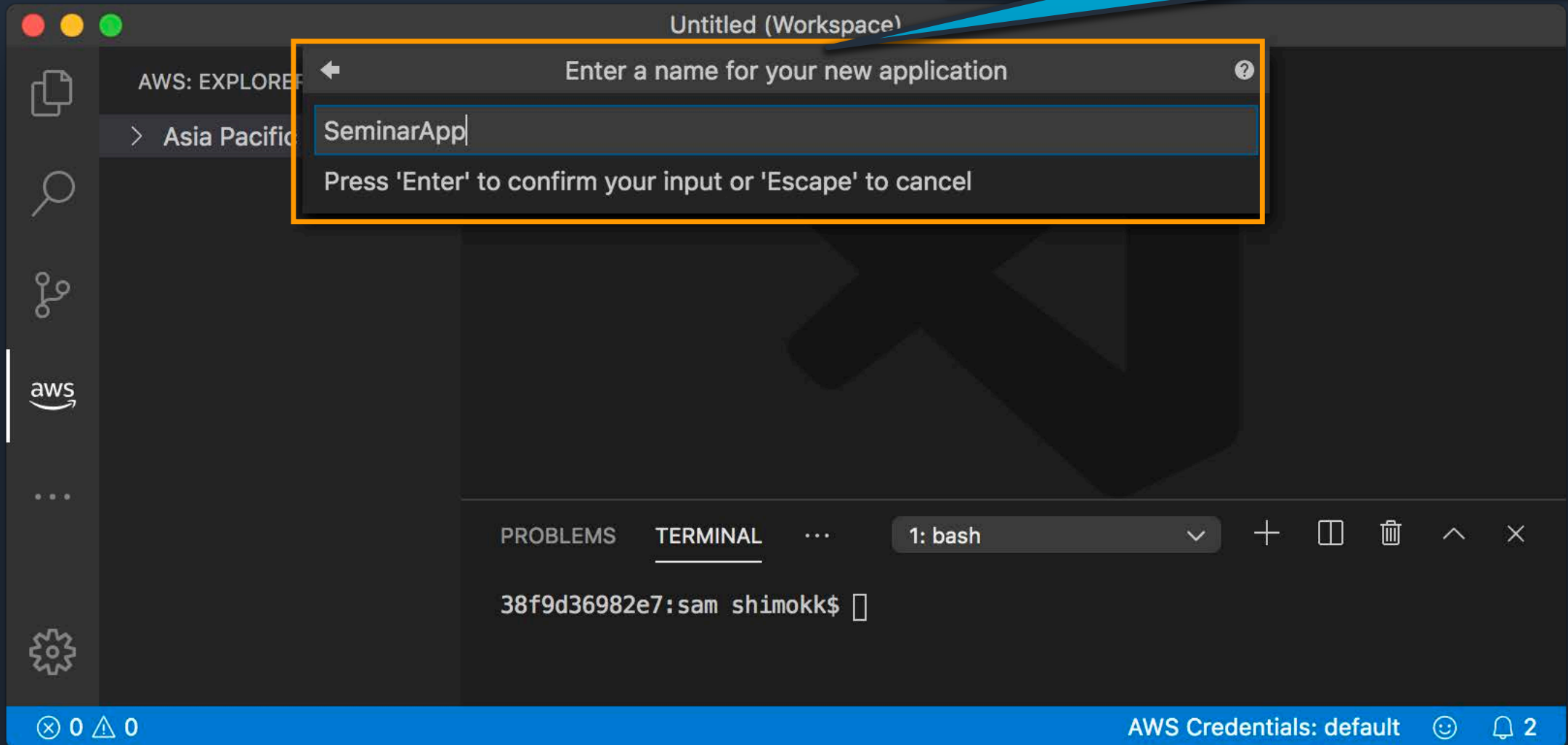
IDE + AWS Toolkit

runtimeを選択



IDE + AWS Toolkit

Applicationの名前を入力



IDE + AWS Toolkit

Template.yamlが自動生成

The screenshot shows the AWS Explorer IDE interface. On the left, the 'AWS: EXPLORER' sidebar is open to the 'Asia Pacific (Tokyo)' region. The main editor area displays a file named 'template.yaml' with the following content:

```
! template.yaml
sam > SeminarApp > ! template.yaml
1  AWSTemplateFormatVersion: '2010-09-09'
2  Transform: AWS::Serverless-2016-10-31
3  Description: >
4    SeminarApp
5
6    Sample SAM Template for SeminarApp
7
8  # More info about Globals: https://github.com/awslabs/sam/blob/main/docs/en/globals.md
9  Globals:
10 Function:
```

Below the editor, a terminal window shows the command execution:

```
38f9d36982e7:sam shimokk$ ls SeminarApp/
README.md      hello_world    tests
events         template.yaml
38f9d36982e7:sam shimokk$
```

At the bottom of the IDE, the status bar indicates 'Ln 1, Col 1 Spaces: 2 UTF-8 LF YAML AWS Credentials: default'.

Application雛形が自動生成

自動生成された SAM の構造

SeminarApp

- ├─ README.md
- ├─ events
 - └─ event.json
- ├─ hello_world
 - ├─ app.py
 - └─ requirements.txt
- ├─ template.yaml
- └─ tests
 - └─ unit
 - └─ test_handler.py

```
import json

def lambda_handler(event, context):

    return {
        "statusCode": 200,
        "body": json.dumps({
            "message": "hello world",
        }),
    }
```

シンプルなLambda関数を生成

自動生成された SAM の構造

SeminarApp

- ├─ README.md
- ├─ events
 - └─ event.json
- ├─ hello_world
 - ├─ app.py
 - └─ requirements.txt
- ├─ template.yaml
- └─ tests
 - └─ unit
 - └─ test_handler.py

```
import json
import pytest
from hello_world import app

def test_lambda_handler(apigw_event, mocker):

    ret = app.lambda_handler(apigw_event, "")
    data = json.loads(ret["body"])

    assert ret["statusCode"] == 200
    assert "message" in ret["body"]
    assert data["message"] == "hello world"
```

シンプルな assertsを生成

自動生成された SAM の構造

SeminarApp

- └─ README.md
- └─ events
 - └─ event.json
- └─ hello_world
 - └─ app.py
 - └─ requirements.txt
- └─ template.yaml
- └─ tests
 - └─ unit
 - └─ test_handler.py

```
{  
  "body": "{¥"message¥": ¥"hello world¥"}",  
  "resource": "/{proxy+}",  
  "path": "/path/to/resource",  
  "httpMethod": "POST",  
  "isBase64Encoded": false,  
  "queryStringParameters": {  
    "foo": "bar"  
  },  
  "pathParameters": {  
    "proxy": "/path/to/resource"  
  },  
  "stageVariables": {  
    "baz": "qux"  
  },  
  "headers": {}  
}
```

API GatewayからのEventを生成

自動生成された SAM の構造

SeminarApp

```
├── README.md
├── events
│   └── event.json
├── hello_world
│   ├── app.py
│   └── requirements.txt
├── template.yaml
└── tests
    └── unit
        └── te
```

***SAM Template*を生成**

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  SeminarApp

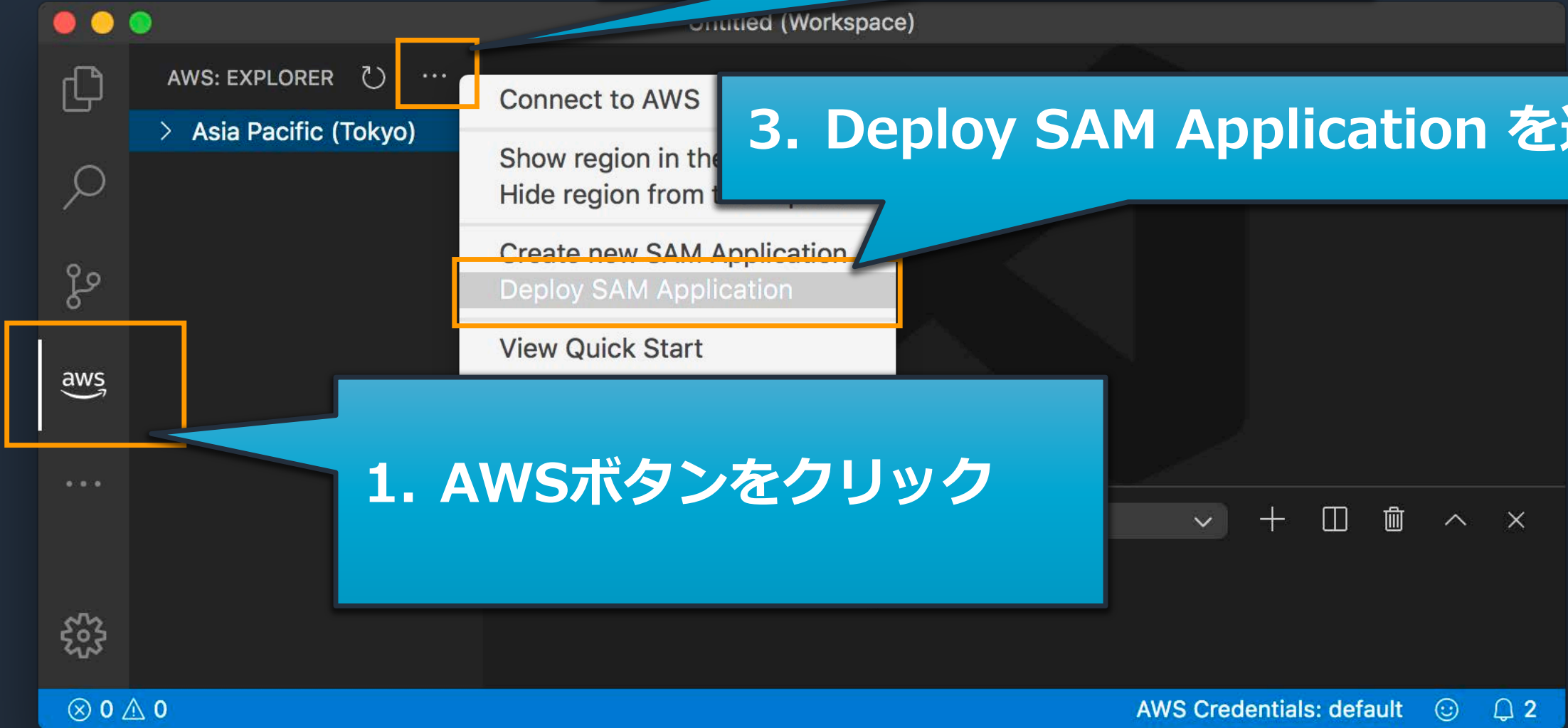
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.8
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
            Method: get
```

SAM の GUI Deploy

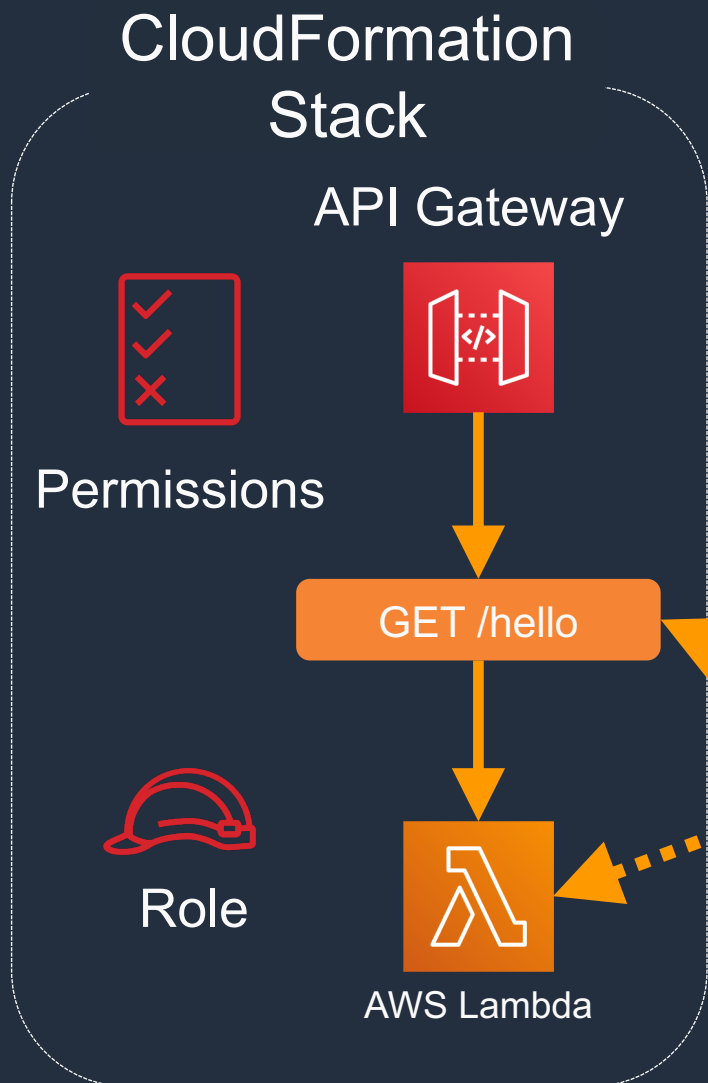
2. [...]をクリック

3. Deploy SAM Application を選択

1. AWSボタンをクリック



Deployされたアーキテクチャとの対比



```
AWSTemplateFormatVersion: '2010-09-09'  
Transform: AWS::Serverless-2016-10-31
```

```
Description:  
Seminar
```

アーキテクチャとの対応

```
Resources:
```

```
HelloWorldFunction:
```

```
Type: AWS::Serverless::Function
```

```
Properties:
```

```
CodeUri: hello_world/
```

```
Handler: app.lambda_handler
```

```
Runtime: python3.8
```

```
Events:
```

```
HelloWorld:
```

```
Type: Api
```

```
Properties:
```

```
Path: /hello
```

```
Method: get
```

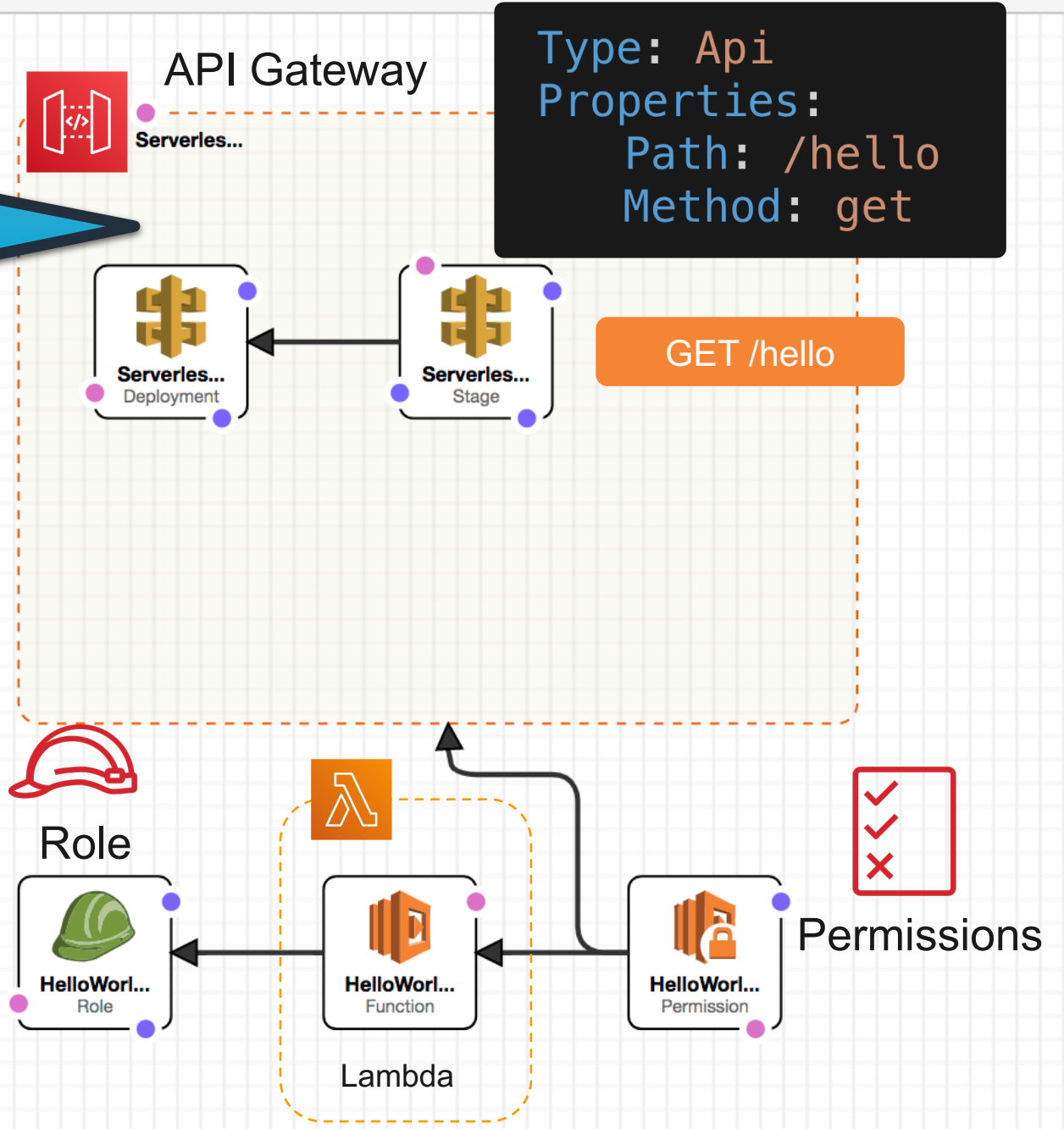
Resource types

- ACMPCA
- ApiGatewayV2
- AppMesh
- AppStream
- AppSync
- ApplicationAutoScaling
- Athena
- AutoScaling

DeployしたStackは
Management Console上
のDesignerで確認できる。

Type: `AWS::Serverless::Function`
 Properties:
 CodeUri: `hello_world/`
 Handler: `app.lambda_handler`
 Runtime: `python3.8`

File: 'template1'



Navigation and zoom controls including a center icon, a plus sign, a vertical slider, and a minus sign.

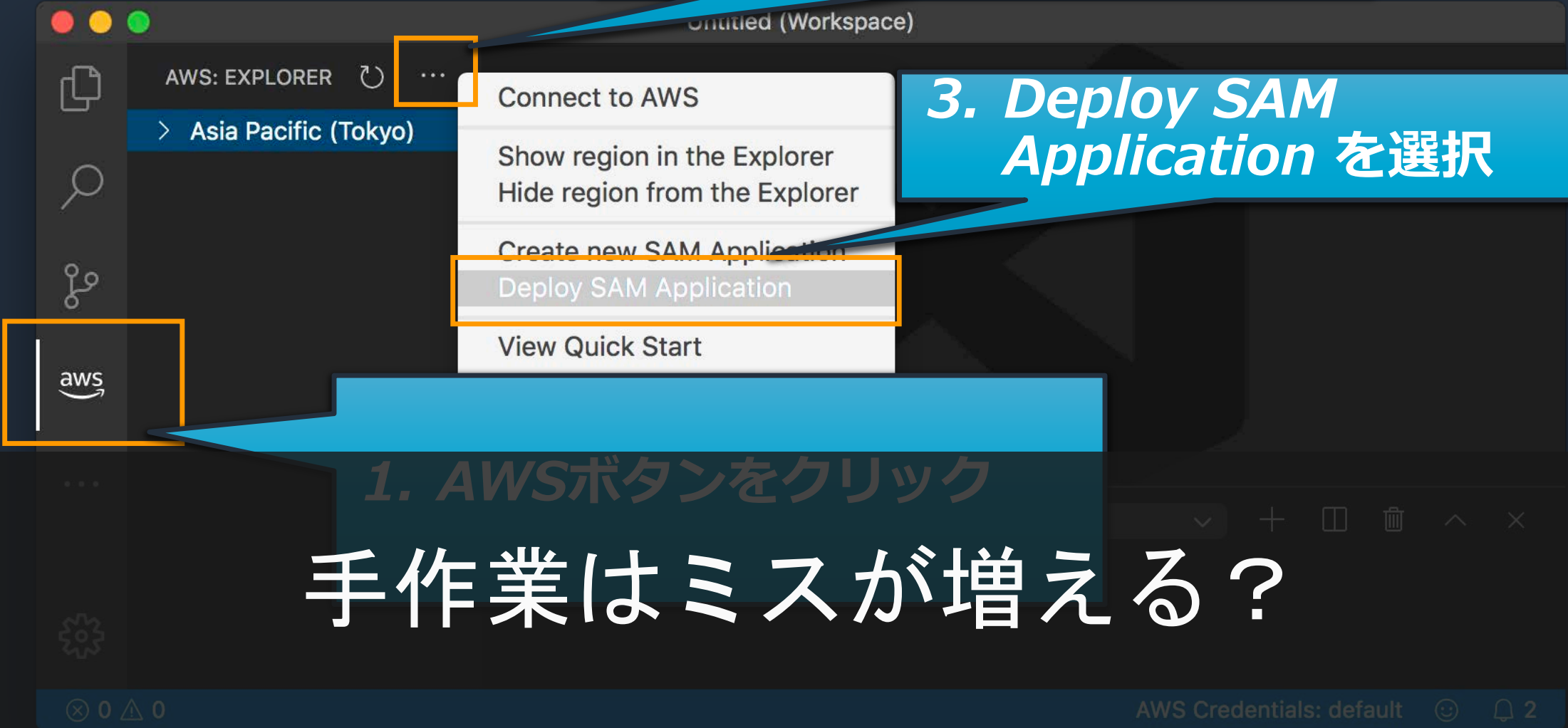
SAM の GUI Deploy

2. [...]をクリック

3. *Deploy SAM Application* を選択

1. AWSボタンをクリック

手作業はミスが増える？



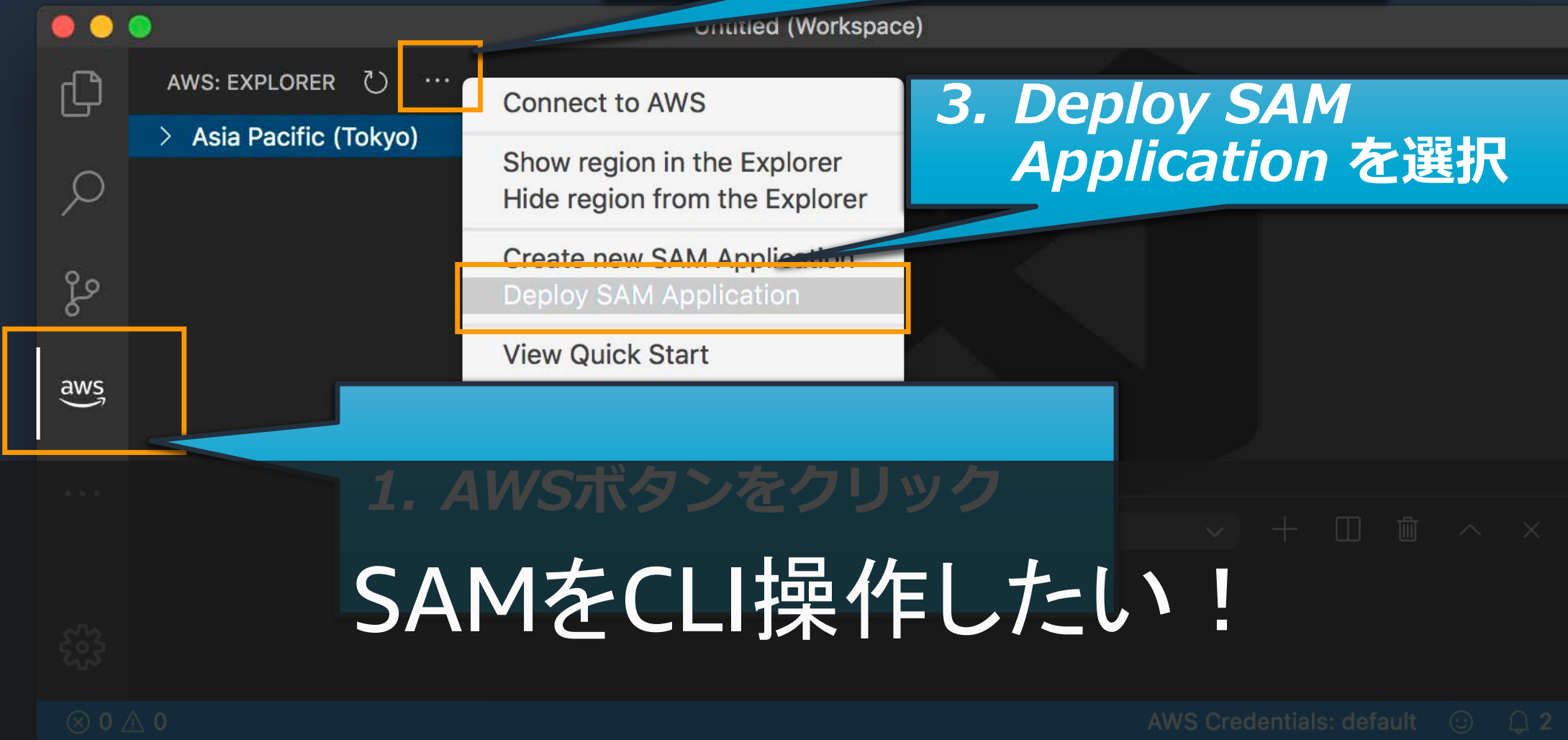
SAM の GUI Deploy

2. [...]をクリック

3. *Deploy SAM Application* を選択

1. AWSボタンをクリック

SAMをCLI操作したい！



SAMを使いやすくしよう！



IDE + AWS Toolkit



SAM CLI

AWS SAM CLI

1) アプリケーションのひな形を生成

```
$ sam init --runtime python3.8
```

2) ローカル環境でビルド

```
$ cd sam-app  
$ sam build
```

3) ビルド後にAWS へデプロイ

```
$ sam deploy --guided
```

※sam deploy の際に必要な S3 バケットは自動生成



SAM CLI

AWS SAM CLI

1) アプリケーションの雛形を生成

```
$ sam init --runtime python3.8
```

2) ローカル環境でビルド

```
$ cd sam-app  
$ sam build
```

- *sam init* コマンドでプロジェクトの雛形を作成
- *--runtime* オプションで言語を指定

3) ビルド後にAWS へデプロイ

```
$ sam deploy --guided
```



SAM CLI

※sam deploy の際に必要な S3 バケットは自動生成

AWS SAM CLI

1) アプリケーションのひな形を生成

```
$ sam init --runtime python3.8
```

2) ローカル環境でビルド

```
$ cd sam-app  
$ sam build
```

3) ビルド後にAWSにデプロイ

```
$ sam deploy
```

- ***--use-container*** オプションで ***Docker Container*** 内でビルドすることも可能



※sam deploy の際に必要な S3 バケットは自動生成

SAM CLI

AWS SAM CLI

1) アプリケーションのひな形を生成

```
$ sam init --runtime python3.8
```

2) ローカル環境でビルド

```
$ cd sam-app  
$ sam build
```

3) ビルド後にAWS へデプロイ

```
$ sam deploy --guided
```



※sam deploy の際に必要な S3 バケットは自動生

- **--guided** オプションで *change set* を確認後にデプロイ可能

--guided オプション

```
sam-deploy-test ) sam deploy --guided
```

```
Configuring SAM deploy
```

```
=====  
Looking for samconfig.toml : Not found
```

```
Setting default arguments for 'sam deploy'
```

```
-----  
Stack Name [sam-app]: sam-deploy-test
```

```
AWS Region [us-east-1]: ap-northeast-1
```

```
#Shows you resources changes to be deployed and require a 'Y' to initiate deploy
```

```
Confirm changes before deploy [y/N]: y
```

```
#SAM needs permission to be able to create roles to connect to the resources in your template
```

```
Allow SAM CLI IAM role creation [Y/n]: y
```

```
Save arguments to samconfig.toml [Y/n]: y
```

```
Looking for resources needed for deployment: Not found.
```

```
Creating the required resources...
```

```
Successfully created!
```

```
Managed S3 bucket: aws-sam-cli-managed-default-samclisourcebucket-sykucxjohevd
```

```
A different default S3 bucket can be set in samconfig.toml
```

デプロイ先のリージョンや、AWS *CloudFormation* のスタック名を *CLI* のプロンプトから指定可能

--guided オプション

```
sam-deploy-test ) sam deploy --guided
```

```
Configuring SAM deploy
```

```
=====  
Looking for samconfig.toml : Not found
```

```
Setting default arguments for 'sam
```

```
=====  
Stack Name [sam-app]: sam-deploy-t
```

```
AWS Region [us-east-1]: ap-northea
```

```
#Shows you resources changes to be
```

```
Confirm changes before deploy [y/N]
```

```
#SAM needs permission to be able t
```

```
Allow SAM CLI IAM role creation [Y/n]: y
```

```
Save arguments to samconfig.toml [Y/n]: y
```

```
Looking for resources needed for deployme not found.
```

```
Creating the required resources...
```

```
Successfully created!
```

自動生成された、*package* 格納用の *S3 Bucket* 情報

template

```
Managed S3 bucket: aws-sam-cli-managed-default-samclisourcebucket-sykucxjohevd  
A different default S3 bucket can be set in samconfig.toml
```

--guided オプション

| Operation | LogicalResourceId | ResourceType |
|-----------|--|-----------------------------|
| + Add | ServerlessRestApiDeployment8cf30ed3cd | AWS::ApiGateway::Deployment |
| * Modify | HelloWorldFunctionHelloWorldPermissionProd | AWS::Lambda::Permission |
| * Modify | ServerlessRestApiProdStage | AWS::ApiGateway::Stage |
| * Modify | ServerlessRestApi | AWS::ApiGateway::RestApi |
| - Delete | ServerlessRestApiDeployment47fc2d5f9d | AWS::ApiGateway::Deployment |

前回デプロイからの変更差分を表示

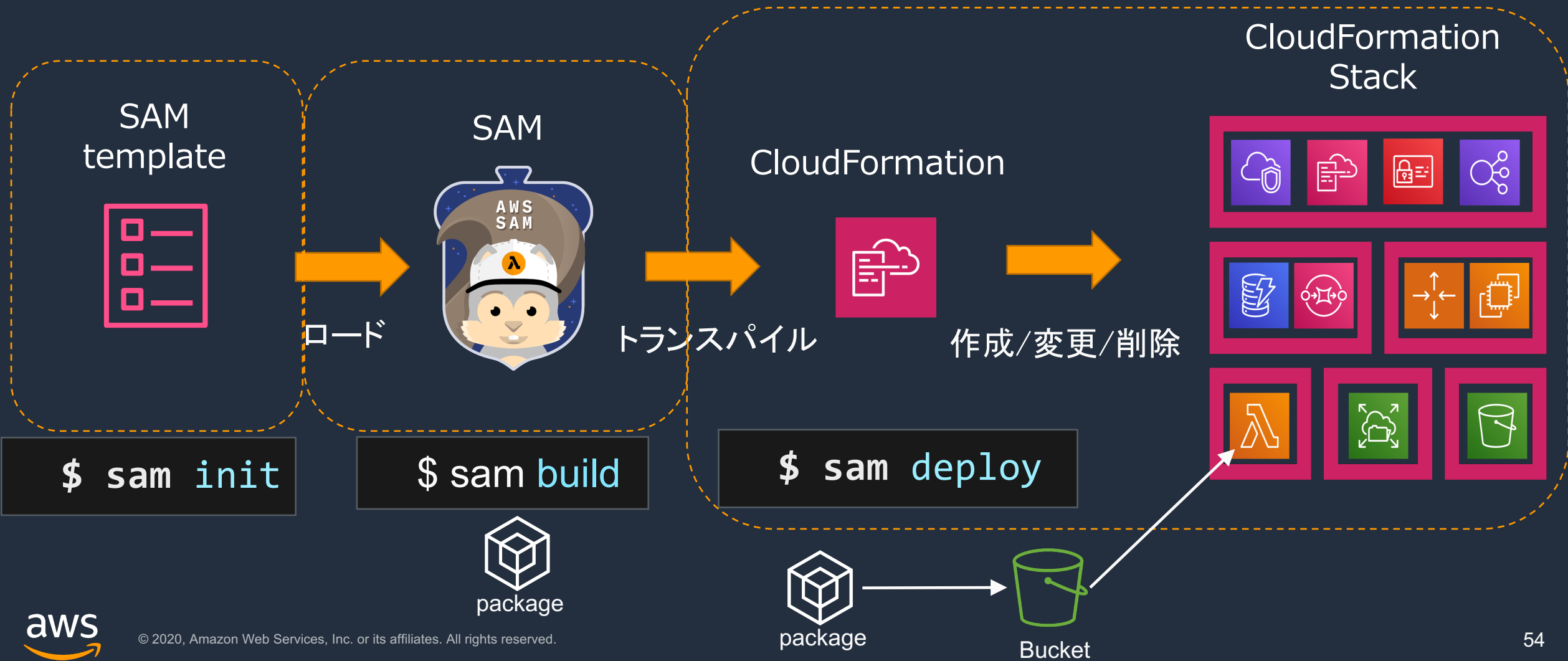
→ この段階でデプロイをキャンセルできる

--guided オプション

CloudFormation 実行後のイベント発生履歴

| ResourceStatus | ResourceType | LogicalResourceId | ResourceStatusReason |
|-------------------------------------|-----------------------------|--|--|
| UPDATE_IN_PROGRESS | AWS::ApiGateway::RestApi | ServerlessRestApi | - |
| UPDATE_COMPLETE | AWS::ApiGateway::RestApi | ServerlessRestApi | - |
| CREATE_IN_PROGRESS | AWS::ApiGateway::Deployment | ServerlessRestApiDeployment8cf30ed3cd | - |
| CREATE_IN_PROGRESS | AWS::ApiGateway::Deployment | ServerlessRestApiDeployment8cf30ed3cd | Resource creation Initiated |
| CREATE_COMPLETE | AWS::ApiGateway::Deployment | ServerlessRestApiDeployment8cf30ed3cd | - |
| UPDATE_IN_PROGRESS | AWS::Lambda::Permission | HelloWorldFunctionHelloWorldPermissionProd | Requested update requires the creation of a new physical resource; hence creating one. |
| UPDATE_IN_PROGRESS | AWS::ApiGateway::Stage | ServerlessRestApiProdStage | - |
| UPDATE_COMPLETE | AWS::ApiGateway::Stage | ServerlessRestApiProdStage | - |
| UPDATE_IN_PROGRESS | AWS::Lambda::Permission | HelloWorldFunctionHelloWorldPermissionProd | Resource creation Initiated |
| UPDATE_COMPLETE | AWS::Lambda::Permission | HelloWorldFunctionHelloWorldPermissionProd | - |
| UPDATE_COMPLETE_CLEANUP_IN_PROGRESS | AWS::CloudFormation::Stack | sam-deploy-test | - |

SAMによるServerless構築の流れ



A group of people in a meeting looking at a laptop screen. The scene is brightly lit, suggesting a modern office environment. The text is overlaid on a semi-transparent dark blue band across the middle of the image.

ServerlessのCI/CDパイプライン を整える

AWS提供の総合的なツールセット

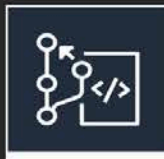
CI/CD Tools



AWS CodeStar



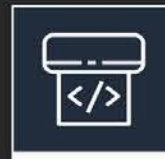
AWS CodeBuild



AWS CodeCommit



AWS CodeDeploy



AWS CodePipeline

Infrastructure as Code



AWS CloudFormation



AWS Cloud Dev. Kit (CDK)

IDE



AWS Cloud9

Monitoring & Tracing



AWS X-Ray



Amazon CloudWatch

Web Apps



AWS Elastic Beanstalk

IDE and DevOps Toolkits



Visual Studio Code



IntelliJ



PyCharm



Visual Studio



Eclipse



VSTS

CLI and Scripting Tools

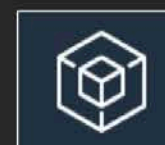


AWS CLI



Tools for PowerShell

Languages



Amazon Corretto

Mobile



AWS Amplify

SDKs



JavaScript



Python



PHP



.NET



Ruby



Java



Go

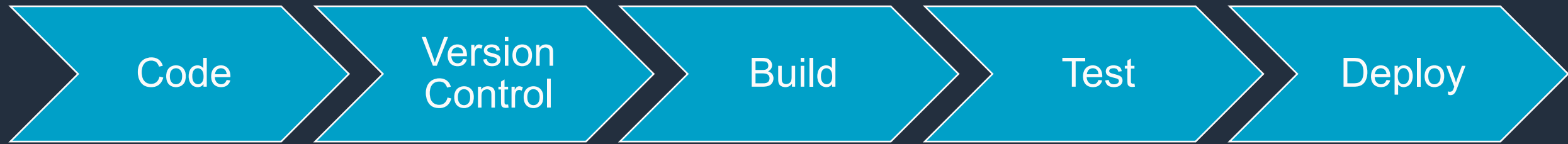


Node.js

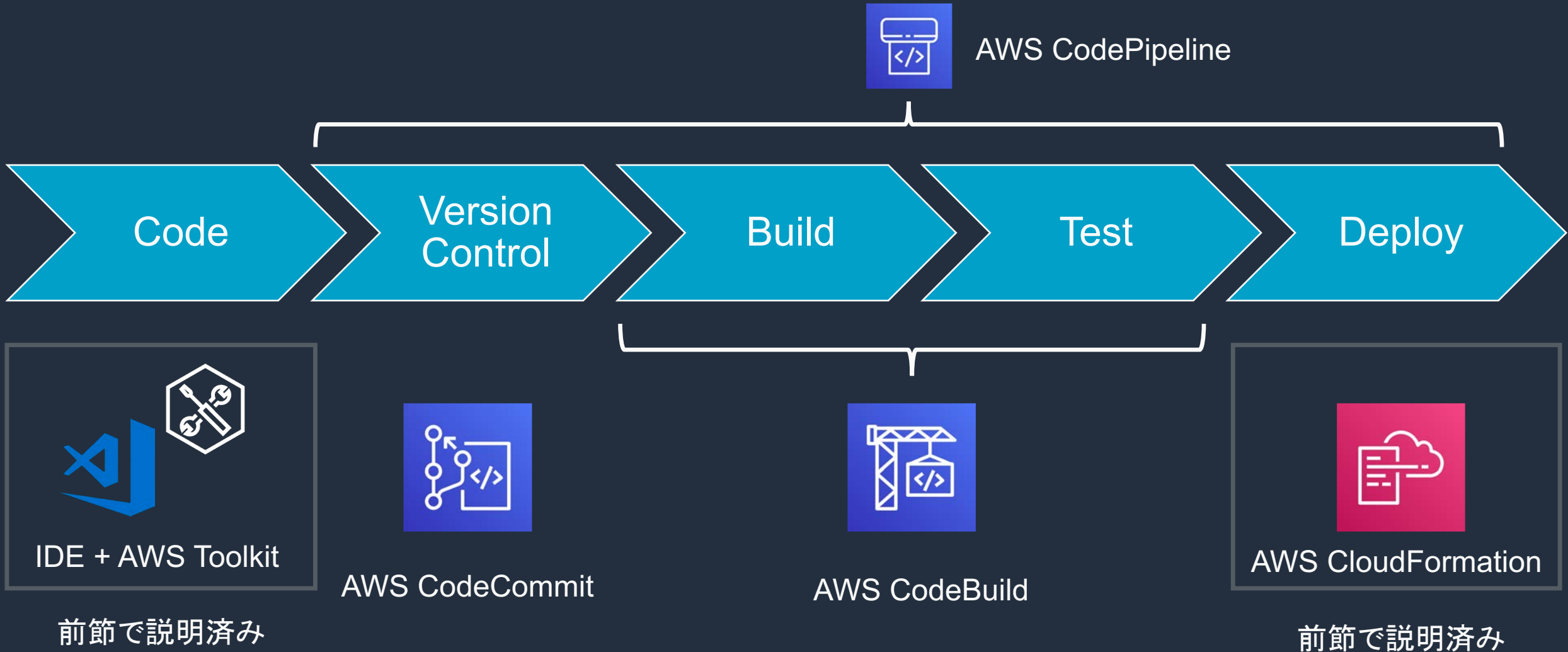


C++

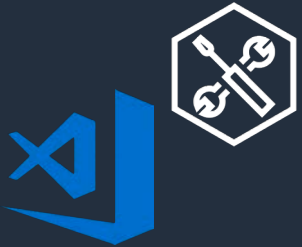
CI/CD for Serverless のワークフロー (復習)



CI/CD for Serverless のワークフロー に利用できるツール



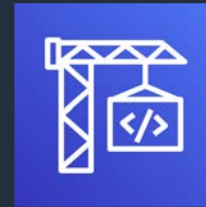
CI/CD for Serverless のよくあるパイプライン化



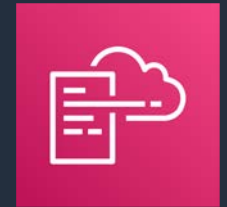
IDE + AWS Toolkit



AWS CodeCommit

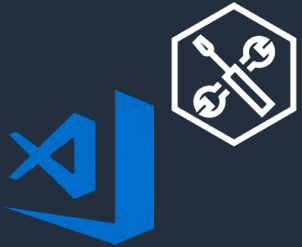
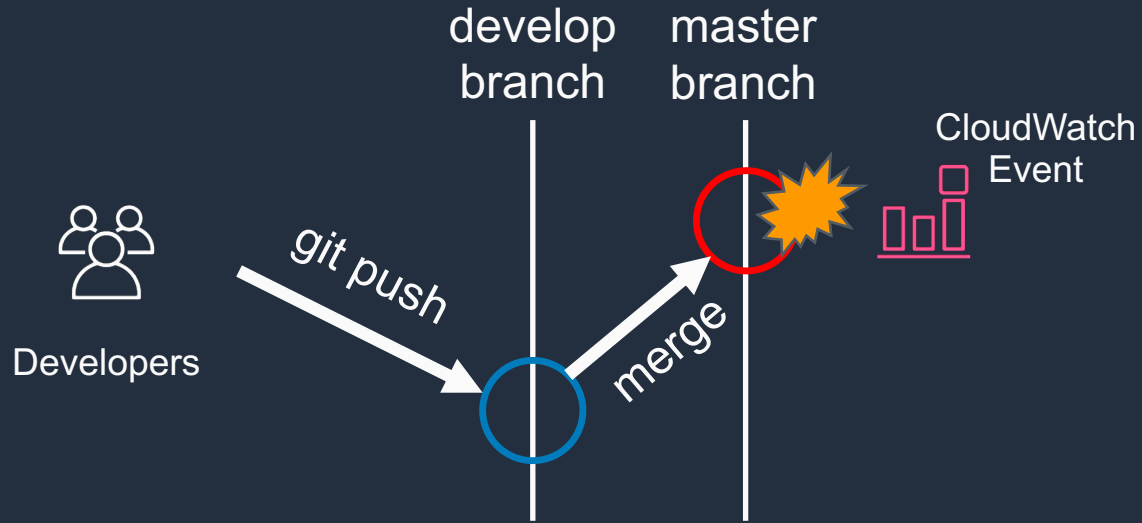


AWS CodeBuild



AWS CloudFormation

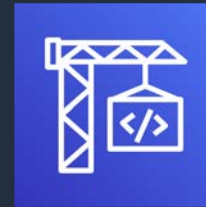
CI/CD for Serverless のよくあるパイプライン化



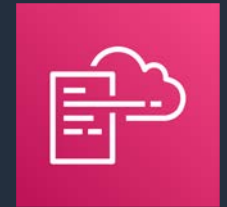
IDE + AWS Toolkit



AWS CodeCommit

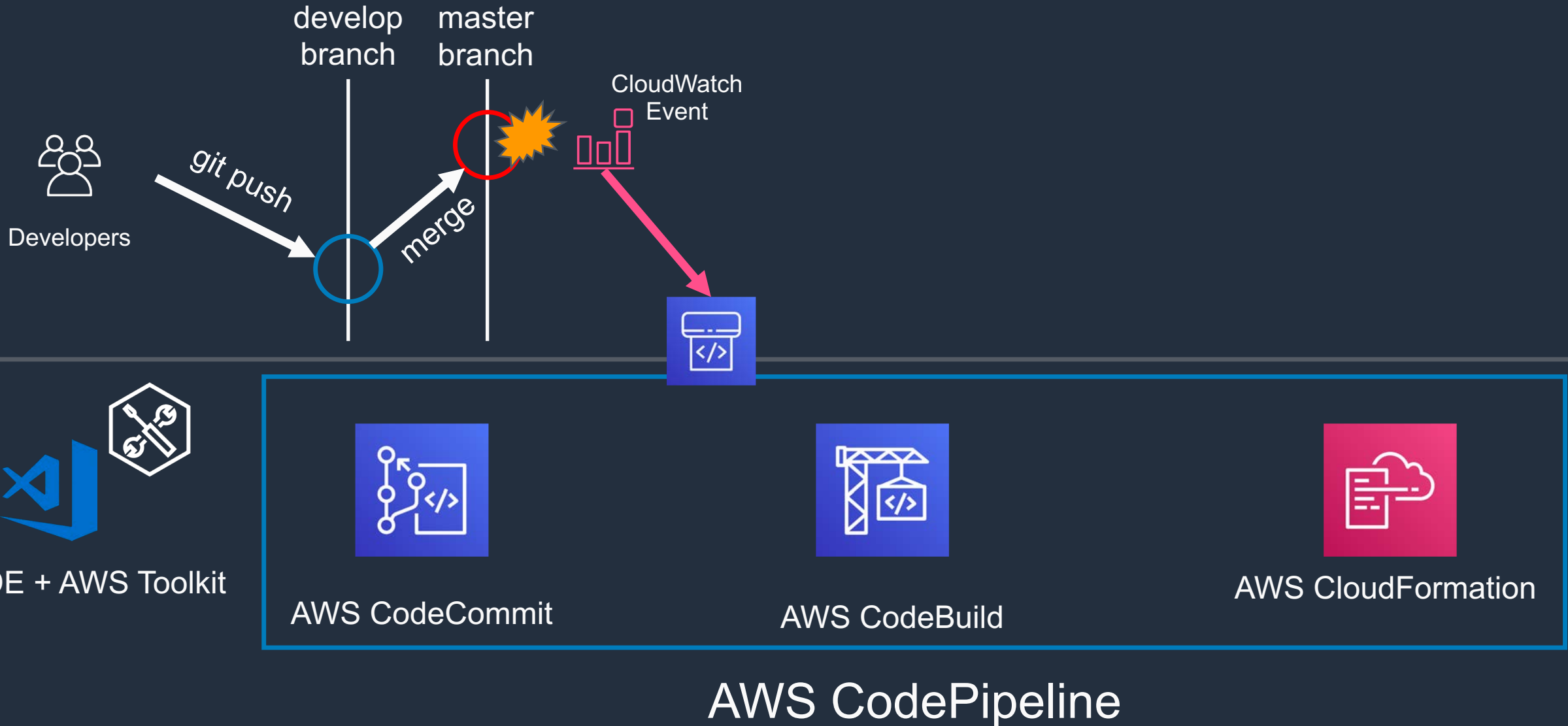


AWS CodeBuild

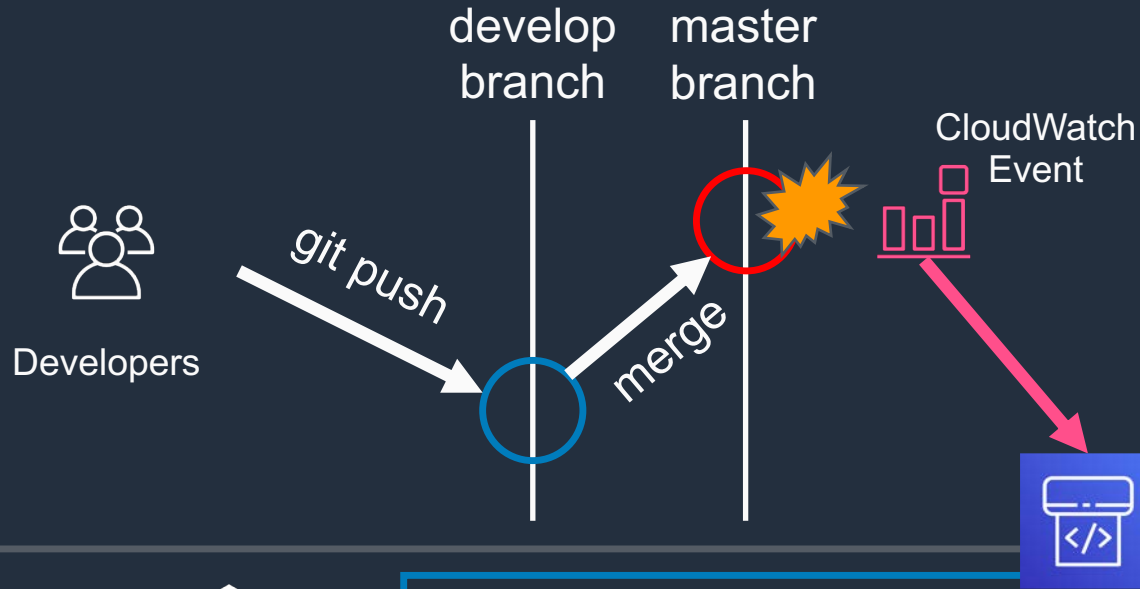


AWS CloudFormation

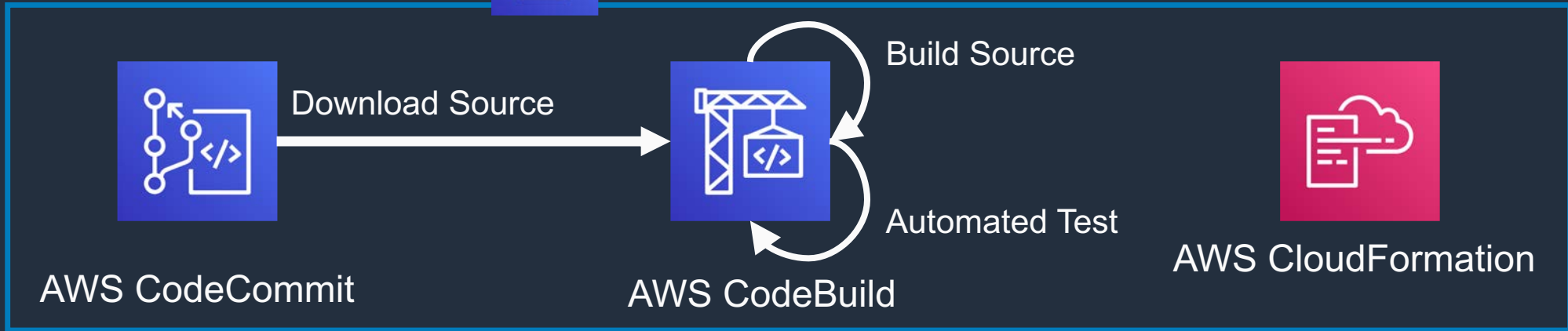
CI/CD for Serverless のよくあるパイプライン化



CI/CD for Serverless のよくあるパイプライン化

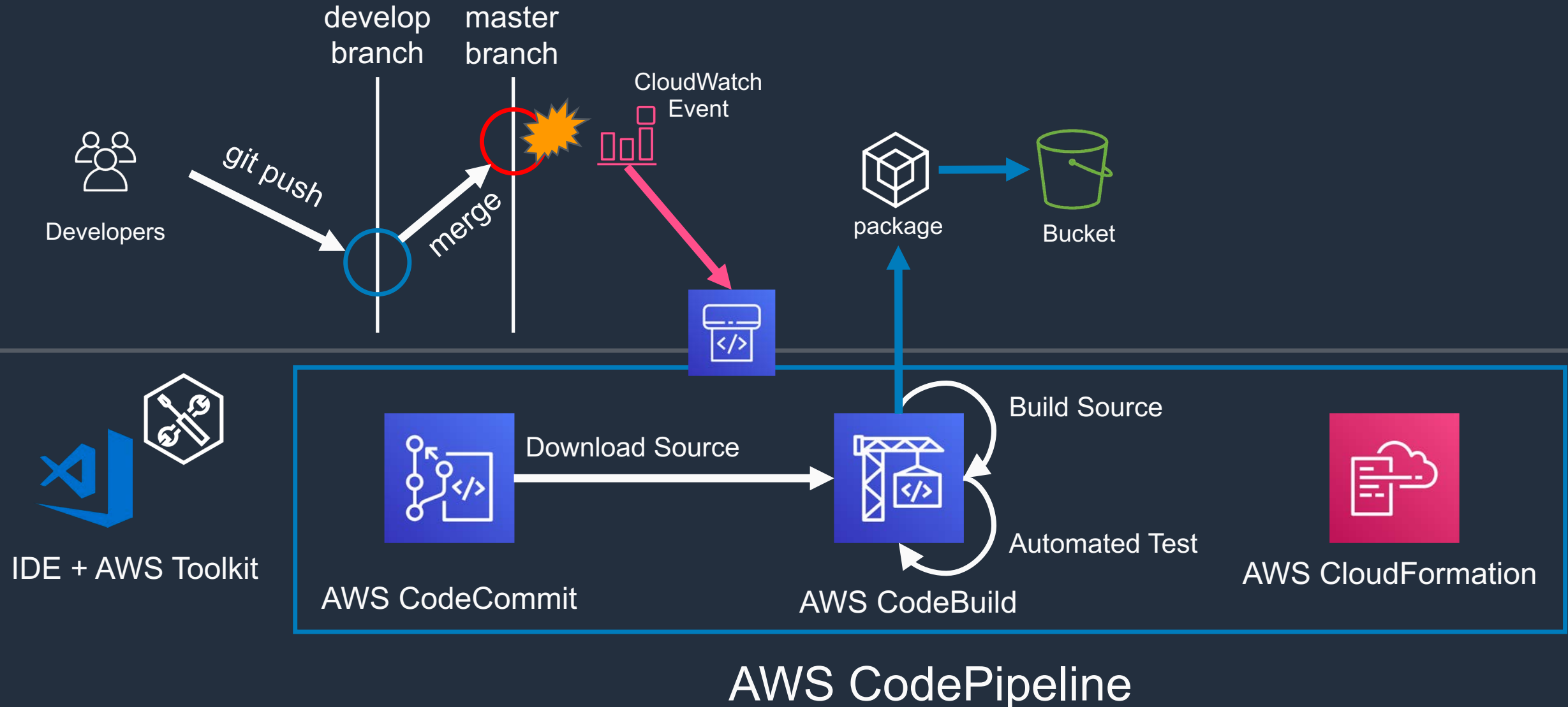


IDE + AWS Toolkit

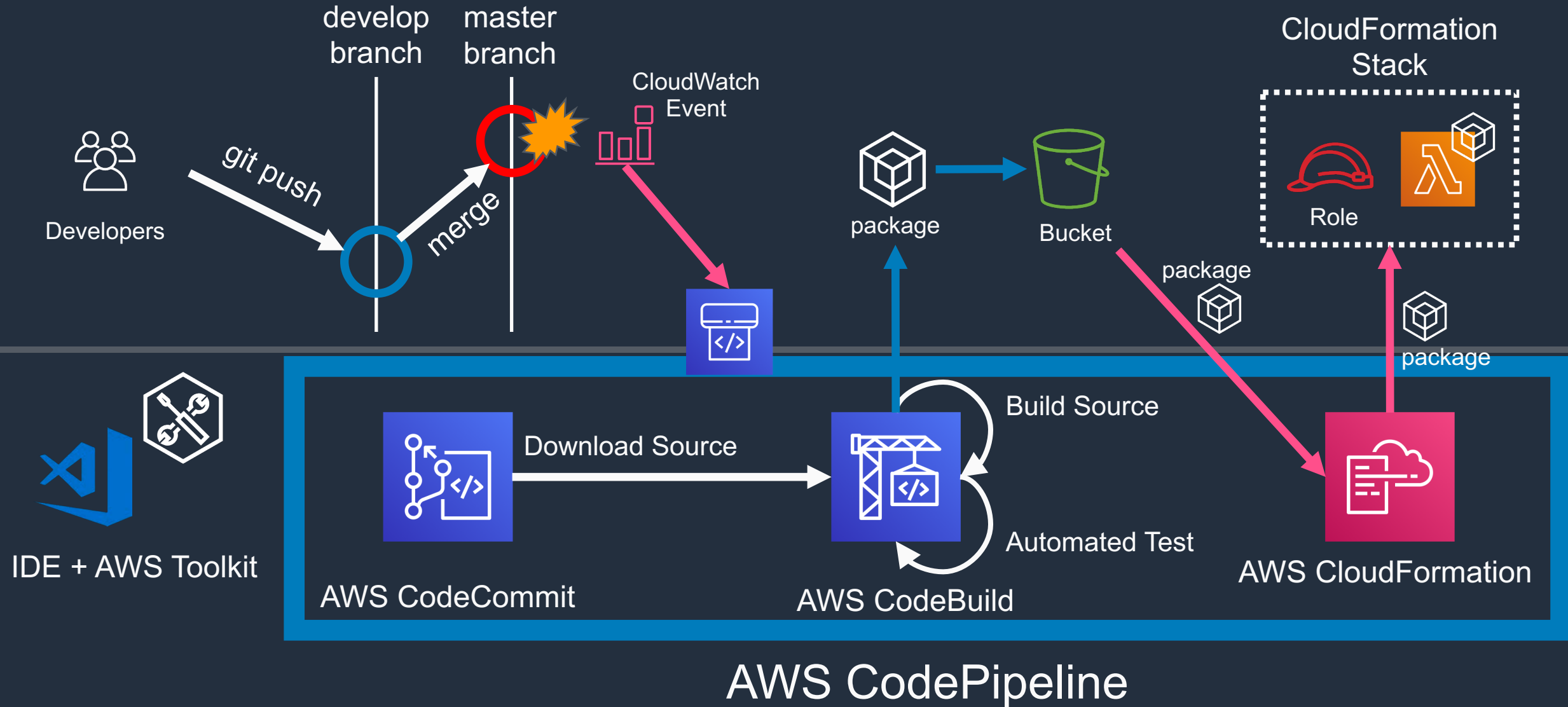


AWS CodePipeline

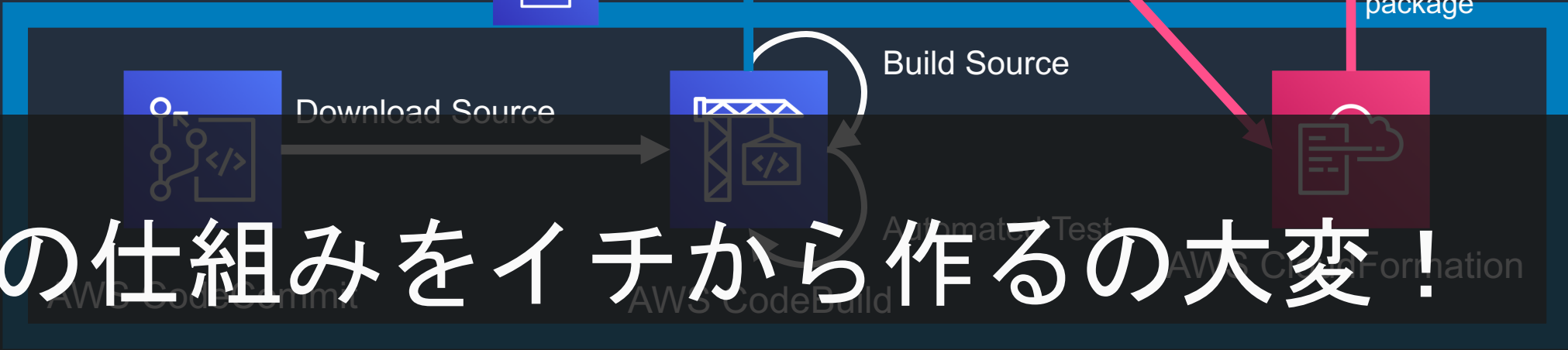
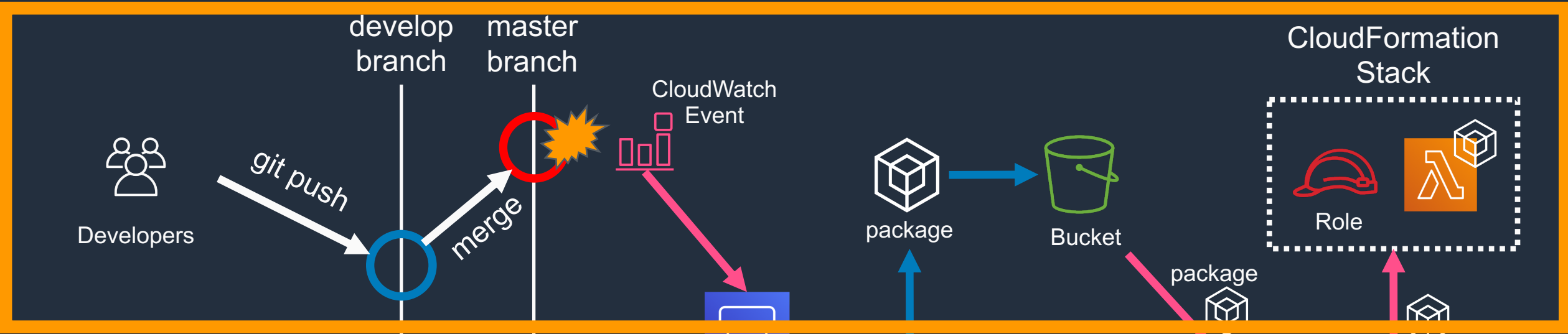
CI/CD for Serverless のよくあるパイプライン化



CI/CD for Serverless のよくあるパイプライン化



CI/CD for Serverless のよくあるパイプライン化



CI/CDの仕組みをイチから作るの大変!

AWS CodePipeline

Serverless Applications Pipeline



AWS Lambda ×

Lambda > Applications

Applications (15) Info ↻ Actions ▼ Create application

🔍 seminar ×

< 1 >

| Name ▼ | Description | Last modified ▲ | Status |
|--------|-------------|-----------------|--------|
|--------|-------------|-----------------|--------|

Serverless Applications の機能でCI/CD !

Serverless Applications Pipeline



1. Applications を選択

2. Create application を押下

Serverless Applications Pipeline



AWS Lambda ×

Lambda > Applications > Create application

- Dashboard
- Applications**
- Functions
- Layers

Create a Lambda application

An AWS Lambda application is a combination of Lambda functions, triggers, and other resources that work together to perform tasks. Choose an option below to create an application with sample code and a continuous delivery pipeline.

SAR(Serverless Application Repository)を使わず、Author from scratch を押下

Other options

AWS Serverless Application Repository

Deploy an application from the [AWS Serverless Application Repository](#) (pipeline not included).

Author from scratch

Create a repository and pipeline to use with your own application.

Author from scratch

Serverless Applications Pipeline



Dashboard

Applications

Functions

Layers

Configure your application

Application details

Application name

Use only lowercase letters, numbers, or hyphens. The maximum length is 20 characters.

Application description

The maximum length is 1000 characters.

Function configuration

Runtime

現在は、*Node.js 10.x*のみが
選択可能

Serverless Applications Pipeline



Dashboard

Applications

Functions

Layers

Source control

Source control service

Choose where to create your application's Git repository.

CodeCommit



Create a repository in your AWS account. Manage SSH keys and HTTP credentials for users in the IAM console.



GitHub



Create a private repository in your GitHub account.



Repository name

seminar-app

The maximum length is 100 characters.

Default では、**Application**名
が自動的に入る

Serverless Applications Pipeline



**Permission Boundaryの作成
チェックボックスが必須**

Permissions [Info](#)

Lambda needs permission to create IAM roles for the resources that support your application. It also needs permission to create a permissions boundary that limits the permissions that can be granted to Lambda functions by modifying execution roles in the application template. [Learn more](#)

Create roles and permissions boundary

Cancel

Create

**Create 押下するとCloudFormation
による生成が始まる**

CloudFormation Stack の確認



CloudFormation Stacks

Stacks (2)

seminar

Active

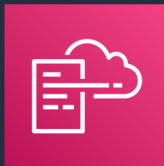
new nested

1

| | Stack name | Status | Created time |
|-----------------------|--------------------------------------|--------------------|-----------------|
| <input type="radio"/> | seminar-app | REVIEW_IN_PROGRESS | 2020-02-10 22:4 |
| <input type="radio"/> | serverlessrepo-seminar-app-toolchain | CREATE_COMPLETE | 2020-02-10 22:4 |

上位と下位のStackが生成される

CloudFormation Stack の確認



上位Stack

seminar-app

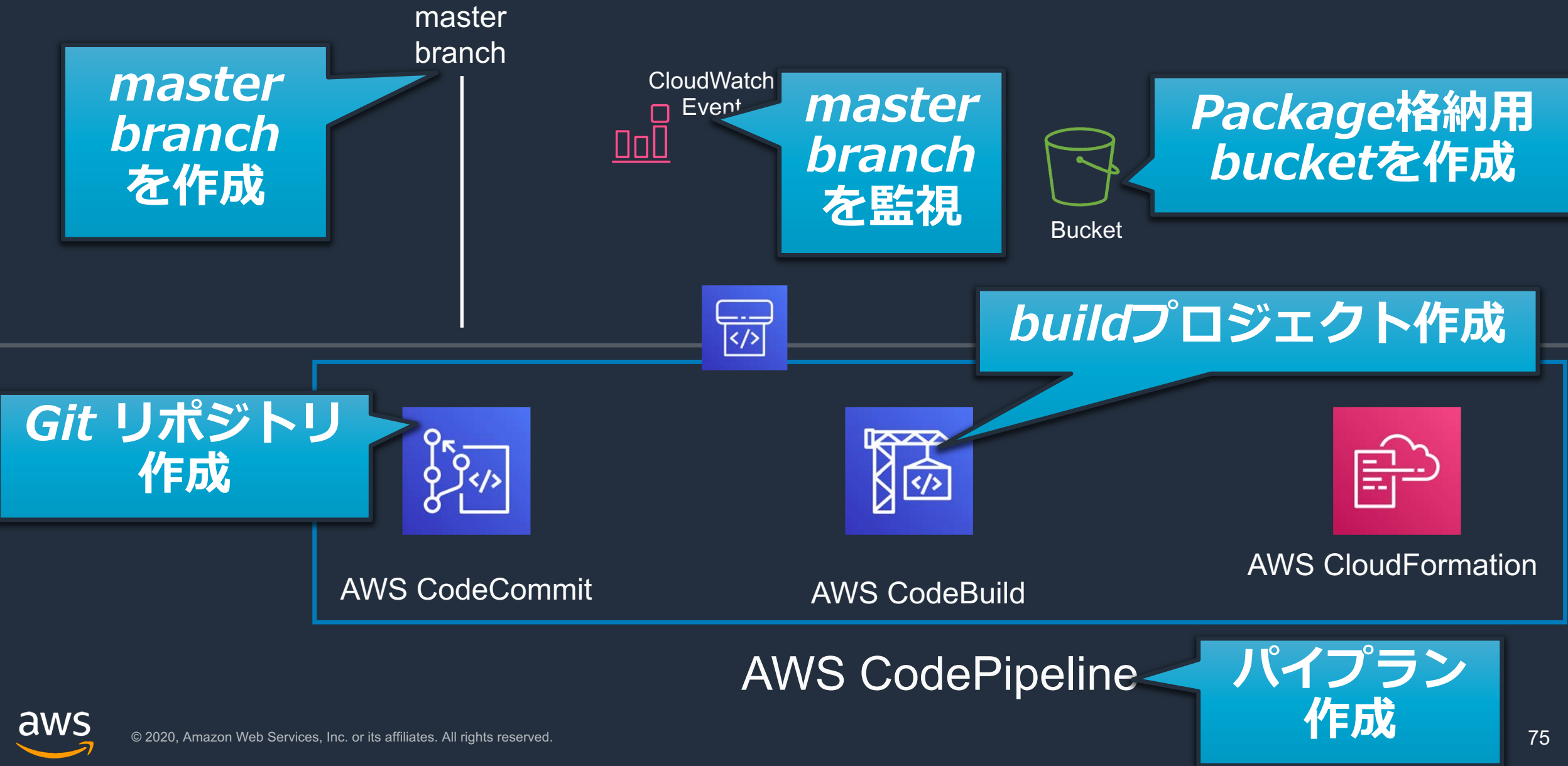
アプリケーションに必要なリソースを生成

下位Stack

serverlessrepo-seminar-app-toolchain

パイプラインに必要なツールリソースを生成

下位 CloudFormation Stackでパイプラインリソース生成

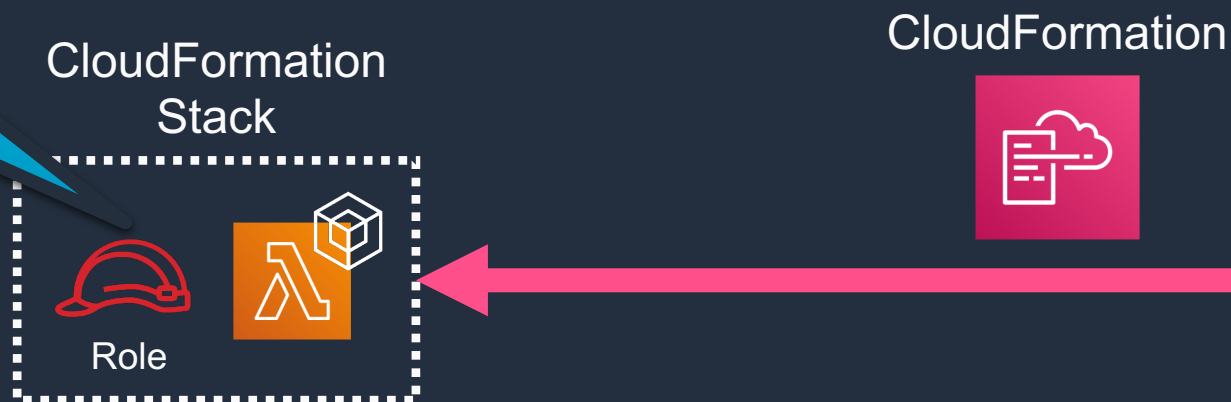


上位 CloudFormation Stackでのリソース生成

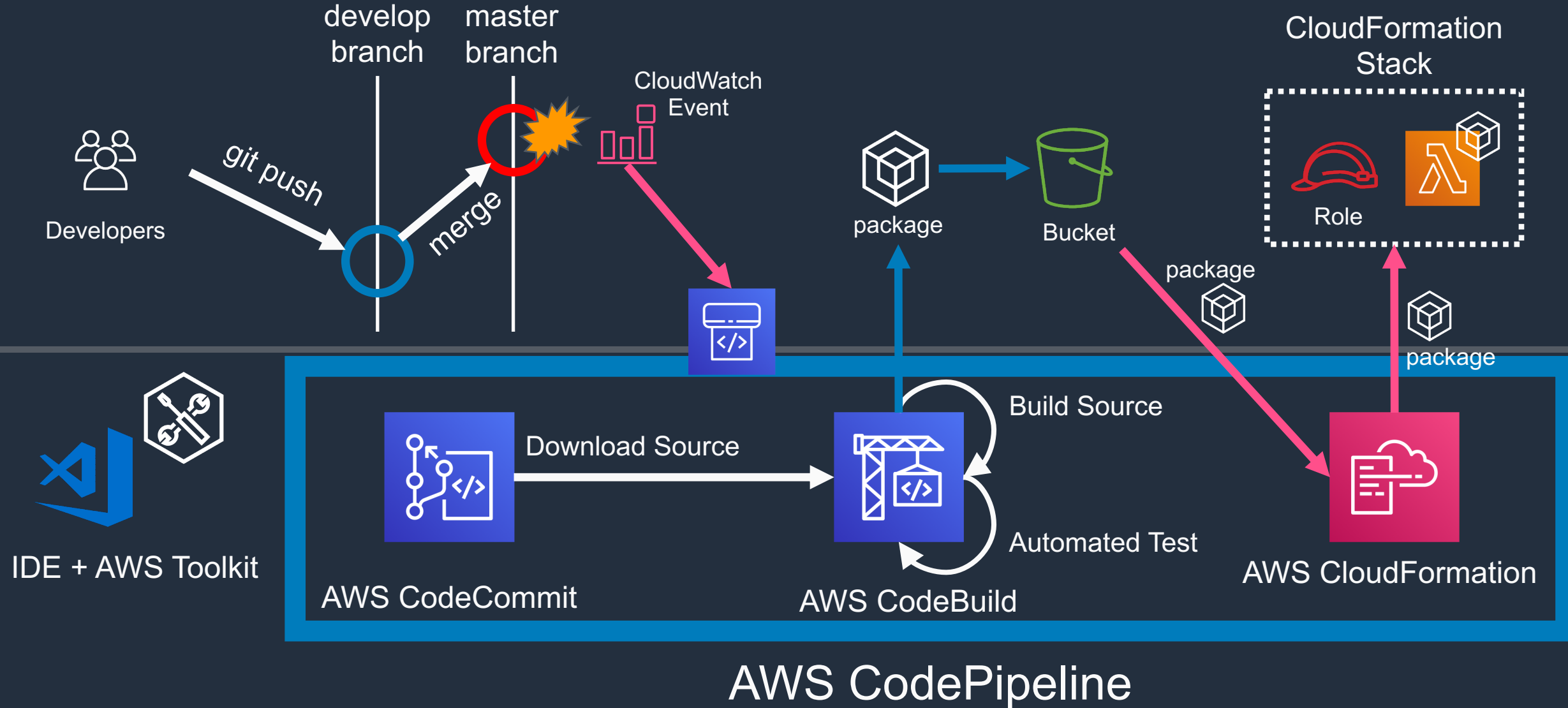
Serverless Applicationsによって、
生成済み下位Stackパイプライン



上位StackはLambda
リソース生成



Lambda関数修正後のmaster mergeでパイプライン起動！



Serverless Applications Pipeline の補足

Serverless Applications Pipelineは Node.jsだけ？



Applications

Functions

Layers

Application details

Application name

seminar-app

Use only lowercase letters, numbers, or hyphens. The maximum length is 20 characters.

Application description

How to create Serverless CICD

The maximum length is 1000 characters.

Function configuration

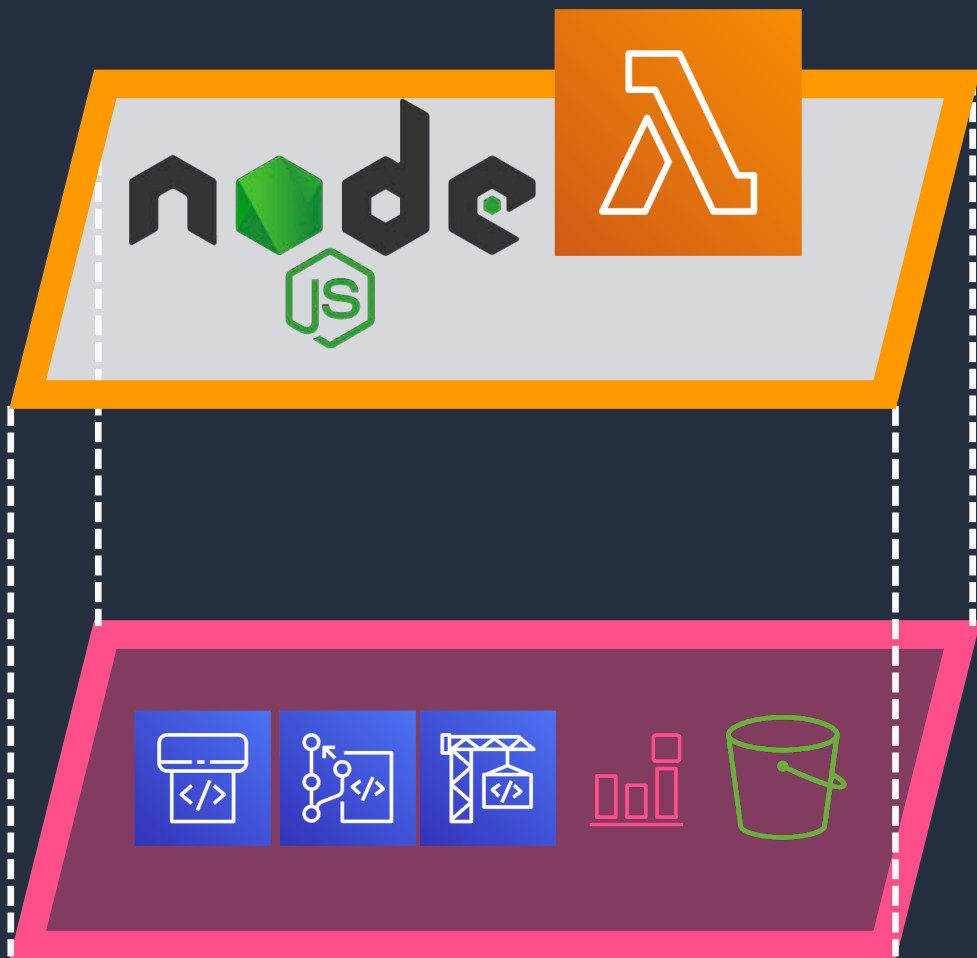
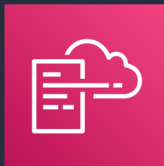
Runtime

Node.js 10.x

Node.js 10.x

現在は、*Node.js 10.x*のみが
選択可能

CloudFormation Stack の上位Stackについて



上位Stack

seminar-app

Node.js アプリケーションの雛形Stack

下位Stack

serverlessrepo-seminar-app-toolchain

アプリに依存しないパイプラインStack

言語置き換え

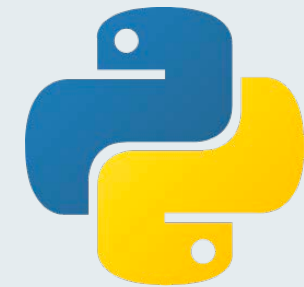
Serverless Applicationsによって生成

SeminarApp

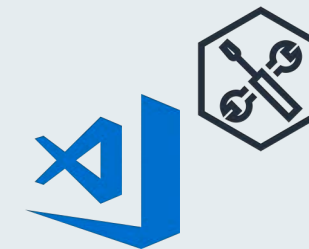


```
├── README.md
├── __tests__
│   └── unit
│       └── handlers
│           └── hello-from-lambda.test.js
├── buildspec.yml
├── package.json
├── src
│   └── handlers
│       └── hello-from-lambda.js
└── template.yml
```

SeminarApp



Python




```
├── README.md
├── events
│   └── event.json
├── hello_world
│   └── app.py
│       └── requirements.txt
├── template.yml
├── tests
│   └── unit
│       └── test_handler.py
```

IDE + AWS Toolkit によって生成


言語置き換え

*Serverless Applications*によって生成

```
SeminarApp
├── README.md
├── __tests__
│   └── unit
│       └── handlers
│           └── hello-from-lambda.test.js
├── buildspec.yml
├── package.json
├── src
│   └── handlers
│       └── hello-from-lambda.js
└── template.yml
```



```
SeminarApp
├── README.md
└── events
```



(置き換え時の注意点)
buildspec.yml だけは、保持したい



AWS CodeBuild

```
├── template.yml
└── tests
    └── unit
        └── test_handler.py
```

IDE + AWS Toolkit によって生成

言語置き換え

*Serverless Applications*によって生成

SeminarApp

- ├── README.md
- ├── __tests__
 - └── unit
 - └── handlers
 - └── hello-from-lambda.test.js



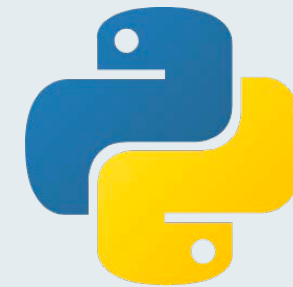
├── **buildspec.yml**

- ├── package.json
- ├── src
 - └── handlers
 - └── hello-from-lambda.js
- └── template.yml

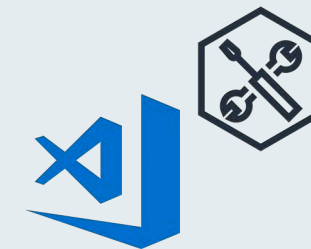
ローカル端末上で置き換え

SeminarApp

- ├── README.md
- ├── events
 - └── event.json
- ├── hello_world
- ├── app.py
- ├── requirements.txt
- ├── template.yml
- └── tests
 - └── unit
 - └── test_handler.py



Python



IDE + AWS Toolkit によって生成



言語置き換え

Serverless Applicationsによって生成

SeminarApp

├── README.md

├── **buildspec.yml**

├── events

├── event.json

├── hello_world

├── app.py

├── requirements.txt

├── template.yml

├── tests

├── unit

├── test_handler.py



Python

buildspec.yml のPython化編集 (抜粋)

```
version: 0.2
```

```
phases:
```

```
  install:
```

```
    commands:
```

```
      - npm install
```

```
  build:
```

```
    commands:
```

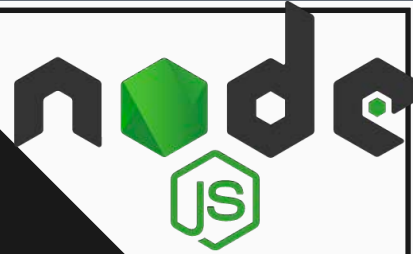
```
      - aws cloudformation package --template  
        template.yml --s3-bucket $S3_BUCKET  
        --output-template template-export.yml
```

```
artifacts:
```

```
  type: zip
```

```
  files:
```

```
    - template-export.yml
```



書き換え

```
version: 0.2
```

```
phases:
```

```
  install:
```

```
    commands:
```

```
      - pip install -r  
        requirements.txt
```

```
  build:
```

```
    commands:
```

```
      - aws cloudformation package --template  
        template.yml --s3-bucket $S3_BUCKET  
        --output-template template-export.yml
```

```
artifacts:
```

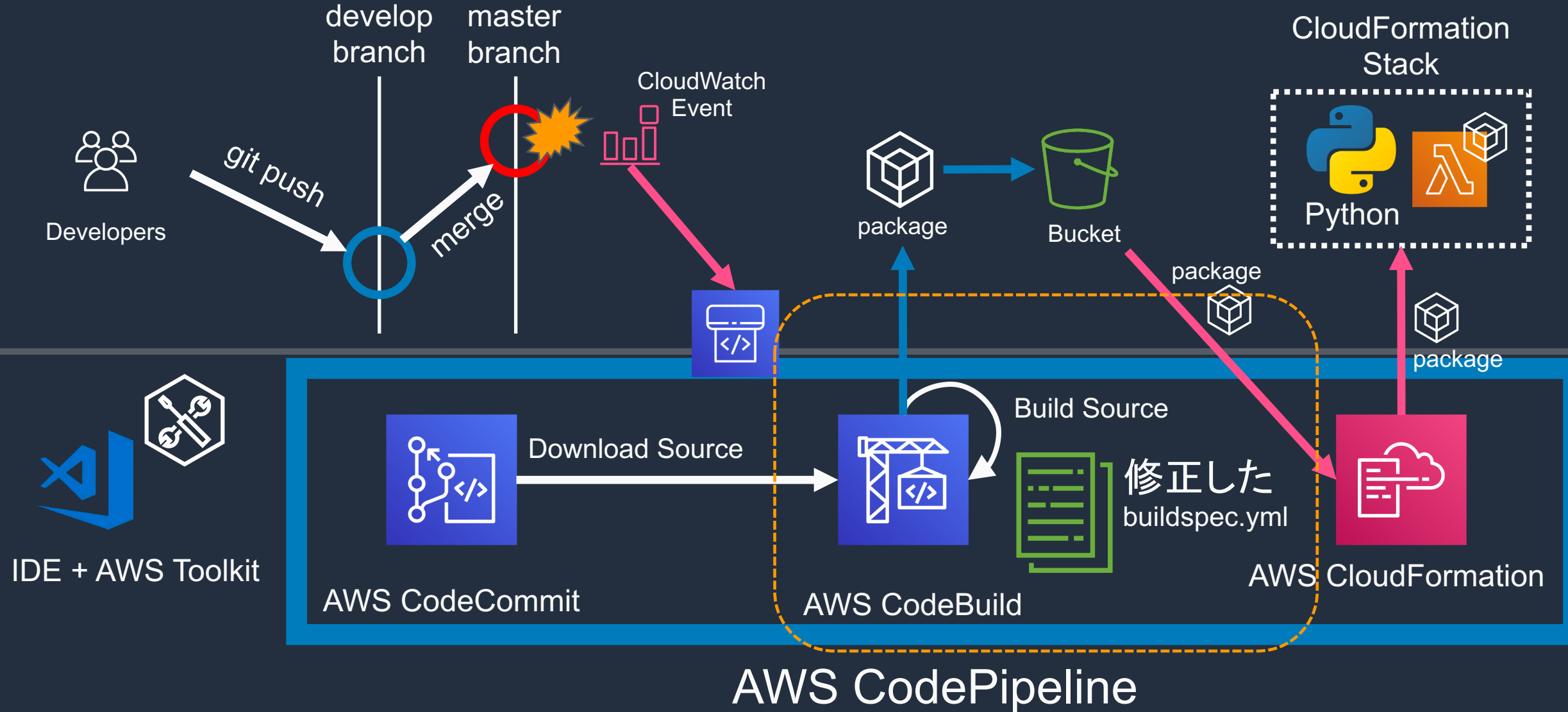
```
  type: zip
```

```
  files:
```

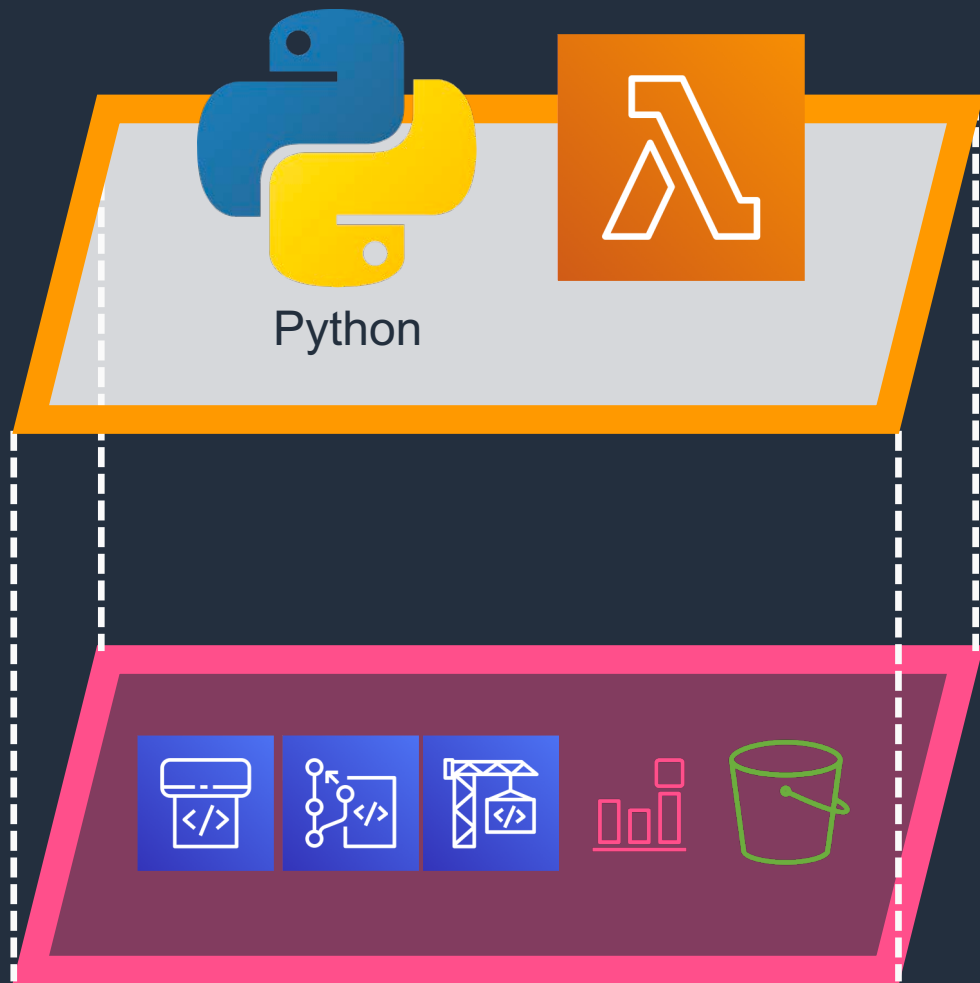
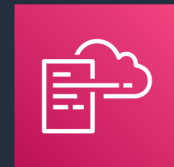
```
    - template-export.yml
```



修正後にmaster mergeでパイプライン起動！



CloudFormation Stack の確認



上位Stack

seminar-app

Node.js を Python に書き換え済み

下位Stack

serverlessrepo-seminar-app-toolchain

とくに修正なし

まとめ

- サーバーレスの開発も、開発者のお気に入りのIDEで行うことができる
- AWS SAMを使って、CLIによるサーバーレス開発を高速化できる
- CI/CDパイプラインを作成して、Git Pushから自動テストの流れまでを作っておき、属人化する開発、デリバリーから脱却できる

