Serverless

# Concepts - AWS

## Functions

A serverless function in AWS is called an AWS lambda function. A function is a independent unit of code, stateless, and performs one job.

## Event

Anything that triggers a function is regarded as an event. Common events are:

* Endpoint request
* AWS S3 bucket upload
* CloudWatch timer

## Resources

Components which the functions use and store state in, such as:

* AWS DynamoDB table
* AWS S3 bucket
* SNS topic

## Service

Service is the frameworks unit of organisation. Similar to a project file, however there can be multiple services for a single application. Services are described in a YAML or JSON file and are commonly called serverless.yml and look like:

service: users

functions: # Your "Functions"

usersCreate:

events: # The "Events" that trigger this function

- http: post users/create

usersDelete:

events:

- http: delete users/delete

resources: # The "Resources" your "Functions" use. Raw AWS CloudFormation goes in here.

# Initialize Serverless

## Init

Add serverless to package:

yarn add serverless

Run serverless and follow setup:

yarn serverless

## Create AWS IAM User

AWS > IAM > Users > Add user

Create User access policy with JSON:

|  |
| --- |
| { |
|  | "Statement": [ |
|  | { |
|  | "Action": [ |
|  | "apigateway:\*", |
|  | "cloudformation:CancelUpdateStack", |
|  | "cloudformation:ContinueUpdateRollback", |
|  | "cloudformation:CreateChangeSet", |
|  | "cloudformation:CreateStack", |
|  | "cloudformation:CreateUploadBucket", |
|  | "cloudformation:DeleteStack", |
|  | "cloudformation:Describe\*", |
|  | "cloudformation:EstimateTemplateCost", |
|  | "cloudformation:ExecuteChangeSet", |
|  | "cloudformation:Get\*", |
|  | "cloudformation:List\*", |
|  | "cloudformation:UpdateStack", |
|  | "cloudformation:UpdateTerminationProtection", |
|  | "cloudformation:ValidateTemplate", |
|  | "dynamodb:CreateTable", |
|  | "dynamodb:DeleteTable", |
|  | "dynamodb:DescribeTable", |
|  | "dynamodb:DescribeTimeToLive", |
|  | "dynamodb:UpdateTimeToLive", |
|  | "ec2:AttachInternetGateway", |
|  | "ec2:AuthorizeSecurityGroupIngress", |
|  | "ec2:CreateInternetGateway", |
|  | "ec2:CreateNetworkAcl", |
|  | "ec2:CreateNetworkAclEntry", |
|  | "ec2:CreateRouteTable", |
|  | "ec2:CreateSecurityGroup", |
|  | "ec2:CreateSubnet", |
|  | "ec2:CreateTags", |
|  | "ec2:CreateVpc", |
|  | "ec2:DeleteInternetGateway", |
|  | "ec2:DeleteNetworkAcl", |
|  | "ec2:DeleteNetworkAclEntry", |
|  | "ec2:DeleteRouteTable", |
|  | "ec2:DeleteSecurityGroup", |
|  | "ec2:DeleteSubnet", |
|  | "ec2:DeleteVpc", |
|  | "ec2:Describe\*", |
|  | "ec2:DetachInternetGateway", |
|  | "ec2:ModifyVpcAttribute", |
|  | "events:DeleteRule", |
|  | "events:DescribeRule", |
|  | "events:ListRuleNamesByTarget", |
|  | "events:ListRules", |
|  | "events:ListTargetsByRule", |
|  | "events:PutRule", |
|  | "events:PutTargets", |
|  | "events:RemoveTargets", |
|  | "iam:AttachRolePolicy", |
|  | "iam:CreateRole", |
|  | "iam:DeleteRole", |
|  | "iam:DeleteRolePolicy", |
|  | "iam:DetachRolePolicy", |
|  | "iam:GetRole", |
|  | "iam:PassRole", |
|  | "iam:PutRolePolicy", |
|  | "iot:CreateTopicRule", |
|  | "iot:DeleteTopicRule", |
|  | "iot:DisableTopicRule", |
|  | "iot:EnableTopicRule", |
|  | "iot:ReplaceTopicRule", |
|  | "kinesis:CreateStream", |
|  | "kinesis:DeleteStream", |
|  | "kinesis:DescribeStream", |
|  | "lambda:\*", |
|  | "logs:CreateLogGroup", |
|  | "logs:DeleteLogGroup", |
|  | "logs:DescribeLogGroups", |
|  | "logs:DescribeLogStreams", |
|  | "logs:FilterLogEvents", |
|  | "logs:GetLogEvents", |
|  | "logs:PutSubscriptionFilter", |
|  | "s3:CreateBucket", |
|  | "s3:DeleteBucket", |
|  | "s3:DeleteBucketPolicy", |
|  | "s3:DeleteObject", |
|  | "s3:DeleteObjectVersion", |
|  | "s3:GetObject", |
|  | "s3:GetObjectVersion", |
|  | "s3:ListAllMyBuckets", |
|  | "s3:ListBucket", |
|  | "s3:PutBucketNotification", |
|  | "s3:PutBucketPolicy", |
|  | "s3:PutBucketTagging", |
|  | "s3:PutBucketWebsite", |
|  | "s3:PutEncryptionConfiguration", |
|  | "s3:PutObject", |
|  | "sns:CreateTopic", |
|  | "sns:DeleteTopic", |
|  | "sns:GetSubscriptionAttributes", |
|  | "sns:GetTopicAttributes", |
|  | "sns:ListSubscriptions", |
|  | "sns:ListSubscriptionsByTopic", |
|  | "sns:ListTopics", |
|  | "sns:SetSubscriptionAttributes", |
|  | "sns:SetTopicAttributes", |
|  | "sns:Subscribe", |
|  | "sns:Unsubscribe", |
|  | "states:CreateStateMachine", |
|  | "states:DeleteStateMachine" |
|  | ], |
|  | "Effect": "Allow", |
|  | "Resource": "\*" |
|  | } |
|  | ], |
|  | "Version": "2012-10-17" |
|  | } |

Create user and login with API keys by adding default aws profile to serverless:

serverless config credentials --provider aws --key <iam-key> --secret <iam-secret>

# Setup Service

## Add Handlers

Add handler functions in root directory. For example:

'use strict';

module.exports.hello = async event => {

return {

statusCode: 200,

body: JSON.stringify(

{

message: 'Go Serverless v1.0! Your function executed successfully!',

input: event,

},

null,

2

),

};

};

Will return the node event triggered to the user as a JSON.

## Add Config

Set up a service configuration file called serverless.yml in the root directory of the service. Include all the config for each endpoint including handlers (referenced by relative paths) and events:

service: bujjit-serverless

app: bujjit

org: bujjit

frameworkVersion: '2'

provider:

name: aws

runtime: nodejs12.x

stage: dev

region: eu-west-2

functions:

hello:

handler: handler.hello

events:

- http:

path: hello

method: post

## Deploy Service

Deploy service to aws using the serverless cli:

serverless deploy -v

## Update Service

Update the service by redeploying a service with the same name

## Remove Service

Remove the service using the serverless cli, the service which will be removed will be the one that is named in the current directory serverless.yml. Additional conditions such as location and stage can be added to remove specific instances of the service:

serverless remove --stage dev --region us-east-1