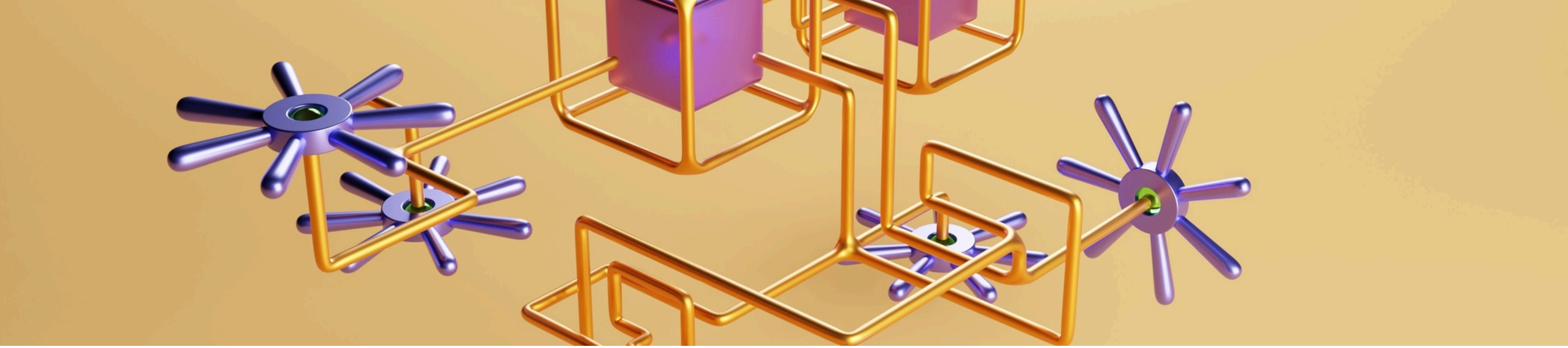


Introduction to Serverless Framework & Lambdas

Building Scalable Applications with AWS

Gilad Even-Tov

gilad@eventov.com

Agenda

Introduction to Serverless

The Framework

1. Open-source framework.
2. Simplifies the deployment & management of serverless applications.
3. Application code + IAC in one place.

What Can We Build

1. Web APIs
2. Background Jobs
3. Scheduled Jobs
4. Standalone Lambdas
5. *Lots more AWS goodies*

High Level Anatomy of a Serverless Project

Components of a Serverless Framework Project

1

serverless.yml file

- The heart of a Serverless Framework project
- Configuration for the serverless resources, like
  - **functions**
  - **events**
  - **other resources**

2

Function Code

- Built around **functions**
- Each **function** should be do as few things as possible
- Perform specific tasks or respond to events
- Mostly language agnostic (application code can be in any language lambda supports)

3

Other Resources

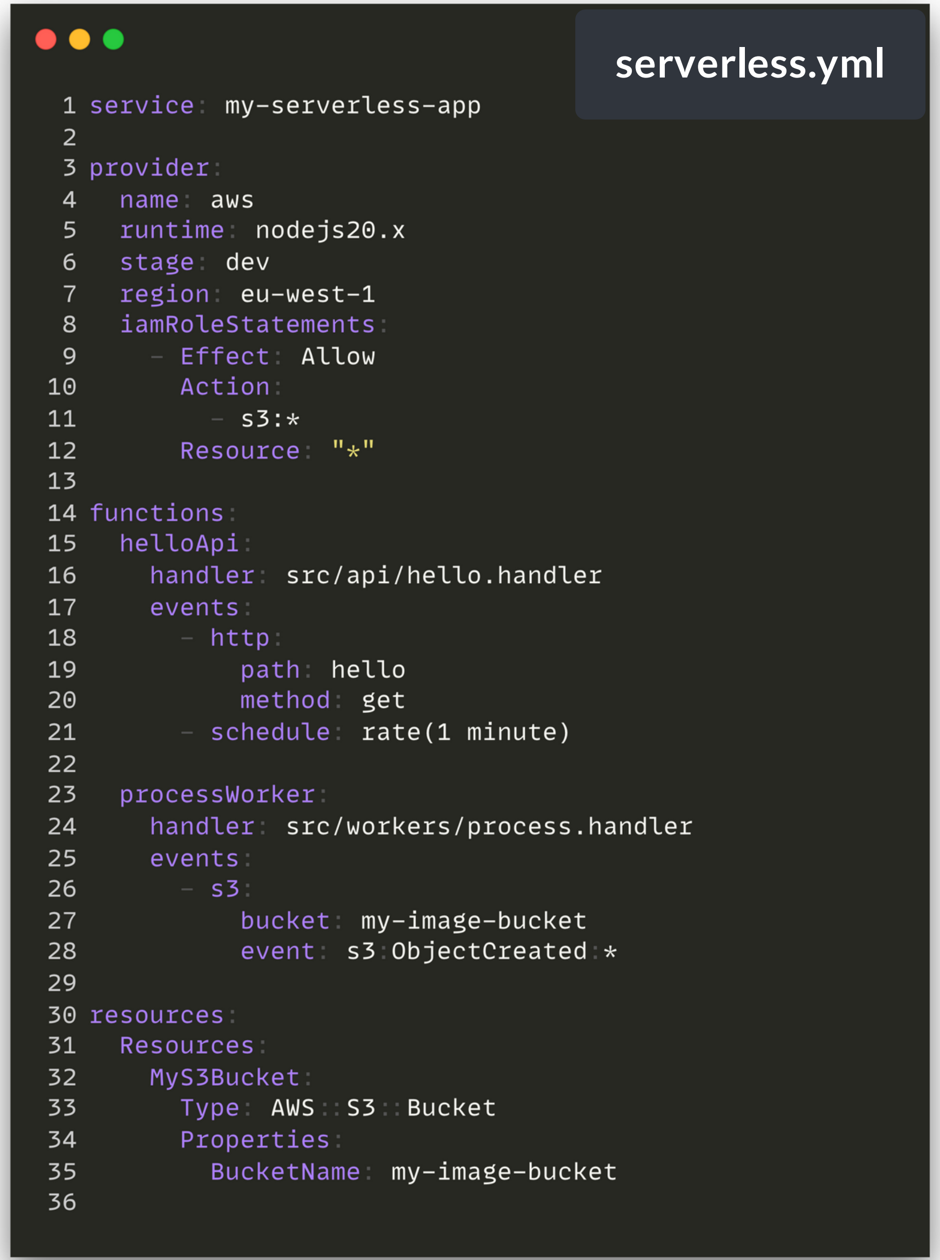
- Other AWS resources can be created through serverless. eg S3 buckets, Dynamo tables, IAM policies, ...

serverless

AWS Resources

You can create many AWS resources and use as events (triggers) for functions

- Lambda
- API Gateway
- Cognito
- S3
- DynamoDB
- SQS
- Step Functions
- Scheduler
- *More...*



serverless.yml

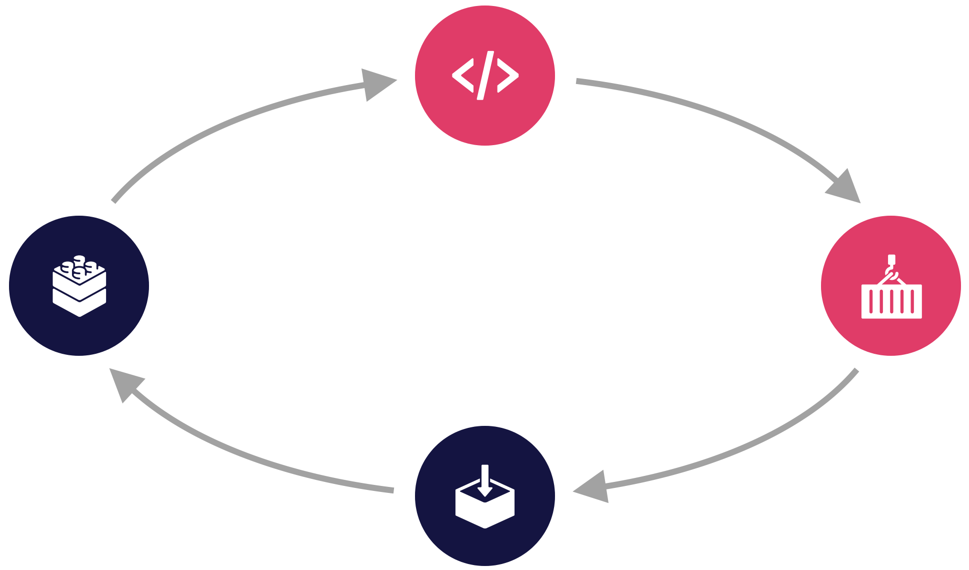
```yaml
service: my-serverless-app

provider:
  name: aws
  runtime: nodejs20.x
  stage: dev
  region: eu-west-1
  iamRoleStatements:
    - Effect: Allow
      Action:
        - s3:*
      Resource: "*"

functions:
  helloApi:
    handler: src/api/hello.handler
    events:
      - http:
          path: hello
          method: get
      - schedule: rate(1 minute)

  processWorker:
    handler: src/workers/process.handler
    events:
      - s3:
          bucket: my-image-bucket
          event: s3:ObjectCreated:*

resources:
  Resources:
    MyS3Bucket:
      Type: AWS::S3::Bucket
      Properties:
        BucketName: my-image-bucket
```

Simple Example

```
my-serverless-app/
├── handler.js
├── serverless.yml
└── src/
    ├── api/
    │   └── hello.js
    └── workers/
        └── process.js
```

- Here we have a sls project that creates 2 python functions, custom permissions, and an S3 bucket
- One function is invoked via a public API as well as invoked every minute.
  - via API Gateway
  - via Amazon EventBridge Scheduler (CRON)
- Another function is invoked whenever an S3 object is created in the specified bucket.
  - via S3 Event Notification
- An S3 bucket is also created.

Understanding the Deployment Process



Configuration Management

Configure serverless.yml and write functions

Dependency Management

External dependencies or libraries.

Packaging

SLS will create a deployment package (including deps)

Can even be a docker image

SLS Deploy

SLS will create and manage the lifecycle of cloudformation stacks and resources

Data Store Options

DynamoDB

Built for serverless environments.

Document data store

Managed service

RDS

Relational DB

Must be used with care since DB connections wont necessarily scale in line with lambda

RDS proxy and other serverless offerings exist to help alleviate this.

As a Web API

API Gateway Integration

API Gateway handles request routing, authorization, authentication, and response formatting.

Lambda Functions as Endpoints

Each HTTP endpoint in your Web API can be implemented as a separate Lambda function.

Serverless Authentication and Authorization

Integrate with Amazon Cognito or third-party identity providers. To handle user authentication, token generation, and access control, outside of business logic.

Amazon
API Gateway

As Background/CRON Jobs

Background Jobs

Amazon SQS (Simple Queue Service) can be used as a message queue to decouple the components of your serverless application and trigger functions.

Event Driven Processing

Services like AWS EventBridge or third-party cron job providers can be used to orchestrate and manage scheduled events seamlessly within your serverless application

Orchestration with AWS Step Functions

define state machines that execute a series of steps or functions in a defined sequence, with support for branching, parallel execution, error handling, and retries