# Writing serverless app with AWS SAM

Aleksandr Barmin

January 2022

# Aleksandr Barmin

Chief Software Engineer I

AWS Solution Architect Associate

Email: Aleksandr_Barmin@epam.com

# Journey to serverless

From hardware and VMs to ephemeral functions

## Journey to serverless

### Hardware and virtual machines

- Docker, LXC, Windows Containers, rkt, Podman
- Docker Swarm, K8S
- ECS, EKS, GKE

### Containers

- Plain hardware
- VMs like ESXi, Xen, KVM
- AWS EC2, Azure VMs, GCP VMs

### Serverless

- Apache Flink
- AWS Lambda
- Google Cloud Function
- Azure Function

## What is serverless?

- You’re focused on the business logic, cloud provider is responsible for the infrastructure
- Pay as you go, scaled based on the workload

### Compute

- AWS Lambda
- AWS Fargate

### Integration

- AWS Step Functions
- Amazon Event Bridge, SQS, SNS
- Amazon API Gateway, App Sync

### Storage

- Amazon S3
- Amazon DynamoDB, RDS
- Amazon Aurora Serverless

## Infrastructure as code

- Create templates that describe and model AWS infrastructure
- CloudFormation then provisions AWS resources based on dependency needs
- Version control/replicate/update the templates like app code
- Integrates with development, CI/CD, management tools
- No additional charge to use
- Supports YAML and JSON notations

### Headers

- Description of what you stack does, contains, etc.

### Parameters

- Provision time values that add structured flexibility and customization

### Mappings

- Pre-defined conditional case statements

### Conditionals

- Conditional values set via evaluation of passed references

### Resources

- AWS resource definitions

### Outputs

- Resulting attributes of stack resource creation

## AWS SAM

Serverless Application Model

## AWS SAM

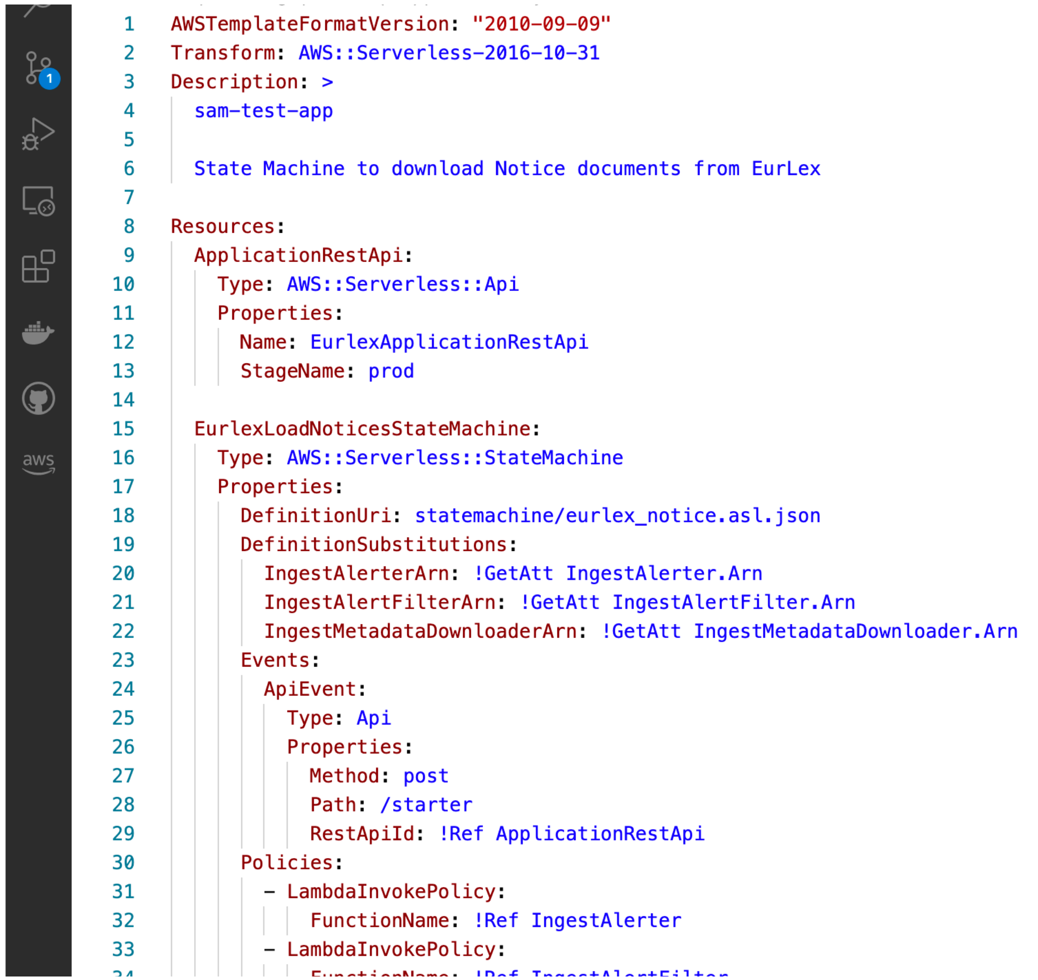
### What is SAM?

- **AWS SAM template specification** - it provides you with a simple and clean syntax to describe the functions, APIs, permissions, configurations, and events that make up a serverless application.
- **AWS SAM command line interface (AWS SAM CLI).** You use this tool to build serverless applications that are defined by AWS SAM templates

### Benefits of using AWS SAM

- Single-deployment configuration
- Extension of AWS CloudFormation
- Built-in best practices
- Local debugging and testing
- Deep integration with development tools

## Single-deployment configuration



```
AWSTemplateFormatVersion: "2010-09-09"
Transform: AWS::Serverless-2016-10-31
Description: >
  sam-test-app

  State Machine to download Notice documents from EurLex

Resources:
  ApplicationRestApi:
    Type: AWS::Serverless::Api
    Properties:
      Name: EurlexApplicationRestApi
      StageName: prod

  EurlexLoadNoticesStateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      DefinitionUri: statemachine/eurlex_notice.asl.json
      DefinitionSubstitutions:
        IngestAlerterArn: !GetAtt IngestAlerter.Arn
        IngestAlertFilterArn: !GetAtt IngestAlertFilter.Arn
        IngestMetadataDownloaderArn: !GetAtt IngestMetadataDownloader.Arn
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Method: post
            Path: /starter
            RestApiId: !Ref ApplicationRestApi
      Policies:
        - LambdaInvokePolicy:
            FunctionName: !Ref IngestAlerter
        - LambdaInvokePolicy:
```

Use SAM to organize related components, share configuration such as memory and timeouts between resources, and deploy all related resources together as a single, versioned entity.

## SAM Benefits

### Local Testing and Debugging

Use SAM CLI to step-through and debug your code. It provides a Lambda-like execution environment locally and helps you catch issues upfront.

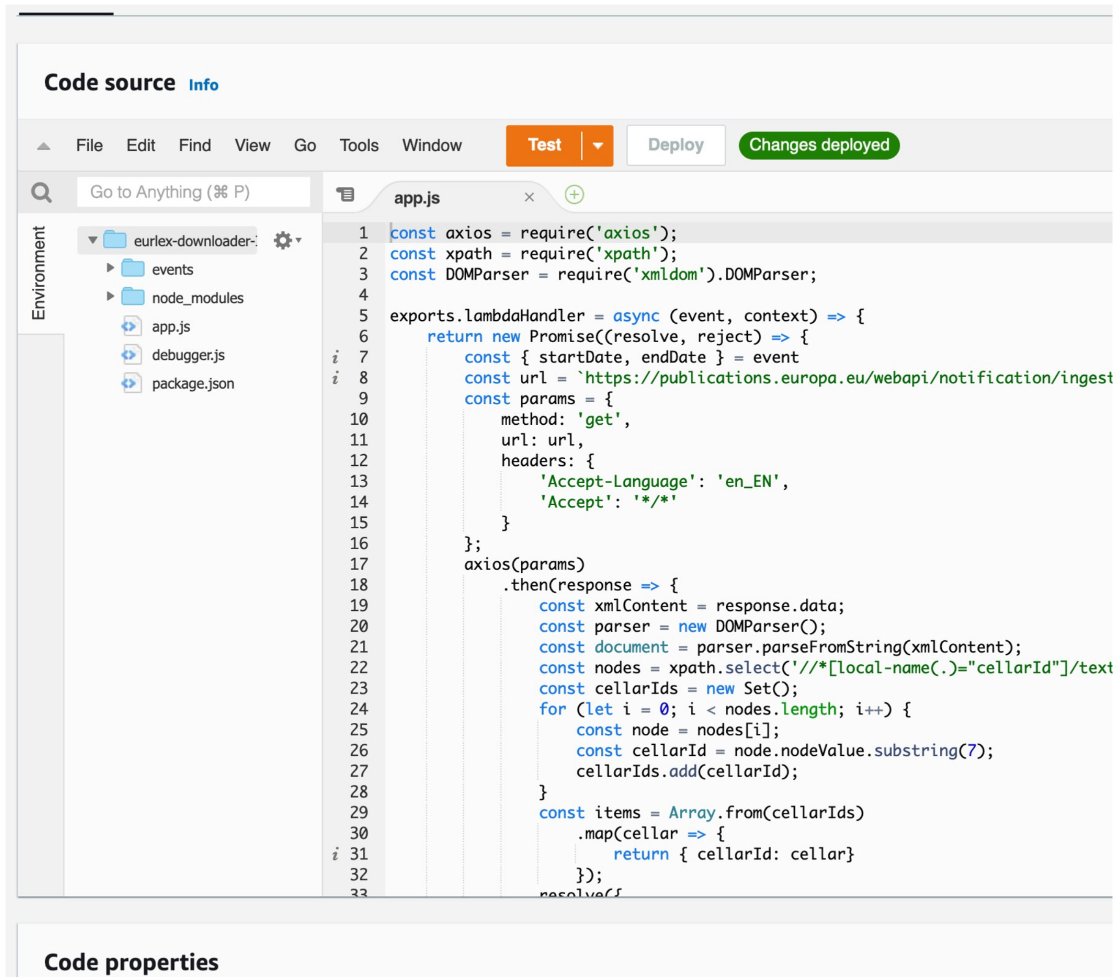
```
# Get a list of functions to invoke
$ sam local invoke

# Invoke a function
$ sam local invoke functionName

# Invoke a function and send an event
$ sam local invoke functionName -e event.json
```

### Built-in Best Practices

Deploy your infrastructure as config to leverage best practices such as code reviews. Enable gradual deployments through AWS CodeDeploy and tracing using AWS X-Ray with just a few lines of SAM config.

```
version = 0.1
[default]
[default.deploy]
[default.deploy.parameters]
stack_name = "stack-name"
s3_bucket = "bucket-name"
s3_prefix = "eurlex-downloader"
region = "us-east-1"
confirm_changeset = true
capabilities = "CAPABILITY_IAM"
```

## Integration with Development Tools



```
const axios = require('axios');
const xpath = require('xpath');
const DOMParser = require('xmldom').DOMParser;

exports.lambdaHandler = async (event, context) => {
    return new Promise((resolve, reject) => {
        const { startDate, endDate } = event
        const url = `https://publications.europa.eu/webapi/notification/ingest
        const params = {
            method: 'get',
            url: url,
            headers: {
                'Accept-Language': 'en_EN',
                'Accept': '*/*'
            }
        };
        axios(params)
            .then(response => {
                const xmlContent = response.data;
                const parser = new DOMParser();
                const document = parser.parseFromString(xmlContent);
                const nodes = xpath.select('//*[local-name(.)="cellarId"]/text
                const cellarIds = new Set();
                for (let i = 0; i < nodes.length; i++) {
                    const node = nodes[i];
                    const cellarId = node.nodeValue.substring(7);
                    cellarIds.add(cellarId);
                }
                const items = Array.from(cellarIds)
                    .map(cellar => {
                        return { cellarId: cellar}
                    });
                resolve({
```

SAM integrates with a suite of AWS serverless tools. Find new applications in the AWS Serverless Application Repository, use AWS Cloud9 IDE to author, test, and debug SAM-based serverless applications, and AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline to build a deployment pipeline. To start with a project structure, code repository, and CI/CD pipeline configured for you, try AWS CodeStar.

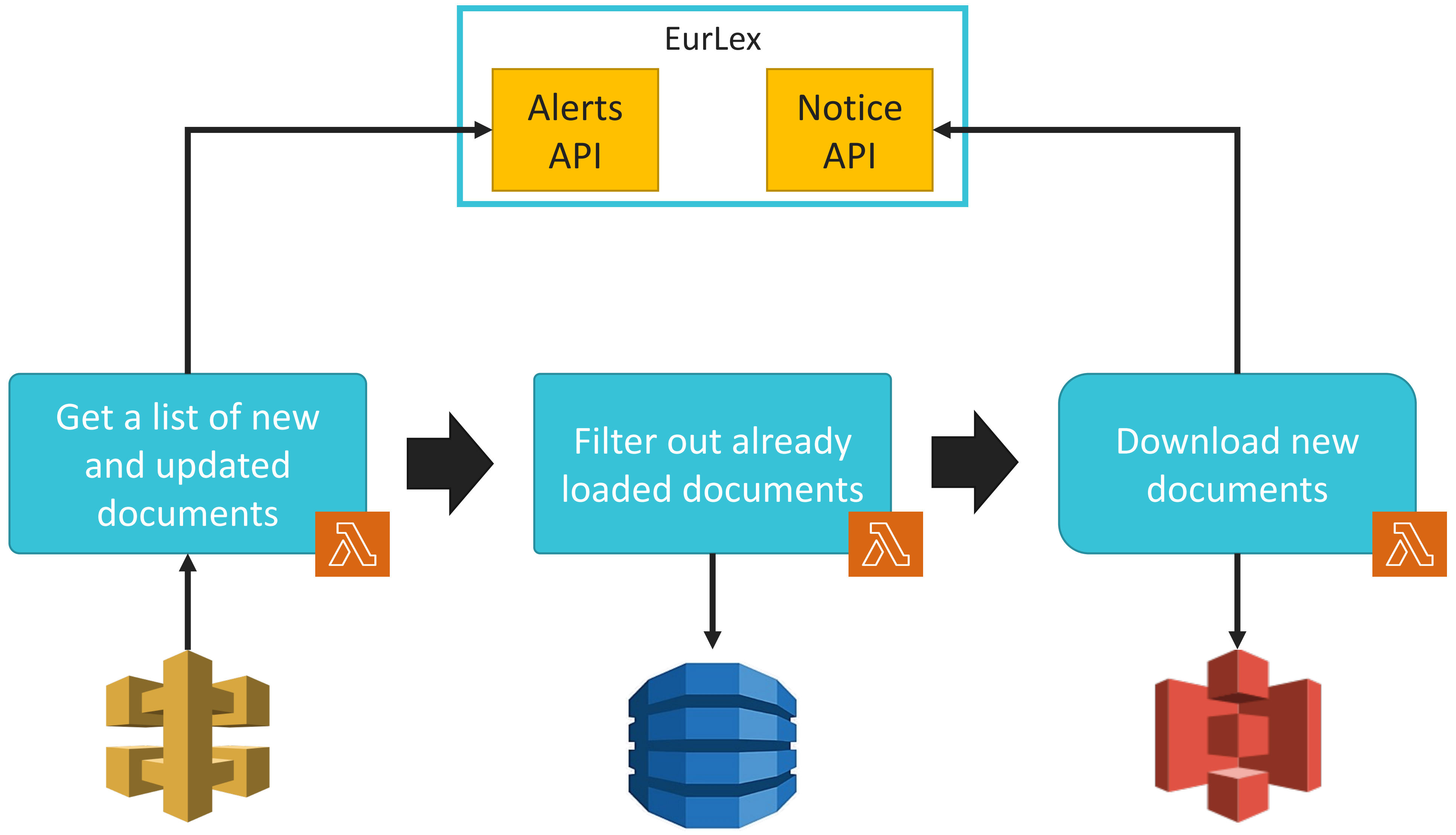
## Demo time

Create a sample app with AWS SAM

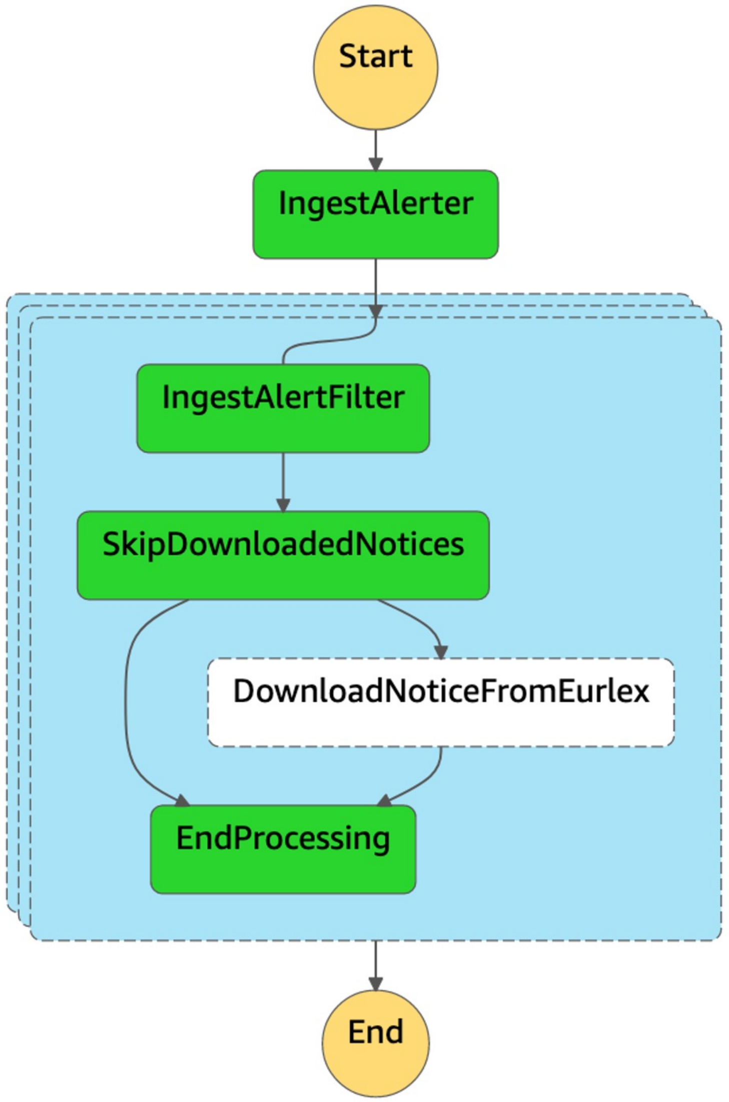
## The App

To download information from 3-rd party service and upload everything to S3 and DynamoDB with AWS Lambda and AWS Step Functions

## The App


EurLex
Alerts API
Notice API
Get a list of new and updated documents
Filter out already loaded documents
Download new documents

## AWS Step Functions



- The workflows you build with Step Functions are called **state machines**, and each step of your workflow is called a **state**.
- **Tasks** perform work, either by coordinating another AWS service or an application that you can host basically anywhere.
- **Pass states** pass their input as output to the next state. You can also delay execution when you need to using **wait states**.
- **Parallel** states begin multiple branches of execution at the same time, such as running multiple Lambda functions at once.
- **Choice states** add branching logic to your state machine and make decisions based on their input.
- When you execute your state machine, each move from one state to the next is called a **state transition**.
- You can reuse components, easily edit the sequence of steps or swap out the code called by task states as your needs change.

# Demo time

Writing code, running, debugging

## Summary

https://github.com/aabarmin/epam-sam-java-example-2022

- Serverless computing allows to focus on the business logic but not infrastructure.
- AWS SAM is built on the top of the CloudFormation but makes writing serverless apps easier.

- `$ sam init` - to start a new project
- `$ sam invoke local` – to execute the code locally
- `$ sam build --use-container` – to build inside docker containers
- `$ sam deploy --guided` – to watch the deployment process

# Thank you!

<epam>