MENTORMATE™
DevCamp

Infrastructure as Code
(for developers)

Yulia Tenincheva

Senior Cloud Engineer, MentorMate

Today’s Agenda

- Infrastructure as Code Introduction
- CloudFormation & AWS CDK
- Serverless Application Model (SAM)
- Wrap up

Infrastructure as Code:

MENTORMATE™
DevCamp

Techniques, practices, and tools from software development applied to creating reusable, maintainable, extensible, and testable infrastructure

Why is this important to you?

Structure

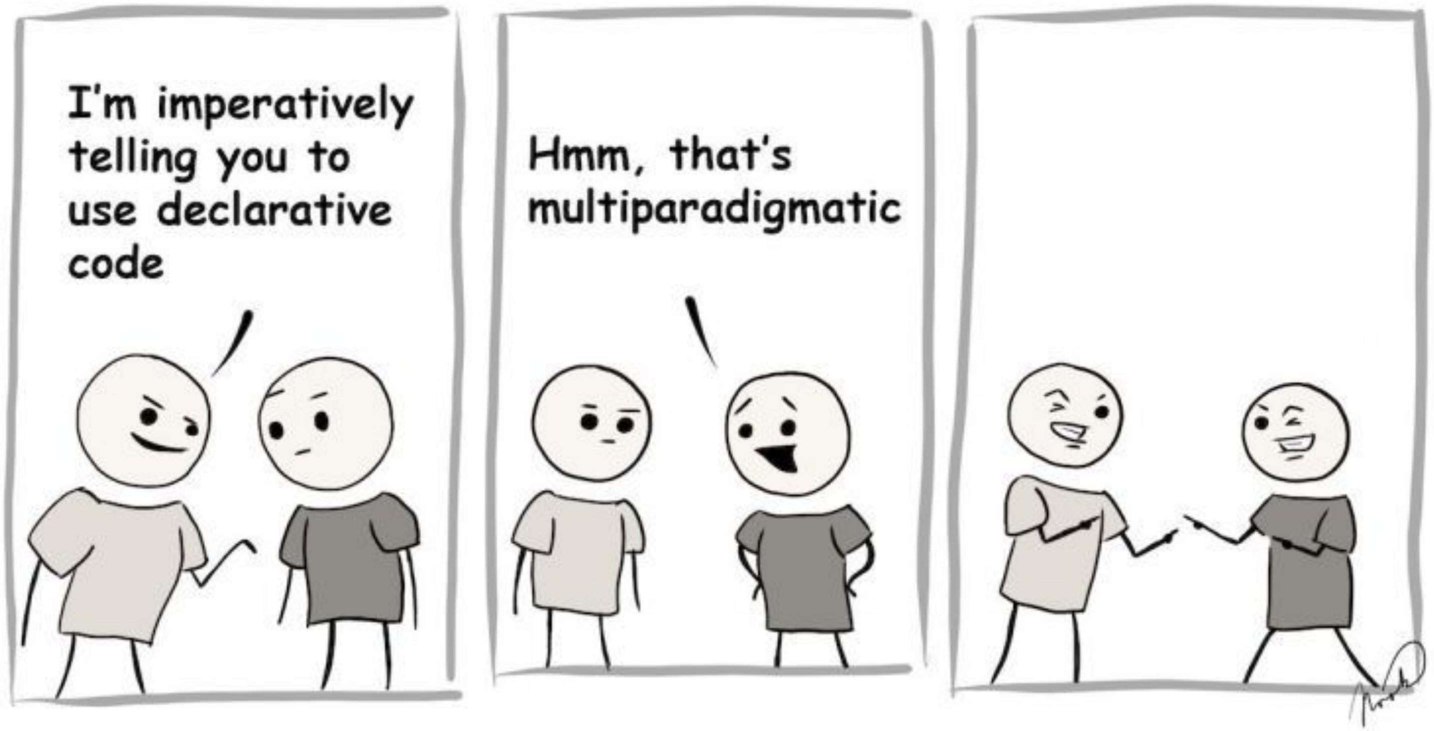
Speed

Health

Security

Approaches to IaC

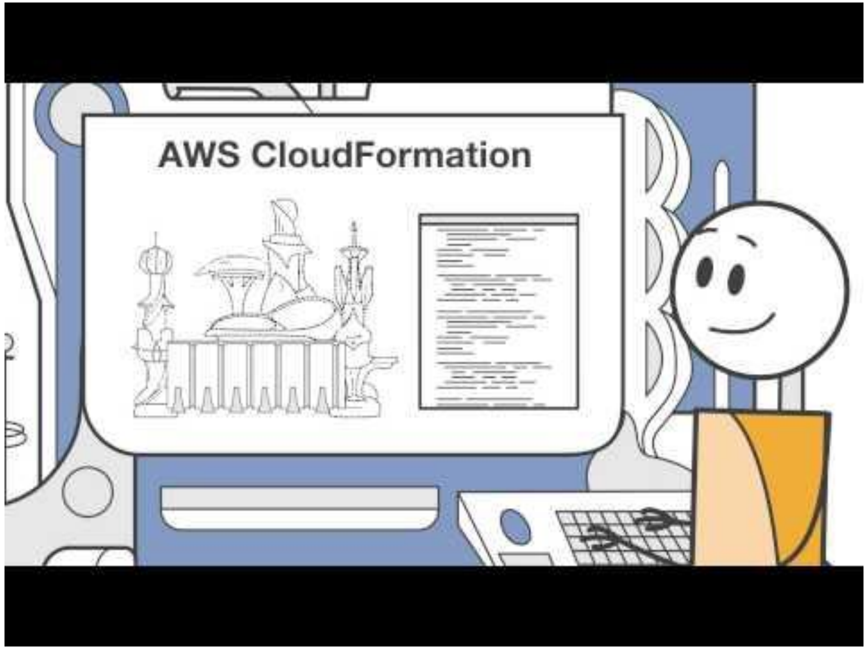
- Imperative
  - Describe *how*, step-by-step
- Declarative
  - Describe *what*
- Immutable
  - Lower risk, discrete versioning, reduced complexity


I'm imperatively telling you to use declarative code
Hmm, that's multiparadigmatic

AWS CloudFormation

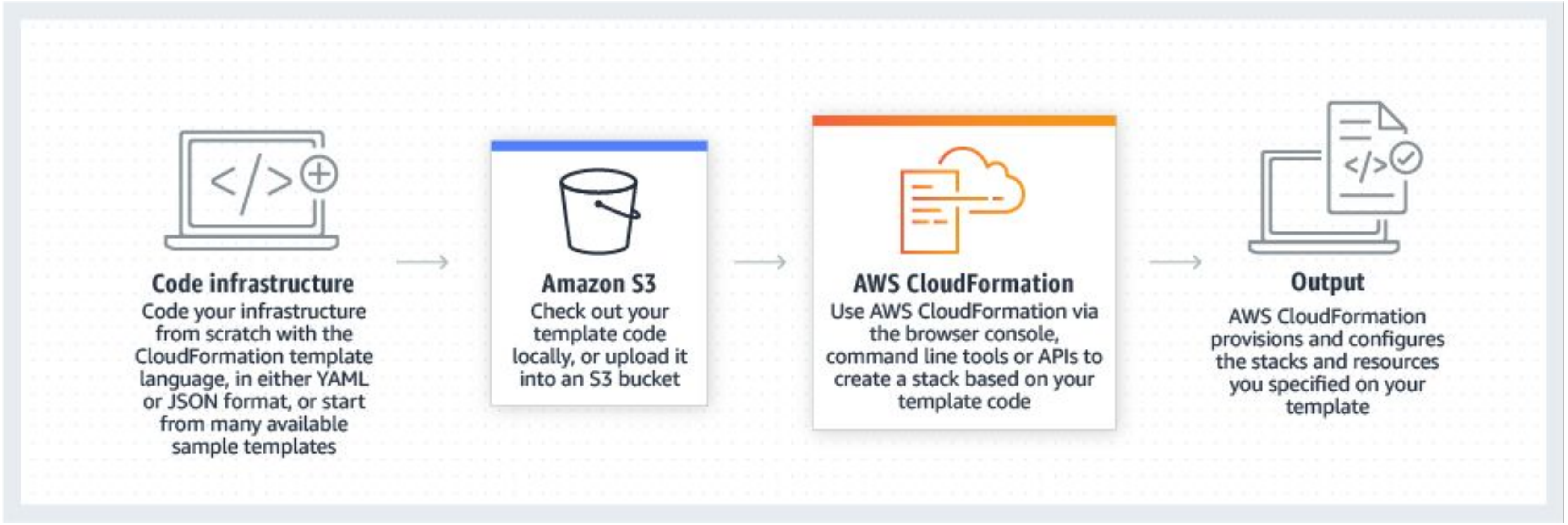
AWS CloudFormation Intro

- Template
  - Simply a JSON or YAML formatted file
- Stack
  - Collection of resources that will be built using templates
- Changeset
  - Proposed set of changes



MENTORMATE™ DevCamp

How it works



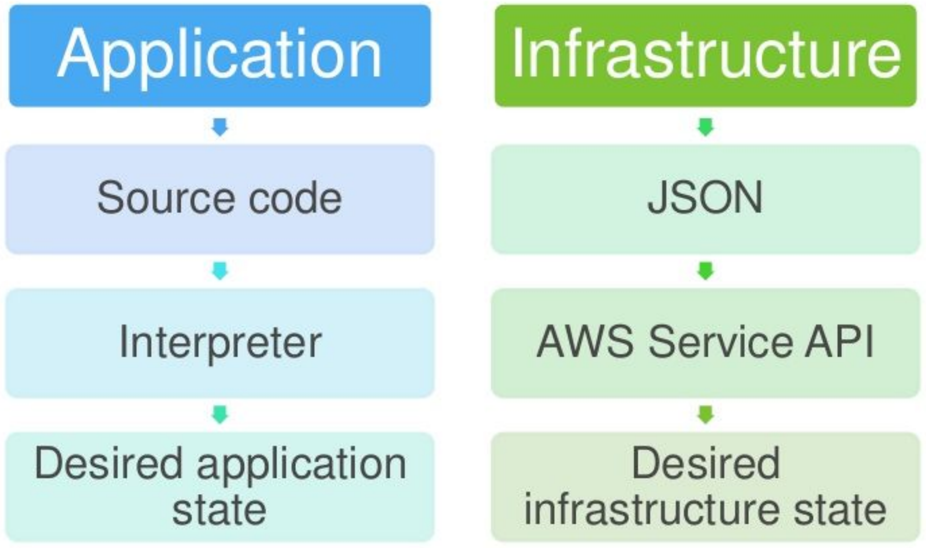
Code infrastructure
Code your infrastructure from scratch with the CloudFormation template language, in either YAML or JSON format, or start from many available sample templates

Amazon S3
Check out your template code locally, or upload it into an S3 bucket

AWS CloudFormation
Use AWS CloudFormation via the browser console, command line tools or APIs to create a stack based on your template code

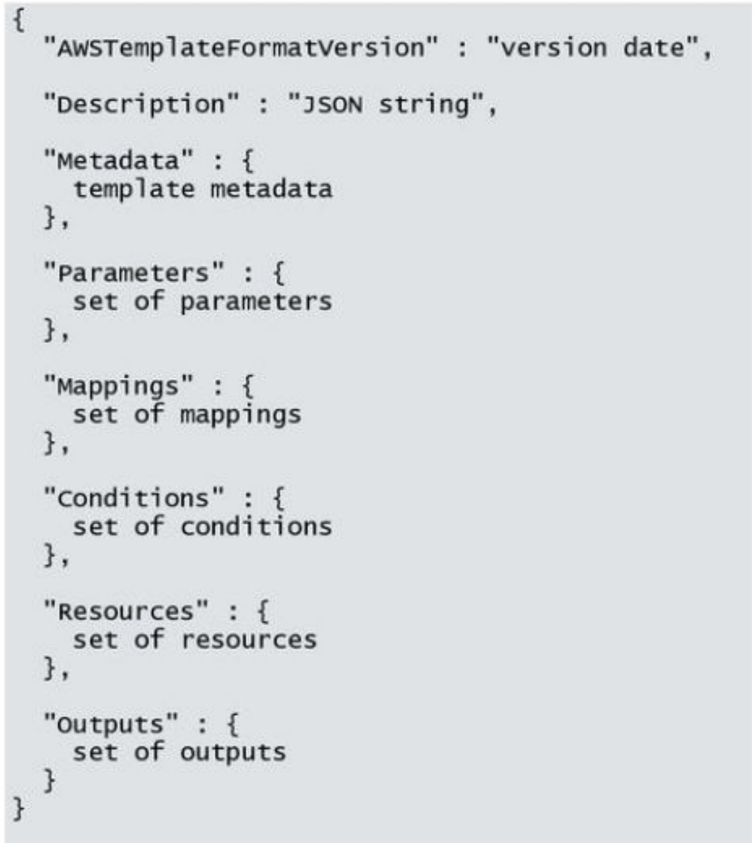
Output
AWS CloudFormation provisions and configures the stacks and resources you specified on your template

Build and Operate Infrastructure as Software



Template Anatomy

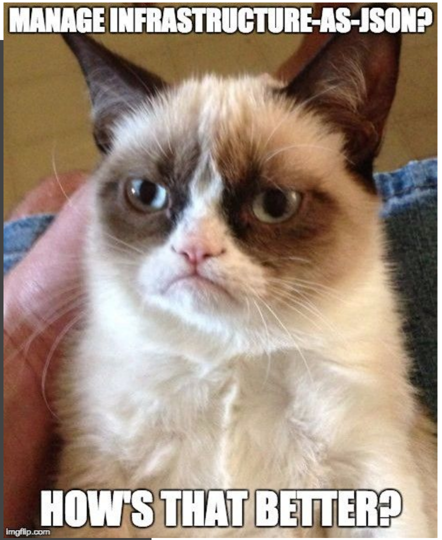
- Parameters can be passed in, the referred to with a { “Ref”: “Param_name” }
- The properties of resources can reference other resources in the same template
- Resources are declared in any order
- Working with templates - guide



```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : "JSON string",

  "Metadata" : {
    template metadata
  },

  "Parameters" : {
    set of parameters
  },

  "Mappings" : {
    set of mappings
  },

  "Conditions" : {
    set of conditions
  },

  "Resources" : {
    set of resources
  },

  "Outputs" : {
    set of outputs
  }
}
```

Example

```
{
 "Resources": {
  "S3Bucket": {
   "Type": "AWS::S3::Bucket",
   "DeletionPolicy": "Delete",
   "Properties": {
    "BucketName": "devcamp-cf-demo",
    "WebsiteConfiguration": {
     "IndexDocument": "index.html"
    },
    "BucketEncryption": {
     "ServerSideEncryptionConfiguration": [{
       "ServerSideEncryptionByDefault": {
        "SSEAlgorithm": "AES256"
       }
      }
     ]
    },
    "Tags": [{
      "Key": "Project",
      "Value": "Demo"
    }]
   }
  }
 },
 "Outputs": {
  "WebsiteURL": {
   "Value": {
    "Fn::GetAtt": [
     "S3Bucket",
     "WebsiteURL"
    ]
   },
   "Description": "URL for website hosted on S3"
  }
 }
}
```



- S3 Full Syntax documentation

MENTORMATE™ DevCamp

Basic operations

- Create CloudFormation **Stack** & upload the website

```
$ aws cloudformation create-stack --stack-name s3-demo --template-body file://cf-base-minimum.json
$ aws s3 sync . s3://devcamp-cf-demo/ --acl "public-read"
```

- Create a **Change Set** for update & Execute it

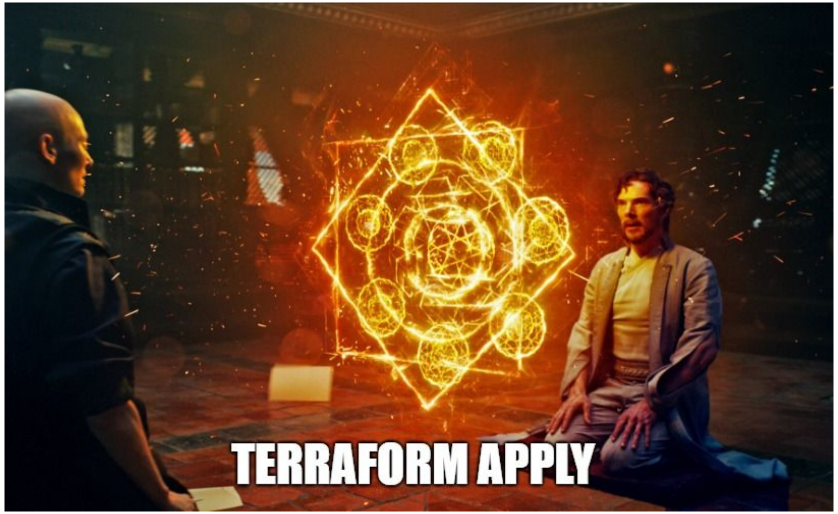
```
$ aws cloudformation create-change-set --stack-name s3-demo --change-set-name add-tag --template-body
file://cf-base-minimum.json
$ aws cloudformation execute-change-set --stack-name s3-demo --change-set-name add-tag
```

- Delete the **Stack**

```
$ aws cloudformation delete-stack --stack-name s3-demo
```

Terraform

- Alternative to CloudFormation for AWS
- **Cloud-agnostic**
- **Great community** (open source) <3
- **Easy** to plan, deploy & update
- Introduction here



AWS Cloud Development Kit

MENTORMATE™
DevCamp

AWS CDK

- Pre-requisites:

```
$ npm install aws-cdk -g
```

- Aaand.... **TypeScript**

- Example of using the CDK to deploy a React Application on AWS - video

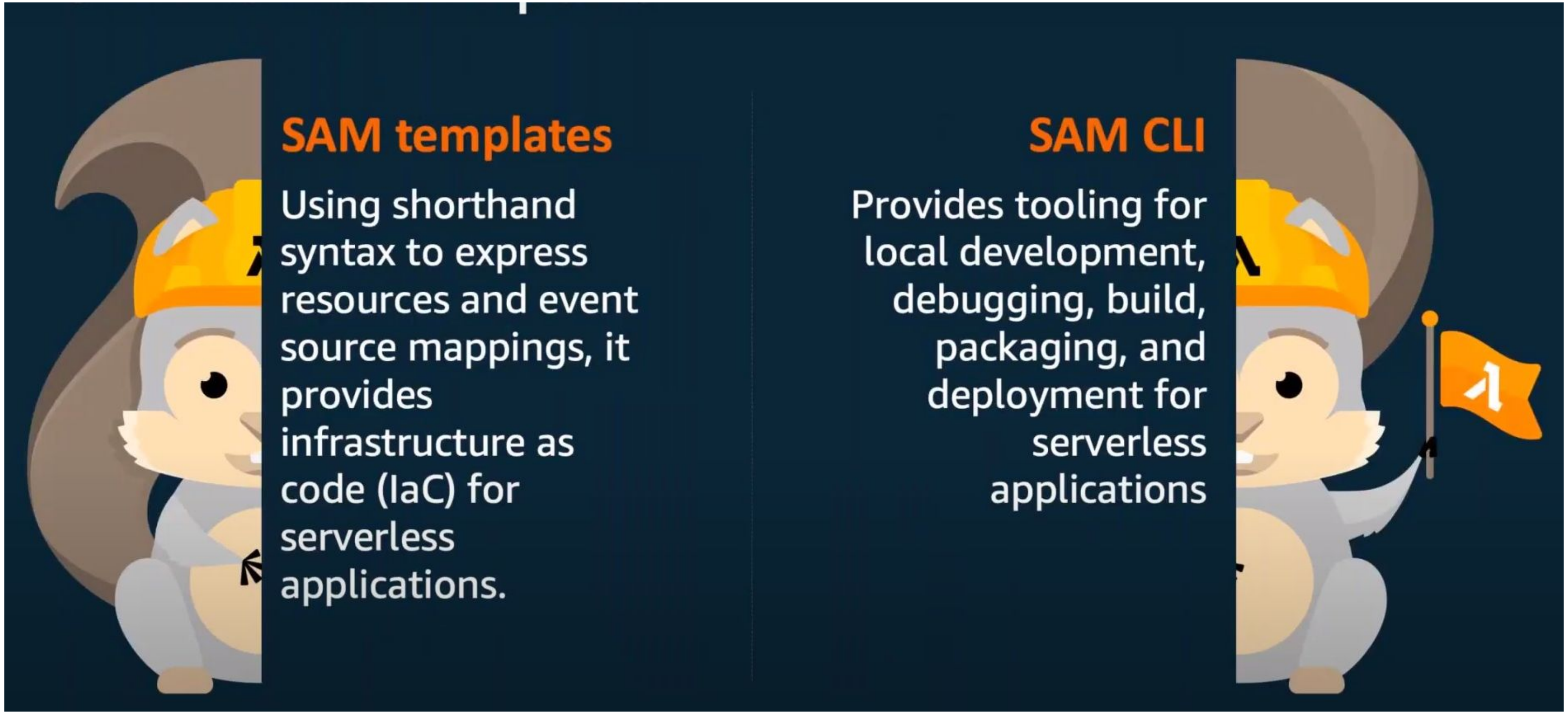
Serverless Application Model

MENTORMATE™
DevCamp

SAM

MEET SAM.
USE SAM TO BUILD TEMPLATES THAT DEFINE YOUR SERVERLESS APPLICATIONS.

SAM



SAM templates

Using shorthand syntax to express resources and event source mappings, it provides infrastructure as code (IaC) for serverless applications.

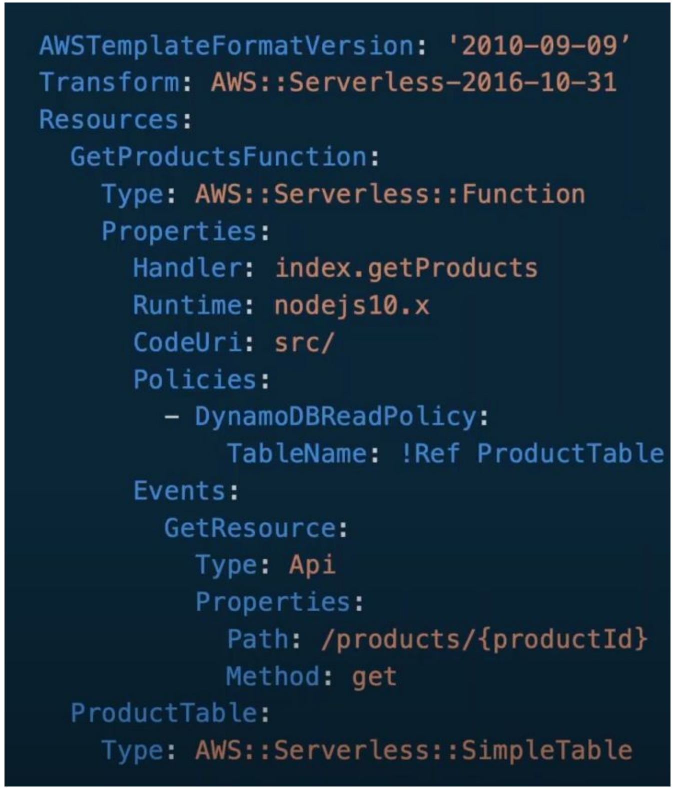
SAM CLI

Provides tooling for local development, debugging, build, packaging, and deployment for serverless applications

https://aws.amazon.com/serverless/sam

AWS SAM Templates

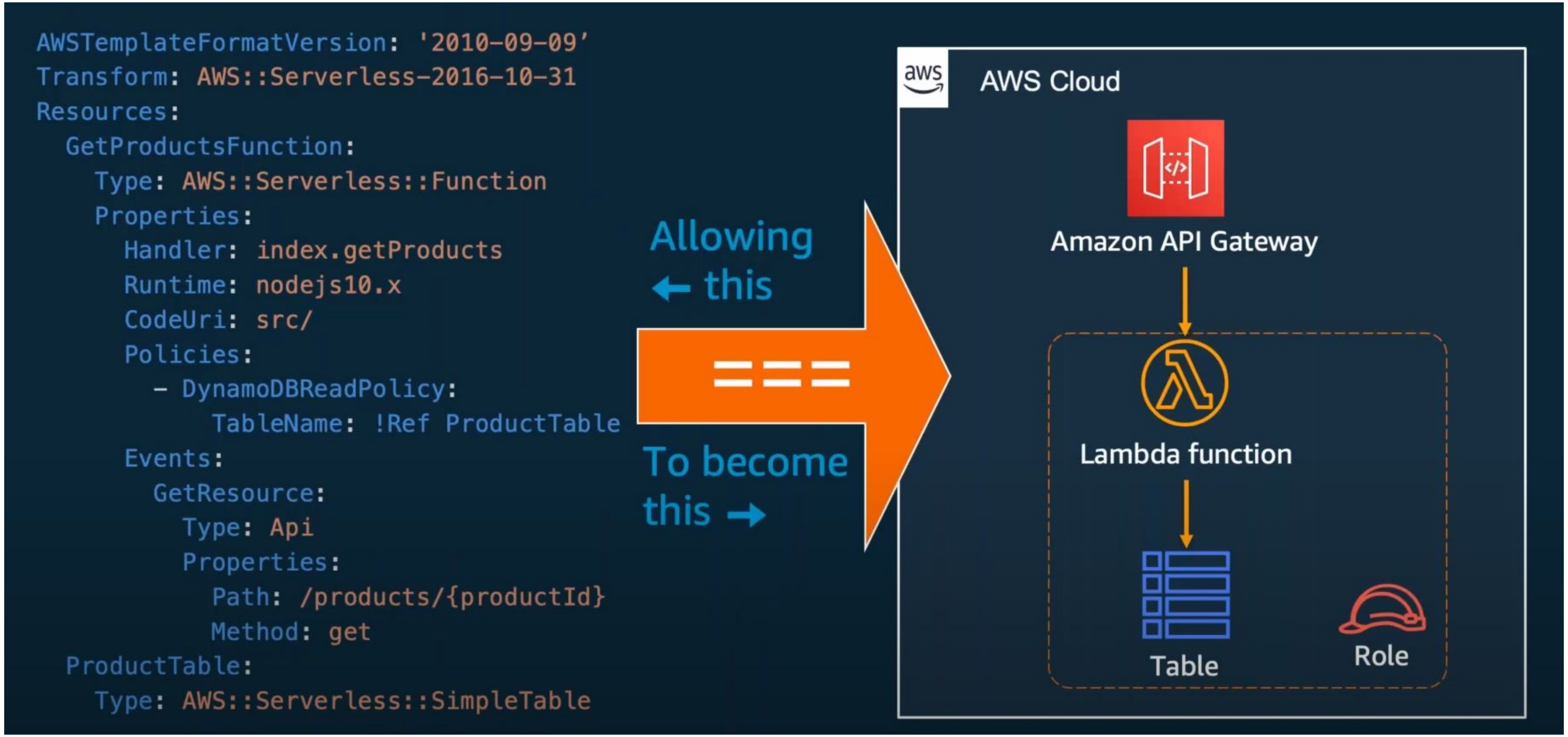
- Can mix in other non-SAM CloudFormation resources in the same template
  - I.e Amazon S3, AWS Step Functions, Amazon Kinesis
- Supports use of Parameters, Mappings, Outputs, etc...
- Supports Intrinsic Functions
  - I.e Ref, Sub, Join, Select, Split
- Can use ImportValue
  - (exceptions for RestAPIid, Policies, StageName attributes
- YAML or JSON



```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetProductsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.getProducts
      Runtime: nodejs10.x
      CodeUri: src/
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref ProductTable
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /products/{productId}
            Method: get
  ProductTable:
    Type: AWS::Serverless::SimpleTable
```

AWS SAM Deep Dive - Video

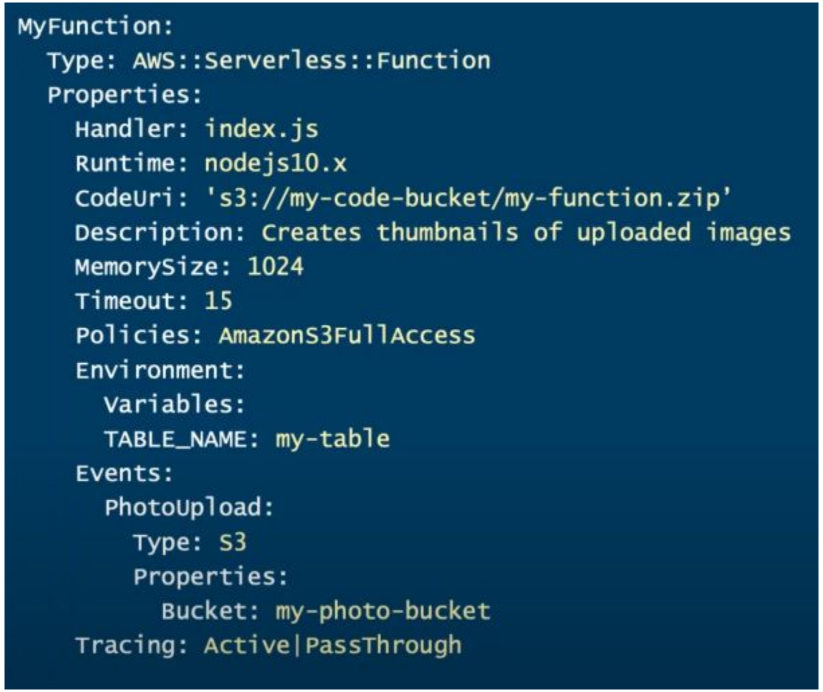
AWS SAM Templates



```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetProductsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.getProducts
      Runtime: nodejs10.x
      CodeUri: src/
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref ProductTable
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /products/{productId}
            Method: get
  ProductTable:
    Type: AWS::Serverless::SimpleTable
```

SAM Serverless resources

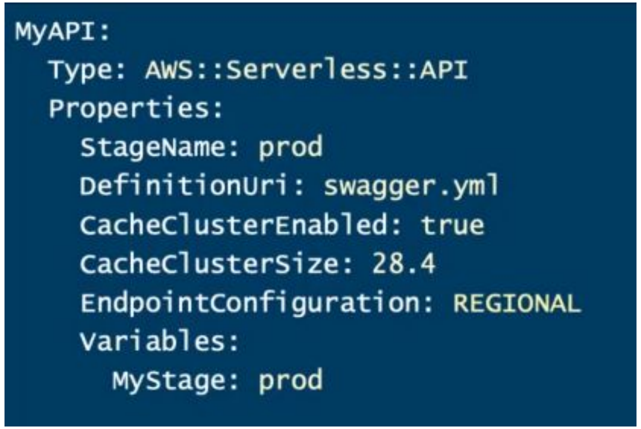
- **AWS::Serverless::Function**
- AWS::Serverless::Api
- AWS::Serverless::SimpleTable
- AWS::Serverless::LayerVersion
- AWS::Serverless:: Application



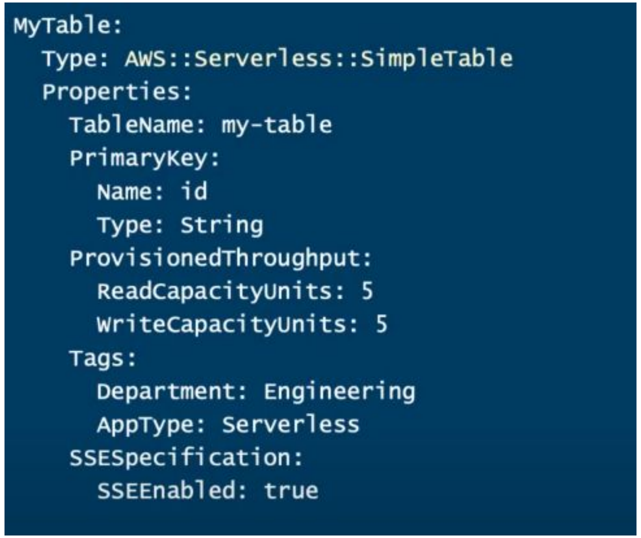
```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.js
    Runtime: nodejs10.x
    CodeUri: 's3://my-code-bucket/my-function.zip'
    Description: Creates thumbnails of uploaded images
    MemorySize: 1024
    Timeout: 15
    Policies: AmazonS3FullAccess
    Environment:
      Variables:
      TABLE_NAME: my-table
    Events:
      PhotoUpload:
        Type: S3
        Properties:
          Bucket: my-photo-bucket
    Tracing: Active|PassThrough
```

SAM Serverless resources

- AWS::Serverless::Function
- **AWS::Serverless::[Api]**
- **AWS::Serverless::[SimpleTable]**
- AWS::Serverless::LayerVersion
- AWS::Serverless:: Application



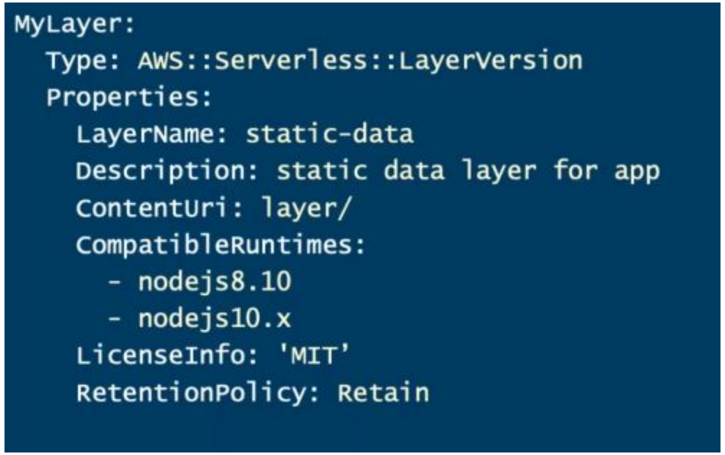
```
MyAPI:
  Type: AWS::Serverless::API
  Properties:
    StageName: prod
    DefinitionUri: swagger.yml
    CacheClusterEnabled: true
    CacheClusterSize: 28.4
    EndpointConfiguration: REGIONAL
    Variables:
      MyStage: prod
```



```
MyTable:
  Type: AWS::Serverless::SimpleTable
  Properties:
    TableName: my-table
    PrimaryKey:
      Name: id
      Type: String
    ProvisionedThroughput:
      ReadCapacityUnits: 5
      WriteCapacityUnits: 5
    Tags:
      Department: Engineering
      AppType: Serverless
    SSESpecification:
      SSEEnabled: true
```

SAM Serverless resources

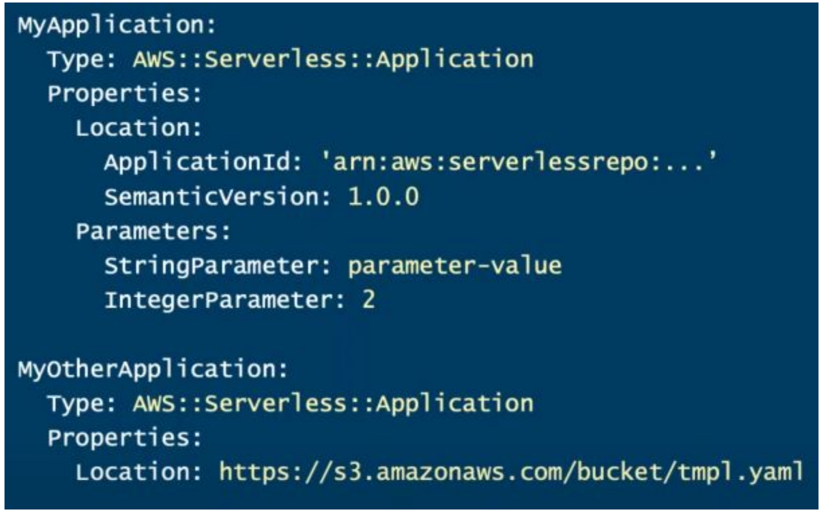
- AWS::Serverless::Function
- AWS::Serverless::Api
- AWS::Serverless::SimpleTable
- **AWS::Serverless::LayerVersion**
- AWS::Serverless:: Application



```
MyLayer:
  Type: AWS::Serverless::LayerVersion
  Properties:
    LayerName: static-data
    Description: static data layer for app
    ContentUri: layer/
    CompatibleRuntimes:
      - nodejs8.10
      - nodejs10.x
    LicenseInfo: 'MIT'
    RetentionPolicy: Retain
```

SAM Serverless resources

- AWS::Serverless::Function
- AWS::Serverless::Api
- AWS::Serverless::SimpleTable
- AWS::Serverless::LayerVersion
- **AWS::Serverless::** **Application**

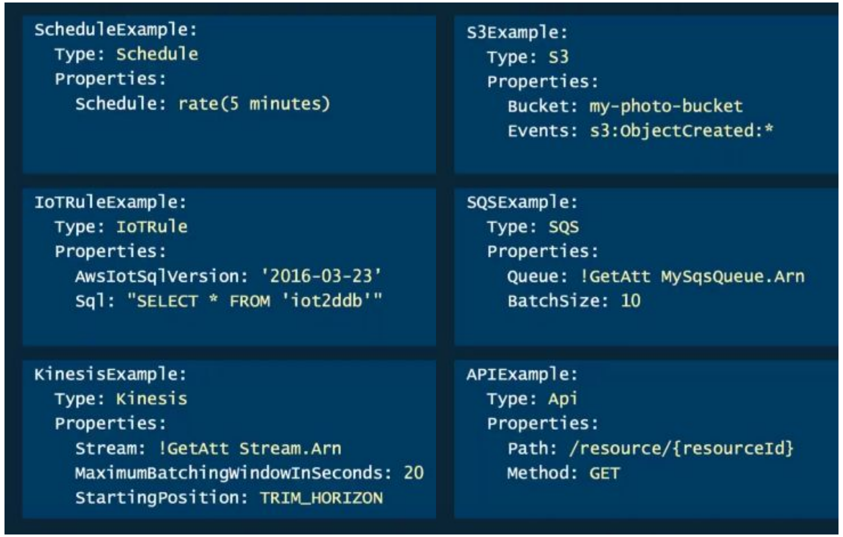


```yaml
MyApplication:
  Type: AWS::Serverless::Application
  Properties:
    Location:
      ApplicationId: 'arn:aws:serverlessrepo:...'
      SemanticVersion: 1.0.0
    Parameters:
      StringParameter: parameter-value
      IntegerParameter: 2

MyOtherApplication:
  Type: AWS::Serverless::Application
  Properties:
    Location: https://s3.amazonaws.com/bucket/tmpl.yaml
```

MENTORMATE™ DevCamp

Lambda function event sources

- **Supported event sources:**
  - S3, SNS, Kinesis, DynamoDB, SQS, Api (Gateway), Schedule, CloudWatch Event, CloudWatch Logs, IoTRule, Alexa Skill, Cognito, etc...
- **Other services that integrate with AWS Lambda:**
  - Elastic Load Balancing, Amazon Lex, Amazon CloudFront, AWS Simple Email Service, AWS CloudFormation, AWS CodeCommit, AWS Config...



```
ScheduleExample:
  Type: Schedule
  Properties:
    Schedule: rate(5 minutes)
```

```
S3Example:
  Type: S3
  Properties:
    Bucket: my-photo-bucket
    Events: s3:ObjectCreated:*
```

```
IoTRuleExample:
  Type: IoTRule
  Properties:
    AwsIotSqlVersion: '2016-03-23'
    Sql: "SELECT * FROM 'iot2ddb'"
```

```
SQSExample:
  Type: SQS
  Properties:
    Queue: !GetAtt MySqsQueue.Arn
    BatchSize: 10
```

```
KinesisExample:
  Type: Kinesis
  Properties:
    Stream: !GetAtt Stream.Arn
    MaximumBatchingWindowInSeconds: 20
    StartingPosition: TRIM_HORIZON
```

```
APIExample:
  Type: Api
  Properties:
    Path: /resource/{resourceId}
    Method: GET
```

MENTORMATE™ DevCamp

SAM CLI

```
#Step 1 - Download a sample application
$ sam init

#Step 2 - Build your application
$ cd sam-app
$ sam build

#Step 3 - Deploy your application
$ sam deploy --guided
```

SAM CLI - Local Development

```
#Invoke a single Lambda function
$ sam local invoke --help
$ sam local start-lambda

#Run API Gateway locally
$ sam local start-api

#Generate a sample event for testing
$ sam local generate-event --help
$ sam local generate-event [SERVICE] --help

#Generate a sample event for testing
$ sam local generate-event --help
```

Homework

Homework

Refactor your homework for DynamoDB and **create/deploy a serverless application as code**, using **SAM**. Your application should have the following components:

- **API Gateway** with **CRUD** for hobbies/activities.
- **AWS Lambda** functions, triggered by API Gateway
- When a new record is added to DynamoDB (**event**), trigger an AWS Lambda to export a JSON report to S3 with aggregated data.
- Create an **S3 bucket** with **static website hosting** with very simple front-end: just read the JSON file from S3 and visualize it in some form - list, table, whatever.

THANKS TO THE CLOUD ALL MY DEVICES ARE IN SYNC... IT'S ONLY MY BRAIN THAT DOESN'T KNOW WHAT'S GOING ON.

Everything should be created and deployed with **code**!
Destroy your AWS resources when you are ready.

MENTORMATE DevCamp