Infrastructure as Code

In this lesson, you will learn about the Cloudformation template and the CloudFormation stack. Additionally, you'll get a detailed look at the `template.yaml` of the project you initialized.

WE'LL COVER THE FOLLOWING

The CloudFormation template

For deploying applications, SAM uses **CloudFormation**, an AWS service for managing infrastructure as code. This means that CloudFormation converts a source file describing an application infrastructure (called `template`) into a set of running, configured cloud resources (called `stack`).

Instead of individually configuring different resources such as *file storage*, *databases*, and *queues* with CloudFormation, you just need to declare the required resources in a textual file. The `template.yaml` file in your project directory is a CloudFormation template.

The CloudFormation stack

You can use CloudFormation to create a whole stack of resources from the template in a single command. It is also smart enough to detect the differences between a *template* and a *deployed stack*, making it easy to update infrastructure resources in the future.

Benefits of CloudFormation

- You can modify the template file and CloudFormation will reconfigure or delete only the resources that actually need to change.
- If a resource update fails for whatever reason, CloudFormation will reset all the other resources to the previous configuration, managing a whole set of infrastructural components as a *single* unit. This makes it easy and safe for a whole team of developers to *add*, *remove*, or *reconfigure* the infrastructural services supporting an application.
- It also becomes trivially simple to know which version of infrastructure is compatible with which version of code. This supports *infrastructure traceability* and *reproducible installations*. (It's not a coincidence that the template file is in the same directory as the function source; they should be committed to a version control system together).
- Another benefit of CloudFormation is that you can share templates with other teams, or even publish them online so that others can set up your application with a single click.

> ### CloudFormation Template Formats
>
> CloudFormation supports **JSON** and **YAML** template formats. In this course, you'll use *YAML* because it is easier to read in print. One downside of YAML is that whitespace is important and getting the right indentation might be a bit fiddly. If you want more control over the structure of your templates, feel free to use JSON instead. YAML is actually a superset of JSON, so you can also embed JSON into YAML for sections where you want to make structure clear and avoid problems with whitespace.

YAML CloudFormation

The following file is the `template.yaml` from the project you created in the previous lesson. Let's discuss it in detail.

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

```
Description: >
  app


  Sample SAM Template for app

# More info about Globals: https://github.com/awslabs/serverless-application-model/blob/maste
Globals:
  Function:
    Timeout: 3

Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource: https://github.com/a
    Properties:
      CodeUri: hello-world/
      Handler: app.lambdaHandler
      Runtime: nodejs12.x
      Events:
        HelloWorld:
          Type: Api # More info about API Event Source: https://github.com/awslabs/serverless
          Properties:
            Path: /hello
            Method: get

Outputs:
  # ServerlessRestApi is an implicit API created out of Events key under Serverless::Functior
  # Find out more about other implicit resources you can reference within SAM
  # https://github.com/awslabs/serverless-application-model/blob/master/docs/internals/genera
  HelloWorldApi:
    Description: "API Gateway endpoint URL for Prod stage for Hello World function"
    Value: !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/Prod/h
  HelloWorldFunction:
    Description: "Hello World Lambda Function ARN"
    Value: !GetAtt HelloWorldFunction.Arn
  HelloWorldFunctionIamRole:
    Description: "Implicit IAM Role created for Hello World function"
    Value: !GetAtt HelloWorldFunctionRole.Arn
```

code/app/template.yaml

Check out the `template.yaml` file above and you'll see the infrastructural description for a basic web service. (The comments in the following code has been removed for simplicity.)

The first line tells CloudFormation which syntax version to use:

```
AWSTemplateFormatVersion: '2010-09-09'
```

This is an optional setting, so you can omit it in your templates, but it is good practice to explicitly specify the syntax version in case it changes in the future.

> **Note:** The current version is `2010-09-09`, which means it hasn't really

> **Note:** The current version is `2010-09-09`, which means it hasn't really changed since September 2010. If you don't specify the version, though, CloudFormation will use the latest available.

The CloudFormation transform property

The second line in the example file tells CloudFormation to *transform* a template before executing it:

```
Transform: AWS::Serverless-2016-10-31
```

Generally speaking, with CloudFormation this is an optional setting. The part of SAM that runs in AWS data centers actually works as a CloudFormation transformation, so pretty much all your SAM applications will list a transformation at this point.

In this case and for the rest of the course, you'll use the `AWS::Serverless-2016-10-31` transformation. Don't be distracted by the date in the transform name; SAM gets updated frequently and this is just a syntax version label. The transform setting activates SAM features and resources, allowing you to use compact descriptions for many building blocks commonly used in serverless applications. For example, this transformation allows you to use the `AWS::Serverless::Function` resource in CloudFormation as a replacement for `AWS::Lambda::Function`. It provides sensible defaults for security roles and logging, and makes it easy to connect Lambda to other AWS services. Similarly, you can use `AWS::Serverless::Api` to set up API Gateway resources more easily than with `AWS::ApiGateway::RestApi`, and `AWS::Serverless::SimpleTable` to configure DynamoDB database tables easily.

> **Using serverless resources without SAM command line tools**
>
> In order to use SAM CloudFormation transformation, you don't need any software locally installed. You can just add the `Transform` header into your CloudFormation templates and deploy it using standard CloudFormation tools. The transformation physically executes inside AWS CloudFormation, as part of the normal deployment pipeline, not on development machines or a continuous integration server uploading the template to AWS.

Get ready to see more about `template.yaml` in the next lesson!