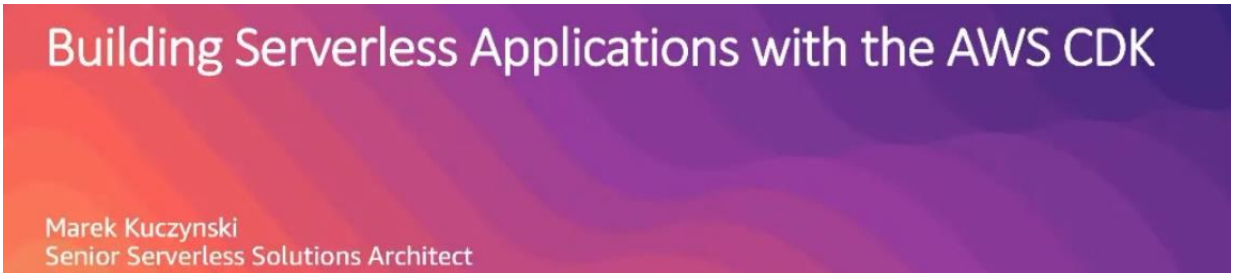
Building serverless applications with the AWS CDK - Marek Kuczynski

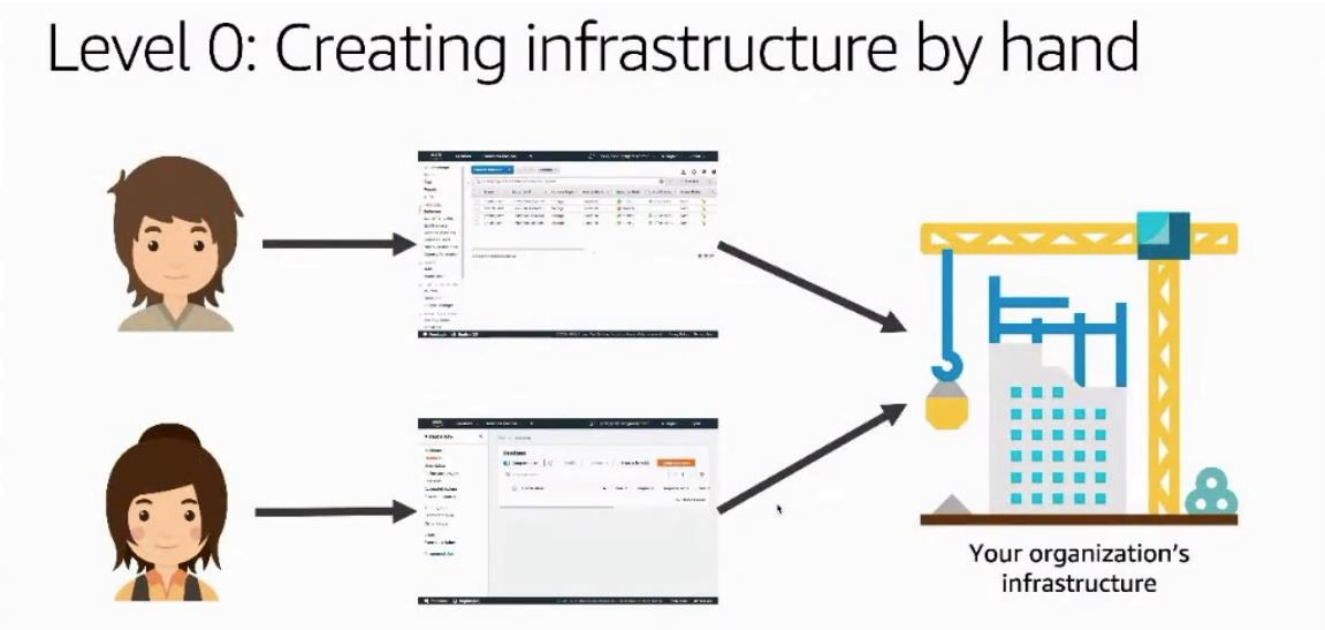
AWS User Groups | The Netherlands
441 subscribers

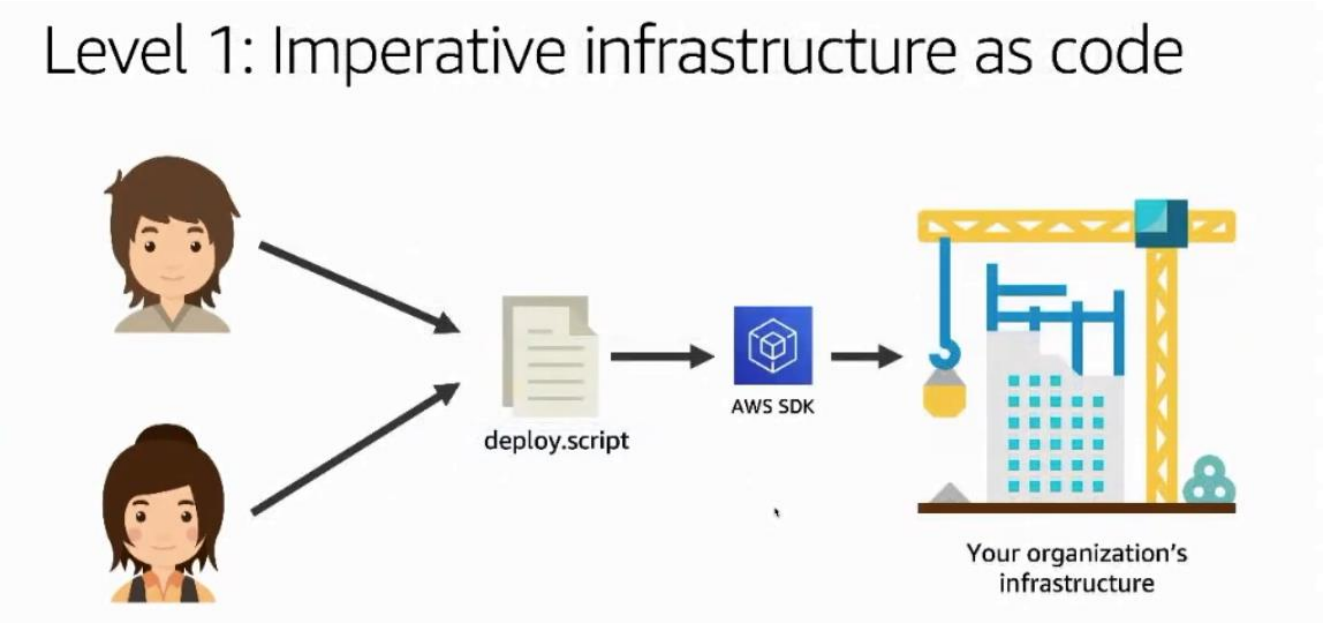
Subscribe

6

222 views Oct 31, 2020
Building serverless applications with the AWS CDK
Marek Kuczynski







Level 1: Imperative infrastructure as code

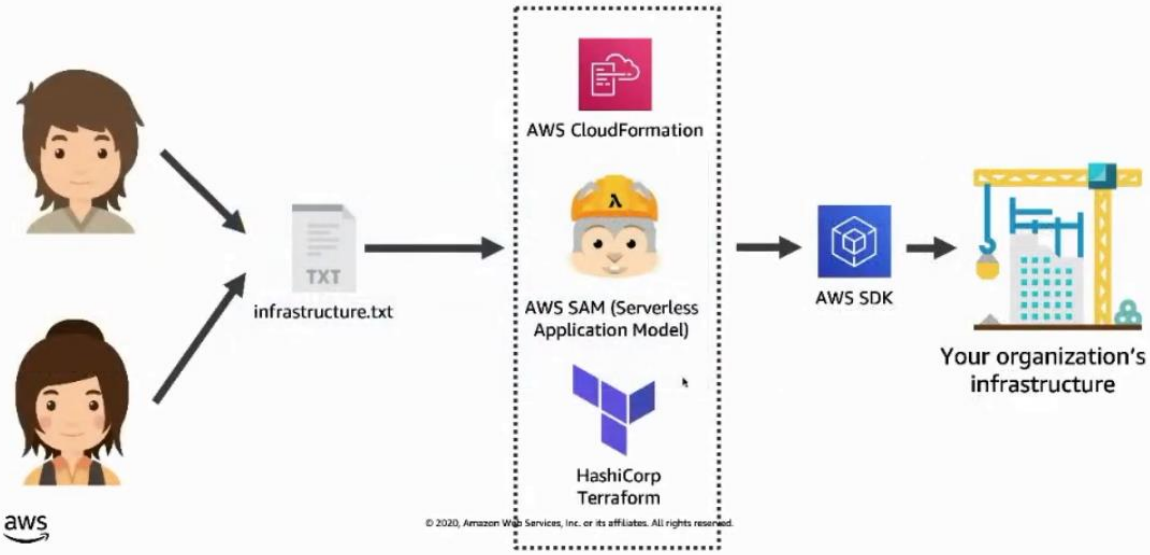
deploy.script

```
resource = getResource(xyz)

if (resource == desiredResource) {
  return
} else if (!resource) {
  createResource(desiredResource)
} else {
  updateResource(desiredResource)
}
```

- Lots of boilerplate
- What if something fails and we need to retry?
- What if two people try to run the script at once?
- Race conditions?

Level 2: Declarative infrastructure as code



Level 2: Declarative stack using CloudFormation

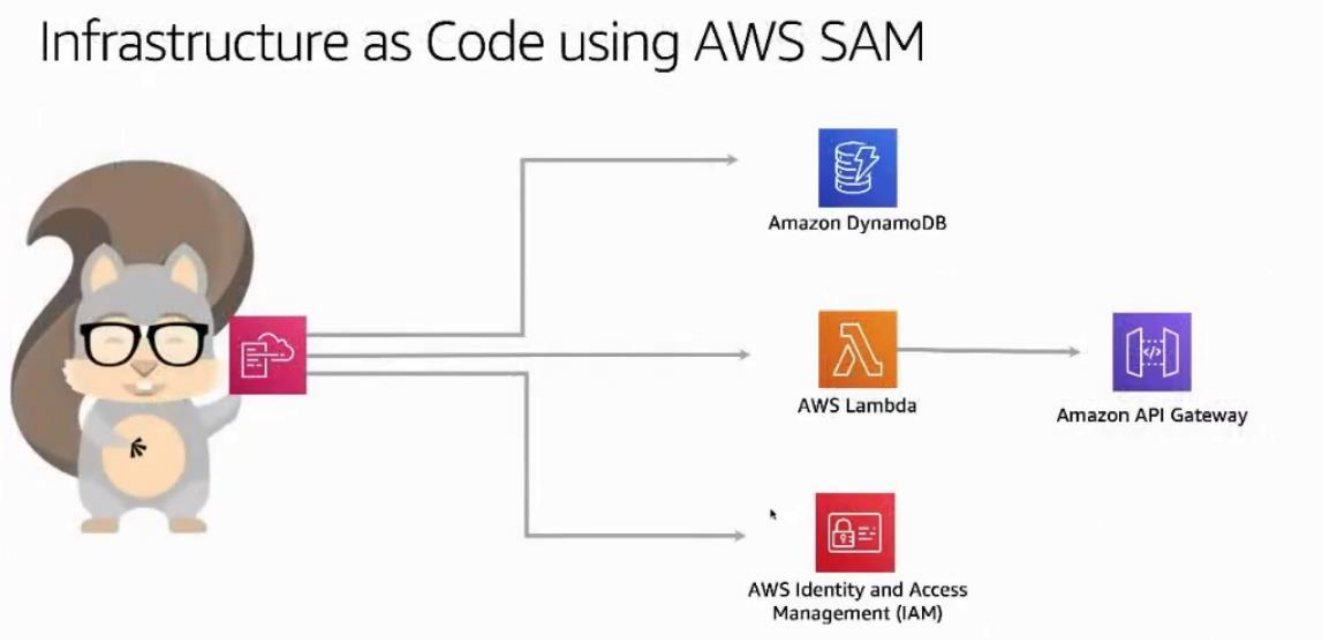
```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetHtmlFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./lambda
      Handler: index.gethtml
      Runtime: nodejs12.x
      Policies: AmazonDynamoDBReadOnlyAccess
      Events:
        GetHtml:
          Type: Api
          Properties:
            Path: /{proxy+}
            Method: ANY

  ListTable:
    Type: AWS::Serverless::SimpleTable
```

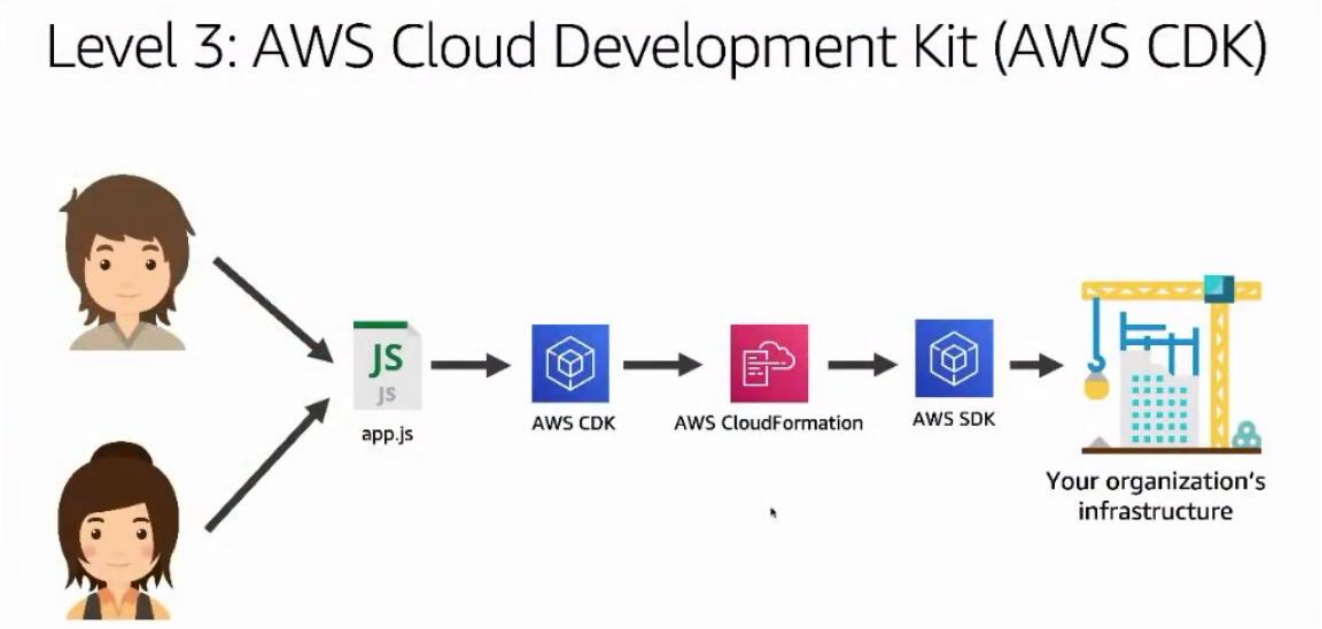
Tells CloudFormation this is a SAM template it needs to "transform"

Creates a Lambda function with the referenced managed IAM policy, runtime, code at the referenced zip location, and handler as defined. Also creates an API Gateway and takes care of all mapping/permissions necessary

Creates a DynamoDB table with 5 Read & Write units



AWS Cloud Development Kit



Level 3: AWS CDK

cdk_app.js

```
import * as lambda from '@aws-cdk/aws-lambda';
import * as path from 'path';

const fn = new lambda.Function(this, 'MyFunction', {
  runtime: lambda.Runtime.PYTHON_3_8,
  handler: 'index.handler',
  code: lambda.Code.fromAsset(path.join(__dirname, 'lambda-handler')),
});
```

lambda_function.py

lambda_function ×

```
import json, boto3

def lambda_handler(event, context):
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from The Hague!')
    }
```

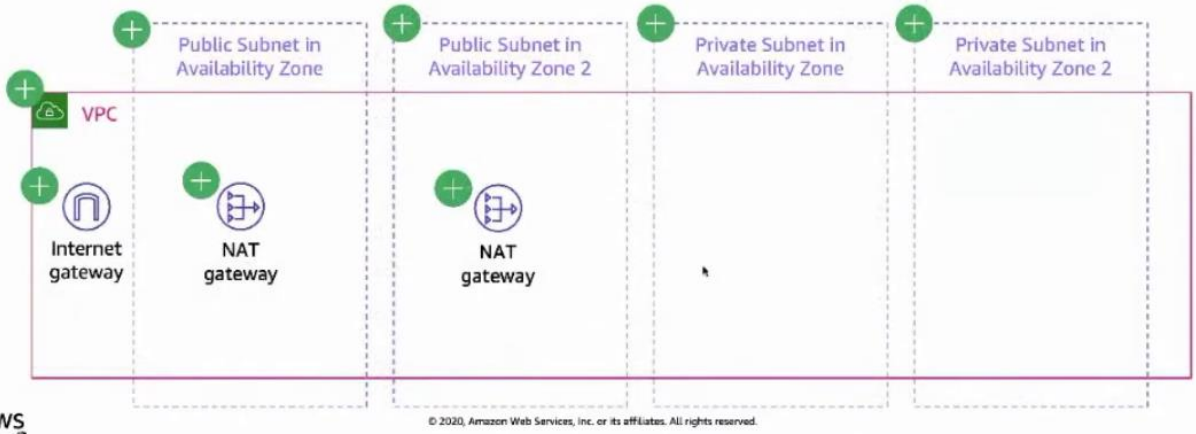
- Write in a familiar programming language, no need to learn a new language
- Create many underlying AWS resources at once with a single construct
- Each stack is made up of "constructs," which are simple classes in the code
- Still declarative, no need to handle create vs update

aws

One CDK construct expands to many underlying resources

```
// Network for all the resources
const vpc = new ec2.Vpc(stack, 'MyVpc', { maxAzs: 2 });
```

```
cdk deploy
```

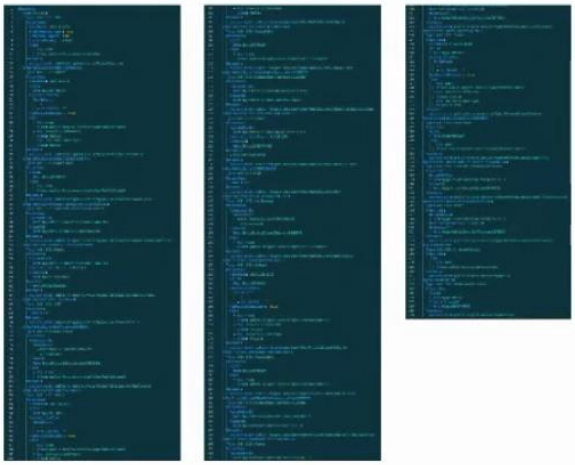


aws

One CDK construct expands to many underlying resources

```
// Network for all the resources
const vpc = new ec2.Vpc(stack, 'MyVpc', { maxAzs: 2 });
```

cdk synth



270 lines of AWS CloudFormation YAML I don't have to write!

CDK constructs are available in multiple languages

AWS CDK 1.70.0 API Reference

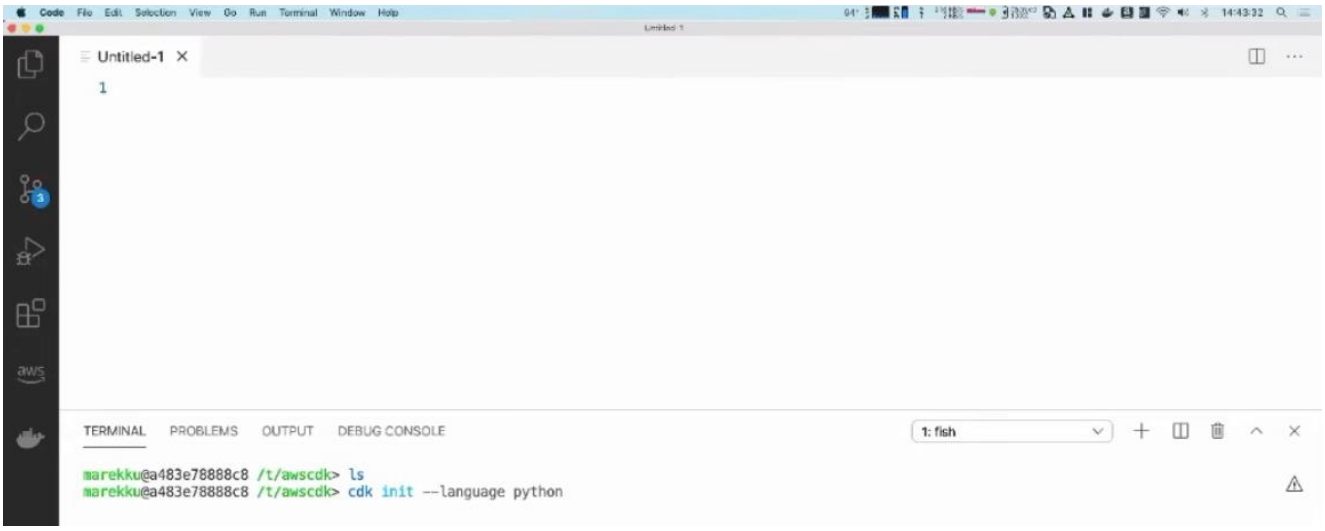
- API Reference
- alexa-ask[1]
- app-delivery
- assets
- aws-accessanalyzer[1]
- aws-acmpca[1]
- aws-amazonmq[1]

aws-lambda module

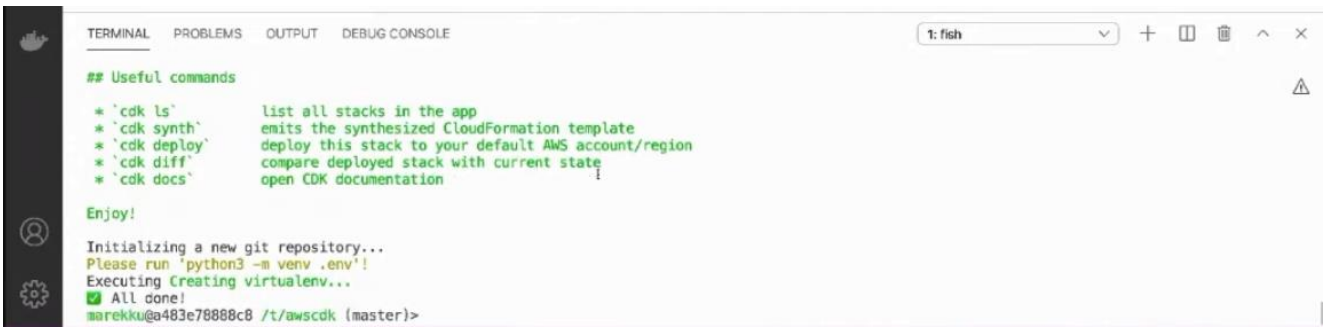
Language	Package
Python	`aws_cdk.aws_lambda`
Java	`software.amazon.awscdk.services.lambda`
.NET	`Amazon.CDK.AWS.Lambda`
TypeScript	`@aws-cdk/aws-lambda`

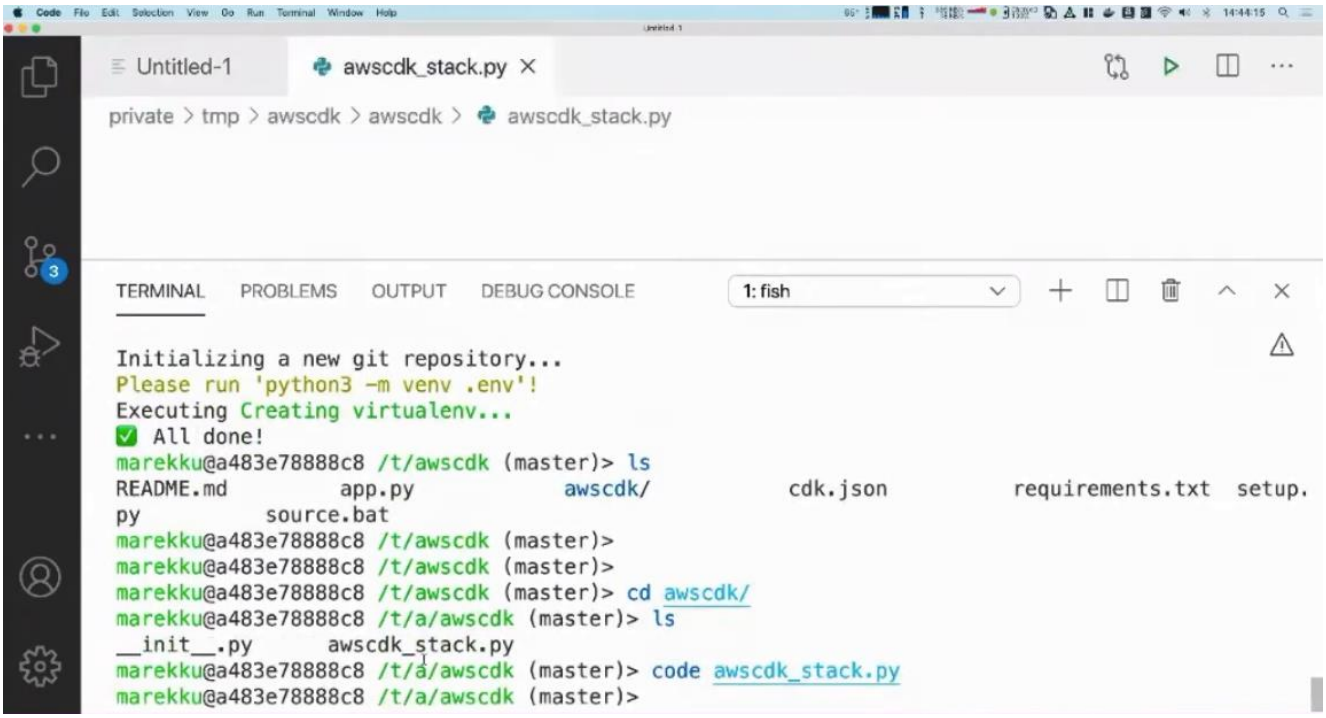
https://docs.aws.amazon.com/cdk/api/latest/docs/aws-lambda-readme.html

A short CDK demo



Run the **cdk init - -language <language>** command to create the bootstrap template and all the necessary resources that we will be using.





```
from aws_cdk import core


class AwscdkStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)

        # The code that defines your stack goes here

```

```
__init__.py      awscdk_stack.py
marekku@a483e78888c8 /t/a/awscdk (master)> code awscdk_stack.py
marekku@a483e78888c8 /t/a/awscdk (master)>
```

This is the stack code that the CDK automatically created for us to start editing with our business logic like defining a Lambda function by imports as below

```
from aws_cdk import (core,
aws
    {} aws_kms
    {} aws_lambda
    {} aws_logs
    {} aws_route53
    {} aws_route53_targets
    {} aws_s3
    {} aws_s3_assets
    {} aws_sam
    {} aws_secretsmanager
    {} aws_servicediscovery
    {} aws_sns
    {} aws_sns_subscriptions
```

```
from aws_cdk import (core,
aws_lambda,
aws
    {} aws_lambda
    {} aws_logs
    {} aws_route53
    {} aws_route53_targets
    {} aws_s3
    {} aws_s3_assets
    {} aws_sam
    {} aws_secretsmanager
    {} aws_servicediscovery
    {} aws_sns
    {} aws_sns_subscriptions
    {} aws_sqs
```

Untitled-1 | awscdk_stack.py

private > tmp > awscdk > awscdk > awscdk_stack.py > ...

```python
from aws_cdk import (core,
    aws_lambda
)

class AwscdkStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)


        # The code that defines your stack goes here

```

Untitled-1 | awscdk_stack.py

private > tmp > awscdk > awscdk > awscdk_stack.py > ...

```python
from aws_cdk import (core,
    aws_lambda
)

class AwscdkStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)

        function1 = aws_lambda.
```

Storage.configure	Manual Storage Config… >
Storage.get	Storage Get (AWS Amplif…
Storage.list	Storage List (AWS Ampli…
Storage.put	Storage Put (AWS Amplif…
Storage.remove	Storage Remove (AWS Amp…
{} abc	
Alias	
AliasAttributes	
AliasOptions	
AliasProps	

TERMINAL PROBLEMS OUTPUT DEBUG CON

```
__init__.py      awscdk_stack.py
marekku@a483e78888c8 /t/a/awscdk (mas
marekku@a483e78888c8 /t/a/awscdk (mas
```

master | Python 3.8.6 64-bit | ⊗ 0 ⚠ 0 | Ln 10

Untitled-1 | awscdk_stack.py

private > tmp > awscdk > awscdk > awscdk_stack.py > AwscdkStack > __init__

```python
from aws_cdk import (core,
    aws_lambda
)

class AwscdkStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)

        function1 = aws_lambda.Function(self, 'testlambda',
            code = aws)
```

{} aws_lambda	
aws-amplify-react	aws-amplify-react (AWS …
AwscdkStack	
async/with	Code snippet for an asy…
allow_public_subnet=	

TERMINAL PROBLEMS OUTPUT

```
__init__.py      awscdk_stack.py
```

Untitled-1 awscdk_stack.py ●

private > tmp > awscdk > awscdk > awscdk_

```
1  from aws_cdk import (core,
2      aws_lambda
3  )
4
5  class AwscdkStack(core.Stack):
6
7      def __init__(self, scope: co
8          super().__init__(scope,
9
10         function1 = aws_lambda.F
11             code = aws_lambda.Code.
```

Storage.configure	Manual Storage Configur…
Storage.get	Storage Get (AWS Amplif…
Storage.list	Storage List (AWS Ampli…
Storage.put	Storage Put (AWS Amplif…
Storage.remove	Storage Remove (AWS Amp…
allow_all_outbound=	
allow_public_subnet=	
asset	
bind	
bind_to_resource	
bucket	
cfn_parameters	

Untitled-1 awscdk_stack.py ●

private > tmp > awscdk > awscdk > awscdk_stack.py > ...

```
2      aws_lambda
3  )
4
5  class AwscdkStack(core.Stack):
6
7      def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
8          super().__init__(scope, id, **kwargs)
9
10         function1 = aws_lambda.Function(self, 'testlambda',
11             code = aws_lambda.Code.from_asset('./lambda'),
12             handler = 'lambda.handler',
13             timeout =
```

Untitled-1 awscdk_stack.py ●

private > tmp > awscdk > awscdk > awscdk_stack.py > ...

```
2      aws_lambda
3  )
4
5  class AwscdkStack(core.Stack):
6
7      def __init__(self, scope: co
8          super().__init__(scope,
9
10         function1 = aws_lambda.F
11             code = aws_lambda.Co
12             handler = 'lambda.ha
13             timeout = core.Duration.sec
```

seconds	
security_group=	
security_groups=	
scope	
to_seconds	
on_success=	
__subclasscheck__	
__subclasses__	

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: fish

```
__init__.py      awscdk_stack.py
marekku@a483e78888c8 /t/a/awscdk (master)> code awscdk_stack.py
marekku@a483e78888c8 /t/a/awscdk (master)>
```

Untitled-1 | awscdk_stack.py

private > tmp > awscdk > awscdk > awscdk_stack.py > AwscdkStack > __init__

```python
)

class AwscdkStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)

        function1 = aws_lambda.Function(self, 'testlambda',
            code = aws_lambda.Code.from_asset('./lambda'),
            handler = 'lambda.handler',
            timeout = core.Duration.seconds(3)
        )
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: node

```
marekku@a483e78888c8 /t/a/awscdk (master) [1]> cd ..
marekku@a483e78888c8 /t/awscdk (master)> cdk diff
```

Untitled-1 | awscdk_stack.py ●

private > tmp > awscdk > awscdk > awscdk_stack.py > AwscdkStack > __init__

```python
        function1 = aws_lambda.Function(self, 'testlambda',
            code = aws_lambda.Code.from_asset('./lambda'),
            handler = 'lambda.handler',
            timeout = core.Duration.seconds(3),
            runtime = aws_lambda.Runtime.py
        )
```

PYTHON_2_7
PYTHON_3_6
PYTHON_3_7
PYTHON_3_8
initial_policy=

TERMINAL PROBLEMS OUTPUT DEBUG CON

```
Traceback (most recent call last):
  File "app.py", line 9, in <module>
    AwscdkStack(app, "awscdk")
  File "/usr/local/lib/python3.8/site-packages/jsii/_runtime.py", line 69, in __call__
    inst = super().__call__(*args, **kwargs)
  File "/private/tmp/awscdk/awscdk/awscdk_stack.py", line 10, in __init__
    function1 = aws_lambda.Function(self, 'testlambda',
  File "/usr/local/lib/python3.8/site-packages/jsii/_runtime.py", line 69, in __call__
    inst = super().__call__(*args, **kwargs)
TypeError: __init__() missing 1 required keyword-only argument: 'runtime'
Subprocess exited with error 1
marekku@a483e78888c8 /t/awscdk (master) [1]>
```

Untitled-1 | awscdk_stack.py

private > tmp > awscdk > awscdk > awscdk_stack.py > AwscdkStack > __init__

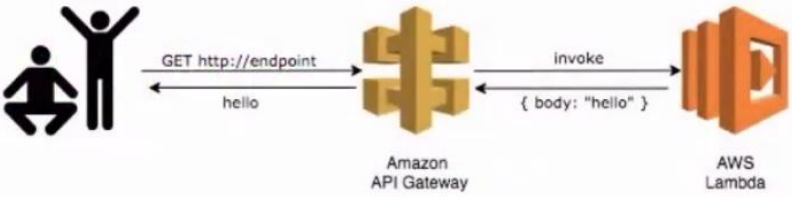
```
10        function1 = aws_lambda.Function(self, 'testlambda',
11            code = aws_lambda.Code.from_asset('./lambda'),
12            handler = 'lambda.handler',
13            timeout = core.Duration.seconds(3),
14            runtime = aws_lambda.Runtime.PYTHON_3_8
15        )
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: node

```
  File "app.py", line 9, in <module>
    AwscdkStack(app, "awscdk")
  File "/usr/local/lib/python3.8/site-packages/jsii/_runtime.py", line 69, in __call__
    inst = super().__call__(*args, **kwargs)
  File "/private/tmp/awscdk/awscdk/awscdk_stack.py", line 10, in __init__
    function1 = aws_lambda.Function(self, 'testlambda',
  File "/usr/local/lib/python3.8/site-packages/jsii/_runtime.py", line 69, in __call__
    inst = super().__call__(*args, **kwargs)
TypeError: __init__() missing 1 required keyword-only argument: 'runtime'
Subprocess exited with error 1
marekku@a483e78888c8 /t/awscdk (master) [1]> cdk ls
```

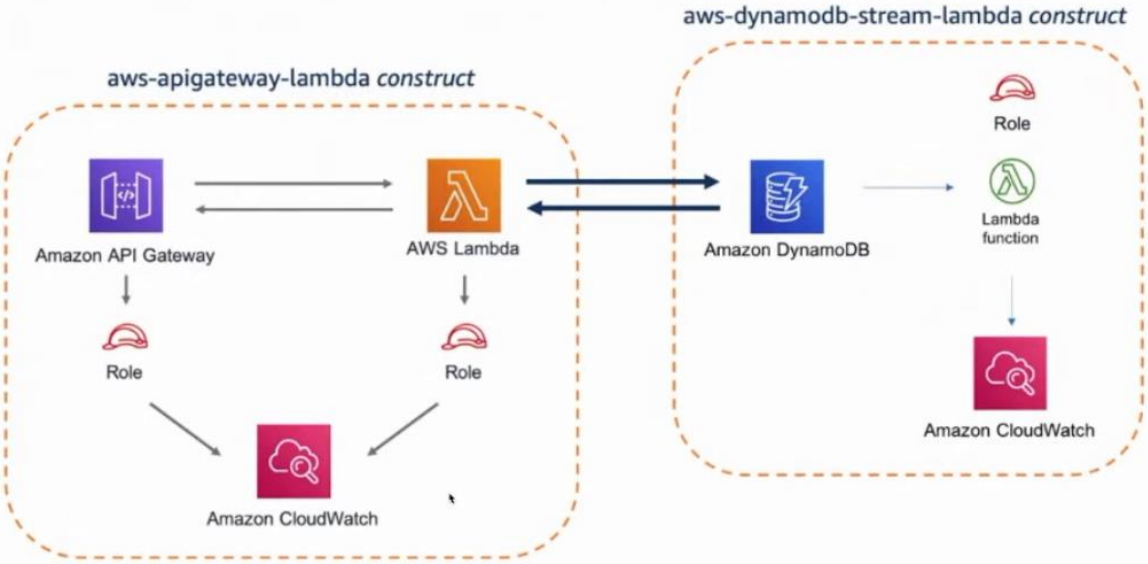
Where can I learn more about CDK?

Follow our self guided workshop: https://cdkworkshop.com/

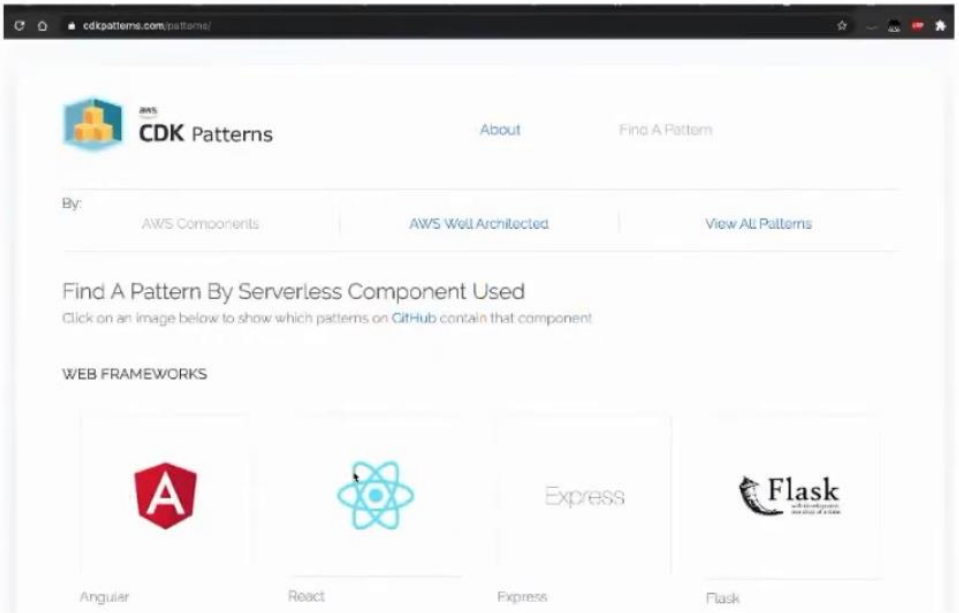


You can complete the workshop in Java, .NET, Python and TypeScript

New - AWS Solutions Constructs for CDK



Find additional constructs on cdkpatterns.com



Join the ServerlessDays Meetup group!

Our next meetup is on November 4th!

https://www.meetup.com/ServerlessDays-Amsterdam/