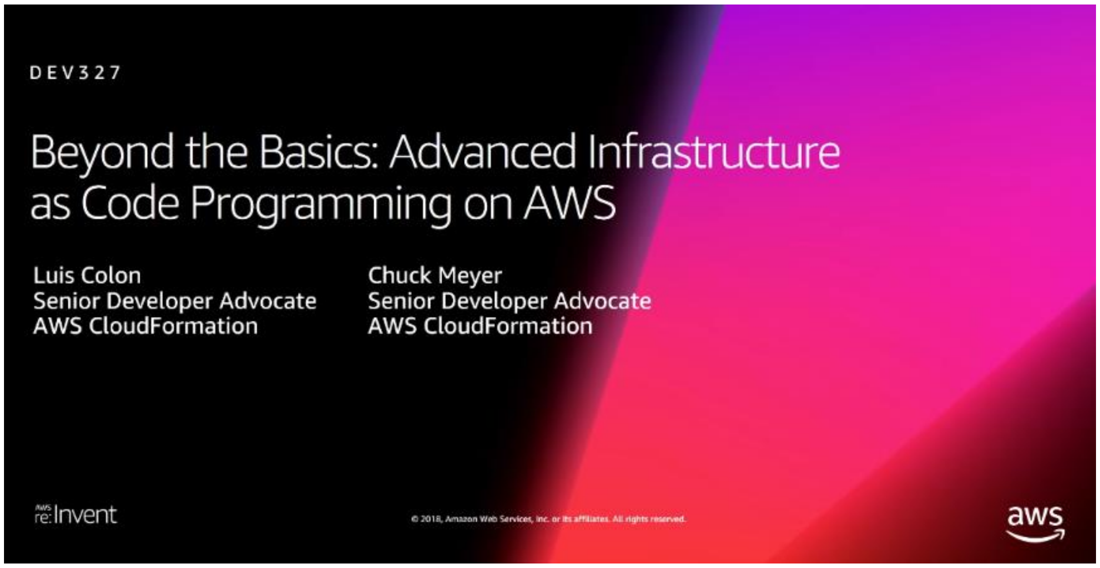


DEV327

Beyond the Basics: Advanced Infrastructure as Code Programming on AWS

Luis Colon
Senior Developer Advocate
AWS CloudFormation

Chuck Meyer
Senior Developer Advocate
AWS CloudFormation

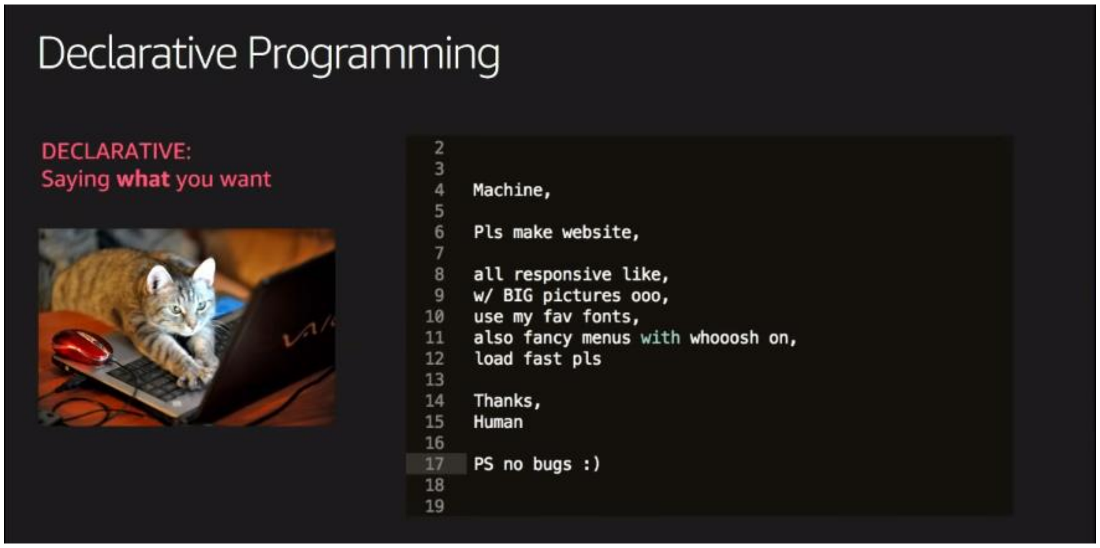
In addition to the basic **infrastructure as code capabilities provided by AWS CloudFormation**, AWS now offers various programmability constructs to power complex provisioning use cases. In this talk, we present several advanced use cases of declarative, imperative, and mixed coding scenarios that cloud infrastructure developers can leverage. Examples include demonstrating how to create custom resources and leveraging transforms, like the AWS Serverless Application Model (AWS SAM), to create both simple and complex macros with AWS CloudFormation. Complete Title: AWS re:Invent 2018: [REPEAT 1] Beyond the Basics: Advanced Infrastructure as Code Programming on AWS (DEV327-R1)

Agenda

- Declarative vs Imperative Coding
- Deep Dive: AWS CloudFormation Macros
- Deep Dive: Imperative AWS CloudFormation

We will see ways to extend CF by injecting more capabilities in CF.

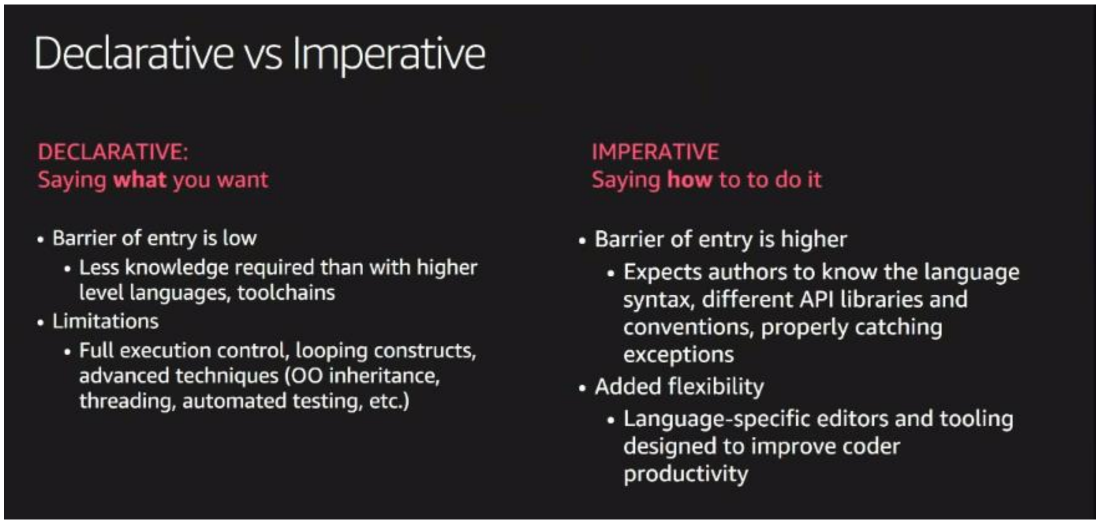
Programming Options Overview



Declarative Programming

DECLARATIVE:
Saying **what** you want

```
Machine,

Pls make website,

all responsive like,
w/ BIG pictures ooo,
use my fav fonts,
also fancy menus with whooosh on,
load fast pls

Thanks,
Human

PS no bugs :)
```



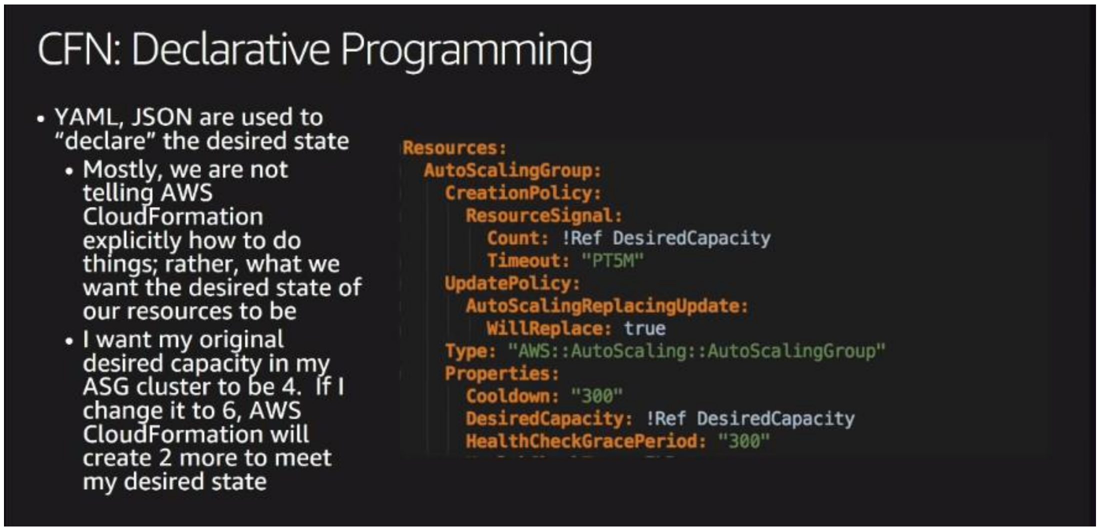
Declarative vs Imperative

DECLARATIVE:
Saying **what** you want

- Barrier of entry is low
  - Less knowledge required than with higher level languages, toolchains
- Limitations
  - Full execution control, looping constructs, advanced techniques (OO inheritance, threading, automated testing, etc.)

IMPERATIVE
Saying **how** to to do it

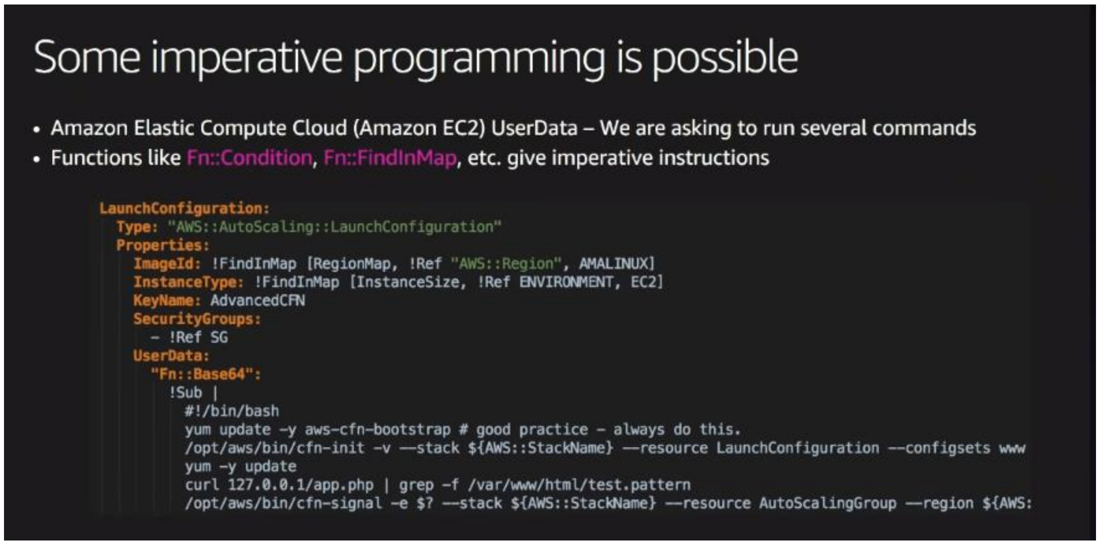
- Barrier of entry is higher
  - Expects authors to know the language syntax, different API libraries and conventions, properly catching exceptions
- Added flexibility
  - Language-specific editors and tooling designed to improve coder productivity



CFN: Declarative Programming

- YAML, JSON are used to "declare" the desired state
  - Mostly, we are not telling AWS CloudFormation explicitly how to do things; rather, what we want the desired state of our resources to be
  - I want my original desired capacity in my ASG cluster to be 4. If I change it to 6, AWS CloudFormation will create 2 more to meet my desired state

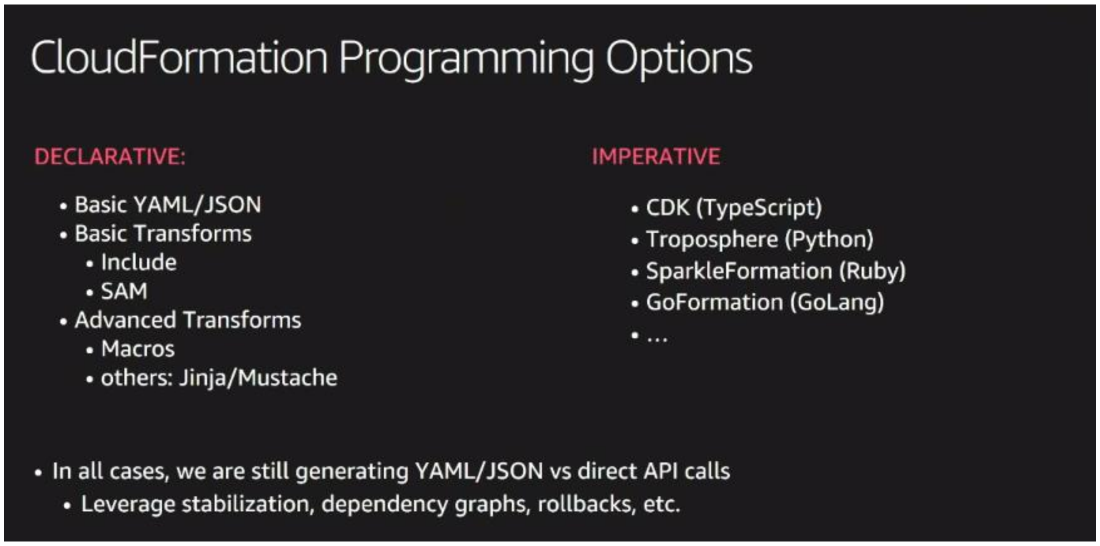
```
Resources:
  AutoScalingGroup:
    CreationPolicy:
      ResourceSignal:
        Count: !Ref DesiredCapacity
        Timeout: "PT5M"
    UpdatePolicy:
      AutoScalingReplacingUpdate:
        WillReplace: true
    Type: "AWS::AutoScaling::AutoScalingGroup"
    Properties:
      Cooldown: "300"
      DesiredCapacity: !Ref DesiredCapacity
      HealthCheckGracePeriod: "300"
```



Some imperative programming is possible

- Amazon Elastic Compute Cloud (Amazon EC2) UserData – We are asking to run several commands
- Functions like Fn::Condition, Fn::FindInMap, etc. give imperative instructions

```
LaunchConfiguration:
  Type: "AWS::AutoScaling::LaunchConfiguration"
  Properties:
    ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", AMALINUX]
    InstanceType: !FindInMap [InstanceSize, !Ref ENVIRONMENT, EC2]
    KeyName: AdvancedCFN
    SecurityGroups:
      - !Ref SG
    UserData:
      "Fn::Base64":
        !Sub |
          #!/bin/bash
          yum update -y aws-cfn-bootstrap # good practice - always do this.
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource LaunchConfiguration --configsets www
          yum -y update
          curl 127.0.0.1/app.php | grep -f /var/www/html/test.pattern
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource AutoScalingGroup --region ${AWS:
```



CloudFormation Programming Options

DECLARATIVE:

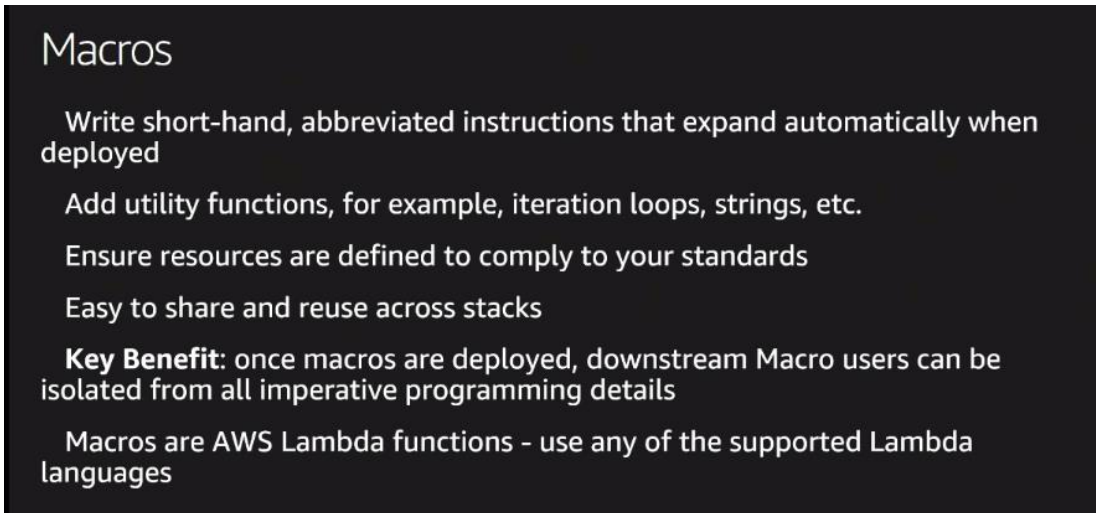
- Basic YAML/JSON
- Basic Transforms
  - Include
  - SAM
- Advanced Transforms
  - Macros
  - others: Jinja/Mustache

IMPERATIVE

- CDK (TypeScript)
- Troposphere (Python)
- SparkleFormation (Ruby)
- GoFormation (GoLang)
- …

- In all cases, we are still generating YAML/JSON vs direct API calls
  - Leverage stabilization, dependency graphs, rollbacks, etc.

Deep Dive: Macros



Macros

Write short-hand, abbreviated instructions that expand automatically when deployed

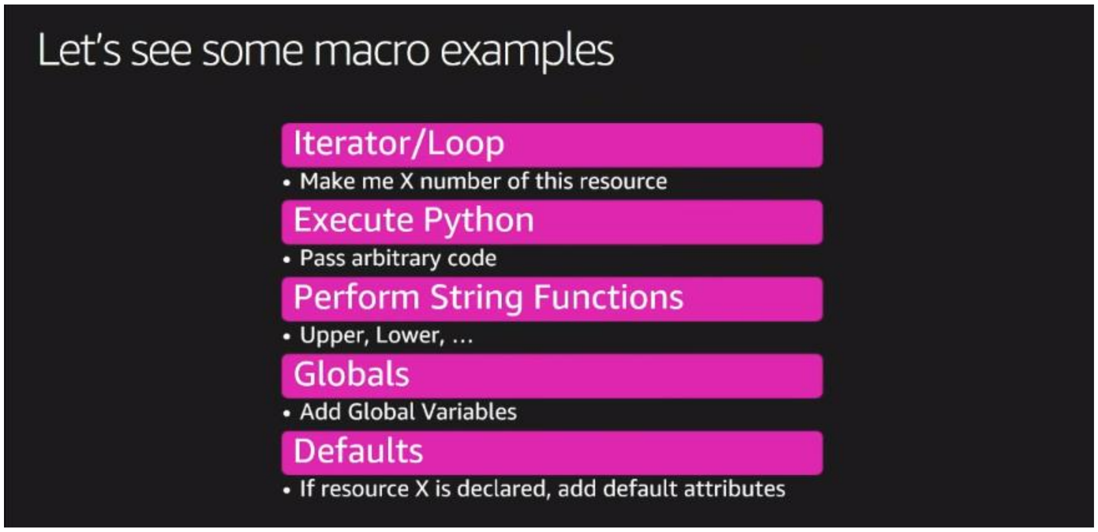
Add utility functions, for example, iteration loops, strings, etc.

Ensure resources are defined to comply to your standards

Easy to share and reuse across stacks

Key Benefit: once macros are deployed, downstream Macro users can be isolated from all imperative programming details

Macros are AWS Lambda functions - use any of the supported Lambda languages



Let's see some macro examples

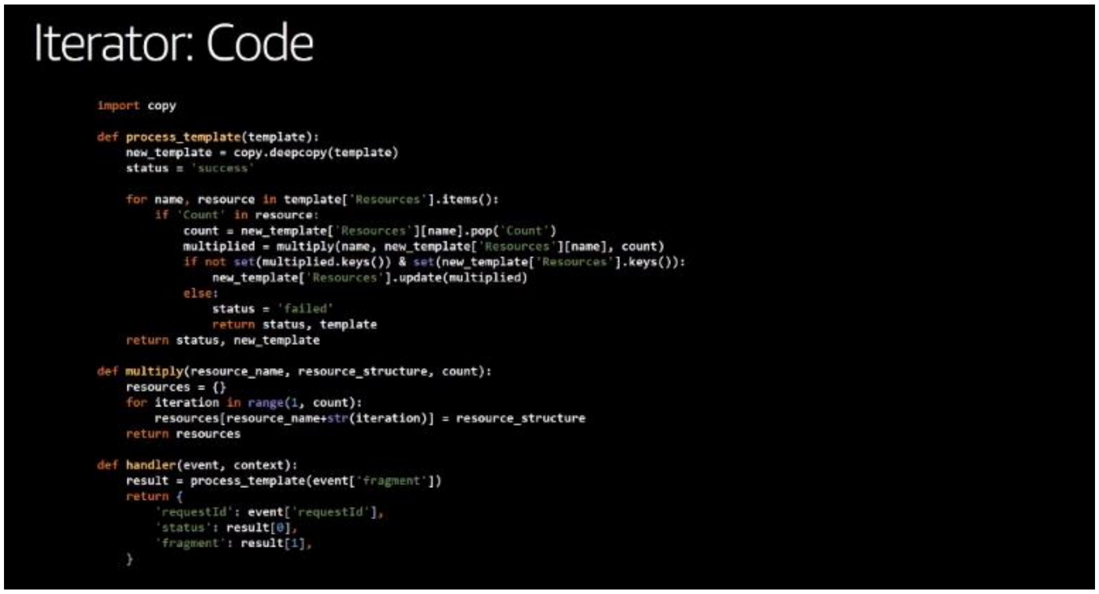
Iterator/Loop
- Make me X number of this resource

Execute Python
- Pass arbitrary code

Perform String Functions
- Upper, Lower, …

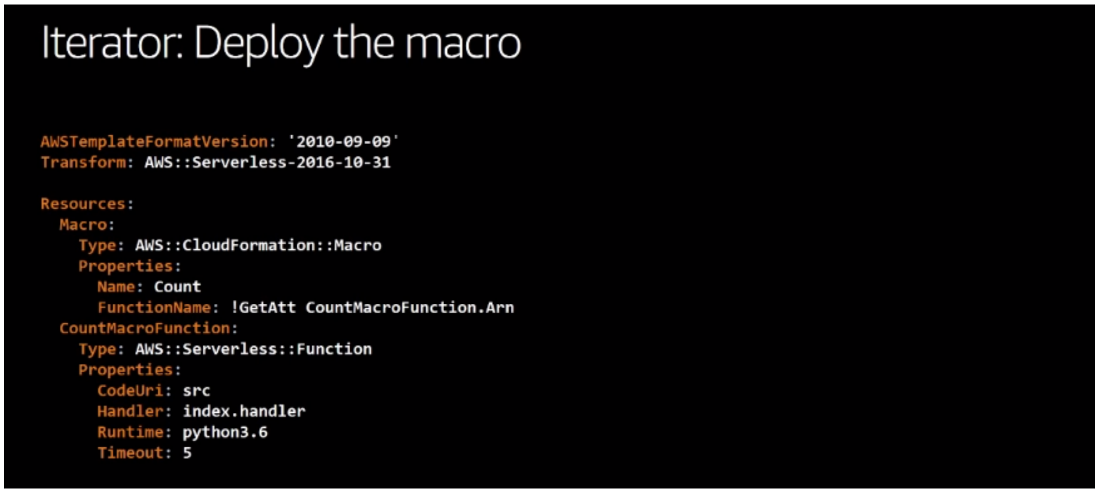
Globals
- Add Global Variables

Defaults
- If resource X is declared, add default attributes



Iterator: Code

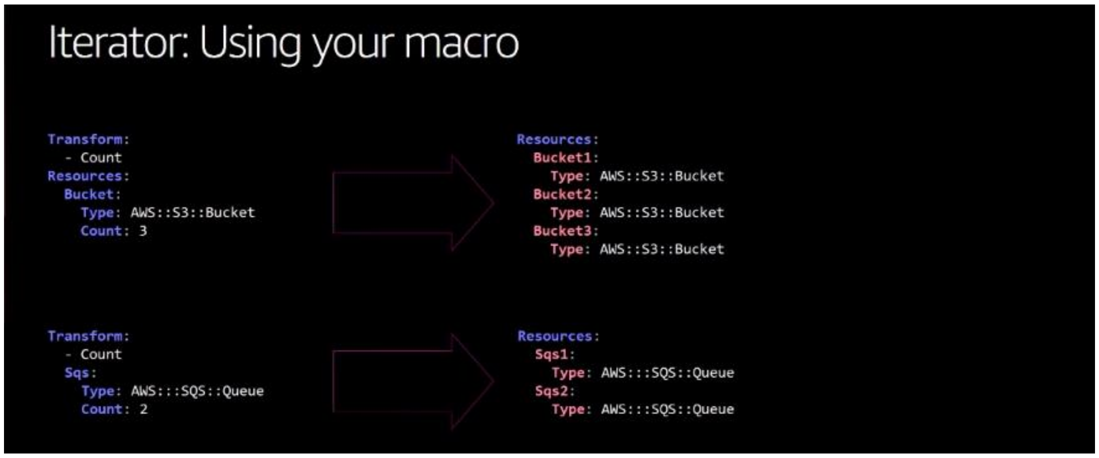
```python
import copy

def process_template(template):
    new_template = copy.deepcopy(template)
    status = 'success'

    for name, resource in template['Resources'].items():
        if 'Count' in resource:
            count = new_template['Resources'][name].pop('Count')
            multiplied = multiply(name, new_template['Resources'][name], count)
            if not set(multiplied.keys()) & set(new_template['Resources'].keys()):
                new_template['Resources'].update(multiplied)
            else:
                status = 'failed'
                return status, template
    return status, new_template

def multiply(resource_name, resource_structure, count):
    resources = {}
    for iteration in range(1, count):
        resources[resource_name+str(iteration)] = resource_structure
    return resources

def handler(event, context):
    result = process_template(event['fragment'])
    return {
        'requestId': event['requestId'],
        'status': result[0],
        'fragment': result[1],
    }
```



Iterator: Deploy the macro

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

Resources:
  Macro:
    Type: AWS::CloudFormation::Macro
    Properties:
      Name: Count
      FunctionName: !GetAtt CountMacroFunction.Arn
  CountMacroFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: src
      Handler: index.handler
      Runtime: python3.6
      Timeout: 5
```

Here, we have a macro called Count that deploys code



Iterator: Using your macro

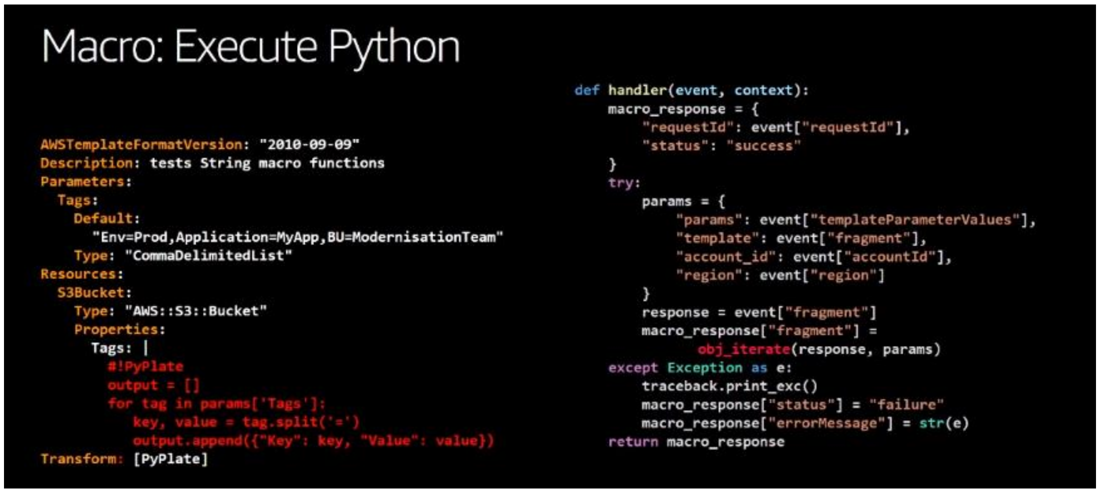
```
Transform:
  - Count
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Count: 3
```

```
Resources:
  Bucket1:
    Type: AWS::S3::Bucket
  Bucket2:
    Type: AWS::S3::Bucket
  Bucket3:
    Type: AWS::S3::Bucket
```

```
Transform:
  - Count
  Sqs:
    Type: AWS:::SQS::Queue
    Count: 2
```

```
Resources:
  Sqs1:
    Type: AWS:::SQS::Queue
  Sqs2:
    Type: AWS:::SQS::Queue
```

This is what the iterator macro looks like when processed by CF

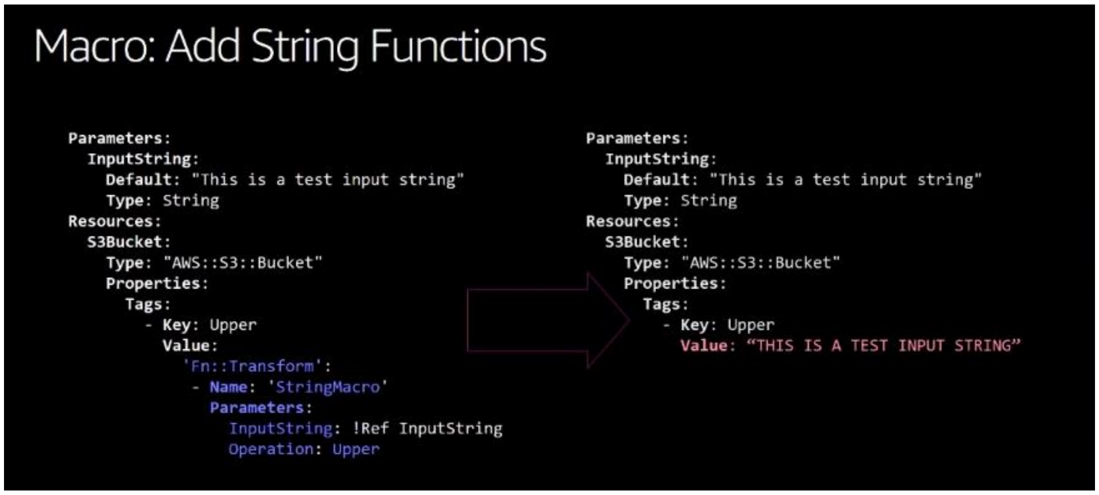


Macro: Execute Python

```
AWSTemplateFormatVersion: "2010-09-09"
Description: tests String macro functions
Parameters:
  Tags:
    Default:
      "Env=Prod,Application=MyApp,BU=ModernisationTeam"
    Type: "CommaDelimitedList"
Resources:
  S3Bucket:
    Type: "AWS::S3::Bucket"
    Properties:
      Tags: |
        #!PyPlate
        output = []
        for tag in params['Tags']:
          key, value = tag.split('=')
          output.append({"Key": key, "Value": value})
Transform: [PyPlate]
```

```
def handler(event, context):
    macro_response = {
        "requestId": event["requestId"],
        "status": "success"
    }
    try:
        params = {
            "params": event["templateParameterValues"],
            "template": event["fragment"],
            "account_id": event["accountId"],
            "region": event["region"]
        }
        response = event["fragment"]
        macro_response["fragment"] =
                obj_iterate(response, params)
    except Exception as e:
        traceback.print_exc()
        macro_response["status"] = "failure"
        macro_response["errorMessage"] = str(e)
    return macro_response
```

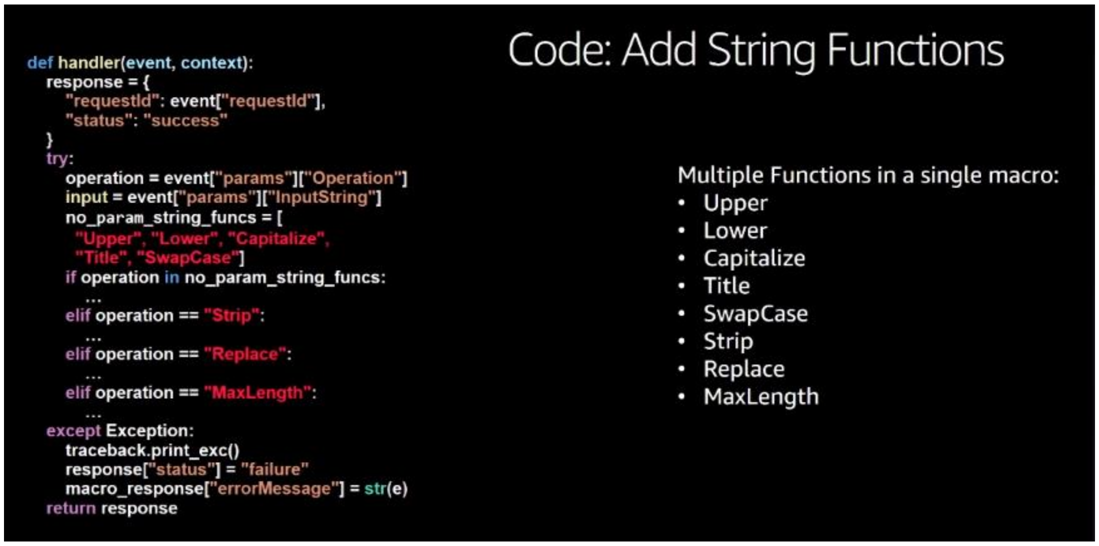
This macro executes arbitrary python code



Macro: Add String Functions

```
Parameters:
  InputString:
    Default: "This is a test input string"
    Type: String
Resources:
  S3Bucket:
    Type: "AWS::S3::Bucket"
    Properties:
      Tags:
        - Key: Upper
          Value:
            'Fn::Transform':
             - Name: 'StringMacro'
               Parameters:
                 InputString: !Ref InputString
                 Operation: Upper
```

```
Parameters:
  InputString:
    Default: "This is a test input string"
    Type: String
Resources:
  S3Bucket:
    Type: "AWS::S3::Bucket"
    Properties:
      Tags:
        - Key: Upper
          Value: "THIS IS A TEST INPUT STRING"
```

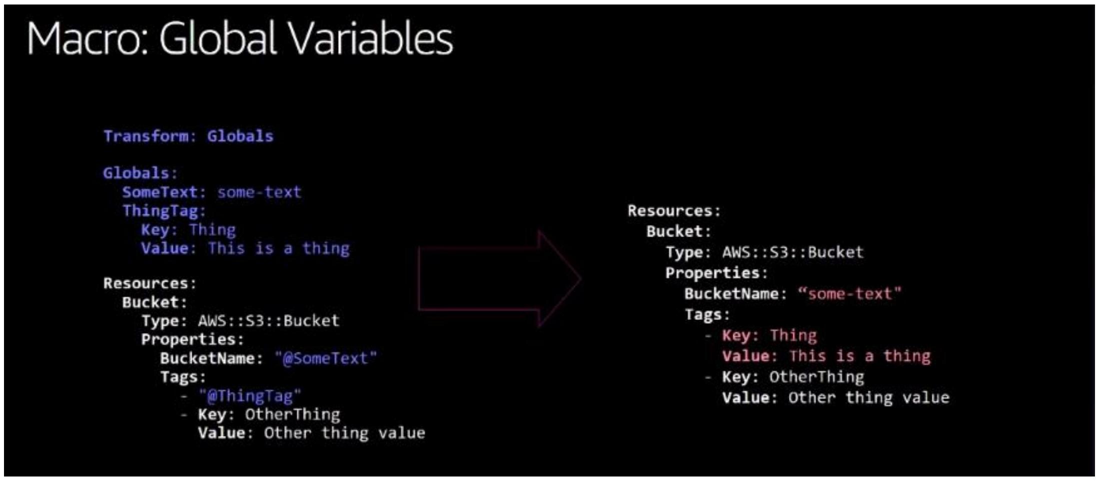


Code: Add String Functions

```python
def handler(event, context):
  response = {
      "requestId": event["requestId"],
      "status": "success"
  }
  try:
      operation = event["params"]["Operation"]
      input = event["params"]["InputString"]
      no_param_string_funcs = [
        "Upper", "Lower", "Capitalize",
        "Title", "SwapCase"]
      if operation in no_param_string_funcs:
          ...
      elif operation == "Strip":
          ...
      elif operation == "Replace":
          ...
      elif operation == "MaxLength":
          ...
  except Exception:
      traceback.print_exc()
      response["status"] = "failure"
      macro_response["errorMessage"] = str(e)
  return response
```

Multiple Functions in a single macro:

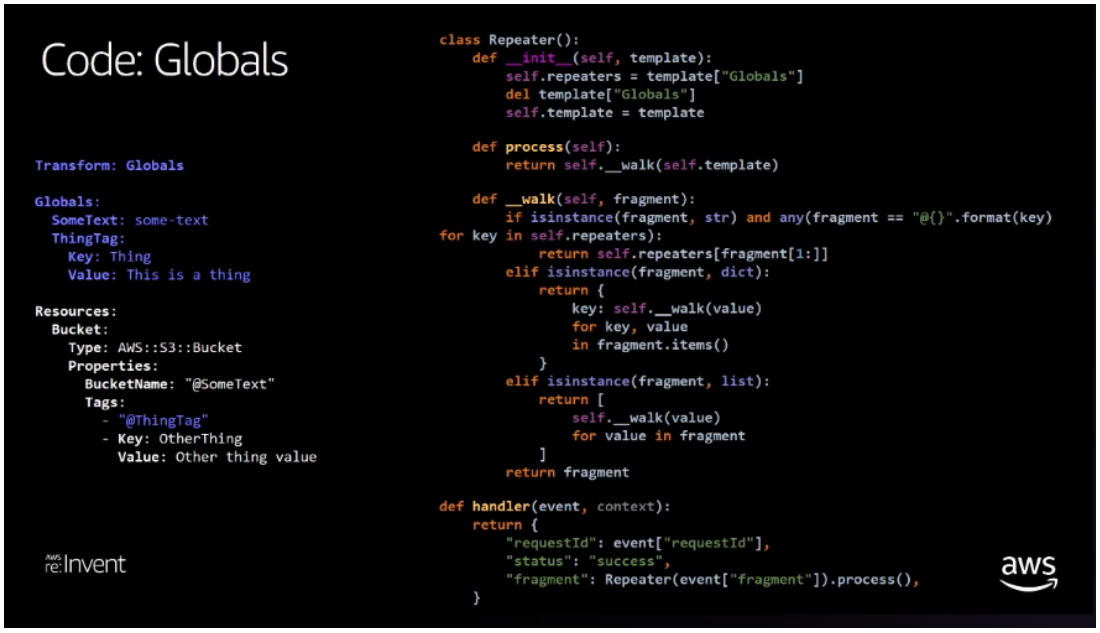
- Upper
- Lower
- Capitalize
- Title
- SwapCase
- Strip
- Replace
- MaxLength



Macro: Global Variables

```yaml
Transform: Globals

Globals:
  SomeText: some-text
  ThingTag:
    Key: Thing
    Value: This is a thing

Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: "@SomeText"
      Tags:
        - "@ThingTag"
        - Key: OtherThing
          Value: Other thing value
```

```yaml
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: “some-text"
      Tags:
        - Key: Thing
          Value: This is a thing
        - Key: OtherThing
          Value: Other thing value
```



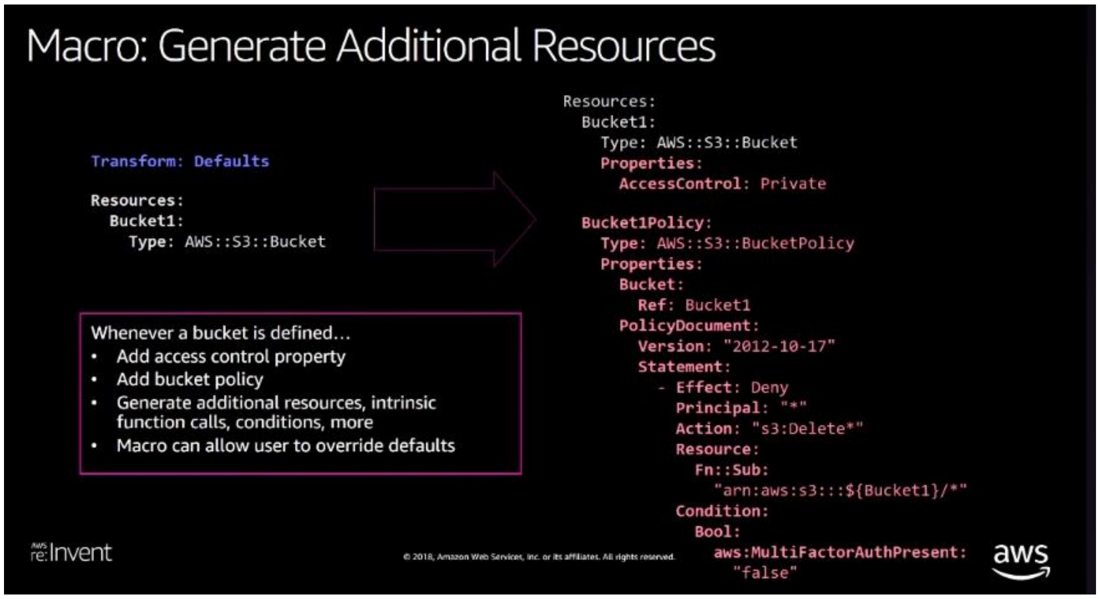
Code: Globals

```yaml
Transform: Globals

Globals:
  SomeText: some-text
  ThingTag:
    Key: Thing
    Value: This is a thing

Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: "@SomeText"
      Tags:
        - "@ThingTag"
        - Key: OtherThing
          Value: Other thing value
```

```python
class Repeater():
    def __init__(self, template):
        self.repeaters = template["Globals"]
        del template["Globals"]
        self.template = template

    def process(self):
        return self.__walk(self.template)

    def __walk(self, fragment):
        if isinstance(fragment, str) and any(fragment == "@{}".format(key)
for key in self.repeaters):
            return self.repeaters[fragment[1:]]
        elif isinstance(fragment, dict):
            return {
                key: self.__walk(value)
                for key, value
                in fragment.items()
            }
        elif isinstance(fragment, list):
            return [
                self.__walk(value)
                for value in fragment
            ]
        return fragment

def handler(event, context):
    return {
        "requestId": event["requestId"],
        "status": "success",
        "fragment": Repeater(event["fragment"]).process(),
    }
```

re:Invent

aws



Macro: Generate Additional Resources

```
Transform: Defaults

Resources:
  Bucket1:
    Type: AWS::S3::Bucket
```

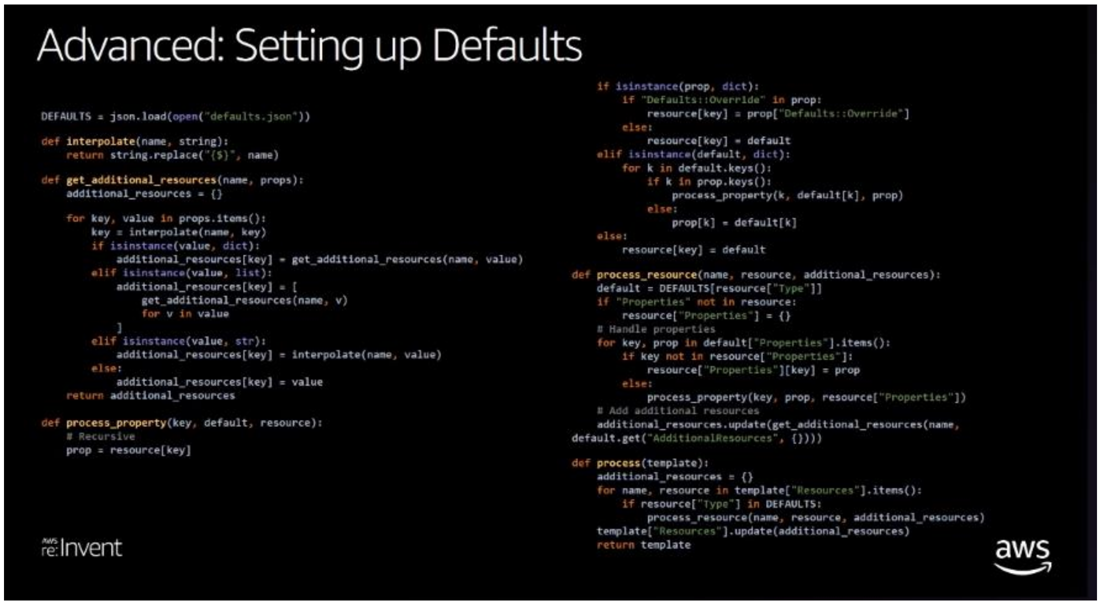
```
Resources:
  Bucket1:
    Type: AWS::S3::Bucket
    Properties:
      AccessControl: Private

  Bucket1Policy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: Bucket1
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Deny
            Principal: "*"
            Action: "s3:Delete*"
            Resource:
              Fn::Sub:
                "arn:aws:s3:::${Bucket1}/*"
            Condition:
              Bool:
                aws:MultiFactorAuthPresent:
                  "false"
```

Whenever a bucket is defined…
- Add access control property
- Add bucket policy
- Generate additional resources, intrinsic function calls, conditions, more
- Macro can allow user to override defaults

re:Invent

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.

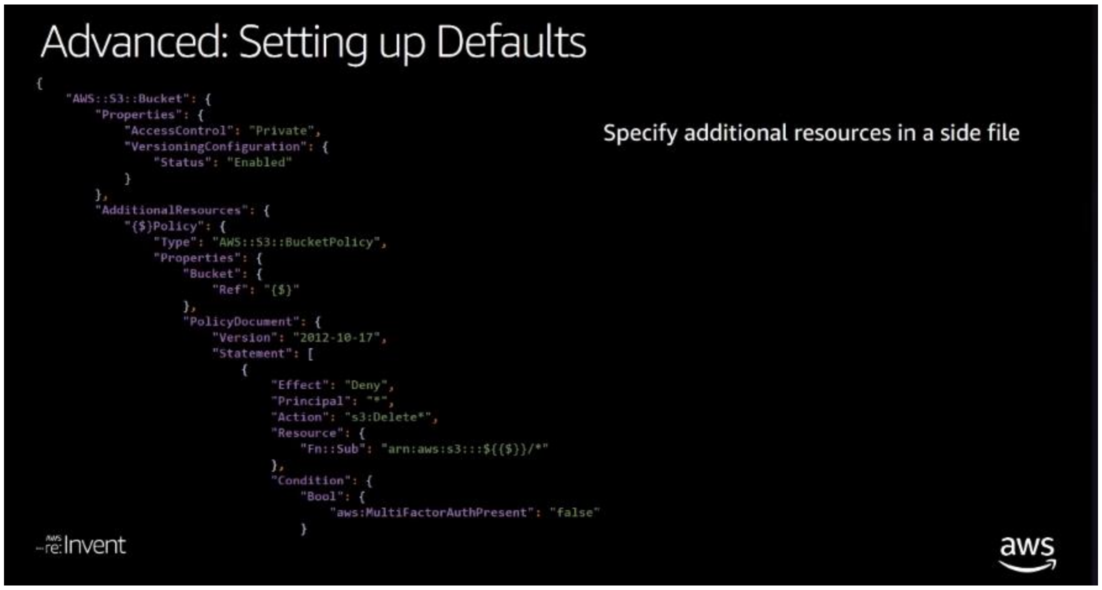
This simply takes what the user wants to do, then overlay some protections on it for them



Advanced: Setting up Defaults

```python
DEFAULTS = json.load(open("defaults.json"))

def interpolate(name, string):
    return string.replace("{$}", name)

def get_additional_resources(name, props):
    additional_resources = {}

    for key, value in props.items():
        key = interpolate(name, key)
        if isinstance(value, dict):
            additional_resources[key] = get_additional_resources(name, value)
        elif isinstance(value, list):
            additional_resources[key] = [
                get_additional_resources(name, v)
                for v in value
            ]
        elif isinstance(value, str):
            additional_resources[key] = interpolate(name, value)
        else:
            additional_resources[key] = value
    return additional_resources

def process_property(key, default, resource):
    # Recursive
    prop = resource[key]
    if isinstance(prop, dict):
        if "Defaults::Override" in prop:
            resource[key] = prop["Defaults::Override"]
        else:
            resource[key] = default
    elif isinstance(default, dict):
        for k in default.keys():
            if k in prop.keys():
                process_property(k, default[k], prop)
            else:
                prop[k] = default[k]
    else:
        resource[key] = default

def process_resource(name, resource, additional_resources):
    default = DEFAULTS[resource["Type"]]
    if "Properties" not in resource:
        resource["Properties"] = {}
    # Handle properties
    for key, prop in default["Properties"].items():
        if key not in resource["Properties"]:
            resource["Properties"][key] = prop
        else:
            process_property(key, prop, resource["Properties"])
    # Add additional resources
    additional_resources.update(get_additional_resources(name,
default.get("AdditionalResources", {})))

def process(template):
    additional_resources = {}
    for name, resource in template["Resources"].items():
        if resource["Type"] in DEFAULTS:
            process_resource(name, resource, additional_resources)
    template["Resources"].update(additional_resources)
    return template
```

re:Invent



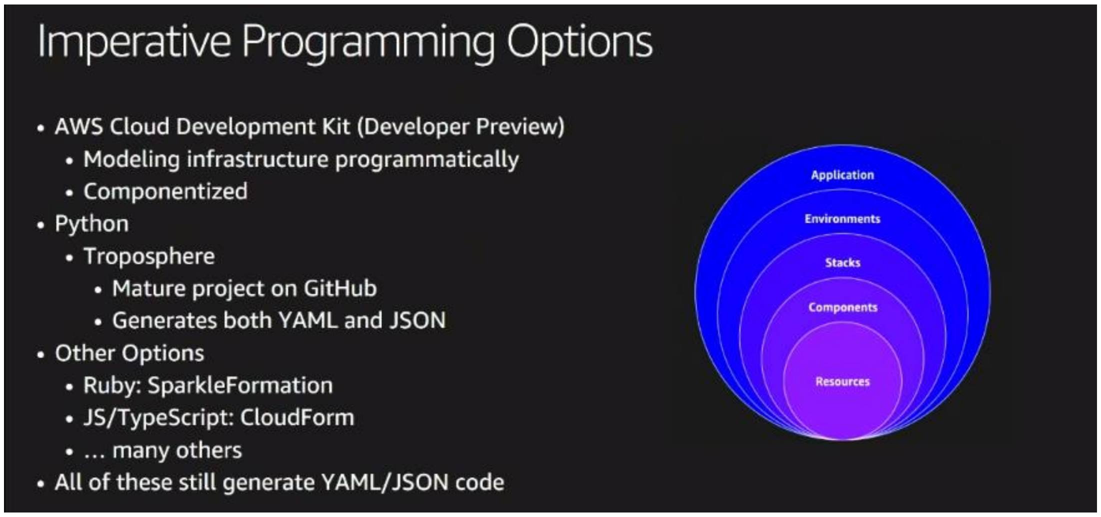
Advanced: Setting up Defaults

```json
{
    "AWS::S3::Bucket": {
        "Properties": {
            "AccessControl": "Private",
            "VersioningConfiguration": {
                "Status": "Enabled"
            }
        },
        "AdditionalResources": {
            "{$}Policy": {
                "Type": "AWS::S3::BucketPolicy",
                "Properties": {
                    "Bucket": {
                        "Ref": "{$}"
                    },
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Deny",
                                "Principal": "*",
                                "Action": "s3:Delete*",
                                "Resource": {
                                    "Fn::Sub": "arn:aws:s3:::${{$}}/*"
                                },
                                "Condition": {
                                    "Bool": {
                                        "aws:MultiFactorAuthPresent": "false"
                                    }
```

Specify additional resources in a side file

re:Invent

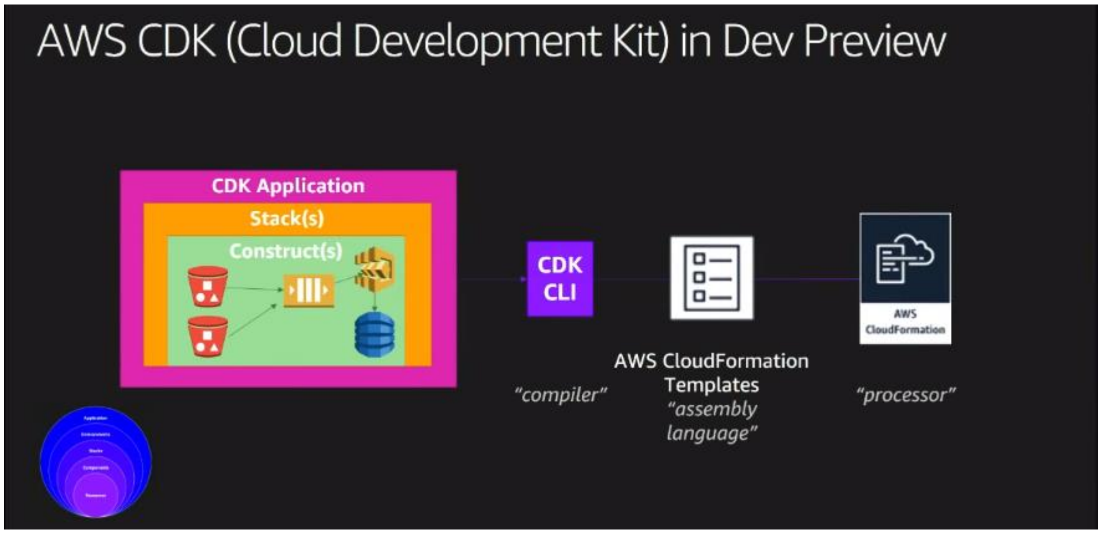
Imperative Programming in AWS CloudFormation



Imperative Programming Options

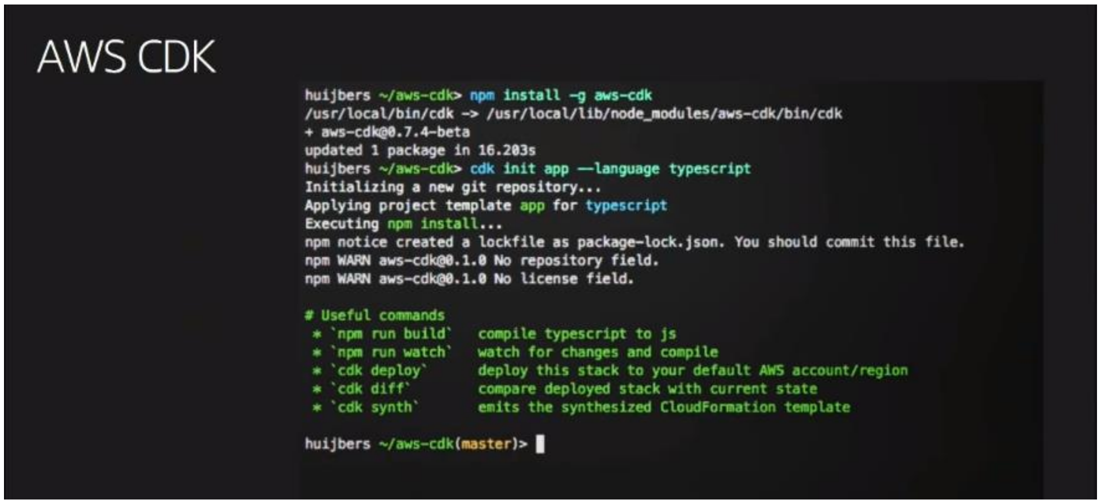
- AWS Cloud Development Kit (Developer Preview)
  - Modeling infrastructure programmatically
  - Componentized
- Python
  - Troposphere
    - Mature project on GitHub
    - Generates both YAML and JSON
- Other Options
  - Ruby: SparkleFormation
  - JS/TypeScript: CloudForm
  - … many others
- All of these still generate YAML/JSON code

Application
Environments
Stacks
Components
Resources



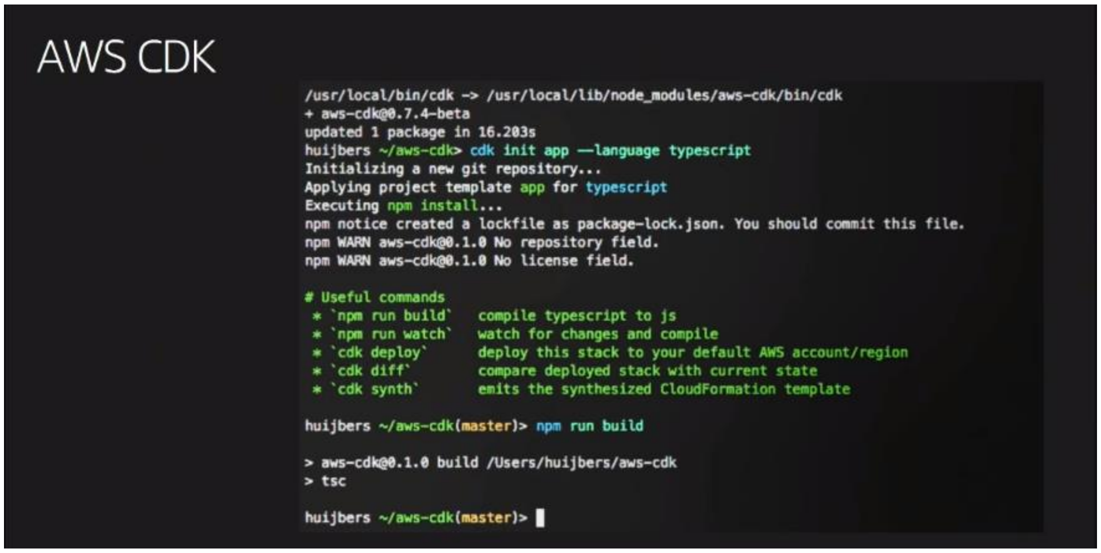
AWS CDK (Cloud Development Kit) in Dev Preview

You download the CDK and get a set of default models created for you that you can then add to, then you compile it to get the CF templates generated for you to use.



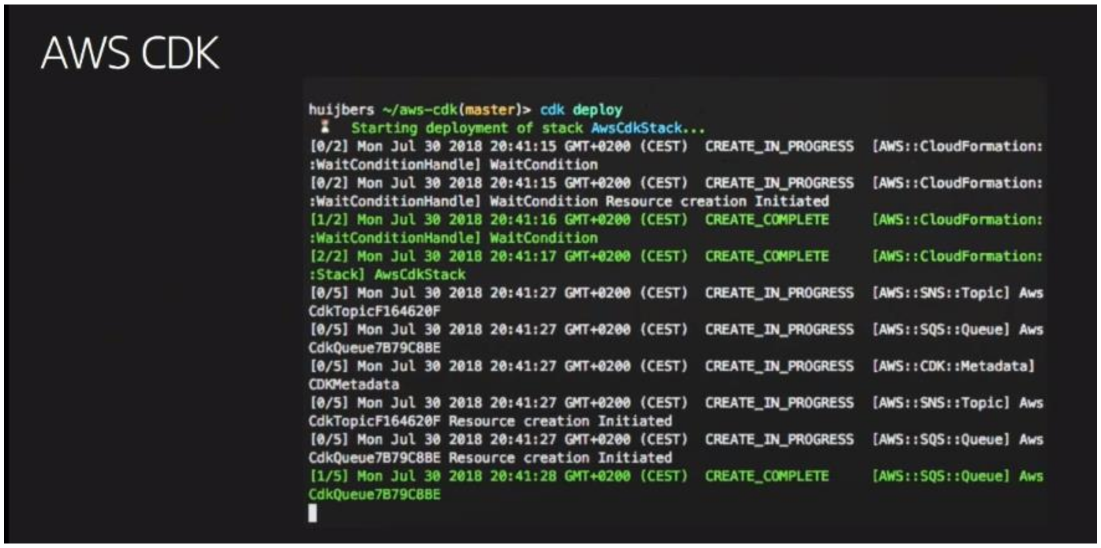
AWS CDK

```
huijbers ~/aws-cdk> npm install -g aws-cdk
/usr/local/bin/cdk -> /usr/local/lib/node_modules/aws-cdk/bin/cdk
+ aws-cdk@0.7.4-beta
updated 1 package in 16.203s
huijbers ~/aws-cdk> cdk init app —language typescript
Initializing a new git repository...
Applying project template app for typescript
Executing npm install...
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN aws-cdk@0.1.0 No repository field.
npm WARN aws-cdk@0.1.0 No license field.

# Useful commands
 * `npm run build`   compile typescript to js
 * `npm run watch`   watch for changes and compile
 * `cdk deploy`      deploy this stack to your default AWS account/region
 * `cdk diff`        compare deployed stack with current state
 * `cdk synth`       emits the synthesized CloudFormation template

huijbers ~/aws-cdk(master)> 
```



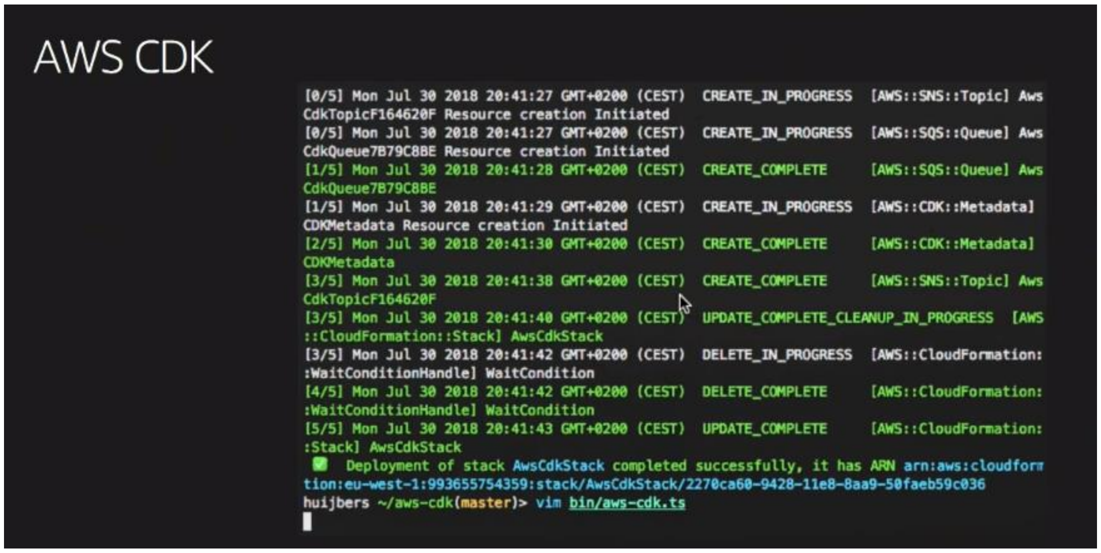
AWS CDK

```
/usr/local/bin/cdk -> /usr/local/lib/node_modules/aws-cdk/bin/cdk
+ aws-cdk@0.7.4-beta
updated 1 package in 16.203s
huijbers ~/aws-cdk> cdk init app —language typescript
Initializing a new git repository...
Applying project template app for typescript
Executing npm install...
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN aws-cdk@0.1.0 No repository field.
npm WARN aws-cdk@0.1.0 No license field.

# Useful commands
 * `npm run build`   compile typescript to js
 * `npm run watch`   watch for changes and compile
 * `cdk deploy`      deploy this stack to your default AWS account/region
 * `cdk diff`        compare deployed stack with current state
 * `cdk synth`       emits the synthesized CloudFormation template

huijbers ~/aws-cdk(master)> npm run build

> aws-cdk@0.1.0 build /Users/huijbers/aws-cdk
> tsc

huijbers ~/aws-cdk(master)> █
```



AWS CDK

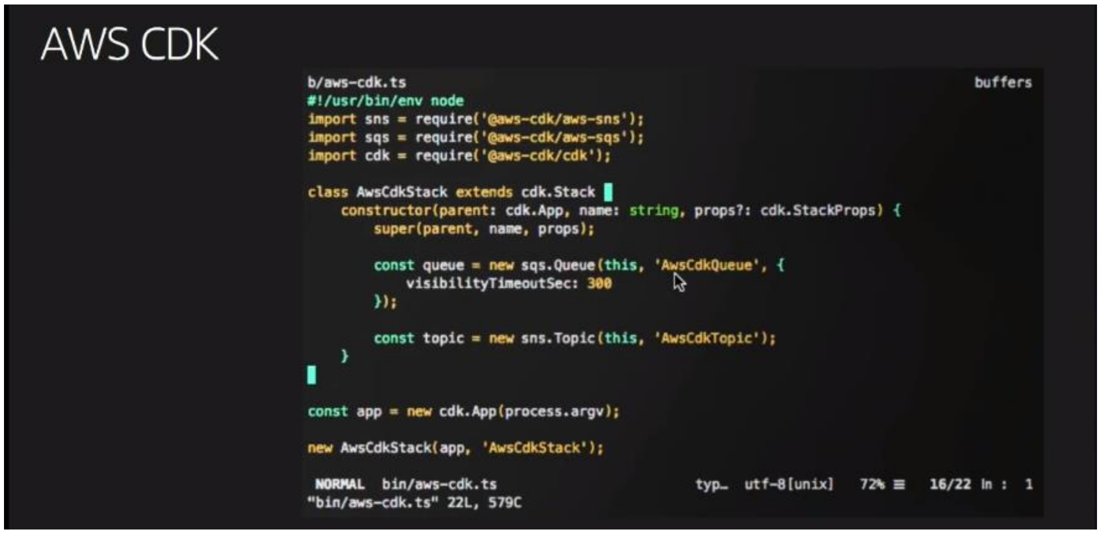
```
huijbers ~/aws-cdk(master)> cdk deploy
 ⏳  Starting deployment of stack AwsCdkStack...
[0/2] Mon Jul 30 2018 20:41:15 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::CloudFormation:
:WaitConditionHandle] WaitCondition
[0/2] Mon Jul 30 2018 20:41:15 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::CloudFormation:
:WaitConditionHandle] WaitCondition Resource creation Initiated
[1/2] Mon Jul 30 2018 20:41:16 GMT+0200 (CEST)  CREATE_COMPLETE     [AWS::CloudFormation:
:WaitConditionHandle] WaitCondition
[2/2] Mon Jul 30 2018 20:41:17 GMT+0200 (CEST)  CREATE_COMPLETE     [AWS::CloudFormation:
:Stack] AwsCdkStack
[0/5] Mon Jul 30 2018 20:41:27 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::SNS::Topic] Aws
CdkTopicF164620F
[0/5] Mon Jul 30 2018 20:41:27 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::SQS::Queue] Aws
CdkQueue7B79C8BE
[0/5] Mon Jul 30 2018 20:41:27 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::CDK::Metadata]
CDKMetadata
[0/5] Mon Jul 30 2018 20:41:27 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::SNS::Topic] Aws
CdkTopicF164620F Resource creation Initiated
[0/5] Mon Jul 30 2018 20:41:27 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::SQS::Queue] Aws
CdkQueue7B79C8BE Resource creation Initiated
[1/5] Mon Jul 30 2018 20:41:28 GMT+0200 (CEST)  CREATE_COMPLETE     [AWS::SQS::Queue] Aws
CdkQueue7B79C8BE
█
```



AWS CDK

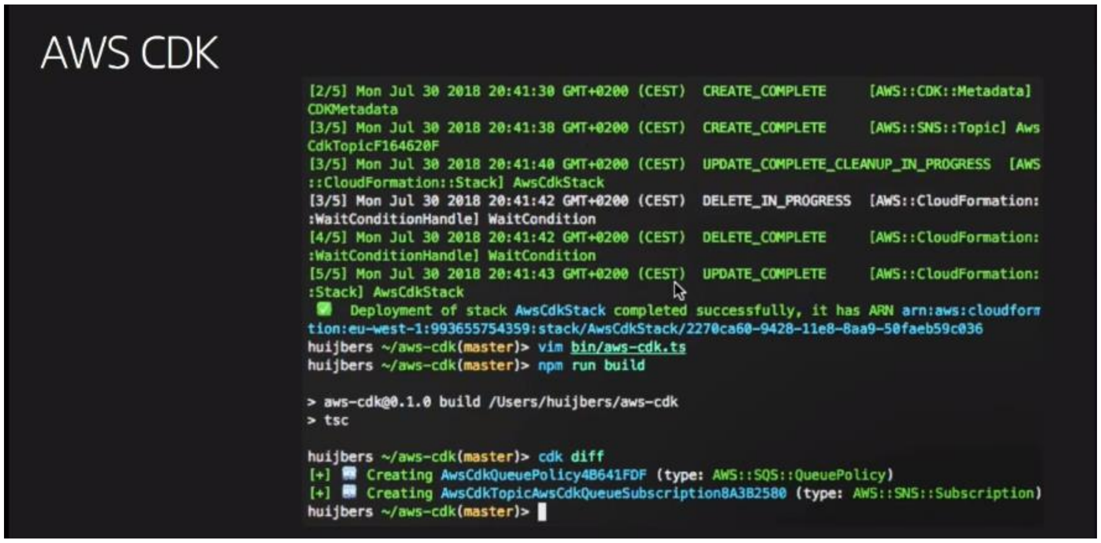
```
[0/5] Mon Jul 30 2018 20:41:27 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::SNS::Topic] Aws
CdkTopicF164620F Resource creation Initiated
[0/5] Mon Jul 30 2018 20:41:27 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::SQS::Queue] Aws
CdkQueue7B79C8BE Resource creation Initiated
[1/5] Mon Jul 30 2018 20:41:28 GMT+0200 (CEST)  CREATE_COMPLETE     [AWS::SQS::Queue] Aws
CdkQueue7B79C8BE
[1/5] Mon Jul 30 2018 20:41:29 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::CDK::Metadata]
CDKMetadata Resource creation Initiated
[2/5] Mon Jul 30 2018 20:41:30 GMT+0200 (CEST)  CREATE_COMPLETE     [AWS::CDK::Metadata]
CDKMetadata
[3/5] Mon Jul 30 2018 20:41:38 GMT+0200 (CEST)  CREATE_COMPLETE     [AWS::SNS::Topic] Aws
CdkTopicF164620F
[3/5] Mon Jul 30 2018 20:41:40 GMT+0200 (CEST)  UPDATE_COMPLETE_CLEANUP_IN_PROGRESS  [AWS
::CloudFormation::Stack] AwsCdkStack
[3/5] Mon Jul 30 2018 20:41:42 GMT+0200 (CEST)  DELETE_IN_PROGRESS  [AWS::CloudFormation:
:WaitConditionHandle] WaitCondition
[4/5] Mon Jul 30 2018 20:41:42 GMT+0200 (CEST)  DELETE_COMPLETE     [AWS::CloudFormation:
:WaitConditionHandle] WaitCondition
[5/5] Mon Jul 30 2018 20:41:43 GMT+0200 (CEST)  UPDATE_COMPLETE     [AWS::CloudFormation:
:Stack] AwsCdkStack
 ✅  Deployment of stack AwsCdkStack completed successfully, it has ARN arn:aws:cloudform
tion:eu-west-1:993655754359:stack/AwsCdkStack/2270ca60-9428-11e8-8aa9-50faeb59c036
huijbers ~/aws-cdk(master)> vim bin/aws-cdk.ts
█
```

Let us now edit part of the code to see what happens



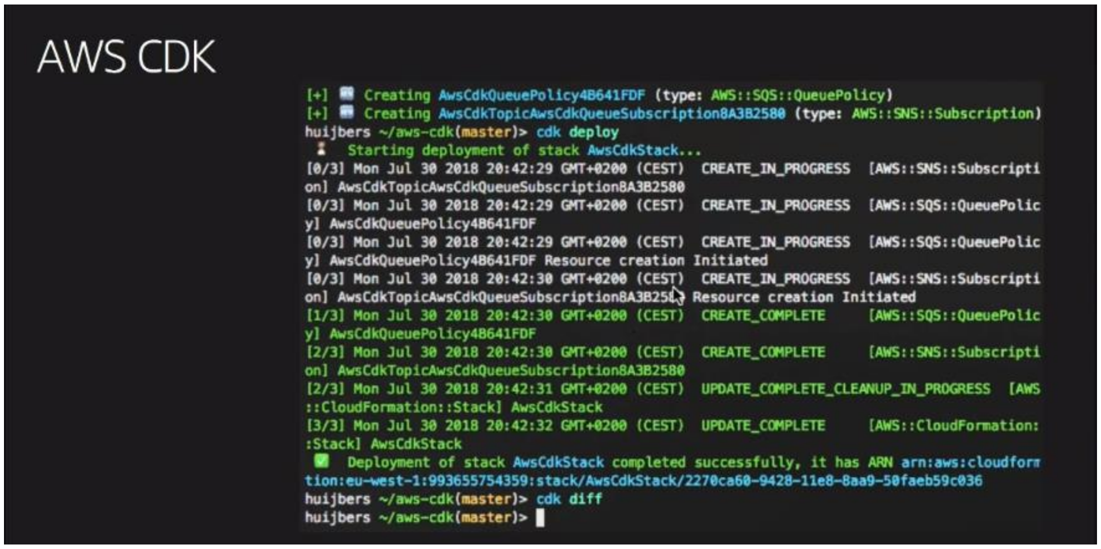
AWS CDK

```
b/aws-cdk.ts                                                                    buffers
#!/usr/bin/env node
import sns = require('@aws-cdk/aws-sns');
import sqs = require('@aws-cdk/aws-sqs');
import cdk = require('@aws-cdk/cdk');

class AwsCdkStack extends cdk.Stack
    constructor(parent: cdk.App, name: string, props?: cdk.StackProps) {
        super(parent, name, props);

        const queue = new sqs.Queue(this, 'AwsCdkQueue', {
            visibilityTimeoutSec: 300
        });

        const topic = new sns.Topic(this, 'AwsCdkTopic');
    }


const app = new cdk.App(process.argv);

new AwsCdkStack(app, 'AwsCdkStack');

 NORMAL  bin/aws-cdk.ts                         typ…  utf-8[unix]   72% ≡   16/22 ln :  1
"bin/aws-cdk.ts" 22L, 579C
```



AWS CDK

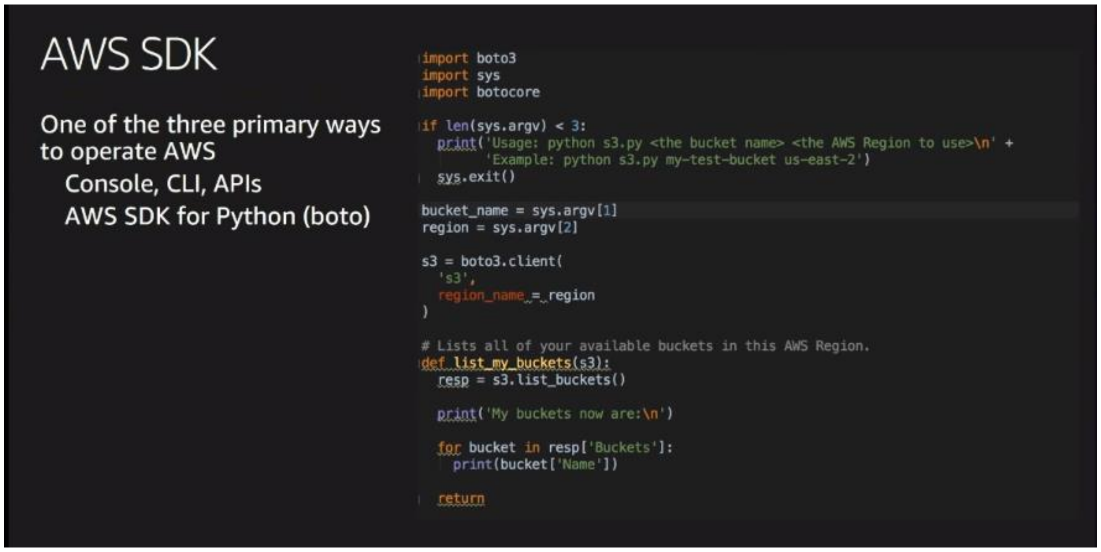
```
[2/5] Mon Jul 30 2018 20:41:30 GMT+0200 (CEST)  CREATE_COMPLETE     [AWS::CDK::Metadata]
CDKMetadata
[3/5] Mon Jul 30 2018 20:41:38 GMT+0200 (CEST)  CREATE_COMPLETE     [AWS::SNS::Topic] Aws
CdkTopicF164620F
[3/5] Mon Jul 30 2018 20:41:40 GMT+0200 (CEST)  UPDATE_COMPLETE_CLEANUP_IN_PROGRESS  [AWS
::CloudFormation::Stack] AwsCdkStack
[3/5] Mon Jul 30 2018 20:41:42 GMT+0200 (CEST)  DELETE_IN_PROGRESS  [AWS::CloudFormation:
:WaitConditionHandle] WaitCondition
[4/5] Mon Jul 30 2018 20:41:42 GMT+0200 (CEST)  DELETE_COMPLETE     [AWS::CloudFormation:
:WaitConditionHandle] WaitCondition
[5/5] Mon Jul 30 2018 20:41:43 GMT+0200 (CEST)  UPDATE_COMPLETE     [AWS::CloudFormation:
:Stack] AwsCdkStack
 ✅  Deployment of stack AwsCdkStack completed successfully, it has ARN arn:aws:cloudform
tion:eu-west-1:993655754359:stack/AwsCdkStack/2270ca60-9428-11e8-8aa9-50faeb59c036
huijbers ~/aws-cdk(master)> vim bin/aws-cdk.ts
huijbers ~/aws-cdk(master)> npm run build

> aws-cdk@0.1.0 build /Users/huijbers/aws-cdk
> tsc

huijbers ~/aws-cdk(master)> cdk diff
[+] 🆕 Creating AwsCdkQueuePolicy4B641FDF (type: AWS::SQS::QueuePolicy)
[+] 🆕 Creating AwsCdkTopicAwsCdkQueueSubscription8A3B2580 (type: AWS::SNS::Subscription)
huijbers ~/aws-cdk(master)> 
```



AWS CDK

```
[+] 🆕 Creating AwsCdkQueuePolicy4B641FDF (type: AWS::SQS::QueuePolicy)
[+] 🆕 Creating AwsCdkTopicAwsCdkQueueSubscription8A3B2580 (type: AWS::SNS::Subscription)
huijbers ~/aws-cdk(master)> cdk deploy
 ⏳  Starting deployment of stack AwsCdkStack...
[0/3] Mon Jul 30 2018 20:42:29 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::SNS::Subscripti
on] AwsCdkTopicAwsCdkQueueSubscription8A3B2580
[0/3] Mon Jul 30 2018 20:42:29 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::SQS::QueuePolic
y] AwsCdkQueuePolicy4B641FDF
[0/3] Mon Jul 30 2018 20:42:29 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::SQS::QueuePolic
y] AwsCdkQueuePolicy4B641FDF Resource creation Initiated
[0/3] Mon Jul 30 2018 20:42:30 GMT+0200 (CEST)  CREATE_IN_PROGRESS  [AWS::SNS::Subscripti
on] AwsCdkTopicAwsCdkQueueSubscription8A3B25 Resource creation Initiated
[1/3] Mon Jul 30 2018 20:42:30 GMT+0200 (CEST)  CREATE_COMPLETE     [AWS::SQS::QueuePolic
y] AwsCdkQueuePolicy4B641FDF
[2/3] Mon Jul 30 2018 20:42:30 GMT+0200 (CEST)  CREATE_COMPLETE     [AWS::SNS::Subscripti
on] AwsCdkTopicAwsCdkQueueSubscription8A3B2580
[2/3] Mon Jul 30 2018 20:42:31 GMT+0200 (CEST)  UPDATE_COMPLETE_CLEANUP_IN_PROGRESS  [AWS
::CloudFormation::Stack] AwsCdkStack
[3/3] Mon Jul 30 2018 20:42:32 GMT+0200 (CEST)  UPDATE_COMPLETE     [AWS::CloudFormation:
:Stack] AwsCdkStack
 ✅  Deployment of stack AwsCdkStack completed successfully, it has ARN arn:aws:cloudform
tion:eu-west-1:993655754359:stack/AwsCdkStack/2270ca60-9428-11e8-8aa9-50faeb59c036
huijbers ~/aws-cdk(master)> cdk diff
huijbers ~/aws-cdk(master)> 
```

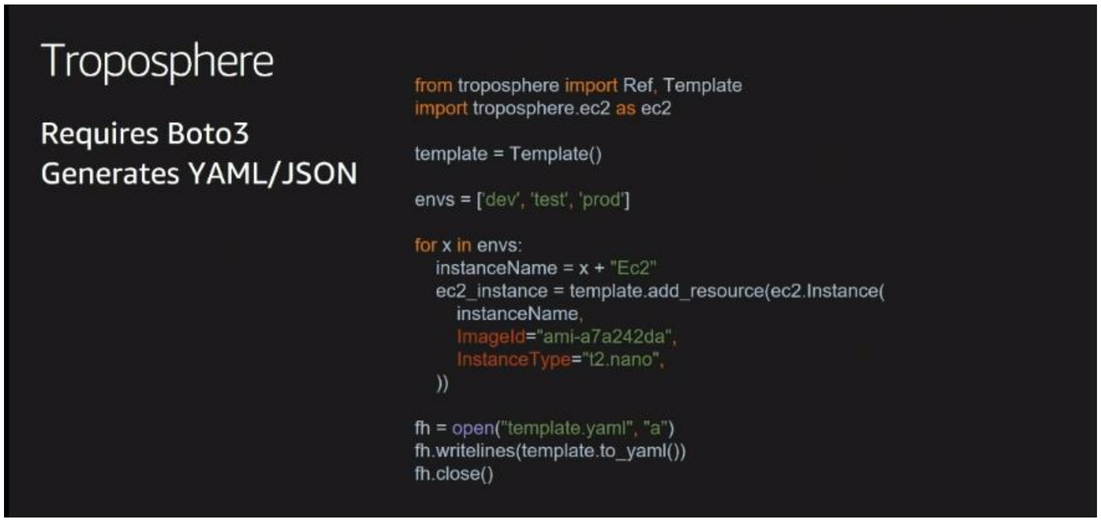
You then deploy the update again to have the stack get updated



AWS SDK

One of the three primary ways to operate AWS
- Console, CLI, APIs
- AWS SDK for Python (boto)

```python
import boto3
import sys
import botocore

if len(sys.argv) < 3:
  print('Usage: python s3.py <the bucket name> <the AWS Region to use>\n' +
        'Example: python s3.py my-test-bucket us-east-2')
  sys.exit()

bucket_name = sys.argv[1]
region = sys.argv[2]

s3 = boto3.client(
  's3',
  region_name = region
)

# Lists all of your available buckets in this AWS Region.
def list_my_buckets(s3):
  resp = s3.list_buckets()

  print('My buckets now are:\n')

  for bucket in resp['Buckets']:
    print(bucket['Name'])

  return
```



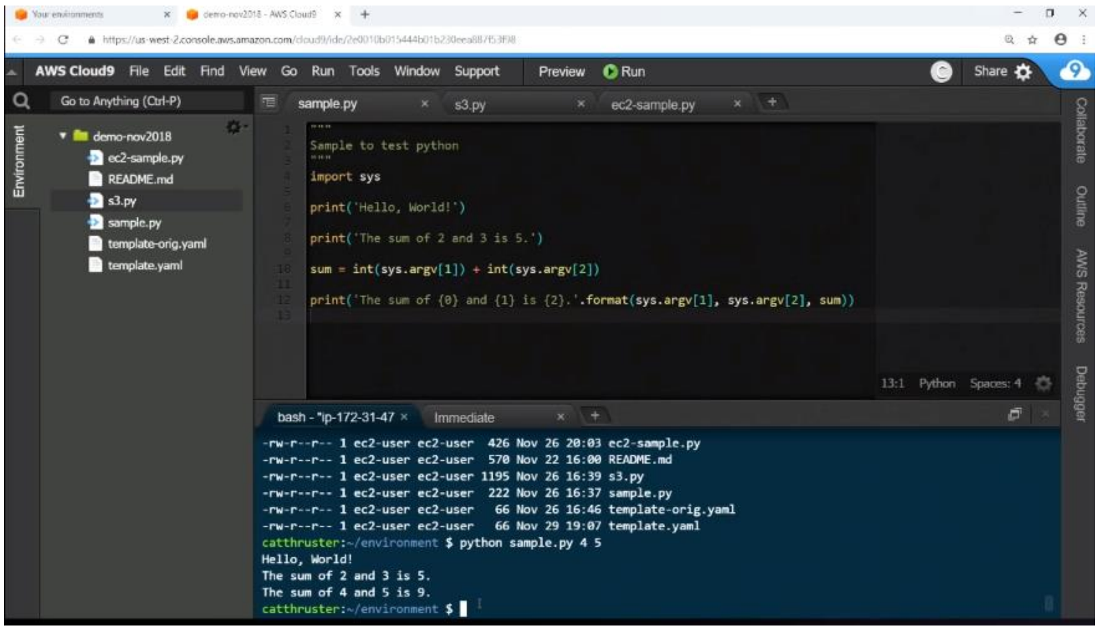
Troposphere

Requires Boto3
Generates YAML/JSON

```python
from troposphere import Ref, Template
import troposphere.ec2 as ec2

template = Template()

envs = ['dev', 'test', 'prod']

for x in envs:
    instanceName = x + "Ec2"
    ec2_instance = template.add_resource(ec2.Instance(
        instanceName,
        ImageId="ami-a7a242da",
        InstanceType="t2.nano",
    ))

fh = open("template.yaml", "a")
fh.writelines(template.to_yaml())
fh.close()
```

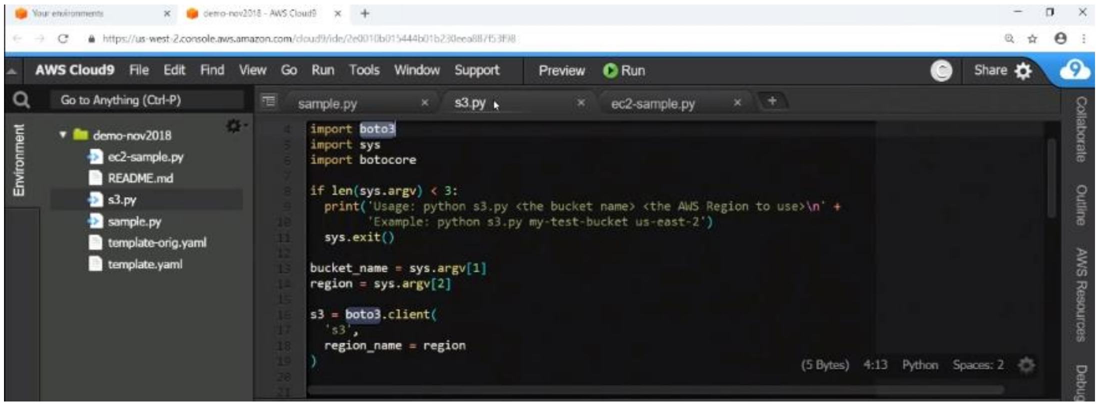
Demo

Let us now see how Troposhere works using the SDK

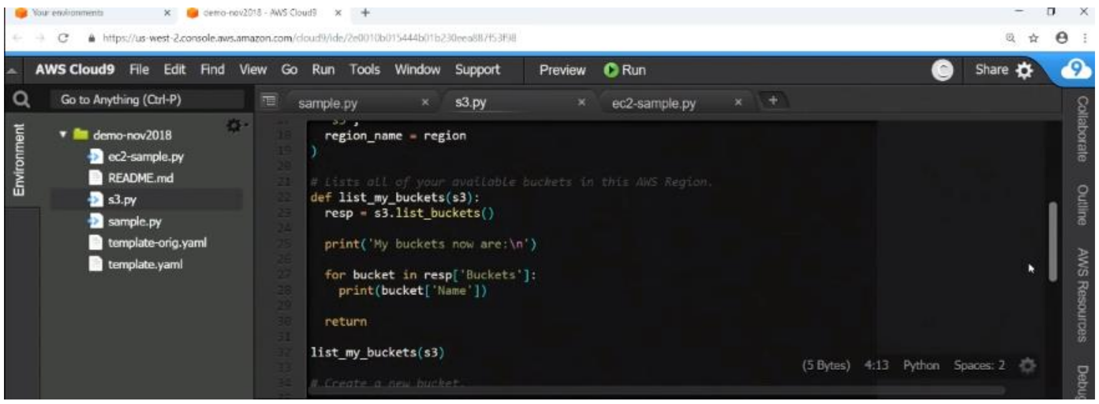


```python
"""
Sample to test python
"""
import sys

print('Hello, World!')

print('The sum of 2 and 3 is 5.')

sum = int(sys.argv[1]) + int(sys.argv[2])

print('The sum of {0} and {1} is {2}.'.format(sys.argv[1], sys.argv[2], sum))
```

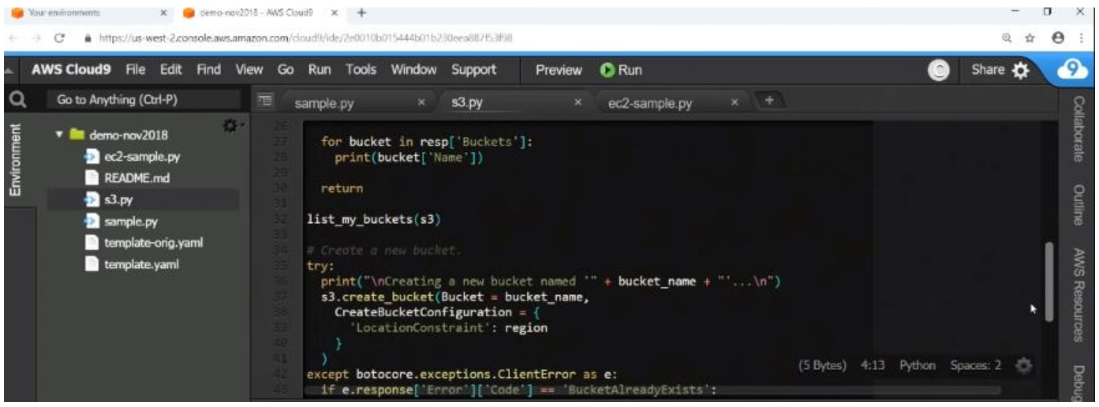
```
-rw-r--r-- 1 ec2-user ec2-user  426 Nov 26 20:03 ec2-sample.py
-rw-r--r-- 1 ec2-user ec2-user  570 Nov 22 16:00 README.md
-rw-r--r-- 1 ec2-user ec2-user 1195 Nov 26 16:39 s3.py
-rw-r--r-- 1 ec2-user ec2-user  222 Nov 26 16:37 sample.py
-rw-r--r-- 1 ec2-user ec2-user   66 Nov 26 16:46 template-orig.yaml
-rw-r--r-- 1 ec2-user ec2-user   66 Nov 29 19:07 template.yaml
catthruster:~/environment $ python sample.py 4 5
Hello, World!
The sum of 2 and 3 is 5.
The sum of 4 and 5 is 9.
catthruster:~/environment $
```



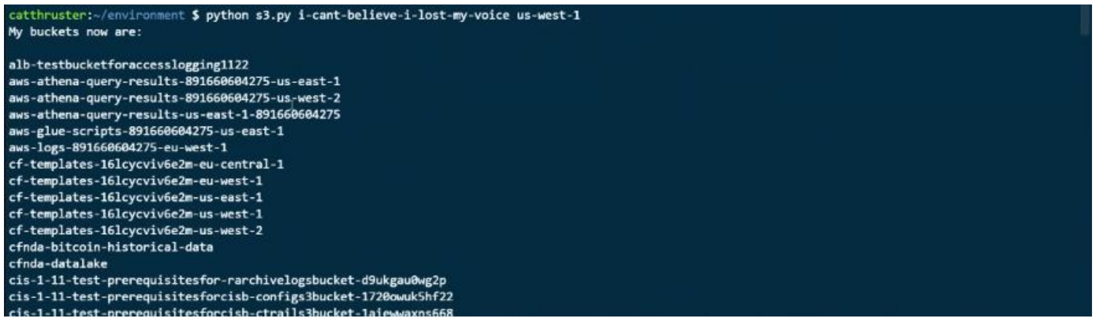
```python
import boto3
import sys
import botocore

if len(sys.argv) < 3:
  print('Usage: python s3.py <the bucket name> <the AWS Region to use>\n' +
        'Example: python s3.py my-test-bucket us-east-2')
  sys.exit()

bucket_name = sys.argv[1]
region = sys.argv[2]

s3 = boto3.client(
  's3',
  region_name = region
)
```



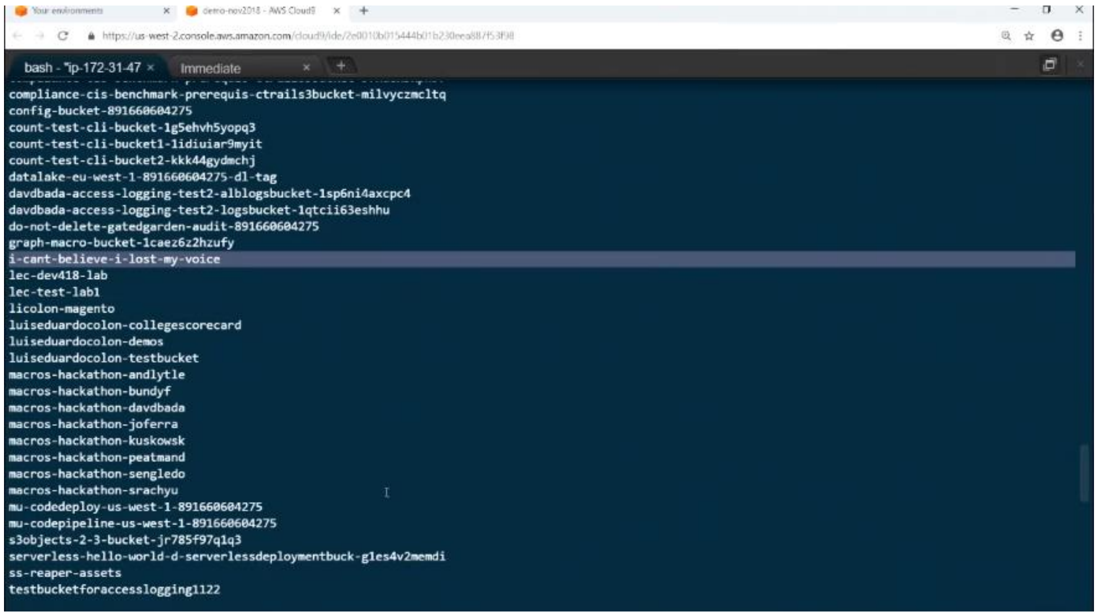
```python
  region_name = region
)

# Lists all of your available buckets in this AWS Region.
def list_my_buckets(s3):
  resp = s3.list_buckets()

  print('My buckets now are:\n')

  for bucket in resp['Buckets']:
    print(bucket['Name'])

  return

list_my_buckets(s3)

# Create a new bucket.
```



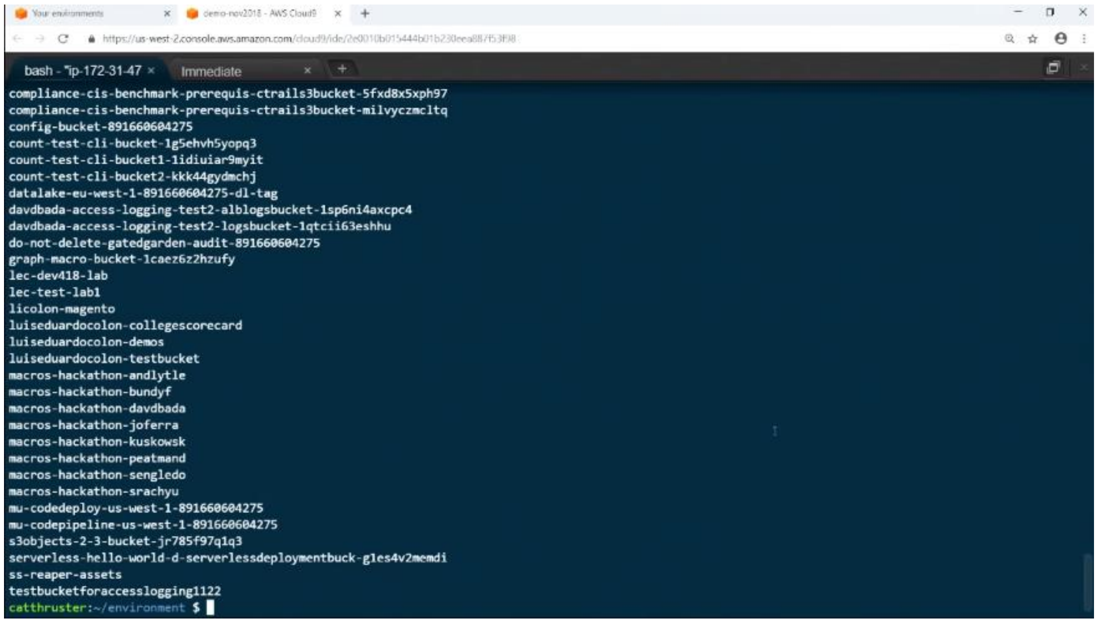
```python
  for bucket in resp['Buckets']:
    print(bucket['Name'])

  return

list_my_buckets(s3)

# Create a new bucket.
try:
  print("\nCreating a new bucket named '" + bucket_name + "'...\n")
  s3.create_bucket(Bucket = bucket_name,
    CreateBucketConfiguration = {
      'LocationConstraint': region
    }
  )
except botocore.exceptions.ClientError as e:
  if e.response['Error']['Code'] == 'BucketAlreadyExists':
```



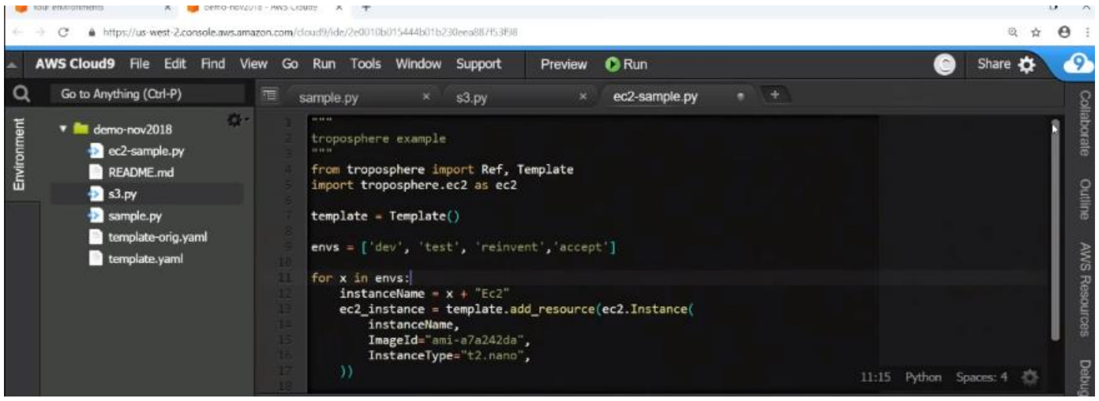
```
catthruster:~/environment $ python s3.py i-cant-believe-i-lost-my-voice us-west-1
My buckets now are:

alb-testbucketforaccesslogging1122
aws-athena-query-results-891660604275-us-east-1
aws-athena-query-results-891660604275-us-west-2
aws-athena-query-results-us-east-1-891660604275
aws-glue-scripts-891660604275-us-east-1
aws-logs-891660604275-eu-west-1
cf-templates-16lcycviv6e2m-eu-central-1
cf-templates-16lcycviv6e2m-eu-west-1
cf-templates-16lcycviv6e2m-us-east-1
cf-templates-16lcycviv6e2m-us-west-1
cf-templates-16lcycviv6e2m-us-west-2
cfnda-bitcoin-historical-data
cfnda-datalake
cis-1-11-test-prerequisitesfor-rarchivelogsbucket-d9ukgau0wg2p
cis-1-11-test-prerequisitesforcisb-configs3bucket-1720owuk5hf22
cis-1-11-test-prerequisitesforcisb-ctrails3bucket-1ajewwaxps668
```


Your environments
demo-nov2018 - AWS Cloud9
https://us-west-2.console.aws.amazon.com/cloud9/ide/2e0010b015444b01b230eea887f53f98
bash - "ip-172-31-47
Immediate

```
compliance-cis-benchmark-prerequis-ctrails3bucket-milvyczmcltq
config-bucket-891660604275
count-test-cli-bucket-1g5ehvh5yopq3
count-test-cli-bucket1-1idiuiar9myit
count-test-cli-bucket2-kkk44gydmchj
datalake-eu-west-1-891660604275-dl-tag
davdbada-access-logging-test2-alblogsbucket-1sp6ni4axcpc4
davdbada-access-logging-test2-logsbucket-1qtcii63eshhu
do-not-delete-gatedgarden-audit-891660604275
graph-macro-bucket-1caez6z2hzufy
i-cant-believe-i-lost-my-voice
lec-dev418-lab
lec-test-lab1
licolon-magento
luiseduardocolon-collegescorecard
luiseduardocolon-demos
luiseduardocolon-testbucket
macros-hackathon-andlytle
macros-hackathon-bundyf
macros-hackathon-davdbada
macros-hackathon-joferra
macros-hackathon-kuskowsk
macros-hackathon-peatmand
macros-hackathon-sengledo
macros-hackathon-srachyu
mu-codedeploy-us-west-1-891660604275
mu-codepipeline-us-west-1-891660604275
s3objects-2-3-bucket-jr785f97q1q3
serverless-hello-world-d-serverlessdeploymentbuck-g1es4v2memdi
ss-reaper-assets
testbucketforaccesslogging1122
```

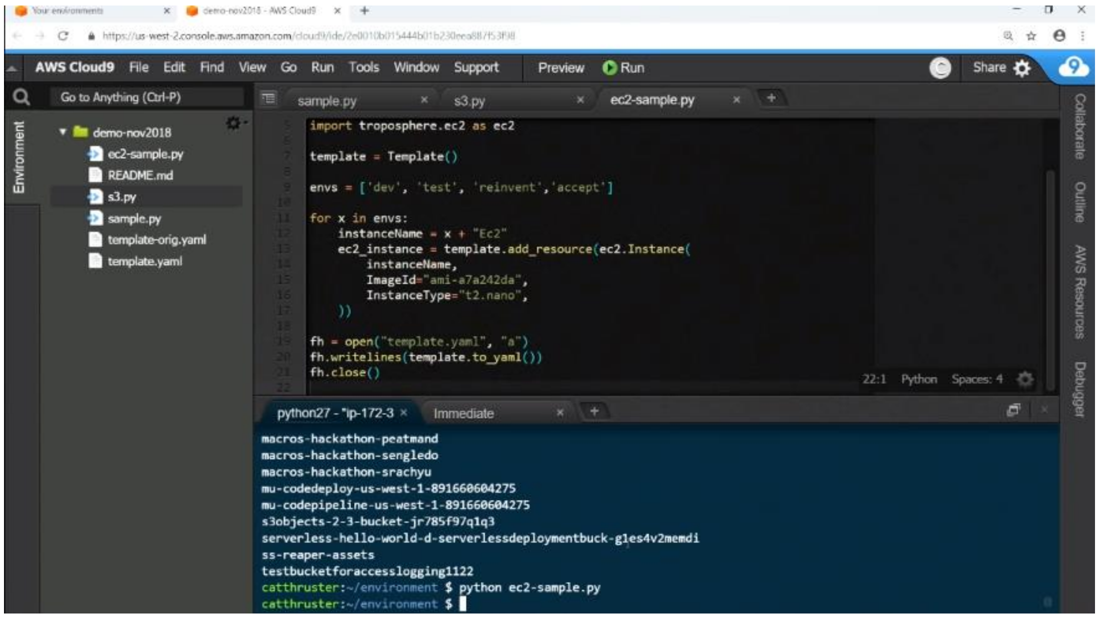

Your environments
demo-nov2018 - AWS Cloud9
https://us-west-2.console.aws.amazon.com/cloud9/ide/2e0010b015444b01b230eea887f53f98
bash - "ip-172-31-47
Immediate

```
compliance-cis-benchmark-prerequis-ctrails3bucket-5fxd8x5xph97
compliance-cis-benchmark-prerequis-ctrails3bucket-milvyczmcltq
config-bucket-891660604275
count-test-cli-bucket-1g5ehvh5yopq3
count-test-cli-bucket1-1idiuiar9myit
count-test-cli-bucket2-kkk44gydmchj
datalake-eu-west-1-891660604275-dl-tag
davdbada-access-logging-test2-alblogsbucket-1sp6ni4axcpc4
davdbada-access-logging-test2-logsbucket-1qtcii63eshhu
do-not-delete-gatedgarden-audit-891660604275
graph-macro-bucket-1caez6z2hzufy
lec-dev418-lab
lec-test-lab1
licolon-magento
luiseduardocolon-collegescorecard
luiseduardocolon-demos
luiseduardocolon-testbucket
macros-hackathon-andlytle
macros-hackathon-bundyf
macros-hackathon-davdbada
macros-hackathon-joferra
macros-hackathon-kuskowsk
macros-hackathon-peatmand
macros-hackathon-sengledo
macros-hackathon-srachyu
mu-codedeploy-us-west-1-891660604275
mu-codepipeline-us-west-1-891660604275
s3objects-2-3-bucket-jr785f97q1q3
serverless-hello-world-d-serverlessdeploymentbuck-g1es4v2memdi
ss-reaper-assets
testbucketforaccesslogging1122
catthruster:~/environment $
```



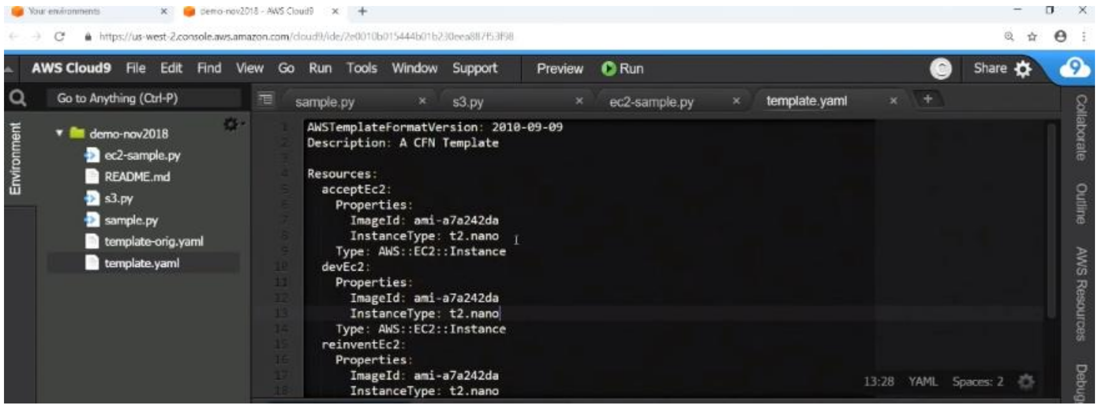
```
"""
troposphere example
"""

from troposphere import Ref, Template
import troposphere.ec2 as ec2

template = Template()

envs = ['dev', 'test', 'reinvent','accept']

for x in envs:
    instanceName = x + "Ec2"
    ec2_instance = template.add_resource(ec2.Instance(
        instanceName,
        ImageId="ami-a7a242da",
        InstanceType="t2.nano",
    ))
```

This code uses troposphere to generate an iterator that generates EC2 instances



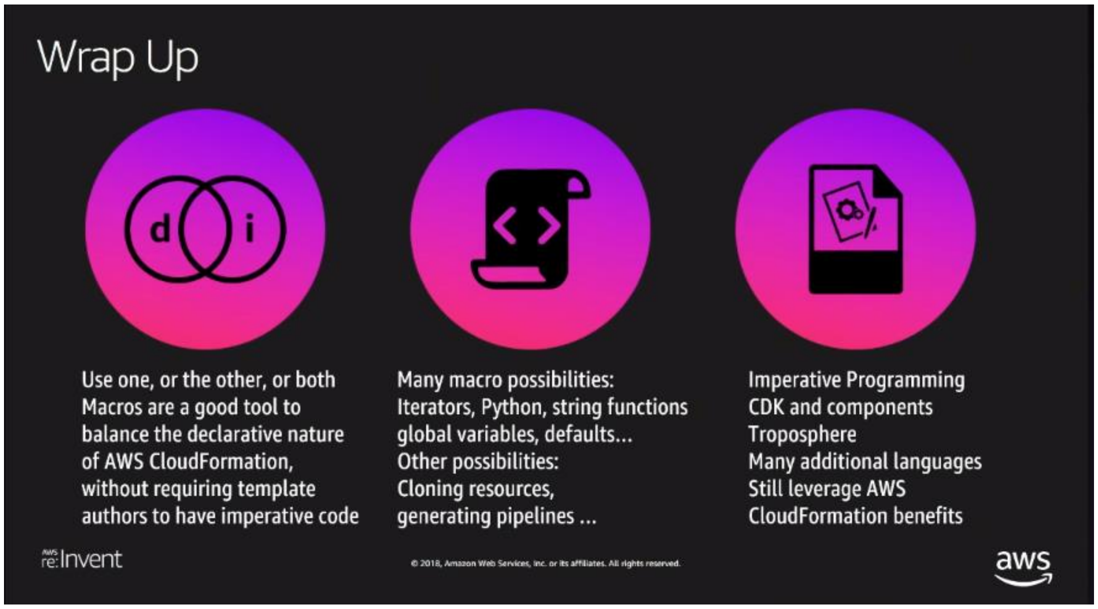
```
import troposphere.ec2 as ec2

template = Template()

envs = ['dev', 'test', 'reinvent','accept']

for x in envs:
    instanceName = x + "Ec2"
    ec2_instance = template.add_resource(ec2.Instance(
        instanceName,
        ImageId="ami-a7a242da",
        InstanceType="t2.nano",
    ))

fh = open("template.yaml", "a")
fh.writelines(template.to_yaml())
fh.close()
```

```
macros-hackathon-peatmand
macros-hackathon-sengledo
macros-hackathon-srachyu
mu-codedeploy-us-west-1-891660604275
mu-codepipeline-us-west-1-891660604275
s3objects-2-3-bucket-jr785f97q1q3
serverless-hello-world-d-serverlessdeploymentbuck-g1es4v2memdi
ss-reaper-assets
testbucketforaccesslogging1122
catthruster:~/environment $ python ec2-sample.py
catthruster:~/environment $
```



```
AWSTemplateFormatVersion: 2010-09-09
Description: A CFN Template

Resources:
  acceptEc2:
    Properties:
      ImageId: ami-a7a242da
      InstanceType: t2.nano
    Type: AWS::EC2::Instance
  devEc2:
    Properties:
      ImageId: ami-a7a242da
      InstanceType: t2.nano
    Type: AWS::EC2::Instance
  reinventEc2:
    Properties:
      ImageId: ami-a7a242da
      InstanceType: t2.nano
```

This is the generated YAML file that was generated using the code, this can then be used to generate your stack in CF. we can use this code approach to do things like run unit tests on your stack creation code.



Wrap Up

Use one, or the other, or both
Macros are a good tool to balance the declarative nature of AWS CloudFormation, without requiring template authors to have imperative code

Many macro possibilities:
Iterators, Python, string functions global variables, defaults…
Other possibilities:
Cloning resources, generating pipelines …

Imperative Programming
CDK and components
Troposphere
Many additional languages
Still leverage AWS CloudFormation benefits

Thank you!

Luis Colon
Senior Developer Advocate
AWS CloudFormation

Chuck Meyer
Senior Developer Advocate
AWS CloudFormation