Developing a Self Destructing Cloudformation Stack

2018-10-22T22:25:47+03:00

Contents

In this tutorial, we will develop a self destructing cloudformation stack. The stack can ben named *harakiri* since it will delete not only the resources created but also itself after a specified duration. The stack is best suited for timely

For example, a potential customer wants to try your SaaS before paid subscription and you do not want to be charged for the unused resources.

It would be better have a basic knowledge about CloudFormation.

> AWS Cloud Formation is a infrastructure as a code that creates, updates and deletes resources with JSON or YAML configuration files.

Defining Stack

AWS Cloud Formation template has five sections:

- Mapping
- Parameters
- Conditions
- Resources
- Outputs

In the resources section we define the AWS Resources that will be created. I will keep it simple and focus on only the resources section.

Trivial Section

{{< highlight yaml "linenos=table,hl_lines=,linenostart=1" >}} AWSTemplateFormatVersion: "2010-09-09"

Mappings: RegionMap: us-east-1: AMI: "ami-0ff8a91507f77f867"

Parameters: {}

Conditions: {} {{< / highlight >}}

There is nothing fancy here. Although we are not going to create and EC2 instance, at least one region (the region you will be creating the stack) is required. Also AMI (Amazon Machine Image) can be wrong. Just skip it.

IAM Role for The Lambda

We use lambda for triggering the stack delete process. This IAM role must be priviled to delete all resources created with stack. You would like give full privilege this role but be safe by giving fine grained privilege by giving privilege to this role for only the resources created with stack.

We gave privilege to this role:

- Deleting all cloud formation stacks
- Deleting all IAM roles. (This role will able to delete itself)
- Deleting all lambda functions
- Deleting all cloudwatch events and rules

{{< highlight yaml "linenos=table,hl_lines=,linenostart=11" >}} Resources: LambdaExecutionRole: Type: AWS::IAM::Role Properties: AssumeRolePolicyDocument: Version: '2012-10-17' Statement: - Effect: Allow Principal: Service: - lambda.amazonaws.com Action: - sts:AssumeRole Path: "/" Policies: - PolicyName: root PolicyDocument: Version: '2012-10-17' Statement: - Effect: Allow Action: - logs: *Resource: arn:aws:logs:::*

```
      - Effect: Allow
        Action:
          - cloudformation:DeleteStack
        Resource: "*"

      - Effect: Allow
        Action:
```

```
     - iam:DeleteRolePolicy
     - iam:DeleteRole
    Resource: "*"

   - Effect: Allow
    Action:
     - lambda:DeleteFunction
    Resource: "*"

   - Effect: Allow
    Action:
     - events:RemoveTargets
    Resource: "*"

   - Effect: Allow
    Action:
     - events:DeleteRule
    Resource: "*"

   - Effect: Allow
    Action:
     - lambda:RemovePermission
    Resource: "*"
```

{{< / highlight >}}

Lambda Function

This lambda function will start stack delete process.

- How does this lambda function trigger?
- Cloud Watch Rules.

We will define a cloud watch timer later. The cloud watch timer will trigger every X times but since the lambda function will delete all resources created with stack, probably there would be no second execution.

> Boto is a Python package that provides interfaces to Amazon Web Services.

{{< highlight python "linenos=table,hl_lines=,linenostart=1" >}} import boto3 client = boto3.client('cloudformation')

def handler(event, context): return client.delete_stack(StackName=event.StackName) {{< / highlight >}}

Source code of lambda function is simple. It requires a simple JSON object with a `StackName` property which is the name of the Cloud Formation stack.

{{< highlight yaml "linenos=table,hl_lines=,linenostart=66" >}} HarakiriLambda: Type: AWS::Lambda::Function Properties: Handler: index.handler Role: !GetAtt LambdaExecutionRole.Arn Code: ZipFile: | import boto3 client = boto3.client('cloudformation')

```
      def handler(event, context):
        return client.delete_stack(
            StackName=event.StackName
        )
```

```
  Runtime: python3.6
```

PermissionForEventsToInvokeLambda: Type: AWS::Lambda::Permission Properties: FunctionName: Ref: "HarakiriLambda" Action: "lambda:InvokeFunction" Principal: "events.amazonaws.com" SourceArn: Fn::GetAtt: - "HarakiriRule" - "Arn"
{{< / highlight >}}

CloudWatch Timer (Rule)

We will define a cloud watch cron rule that will be executed every thirty minutes but we expect it does not executed twice about five or ten minutes after the first execution, all resources including this rule will be deleted.

We prepare a simple JSON object at line 107, for the lambda by substituting the stack name.

{{< highlight yaml "linenos=table,hl_lines=,linenostart=93" >}}

HarakiriRule: Type: AWS::Events::Rule Properties: Description: "ScheduledRule" ScheduleExpression: "cron(0/30 * * * ? *)" State: "ENABLED" Targets: - Arn: Fn::GetAtt: - "HarakiriLambda" - "Arn" Id: "HarakiriLambdaV1" Input: !Sub - "{"StackName": "${Stack}"}" - { Stack: !Ref "AWS::StackName" }

{{< / highlight >}}

Final

- Go to AWS Cloud Formation Console: console.aws.amazon.com/cloudformation
- Download the full template from template.yml and change the 30 minutes section to 2-3 minutes.

- Click "Create Stack" button and upload the template.
- Hit the "Next" button and give a name to your stack.
- Watch the events tabs at the bottom.

You will see the stack create process, just after 2-3 minutes, "stack delete in progress" event will appear.

From the CLI you can create the stack by typing:

```
aws cloudformation deploy \
        --stack-name myteststack \
        --capabilities CAPABILITY_IAM \
--template-file template.yml
```

Conclusion

I hope you enjoyed and learnt a lot.

Happy coding.