Project Documentation

### PROJECT DESCRIPTION:

### The project implements function to check web health metric of Urls provided and Publish those Metrics to Cloud watch and after publishing Create Alarms on those metric values crossing certain threshold values, and after that receive notification from an SNS to email address subscribed and store the data in DynamoDB database on cloud9 using aws\_cdk to build cloud infrastructure with programming.

### WHAT IS AWS\_CDK?

The AWS Cloud Development Kit (AWS CDK) is an open-source software development framework to define your cloud application resources using familiar programming languages.

[[Reference]](https://aws.amazon.com/cdk/) [[Documentation]](https://docs.aws.amazon.com/cdk/api/latest/python/index.html)

### OVERVIEW

An Application has been build using Aws\_cdk and lambda function has been written to monitor Latencies and Availabilities of URLs passed in URL array which are accessed using S3 bucket and Json repose file contained Urls . And For those Urls Cloud watch matrices has been created based on latency and availability values for each URL and alarms has been set on these metrices for specific Threshold values. For each alarm sns topic has been created that sends notification to a lambda function with the payload of message when alarm exceed threshold value. This lambda function has access to DynamoDB to write the attributes of message received in sns notifications. All constant values has been defined in separate file constants.py and above functionality is implemented in aws\_cdk tool in using Python Language.

# PROJECT MOTIVATION

Infrastructure as code has quickly become a go-to process to automatically provision and manage cloud resources. With increasing sophistication, engineers and DevOps teams are codifying infrastructure for greater application flexibility and functionality, with a single-source language across an organization.

IT teams have two AWS-native options for infrastructure as code -- AWS CloudFormation and the AWS Cloud Development Kit (CDK). CloudFormation templates were AWS' first foray into cloud-based infrastructure as code, and while still useful, CloudFormation has clear weaknesses. More specifically, it doesn't offer built-in logic capabilities and has a steep learning curve.

The AWS CDK, an open-source software development framework to define cloud infrastructure, addresses these weaknesses. The AWS CDK supports popular programming languages, which developers can use to build, automate and manage infrastructure based on an imperative approach. Finally, developers can provision these commands through CloudFormation.

As an extensible, open software development framework, the AWS CDK features integrated development environment (IDE) capabilities. As of publication, the AWS CDK supports TypeScript, JavaScript, Python, Java and C#/.Net. In this article, we'll compare the AWS CDK vs. CloudFormation, including their key features, the role of constructs in building application stacks and the benefits of using a common language for AWS-native infrastructure as code.

### PROJECT REQUIREMENTS

### FUNCTIONAL REQUIREMENTS

* + - Project should be implemented using AWS Cloud 9 Environment.
    - Project should be implemented in python
    - Project should be implemented using Lambda function from awd\_cdk
    - Lambda function should check Health of url passed
    - Lambda function must run every 1 minute to monitor website
    - Create metrics on cloud watch for each latency and availability data point for each url
    - Set and alarm on CloudWatch metrics
    - Enable Notification on email
    - Enable sns notification service on Lambda
    - DataBase must be created using DyanoDB.
    - Lambda must write the sns event timestamp and message in DynamoDB

### NON-FUNCTIONAL REQUIREMENTS

* + - Comments must be added in the code
    - Code must be structured well
    - Modular approach should be used
    - Unnecessary global variables must not be used
    - Best practices must be followed

### DETAIL DESIGN AND ARCHITECTURE

### FLOW DIAGRAM



1. Lambda sent metric data to Cloud Watch
2. Sns take message from Cloud watch if Alarm triggers
3. Sns send notification to lambda with message as a payload
4. Lambda save message in DynamoDB

### 

### SETUP ENVIRONMENT

### INSTALLATIONS

As this project use cloud9 to create our cloud infrastructure. It has everything pre-installed so there is no need to install anything on local machine

### STEPS TO FOLLOW

* + - Login to AWS Management Console[[Login]](https://us-east-2.console.aws.amazon.com/console/home?region=us-east-2)
    - Create an instance in Cloud9
    - Set python 3 in bashrc file as python alias
    - Start virtual environment
    - Install requirements with requirements.txt
    - Create cloud template
    - Deploy

Commands

* To open bashrc file

*vim ~/.bashrc*

Add this code in end

*alias python="/usr/bin/python3"*

* To start virtual environment

*source .venv/bin/activate*

* install requirements

*pip install -r requirements.txt*

* Synthesize

*cdk synth*

* Deploy

*cdk deploy*

Explanation Code Files:

* cloudwatchmatric.py
* constants.py
* Dynamo\_db.py
* DynamoLambda.py
* PreiodicLambda.py
* s3\_bucket.py

**cloudwatchmatric.py:**

This file contains Cloud watch boto3 client and also this file has the function of putting data onto the cloud watch as that data is provided by Periodic Lambda.py.

**PreiodicLambda.py:**

This Lambda File contains the overall functionality to check health status of a Url and then report it back so that those values can be published on a periodic basis of 1 minute.

**Dynamo\_db.py**

This file is sued to put data into database table named ‘Ali\_Alarm\_table’.

**DynamoLambda.py**

This file is used to to Define and create a non existing table into database.

**s3\_bucket.py**

This file is used to connect s3 bucket resource to Stack and access files uploaded to it and then sue accordingly.

SUCCESSFUL RUN

* Successfully Extracted Url Health.
* Successfully Published Periodic Health on Cloudwatch.
* Successfully Populated Metrics of cloudwatch back into Infra.
* Successfully Put Alarms Considering Thresh hold value on Metrics.
* Successfully Sent Email Notifications.

# DIFFICULTIES FACED

### DYNAMODB READ WRITE ACCESS

I was having access denied issue from dynamo db read write. I solved the issue by adding dynamodb full access policy in create role function

### SNS NOTIFCATION ERROR

lambda was unable to receive Event Data from sns

References <https://aws.amazon.com/cdk/> <https://aws.amazon.com/cloud9/> <https://aws.amazon.com/lambda/>

[https://aws.amazon.com/sns/?whats-new-cards.sort-](https://aws.amazon.com/sns/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc) [by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc](https://aws.amazon.com/sns/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc)