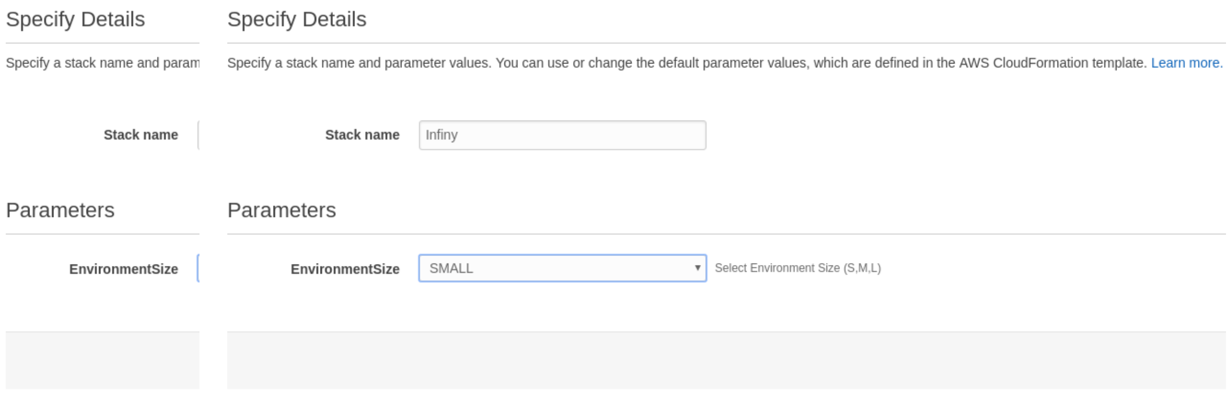
To Infiny and Beyond Blog exercise 2

This is a further development of a coding exercise with the goal to automatically publish a Flask webapp within AWS. The first outing achieved this goal purely from the command line and via a bootstrap script. In this version, I have employed AWS CloudFormation to leverage CI/CD functionality.

In this version, a Flask webapp server can be spun up quickly from the command line via the script 'new-flask-host.sh' or the CloudFormation JSON script can be executed directly within CloudFormation to allow the user to select particular settings for their deployment.

Figure 1 below illustrates how EC2 Instance size can be selected from a drop-down menu. This restricts selections to valid choices while also giving the administrator the ability to limit the allowable EC2 Instance types. In this case; SMALL = t2.micro, MEDIUM = t2.small, LARGE = t2.medium.

Figure 1 – CloudFormation screen capture



Specify Details

Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. Learn more.

Stack name: Infiny

Parameters

EnvironmentSize: SMALL — Select Environment Size (S,M,L)

Source Code

The source code for this solution is split into two separate entities: client and server and is stored entirely in GitHub here:

Client source: https://github.com/nickholbrook/Infiny2.git

Server source: https://github.com/nickholbrook/ToInfinyAndBeyond-server.git

Client Setup

To simplify user configuration, I have created a setup script called 'setup.sh' which pulls down all client files from Github (a repository of files) to the user's workstation placing them in ~/Infiny02/. The last action of the script displays usage instructions.

Usage

To create a new Flask webapp server: new-flask-host

To erase the Flask webapp server: del-flask-host

Further Deployment

Using CloudFormation offers the ability to completely remove a CloudFormation Stack which erases all components that constitutes the Stack or to selectively delete elements. This could be beneficial for solutions that utilise databases as part of a system but the data must be retained through iterations. In these scenarios the entire Stack could be erased but the database could be retained (CFN deletion policy).