# Introduction

Implement a Dockerized Webservice to generate a cryptographic hash/digest for a supplied message and retrieve a message based on the hash.

# Solution Approach

The required webservice (hashdehash) is implemented using Spring Boot framework. Two REST request handlers are implemented i.e. one for POST and other for GET.

The POST request handler takes a message as input and calculates the sha256 digest in hex format. It stores this digest along with the message in a local cache (implemented as a java hashmap).

The GET request handler takes a digest as input and retrieves the associated message from the local cache and sends that as a response. If no message is found then it sends back a 404 error.

The source code is available in the paxos/source folder

The Dockerfile uses a base alpine image with JDK 1.8. The cert keystore is copied into the image for use by the spring boot app.

The cert was created by using **openssl** utility and then exported to the pcks12 keystore as required by the embedded tomcat in spring boot.

# Installation

Please execute the following steps in the specified order (Note: instructions are for an OSX or Linux environment)

1. Ensure that all the pre-requisites are present i.e
   1. python 2.7
   2. pip
   3. import “requests” module using pip
   4. Docker 17.x-ce
2. Open a terminal and explode the paxos.tar in a suitable location (tar –xvf paxos.tar).
3. Change to the paxos folder by typing **cd paxos** in the terminal
4. Execute the script run script by typing ./run.sh in the terminal (Thus will bring up the docker container and execute the python test script). I have currently set a delay of 5s for the container app to completely startup within the container. Pls tweak it as required based on your env)

# Scaling Options

I have implemented the desired functionality using a simple local cache. However, this will need to be upgraded to make it enterprise grade. I have included a power point (see paxos/doc/Scaling-Options.pptx) presentation where I have tried to present 2 options to extend this service with other components to improve the scale, availability and reliability across all tiers.