Technical Tests v1

This is an ANZ test to be taken as an individual, please do not collaborate when completing the assessment.

Test 1

The following test will require you to do the following:

- Convert the current Dockerfile into a multistage build Dockerfile.
- Optimise Dockerfile for caching benefits.
- Import provided cacert
- Ensure the Java truststore is in the final image.

The current Dockerfile stores all the required certificates into the image prior to adding them to the Java trust store. The certificates were not removed however, and are not needed after they have been added to the truststore. There are many benefits to using multistage build techniques when creating Dockerfiles, such as mitigating security risk, this is accomplished because the attack surface size of the image can be greatly reduced when the image no longer contains unnecessary files, packages, or binaries. It can also enhance caching on layers in previous build steps that no longer need to clustered in single RUN statements for optimal layering because the image is discarded and only those artifacts necessary are kept.

Original Dockerfile

Note: To find more information around the concept of multi-stage docker builds see the following link: https://docs.docker.com/develop/develop-images/multistage-build/

Dockerfile

```
FROM openjdk:9-jdk-slim

COPY certificates /usr/local/share/ca-certificates/certificates

RUN apt-get update && apt-get install --no-install-recommends -y -qq ca-certificates-java && \
         update-ca-certificates


RUN groupadd --gid 1000 java &&\
    useradd --uid 1000 --gid java --shell /bin/bash --create-home java && \
    chmod -R a+w /home/java

WORKDIR /home/java
```

Provided CACERT

cacert

```
-----BEGIN CERTIFICATE-----
MIIEvTCCA6WgAwIBAgIBADANBgkqhkiG9w0BAQUFADB/MQswCQYDVQQGEwJFVTEn
MCUGA1UEChMeQUMgQ2FtZXJmaXJtYSBTQSBDSUYgQTgyNzQzMjg3MSMwIQYDVQQL
ExpodHRwOi8vd3d3LmNoYW1iZXJzaWduLm9yZzEiMCAGA1UEAxMZQ2hhbWJlcnMg
b2YgQ29tbWVyY2UgUm9vdDAeFw0wMzA5MzAxNjEzNDNaFw0zNzA5MzAxNjEzNDRa
MH8xCzAJBgNVBAYTAkVVMScwJQYDVQQKEx5BQyBDYW1lcmZpcm1hIFNBIENJRiBB
ODI3NDMyODcxIzAhBgNVBAsTGmh0dHA6Ly93d3cuY2hhbWJlcnNpZ24ub3JnMSIw
IAYDVQQDExlDaGFtYmVycyBvZiBDb21tZXJjZSBSb290MIIBIDANBgkqhkiG9w0B
AQEFAAOCAQ0AMIIBCAKCAQEAtzZV5aVdGDDg2olUkfzIx1L4L1DZ77F1c2VHfRtb
unXF/KGIJPov7coISjlUxFF6tdpg6jg8gbLL8bvZkSM/SAFwdakFKq0fcfPJVD0d
BmpAPrMMhe5cG3nCYsS4No41XQEMIwRHNaqbYE6gZj3LJgqcQKH0XZi/caulAGgq
7YN6D6IUtdQis4CwPAxaUWktWBiP7Zme8a7ileb2R6jWDA+wWFjbw2Y3npuRVDM3
0pQcakjJyfKl2qUMI/cjDpwyVV5xnIQFUZot/eZOKjRa3spAN2cMVCFVd9oKDMyX
roDclDZK9D7ONhMeU+SsTjoF7Nuucpw4i9A5O4kKPnf+dQIBA6OCAUQwggFAMBIG
A1UdEwEB/wQIMAYBAf8CAQwwPAYDVR0fBDUwMzAxoC+gLYYraHR0cDovL2NybC5j
aGFtYmVyc2lnbi5vcmcvY2hhbWJlcnNyb290LmNybDAdBgNVHQ4EFgQU45T1sU3p
26EpW1eLTXYGduHRooowDgYDVR0PAQH/BAQDAgEGMBEGCWCGSAGG+EIBAQQEAwIA
BzAnBgNVHREEIDAegRxjaGFtYmVyc3Jvb3RAY2hhbWJlcnNpZ24ub3JnMCcGA1Ud
EgQgMB6BHGNoYW1iZXJzcm9vdEBjaGFtYmVyc2lnbi5vcmcwWAYDVR0gBFEwTzBN
BgsrBgEEAYGHLgoDATA+MDwGCCsGAQUFBwIBFjBodHRwOi8vY3BzLmNoYW1iZXJz
aWduLm9yZy9jcHMvY2hhbWJlcnNyb290Lmh0bWwwDQYJKoZIhvcNAQEFBQADggEB
AAxBl8IahsAifJ/7kPMa0QOx7xP5IV8EnNrJpY0nbJaHkb5BkAFyk+cefV/2icZd
p0AJPaxJRUXcLo0waLIJuvvDL8y6C98/d3tGfToSJI6WjzwFCm/SlCgdbQzALogi
1djPHRPH8EjX1wWnz8dHnjs8NMiAT9QUu/wNUPf6s+xCX6ndbcj0dc97wXImsQEc
XCz9ek60AcUFV7nnPKoF2YjpB0ZBzu9Bga5Y34OirsrXdx/nADydb47kMgkdTXg0
eDQ8lJsm7U9xxhl6vSAiSFr+S30Dt+dYvsYyTnQeaN2oaFuzPu5ifdmA6Ap1erfu
tGWaIZDgqtCYvDi1czyL+Nw=
-----END CERTIFICATE-----
```

Test 2

The following test will require you to do the following:

- Create a simple application which has a single "/status" endpoint.
- Containerise your application as a single deployable artifact, encapsulating all dependencies.
- Create a CI pipeline for your application

The application can be written in any programming language. We'd recommend using one the following: NodeJS or GoLang.

Please indicate your preferred programming language.

The application should be a simple, small, operable web-style API or service provider. It should implement the following:

- An endpoint which returns basic information about your application in JSON format which is dynamically generated; The following is expected:
  - Applications Version.
  - Last Commit SHA.
  - Description. (This can be hard-coded)

API Example Response

```
"myapplication": [
  {
    "version": "1.0",
    "lastcommitsha": "abc57858585",
    "description" : "pre-interview technical test"
  }
]
```

The application should have a CI pipeline that is executed when new code is committed and pushed, this pipeline should be comprehensive and cover aspects such as quality, and security; Travis or similar, for example.

Other things to consider as additions:

- Create unit tests and/or a test suite that validates your code.
- Describe or demonstrate any risks associated with your application/deployment.
- Describe your approach to versioning your application/deployment.
- Write a clear and understandable `README` which explains your application and its deployment steps.

The application code should be within a Github or Gitlab repository where we can review your source code and any configuration required for your project to execute. Please make sure the repository is public so it's viewable.