Arqiva Technical Challenge #2 - Backend Software Developer

Lawrence Bennett

The solution comprises:

- **string_app.py** – Python program that uses FastAPI to provide a GET endpoint that returns the string and a PUT endpoint that changes the string. The program runs as an AWS Lambda and keeps the dynamic string in DynamoDB.
- **main.tf** – Terraform code that deploys the application into AWS. This uses Docker to package the Python program into a container and put the container into AWS Elastic Container Services.
- **Dockerfile** – instructions for Docker to build the container.
- **requirements.txt** – list of Python packages needed in the Docker container.

The choices and rationale are:

- AWS is the cloud platform since it is widely used and full-featured.
- Python is the language since it is useful for building APIs and I am very familiar with it.
- Lambda is the AWS platform since it is straightforward and avoids the downsides of setting up and running server instances.
- FastAPI is the framework since it is an easy way to build an API and provides API documentation for no extra effort.
- The Lambda is deployed as a Docker container since this ensures all software dependencies are met and isolates the application from changes in the underlying platform.
- DynamoDB for persistent storage of the string. I chose this mainly because it was easy. There are many other options in AWS, for example S3 which would probably have been even easier.
- The application includes an endpoint for setting the string. This makes it possible to change the string without redeploying. Depending on the requirements, this endpoint might be unnecessary since one could also change the string via the AWS command line, SDK or management console.

With more time, some improvements would be:

- Put the API onto HTTPS and require authentication. A good way to do that would be with AWS API Gateway.
- Add some basic error checking to the Python application, for example to ensure the string being saved is not large enough to cause problems.
- Encode/sanitize the string being set so that an attacker cannot inject code into a user's browser.
- Provide a web page for changing the string.
- Change the deployment code so that the container is built in the cloud rather than locally on a Linux system.