DevOps Engineering
Assessment 2 Technical Scope

Summary
The technical scope of assessment 2 is as follows:

- ☐ Create a survey application using Python
- ☐ Create infrastructure via code to run the application in the cloud
- ☐ Introduce automation to check and deploy updates to the application

Prior to beginning the assessment you will be paired with another apprentice and also join a larger group as a team. Your assessment code will be individually written but in your pair, you will perform peer reviews of each other's code via GitHub Pull Requests.

On being introduced to Infrastructure as Code, you will create some basic code to deploy your application to the Amazon Web Services (AWS) cloud.

Finally, you will introduce automation using the principles of CI/CD to automatically test your application following changes and redeploy it to the cloud.

Technical Requirements
In order to deliver the technical output, you will need access to the following tools:

- ☐ Python (version 3.10+)
- ☐ Docker
- ☐ GitHub Account
- ☐ Terraform (version 1.3.7+)
- ☐ Amazon Web Services Access

Version Control
(Version control will be introduced in Workshop 3.)
As you produce your code, all will be stored in a GitHub repository in your own personal space.
Your repository for the assessment will be named **multiverse-devops-assessment-2** and will have the following folder structure:

```
.
├── .github
│   └── workflows
├── python-survey-app
└── terraform-iac
```

Python Application

(Python programming will be introduced in Workshop 3 and TDD will be introduced in Workshop 5.)
For the assessment you will be creating a survey result processing application which will read in a provided CSV file, cleanse the data and add it to a database.
The application will consist of two separate scripts: a processing script to read in and cleanse the data, and an output script which will output the results stored in the database.
In creating the application, you will need to evidence Test Driven Development, or TDD, by writing unit tests and code modularisation.

File rw example
As we've not covered file reading or writing, or CSV files in Python so far, here is a small snippet of code to demonstrate how this works in Python:

```
import csv

def read_csv():
  rows = []
  with open('./testFile.csv', 'r') as f:
    reader = csv.reader(f, delimiter=',')
    for row in reader:
      rows.append(row)
  return rows

def write_csv(row):
  with open('./testFile.csv', 'w') as f:
    writer = csv.writer(f, delimiter=',')
    writer.writerow(row)

def append_csv(row):
  with open('./testFile.csv', 'a') as f:
    writer = csv.writer(f, delimiter=',')
    writer.writerow(row)


print(read_csv())
write_csv([1,2,3,4,5])
print(read_csv())
append_csv([6,7,8,9,10])
print(read_csv())
```

This code reads in a file named testFile.csv and does the following:

- Outputs the existing data in an array of rows
- Replaces the original file contents with a row of `1, 2, 3, 4, 5`
- Adds a second row to the file of `6, 7, 8, 9, 10`

Given an initial file containing the row `a, b, c, d, e`, the output from the example script is:

```
[['a', 'b', 'c', 'd', 'e']]
[['1', '2', '3', '4', '5']]
[['1', '2', '3', '4', '5'], ['6', '7', '8', '9', '10']]
```

You can read more on processing CSV data in Python at https://realpython.com/python-csv/.

Infrastructure as Code

(Infrastructure as Code will be covered in Workshops 8 and 9.)
Amazon Web Services will be used to host your application and you will be required to develop the Terraform code that will provision this infrastructure.

The final infrastructure will consist of:

- An S3 Bucket to host the input file your application ingests as well as a file with output it produces.
- An EC2 instance that executes your application, supported by an AutoScaling Group to ensure an instance is always available.
- An associated VPC and subnets for the instance to run within.
- Appropriate Security Group(s).

The Infrastructure as Code will also handle the upload of new input files to the S3 Bucket so that you can demonstrate updates and continuous deployment.

Automation

(Automation will be introduced in Workshop 10.)
GitHub Actions will be used to demonstrate automation of your application code.
These will:

1. Run your unit tests on the opening of any Pull Request or merge to the main branch.
2. Execute Terraform on a merge to the main branch, contingent on the unit tests passing.