Developer Operations

Assignment 1

1. Core assignment specification

The overall objective of this assignment is to automate using Python 3 the process of creating, launching and monitoring a public-facing web server in the Amazon cloud. The web server will run on an EC2 instance and display some static content that is stored in S3. The program that does this must be called **run_newwebserver.py**

More detailed specification:

- Firstly, your Python program should create and launch a new Amazon EC2 micro instance. You must use the boto3 API library to launch from a free tier Amazon Linux AMI. You will need to have your Amazon credentials in a configuration file (`~/.aws/credentials`) and not in the Python code.
- Ensure your program launches the instance into an appropriate security group (you can optionally create one programmatically) and is accessible using your SSH key.
- You should provide a *"User Data"* start-up script when creating the instance. This start-up script should apply any required patches to the operating system and then install the *nginx* web server.
- Next, your program should use *scp* to copy a **check_webserver.py** script from your machine to the new instance and then execute this script (using *ssh* remote command execution, for example) to check if the nginx process is running. You will need to use the public IP address or DNS name assigned to your instance to connect to it using scp or ssh. If check_webserver script indicates that nginx is not running, then it should be started up.
- Another core requirement is that you write Python 3 code to create an S3 bucket and copy an image up to this bucket. Your nginx webserver main (home) page should then be configured so that this image will be visible by a browser visiting the website.
- Your code should perform appropriate error handling (using exceptions) and output meaningful messages to the user. Implement logging functionality so that details of what is happening (whether normal or errors) can be written to the console or a file.

2. Some additional functionality

The above is the core assignment specification. In addition you are expected to explore one or more other tasks that you can carry out with utility scripts like these. We are leaving it up to your individual initiative to decide what to do, but the following would be reasonable examples:

- Configure other services remotely
- Pass parameters as command line arguments
- Query web server access/error logs, possibly using *grep* or equivalent
- Investigate how to monitor resource usage (e.g. CPU utilisation) and, if above certain threshold, create and start another instance automatically
- Make user friendly

3. Non-functional issues: readability, robustness, user-friendliness

As well as for core functionality and testing, marks will be awarded for:

- Testing – for example using Python's *unittest* module or alternatives.
- Script readability – it helps to have good code comments, appropriate code layout/spacing, and good variable and function names.
- Robustness – how the script deals with error conditions and unexpected situations.
- User-friendliness of the scripts – meaningful messages provided to user and opportunities for user interaction.

Marking scheme:

- 50% (1) Core functionality – as specified
- 20% (2) Additional functionality – to your own specification (at least one "extra")
- 30% (3) Non-functional issues including testing

Note on working with EC2 for your assignment:

You should not need to save any work on an Amazon EC2 instance while working on this assignment. You should create and maintain your scripts locally and also take backups for yourself. Thus any termination of your EC2 instances should not affect your work.

Assignment submission:

Please submit your scripts to Moodle as a single **zip** archive.