Automated Cloud Services

Assignment 1

A. Core functionality required

The overall objective of this assignment is to write a Python 3 program to automate the process of creating, launching and monitoring public-facing web servers in the Amazon cloud. A web server will run on an EC2 instance and an S3 bucket will also be used to serve some static content. The program that does this must be called **Assignment1.py**

More detailed specification:

1. ***Launch EC2 instance***. Firstly, your Python program should create and launch a new Amazon EC2 *nano* instance. You must use the Boto3 API library to launch from an Amazon Linux 2 AMI. Use an up-to-date AMI version. You will need to have your API credentials in a configuration file (`~/.aws/credentials`) and not in the code.
2. ***Configure appropriate instance settings (at launch).*** Ensure that your program launches the instance into an appropriate security group (you can optionally create one programmatically) and that the instance is accessible using your SSH key
3. ***Set up EC2 website.*** You should provide a *"User Data"* script when creating the instance. This start-up script should apply any required patches to the operating system and then install the web server (e.g. *Apache*). The start-up User Data script should also configure the web server index page to display *instance metadata* (e.g. availability zone, private IP address, subnet) and optionally some other content e.g. text or image. When your instance has launched you should use ssh remote command execution to retrieve instance metadata (e.g. public ip address) and print to you program output.
4. ***Set up S3 website***. Another core requirement is that you write Python 3 code to create an S3 bucket. This bucket needs to contain at least two items:
    - An image which will be available at ***https://witacsresources.s3-eu-west-1.amazonaws.com/image.jpg***. Your Python program should download this image and then upload it to your newly-created bucket. The image at this URL will change from time to time, so your code will need to handle this in a generic manner.
    - A web page called *index.html* which displays the image - e.g. using <img> tag.

    Configure the S3 bucket for static website hosting so that the image can be accessed with a URL of the form *http://bucket-name.s3-website-eu-west-1.amazonaws.com* (note that index and image file names are not in the URL, just the bucket name)
5. ***Launch browser and display both URLs***. Your Python program should launch a web browser (e.g. Firefox) and open both your EC2 and your S3 web pages.
6. ***Monitoring.*** A bash script is provided called *monitor.sh* that runs some sample terminal commands that carry out monitoring. You should enhance this script to monitor some additional items. Then, from your Python script, use *scp* (secure copy) to copy this script up to your newly-created instance and then use SSH remote command execution to set the appropriate permissions and execute this script on the instance. You will need to use the public IP address or DNS name assigned to the instance to connect to it via SSH.

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

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

- **Robustness**. Your code should do appropriate error handling (using exceptions) and output meaningful messages to the user when errors and unexpected situations occur. Ensure that your code is well tested and debugged.
- **Logging**. Display to the console (and/or log to a file) details of what is happening, whether normal or errors.
- **Code readability.** It helps to have good code comments, appropriate code layout/spacing, and good variable and function names.

C. Additional functionality – to your own specification

The above is the core assignment specification. In addition you are expected to explore one or more other tasks. It is up to your individual initiative to decide what to do, but the following would be reasonable examples:

- Configure other AWS services remotely
- Query web server access/error logs, possibly using *grep* or equivalent
- Use boto3 to monitor instance metrics such as CPU utilisation using CloudWatch

N.B. Do not implement an elaborate menu/UI. The program is meant to demonstrate cloud services automation with minimal user interaction. However it is ok to have some user prompts – for example at the start you might wish to prompt for key name, bucket name, etc.

Assignment submission:

You are required to submit a single **zip** archive named ***jbloggs.zip*** (replacing your name) containing the following:

1. Your Python program called *Assignment1.py*
2. Your updated *monitor.sh* script
3. Completed *Assignment Review spreadsheet* provided.

Assignment review:

A demo will be scheduled for each student to provide a short demonstration of your submission and answer some questions. These demos will take place during Week 7 and a schedule will be published closer to the time.

Marking scheme:

- 60% Core functionality – as specified
- 10% Additional functionality – to your own specification (at least one "extra")
- 10% Non-functional issues including testing, code quality, robustness, etc
- 20% Demonstration, overall understanding, coherence and clarity

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.

Warning re originality of code provided:

It is **very** important to provide a proper citation/reference for any code that appears in your submission that is not entirely written by you. Any such code should be used very sparingly. It is much better to write a short program from scratch that meets the specification than to submit something long and elaborate if that means using substantial chunks of code or a design that is not your own.