COCUS Assignment - Coding

In this recruitment exercise, we are going to assess your coding skills. All of the following tasks should be done using Python or JavaScript. All the developed code should have proper testing coverage.

First exercise

Definition

Find all files within a path, with a given file name suffix. Note that a path may contain further subdirectories and those subdirectories may also contain further subdirectories. There is no limit to the depth of the subdirectories.

Arguments:

suffix(str): suffix if the file name to be found
path(str): path of the file system

Returns:

a list of paths and file names that match the suffix

Expected deliverable

Your code should be executable with the following call “yourScript.py *.log /var/tmp”.

See below for an example of what the output should be (for a given scenario):

Scenario:

```
|-- a
| |-- bbb
| |-- bbb.log
| |-- ddd
|-- aaa.log
|-- abc.txt
```

Output:

```
./aaa.log
./a/bbb.log
```

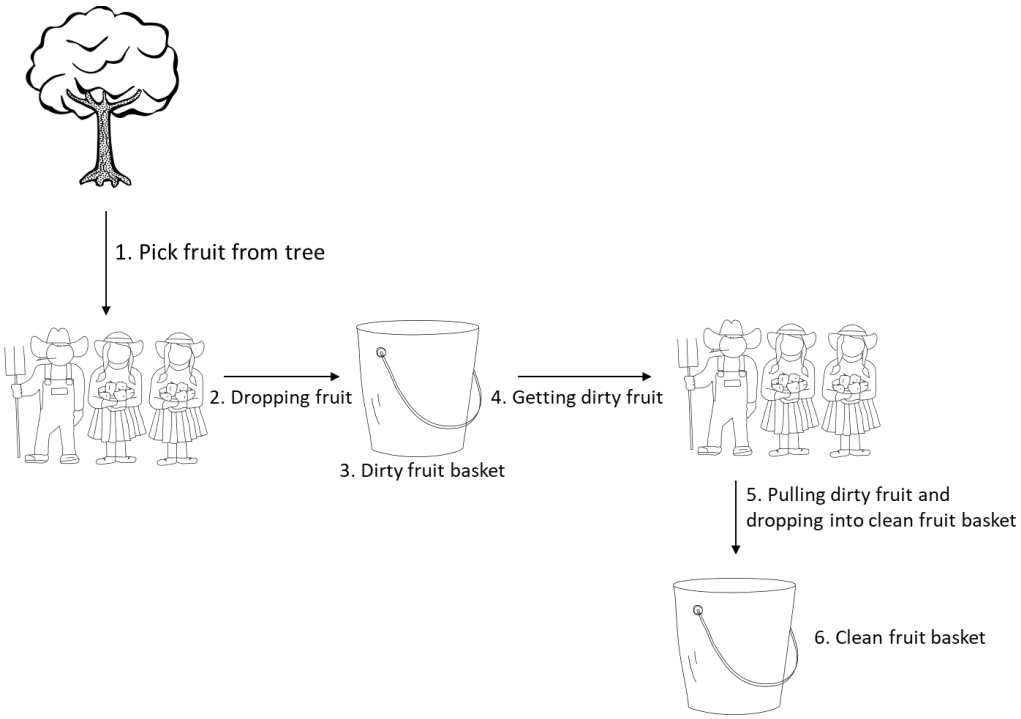
Second exercise

Definition

In this exercise we will be testing your multithreading skills.

Suppose that you want to simulate a fruit farm:

- Three farmers are collecting fruits from a single tree to a single dirty fruit basket.
- In parallel, three other farmers are getting the fruits from the dirty fruit basket, cleaning them, and pushing them into the single cleaned fruit basket.
- All the farmers are managing the fruit individually
- The tree has 50 fruits (and only one farmer at one time can pick fruit from the tree)
- Time to collect fruits from the trees into the basket: random(3,6) seconds
- Time to clean the fruits into the cleaned fruit basket: random(2,4) seconds
- The simulation ends when all the fruits from the tree are collected and cleaned.
- The number of fruits in the tree and in the baskets must be logged every second.
-



Expected deliverable

Your code should be executable with the following call “yourScript.py”, with a similar output log to the one below:

- 2020-12-01 19:02:00 Tree (50 fruits) - dirty basket (0) - Clean Basket (0) – farmer1 (0) – farmer2 (0) – cleaner1(0) – cleaner2 (0)
- 2020-12-01 19:03:00 Tree (45 fruits) - dirty basket (3) - Clean Basket (1) – farmer1 (1) – farmer2 (0) – cleaner1(0) – cleaner2 (0)
- 2020-12-01 19:10:00 Tree (0 fruits) - dirty basket (0) - Clean Basket (50) – farmer1 (0) – farmer2 (0) – cleaner1(0) – cleaner2 (0)

COCUS Assignment - Architecture

In this recruiting stage, we are going to assess if you have the skills and knowledge necessary to design and develop systems capable of ingesting and processing large amounts of data. All the requested explanations in the deliverables below should be added to a presentation in PDF (in english).

First exercise – Architecture

Definition

The requirement is to design a data pipeline architecture which will enable the ingestion of data from a third party tool and dumping it on a data lake (S3 Bucket). The third party tool exposes an HTTP endpoint, reachable only with a GET method. The data must be stored in raw format. A transforming layer must convert the data to a binary format (compressed). A serving layer through a SQL interface must be offered to the data scientists, so they can interact with the transformed data in the bucket.

Requirements:

- Error-handling
- Retry mechanism
- Monitoring
- Scaling horizontally
- Data is stored into S3 to be serviced to the data scientists

Expected deliverable

The main output of this stage should be one diagram with the proposed solution and an explanation of the technological and architectural choices which were made. All the solution must be based on AWS components.

Second exercise – DevOps considerations

Definition

Managing the data flows like the ones requested in the goals above requires a strong DevOps component.

Please discuss what strategies, technologies and tools you would use to simplify and automate this day to day management, namely:

- metrics and monitoring for the running dataflows
- recovering from a failed dataflow
- testing a dataflow in a development environment and then deploying it to a production environment
- modifying code on the dataflow without stopping it

Expected deliverable

The main output of this goal is a discussion of the above topics and proposed approaches on how to tackle them. All the solution must be based on AWS components.