Problem Statement

In Nutanix clusters, we heavily employ distributed architectures where one cluster comprises of multiple machines (nodes) and most of our software run in all the nodes simultaneously acting as multiple masters or slaves to one master node. The processes running in each of these nodes emit a lot of logs which are used later for debugging issues or oncalls (i.e. debugging issues on customer setup). Because of this distributed architecture, the log files also are scattered across these nodes and they run into multiples of GBs. Older log files are archived into tarballs to save space.

The **Hackathon Challenge** is to come up with a **Distributed Logs Analyzer app** (web or desktop) which will help us scan and search across these various log files. Some of the use cases we are looking for in the app are:

1. Ability to scan and search for certain exception patterns or any search phrase across all the log files on all nodes.
   This is similar to ***grep*** command we have in Unix.
2. Live debugging i.e. app should provide the ability to monitor logs of a particular process in all the nodes, LIVE, in the form of streaming logs. We should be able to input an exception pattern there too which the app should highlight as they encounter that in any of the streaming log files.
   For streaming, consider it similar to ***head*** and ***tail*** commands in Unix.
3. The app should allow the user to easily extract logs between two given timestamps from all processes.
4. Additional, good to have features would be:
   a. To allow users to set up alerts over specific exception types
   b. Correlation of logs by identifiers appearing in the log messages - you can use identifiers to club together different types of log messages (for example, based on INFO, DBG or ERROR).
   c. Timeline view of a searched keyword E.g. I could search by an ID or a name and would want a timeline view of logs for that.
   d. Exception heat maps

Participants are encouraged to innovate further and come up with their own features for this solution. The features given above are just examples of what can be done - you should be creative with your solutions! To help the participants, we have provided a simulator program

which will generate random logs of the kind of distributed nature described in this problem. Participants can use that script for their testing and understanding of the problem.

Submission is expected to be in the form of:

a. A demo video (<= 2mins)

b. App code (zip file / github link etc.)

c. A short README explaining how to use the app and all.

How to run and use the Simulator script:

You have been provided by a log_simulator.zip file. This file is meant to be a simple testing set for your application.

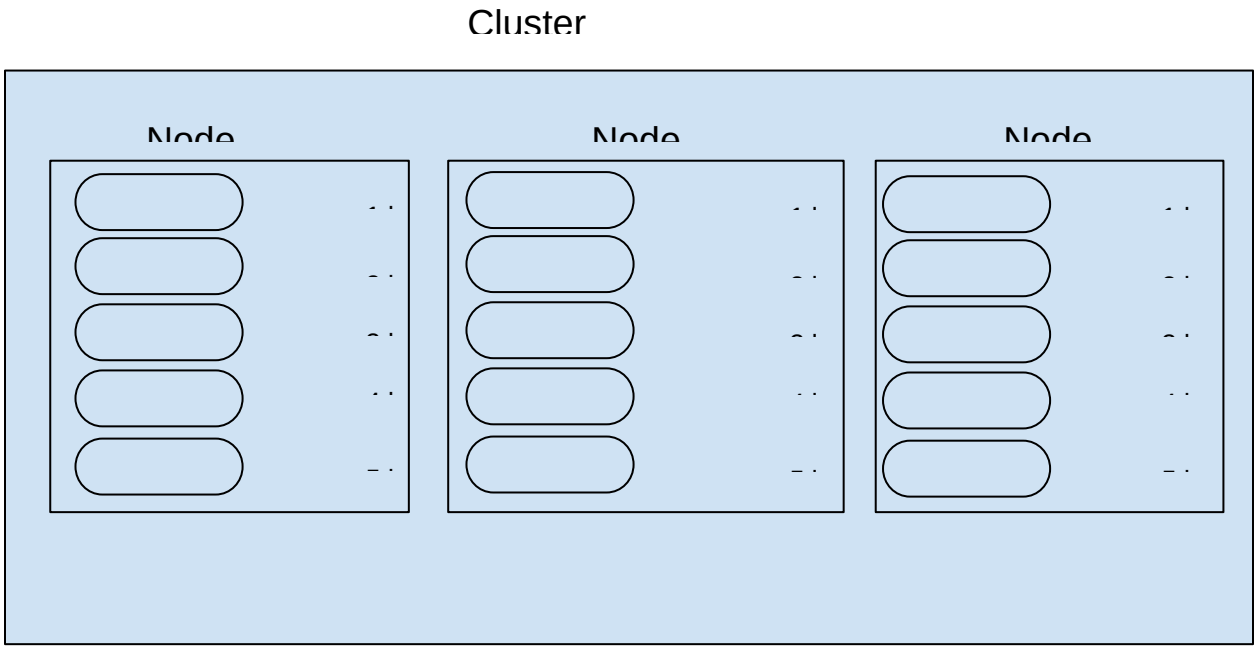
```
usage: log_simulator.zip [-h] -n NODES -p PROCESSES [--new]
                         [--sleep_time SLEEP_TIME]

optional arguments:
  -h, --help            show this help message and exit
  -n NODES, --nodes NODES
                        Number of nodes to simulate for log collection.
  -p PROCESSES, --processes PROCESSES
                        Number of processes outputting logs
  --new                 Erase previous directories and start new logging
  --sleep_time SLEEP_TIME
                        Increase this time to slow down log generation.
                        Default value is 0.
```

Copy over this zip file into some directory.
You can run it using python log_simulator.zip -n (num-nodes) -p (num-processes). This will create a log_simulator directory in the current working directory. All the log information will be dumped into this directory.
Specifying --new will delete all previous directories created by the script in the current directory.
Sleep time is an optional argument which will slow down log generation.

(Argparse module will be needed to run the script. You can install this using `pip install argparse`.
The script is supported on python version 2.7.)

Example:

Let’s say, we have 5 software components which are deployed in a cluster with 3 nodes. So each of these 5 components will run as separate processes in each of the nodes and each of them will emit logs.



To run the log_simulator for this kind of setup, run the following:

python log_simulator.zip -n 3 -p 5

The logs will be dumped in a directory named log_simulator/:

```
log_simulator
        |
        |->HackNode1
                |
                |-----> process1.log
                |-----> process2.log
                |...
                |------> process5.log

        |->HackNode2
                |
                |-----> process1.log
                |-----> process2.log
                |...
                |------> process5.log
```

|->HackNode3
|
|-----> process1.log
|-----> process2.log
|...
|------> process5.log