1. You will be in charge of running a set of JVM-based microservices connected to MongoDB, exchanging messages through a kafka broker and communicating with external clients using HTTP Restful Services. What metrics do you consider the most critical to monitor for the system and what tools would you use?

Metrics:

* HTTP Traffic: to observe the traffic of or API's when clients use the different services.
* Broker network Throughput: monitor the broker's capacity to observe the broker's performance on both Tx and Rx
* Under-replicated Partitions: in order to ensure that brokers are always in an active state and to avoid loss of data delivery.
* Log Flush Latency: due to several factors Kafka performs internal processes in an asynchronous manner that when saving with the log data in memory can generate some latency, where, if not resolved in time may indicate hardware problems or storage scaling problems.
* Bytes out: in order to see the rate of outgoing bytes going to all servers and in turn determine the amount of outgoing traffic from all services, it can somewhat influence the costs.
* Database connections: In order to observe the constant connection with the MongoDB database to determine the durability of the data with all the API's.

Tools:

* Solarwinds and nagios: this software allowed establish critical thresholds and warnings when the limits are exceeded, also, it is capable of probing the network and fetch the bandwidth in real time.
* Datadog: through this service can be monitored some TI services and cloud services.

1. Provide a Linux command to final all files which have been accessed between 20 and 30 days ago. Explain your command

find . -type f -newermt 2021-02-25 ! -newermt 2021-03-21

The ***find*** parameter to search and find files, and is added the path, in this case all directories, specified the file types of filenames. The parameter ***newermt*** is divided into three sections, where ***newerXY*** compares timestamps of files and has additional parameters to adjust, in this case the ***m*** parameter specifies the last file changed, and the ***t*** parameter determines the time reference to be adjusted.

<https://unix.stackexchange.com/questions/169798/what-does-newermt-mean-in-find-command>

1. Write a simple Python script that lists the 5 biggest files in a given directory (sorted by size), as well as the total size, number of files and average size of files in the directory.

import os #Library to manipulate paths and write or read a file  
dir = "./files"  
files = os.listdir(dir) #to fecth the filenames  
files\_by\_size = sorted(files, reverse = True, key=lambda f: os.path.getsize(os.path.join(dir, f))) #there are  
# organized by size from smallest to largest (reverse = True largest to smallest)  
nfiles = files\_by\_size[:5] #it is specified that there are only 5  
#print(files)  
#print('------------------------------------------------------------------------')  
#print(nfiles)  
#print('------------------------------------------------------------------------')  
for i in range(5):  
 print(str(nfiles[i]) + " " + str(os.stat(nfiles[i]).st\_size)) #Print a file with the exacly size in front there  
a=0  
for i in range(len(files)):  
 a = a + os.stat(files[i]).st\_size #to fetch a file size and following size is added  
print("Total size:" + str(a) + " " +"Average file: " + str(a/len(files))+ " " + "Total files: " + str(len(files)

1. Most of the time the service is running without problems. However, we are facing situations where the flask application stops working or fails, yet the docker container keeps running making it difficult to detect when it happens. Additionally, when running as a standalone application, the flask application logs its activity and errors to the standard output, but in production we are not able to see the log. Please highlight any problem you can spot (if any) in those files. Then diagnose the problem we are facing and propose a working solution Note: The actual Python code is irrelevant for our purposes, it could be any arbitrary Flask application

When the Flask application fails or stops in the container, it might be due to the main process of the container. In the Dockerfile, it is declared in the CMD layer the following line: **“CMD [ "/bin/docker-entrypoint" ]”** to be run as main EXEC process, and in the docker-entrypoint, the main process will be executed by **#!/bin/ash** or **#!/bin/sh**, that is, SHELL and not EXEC, that could cause internal execution problems of the application, and causing an unexpected closing of the application. Probably, we can use **#!/bin/bash** or in the Dockerfile it can be written in the SHELL form (**CMD “/bin/docker-entrypoint"**). Also, the main process in the container could be changed by one that works with the desired requirements.

For the case of the logs error, it could be due to the user permissions of the nginx in the directory. It could work by changing those permissions from the nginx user.