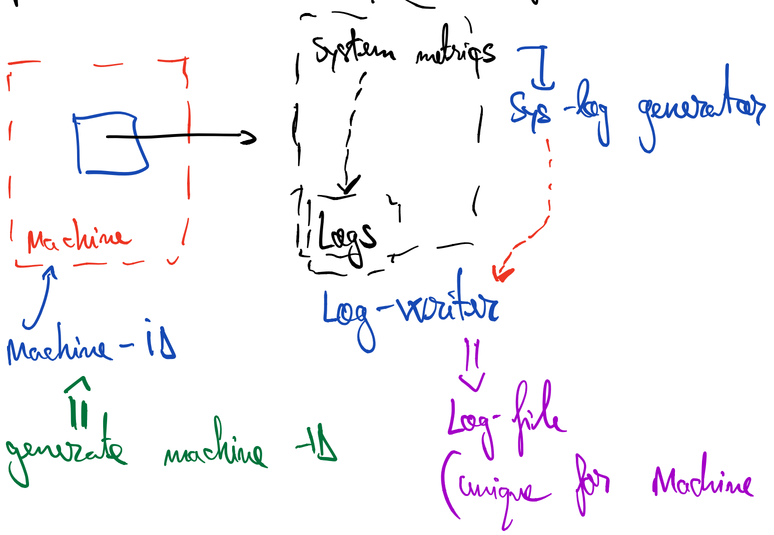
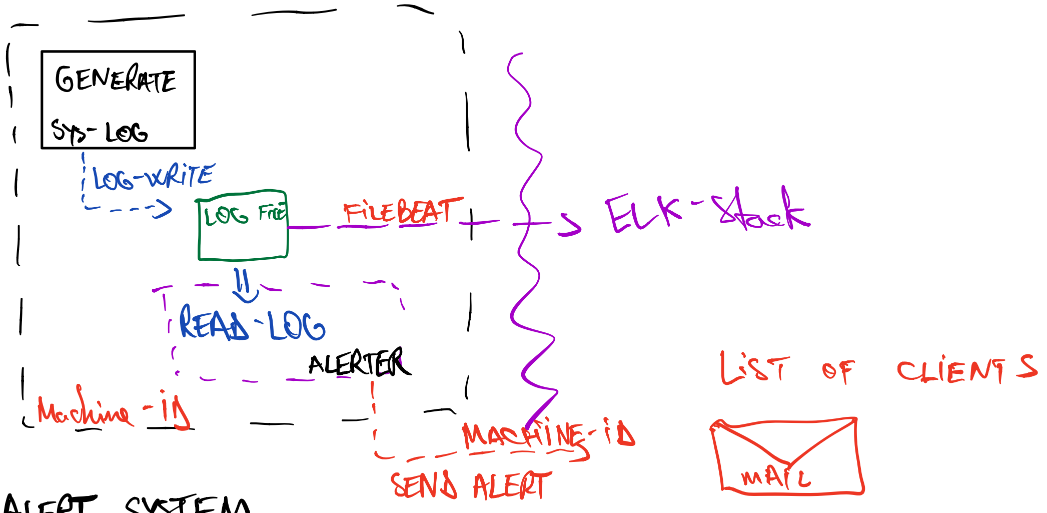
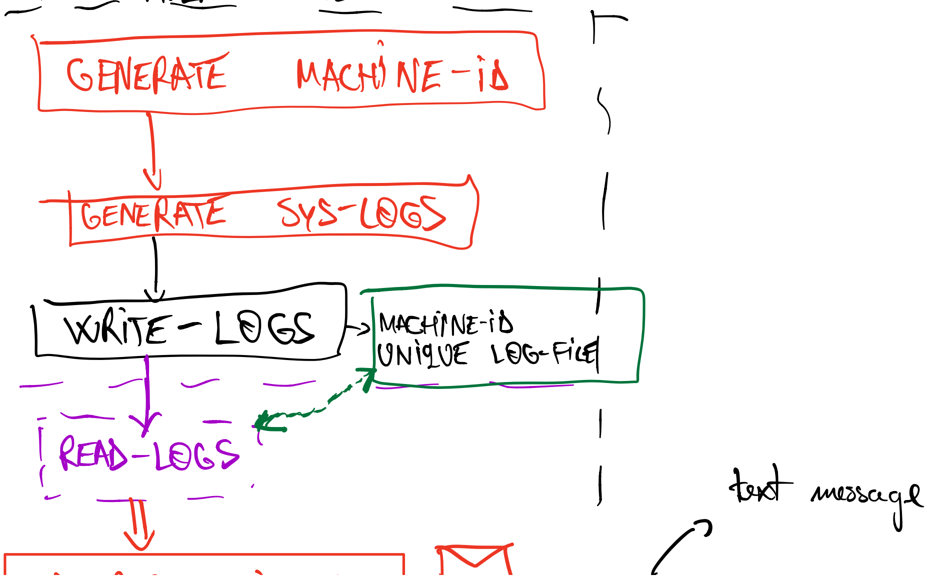
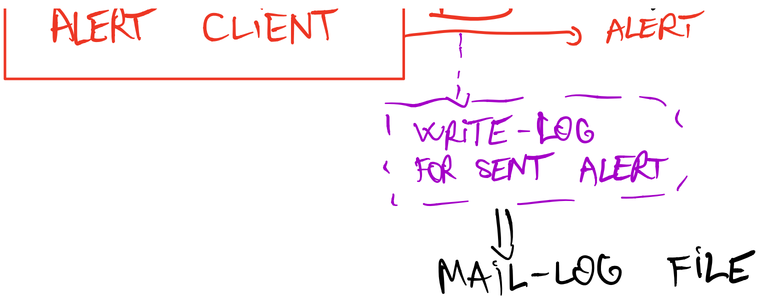
- implement an alert system for Kibana & ELK stack

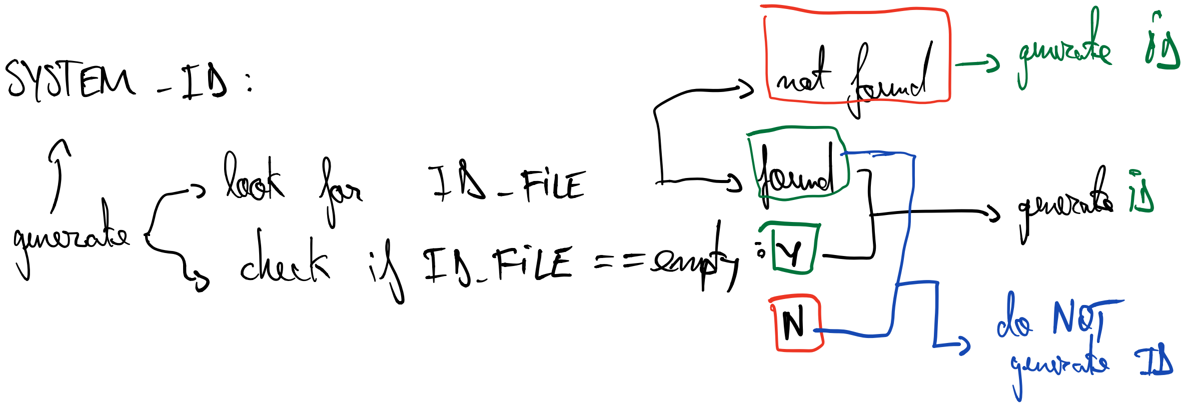




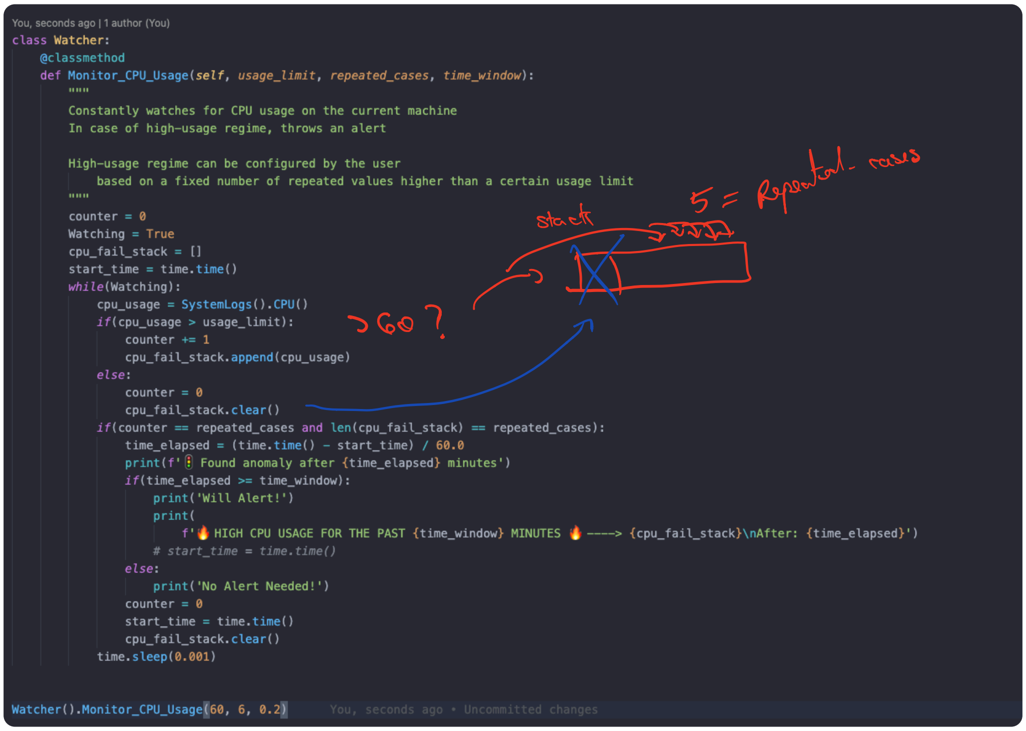
PYTHON ALERT SYSTEM







PULL MACHINE-ID



```
You, seconds ago | 1 author (You)
class Watcher:
    @classmethod
    def Monitor_CPU_Usage(self, usage_limit, repeated_cases, time_window):
        """
        Constantly watches for CPU usage on the current machine
        In case of high-usage regime, throws an alert

        High-usage regime can be configured by the user
            based on a fixed number of repeated values higher than a certain usage limit
        """
        counter = 0
        Watching = True
        cpu_fail_stack = []
        start_time = time.time()
        while(Watching):
            cpu_usage = SystemLogs().CPU()
            if(cpu_usage > usage_limit):
                counter += 1
                cpu_fail_stack.append(cpu_usage)
            else:
                counter = 0
                cpu_fail_stack.clear()
            if(counter == repeated_cases and len(cpu_fail_stack) == repeated_cases):
                time_elapsed = (time.time() - start_time) / 60.0
                print(f'🚦 Found anomaly after {time_elapsed} minutes')
                if(time_elapsed >= time_window):
                    print('Will Alert!')
                    print(
                        f'🔥 HIGH CPU USAGE FOR THE PAST {time_window} MINUTES 🔥 ----> {cpu_fail_stack}\nAfter: {time_elapsed}')
                    # start_time = time.time()
                else:
                    print('No Alert Needed!')
                counter = 0
                start_time = time.time()
                cpu_fail_stack.clear()
            time.sleep(0.001)


Watcher().Monitor_CPU_Usage(60, 6, 0.2)      You, seconds ago • Uncommitted changes
```

Over [75 %] in the last [5] seconds

↓

pull CPU_USAGE ↓ 1/s = frequency_pull

x times per second

COUNT

if COUNT >

count = 0

1/second

time:			
1 ->	80 > 75	+1	
2 ->	78 > 75	+2	
3 ->	81 > 75	+ 3	
4 ->	81 > 75	+4	$\frac{5}{5}$
5 ->	83 > 75	+5	

if elapsed_time = time_window & count = $\frac{\overset{5}{\text{time\_window}}}{\text{frequency}}$

5 sec 5 sec

elapsed-time seconds

frequency: 2/s

0.5	80 > 75	1
1		2
1.5	80 > 75	3
2		4
2.5		5
3		6
3.5		7
4		8
4.5		9
5		10

$\underset{(a)}{\text{elapsed\_time}}$ = time_window count = 10

=> if (a) and if (count / freq = $\overset{(b)}{\text{time}}$_window)

```
counter = 0        start_time( )
While True :    ——>   maximum pulls /second : 2
    measure_time( )
    get_cpu_usage( )
    check_high_usage( )
        ↳ if (true) : counter ++
    wait_for ( 1/ max_pull )
elapsed_time( )
    ↳ if elapsed_time( ) ≥ time_window
        { check_counter( )
```

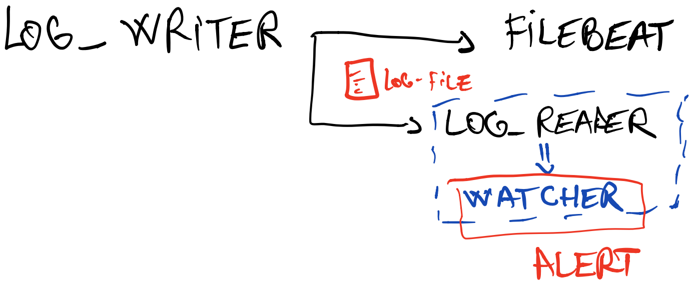
2 pulls /second → X seconds

⇓

counter ≥ 2 X counts

if counter ≥ 2x counts => ALERT

SCRIPTS RUNNING ON THE SYSTEM


LOG_WRITER
FILEBEAT
LOG-FILE
LOG_READER
WATCHER
ALERT