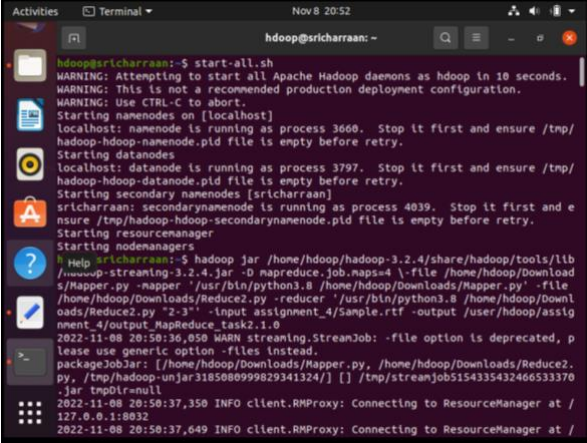


# ENGR-E: 516 - Engineering Cloud Computing

## Assignment 1: MapReduce using Hadoop

### 1.1 Setting Up: Hadoop Environment

- After Setting up Ubuntu environment I downloaded the latest version of jdk
- Then, the setup of SSH localhost is done
- Then, download the Apache Hadoop version 3.2.4
- Then, setting path of different .xml files was done.
- Then, after allocating the path the whole file is run and Hadoop is setup
- Then to start Hadoop the command “start-all.sh” is used as one can see in the *Fig. 1.1*
- The whole process was referenced, and the links of reference are mentioned in 1.6

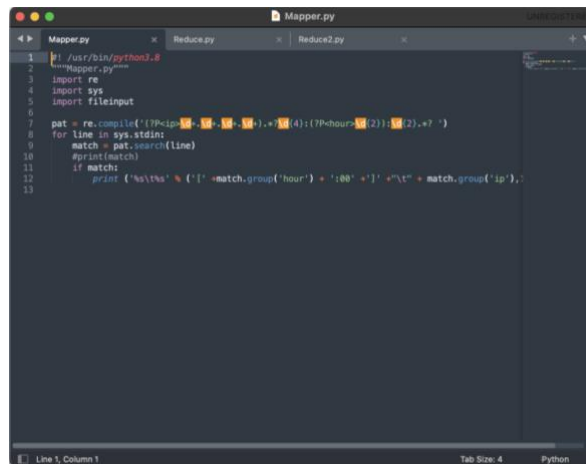


```
hadoop@sricharraan:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
localhost: namenode is running as process 3660. Stop it first and ensure /tmp/
hadoop-hadoop-namenode.pid file is empty before retry.
Starting datanodes
localhost: datanode is running as process 3797. Stop it first and ensure /tmp/
hadoop-hadoop-datanode.pid file is empty before retry.
Starting secondary namenodes [sricharraan]
sricharraan: secondarynamenode is running as process 4039. Stop it first and e
nsure /tmp/hadoop-hadoop-secondarynamenode.pid file is empty before retry.
Starting resource manager
Starting nodemanagers
hadoop@sricharraan:~$ hadoop jar /home/hadoop/hadoop-3.2.4/share/hadoop/tools/lib
/./hadoop-streaming-3.2.4.jar -D mapreduce.job.maps=4 -file /home/hadoop/Downloa
d/Reducer.py -mapper /usr/bin/python3.8 /home/hadoop/Downloads/Reducer.py -file
/home/hadoop/Downloads/Reducer2.py -reducer /usr/bin/python3.8 /home/hadoop/Downl
oads/Reducer2.py "2-3" -input assignment_4/sample.rtf -output /user/hadoop/assig
nment_4/output_MapReduce_task2.1.0
2022-11-08 20:50:36,050 WARN streaming.StreamJob: -file option is deprecated, p
lease use generic option -files instead.
packageJobJar: [/home/hadoop/Downloads/Reducer.py, /home/hadoop/Downloads/Reducer2.
py, /tmp/hadoop-unjar318508099829341324/] [] /tmp/streamjob5154335432466533370
.jar tmpDir=null
2022-11-08 20:50:37,350 INFO client.RMProxy: Connecting to ResourceManager at /
127.0.0.1:8032
2022-11-08 20:50:37,649 INFO client.RMProxy: Connecting to ResourceManager at /
```

Fig. 1.1: Starting Hadoop Environment

### 1.2 Coding Mapper and Reducer Function

The Fig.1.2.1 shows the Mapper function of the MapReduce function in which the time and IP address is scrapped from the log file and each match of time and IP address is given the count of 1.



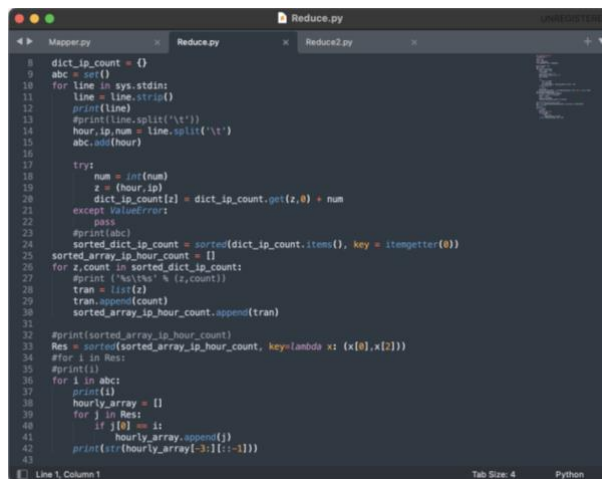
```

1  #!/usr/bin/python3.8
2  """Mapper.py"""
3  import re
4  import sys
5  import sys.stdin
6
7  pat = re.compile(r'(?P<ip>[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}) (?P<hour>[0-2]{1})')
8  for line in sys.stdin:
9      match = pat.search(line)
10     #print(match)
11     if match:
12         print('%s\t%s' % (match.group('hour') + ':00' + '\t' + match.group('ip')))
13

```

Fig. 1.2.1: Mapper Function

The Fig.1.2.2 shows the Reducer function of the MapReduce function in which the counting of repetition of IP address is done according to the time and reporting the **top 3-IP addresses for every hour**.



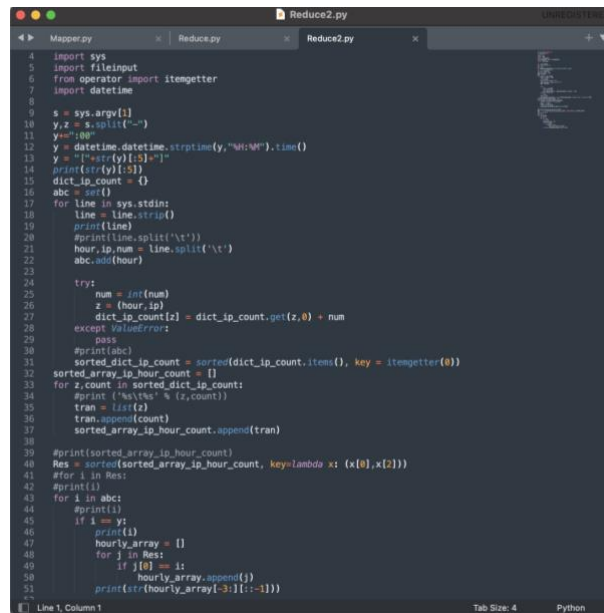
```

8  dict_ip_count = {}
9  abc = set()
10 for line in sys.stdin:
11     line = line.strip()
12     #print(line)
13     #print(line.split('\t'))
14     hour,ip,num = line.split('\t')
15     abc.add(hour)
16
17     try:
18         num = int(num)
19         z = (hour,ip)
20         dict_ip_count[z] = dict_ip_count.get(z,0) + num
21     except ValueError:
22         pass
23     #print(abc)
24     sorted_dict_ip_count = sorted(dict_ip_count.items(), key = itemgetter(0))
25     sorted_array_ip_hour_count = []
26     for z,count in sorted_dict_ip_count:
27         #print('%s\t%s' % (z,count))
28         tran = list(z)
29         tran.append(count)
30         sorted_array_ip_hour_count.append(tran)
31
32     #print(sorted_array_ip_hour_count)
33     Res = sorted(sorted_array_ip_hour_count, key=lambda x: (x[0],x[2]))
34     #for i in Res:
35     #    print(i)
36     for i in abc:
37         print(i)
38         hourly_array = []
39         for j in Res:
40             if j[0] == i:
41                 hourly_array.append(j)
42         print(sorted(hourly_array[-3:][:-1]))
43

```

Fig. 1.2.2: Reducer Function for top 3 IP Addresses

The Fig.1.2.3 shows the Reducer function of the MapReduce function in which the counting of repetition of IP address is done according to the time and only reporting the top 3-IP addresses of the user mentioned timeframe used for *Database Search*.



```

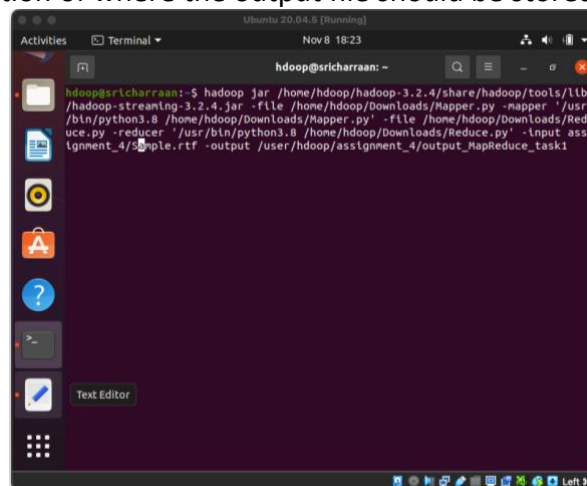
4 import sys
5 import fileinput
6 from operator import itemgetter
7 import datetime
8
9 s = sys.argv[1]
10 y,z = s.split("-")
11 y="00"
12 y = datetime.datetime.strptime(y,"%H%M").time()
13 y = "[%s-%s]" % (y[0:2],y[3:5])
14 print("y:",y)
15 dict_ip_count = {}
16 abc = set()
17 for line in sys.stdin:
18     line = line.strip()
19     print(line)
20     #print(line.split("\t"))
21     hour,ip,num = line.split("\t")
22     abc.add(hour)
23
24     try:
25         num = int(num)
26         z = (hour,ip)
27         dict_ip_count[z] = dict_ip_count.get(z,0) + num
28     except ValueError:
29         pass
30     #print(abc)
31     sorted_dict_ip_count = sorted(dict_ip_count.items(), key = itemgetter(0))
32     sorted_array_ip_hour_count = []
33     for z,count in sorted_dict_ip_count:
34         #print ("%s\t%s" % (z,count))
35         tran = list(z)
36         tran.append(count)
37         sorted_array_ip_hour_count.append(tran)
38
39     #print(sorted_array_ip_hour_count)
40     Res = sorted(sorted_array_ip_hour_count, key=lambda x: (x[0],x[2]))
41     #for i in Res:
42     #    print(i)
43     for i in abc:
44         #print(i)
45         if i == y:
46             hourly_array = []
47             for j in Res:
48                 if j[0] == i:
49                     hourly_array.append(j)
50             print(str(hourly_array[-3][:-1]))
51

```

Fig. 1.2.3: Reducer Function for Database Search

### 1.3 Output the top 3 IP addresses with the granularity of an hour

The MapReduce cluster is inbuilt function in Hadoop to build it one must call the Hadoop streaming file in which the location of mapper and reducer function must be mentioned, and input file should be uploaded in the Hadoop cluster and its location for smooth reading of the input and the new location of where the output file should be stored.



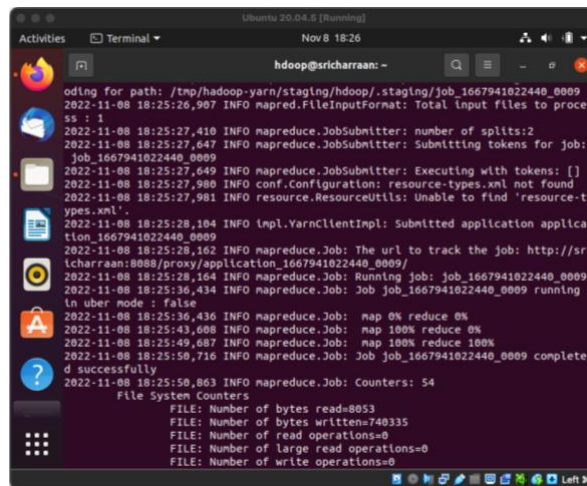
```

hadoop@sricharraan: ~
hadoop@sricharraan:~$ hadoop jar /home/hadoop/hadoop-3.2.4/share/hadoop/tools/lib/hadoop-streaming-3.2.4.jar -file /home/hadoop/Downloads/Mapper.py -mapper /usr/bin/python3.8 /home/hadoop/Downloads/Mapper.py -file /home/hadoop/Downloads/Reducer.py -reducer /usr/bin/python3.8 /home/hadoop/Downloads/Reducer.py -input assignment_4/sample.rtf -output /user/hadoop/assignment_4/output_MapReduce_task1

```

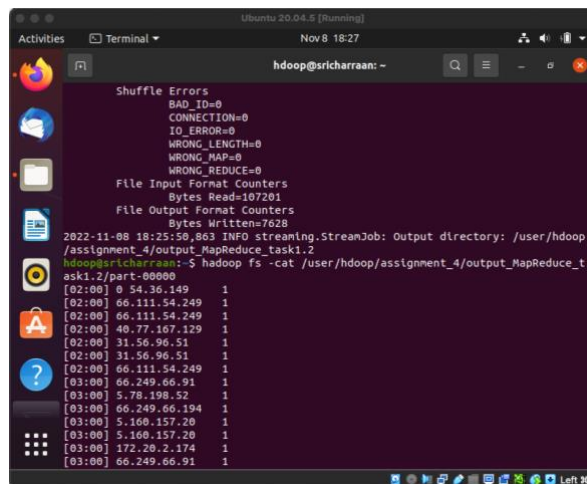
Fig. 1.3.1: MapReduce command for top 3 IP Address

After successfully giving all the details and location of the required documents when can see in Fig.1.3.2 the successful running of mapper and reducer function.



*Fig. 1.3.2: Successfully running of Mapper and Reducer*

After successful completion of the whole MapReduce function the output file stored in the designated location mentioned, and to access it from the Hadoop cluster we need to use the ***-cat*** function.



*Fig. 1.3.3: End MapReduce function and accessing output file*

The output of the whole MapReduce function can be seen in the *Fig. 1.3.4*

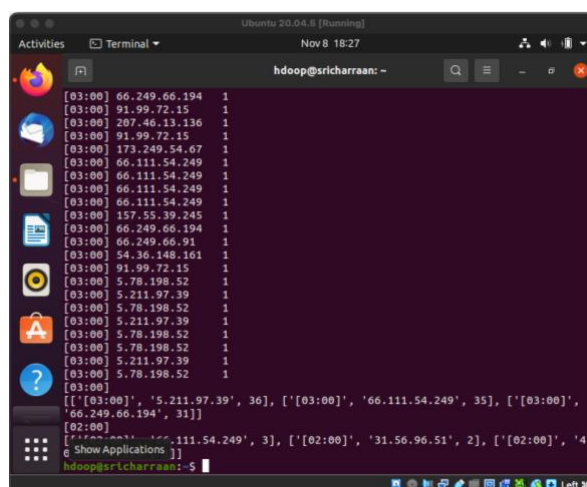


Fig. 1.3.4: Output of top 3-IP address with granularity of an hour

## 1.4 Database Search

The MapReduce cluster is inbuilt function in Hadoop to build it one must call the Hadoop streaming file in which the location of mapper and reducer function must be mentioned, and input file should be uploaded in the Hadoop cluster and its location for smooth reading of the input and the new location of where the output file should be stored. The Database is searched for 2:00 - 3:00

```

hadoop@sricharraan: ~
$ hadoop jar /home/hadoop/hadoop-3.2.4/share/hadoop/tools/lib/hadoop-streaming-3.2.4.jar -D mapreduce.job.maps=4 -f /home/hadoop/Downloads/Reducer2.py -r /usr/bin/python3.8 /home/hadoop/Downloads/Reducer2.py -input assignment_4/sample.rtf -output /user/hadoop/assignment_4/output_MapReduce_task2.1.1
2022-11-08 20:50:56,023 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/home/hadoop/Downloads/Reducer2.py, /home/hadoop/Downloads/Reducer2.py, /tmp/hadoop-unjar8034252455847050925/] [] /tmp/streamjob3143358070587752180.jar tmpDir=null
2022-11-08 20:50:56,961 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
2022-11-08 20:50:57,182 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
2022-11-08 20:50:57,458 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/staging/job_1667958597430_0001
2022-11-08 20:50:59,153 INFO mapred.FileInputFormat: Total input files to process : 1
2022-11-08 20:50:59,267 INFO mapreduce.JobSubmitter: number of splits:4
2022-11-08 20:50:59,573 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1667958597430_0001
2022-11-08 20:50:59,576 INFO mapreduce.JobSubmitter: Executing with tokens: {}
2022-11-08 20:50:59,900 INFO conf.Configuration: resource-types.xml not found
2022-11-08 20:50:59,901 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-11-08 20:51:00,390 INFO Impl.YarnClientImpl: Submitted application application_1667958597430_0001

```

Fig. 1.4.1: MapReduce command for Database Search from 2:00 – 3:00

After successfully giving all the details and location of the required documents when can see in Fig.1.4.2 the successful running of mapper and reducer function.

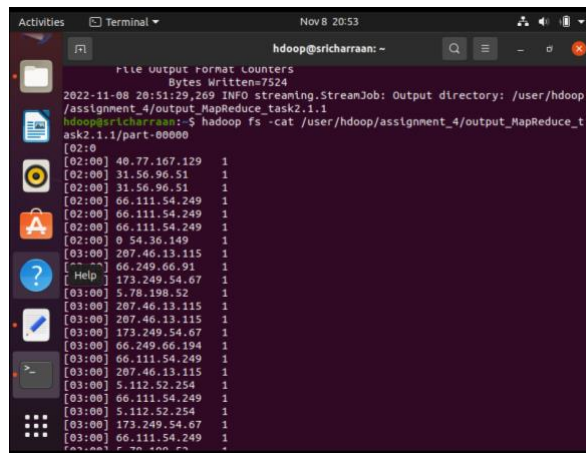
```

2022-11-08 20:50:59,900 INFO conf.Configuration: resource-types.xml not found
2022-11-08 20:50:59,901 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-11-08 20:51:00,390 INFO Impl.YarnClientImpl: Submitted application application_1667958597430_0001
2022-11-08 20:51:00,476 INFO mapreduce.Job: The url to track the job: http://sricharraan:8088/proxy/application_1667958597430_0001/
2022-11-08 20:51:00,479 INFO mapreduce.Job: Running job: job_1667958597430_0001
2022-11-08 20:51:09,718 INFO mapreduce.Job: Job job_1667958597430_0001 running in uber mode : false
2022-11-08 20:51:09,725 INFO mapreduce.Job: map 0% reduce 0%
2022-11-08 20:51:20,027 INFO mapreduce.Job: map 100% reduce 0%
2022-11-08 20:51:28,110 INFO mapreduce.Job: map 100% reduce 100%
2022-11-08 20:51:29,127 INFO mapreduce.Job: Job job_1667958597430_0001 completed successfully
2022-11-08 20:51:29,268 INFO mapreduce.Job: Counters: 54
File System Counters
  FILE: Number of bytes read=8053
  FILE: Number of bytes written=1223269
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=115825
  HDFS: Number of bytes written=7524
  HDFS: Number of read operations=17
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
  HDFS: Number of bytes read erasure-coded=0
Job Counters

```

Fig. 1.4.2: Successfully running of Mapper and Reducer

After successful completion of the whole MapReduce function the output file stored in the designated location mentioned, and to access it from the Hadoop cluster we need to use the **-cat** function.



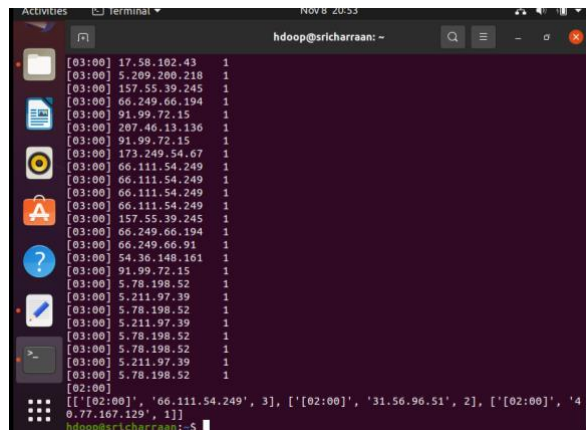
```

file Output format Counters
Bytes Written=7524
2022-11-08 20:51:29,269 INFO streaming.StreamJob: Output directory: /user/hadoop
/assignment_4/output_MapReduce_task2.1.1
hadoop@sricharraan:~$ hadoop fs -cat /user/hadoop/assignment_4/output_MapReduce_t
ask2.1.1/part-00000
[02:00] 40.77.167.129 1
[02:00] 31.56.96.51 1
[02:00] 31.56.96.51 1
[02:00] 66.111.54.249 1
[02:00] 66.111.54.249 1
[02:00] 66.111.54.249 1
[02:00] 0 54.36.149 1
[03:00] 207.46.13.115 1
[03:00] 66.249.66.91 1
[03:00] 173.249.54.67 1
[03:00] 5.78.198.52 1
[03:00] 207.46.13.115 1
[03:00] 207.46.13.115 1
[03:00] 173.249.54.67 1
[03:00] 66.249.66.194 1
[03:00] 66.111.54.249 1
[03:00] 207.46.13.115 1
[03:00] 5.112.52.254 1
[03:00] 66.111.54.249 1
[03:00] 5.112.52.254 1
[03:00] 173.249.54.67 1
[03:00] 66.111.54.249 1

```

Fig. 1.4.3: End MapReduce function and accessing output file

The output of the whole MapReduce function can be seen in the Fig. 1.4.4. The output shows the top 3 IP address from 2:00 – 3:00.



```

[03:00] 17.58.102.43 1
[03:00] 5.209.200.218 1
[03:00] 157.55.39.245 1
[03:00] 66.249.66.194 1
[03:00] 91.99.72.15 1
[03:00] 207.46.13.136 1
[03:00] 91.99.72.15 1
[03:00] 173.249.54.67 1
[03:00] 66.111.54.249 1
[03:00] 66.111.54.249 1
[03:00] 66.111.54.249 1
[03:00] 157.55.39.245 1
[03:00] 66.249.66.194 1
[03:00] 66.249.66.91 1
[03:00] 54.36.148.161 1
[03:00] 91.99.72.15 1
[03:00] 5.78.198.52 1
[03:00] 5.211.97.39 1
[03:00] 5.78.198.52 1
[03:00] 5.211.97.39 1
[03:00] 5.78.198.52 1
[03:00] 5.78.198.52 1
[03:00] 5.211.97.39 1
[03:00] 5.78.198.52 1
[02:00]
[["[02:00]", "66.111.54.249", 3], ["[02:00]", "31.56.96.51", 2], ["[02:00]", "4
0.77.167.129", 1]]
hadoop@sricharraan:~$

```

Fig. 1.4.4: Output of Database Search for 2:00 - 3:00

## 1.5 GitHub Assignment Link

<https://github.com/sricharraan/ENGR-E-516-Cloud-Computing/tree/main/Assignment1>

## 1.5 References:

1. <https://phoenixnap.com/kb/install-hadoop-ubuntu>
2. <https://www.youtube.com/watch?v=mhisPejz-6c&t=788s>
3. [https://www.youtube.com/watch?v=9CW8rD\\_MF\\_o&list=PLyIBsOS1de9igxKhZJNpB32sxLTKY43\\_j&index=15&t=764s](https://www.youtube.com/watch?v=9CW8rD_MF_o&list=PLyIBsOS1de9igxKhZJNpB32sxLTKY43_j&index=15&t=764s)