## VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

Linux Internals Course based project Documentation

PCMONITOR

**Abstract:**

This project addresses the problems of a system administrator in constantly monitoring the system resources. Insufficient resources can lead to poor performance of the system and an immediate action is to be taken by the administrator to ensure the performance. Since administrator is a busy person and he cannot monitor the system all day, this project will automate one of his system monitoring tasks and it will notify him through email even when he is not at the system. In this way the system administrator will be notified whenever there is need of his attention for managing the resources.

**Disadvantages of a person manually monitoring the system:**

It is a very tedious task, since no one knows when what can happen, so a person must sit in front of the system and manually monitor it continuously. Insufficient resources can lead to poor performance of the system and an immediate action is to be taken by the administrator to ensure the performance. If he is away from the system, there is no way of notifying him of the issues in the system

**Proposed method:**

The main idea of the project is to write scripts using python and Linux to monitor the system resources like CPU usage, DISK usage, RAM usage and store them in a log file daily. Using this log information whenever there is any abnormal usage of resources or a resource usage is above a fixed threshold a graph image will be generated of that resource and the same image will be sent to the system administrator through Gmail or who so ever responsible for monitoring the resources immediately. In addition, there will also be individual a possibility to see a live feed of the system resources like that of a moving graph in task manager in windows operating system to visually monitor the system resources.

**Advantages of new system:**

In the new system we get a visual feed of the resources being used, which is more easy and simple to understand, and the details are also stored in the log file. Also, whenever there is an issue the administrator is being notified through the email , hence there is no need to sit and continuously monitor the system performance.

**Dependencies to run the code:**

Python version 3 or >3 – install python packages matplotlib, pandas, NumPy, email, smtplib and a proper internet connection.

**Code:**

The code has three different parts

1 The shell script which saves the info into the log file.

2 The python script which creates a graph from the data in the log file.

3 Another python script which sends the email.

Both the python scripts can be run manually also, but we have made sure that they are run by the shell script directly whenever necessary.

**Code:**

Stats.sh

#! /bin/bash

if [ -f "log.txt" ]; then

rm "log.txt"

fi

c1=0

f1=0

c2=0

f2=0

c3=0

f3=0

printf "RAM\tURAM\tSWAP\tUSWAP\tDISK\tUDISK\tCPU\t\n" >>log.txt

end=$((SECONDS+36000))

while [ $SECONDS -lt $end ]; do

RAM=$(free | awk 'NR==2{printf "%.2f\t", $2/(1000\*1000) }')

URAM=$(free | awk 'NR==2{printf "%.2f\t", $3/(1000\*1000) }')

SWAP=$(free | awk 'NR==3{printf "%.2f\t", $2/(1000\*1000)}')

USWAP=$(free | awk 'NR==3{printf "%.2f\t", $3/(1000\*1000) }')

DISK=$(df | awk '$NF=="/"{printf "%.2f\t", $2/(1000\*1000) }')

UDISK=$(df | awk '$NF=="/"{printf "%.2f\t", $3/(1000\*1000) }')

CPU=$(top -bn1 | grep load | awk '{printf "%.2f\t\n", $(NF-2)}')

echo "$RAM$URAM$SWAP$USWAP$DISK$UDISK$CPU" >> log.txt

./cpu.py

./disk.py

./ram.py

if (( $(expr $f1 \> 0 ) )); then

c1=$(expr $c1 + 1)

fi

if (( $(expr $c1 \> 30 ) )); then

f1=0

c1=0

fi

if (( $(expr $CPU \> 4 ) )); then

if (( $(expr $f1 \< 1 ) )); then

f1=1

./mail.py

fi

fi

if (( $(expr $f2 \> 0 ) )); then

c2=$(expr $c2 + 1)

fi

if (( $(expr $c2 \> 30 ) )); then

f2=0

c2=0

fi

if (( $(expr $UDISK \> 1 ) )); then

if (( $(expr $f2 \< 1 ) )); then

f2=1

./maild.py

fi

fi

if (( $(expr $f3 \> 0 ) )); then

c3=$(expr $c3 + 1)

fi

if (( $(expr $c3 \> 30 ) )); then

f3=0

c3=0

fi

if (( $(expr $URAM \> 1 ) )); then

if (( $(expr $f3 \< 1 ) )); then

f3=1

./mailr.py

fi

fi

done

cpu.py

#!/usr/bin/env python

import matplotlib.pyplot as plt

import matplotlib.animation as animation

from matplotlib import style

import pandas as pd

import numpy

fig3 = plt.figure(frameon=False)

plt.grid=(True)

ax4 = fig3.add\_subplot(1,1,1)

def animate():

graph\_data = open('log.txt','r').read()

replaces = graph\_data.replace("\n", "");

xs = []

ys = []

replaces=replaces.split("\t")

newdata=[]

labels=replaces[0:7]

i=0;j=7;

for p in range(len(replaces)//7):

newdata.append(replaces[i:j])

i=j

j=j+7

del(newdata[:1])

ys3=[]

xs=[]

for i in range(len(newdata)):

xs.append(i)

ys3.append(float(newdata[i][6]))

if(len(xs)>100):

xs=xs[-100:]

ys3=ys3[-100:]

ax4.clear();

ax4.plot(xs, ys3,color="red")

plt.title("CPU")

plt.xlabel("TIME")

plt.ylabel("USAGE(%)")

plt.savefig("cpuspike.png")

animate()

ram.py

#!/usr/bin/env python

import matplotlib.pyplot as plt

import matplotlib.animation as animation

from matplotlib import style

import pandas as pd

import numpy

fig = plt.figure(frameon=False)

plt.grid=(True)

ax1 = fig.add\_subplot(1,1,1)

def animate():

graph\_data = open('log.txt','r').read()

replaces = graph\_data.replace("\n", "");

xs = []

ys = []

replaces=replaces.split("\t")

newdata=[]

labels=replaces[0:7]

i=0;j=7;

for p in range(len(replaces)//7):

newdata.append(replaces[i:j])

i=j

j=j+7

del(newdata[:1])

ys3=[]

xs=[]

for i in range(len(newdata)):

xs.append(i)

ys3.append(float(newdata[i][1]))

ax1.clear();

ax1.plot(xs, ys3,color="blue")

plt.title("RAM")

plt.xlabel("TIME")

plt.ylabel("USAGE(GB)")

plt.savefig("ramspike.png")

animate()

disk.py

#!/usr/bin/env python

import matplotlib.pyplot as plt

import matplotlib.animation as animation

from matplotlib import style

import pandas as pd

import numpy

fig1 = plt.figure(frameon=False)

plt.grid=(True)

ax2 = fig1.add\_subplot(1,1,1)

def animate():

graph\_data = open('log.txt','r').read()

replaces = graph\_data.replace("\n", "");

xs = []

ys = []

replaces=replaces.split("\t")

newdata=[]

labels=replaces[0:7]

i=0;j=7;

for p in range(len(replaces)//7):

newdata.append(replaces[i:j])

i=j

j=j+7

del(newdata[:1])

ys3=[]

xs=[]

for i in range(len(newdata)):

xs.append(i)

ys3.append(float(newdata[i][5]))

ax2.clear();

ax2.plot(xs, ys3,color="blue")

plt.title("DISK")

plt.xlabel("TIME")

plt.ylabel("USAGE(GB)")

plt.savefig("diskspike.png")

animate()

mail.py

#!/usr/bin/env python

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

from email.mime.base import MIMEBase

from email import encoders

email\_user = 'automailpy3@gmail.com'

email\_password = 'auto\_mailpy3'

email\_send = 'nikhilsai313@gmail.com'

subject = 'system cpu issue'

msg = MIMEMultipart()

msg['From'] = "Diagnostics"

msg['To'] = "Admin"

msg['Subject'] = subject

body = 'system stats are in the attachment'

msg.attach(MIMEText(body,'plain'))

filename='cpuspike.png'

attachment =open(filename,'rb')

part = MIMEBase('application','octet-stream')

part.set\_payload((attachment).read())

encoders.encode\_base64(part)

part.add\_header('Content-Disposition',"attachment; filename= "+filename)

msg.attach(part)

text = msg.as\_string()

server = smtplib.SMTP('smtp.gmail.com',587)

server.starttls()

server.login(email\_user,email\_password)

server.sendmail(email\_user,email\_send,text)

server.quit()

mailr.py

#!/usr/bin/env python

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

from email.mime.base import MIMEBase

from email import encoders

email\_user = 'automailpy3@gmail.com'

email\_password = 'auto\_mailpy3'

email\_send = 'nikhilsai313@gmail.com'

subject = 'system ram issue'

msg = MIMEMultipart()

msg['From'] = "Diagnostics"

msg['To'] = "Admin"

msg['Subject'] = subject

body = 'system stats are in the attachment'

msg.attach(MIMEText(body,'plain'))

filename='ramspike.png'

attachment =open(filename,'rb')

part = MIMEBase('application','octet-stream')

part.set\_payload((attachment).read())

encoders.encode\_base64(part)

part.add\_header('Content-Disposition',"attachment; filename= "+filename)

msg.attach(part)

text = msg.as\_string()

server = smtplib.SMTP('smtp.gmail.com',587)

server.starttls()

server.login(email\_user,email\_password)

server.sendmail(email\_user,email\_send,text)

server.quit()

maild.py

#!/usr/bin/env python

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

from email.mime.base import MIMEBase

from email import encoders

email\_user = 'automailpy3@gmail.com'

email\_password = 'auto\_mailpy3'

email\_send = 'nikhilsai313@gmail.com'

subject = 'system disk issue'

msg = MIMEMultipart()

msg['From'] = "Diagnostics"

msg['To'] = "Admin"

msg['Subject'] = subject

body = 'system stats are in the attachment'

msg.attach(MIMEText(body,'plain'))

filename='diskspike.png'

attachment =open(filename,'rb')

part = MIMEBase('application','octet-stream')

part.set\_payload((attachment).read())

encoders.encode\_base64(part)

part.add\_header('Content-Disposition',"attachment; filename= "+filename)

msg.attach(part)

text = msg.as\_string()

server = smtplib.SMTP('smtp.gmail.com',587)

server.starttls()

server.login(email\_user,email\_password)

server.sendmail(email\_user,email\_send,text)

server.quit()

**Running of the project:**

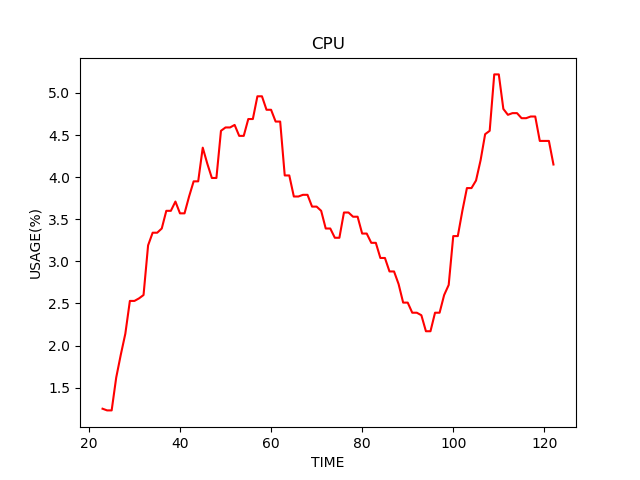
Create and paste all the above code with the specified name and specified extension into one single folder on Linux operating system and then set executable permissions for all the created code files.

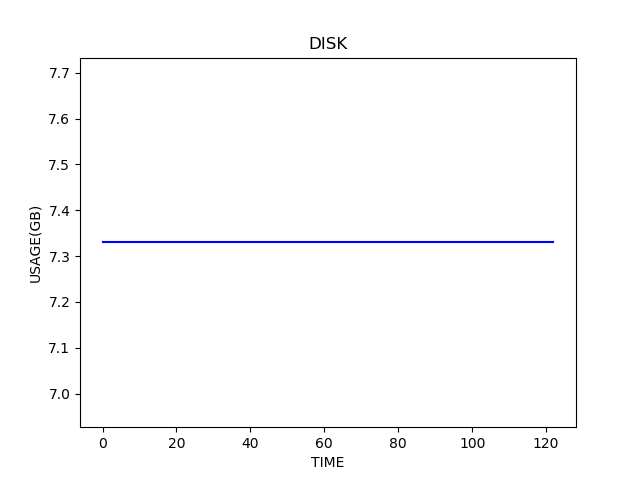
Then open the terminal in that folder and execute the “./stats.sh”.

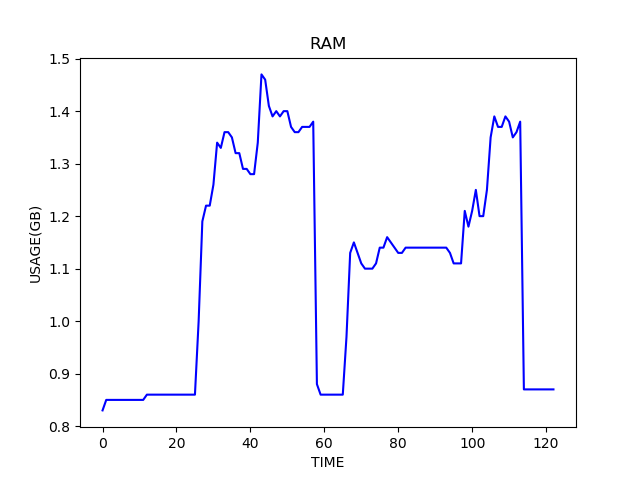
Then open any of the image files cpuspike.png, diskspike.png, ramspike.png,

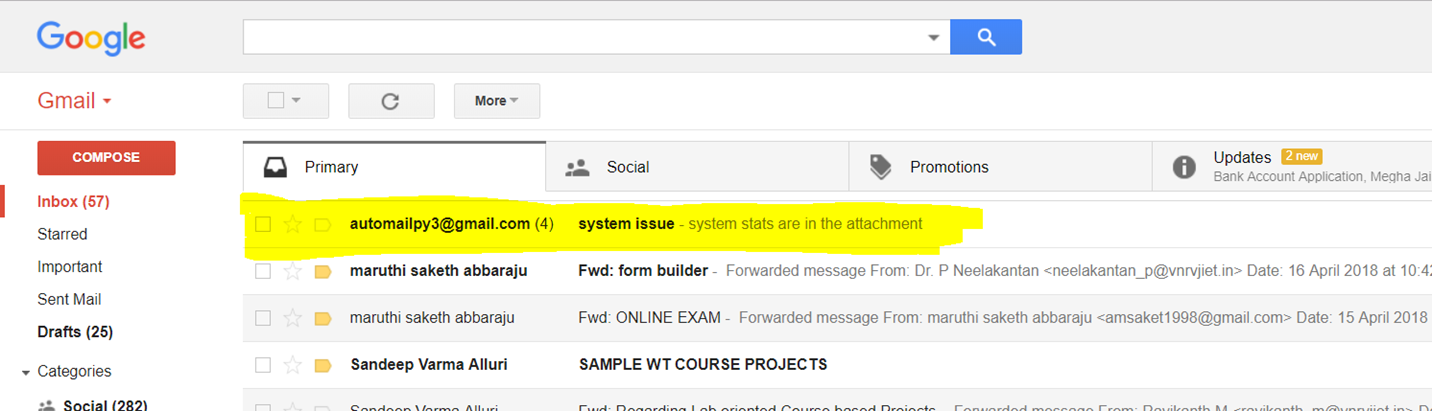
You will be able to see a live feed of these resources. Whenever a certain resource exceeds a certain specified threshold set in the shell script “stast.sh” an email will be sent to the person whose emailed is given in the “mail.py” file.

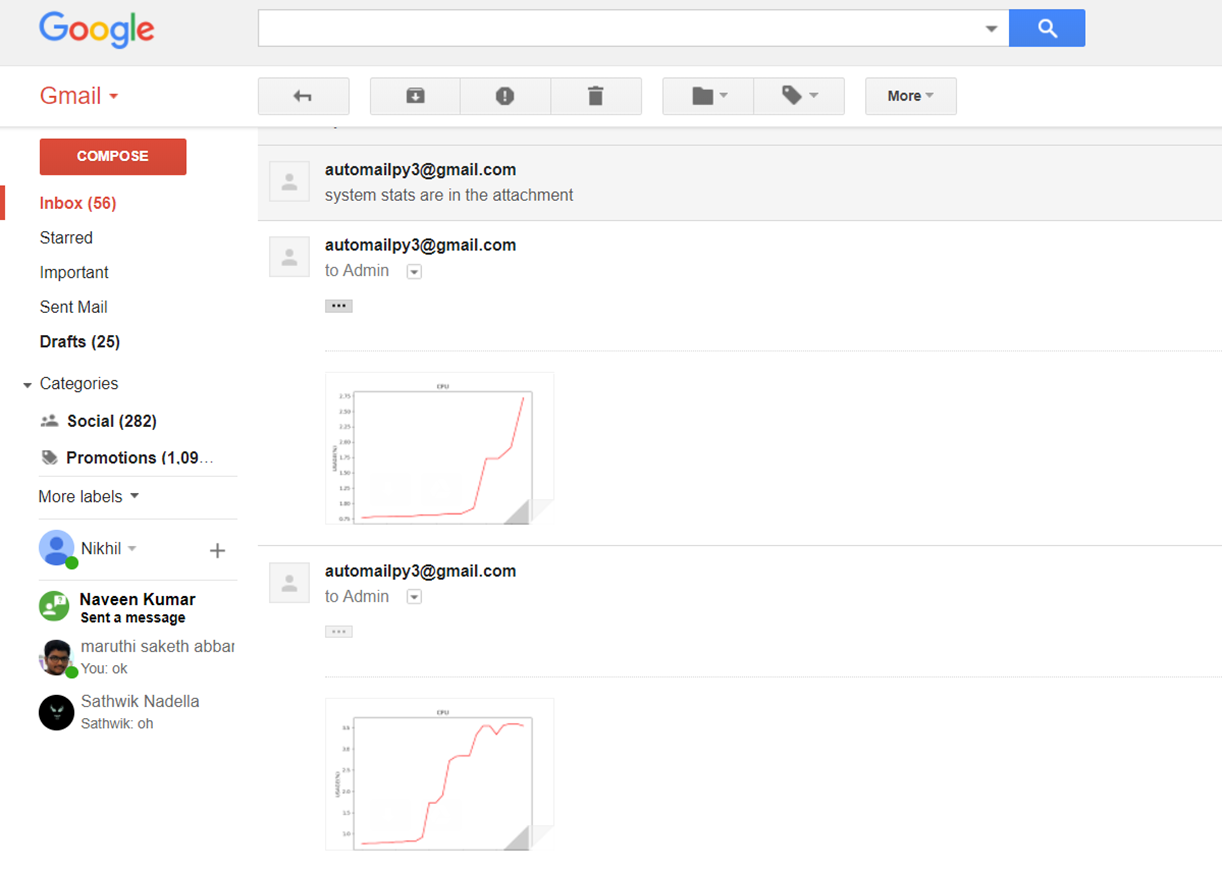
**Results:**











**Conclusion:**

This project has automated the system monitoring task of a system administrator, using this he can not only monitor cpu,ram,disk but also any resource on the linux operating system.