**1.System Health Monitoring script**

import psutil

import logging

from datetime import datetime

# Set up logging

logging.basicConfig(filename='system\_health.log', level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

# Define thresholds

CPU\_THRESHOLD = 80 # in percent

MEMORY\_THRESHOLD = 80 # in percent

DISK\_THRESHOLD = 80 # in percent

def check\_cpu():

cpu\_usage = psutil.cpu\_percent(interval=1)

if cpu\_usage > CPU\_THRESHOLD:

logging.warning(f'High CPU usage detected: {cpu\_usage}%')

return cpu\_usage

def check\_memory():

memory = psutil.virtual\_memory()

memory\_usage = memory.percent

if memory\_usage > MEMORY\_THRESHOLD:

logging.warning(f'High Memory usage detected: {memory\_usage}%')

return memory\_usage

def check\_disk():

disk = psutil.disk\_usage('/')

disk\_usage = disk.percent

if disk\_usage > DISK\_THRESHOLD:

logging.warning(f'Low Disk space detected: {disk\_usage}%')

return disk\_usage

def check\_processes():

processes = [proc.info for proc in psutil.process\_iter(attrs=['pid', 'name', 'username'])]

return processes

def monitor\_system():

cpu\_usage = check\_cpu()

memory\_usage = check\_memory()

disk\_usage = check\_disk()

processes = check\_processes()

logging.info(f'System Health - CPU: {cpu\_usage}%, Memory: {memory\_usage}%, Disk: {disk\_usage}%')

for process in processes:

logging.debug(f'Process - PID: {process["pid"]}, Name: {process["name"]}, User: {process["username"]}')

if \_name\_ == "\_main\_":

monitor\_system()

**2.Automated backup solution**

import subprocess

import logging

from datetime import datetime

# Set up logging

logging.basicConfig(filename='backup.log', level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

# Configuration

SOURCE\_DIR = '/path/to/source'

DESTINATION\_DIR = 'user@remote\_server:/path/to/destination'

def run\_backup():

try:

result = subprocess.run(['rsync', '-avz', SOURCE\_DIR, DESTINATION\_DIR], check=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

logging.info(f'Backup successful: {result.stdout}')

except subprocess.CalledProcessError as e:

logging.error(f'Backup failed: {e.stderr}')

if \_name\_ == "\_main\_":

run\_backup()