import re

import base64

import subprocess

from datetime import datetime

from openpyxl import Workbook

def decode\_base64(s):

    """Decode a string using base64."""

    return base64.b64decode(s.encode('utf-8')).decode('utf-8')

def read\_credentials(filename):

    """Read credentials from a file."""

    credentials = []

    with open(filename, 'r') as file:

        block = {}

        for line in file:

            line = line.strip()

            if line:

                key, value = line.split('=', 1)

                block[key] = value.strip('\"')

            elif block:

                credentials.append(block)

                block = {}

        if block: # to handle no newline at the end

            credentials.append(block)

    return credentials

all\_creds = read\_credentials('/root/automation/cred.txt')

# Generate timestamp for the report filename

timestamp = datetime.now().strftime("%Y-%m-%d\_%H-%M-%S")

excel\_filename = f'status-{timestamp}.xlsx'

wfilename = f'Healthcheck-{timestamp}.txt'

# Create a new workbook

print("Creating Excel workbook...")

wb = Workbook()

ws = wb.active

# Writing headers

ws.append(["IP", "Disk Utilization Status", "Available Memory Status", "Uptime Status", "Chrony Status", "Node Status", "Pod Status"])

print("\t\t\*\*\*\*\*HealthCheck Report\*\*\*\*\*")

with open(wfilename, 'w') as wfile:

    for creds in all\_creds:

        IP = creds['IP']

        USERNAME = creds['USERNAME']

        PASSWORD = decode\_base64(creds.get('PASSWORD', ''))

        K8s\_Master = creds['K8s\_Master']

        print(f"Performing health check for {IP}...")

        disk\_status = "Pass"

        memory\_status = "Pass"

        uptime\_status = "Pass"

        leap\_status = "Pass"

        # Disk utilization check

        disk\_cmd = f'sshpass -p "{PASSWORD}" ssh -o StrictHostKeyChecking=no {USERNAME}@{IP} "df -h"'

        disk\_util = subprocess.check\_output(disk\_cmd, shell=True, stderr=subprocess.STDOUT).decode('utf-8')

        pattern = r'^\S+\s+\S+\s+\S+\s+\S+\s+(\d+)%\s+\S+$' # Updated pattern to match the "Use%" value

        # Iterate over each line of the output

        for line in disk\_util.split('\n'):

            match = re.search(pattern, line)

            if match:

                utilization = int(match.group(1))

                if utilization >= 90:

                    disk\_status = "Fail"

                    break # If any line meets the condition, we can break out of the loop

        # Memory check

        mem\_cmd = f'sshpass -p "{PASSWORD}" ssh -o StrictHostKeyChecking=no {USERNAME}@{IP} "free -g"'

        mem\_info = subprocess.check\_output(mem\_cmd, shell=True, stderr=subprocess.STDOUT).decode('utf-8')

        pattern = r'^Mem:\s+(\d+)\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+$' # Match total memory

        match = re.search(pattern, mem\_info, re.MULTILINE)

        if match:

            total\_memory = int(match.group(1))

        pattern = r'^Mem:\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+(\d+)$' # Match available memory

        match = re.search(pattern, mem\_info, re.MULTILINE)

        if match:

            available\_memory = int(match.group(1))

            if available\_memory < 0.5 \* total\_memory:

                memory\_status = "Fail"

        # Uptime check

        uptime\_cmd = f'sshpass -p "{PASSWORD}" ssh -o StrictHostKeyChecking=no {USERNAME}@{IP} "uptime"'

        uptime\_info = subprocess.check\_output(uptime\_cmd, shell=True, stderr=subprocess.STDOUT).decode('utf-8')

        pattern = r'up\s+(\d+)\s+day'

        uptime\_match = re.search(pattern, uptime\_info)

        if uptime\_match:

            days = int(uptime\_match.group(1))

            if days <= 5:

                uptime\_status = "Fail"

        #NTP configuration

        chrony\_cmd = f'sshpass -p "{PASSWORD}" ssh -o StrictHostKeyChecking=no {USERNAME}@{IP} "chronyc tracking"'

        chrony\_info = subprocess.check\_output(chrony\_cmd, shell=True, stderr=subprocess.STDOUT).decode('utf-8')

        leap\_pattern = r'Leap status\s+:\s+(\w+)'

        leap\_match = re.search(leap\_pattern, chrony\_info)

        if leap\_match:

            leap\_status = "Fail" if leap\_match.group(1) != "Normal" else "Pass"

        # CPU Info

        cpu\_cmd = f'sshpass -p "{PASSWORD}" ssh -o StrictHostKeyChecking=no {USERNAME}@{IP} "lscpu"'

        cpu\_info = subprocess.check\_output(cpu\_cmd, shell=True, stderr=subprocess.STDOUT).decode('utf-8')

        # Print results

        wfile.write("\t\tHealthcheck status of " + IP + "\n")

        wfile.write("Disk utilization:\n" + disk\_util + "\n")

        wfile.write("CPU info:\n" + cpu\_info + "\n")

        wfile.write("Memory info:\n" + mem\_info + "\n")

        wfile.write("Uptime info:\n" + uptime\_info + "\n")

        wfile.write("Chrony Status:\n" + chrony\_info + "\n")

        #ws.append([IP, disk\_status, memory\_status, uptime\_status])

#################### K8s Check ################################

# Execute additional commands for Kubernetes master

        node\_status = "NA"

        pod\_status = "NA"

        component\_status = "NA"

        if K8s\_Master.lower() == 'yes':

            node\_status = "Pass"

            pod\_status = "Pass"

            kubectl\_get\_nodes\_cmd = f'sshpass -p "{PASSWORD}" ssh -o StrictHostKeyChecking=no {USERNAME}@{IP} "kubectl get nodes"'

            kubectl\_get\_nodes\_output = subprocess.check\_output(kubectl\_get\_nodes\_cmd, shell=True, stderr=subprocess.STDOUT).decode('utf-8')

            # Define pattern to match node status

            pattern = re.compile(r'^\S+\s+(\S+)\s+\S+\s+\S+\s+\S+')

            # Search for node status in the output

            for line in kubectl\_get\_nodes\_output.split('\n')[1:]:

                if line.strip():

                    match = pattern.match(line)

                    if match:

                        x = match.group(1)

                        if x != "Ready":

                            node\_status = "Fail"

                            break

            # Write node status to the file

            wfile.write("Nodes:\n" + kubectl\_get\_nodes\_output + "\n")

            kubectl\_get\_pod\_cmd = f'sshpass -p "{PASSWORD}" ssh -o StrictHostKeyChecking=no {USERNAME}@{IP} "kubectl get pod -n kube-system"'

            kubectl\_get\_pod\_output = subprocess.check\_output(kubectl\_get\_pod\_cmd, shell=True, stderr=subprocess.STDOUT).decode('utf-8')

            pod\_status\_pattern = re.compile(r'^\S+\s+\S+\s+(\S+)\s+\S+\s+\S+\s+\S+')

            # Iterate over each line of the output

            for line in kubectl\_get\_pod\_output.split('\n'):

                match = pod\_status\_pattern.match(line)

                if match:

                    pod\_status\_value = match.group(1)

                    if pod\_status\_value != "Running":

                        pod\_status = "Fail"

                        break

            wfile.write("Pods in all namespaces:\n" + kubectl\_get\_pod\_output + "\n")

            kubectl\_get\_pvc\_cmd = f'sshpass -p "{PASSWORD}" ssh -o StrictHostKeyChecking=no {USERNAME}@{IP} "kubectl get pvc --all-namespaces"'

            kubectl\_get\_pvc\_output = subprocess.check\_output(kubectl\_get\_pvc\_cmd, shell=True, stderr=subprocess.STDOUT).decode('utf-8')

            wfile.write("PVCs in all namespaces:\n" + kubectl\_get\_pvc\_output + "\n")

            # Execute the command to get component statuses

            kubectl\_get\_componentstatuses\_cmd = f'sshpass -p "{PASSWORD}" ssh -o StrictHostKeyChecking=no {USERNAME}@{IP} "kubectl get componentstatuses"'

            kubectl\_get\_componentstatuses\_output = subprocess.check\_output(kubectl\_get\_componentstatuses\_cmd, shell=True, stderr=subprocess.STDOUT).decode('utf-8')

            wfile.write("Component Statuses:\n" + kubectl\_get\_componentstatuses\_output + "\n")

            wfile.write("##########################################################################################\n")

    # Write data to Excel

        ws.append([IP, disk\_status, memory\_status, uptime\_status, leap\_status, node\_status, pod\_status])

wb.save(excel\_filename)

print("Excel workbook saved successfully.")

print("Health check report has been saved in", wfilename, "file.")