**1. Definir el alcance de Monitoreo:**

**Servicios de Sistema Operativo:**

* **CPU:** Uso de la CPU, carga promedio.
* **Memoria:** Uso de la memoria, memoria libre/disponible.
* **Filesystem:** Espacio en disco, estado de los archivos.

**Bases de Datos:**

* **Validar conexión a la BD:** Verificar si la base de datos está accesible.
* **Inválidos:** Chequear integridad de datos.
* **Sesiones Activas e Inactivas:** Monitorizar el número de sesiones conectadas.

**Aplicaciones:**

* **Weblogic, Tomcat, y Nginx:** Verificar que los servicios estén corriendo y respondiendo adecuadamente.

**2. Elegir Herramientas de Monitoreo:**

**Bash o Python Scripts:**

* Para realizar chequeos personalizados en el SO y BD.

**3. Automatización de Validaciones:**

**Cron (Linux):**

* Usar cron jobs para ejecutar los scripts de monitoreo a intervalos regulares.

**4. Generación de Reporte .html:**

**Python (jinja2):**

* Utilizar librerías como jinja2 para generar plantillas HTML. Aquí un ejemplo básico:

from jinja2 import Environment, FileSystemLoader

import os

# Cargar plantillas

file\_loader = FileSystemLoader('templates')

env = Environment(loader=file\_loader)

# Cargar plantilla HTML

template = env.get\_template('reporte.html')

# Datos para el reporte

data = {

'cpu': '20%',

'memoria': '50%',

'filesystem': '70%',

'bd\_conexion': 'Exitosa',

'sesiones\_activas': 5,

'sesiones\_inactivas': 2,

'app\_status': 'Running'

}

# Renderizar el HTML con los datos

output = template.render(data=data)

# Guardar el reporte

with open('reporte.html', 'w') as file:

file.write(output)

5. Envío de Reporte por Correo Electrónico:

**Python (smtplib):**

* Usar smtplib para enviar el correo con el archivo .html adjunto.:

**1. Configuración del Entorno de Trabajo**

**Crear el Directorio del Proyecto:**

Si aún no lo has hecho, crea el directorio y navega hasta él:

mkdir -p /home/sisadmin/Monitoring

cd /home/sisadmin/Monitoring

**Organización de la Estructura de Directorios:**

Te recomiendo organizar tu proyecto de la siguiente manera para mantener todo bien estructurado:

/home/sisadmin/Monitoring

├── scripts/ # Aquí estarán tus scripts Python

├── templates/ # Plantillas HTML para los reportes

├── logs/ # Archivos de logs y registros

└── config/ # Archivos de configuración (opcional)

mkdir scripts templates logs config

**Índice**

1. Estructura del Proyecto
2. Configuración del Entorno
3. Instalación de Dependencias
4. Script Principal (main.py)
5. Plantilla HTML (reporte.html)
6. Archivo de Configuración (config.ini)
7. Configuración de Variables de Entorno
8. Programación con Cron
9. Notas Importantes
10. Pasos Finales y Pruebas

**1. Estructura del Proyecto**

En el directorio /home/sisadmin/Monitoring, organiza tu proyecto de la siguiente manera:

/home/sisadmin/Monitoring

├── scripts/

│ └── main.py

├── templates/

│ └── reporte.html

├── config/

│ └── config.ini

├── logs/

│ └── monitoring.log

├── venv/ (entorno virtual de Python)

**2. Configuración del Entorno**

**a. Crear el directorio del proyecto y sus subdirectorios:**

mkdir -p /home/sisadmin/Monitoring/{scripts,templates,config,logs}

cd /home/sisadmin/Monitoring

**b. Crear un entorno virtual de Python (recomendado):**

python3 -m venv venv

source venv/bin/actívate

**3. Instalación de Dependencias**

Instala las bibliotecas necesarias en el entorno virtual:

pip install psutil cx\_Oracle jinja2

[sisadmin@coclocccprdl16 Monitoring]$ source venv/bin/activate

(venv) [sisadmin@coclocccprdl16 Monitoring]$

(venv) [sisadmin@coclocccprdl16 Monitoring]$

(venv) [sisadmin@coclocccprdl16 Monitoring]$ pip install psutil cx\_Oracle jinja2

Collecting psutil

Downloading psutil-7.0.0-cp36-abi3-manylinux\_2\_12\_x86\_64.manylinux2010\_x86\_64.manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl (277 kB)

|████████████████████████████████| 277 kB 2.4 MB/s

Collecting cx\_Oracle

Downloading cx\_Oracle-8.3.0-cp39-cp39-manylinux\_2\_5\_x86\_64.manylinux1\_x86\_64.manylinux\_2\_12\_x86\_64.manylinux2010\_x86\_64.whl (888 kB)

|████████████████████████████████| 888 kB 51.3 MB/s

Collecting jinja2

Downloading jinja2-3.1.5-py3-none-any.whl (134 kB)

|████████████████████████████████| 134 kB 80.5 MB/s

Collecting MarkupSafe>=2.0

Downloading MarkupSafe-3.0.2-cp39-cp39-manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl (20 kB)

Installing collected packages: MarkupSafe, psutil, jinja2, cx-Oracle

Successfully installed MarkupSafe-3.0.2 cx-Oracle-8.3.0 jinja2-3.1.5 psutil-7.0.0

WARNING: You are using pip version 21.2.3; however, version 25.0.1 is available.

You should consider upgrading via the '/home/sisadmin/Monitoring/venv/bin/python3 -m pip install --upgrade pip' command.

(venv) [sisadmin@coclocccprdl16 Monitoring]$

(venv) [sisadmin@coclocccprdl16 Monitoring]$ python3 -m pip install --upgrade pip

Requirement already satisfied: pip in ./venv/lib/python3.9/site-packages (21.2.3)

Collecting pip

Downloading pip-25.0.1-py3-none-any.whl (1.8 MB)

|████████████████████████████████| 1.8 MB 2.3 MB/s

Installing collected packages: pip

Attempting uninstall: pip

Found existing installation: pip 21.2.3

Uninstalling pip-21.2.3:

Successfully uninstalled pip-21.2.3

Successfully installed pip-25.0.1

(venv) [sisadmin@coclocccprdl16 Monitoring]$

**Nota:** cx\_Oracle requiere que tengas instalado el **Oracle Instant Client**. Asegúrate de descargarlo e instalarlo desde Oracle Instant Client Downloads

**4. Script Principal (main.py)**

Crea el archivo /home/sisadmin/Monitoring/scripts/main.py con el siguiente contenido:

import os

import psutil

import cx\_Oracle

from jinja2 import Environment, FileSystemLoader

from datetime import datetime

# Funciones de Monitoreo del Sistema Operativo

def obtener\_estado\_cpu():

uso\_cpu = psutil.cpu\_percent(interval=1)

return 'okay' if uso\_cpu < 80 else 'Fallido'

def obtener\_estado\_memoria():

uso\_memoria = psutil.virtual\_memory().percent

return 'okay' if uso\_memoria < 80 else 'Fallido'

def obtener\_estado\_filesystem():

usos = psutil.disk\_partitions()

for particion in usos:

if 'rw' in particion.opts:

uso = psutil.disk\_usage(particion.mountpoint).percent

if uso > 90:

return 'Fallido'

return 'okay'

def obtener\_clase\_estado(estado):

estado = estado.lower()

if estado == 'okay':

return 'okay'

elif estado == 'fallido':

return 'fallido'

else:

return 'n-a'

# Funciones de Monitoreo de la Base de Datos Oracle

def validar\_conexion\_bd():

try:

dsn\_tns = cx\_Oracle.makedsn(os.getenv('ORACLE\_HOST'), int(os.getenv('ORACLE\_PORT')), service\_name=os.getenv('ORACLE\_SID'))

connection = cx\_Oracle.connect(user=os.getenv('ORACLE\_USER'), password=os.getenv('ORACLE\_PASSWORD'), dsn=dsn\_tns)

connection.close()

return "Exitosa"

except Exception as e:

return f"Fallida: {e}"

def obtener\_sesiones\_bd():

sesiones = {'activas': 'N/A', 'inactivas': 'N/A'}

try:

dsn\_tns = cx\_Oracle.makedsn(os.getenv('ORACLE\_HOST'), int(os.getenv('ORACLE\_PORT')), service\_name=os.getenv('ORACLE\_SID'))

connection = cx\_Oracle.connect(user=os.getenv('ORACLE\_USER'), password=os.getenv('ORACLE\_PASSWORD'), dsn=dsn\_tns)

cursor = connection.cursor()

cursor.execute("SELECT COUNT(\*) FROM v$session WHERE status = 'ACTIVE'")

sesiones['activas'] = cursor.fetchone()[0]

cursor.execute("SELECT COUNT(\*) FROM v$session WHERE status = 'INACTIVE'")

sesiones['inactivas'] = cursor.fetchone()[0]

cursor.close()

connection.close()

except Exception as e:

print(f"Error al obtener sesiones: {e}")

return sesiones

# Función para Verificar Servicios de Aplicaciones

def verificar\_app(servicio):

for proc in psutil.process\_iter(['name']):

if servicio.lower() in proc.info['name'].lower():

return 'okay'

return 'Fallido'

# Función para Generar el Reporte HTML

def generar\_reporte(servers):

file\_loader = FileSystemLoader('/home/sisadmin/Monitoring/templates')

env = Environment(loader=file\_loader)

template = env.get\_template('reporte.html')

output = template.render(servers=servers, fecha=datetime.now().strftime('%Y-%m-%d %H:%M:%S'))

with open('/home/sisadmin/Monitoring/reporte.html', 'w', encoding='utf-8') as file:

file.write(output)

# Función Principal de Monitoreo

def monitorear\_servidor(servidor):

# Monitoreo del SO

cpu\_status = obtener\_estado\_cpu()

memoria\_status = obtener\_estado\_memoria()

filesystem\_status = obtener\_estado\_filesystem()

# Monitoreo de la BD

if servidor['mon\_bd']:

bd\_status = validar\_conexion\_bd()

sesiones = obtener\_sesiones\_bd()

sesiones\_activas = sesiones['activas']

sesiones\_inactivas = sesiones['inactivas']

else:

bd\_status = 'N/A'

sesiones\_activas = 'N/A'

sesiones\_inactivas = 'N/A'

# Monitoreo de la Aplicación

if servidor['mon\_app']:

app\_status = verificar\_app(servidor['app\_name'])

else:

app\_status = 'N/A'

data = {

'ip': servidor['ip'],

'cpu\_status': cpu\_status,

'cpu\_status\_class': obtener\_clase\_estado(cpu\_status),

'memoria\_status': memoria\_status,

'memoria\_status\_class': obtener\_clase\_estado(memoria\_status),

'filesystem\_status': filesystem\_status,

'filesystem\_status\_class': obtener\_clase\_estado(filesystem\_status),

'bd\_status': bd\_status,

'bd\_status\_class': obtener\_clase\_estado(bd\_status),

'app\_status': app\_status,

'app\_status\_class': obtener\_clase\_estado(app\_status),

'sesiones\_activas': sesiones\_activas,

'sesiones\_inactivas': sesiones\_inactivas

}

return data

# Función para Enviar el Reporte por Correo Electrónico

def enviar\_email():

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

fromaddr = os.getenv('SMTP\_USER')

toaddr = os.getenv('EMAIL\_RECIPIENT')

msg = MIMEMultipart()

msg['From'] = fromaddr

msg['To'] = toaddr

msg['Subject'] = "Reporte de Monitoreo"

# Adjuntar el reporte HTML en el cuerpo del mensaje

with open('/home/sisadmin/Monitoring/reporte.html', 'r', encoding='utf-8') as file:

reporte\_html = file.read()

msg.attach(MIMEText(reporte\_html, 'html'))

server = smtplib.SMTP(os.getenv('SMTP\_SERVER'), int(os.getenv('SMTP\_PORT')))

server.starttls()

server.login(fromaddr, os.getenv('SMTP\_PASSWORD'))

text = msg.as\_string()

server.sendmail(fromaddr, toaddr.split(','), text)

server.quit()

# Función Principal

def main():

# Cargar las variables de entorno desde el archivo de configuración

from dotenv import load\_dotenv

load\_dotenv('/home/sisadmin/Monitoring/config/config.ini')

# Configuración de los servidores a monitorear

servidores = [

{'ip': '192.168.1.10', 'mon\_bd': True, 'mon\_app': False, 'app\_name': ''},

{'ip': '192.168.1.11', 'mon\_bd': False, 'mon\_app': True, 'app\_name': 'nginx'},

{'ip': '192.168.1.12', 'mon\_bd': True, 'mon\_app': False, 'app\_name': ''},

{'ip': '192.168.1.13', 'mon\_bd': False, 'mon\_app': True, 'app\_name': 'tomcat'},

]

servers = []

for servidor in servidores:

data = monitorear\_servidor(servidor)

servers.append(data)

generar\_reporte(servers)

enviar\_email()

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

main()

**Explicaciones:**

* **Variables de Entorno:** Utilizamos os.getenv() para obtener las credenciales y configuraciones sensibles desde el archivo de configuración o variables de entorno.
* **Monitoreo de la BD Oracle:** Utilizamos cx\_Oracle para conectar y obtener las sesiones activas e inactivas.
* **Monitoreo de Aplicaciones:** La función verificar\_app verifica si el proceso de la aplicación está corriendo.
* **Generación de Reporte:** La función generar\_reporte utiliza Jinja2 para renderizar el reporte HTML.
* **Envío de Correo:** La función enviar\_email envía el reporte por correo electrónico utilizando smtplib.

**5. Plantilla HTML (reporte.html)**

Crea el archivo /home/sisadmin/Monitoring/templates/reporte.html con el siguiente contenido:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<title>Estado de Salud Servidores</title>

<style>

body {

font-family: Arial, sans-serif;

}

h1 {

text-align: center;

color: #333333;

}

p {

text-align: center;

color: #333333;

}

table {

border-collapse: collapse;

width: 90%;

margin: 0 auto;

}

th, td {

border: 1px solid #dddddd;

text-align: center;

padding: 8px;

}

th {

background-color: #f2f2f2;

color: #333333;

}

.okay {

color: green;

font-weight: bold;

}

.fallido {

color: red;

font-weight: bold;

}

.n-a {

color: gray;

font-style: italic;

}

</style>

</head>

<body>

<h1>Estado de Salud Servidores</h1>

<p>Reporte generado: {{ fecha }}</p>

<table>

<thead>

<tr>

<th>Servidor</th>

<th>CPU</th>

<th>Memoria</th>

<th>Filesystem</th>

<th>BD</th>

<th>Sesiones Activas</th>

<th>Sesiones Inactivas</th>

<th>App (Tomcat, Nginx o Weblogic)</th>

</tr>

</thead>

<tbody>

{% for server in servers %}

<tr>

<td>{{ server.ip }}</td>

<td class="{{ server.cpu\_status\_class }}">{{ server.cpu\_status }}</td>

<td class="{{ server.memoria\_status\_class }}">{{ server.memoria\_status }}</td>

<td class="{{ server.filesystem\_status\_class }}">{{ server.filesystem\_status }}</td>

<td class="{{ server.bd\_status\_class }}">{{ server.bd\_status }}</td>

<td>{{ server.sesiones\_activas }}</td>

<td>{{ server.sesiones\_inactivas }}</td>

<td class="{{ server.app\_status\_class }}">{{ server.app\_status }}</td>

</tr>

{% endfor %}

</tbody>

</table>

</body>

</html>

**Notas:**

* La plantilla muestra todos los datos relevantes, incluyendo sesiones activas e inactivas.
* Las clases CSS (okay, fallido, n-a) estilizan las celdas según el estado.

**6. Archivo de Configuración (config.ini)**

Crea el archivo **/home/sisadmin/Monitoring/config/config.ini** con el siguiente contenido:

# Configuración del servidor SMTP

SMTP\_SERVER=tu\_servidor\_smtp

SMTP\_PORT=587

SMTP\_USER=tu\_correo@example.com

SMTP\_PASSWORD=tu\_contraseña\_correo

# Destinatarios del correo electrónico (separados por comas)

EMAIL\_RECIPIENT=destino1@example.com,destino2@example.com

# Credenciales de la base de datos (comunes a todas las bases de datos)

ORACLE\_USER=system

ORACLE\_PASSWORD=ZSbooyioUkE1tlFT3NY2

[DATABASE\_SERVERS]

# Lista de servidores de bases de datos

SERVERS=DB1,DB2,DB3

[DB1]

HOST=192.168.202.4

PORT=1521

SID=REGISTRO

IP=192.168.202.4 # IP del servidor correspondiente

[DB2]

HOST=192.168.202.3

PORT=1641

SID=ERP

IP=192.168.202.3

[DB3]

HOST=192.168.202.2

PORT=1521

SID=DOCUNET

IP=192.168.202.2

[APPLICATION\_SERVERS]

# Lista de servidores de aplicaciones

SERVERS=APP1,APP2

[APP1]

IP=192.168.202.5

APP\_NAME=weblogic

[APP2]

IP=192.168.202.6

APP\_NAME=nginx

**Notas Importantes:**

* **Seguridad:** Asegúrate de proteger este archivo, ya que contiene credenciales sensibles.
* **Permisos:** Establece permisos restrictivos.

**chmod 600 /home/sisadmin/Monitoring/config/config.ini**

## ****7. Configuración de Variables de Entorno****

Para que el script pueda leer las variables de configuración, usaremos la librería python-dotenv. Instálala:

pip install python-dotenv

Luego, modifica la sección de importación en main.py:

from dotenv import load\_dotenv

load\_dotenv('/home/sisadmin/Monitoring/config/config.ini')

## ****8. Programación con Cron****

Edita el crontab para ejecutar el script automáticamente:

crontab –e

0 8 \* \* \* source /home/sisadmin/Monitoring/venv/bin/activate && /home/sisadmin/Monitoring/venv/bin/python /home/sisadmin/Monitoring/scripts/main.py >> /home/sisadmin/Monitoring/logs/monitoring.log 2>&1

9. Notas Importantes

chmod +x /home/sisadmin/Monitoring/scripts/main.py

**10. Pasos Finales y Pruebas**

**a. Verificar la Conexión a la Base de Datos**

Ejecuta una prueba de conexión para asegurarte de que cx\_Oracle funciona correctamente:

python

import cx\_Oracle

dsn\_tns = cx\_Oracle.makedsn('localhost', 1521, service\_name='XE')

connection = cx\_Oracle.connect(user='tu\_usuario', password='tu\_contraseña', dsn=dsn\_tns)

print("Conexión exitosa")

connection.close()

**b. Ejecutar el Script Manualmente**

Ejecuta el script principal para verificar que todo funciona:

bash

source /home/sisadmin/Monitoring/venv/bin/activate

python /home/sisadmin/Monitoring/scripts/main.py

**c. Verificar el Reporte**

Revisa el archivo /home/sisadmin/Monitoring/reporte.html para asegurarte de que los datos se muestran correctamente.

**d. Comprobar el Envío de Correo**

Confirma que el correo electrónico ha sido enviado y recibido, y que el reporte se ve bien en el cuerpo del correo.

**e. Revisar el Log**

Consulta el archivo /home/sisadmin/Monitoring/logs/monitoring.log para identificar posibles errores o advertencias.

**Consideraciones Finales**

* **Manejo de Errores:**
  + Implementa bloques try-except en secciones críticas para manejar posibles excepciones y garantizar que el script no falle inesperadamente.
* **Seguridad:**
  + Nunca compartas tus credenciales. En un entorno de producción, utiliza métodos seguros para el manejo de secretos, como el almacenamiento de credenciales en un gestor de secretos.
* **Escalabilidad:**
  + Si en el futuro necesitas monitorear más servidores o servicios, puedes ampliar la lista servidores en el script principal.
* **Documentación:**
  + Mantén una documentación detallada de tu proyecto para facilitar el mantenimiento y posibles futuras iteraciones.

¡Con esto, tienes todas las validaciones e implementaciones necesarias para tu proyecto de monitoreo! Este sistema te permitirá agilizar las validaciones matutinas y compartir los hallazgos con el equipo de manera eficiente.

Si encuentras algún obstáculo durante la implementación o necesitas aclaraciones adicionales sobre algún punto, no dudes en hacérmelo saber. Estoy aquí para ayudarte en todo lo que necesites.

Quiero crear un sistema de alertamiento con Python, que dependiendo el servicio App(Weblogic, Tomcat o Nginx) o BD(Oracle) valide que los servicios estén arriba y los recurso de sistema operativo (memoria, CPU, Filesystem).

Quiero que el sistema se pueda hacer lo siguiente:

* 1. Genere un archivo .html que tenga la siguiente estructura:
* Servidor memoria cpu filesystem servicio App(welogic, tomacat o nginx) error\_log.