**Taller**

**autor:**

**Jorge Moreno**

**Ejercicio1**

Configura un cron que realice un backup diario

Paso 1: Crear el script de backup (guarde el script en la misma carpeta de ruta/al/script/)

#!/bin/bash

# Configuración

FECHA=$(date +%Y-%m-%d)

#BACKUP\_DIR="/home/jorge/Escritorio/backups"

ORIGEN="/home/jorge/Documentos/"

DESTINO="/home/jorge/Escritorio/backups/"

LOG\_FILE="/home/jorge/Escritorio/backups/backup.log"

# Crear directorio de backups si no existe

mkdir -p $DESTINO

# Crear backup comprimido

echo "[$(date +%Y-%m-%d\_%H:%M:%S)] Iniciando backup..." >> $LOG\_FILE

tar -czf "$DESTINO backup\_$FECHA.tar.gz" $ORIGEN 2>> $LOG\_FILE

# Verificar éxito

if [ $? -eq 0 ]; then

echo "[$(date +%Y-%m-%d\_%H:%M:%S)] Backup completado: $DESTINO backup\_$FECHA.tar.gz" >> "$LOG\_FILE"

else

echo "[$(date +%Y-%m-%d\_%H:%M:%S)] Error al crear backup!" >> $LOG\_FILE

fi

# Eliminar backups antiguos (más de 7 días)

find $DESTINO -name "backup\_\*.tar.gz" -mtime +7 -delete 2>> $LOG\_FILE

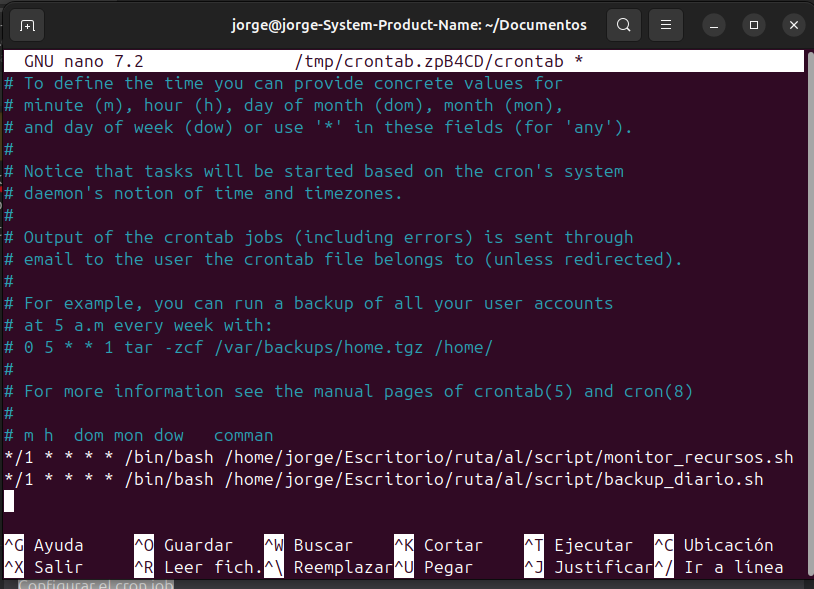
Paso 2: Hacer el script ejecutable

chmod +x /home/jorge/Escritorio/ruta/al/scripts/backup\_diario.sh

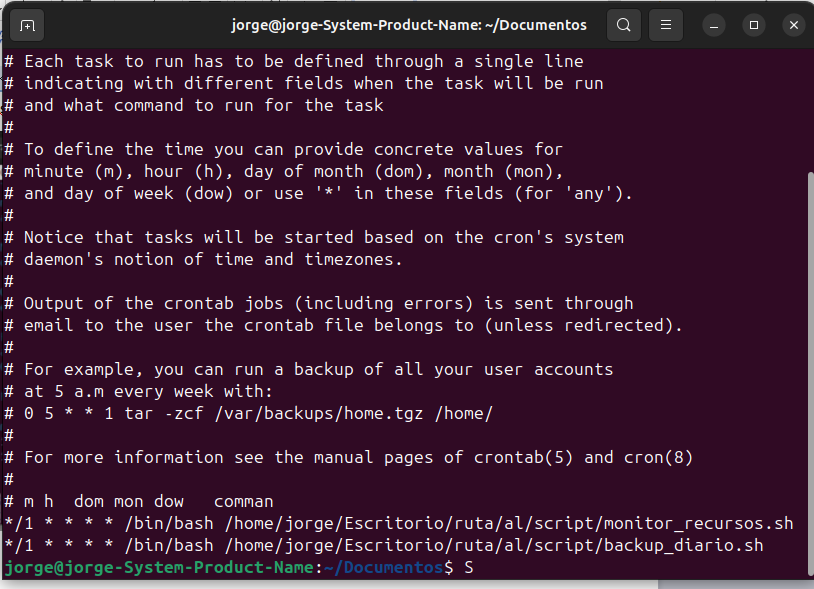
Paso 3: Configurar el cron

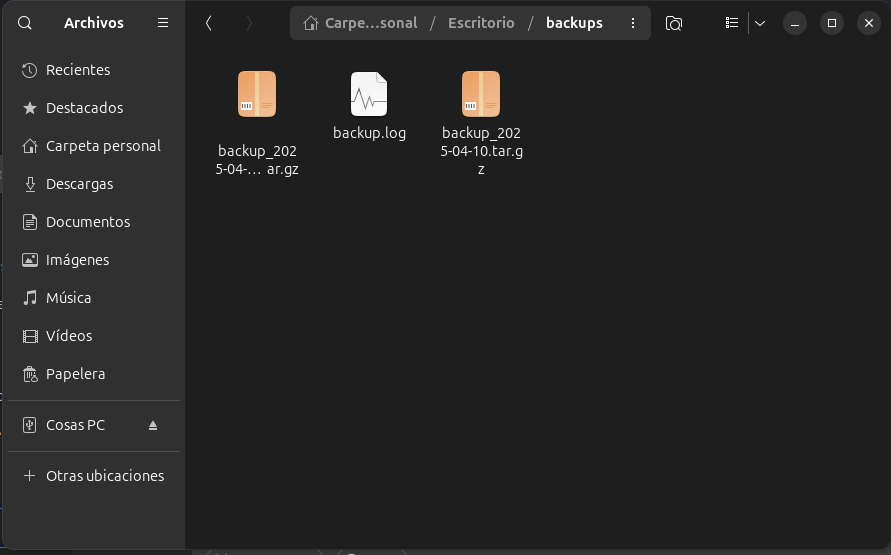
Usamos crontab -e

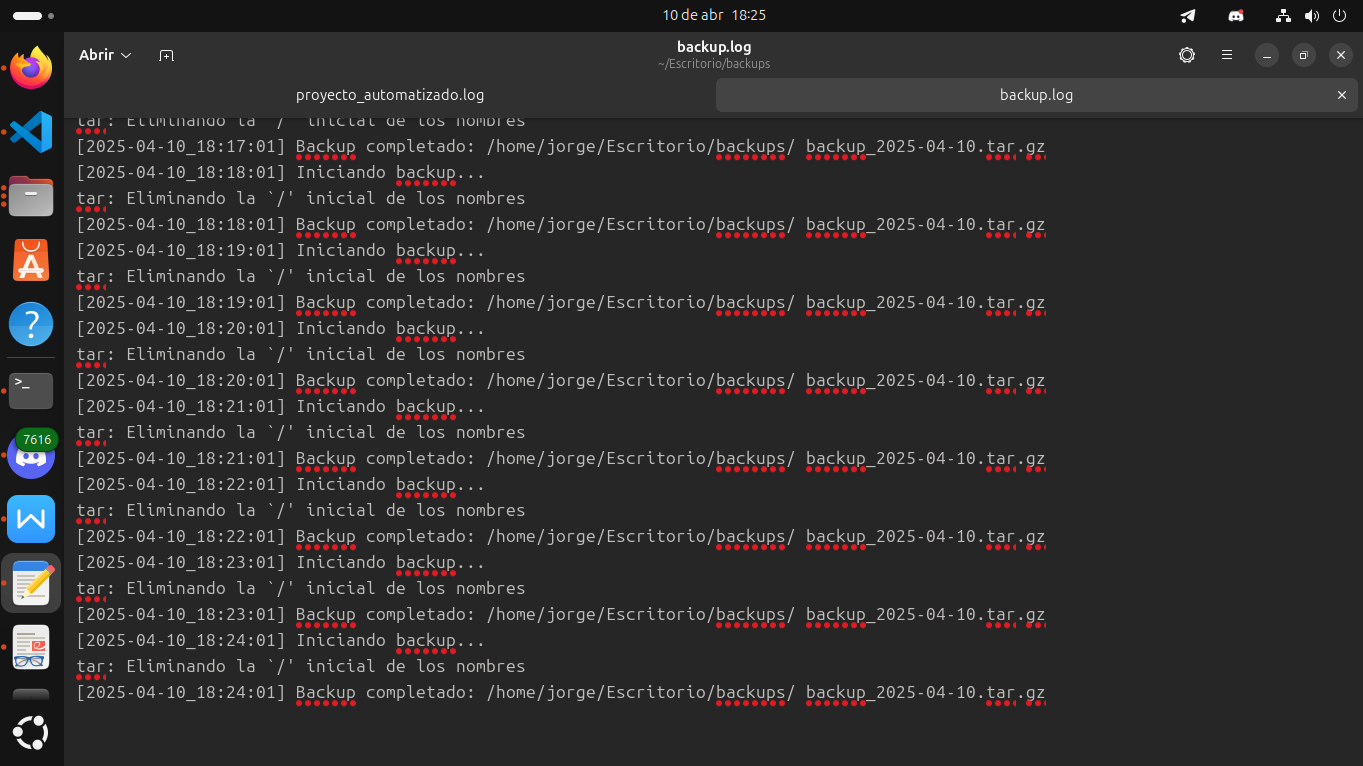
Lo coloque cada 1 minuto por metodo de practica



Usamos crontab -l



Se generan los backups con su archivo de logs 



**Ejercicio 2**

Crearemos un script que monitoree el uso de CPU y memoria del sistema, y lo programaremos para que se ejecute cada 5 minutos

mediante cron.

Paso 1: Crear el script de monitor\_recursos.sh y un script de python para crear la grafica como una imagen al lado de los logs (requisitos: tener matplotlib y pandas.

#!/bin/bash

PYTHON\_SCRIPT="/home/jorge/Escritorio/ruta/al/script/graficaMonitor.py"

LOG\_FILE="/home/jorge/Escritorio/var/log/monitor\_recursos.log"

TIMESTAMP=$(date "+%Y-%m-%d %H:%M:%S")

CPU\_USAGE=$(top -bn1 | grep "Cpu(s)" | sed "s/.\*, \*\([0-9.]\*\)%\* id.\*/\1/" | awk '{print 100 - $1}')

MEMORY\_USAGE=$(free -m | awk '/Mem:/ {printf "%.2f", $3/$2\*100}')

echo "$TIMESTAMP - CPU: $CPU\_USAGE% - Memoria: $MEMORY\_USAGE%" >> $LOG\_FILE

#crea la imagen de la grafica para que acompañe al archivo log

LINE\_COUNT=$(wc -l < "$LOG\_FILE")

python3 "$PYTHON\_SCRIPT"

Python:

import matplotlib.pyplot as plt

import pandas as pd

from datetime import datetime

LOG\_FILE = "/home/jorge/Escritorio/var/log/monitor\_recursos.log"

timestamps = []

cpu\_usage = []

mem\_usage = []

with open(LOG\_FILE, 'r') as file:

for line in file:

if "CPU" in line and "Memoria" in line:

timestamp\_str = line.split(" - ")[0]

timestamp = datetime.strptime(timestamp\_str, "%Y-%m-%d %H:%M:%S")

timestamps.append(timestamp)

# CPU

cpu\_str = line.split("CPU: ")[1].split("%")[0].strip()

cpu\_str = cpu\_str.replace(',', '.')

cpu\_usage.append(float(cpu\_str))

# MEMORIA

mem\_str = line.split("Memoria: ")[1].split("%")[0].strip()

mem\_str = mem\_str.replace(',', '.')

mem\_usage.append(float(mem\_str))

# cuadritos de datos

data = pd.DataFrame({

'Timestamp': timestamps,

'CPU (%)': cpu\_usage,

'Memoria (%)': mem\_usage

})

# Graficar

plt.figure(figsize=(12, 6))

# Gráfico de CPU

plt.subplot(2, 1, 1)

plt.plot(data['Timestamp'], data['CPU (%)'], color='red', label='CPU (%)')

plt.title('Monitor de Recursos - Uso de CPU y Memoria')

plt.ylabel('Uso de CPU (%)')

plt.grid(True)

plt.legend()

# Gráfico de Memoria

plt.subplot(2, 1, 2)

plt.plot(data['Timestamp'], data['Memoria (%)'], color='blue', label='Memoria (%)')

plt.xlabel('Tiempo')

plt.ylabel('Uso de Memoria (%)')

plt.grid(True)

plt.legend()

plt.tight\_layout()

# Guardado

plt.savefig('/home/jorge/Escritorio/var/log/monitor\_recursos.png')

print("Gráfico guardado como '/home/jorge/Escritorio/var/log/monitor\_recursos.png'")

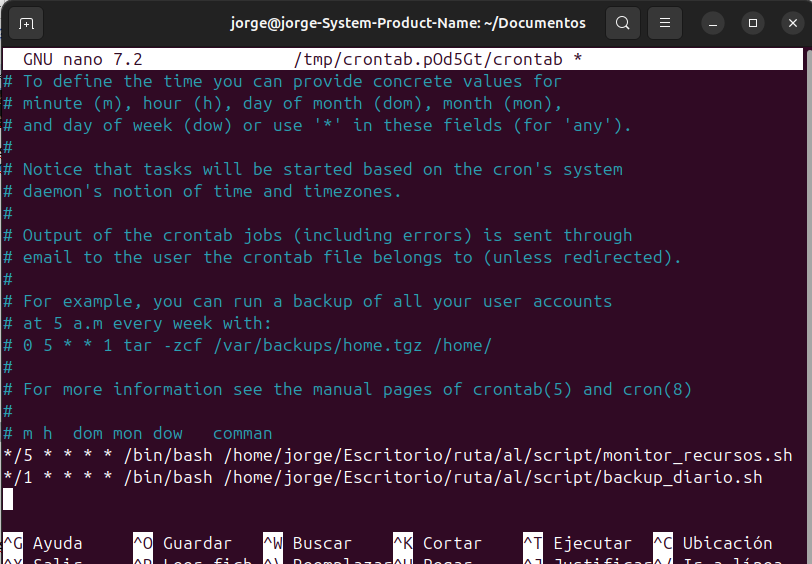
Paso 2: Hacer el script ejecutable

chmod +x /home/jorge/Escritorio/ruta/al/scripts/monitor\_recursos.sh

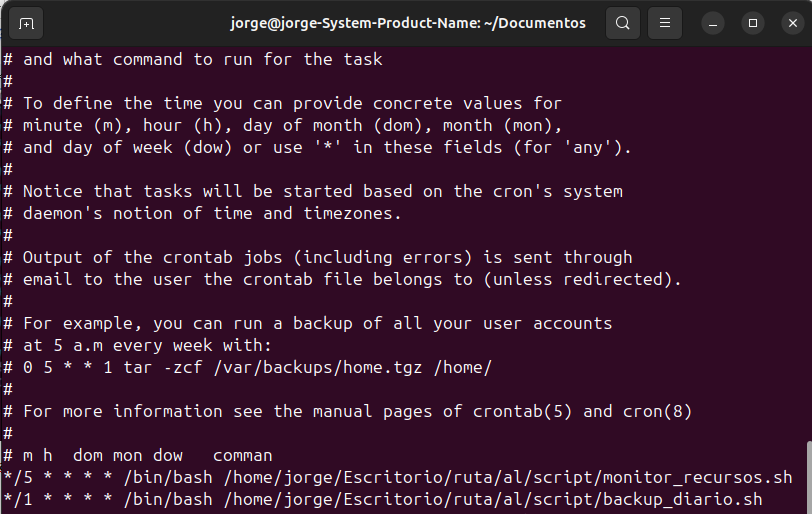
chmod +x /home/jorge/Escritorio/ruta/al/scripts/graficaMonitor.py

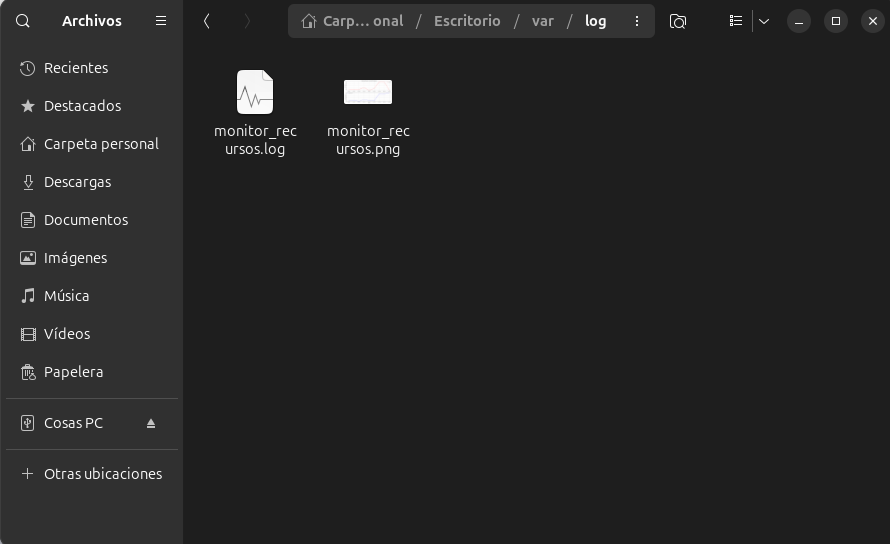
Paso 3: Configurar el cron

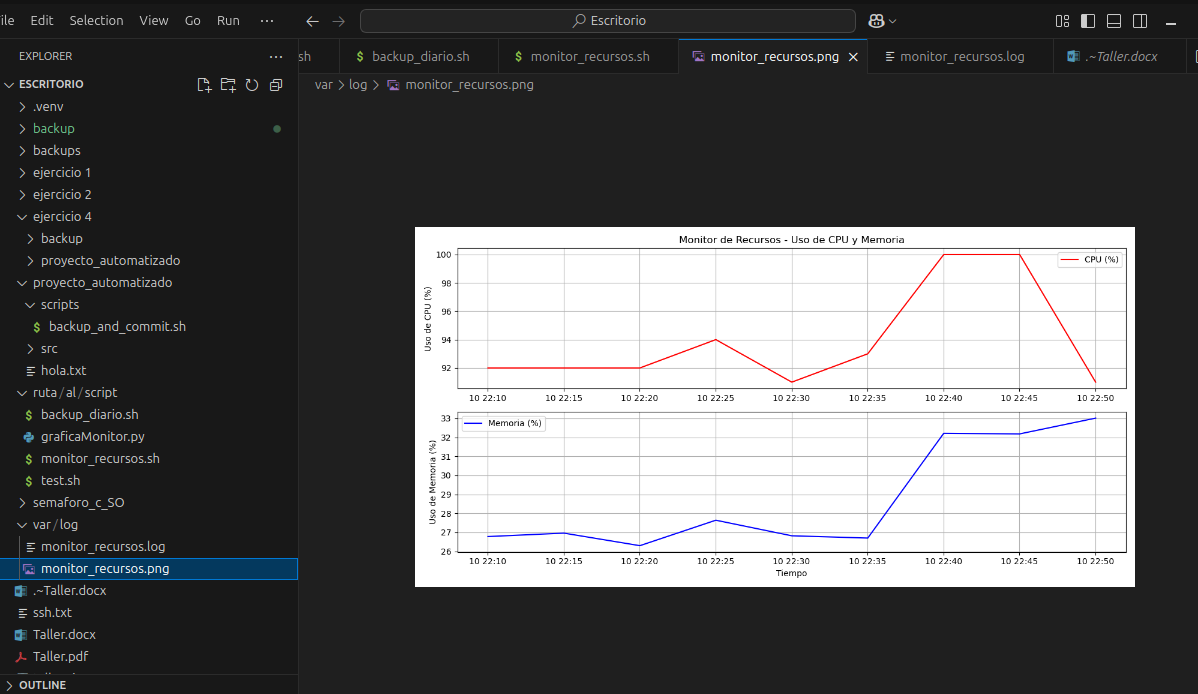
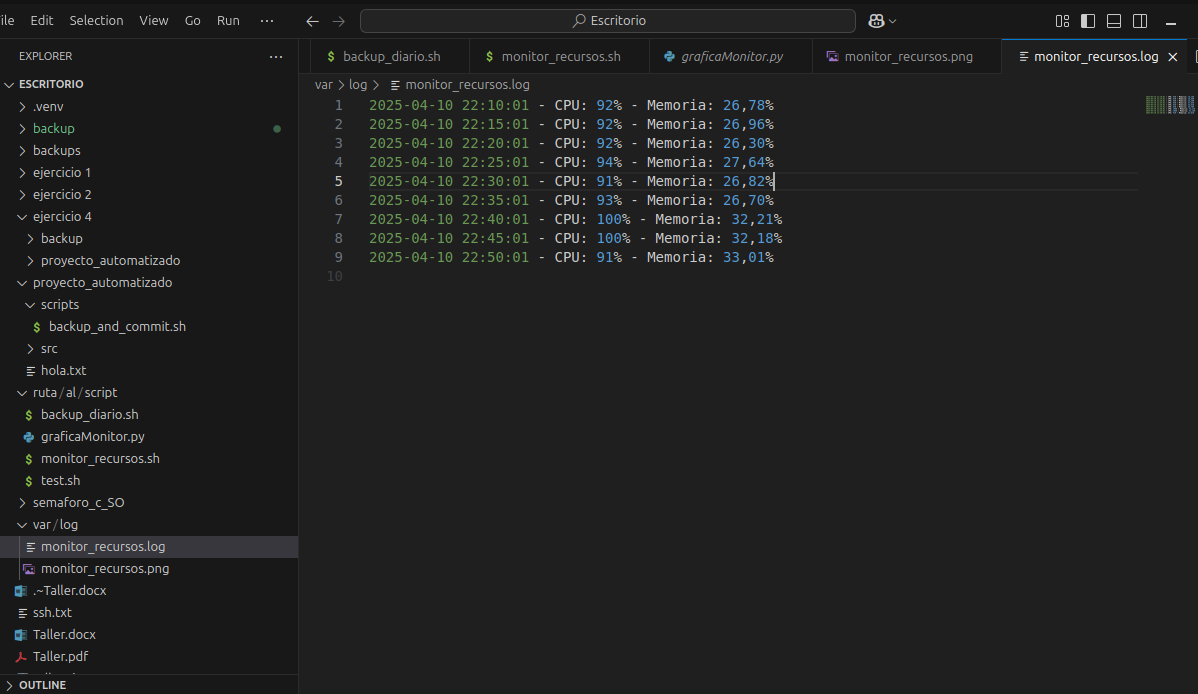
Usamos crontab -e



Usamos crontab -l



Se generan los logs 



**Ejercicio 3**

Trabaja con un compañero en un repositorio compartido,

creando ramas y resolviendo conflictos.

En este git estuvimos resolviendo el proyecto de concurrencia de sistemas operativos.

<https://github.com/xdbemoxd/semaforo_c_SO.git>



**Ejercicio 4**

<https://github.com/jorge2939/backup.git>

Paso 1: Crear el script de monitor\_recursos.sh (no utilice el del pdf porque genero muchos problemas en git y tuve que hacer uno un poco mas robusto)

#!/bin/bash

# Configuración

PROJECT\_DIR="/home/jorge/Escritorio/proyecto\_automatizado"

BACKUP\_DIR="/home/jorge/Escritorio/backup/proyecto"

LOG\_FILE="/home/jorge/Escritorio/backup/proyecto\_automatizado.log"

GIT\_REPO="git@github.com:jorge2939/backup.git"

# Configurar Git si no está configurado

if [ -z "$(git config --global user.email)" ]; then

git config --global user.email "jorge66566@gmail.com"

git config --global user.name "Jorge"

fi

# Crear directorios si no existen

mkdir -p "$BACKUP\_DIR"

touch "$LOG\_FILE"

chmod 644 "$LOG\_FILE"

# Entrar al directorio

cd "$PROJECT\_DIR" || { echo "Error: No se pudo acceder a $PROJECT\_DIR" >> "$LOG\_FILE"; exit 1; }

# Registrar inicio

echo "[$(date '+%Y-%m-%d %H:%M:%S')] Iniciando proceso..." >> "$LOG\_FILE"

# 1. Hacer backup

mkdir -p "$BACKUP\_DIR"

tar -czf "$BACKUP\_DIR/backup\_$(date +%Y%m%d%H%M%S).tar.gz" --absolute-names "$PROJECT\_DIR" >> "$LOG\_FILE" 2>&1

# 2. Actualizar repo git

# Agregar todos los cambios, incluyendo submodulos si existen

git add -A >> "$LOG\_FILE" 2>&1

# Verificar si hay cambios para commit

if [ -n "$(git status --porcelain)" ]; then

git commit -m "Auto-commit $(date '+%Y-%m-%d %H:%M:%S')" >> "$LOG\_FILE" 2>&1

# 3. Sincronizar cambios con el remoto

eval "$(ssh-agent -s)" >> "$LOG\_FILE" 2>&1

ssh-add /home/jorge/.ssh/id\_rsa >> "$LOG\_FILE" 2>&1

# Primero hacer pull con rebase para evitar conflictos

git pull --rebase origin main >> "$LOG\_FILE" 2>&1

# Luego hacer push

git push origin main >> "$LOG\_FILE" 2>&1

else

echo "No hay cambios para commit" >> "$LOG\_FILE"

fi

# 4. Limpieza (opcional)

find "$BACKUP\_DIR" -name "\*.tar.gz" -mtime +7 -delete >> "$LOG\_FILE" 2>&1

# Registrar finalización

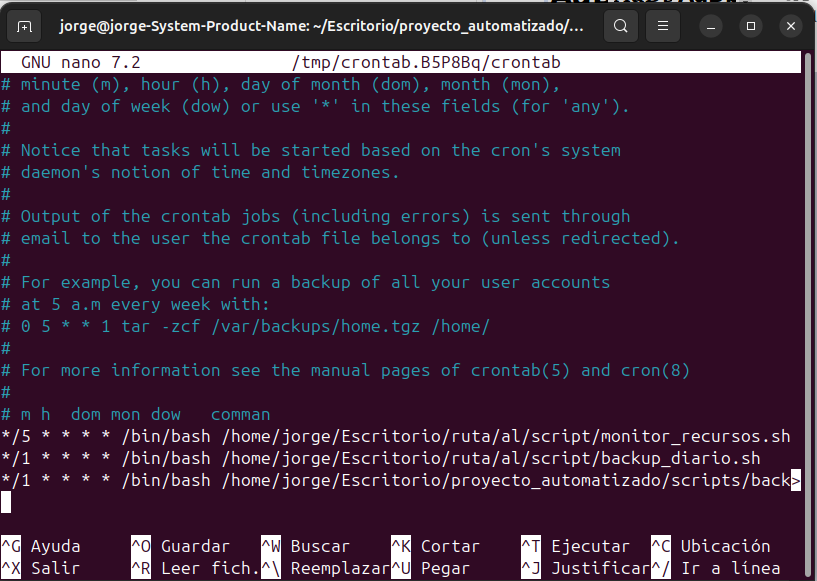
echo "[$(date '+%Y-%m-%d %H:%M:%S')] Proceso completado con estado: $?" >> "$LOG\_FILE"

Paso 2: Hacer el script ejecutable

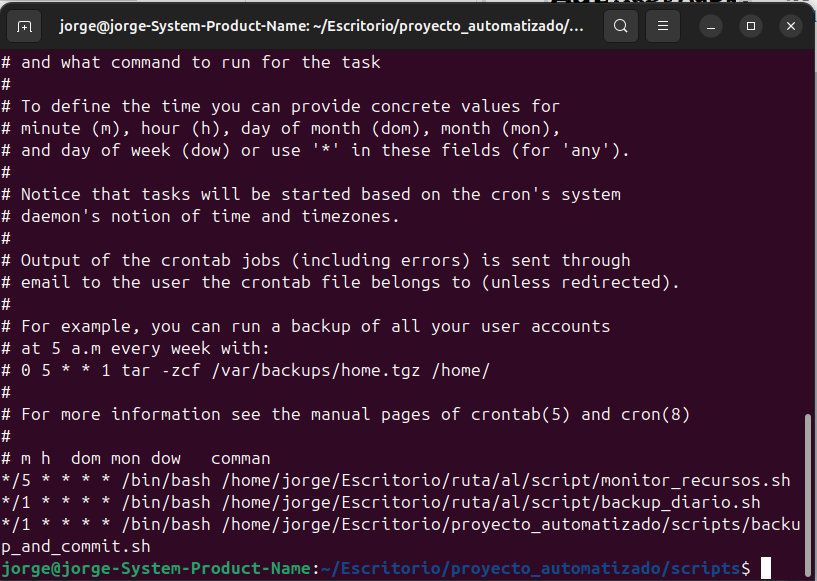
chmod +x /home/jorge/Escritorio/proyecto\_automatizado/scripts/backup\_and\_commit.sh

Paso 3: Configurar el cron

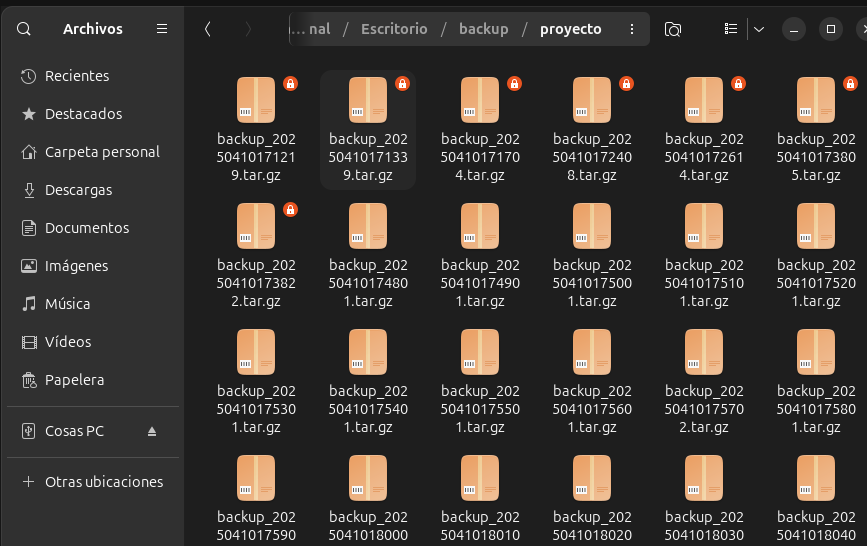
Usamos crontab -e



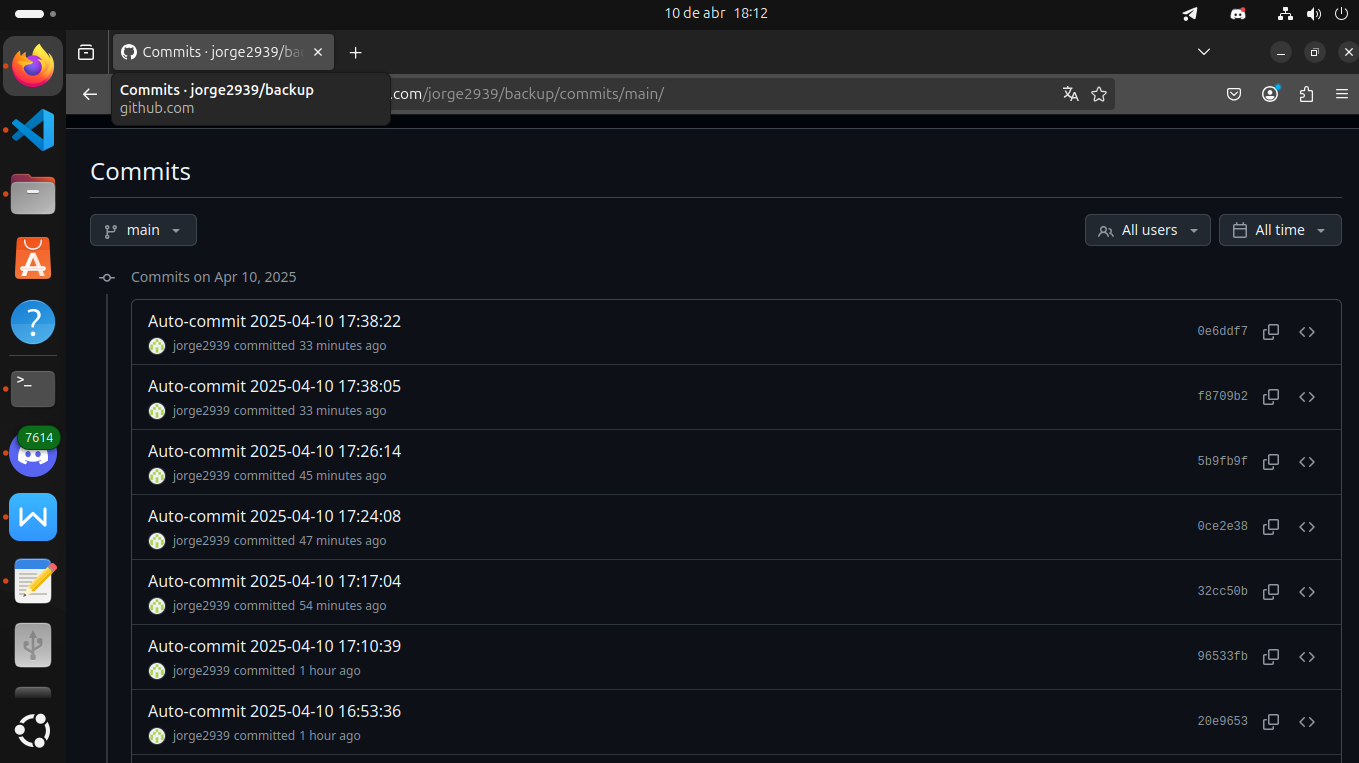
Usamos crontab -l



Se generan los backups locales



Y en github se hace commit, pull and push



Y tenemos los logs almacenados en /home/jorge/Escritorio/backup/proyecto\_automatizado.log

