

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

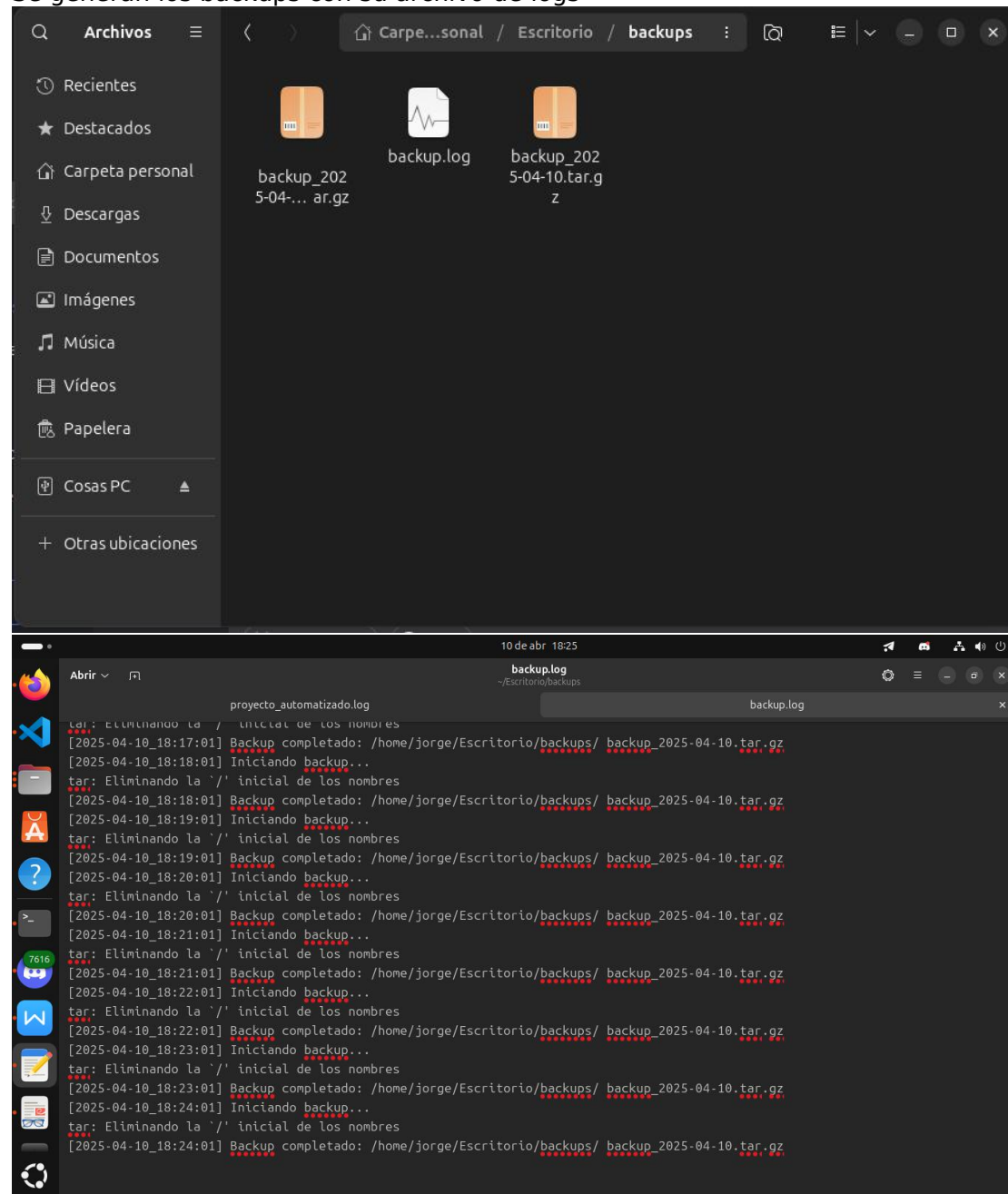
```
jorge@jorge-System-Product-Name: ~/Documentos
GNU nano 7.2 /tmp/crontab.zpB4CD/crontab *
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
*/1 * * * * /bin/bash /home/jorge/Escritorio/ruta/al/script/monitor_recursos.sh
*/1 * * * * /bin/bash /home/jorge/Escritorio/ruta/al/script/backup_diario.sh

```

Usamos crontab -l

```
jorge@jorge-System-Product-Name: ~/Documentos
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
*/1 * * * * /bin/bash /home/jorge/Escritorio/ruta/al/script/monitor_recursos.sh
*/1 * * * * /bin/bash /home/jorge/Escritorio/ruta/al/script/backup_diario.sh
jorge@jorge-System-Product-Name:~/Documentos$ S
```

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/futa/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/lorge/Escritorio/trar/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'")
```

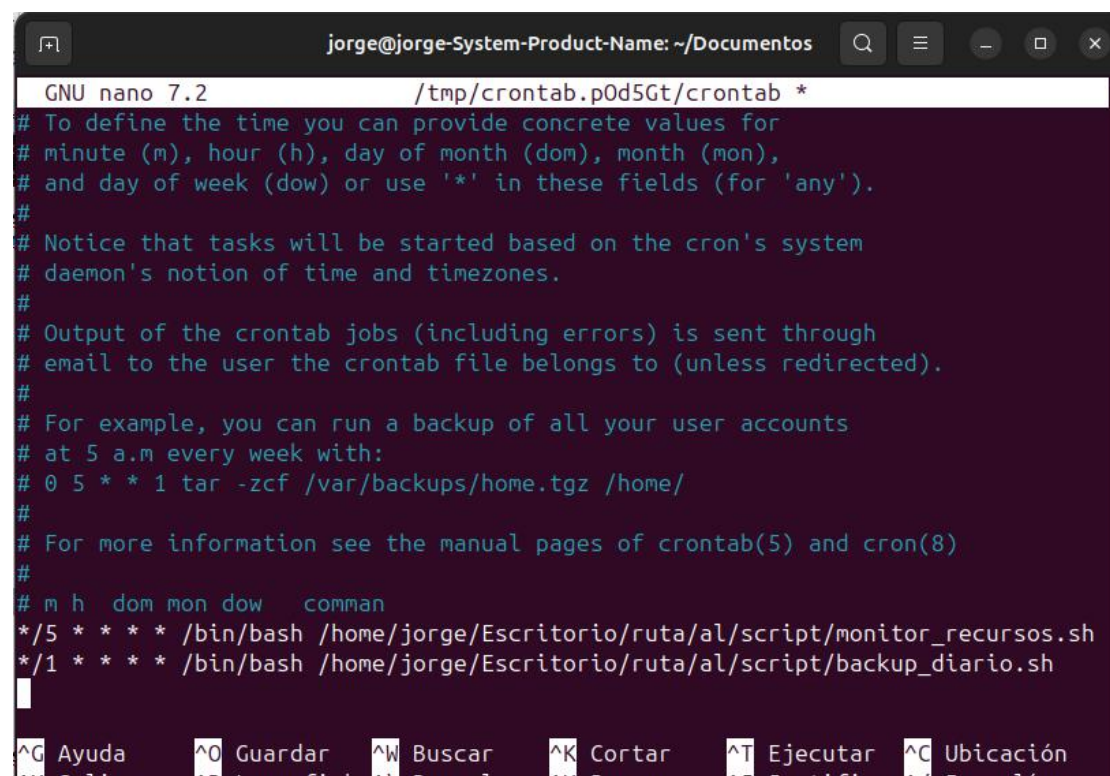
Paso 2: Hacer el script ejecutable

```
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

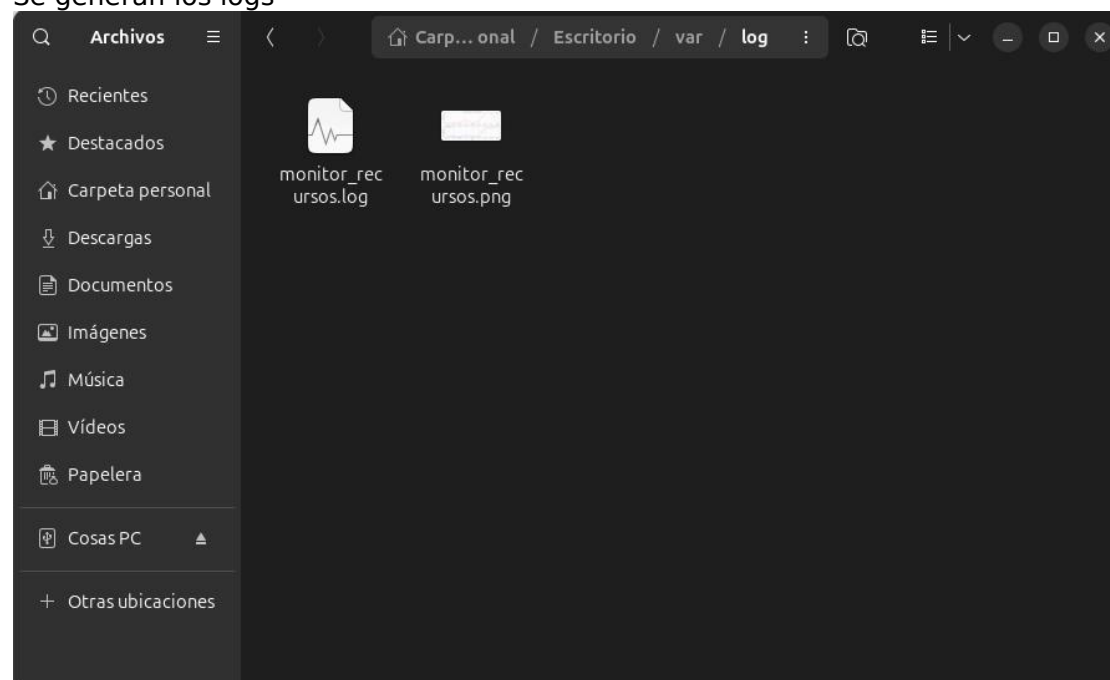


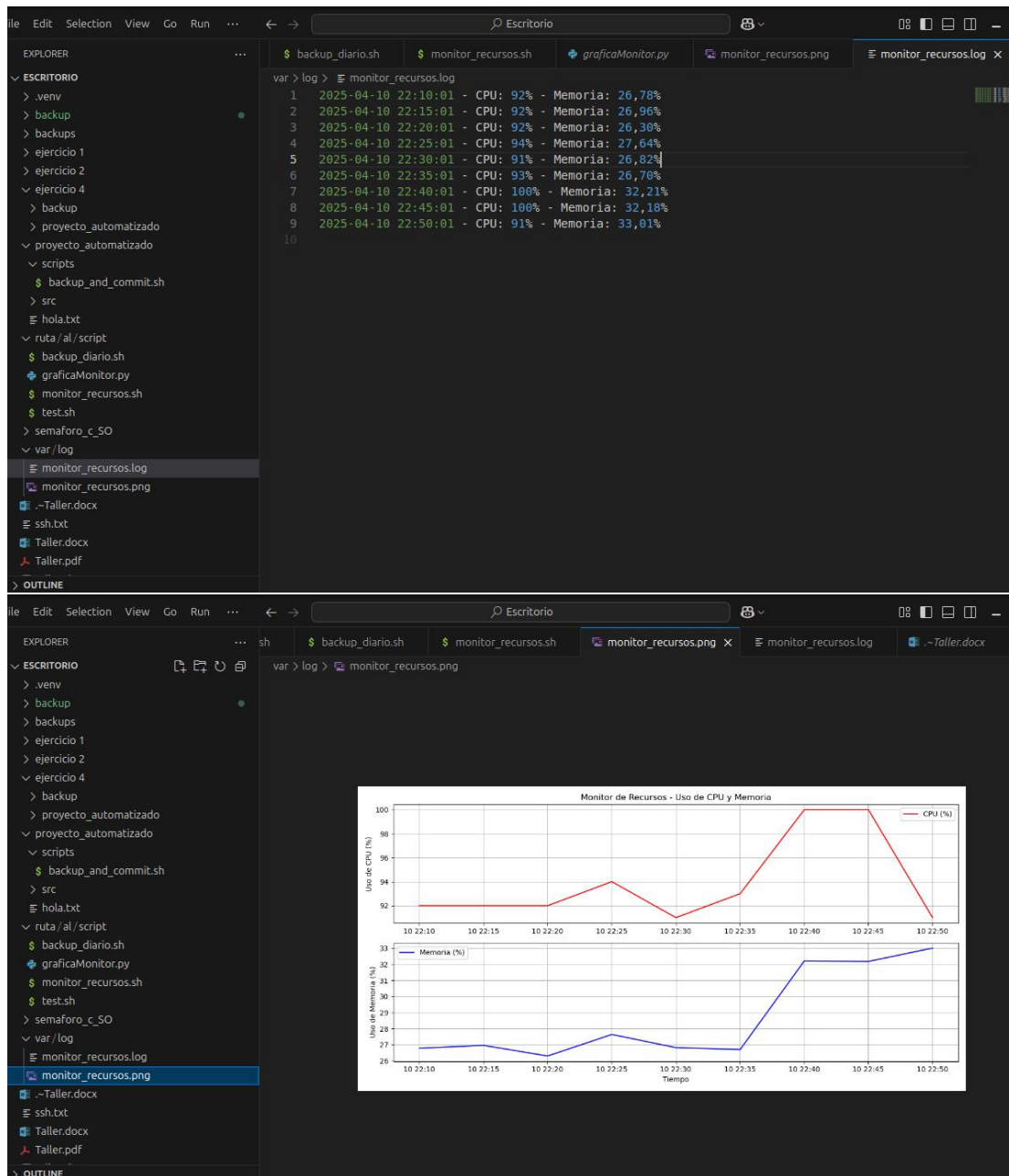
```
jorge@jorge-System-Product-Name: ~/Documentos
GNU nano 7.2 /tmp/crontab.p0d5Gt/crontab *
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
*/5 * * * * /bin/bash /home/jorge/Escritorio/ruta/al/script/monitor_recursos.sh
*/1 * * * * /bin/bash /home/jorge/Escritorio/ruta/al/script/backup_diario.sh
^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación
^V Salir ^D Borrar línea ^U Borrar línea ^_ Borrar línea ^_ Borrar línea
```


Usamos crontab -l

```
jorge@jorge-System-Product-Name: ~/Documentos
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
*/5 * * * * /bin/bash /home/jorge/Escritorio/ruta/al/script/monitor_recursos.sh
*/1 * * * * /bin/bash /home/jorge/Escritorio/ruta/al/script/backup_diario.sh
```

Se generan los logs



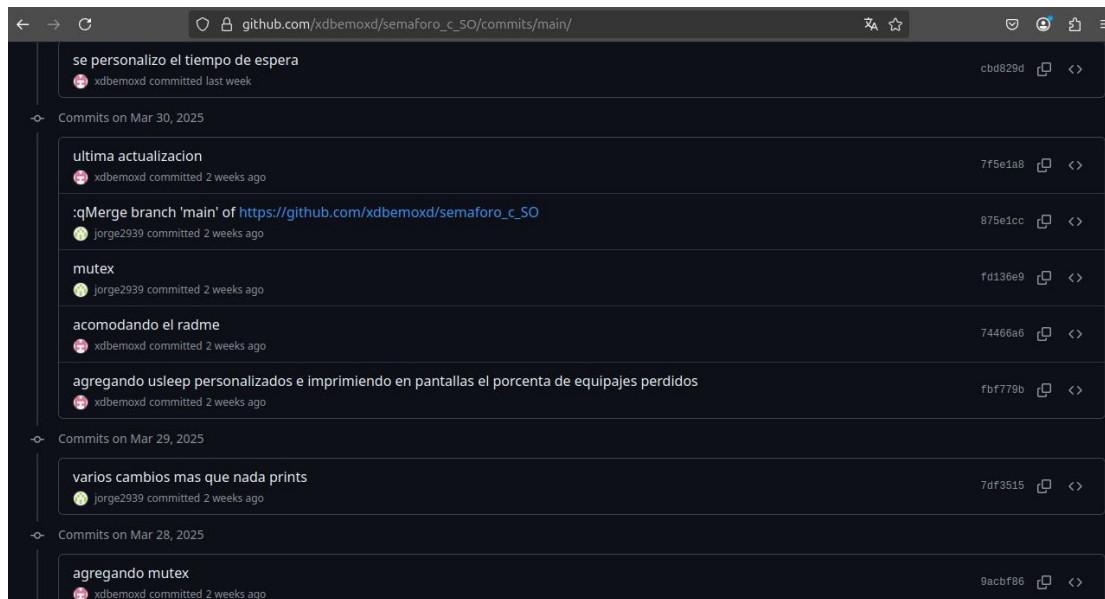


Ejercicio 3

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

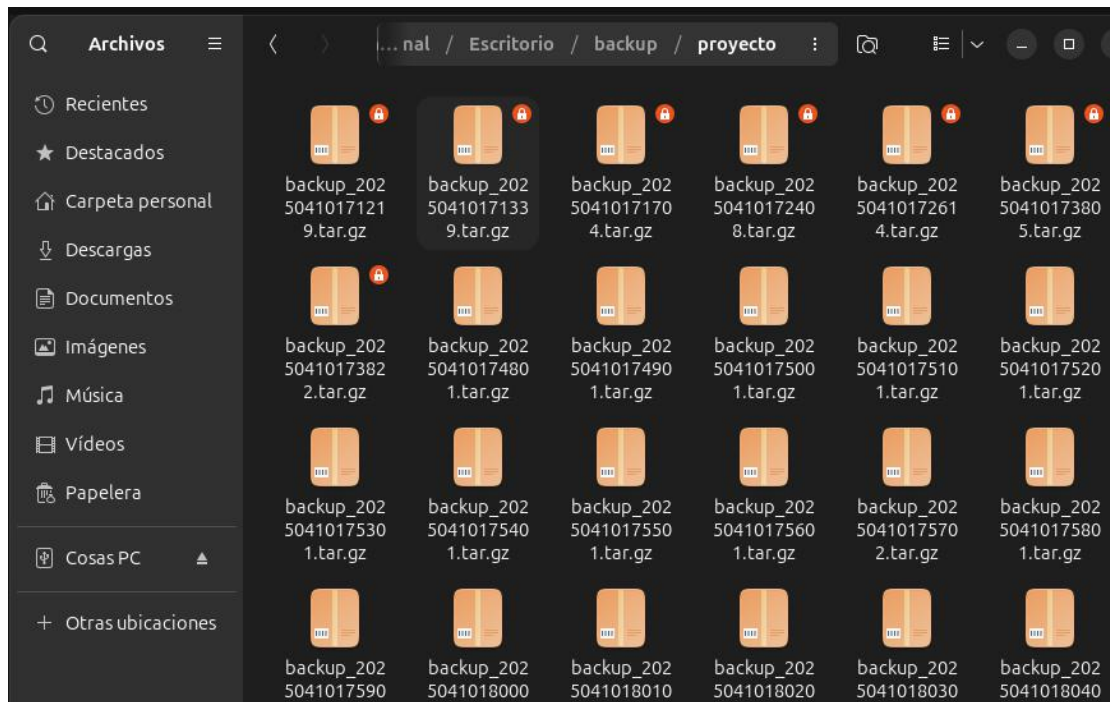
Usamos crontab -e

```
jorge@jorge-System-Product-Name: ~/Escritorio/proyecto_automatizado/...
GNU nano 7.2 /tmp/crontab.B5P8Bq/crontab
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  comman
*/5 * * * * /bin/bash /home/jorge/Escritorio/ruta/al/script/monitor_recursos.sh
*/1 * * * * /bin/bash /home/jorge/Escritorio/ruta/al/script/backup_diario.sh
*/1 * * * * /bin/bash /home/jorge/Escritorio/proyecto_automatizado/scripts/backu
^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^J Justificar ^/ Ir a línea
```

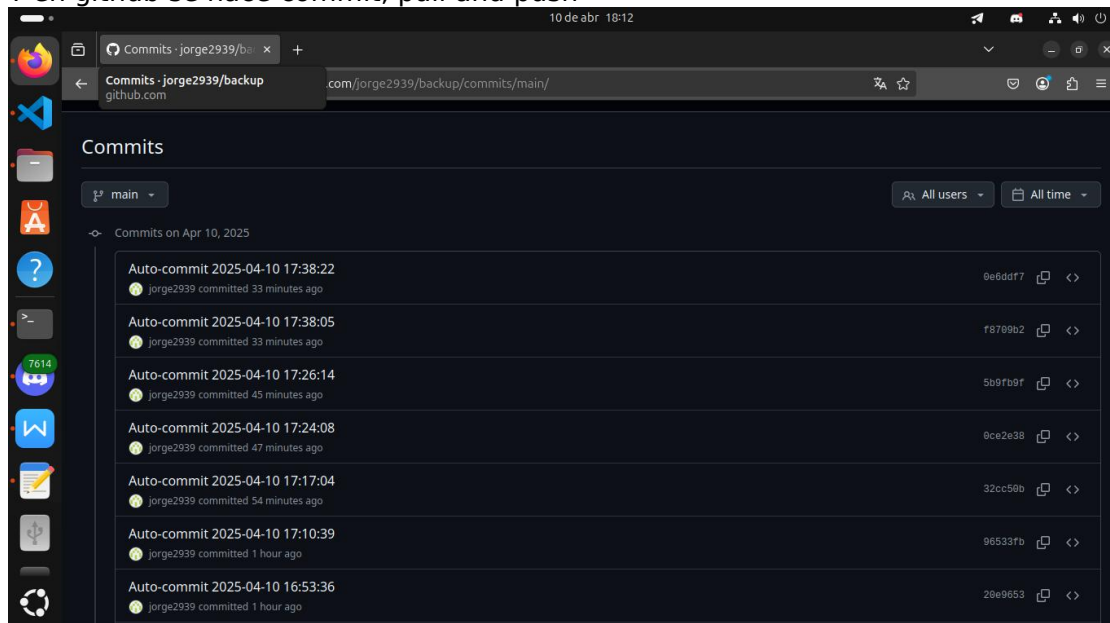
Usamos crontab -l

```
jorge@jorge-System-Product-Name: ~/Escritorio/proyecto_automatizado/...
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  comman
*/5 * * * * /bin/bash /home/jorge/Escritorio/ruta/al/script/monitor_recursos.sh
*/1 * * * * /bin/bash /home/jorge/Escritorio/ruta/al/script/backup_diario.sh
*/1 * * * * /bin/bash /home/jorge/Escritorio/proyecto_automatizado/scripts/backu
p_and_commit.sh
jorge@jorge-System-Product-Name:~/Escritorio/proyecto_automatizado/scripts$
```

Se generan los backups locales



Y en github se hace commit, pull and push



Y tenemos los logs almacenados en
/home/jorge/Esritorio/backup/proyecto_automatizado.log

```
Abrir ▾  proyecto_automatizado.log  ~/Escritorio/backup
No hay cambios para commit
[2025-04-10 18:09:01] Proceso completado con estado: 0
[2025-04-10 18:10:02] Iniciando proceso...
No hay cambios para commit
[2025-04-10 18:10:02] Proceso completado con estado: 0
[2025-04-10 18:11:01] Iniciando proceso...
No hay cambios para commit
[2025-04-10 18:11:01] Proceso completado con estado: 0
[2025-04-10 18:12:01] Iniciando proceso...
No hay cambios para commit
[2025-04-10 18:12:01] Proceso completado con estado: 0
[2025-04-10 18:13:01] Iniciando proceso...
No hay cambios para commit
[2025-04-10 18:13:01] Proceso completado con estado: 0
[2025-04-10 18:14:01] Iniciando proceso...
No hay cambios para commit
[2025-04-10 18:14:01] Proceso completado con estado: 0
```