Obligatorio

Taller de Administración de servidores Linux

Febrero 2024

**Francisco Natteri N.º 306107**

**Indice**

1. Introducción

2. Breve descripción del trabajo a realizar

3. Parte 1

4. Parte 2

5. Parte 3

6. Declaración de autoría

7. Fuentes de consulta

**Introducción**

El siguiente trabajo tiene como objetivo cumplir con las consignas solicitadas en el obligatorio del taller de Administración de Servidores Linux de Febrero 2024. Se han tratado de plasmar todas las tareas solicitadas, respetando los criterios, conocimientos y métodos adquiridos en la clase. Agradezco a las personas que desinteresadamente aportaron a la culminación, ya que el mismo exigió un alto nivel de esfuerzo y dedicación, considerando que los contenidos del sistema operativo Linux y el trabajo solicitado tienen cierto nivel de complejidad. Se puso especial énfasis en el orden, redacción y presentación de este obligatorio.

**Breve descripción del trabajo a realizar**

Parte 1

Instalación de dos servidores, un Rocky y un Ubuntu, con volúmenes lógicos de 5GB para /, 3GB para /var, 2GB para /home y 2GB para swap. Ambos con dos interfases de red (NAT y red interna), ambos se conectarán a la maquina Bastion. Adicionalmente, habrá que crear un usuario ansible, que tendrá permisos sudo sin contraseña. La clave publica de los servidores será copiada al equipo bastion para conectarse a los servidores sin contraseña.

Parte 2

Usaremos ansible para:

Actualizar los dos servidores

Instalación de OpenJDK y Tomcat 8 en uno de los servidores.

En el otro servidor, se configurará un proxy reverso para acceder a OpenJDK en el servidor donde esta se configuro.

El o los playbook con sus archivos se subirán a un repositorio git llamado “tallerfebrero2024”. Se generará un zip al descargar el repositorio, el cual será la entrega final del trabajo.

Parte 3

Documentacion con capturas de pantalla y explicaciones de los pasos para desarrollar el trabajo, asi como evidencia de la ejecución de los playbooks y sus efectos. También contendrá un enlace al repositorio donde se encuentra el playbook junto con el resto de los archivos. Finalmente, se incluirá un archivo readme.md que describirá el funcionamiento del playbook y como este debe ser utilizado.

\*Es importante mencionar que luego de la instalación de ambos servidores (además del bastion) estos serán manejados remotamente por SSH.

**Parte 1**

Vamos a comenzar instalando el servidor Rocky, al cual llamaremos serverrocky, y le pondremos un disco de 12GB como se requiere:

A screenshot of a computer

Description automatically generated

Luego de poner los detalles básicos, nos falta tener las dos tarjetas de red como se pide en la letra. Primero, la que tendra NAT:

A screenshot of a computer

Description automatically generated

Y luego la que tendra Host-Only Network:

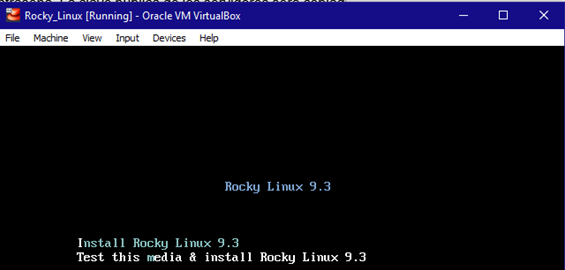
A computer screen shot of a computer

Description automatically generated

A computer screen shot of a computer

Description automatically generated

Completado este paso, ahora si arrancaremos la VM y comenzaremos la instalación:



En idioma escogeremos inglés:

A screenshot of a computer

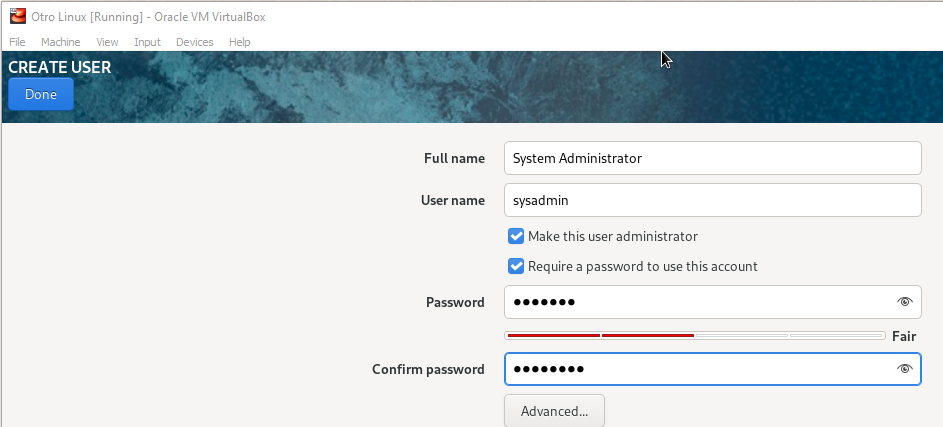
Description automatically generated

Configuramos el acceso con root. También se configuro, luego, el usuario sysadmin con privilegios de administrador:

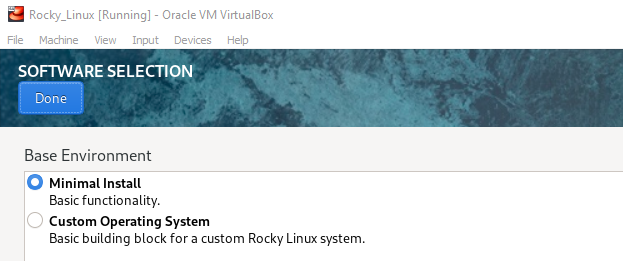
A screenshot of a computer

Description automatically generated

Creamos y configuramos el usuario sysadmin:



El server será instalado en minimal:



Configuración de la red con las dos tarjetas (NAT y host-only):

A screenshot of a computer

Description automatically generated

Por último, vamos a configurar las particiones del disco, de acuerdo con los requerimientos:

A screenshot of a computer

Description automatically generated

Cumplidos todos estos pasos, podemos dar inicio a la instalación del Rocky Linux

A screenshot of a computer

Description automatically generated

Una vez terminada la instalación del Rocky Linux, procedemos de la misma manera e instalaremos el servidor Ubuntu. Este servidor se llamará serverubuntu:

A screenshot of a computer

Description automatically generated

Creamos la configuración de red al igual que Rocky (una NAT, otra host-only adapter):

A computer screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

Iniciamos la máquina virtual y arrancamos la instalación:

A screenshot of a computer

Description automatically generated

Luego de escoger idioma, seleccionamos Ubuntu Server minimizado:

A screenshot of a computer

Description automatically generated

Configuración de las tarjetas de red:

A screenshot of a computer

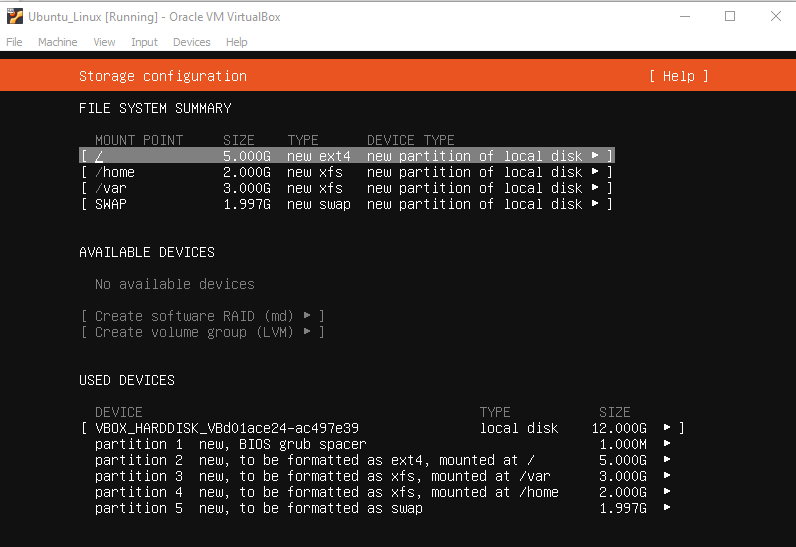
Description automatically generated

Luego de pasar las ventanas de proxy (no es necesario) y el mirror, vamos a la configuración del disco:

A screenshot of a computer

Description automatically generated

Y se procede a configurarlo con los tamaños para cada partición al igual que en Rocky:



Aquí vamos a crear el user sysadmin ya que no podemos crear en la instalación el usuario ansible sin password. Este será creado luego:

A screenshot of a computer

Description automatically generated

El servidor Open SSH es opcional, pero en nuestro caso lo instalaremos para tener un acceso más sencillo al Ubuntu:

A screenshot of a computer

Description automatically generated

Los servers snaps los ignoramos por ahora, y procedemos a instalar:

A screenshot of a computer

Description automatically generated

Luego de terminar de instalar los dos servers, vamos también a crear una VM con Rocky Linux, la cual será nuestro bastion de donde ejecutaremos los playbooks solicitados en la parte 2. Es básicamente el mismo proceso que el Rocky ya creado asi que obviaremos en el documento la parte de instalación.

Finalmente, y para terminar la primera parte, crearemos el usuario ansible en los dos servers. Primero en el servidor Rocky\_Linux:

Creamos el usuario ansible:

**sudo adduser ansible**

Agregamos el usuario al grupo “wheel” para darle privilegios sudo:

**sudo usermod -aG wheel ansible**

Editamos el archivo de sudoers:

**sudo visudo**

Nos vamos hasta el final del archivo y agregamos la siguiente línea que permitirá realizar acciones sudo sin password:

**ansible ALL=(ALL) NOPASSWD: ALL**

Luego, en Ubuntu el proceso es similar, pero no exactamente igual:

Creamos el usuario ansible:

**sudo adduser ansible**

A diferencia de Rocky, en Ubuntu el grupo de sudoers se llama “sudo”. Para agregar ansible a ese grupo, ingresamos el siguiente comando:

**sudo usermod -aG sudo ansible**

Finalmente, y también distinto al proceso de Rocky, en Ubuntu lo que hacemos para agregar esa línea que hará que ansible haga acciones de sudo sin password es, primero, acceder al archivo de sudoers /etc/sudoers

**sudo vim /etc/sudoers**

Luego, el proceso es el mismo que en Rocky: agregar la siguiente línea al final del archivo:

**ansible ALL=(ALL) NOPASSWD: ALL**

Es importante mencionar como quedan los IP de cada maquina:

**Rocky: 192.168.56.101**

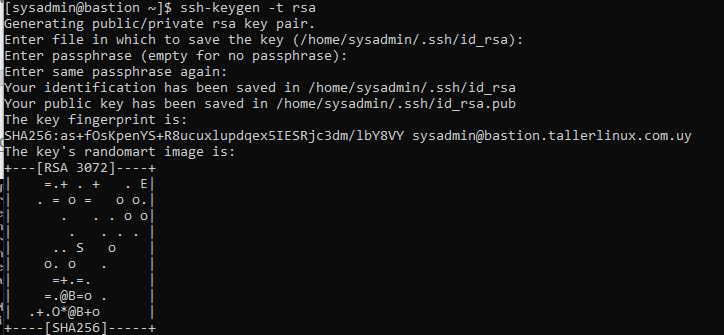
**Ubuntu: 192.168.56.102**

**Bastión: 192.168.56.103**

Finalmente, vamos a generar los .ssh keys del Bastion, para copiarlos a cada servidor que queremos manejar. Para generar los ssh keys del Bastión, usaremos el comando:

**ssh-keygen -t rsa**

Nos pedirá donde salvar las claves, le damos enter para lo que haga en la ubicación por defecto. Haremos lo mismo con el passphrare (doble enter). Luego, vamos al directorio /home/ansible/.ssh y comprobamos que se hayan generado las claves. Aquí, por ejemplo, lo vemos en el servidor Ubuntu:



Finalmente, solo nos queda copiar las claves públicas del Bastión a los dos servidores Rocky y Ubuntu con el comando:

ssh-copy-id ansible@192.168.56.101 (Rocky Server)

ssh-copy-id ansible@192.168.56.102 (Ubuntu Server)

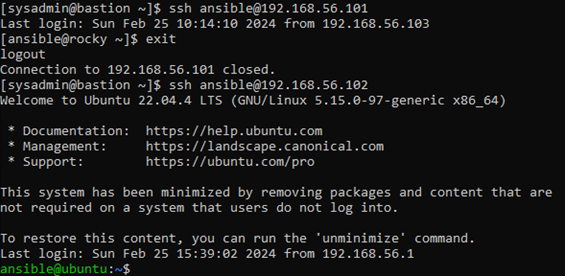
A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

Y comprobamos que si nos podemos unir desde el Bastion via SSH sin password:



**Parte 2**

Ahora vamos a armar los playbooks con lo que se nos solicita en la segunda parte del trabajo. Iremos armando de a pocos la automatización de manera que sea entendible. Antes de eso, procederemos a crear el ambiente para nuestro trabajo:

En el Bastión, procederemos a crear el directorio tallerfebrero2024 dentro de /home/sysadmin

A screen shot of a computer

Description automatically generated

Dentro del directorio tallerfebrero2024, crearemos el directorio inventories

A black background with white text

Description automatically generated

Crearemos el archivo hosts dentro del directorio inventories para especificar las direcciones IP de los servers, y para esto usaremos el comando:

**sudo vim inventories/hosts**

\*Nota: hay que tener instalado vim en el Bastion para esta función.

A computer screen with white text

Description automatically generated

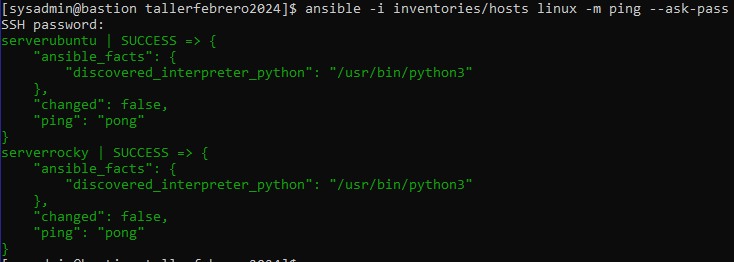
También hay que instalar el paquete ansible-core en el bastion con el comando

**sudo dnf install ansible-core**

Teniendo ya el archivo hosts, el directorio de trabajo y el paquete ansible-core, ya podemos comenzar a crear nuestros playbooks.

De todas maneras, antes de crear el .yml, se hizo una prueba con el comando

**ansible -i inventories/hosts linux -m ping --ask-pass**

****

Esto es con el fin de comprobar la conectividad con los nodos de destino.

**Actualización de servidores**

Nuestro primer plabook, que es el que actualiza los servidores, le llamaremos obligatorio.yml, el cual estará alojado en el directorio /home/sysadmin/tallerfebrero2024. Creamos el archivo con el comando:

**sudo vim obligatorio.yml**

Comenzamos haciendo en el playbook la tarea de actualización de los servidores, con el handler para que una vez actualizado (si asi fuera) se reinicien los servidores):

A screen shot of a computer program

Description automatically generated

El respectivo check del playbook, con lo cual se demuestra que funciona:

A screen shot of a computer

Description automatically generated

**Instalación de OpenJDK 17 en el servidor Rocky**

Continuaremos editando nuestro obligatorio.yml:

A black background with white text

Description automatically generated

Como prueba, corrimos el playbook y hasta este momento el deploy es exitoso:

A screen shot of a computer

Description automatically generated

En Rocky, se muestra OpenJDK ya instalado:

A number and numbers on a black background

Description automatically generated

**Instalación de Tomcat 8 en el servidor Rocky**

En este caso, fue necesaria mucha investigación y se me complico mucho construir un playbook que lograra cumplir con esta premisa. Se instalo la versión 8.5.72 y en el mismo playbook, que es uno creado especialmente para esta tarea llamado tomcat2.yml (porque el primero no funciono), se hicieron varias tareas:

**Instalación de Tomcat 8.5.72**

**Instalación de comandos para descomprimir el archivo tar.gz que contenía Tomcat**

**Creacion de usuario y grupo tomcat**

**Creacion del directorio /opt/tomcat para la aplicación**

**Descarga y extracción/descompresión de Tomcat**

**Movimiento de archivos de /opt/tomcat/apache-tomcat-\*/\* /opt/tomcat"**

**Creacion del systemd unit file para Tomcat**

**Recarga de systems, y además inicio y registro como servicio a Tomcat**

**Abrir puerto de acceso a la interfase web de Tomcat**

Es importante destacar que para la configuración de firewalld, que hace la tarea de la apertura de puertos para acceder a Tomcat, necesitamos el módulo de ansible “ansible.posix” el cual instalaremos con el comando:

**ansible-galaxy collection install ansible.posix (firewalld)**

**A screen shot of a computer

Description automatically generated**

No presento captura del playbook ya que tiene muchas líneas. Se podrá ver sin problemas en el repositorio en Github.

Aquí podemos ver el playbook como se ejecutó perfectamente:

A screenshot of a computer program

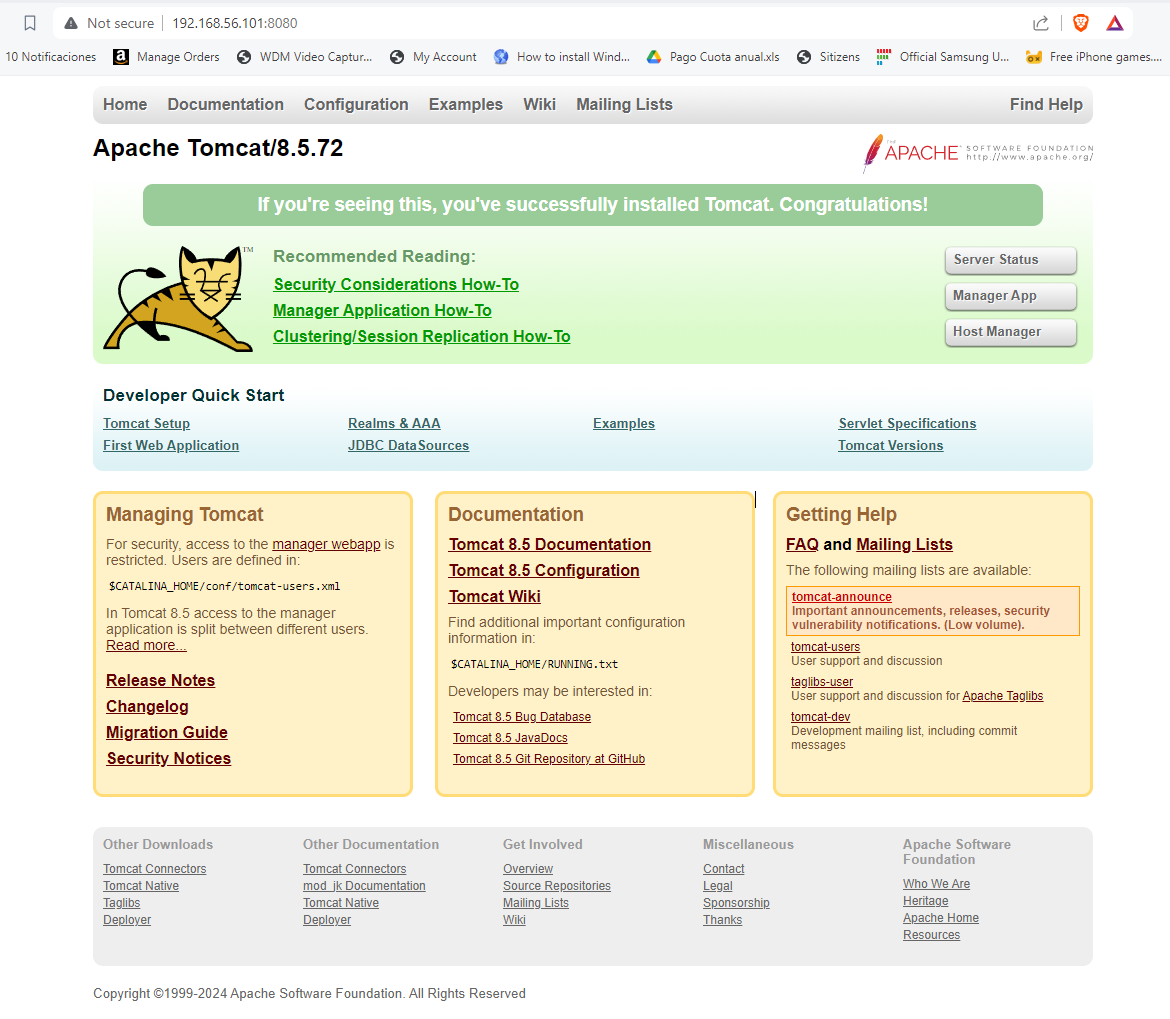
Description automatically generated

Y aquí tenemos el servicio Tomcat esta corriendo perfectamente en el servidor Rocky Linux:

A screen shot of a computer program

Description automatically generated

Finalmente, una prueba para ver si se podía acceder desde la computadora host (Windows 10) a la interfase web, la cual fue exitosa:



**Configuración/instalación del reverse proxy en el otro servidor (Ubuntu)**

Por último, vamos a crear un nuevo playbook para el Ubuntu, que hará de reverse proxy. Este se llamará reverse\_proxy.yml y hará varias tareas:

**Instalar apache server**

**Iniciar apache y configurarlo como servicio**

**Copiar la configuración para el reverse proxy, que ya tenemos en un template llamado httpd.conf.j2**

**Habilitar los módulos de Apache requeridos para la familia Debian**

**Instalar el firewall ufw**

**Configurar ufw como servicio**

**Abrir el puerto 80 en el firewall para el apache server**

**Permitir el acceso SSH**

El playbook es extenso asi que lo dejamos parea que se vea en el repositorio.

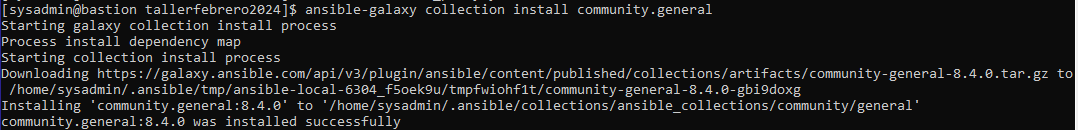
Este es el template:

A screen shot of a computer

Description automatically generated

Es importante señalar que para la instalación del firewall ufw es necesario tener el módulo community.general, que se instala con el siguiente comando:

**ansible-galaxy collection install community.general**



Aquí podemos ver la ejecución del playbook sin errores:

A screenshot of a computer program

Description automatically generated

Y la prueba de que el Tomcat 8.5.72 se accede por el IP del servidor Ubuntu (reverse proxy):

A screenshot of a computer

Description automatically generated

**Parte 3**

Para reflejar nuestros archivos en un repo, es necesario instalar git en nuestro Bastion:

**sudo dnf install git**

En mi caso, comencé a trabajar en el proyecto antes de conectar el directorio local con el repo, asi que utilicé los siguientes comandos:

Primero, para inicializar un nuevo repositorio git en mi folder de trabajo:

**sudo git init**

Agregue el repositorio remoto de Github:

**sudo git remote add origin** [**https://github.com/putuco/tallerfebrero2024**](https://github.com/putuco/tallerfebrero2024)

Luego, renombre (por instrucciones que tenía en Github) el branch a “main”

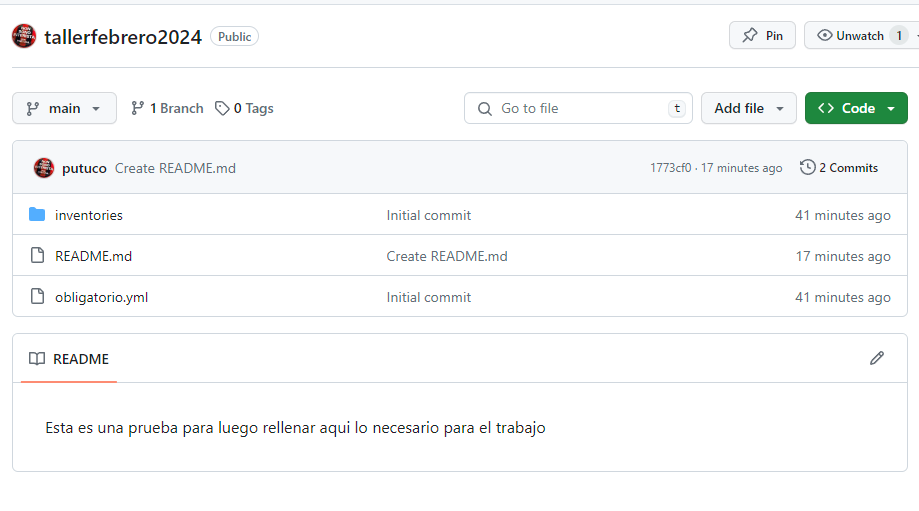
**sudo git branch -M main**

Para finalmente empujar todo con el comando

**sudo git push -u origin main**

En este punto tuve una dificultad, ya que el acceso con usuario y clave ya no está permitido en Github. Tuve que entrar a mi cuenta, a developer options y crear un “fine-grain token”. Una vez configurado este token, copie el código y cuando me pidió el usuario y luego el password ingrese el código y el repositorio quedo sincronizado/clonado:

Captura para referencia, ya que en este momento aún no está terminado el trabajo:

****

El repo ha quedado clonado en [**https://github.com/putuco/tallerfebrero2024**](https://github.com/putuco/tallerfebrero2024)

Es importante mencionar que luego de esto cree el readme.md en el repo remoto y lo sincronice con el local con el comando:

**sudo git fetch origin**

Cada vez que tengo un cambio local y lo quiero reflejar en el repositorio remoto, sigo estos pasos:

Agrego cambios al staging area:

**sudo git add** .

Hago el commit de los cambios con un mensaje:

**sudo git commit -m "Mensaje” (el mensaje que se desee)**

Y finalmente empujo los cambios al repositorio remoto:

**sudo git push origin main**

**Declaración de autoría**

El documento presentado está diseñado por el estudiante de la carrera de Analista de Infraestructura Informática Francisco Natteri.

Este documento es un trabajo práctico del taller de Administración de Servidores Linux dictado por el docente Enrique Verdes.

Cabe destacar que el material está formado en su totalidad por dicha persona, sin copia de ninguna índole de otras fuentes salvo las que son mencionadas de manera explícita en los enlaces al final del documento. También se mencionan en ese apartado las herramientas utilizadas.

Si se hace un uso indebido de este documento, el autor tendrá derecho a un reclamo con posibles sanciones al individuo mal intencionado.

**Fuentes de consulta**

Aulas ORT

Chat GPT

Copilot

Google

El Mati (al que ChatGPT le pregunta)