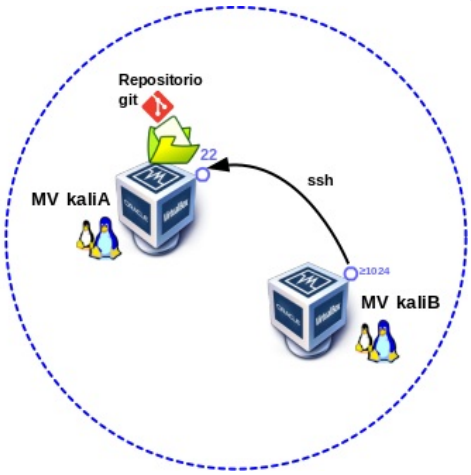
git

ESCENARIO

Máquinas virtuais:
RAM ≥ 2048MB CPU ≥ 2 PAE/NX habilitado
Rede: 192.168.120.0/24
ISO: Kali Live amd64
BIOS: Permite arranque dispositivo extraíble: CD/DVD, USB
Rede Interna (eth0)

Máquina virtual A:
Servidor SSH: openssh-server
git init (repositorio)
IP/MS: 192.168.120.100/24

Máquina virtual B:
Cliente SSH: openssh-client (ssh)
git clone
IP/MS: 192.168.120.101/24

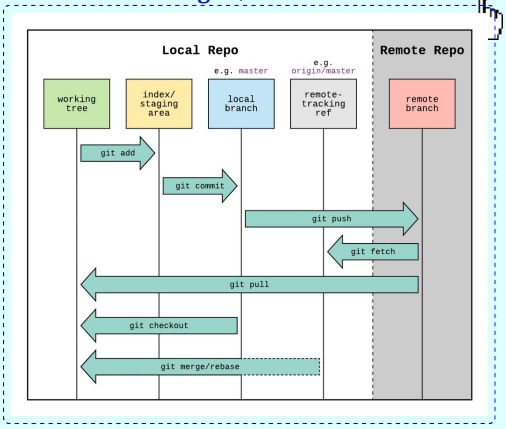


Prerrequisito:

- **Uso de git en local**

NOTAS: Documentación de interese:

- **git cheat sheet education**
- **Git and Git Flow Cheat Sheet**
- **Pro Git book**
- **Fundamentos de git (ver Áreas de traballo)**



- **Learn Git Branching**

Práctica git

Máquina virtual A: Kali amd64

1. Cambiar hostname da máquina virtual A. Por kaliA como hostname:

   Na contorna gráfica abrir un terminal e executar:

   kali@kali:~$ setxkbmap es #Cambiar o mapa de teclado ao idioma español.
   kali@kali:~$ sudo su - #Acceder á consola de root(administrador) a través dos permisos configurados co comando sudo (/etc/sudoers, visudo)

   Opción A:

   root@kali:~# echo 'kaliA' > /etc/hostname #Indicar ao sistema o valor do hostname.
   root@kali:~# echo 'kernel.hostname=kaliA' >> /etc/sysctl.conf #Indicar ao kernel o valor do hostname.
   root@kali:~# sysctl -p #Activar o cambio de hostname sen ter que pechar sesión nin reiniciar
   root@kali:~# exit #Saír da consola local sudo na que estabamos a traballar para voltar á consola local de **kali**.
   kali@kali:~$ exit #Pechar o terminal saíndo da consola local do usuario **kali**.

   Opción B:

   root@kali:~# hostnamectl hostname kaliA #Modificar o valor do hostname a kaliA.
   root@kali:~# exit #Saír da consola local sudo na que estabamos a traballar para voltar á consola local de **kali**.
   kali@kali:~$ exit #Pechar o terminal saíndo da consola local do usuario **kali**.

2. Configurar a rede:

   Na contorna gráfica abrir un terminal e executar:

   kali@kaliA:~$ sudo su - #Acceder á consola de root(administrador) a través dos permisos configurados co comando sudo (/etc/sudoers, visudo)
   root@kaliA:~# /etc/init.d/avahi-daemon stop || systemctl stop avahi-daemon #Parar o demo avahi-daemon(control resolución de nomes) para poder configurar de forma manual a configuración de rede e non ter conflicto con este demo.
   root@kaliA:~# /etc/init.d/network-manager stop || pkill NetworkManager #Parar o demo network-manager(xestor de rede) ou o script NetworkManager (executado sen ser demo) para poder configurar doutro xeito (co comando ip(ifconfig) de forma manual ou mediante networking (ficheiros /etc/init.d/networking, /etc/init.d/networking.d) a configuración de rede e non ter conflicto con este xestor.
   root@kaliA:~# ip addr show #Amosar a configuración de todas as tarxetas de rede. Nesta caso, na máquina A, as tarxetas de redes: loopback(lo) e interna(eth0).
   root@kaliA:~# ip addr add 192.168.120.100/24 dev eth0 #Configurar a tarxeta de rede interna eth0, coa IP: 192.168.120.100 e máscara de subrede: 255.255.255.0.
   root@kaliA:~# ip addr show #Amosar a configuración de todas as tarxetas de rede. Nesta caso, na máquina A, as tarxetas de redes: loopback(lo) e interna(eth0).
   root@kaliA:~# ping -c4 192.168.120.100 #Comprobar mediante o comando ping a conectividade coa interface de rede local eth0

3. Comprobar estado do Servidor SSH:
   root@kaliA:~# /etc/init.d/ssh status #Comprobar o estado do servidor SSH, por defecto non está arrancado.
   root@kaliA:~# nc -vz localhost 22 #Mediante o comando nc(netcat) comprobar se o porto 22 do servidor ssh está en estado escoita(listen), esperando conexións. A opción -v corresponde á opción verbose, o que permite amosar información máis detallada na saída do comando. A opción -z permite devolver PROMPT do sistema e de igual xeito facer o escaneo ao/s porto/s solicitados. O número 22 é o porto TCP a escanear.
   root@kaliA:~# nc -vz 192.168.120.100 22 #Mediante o comando nc(netcat) comprobar se o porto 22 do servidor ssh está en estado escoita(listen), esperando conexións. A opción -v corresponde á opción verbose, o que permite amosar información máis detallada na saída do comando. A opción -z permite devolver PROMPT do sistema e de igual xeito facer o escaneo ao/s porto/s solicitados. O número 22 é o porto TCP a escanear.
   root@kaliA:~# netstat -natp | grep 22 #Mediante o comando netstat comprobar que o porto 22 do servidor SSH está en estado escoita(listen), esperando conexións. A opción -n permite non resolver nomes amosando así soamente as IPs e o comando ser máis rápido na execución. A opción -a equivale á opción all o que permite amosar todos os sockets (conectores) á escoita no servidor. A opción -t equivale a tcp o que permite buscar soamente información sobre o protocolo TCP. A opción -p equivale a program e amosa o PID e nome do programa ao cal pertence o socket.
   root@kaliA:~# ss -natp | grep 22 #Mediante o comando ss comprobar que o porto 22 do servidor SSH está en estado escoita(listen), esperando conexións. A opción -n permite non resolver nomes amosando así soamente as IPs e o comando ser máis rápido na execución. A opción -a equivale á opción all o que permite amosar todos os sockets (conectores) á escoita no servidor. A opción -t equivale a tcp o que permite buscar soamente información sobre o protocolo TCP. A opción -p equivale a program e amosa o PID e nome do programa ao cal pertence o socket.
   root@kaliA:~# /etc/init.d/ssh start #Arrancar o servidor SSH.
   root@kaliA:~# /etc/init.d/ssh status #Comprobar o estado do servidor SSH, agora debe estar arrancado.
   root@kaliA:~# find /etc/rc* -name "*ssh*" #Busca polas links runlevels nos cartafoles /etc/rc*
   root@kaliA:~# systemctl enable ssh #Permite que o servizo ssh sexa iniciado no arranque xerando os links nos runlevels (/etc/rcX.d)
   root@kaliA:~# find /etc/rc* -name "*ssh*" #Busca polas links runlevels nos cartafoles /etc/rc*

root@kaliA:~# systemctl is-enabled ssh.service #Amosa se o servizo ssh está enabled ou disabled

root@kaliA:~# nc -vz 192.168.120.100 22 #Mediante o comando nc(netcat) comprobar se o porto 22 do servidor ssh está en estado escoita(listen), esperando conexións. A opción -v corresponde á opción verbose, o que permite amosar información máis detallada na saída do comando. A opción -z permite devolver PROMPT do sistema e de igual xeito facer o escaneo ao/s porto/s solicitados. O número 22 é o porto TCP a escanear.

root@kaliA:~# ssh -v kali@localhost #Comprobar se o servidor SSH está activo e podemos conectarnos a el dende localhost co usuario kali e o seu contrasinal. Se é a primeira ver que nos conectamos o servidor avísanos se estamos de acordo coa autenticación. Respostamos yes e pulsamos Enter. A opción -v (modo verbose) aporta información máis detallada da conexión.

kali@kaliA:~$ exit #Saír da consola remota ssh a que acabamos de acceder, para voltar á consola local de **root**.

root@kaliA:~# exit #Saír da consola local sudo na que estabamos a traballar para voltar á consola local de **kali**.

kali@kaliA:~$

Máquina virtual B: Kali amd64

4. Cambiar hostname da máquina virtual B. Por kaliB como hostname:

   Na contorna gráfica abrir un terminal e executar:

   kali@kali:~$ setxkbmap es #Cambiar o mapa de teclado ao idioma español.
   kali@kali:~$ sudo su - #Acceder á consola de root(administrador) a través dos permisos configurados co comando sudo (/etc/sudoers, visudo)
   Opción A:
   root@kali:~# echo 'kaliB' > /etc/hostname #Indicar ao sistema o valor do hostname.
   root@kali:~# echo 'kernel.hostname=kaliB' >> /etc/sysctl.conf #Indicar ao kernel o valor do hostname.
   root@kali:~# sysctl -p #Activar o cambio de hostname sen ter que pechar sesión nin reiniciar
   root@kali:~# exit #Saír da consola local sudo na que estabamos a traballar para voltar á consola local de **kali**.
   kali@kali:~$ exit #Pechar o terminal saíndo da consola local do usuario **kali**.
   Opción B:
   root@kali:~# hostnamectl hostname kaliB #Modificar o valor do hostname a kaliB.
   root@kali:~# exit #Saír da consola local sudo na que estabamos a traballar para voltar á consola local de **kali**.
   kali@kali:~$ exit #Pechar o terminal saíndo da consola local do usuario **kali**.

5. Configurar a rede:

   Na contorna gráfica abrir un terminal e executar:

   kali@kaliB:~$ sudo su - #Acceder á consola de root(administrador) a través dos permisos configurados co comando sudo (/etc/sudoers, visudo)
   root@kaliB:~# /etc/init.d/avahi-daemon stop || systemctl stop avahi-daemon #Parar o demo avahi-daemon(control resolución de nomes) para poder configurar de forma manual a configuración de rede e non ter conflicto con este demo.
   root@kaliB:~# /etc/init.d/network-manager stop || pkill NetworkManager #Parar o demo network-manager(xestor de rede) ou o script NetworkManager (executado sen ser demo) para poder configurar doutro xeito (co comando ip(ifconfig) de forma manual ou mediante networking (ficheiros /etc/init.d/networking, /etc/init.d/networking.d) a configuración de rede e non ter conflicto con este xestor.
   root@kaliB:~# ip addr show #Amosar a configuración de todas as tarxetas de rede. Nesta caso, na máquina B as tarxetas de redes: loopback(lo) e interna(eth0).
   root@kaliB:~# ip addr add 192.168.120.101/24 dev eth0 #Configurar a tarxeta de rede interna eth0, coa IP: 192.168.120.101 e máscara de subrede: 255.255.255.0.
   root@kaliB:~# ip addr show #Amosar a configuración de todas as tarxetas de rede. Nesta caso, na máquina B as tarxetas de redes: loopback(lo) e interna(eth0).
   root@kaliB:~# ping -c4 192.168.120.101 #Comprobar mediante o comando ping a conectividade coa interface de rede local eth0
   root@kaliB:~# ping -c4 192.168.120.100 #Comprobar mediante o comando ping a conectividade coa interface de rede da máquina virtual A
   root@kaliB:~# echo '192.168.120.100 kaliA' >> /etc/hosts #Engadir no ficheiro /etc/hosts, é dicir, na táboa estática de búsqueda para nomes de host (DNS) o nome kaliA, para que atenda á IP 192.168.120.100
   root@kaliB:~# ping -c4 kaliA #Comprobar mediante o comando ping a conectividade coa interface de rede da máquina virtual A

6. **B → A** SSH Comprobar o acceso SSH dende a máquina virtual B á máquina virtual A:

   No terminal anterior executar:

   root@kaliB:~# echo '192.168.120.100 kaliA' >> /etc/hosts #Engadir no ficheiro /etc/hosts, é dicir, na táboa estática de búsqueda para nomes de host (DNS) o nome kaliA, para que atenda á IP 192.168.120.100
   root@kaliB:~# nc -vz 192.168.120.100 22 #Mediante o comando nc(netcat) comprobar que o porto 22 do servidor SSH está en estado escoita(listen), esperando conexións. A opción -v corresponde á opción verbose, o que permite amosar información máis detallada na saída do comando. A opción -z permite devolver PROMPT do sistema e de igual xeito facer o escaneo ao/s porto/s solicitados. O número 22 é o porto TCP a escanear.
   root@kaliB:~# nc -vz kaliA 22 #Mediante o comando nc(netcat) comprobar que o porto 22 do servidor SSH está en estado escoita(listen), esperando conexións. A opción -v corresponde á opción verbose, o que permite amosar información máis detallada na saída do comando. A opción -z permite devolver PROMPT do sistema e de igual xeito facer o escaneo ao/s porto/s solicitados. O número 22 é o porto TCP a escanear.
   root@kaliB:~# ssh -v kali@kaliA #Comprobar se o servidor SSH está activo e podemos conectarnos a el. Agora accedemos como o usuario **kali** a través da conexión cifrada SSH.
   kali@kaliA:~$ exit #Saír da consola remota ssh a que acabamos de acceder, para voltar á consola local de **kali** na máquina B.
   root@kaliB:~#

Máquina virtual A: Kali amd64

7. Crear o directorio do proxecto
   kali@kaliA:~$ rm -rf probando-git # Eliminar o cartafol probando-git
   kali@kaliA:~$ mkdir probando-git # Crear unha carpeta chamada probando-git como o directorio do proxecto
   kali@kaliA:~$ cd probando-git # Entrar na carpeta do proxecto

8. Inicializar o repositorio para compartir
   kali@kaliA:~$ git init # Inicializar un novo repositorio de Git

   $ git init --bare # Inicializar un novo repositorio de Git, para compartir código entre desenvolvedores ou máquinas e non se desexa que ninguén edite os arquivos directamente nese repositorio.
   Así, se se crea un repositorio bare con *git init --bare* en kaliA, ninguén pode editar directamente os arquivos do proxecto no repositorio bare. Os repositorios bare están deseñados unicamente para actuar como punto central de sincronización mediante operacións de Git como *clone*, *push*, *fetch* ou *pull*.

9. Configurar usuario para este proxecto
   kali@kaliA:~$ git config user.name "Teu Nome" # Configurar o nome do usuario para definir o nome do usuario que se asociará cos commits que realices no repositorio. Este nome aparecerá como parte da autoría de cada commit.
   kali@kaliA:~$ git config user.email "teu.email@example.local" # Configurar o email do usuario. Soe combinarse co comando anterior para configurar a información de autoría dos teus commits.
   kali@kaliA:~$ git config -l # Amosa todas as configuracións activas (local, global, e de sistema). Ficheiros: .git/config, ~/.gitconfig, /etc/gitconfig
   kali@kaliA:~$ git config --local -l # Amosa as configuracións locais para o repositorio actual. Ficheiro: .git/config
   kali@kaliA:~$ git config --system -l # Amosa as configuracións do sistema. Ficheiro: /etc/gitconfig
   kali@kaliA:~$ git config --global -l # Amosa as configuracións globais. Ficheiro: ~/.gitconfig

   Orde de prevalencia:

   Local → Global → System

   Explicación:
   As configuracións locais (para o repositorio actual) teñen prioridade sobre as globais e de sistema. Isto significa que, se hai unha configuración tanto no ficheiro local como no global ou de sistema, será a configuración local a que se aplique.
   As configuracións globais teñen prioridade sobre as de sistema, pero só se non hai unha configuración no nivel local.
   Exemplo:
   Se configuras un *user.name* de forma global e logo estableces un *user.name* diferente no repositorio actual, a configuración local será a que se aplique para ese repositorio.

10. Crear un ficheiro e engadilo ao repositorio
    kali@kaliA:~$ echo "Primeiro contido" > ficheiro.txt # Crear un ficheiro con contido inicial no repositorio
    kali@kaliA:~$ git add ficheiro.txt # Engadir o ficheiro ao staging area
    kali@kaliA:~$ git commit -m "Primeiro commit: Engadir ficheiro.txt" # Crear o primeiro commit

11. Crear unha nova rama para desenvolver unha funcionalidade
    kali@kaliA:~$ git branch nova-funcionalidade # Crear unha rama chamada nova-funcionalidade
    kali@kaliA:~$ git checkout nova-funcionalidade # Cambiar á nova rama

12. Modificar o ficheiro e facer outro commit
    kali@kaliA:~$ echo "Engadir nova funcionalidade" >> ficheiro.txt # Engadir liña ao ficheiro
    kali@kaliA:~$ git add ficheiro.txt # Engadir os cambios ao staging area
    kali@kaliA:~$ git commit -m "Nova funcionalidade: Actualizar ficheiro.txt" # Crear un commit na nova rama

13. Volver á rama principal
    kali@kaliA:~$ git checkout master # Cambiar á rama principal

14. Fusionar os cambios da nova rama na rama principal
    kali@kaliA:~$ git merge nova-funcionalidade # Fusionar a rama nova-funcionalidade en main

15. Verificar o historial de commits
    kali@kaliA:~$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.

16. Crear un novo ficheiro e movelo
    kali@kaliA:~$ echo "Contido adicional" > outro.txt # Crear un segundo ficheiro

kali@kaliA:~$ git add outro.txt # Engadir o ficheiro ao staging area
kali@kaliA:~$ git commit -m "Engadir outro ficheiro" # Crear un commit
kali@kaliA:~$ git mv outro.txt cambiado.txt # Renomear o ficheiro
kali@kaliA:~$ git commit -m "Renomear outro.txt a cambiado.txt" # Commit do cambio de nome

17. Amosar as diferencias entre commits
    kali@kaliA:~$ git diff HEAD~1 HEAD # Ver as diferenzas co commit anterior

18. Simular un cambio crítico e correxilo
    kali@kaliA:~$ git checkout -b hotfix # Crear unha rama hotfix para correccións urxentes
    kali@kaliA:~$ echo "Corrección crítica" >> ficheiro.txt # Engadir unha corrección
    kali@kaliA:~$ git add ficheiro.txt # Engadir os cambios
    kali@kaliA:~$ git commit -m "Corrección crítica na rama hotfix" # Crear un commit de corrección
    kali@kaliA:~$ git checkout master # Cambiar de volta á rama principal
    kali@kaliA:~$ git merge hotfix # Fusionar a corrección na rama principal

19. Eliminar ramas que xa non se necesitan
    kali@kaliA:~$ git branch -d nova-funcionalidade # Eliminar a rama nova-funcionalidade
    kali@kaliA:~$ git branch -d hotfix # Eliminar a rama hotfix

Máquina virtual B: Kali amd64

20. Clonación do repositorio

kali@kaliB:~$ git clone kali@kaliA:probando-git # Clonar o repositorio de kaliA
kali@kaliB:~$ cd probando-git # Entrar no directorio clonado
kali@kaliB:~$ git branch # Lista todas as ramas locais. A rama actual aparece marcada con * antes do nome.

Cando fas un *git clone*, por defecto **só se clona a rama principal** (normalmente master ou main), e as outras ramas remotas non se crean automaticamente como ramas locais. Polo tanto, despois de clonar o repositorio en kaliB, soamente ves a rama master.

Como ver todas as ramas remotas en kaliB?
Despois de clonar, podes listar todas as ramas dispoñibles no repositorio remoto con:

```
$ git branch -r
```

Isto mostrará algo como:

```
  origin/HEAD -> origin/master
  origin/hotfix
  origin/master
  origin/nova-funcionalidade
```

Como traer as outras ramas para traballar con elas?
Se queres traballar nunha rama específica (por exemplo, *hotfix*), debes creala localmente baseándote na versión remota con:

```
$ git checkout -b hotfix origin/hotfix
```

```
Ou, se estás usando unha versión recente de Git:
```

```
$ git switch -c hotfix origin/hotfix
```

```
Agora podes facer cambios no código:
```

```
$ git status
$ git log --onefile --graph
$ echo 'Comprobando cambios na rama hotfix' >> comprobando.txt
$ git add comprobando.txt
$ git status
$ git config --global user.email "you@example.com"
$ git config --global user.name "Your Name"
$ git commit -m "Cambios na rama hotfix"
```

```
E para enviar os cambios ao repositorio remoto (o de kaliA):
```

```
$ git push origin hotfix
```

```
Podemos comprobalo en kaliA:
```

```
┌──(kali㉿kaliA)-[~/probando-git]
└─$ git checkout hotfix
Switched to branch 'hotfix'

┌──(kali㉿kaliA)-[~/probando-git]
└─$ git log --oneline --graph
* 5ed494f (HEAD -> hotfix) Cambios na rama hotfix
* f8e369f (master) Corrección crítica na rama hotfix
* 6eafbea Renomear outro.txt a cambiado.txt
* 83007cf Engadir outro ficheiro
* 7e73048 (nova-funcionalidade) Nova funcionalidade: Actualizar ficheiro.txt
* 331bed4 Primeiro commit: Engadir ficheiro.txt
```

21. Crear ficheiros e subir cambios

    kali@kaliB:~$ echo "Primeiro ficheiro en kaliB" > ficheiro.txt # Crear un ficheiro inicial
    kali@kaliB:~$ git add ficheiro.txt # Engadir o ficheiro ao staging area
    kali@kaliB:~$ git config user.name "Your Nome" # Configurar o nome do usuario para definir o nome do usuario que se asociará cos commits que realices no repositorio. Este nome aparecerá como parte da autoría de cada commit.
    kali@kaliB:~$ git config user.email "you@example.local" # Configurar o email do usuario. Soe combinarse co comando anterior para configurar a información de autoría dos teus commits.
    kali@kaliB:~$ git commit -m "Primeiro commit dende kaliB" # Crear o commit
    kali@kaliB:~$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.
    kali@kaliB:~$ git push origin master:master-temp # Subir os cambios a `kaliA` a unha nova rama temporal master-temp
    kali@kaliB:~$ ssh kali@kaliA "cd ~/probando-git && git checkout master && git merge master-temp" # En kaliA, cambiar á rama master e facer merge

22. Acceder por SSH a kaliA e comprobar e verificar cambios

    kali@kaliB:~$ ssh kali@kaliA "cd ~/probando-git && git log --oneline --graph" # Entrar en kaliA no repositorio e Verificar o commit enviado desde kaliB

23. Actualizar e facer máis cambios

    kali@kaliB:~$ ssh kali@kaliA "cd ~/probando-git && echo 'Novos datos' > ficheiro.txt && echo 'Arquivo novo' > novo.txt" # Entrar en kaliA: modificar un arquivo e crear outro
    kali@kaliB:~$ ssh kali@kaliA "cd ~/probando-git && git add ficheiro.txt novo.txt" # Engadir en kaliA os cambios ao staging area
    kali@kaliB:~$ ssh kali@kaliA "cd ~/probando-git && git commit -m 'Cambios realizados dende kaliA'" # Crear en kaliA un commit cos cambios
    kali@kaliB:~$ git fetch origin # Traer cambios do repositorio remoto
    kali@kaliB:~$ git status # Ver o estado actual do repositorio. En kaliB o repositorio está 1 commit por detraś do repositorio de kaliA
    kali@kaliB:~$ git pull origin master # Actualizar o directorio local
    kali@kaliB:~$ echo "Engadir unha nova funcionalidade" >> ficheiro.txt # Modificar o ficheiro
    kali@kaliB:~$ git add ficheiro.txt # Engadir os cambios ao staging area
    kali@kaliB:~$ git commit -m "Nova funcionalidade en kaliB" # Crear un commit cos cambios
    kali@kaliB:~$ git push origin master:master-temp # Subir os cambios a `kaliA` a unha nova rama temporal master-temp
    kali@kaliB:~$ ssh kali@kaliA "cd ~/probando-git && git checkout master && git merge master-temp" # En kaliA, cambiar á rama master e facer merge
    kali@kaliB:~$ ssh kali@kaliA "cd ~/probando-git && git log --oneline --graph" # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.

Máquina virtual A: Kali amd64

24. Realizar commits:

```
$ cd ~/probando-git
$ git checkout master
$ for in in $(seq 1 6); do echo $i > file$i.txt; done
$ git status
$ git add file[1-6].txt
$ git status
$ git commit -m "6 novos ficheiros"
$ git log --oneline --graph
```

Máquina virtual B: Kali amd64

25. Descargar cambios:

```
$ cd ~/probando-git
$ ls -l
$ git log --oneline --graph
$ git fetch  #Descarga cambios do repositorio remoto pero non os aplica á rama local.
$ git diff --name-only master origin/master #Ver os ficheiros descargados.
$ ls -l
$ git log --oneline --graph
$ git pull  #Descarga os cambios (como fetch) e mestúraos automaticamente na rama local (fetch + merge).
$ ls -l
$ git log --oneline --graph
```

Ricardo Feijoo Costa