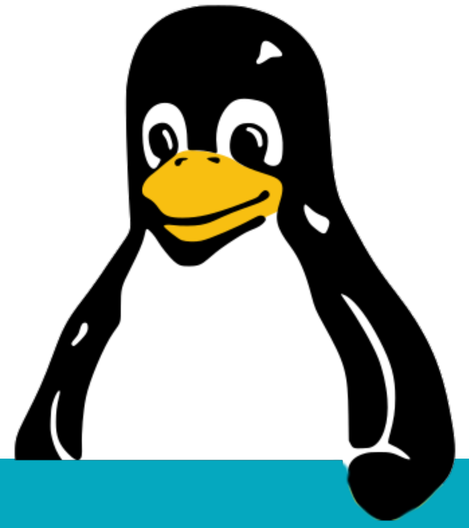
Linux, day 11



Objectives covered

Objective	Summary	Boek
1.1	Device types and hardware	23
1.4	Process management	21
1.4	Scheduling	26
3.3	Git operations	27
3.4	Advanced Git operations	27

LAB: Job control

A bit of preparation

- You will probably need to install *at* and *cron*.
- For example, on Fedora:
  - Install, enable and start *crond*.
- On Ubuntu, it's called *cron*.

Assignment 1

- Create a small shell script "*~/at-test.sh*".
  - This should append "Hello!" to */tmp/at-test.txt.*
  - It should also include the output of "*date*".

- Use *"at*" to run it in 1 minute.
- Use "*crontab*" to schedule it for every minute.

Assignment 2

- Create a small shell script */opt/mysvc.sh*
  - This should write into */var/log/mysvc.log*
  - First test it manually.

- Use this: Create a simple systemd service unit.
  - Make *mysvc.sh* a systemd service that runs at reboot.
  - Prove that it works.

You try…

```
$ git config --global user.name \
"Tess Sluijter"

$ git config --global user.email \
tess@unixerius.nl
```

You try…

```
$ mkdir ~/gitdemo
$ cd ~/gitdemo
$ git init
$ ls -al
$ ls -al .git
```

You try…

```
$ git status

$ echo "Hallo" > readme.txt
$ git status
```

You try…

```
$ git add readme.txt
$ git status

$ git commit -m "My first file"
$ git status
```

You try…

```
$ git log

$ rm readme.txt

$ git reset readme.txt
$ ls
```

You try…

```
$ git branch bugfix
$ git checkout bugfix

$ nano readme.txt  # make some changes
$ git add readme.txt
$ git commit -m "Fixed it!"
```

You try…

```
$ ls -al; cat readme.txt

$ git checkout master          # … back
$ ls -al; cat readme.txt

$ git checkout bugfix          # … and forth
$ ls -al; cat readme.txt
```

You try…

```
$ git checkout master
$ echo "Hallo" > newfile
$ git add newfile

$ git commit -m "New file made."
```

You try…

```
$ git checkout bugfix
$ ls newfile

$ git merge master
$ ls newfile
```

You try…

```
$ git checkout master
$ cat readme.txt

$ git merge bugfix
$ cat readme.txt
```

You try…

```
$ cd ~/Documents

$ git clone \
https://github.com/tsluyter/exploits

$ cd exploits; git log
```

Recap: Git

- The "Git" command has many sub-commands.
- Git tracks changes to files.
- Git can give you past versions of files.

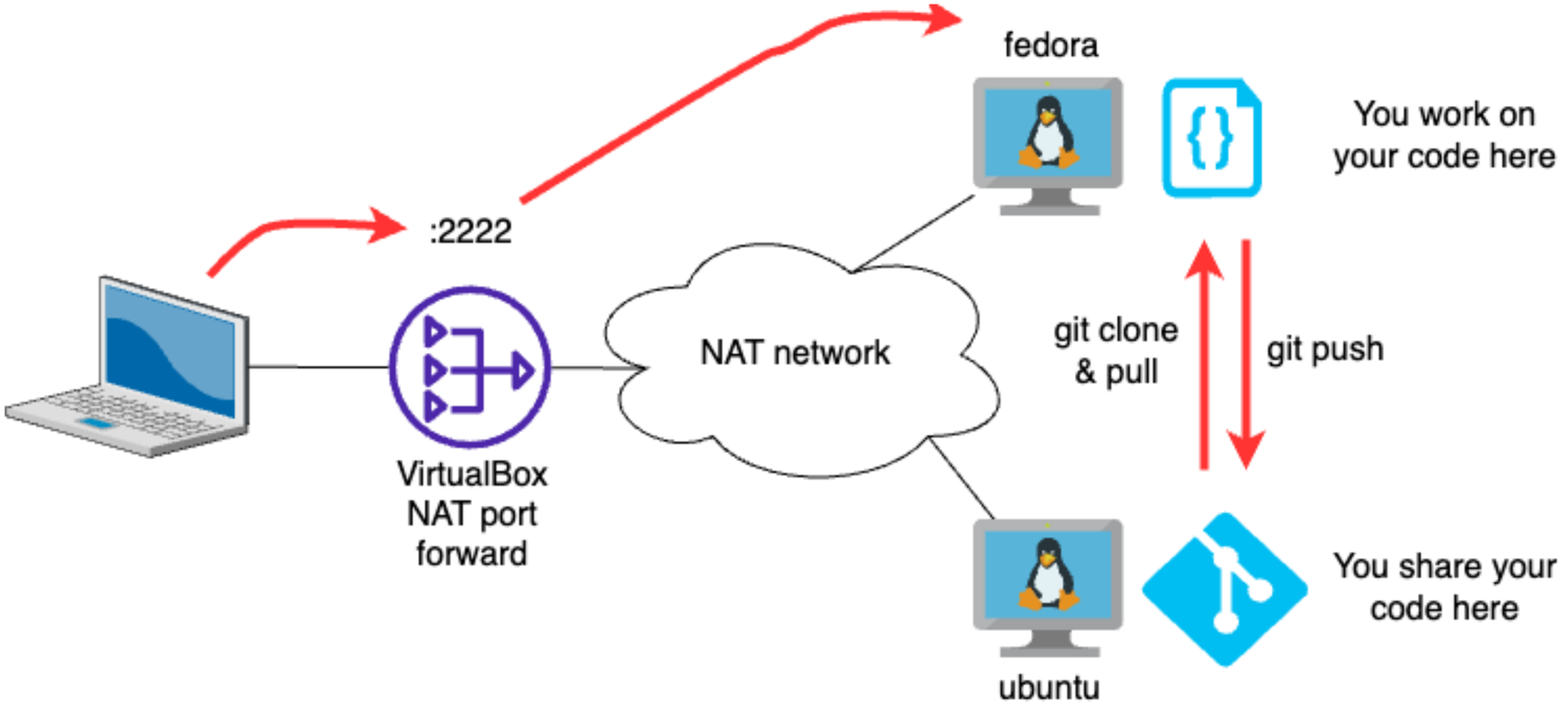
LAB: Git

What's the point again?

- Teams want to work together on the same code.
  - When code is ready for release,
  - The central copy will be pushed to production.

Your own, "remote" repo

- Let’s make the Ubuntu VM our **Git server**.



Setting up the server

- On the Ubuntu VM, make user account "git".
  - With homedir "*/home/git*".
  - And a password you won’t mind typing.

- Test that you can SSH from Fedora,
  - To the user "git" on the new VM.

Making a repo

- On the **Ubuntu VM**, login as user "git".
  - Configure their name and email (slide 63).

- Make the dir "*/home/git/firstrepo*".
- "*cd*" into "*firstrepo*" and init a Git repo.
  - Use: "*git init --bare*"!!
- See: Bare vs non-bare repositories

Cloning the repo

- On the **Fedora VM**, login as yourself.
- "*cd*" into your home directory.
- Clone the repository from the new VM:

```
$ git clone ssh://git@ubuntu:/home/git/firstrepo
```

Making a change

- On the **Fedora VM**, "*cd*" into the Git repo.
  - *cd ~/firstrepo*

- Make a new file and commit the change.
- Then "*git push*" the update.

Comparing

- Compare the contents of:
  - The cloned git repo on your Fedora box.
  - The bare repo on the Ubuntu VM.
  - "*git log*" on the two repository locations.

- Research question: where are the files on Ubuntu?!

Co-working

- On the **Fedora VM**, login as user “dummy”.
- Make sure you're in the homedir of "dummy".
- Clone the repository from the new VM:

```
$ git clone ssh://git@ubuntu:/home/git/firstrepo
```

Co-working

- Do you see the file(s) you just pushed?
- Now make another file, as dummy.
- Commit and push it.

- Then switch back to your own account.
- And “*git pull*”. Does the changed file show up?

Spoilers - on Ubuntu

- With your own account …

```
$ sudo useradd -m -s /bin/bash git

$ sudo passwd git

$ su - git
```

Spoilers - on Ubuntu

- You are now the “git user”…

```
$ git config --global user.name "Git"
$ git config --global user.email "git@ubuntu"

$ mkdir firstrepo; cd firstrepo
$ git init --bare
$ exit
```

Spoilers - on Fedora

- With your own account.

```
$ cd ~

$ git clone ssh://git@ubuntu:/home/git/firstrepo

$ cd firstrepo
```

Spoilers - on Fedora

- With your own account.

```
$ echo "Hoi." > readme.txt
$ git add readme.txt
$ git commit -m "My first file."
$ git push

$ su - dummy
```

Spoilers - on Fedora

- Now you are user "dummy".

```
$ git config --global user.name "Dummy"
$ git config --global user.email "dummy@fedora"

$ git clone ssh://git@ubuntu:/home/git/firstrepo
$ cd firstrepo
```

Spoilers - on Fedora

- You are still user "dummy".

```
$ echo "Dummy wrote this." > dummy.txt
$ git add dummy.txt
$ git commit -m "Dummy file."
$ git push

$ exit
```

Spoilers - on Fedora

- You are now using your own account again.

```
$ cd ~/Documents/firstrepo

$ git pull

$ ls -al
$ cat dummy.txt
```

Recap: Git server

- Git can be used to managed a local directory.
- Git can also be hosted via HTTP(S) and SSH.
- You can clone a hosted repository.
- Pull and pushing is to share code updates.

Closing

Homework

- Reading:
  - Chapter 9, entirely.
  - Chapter 13, entirely.
  - Chapter 17, entirely.

Homework

- Finish the Git server-client lab.
  - The other homework relies on it.
  - You need to have a working Git server on Ubuntu.

Homework

- Let's make a "Scripts" repository:
- Login to your Ubuntu VM and "su - git". Run the following commands:
  - hostname
  - whoami # this should be user git
  - mkdir Scripts
  - cd Scripts
  - git init --bare
  - exit

Homework

- Login to your account on Fedora and run these commands.
  - hostname
  - whoami
  - cd ~
  - git clone ssh://git@ubuntu:/home/git/Scripts
  - cd Scripts
  - echo "#\!/bin/bash" > hello-world.sh
  - echo "echo \"Hello world\!\"" >> hello-world.sh
  - git add hello-world.sh
  - git commit -m "My first script, a hello-world sample."
  - git push

Reference materials

Resources

- Sending signal to processes
- How to use tmux (and why it's better than screen)
- Create a simple systemd service unit.
- crontab.guru
- How to write basic udev rules
- An introduction to udev
- Why use cpio for initramfs

Resources

- Learning about systemd timers
- Cron, Anacron, Systemd and scheduling
- ArchWiki Systemd timers
- TTY demystified

Resources

- Git internals
- Free book: Pro Git
- Gamified Git-learning: Oh my Git!
- Intro to Git for security professionals
- Bare vs non-bare repositories
- In-depth: So you think you know Git?
- In-depth: So you think you know Git? Part 2

Resources

- PluralSight learning path for XK0-005
- Creating shell scripts in Enterprise Linux
- Small set of scripting exercises (U of C)
- More scripting exercises (TLDP)
- CertDepot.net daily Linux tasks.

Resources

- Linux Upskill Challenge
- KodeKloud Engineer
- Exercism.org, with Bash exercises
- Sander van Vugt's RHCSA practice tests
- Linux Foundation's LFCS practice questions
- RedHat online practice labs