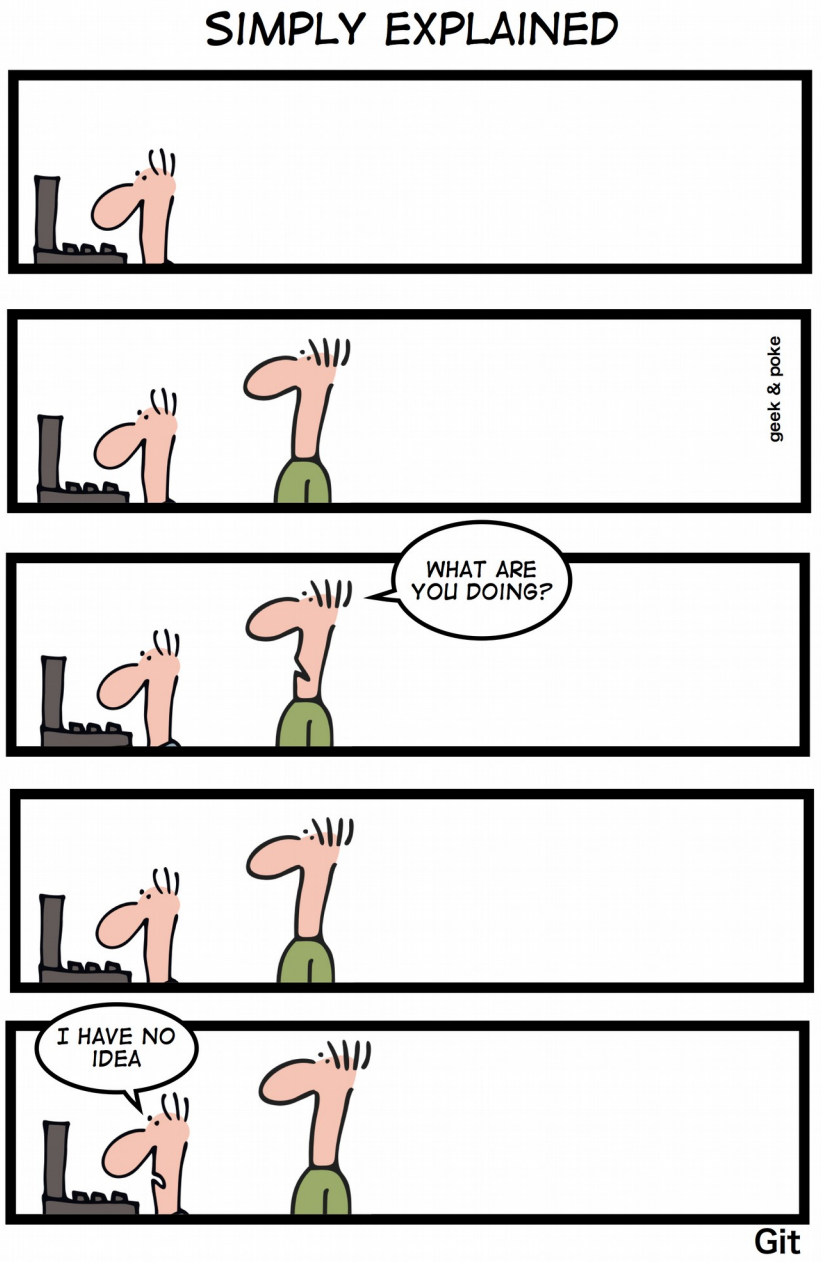
Speaking about securing code, let start with git

Anne NICOLAS - hupstream

git

WTF ! I don't get how it works !



whoami

I'm not

- A developper
- A fan of nodejs
- A member of $(projectx)'s for woman
- A singer

I'm

- A very old fan of Open Source Software
- A contributer of some Open Source projects
- One of the founder of hupstream
- Kernel Recipes conference organizer

hupstream

Why this talk about git?

- I was reported to be a potential speaker by a (suri)cat(a) lover

- I've been a git trainer for 2 years: more than 150 trainees and 20 sessions

The vision of our client
< 10 % know nothing about git
50 % know how to deal with basic commands
40 % are advanced users

What we see really
>50 % know nothung about git
40 % know how to deal with basic commands
< 10 % are advanced users

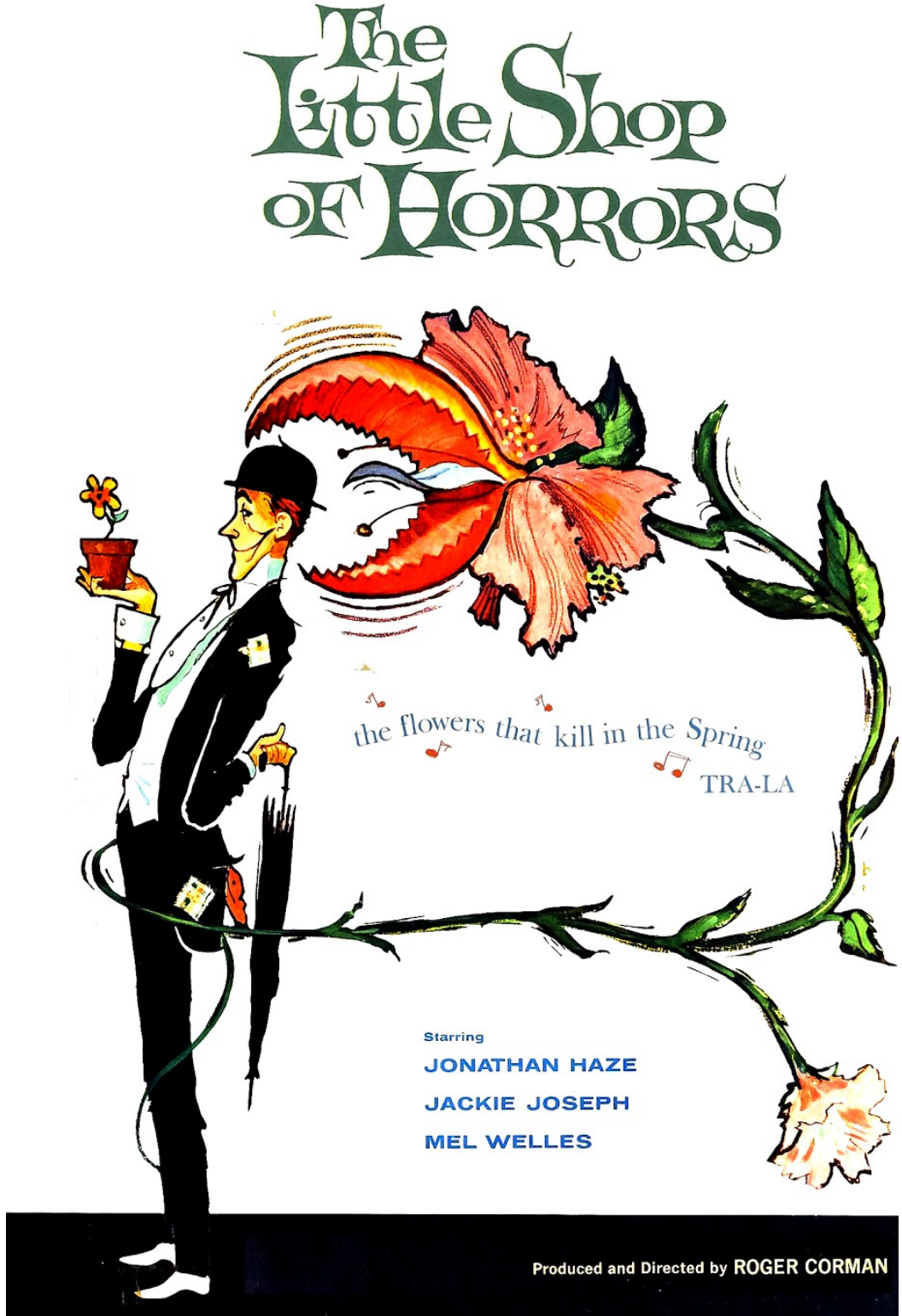
- Any kind of company or administration migrating to git or setting up a new VCS

hupstream

Git in Real Life: The Little Shop of Horrors

hupstream

Git in real life



On web servers side

Deploying a web site on a web server

- Clone of an existing repository /branch
- Use of an unclean archive of an existing project

Non protected .git repository

- Indexes allowed
- No specific right on .git directory

Source: https://what-if.xkcd.com

hupstream

On web servers side

Download .git repository

- wget --mirror <url>/mywebsite/.git/

Content of .git directory : thanks guys !

```
├── info
├── logs
├── objects
│   ├── 0d
│   ├── 0f
│   ├── info
│   └── pack
└── refs
    ├── heads
    ├── remotes
    │   └── origin
    └── tags
```

```
git cat-file -p <SHA1>
git checkout -- <file>
```

hupstream

Oups I dit it... (yes that's true!)

I've added a password in a git tracked file

- Clone of an existing repository /branch (git rebase -i, git filter-branch)
- Use of an unclean archive of an existing project

I've sent my private key on a public mailing-list

- Very low level of GPG and ssh knowledge
- No training for new VCS and good practice

Let's migrate to git !

- No training for people working on the project
- No policy for using git and securing code
- No defined workflow for branches

hupstream

Git foundations

hupstream

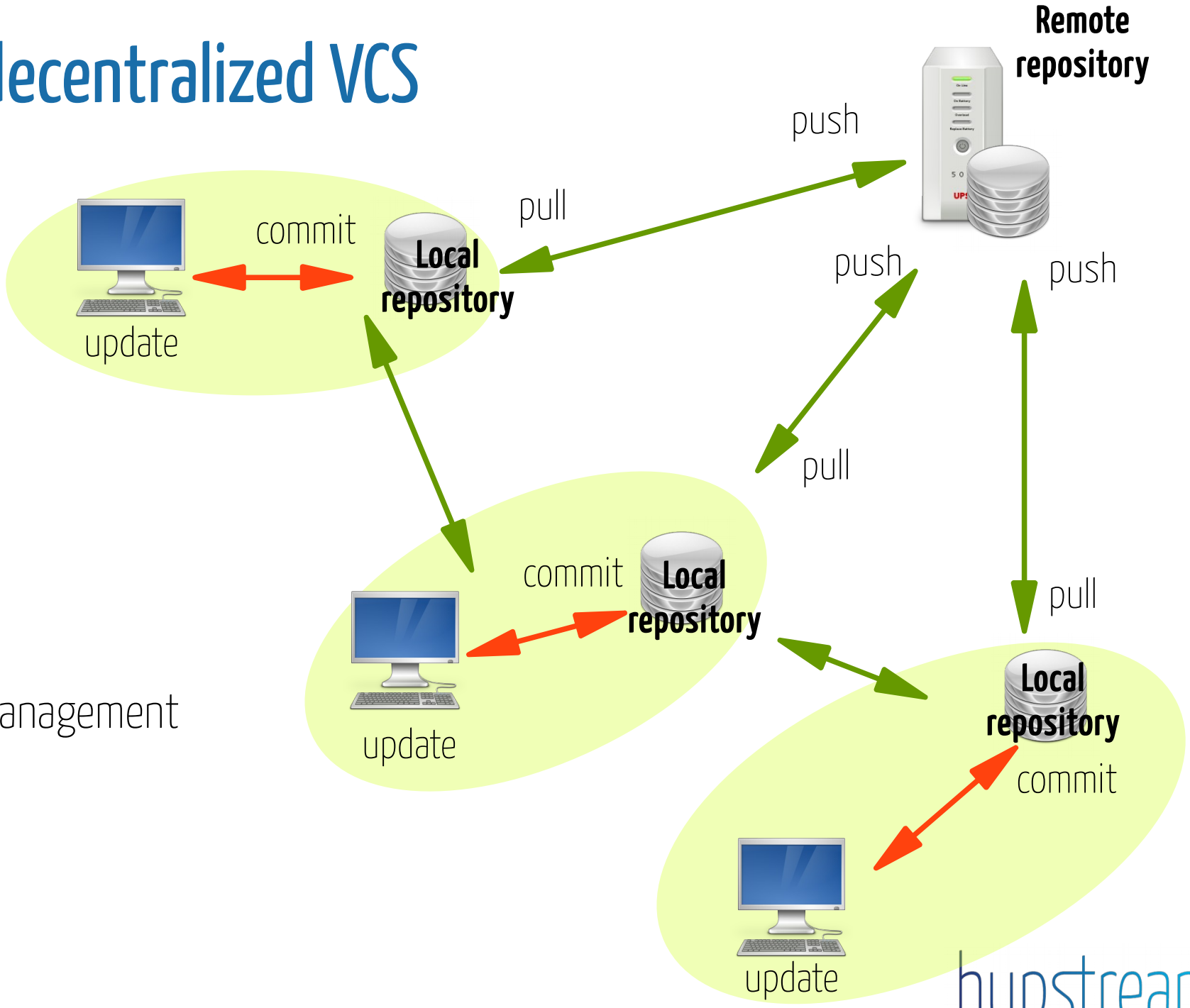
Git objects: some basic statements to start with

Data integrity and consistency versus data security

*« **Data integrity** refers to maintaining and assuring the accuracy and consistency of data over its entire life-cycle, and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data. »* - Wikipédia

« The point is the SHA-1, as far as Git is concerned, isn't even a security feature. It's purely a consistency check. The security parts are elsewhere, so a lot of people assume that since Git uses SHA-1 and SHA-1 is used for cryptographically secure stuff, they think that, OK, it's a huge security feature. It has nothing at all to do with security, it's just the best hash you can get. [...] ***Git uses SHA-1 not for security [...] The security parts are elsewhere*** *»* - Linus Torvalds

Git is a decentralized VCS



- availibility
- redundancy
- Disk space management

- complex

hupstream

Git objects: some basic statements to start with

Stored in .git/objects

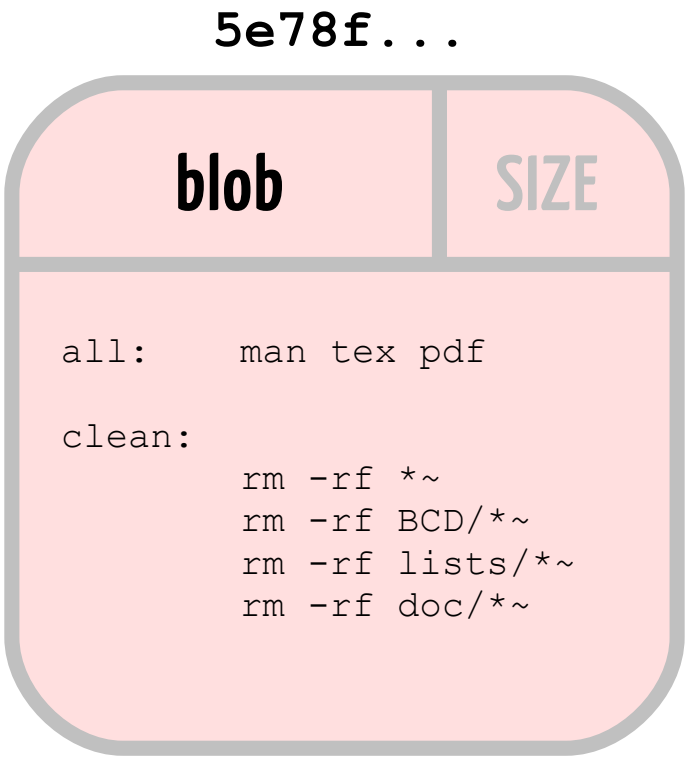
Immutable objects

- Git does not store differences but a complete file
- An existing object cannot be modified, a new one is created

Git objects

Blobs

- File content only
- Uniq identifier (SHA1)
- 2 files with same content use the same blob



Objets Git

Trees

- Index of all blobs and trees, names, Ids (SHA1), dates, permissions...
- Uniq identifier (SHA1)

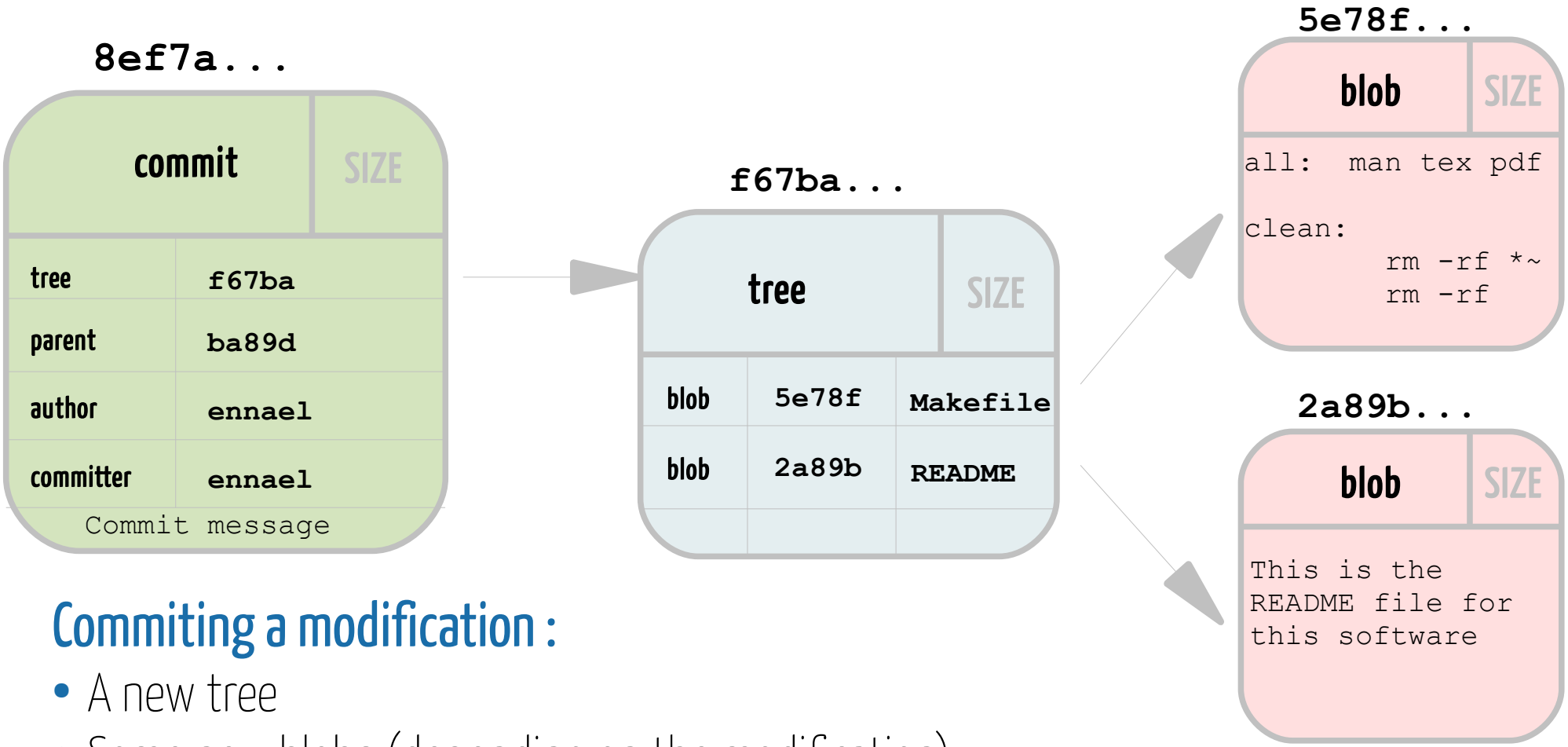
Objets Git

Commits

- A reference to a tree
- Parent commit (previous one)
- Message
- Uniq identifier (SHA1)

hupstream

Let's put all this together...



Commiting a modification :

- A new tree
- Some new blobs (depending on the modification)
- A new commit

hupstream

SHA1 collisions attacks

Low risks for a colision

« *the use cases git is used for are more resistant against collision attacks, than most other SHA-1 use cases. it is just a SHA1 hash, but it is a hash of a structured data format.* »

- Refers to the metadata structure of the git database
- Collision: add modification in repository without SHA1 changes
- More and more binary pieces in git repositories

hupstream

Best pratice for client and server

hupstream

No proper identity policy, no commit

Identification based on user account

```
git config —-global  user.name <name>
git config —-global  user.mail <mail>
```

Git account policy

- No system account
- No privileged users
- Identity is not enough to authenticate users

hupstream

Signed commits and tags

Signed commit

```
gpg --gen-key
gpg --list-keys
git config --global user.signingkey <HASH_KEY>
git config ––global  user.mail <mail>
git commit -S
```

Sign your tag

```
git tag -s <tag> [<SHA1>] -m '<message>'
```

Check your commit / tag

```
git show <SHA1>
```

hupstream

Signed commits and tags

More tools

```
git log --show-signature
git verify-commit <SHA1>
git verify-tag <tag>
```

Signed-off is not signature !

- Signed-off : not a GPG signature but an agreement on the license to be used

```
git commit -s
```

hupstream

Proper policy for efficient signed commits and tags

PGP key policy

- Setup a trust chain for any project contributer to check the PGP key
- Size of the key
- Private key must be protected or revoked
- Avoid infinite validity
- Use password on key

All contributers must sign commits

hupstream

What can happen in case of server attack

4th march, 2012: github

- public key security vulnerability was exploited
- it allowed to successfully commit to the master branch of the Ruby on Rails framework repository hosted on GitHub

August 2011 : kernel.org

- An attack gave root access
- L. Torvalds has its own repository for all the project
- All contributers has a copy of the public repository
- No way to modify existing code without modifying internals or adding new objects
- *« Kernel.org is only a distribution point »* – Jonathan Corbet

hupstream

Secure git server access

Developpers access

- SSH + key authentication
- Git shell

Secured access protocols

- Limited list of protocols
- Limited access for these protocols

Secured policy must also apply to CI

hupstream

Git shell

What for

- a login shell for SSH accounts to provide restricted Git access.
- allows execution only of server-side Git commands (pull/push functionality, list of custom commands)
- Usual tasks : list repositories, create, delete, or rename it, change descriptions and permissions.

Configure it

```
cp $docdir/git/contrib/git-shell-commands $HOME
```

Non interactive access

- -c option
- git receive-pack, git upload-pack, git upload-archive

hupstream

Secure Server

Basic but often forgotten...

- Remove X11 forwarding
- Disable PasswordAuthentication and PermitRootLogin in /etc/ssh/sshd_config
- Enable fail2ban for SSH
- Restart SSH daemon
- Use HTTPS to access any web-based manager for git (gitlab...)
- Firewall off all other ports than 22 and 443
- Update your servers ! Including git,,,