Using Git and GitHub

Tecnología de Videojuegos

Objectives

1. Understand the need of SCM
2. Implement software development workflows with Git and Github

Bibliography

1. GitHub Guides. (Link)

Table of Contents

Software Configuration Management (SCM)

Version Control

Version control systems (VCS) keep track of changes to source code. Allows multiple people to edit a project in a predictable manner.

Software configuration Management (SCM)

Source Configuration Management

Software configuration management is the task of tracking and controlling changes in the software, part of the larger cross-disciplinary field of configuration management.
(`https://en.wikipedia.org/wiki/Software_configuration_management`)
Main open source software configuration management systems

- 1982 RCS
- 1990 CVS
- 2000 Subversion
- 2005 Git/Mercurial

There are many proprietary ones but `Git` is now the most popular one by far.
All software should be under a version control system, if not, it ain't software!

Git

What is Git?

Git is an open source distributed version control system, created by Linus Torvald.

`https://git-scm.com/`

(Interactive tutorial)

Git

Git sites

It is easier to start with free hosting sites instead of maintaining your own server.

- GitHub: public repositories (as many as you want), but private ones are not free (except for academia). It is now part of Microsoft
- Bitbucket: allow us to keep private repositories limiting the number of collaborators.
- GitLab: IT allows both public and private without limitations. It is becoming more popular.
- Others ...

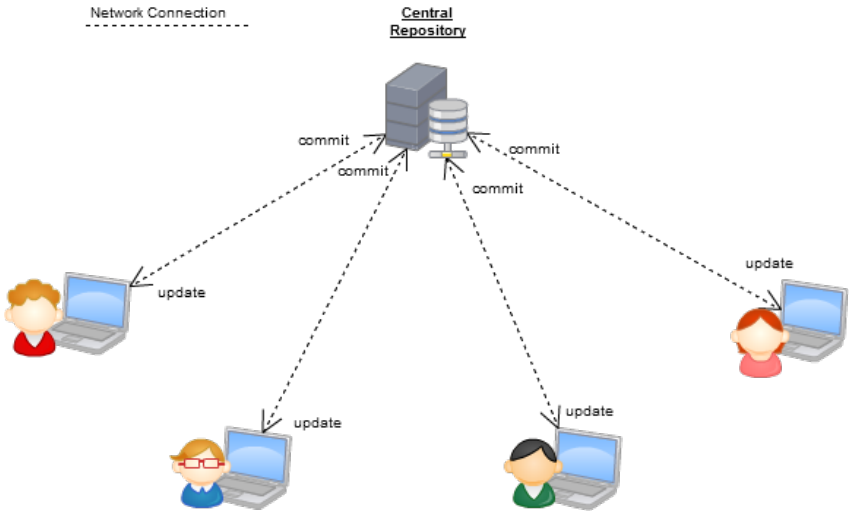
It is typically used as central repository:

- from which everyone pulls other people's changes
- to which everyone pushes changes they have made

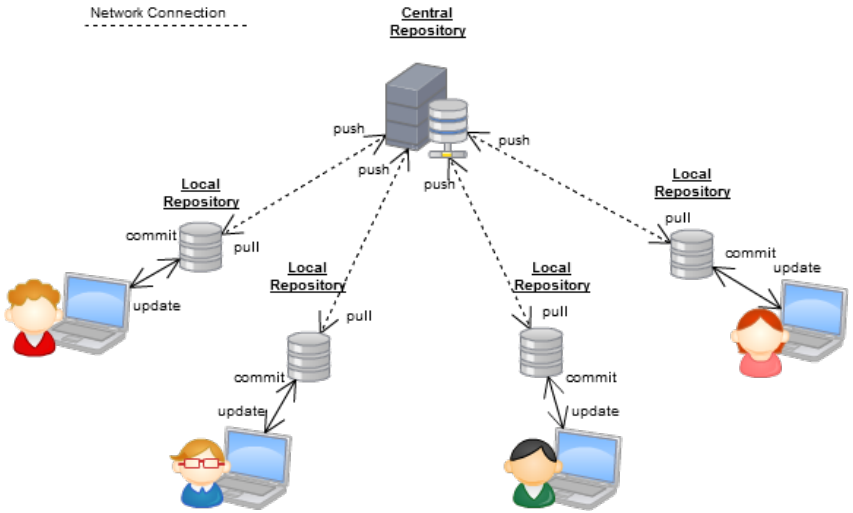
Git

Git vs. SVN (I)

Centralized (SVN)



Distributed (Git)

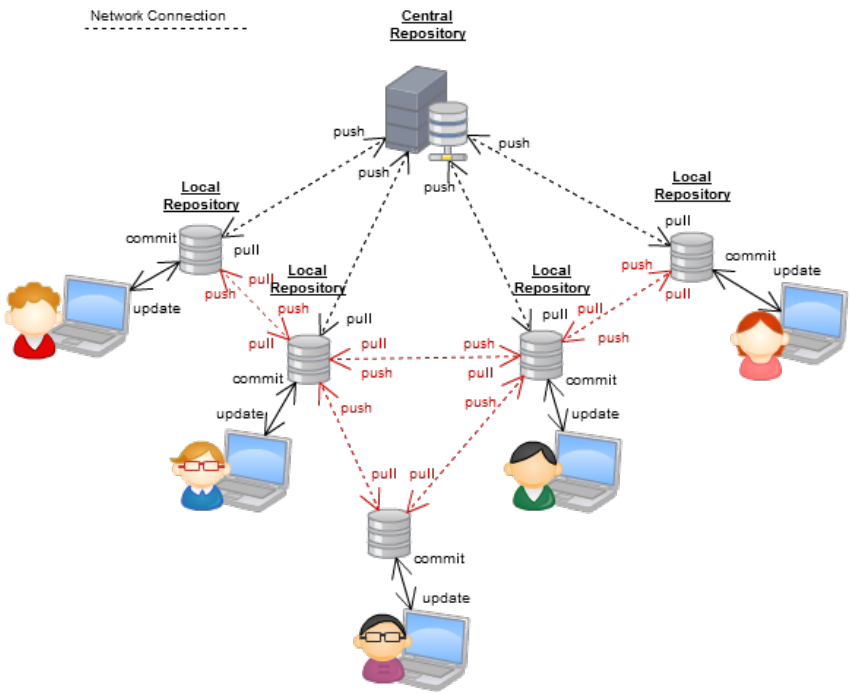


(Source)

Git

Git vs. SVN (II)

Fully distributed (Git)



(Source)

Git concepts to know

- `commit`, `update`
- `push`, `pull`
- `origin`, `remote`

Using Git

Basic commands: Repository initialization

When using Git for the first time:

```
git config --global user.email user@uah.es
git config --global user.name  "Jane Doe"
```

Initialization:

```
mkdir /path/to/your/project
cd /path/to/your/project
git init
git remote add origin https://<where>/<path>/<project.git>
git push -u origin --all # pushes up the repo and its refs for t
```

Using Git

Basic commands: Repository clonning

To work with someone else's repository, we first need to *clone* it to get a local copy.

```
git clone <repo>
```

E.g.:

```
git clone https://github.com/danrodgar/gitSlides.git
```

Note: once cloned, you can edit the repository as much as you want. No changes make their way back to the 'central' repository until you explicitly do so.

Using Git

Basic commands: tracking files

Then, we can start tracking files. To do so, we need to `add`, `commit`, and `push` the file(s) that we want to track.

```
echo "A new file..." >> Readme.md
git add Readme.md
git commit -m 'Initial commit'
git push -u origin master
```

Using Git

Basic commands: Pulling

- If you have made local changes you have to
  `git stash`
  before pulling, then
  `git stash pop`
  afterwards
- You can see which files you've modified with
  `git status`
- You can permanently remove your local changes by
  `git checkout <file>`

Using Git

Basic commands: Pushing

`git add <file>` makes git track the file <file>
Or to record all changes into a commit (notice the '.'):
`git commit .`
`git push origin master` This pushes all new commits to the repository.

Using Git

Merge and conflicts

If two people both modify the same file, the first to push *wins*. The second person will have to pull and merge before pushing.

- Changes in different parts of a file are automatically merged
- Changes in the same part of a file cause conflicts (between «< === »>) and require the user to manually resolve them. Can select either HEAD (your changes) or remote, or a mix of the two
- Two merging cases: have / haven't committed

Using Git

Merge and conflicts: `diff`

`diff -u <old file> <new file>`

This command shows what changes you would need to apply to old file to change it into new file.

Lines beginning with:

- `- - -` or `+++` tell you the old / new filenames
- `@@` points to where within the file you are looking
  (i.e. a space) are lines that are unchanged
- `-` is a deleted line
- `+` is a newly added line

Using Git

Merge and conflicts: `diff` example

```
#include <stdio.h>
int main() {
    printf("Hello World\n");
}
```

```
#include <stdio.h>
int main(int argc, char *argv[]) {
  printf("Hello World\n");
  return 0;
}
```

Applying the `diff` command:

```
$ diff -u hello.c hello_new.c > hello.patch
```

We get the following patch:

```
--- hello.c^^I2014-10-07 18:17:49.000000000 +0530
+++ hello_new.c^^I2014-10-07 18:17:54.000000000 +0530
@@ -1,5 +1,6 @@
 #include <stdio.h>
-int main() {
+int main(int argc, char *argv[]) {
  ^^Iprintf("Hello World\n");
+^^Ireturn 0;
 }
```

Using Git

Merge and conflicts: Applying `diff` changes (patch command)

After the `patch.diff` is created as:

```
diff -u <old file> <new file> > file.patch
```

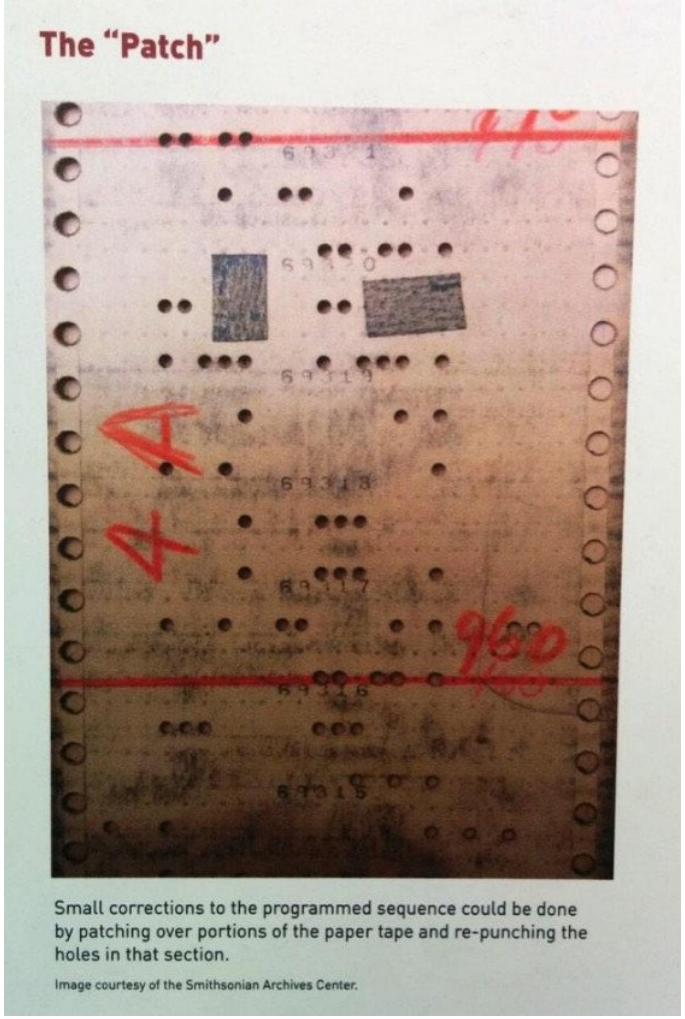
We can apply it with the `patch` command:

```
patch < file.patch
```

Note that the `file.patch` knows the name of the file to be patched.

Using Git

Merge and conflicts: Original Patch!



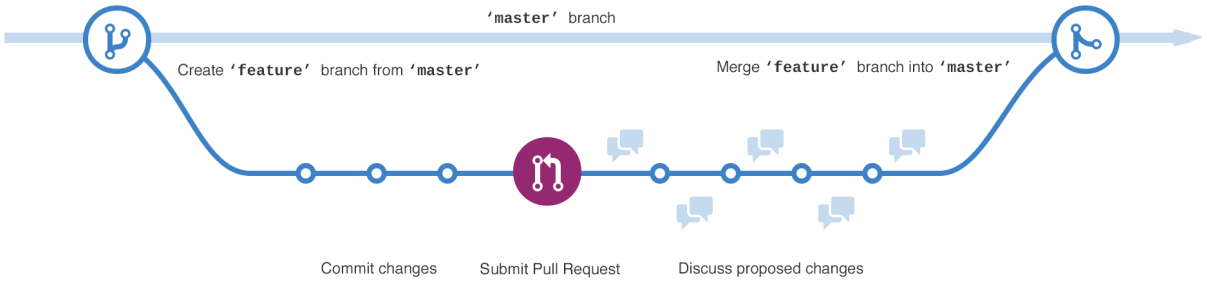
Using Git

Commits

- Merge commits record where parallel development unified
- How does Git keep track of things when parallel development happens?
- Every commit has an ID (its hash), which is a 40 character SHA-1 hash based on the commit's content. Not guaranteed to be unique; but it probably is

Using Git

Branches



Branches are used extensively (e.g. some like feature branches).

- A repository (local and remote) can have explicit branches
- The default branch is called master
  `git branch <name>` creates branches
  `git checkout <branch name>` switch branches
- To merge branch X into Y, checkout Y and run git merge X (i.e. you say "I want to merge another branch into me")

Using Git

Advanced Git: Getting an old commit

Sometimes you need to get an old file or discard some changes. With

- `git log`
- `git log -- oneline`

we can check previous commits and select one with `checkout`, e.g.:

- `git checkout c71d008`

Using Git

Advanced Git: Good practices

Tipically changes are checked by someone other than their author before being merged into master. This kind of **code review** is is naturally captured by pull requests in Git. Learn on the job: the best way to learn it is by using it. However:

- Best practice: regularly push and pull (at least daily, in general).
- Don't push half-baked changes or pull if you're in the middle of a task.

GitHub

Features

Free Git hosting provider

- Free public repositories

User interface to Git

- Repository browser

Added value Git operations

- Gist
- Pull requests
- Collaborative tools
- Issue tracking
- Web hosting
- Integrated Jekyll processor
- Markdown integration

GitHub

Repository creation

Configure the repository

- Name
- Description
- README (quite important!)
- gitignore and licence

Special file: README.md

- Contains information about the project
- Automatically visualized
- md means Markdown

Task: Create a Hello Wold

Read and follow the following instructions
`https://guides.github.com/activities/hello-world/`

Markdown

Markdown: Trivial markup

- Simple
- Very simple
- Extremely simple
- Did I say it's simple?

VERY powerful

- Several outputs
- Professional quality
- ... and simple!

Markdown example

```
# I am a header
## I am a subheader

Regular, *italic* and **bold**

- List item 1
- List item 2

[I am a link](http://foo.com)

![I am a pic](markdown.png)

~~~C
printf("Hello, world");
~~~
^^I^^I^^I
```

GitHub

GitHub Pages

Pages integrate web site in the GitHub workflow

- Creation of full web sites
- Project web site
- Documentation
- Based on Markdown (and something named *Jekyll*

GitHub locates the content to publish in three places:

- A branch named `gh-pages`
- master itself
- A folder `docs` in master
- **Page available on `https://<username>.github.io/<repository>`**

By default, Pages are disabled

- Enable them in settings

User Page. Site accesible in `https://<username>.github.io`

- The repository must be named `<username>.github.io`
- master branch