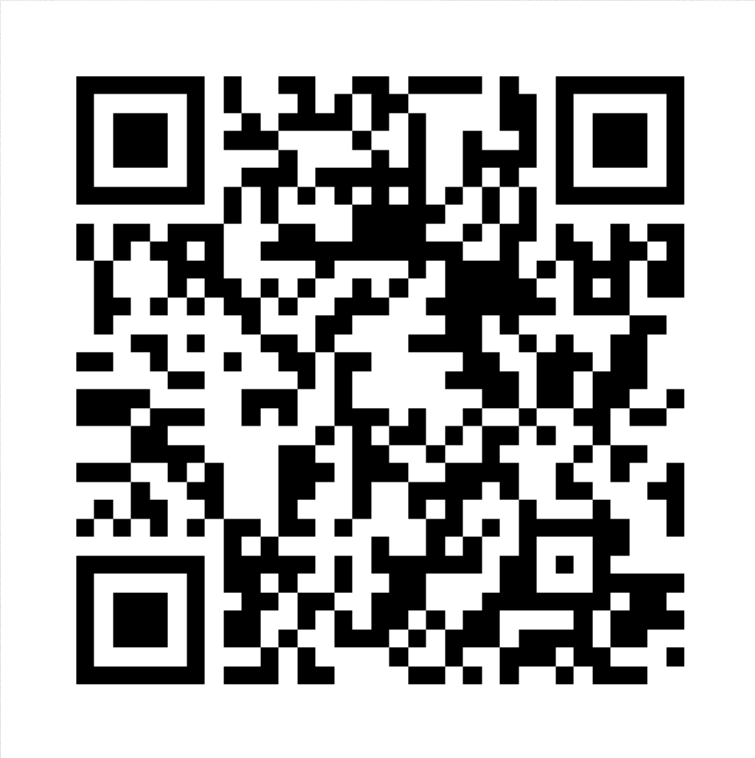
Amministrazione serena di Moodle

Roberto Pinna e Sergio Rabellino

AIUM

Cosa mi aspetto oggi da questo workshop ?



Wooclap.com

Code: **HRKFAE**

aiuM
associazione italiana utenti moodle aps

moodlemoot ITALIA 2025

GIT crash course

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

What is a version control tool

Version control systems are a category of software tools that help a software team manage changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, “developers” can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

Benefits of a version control system

- **A complete long-term change history of every file**.
  - Changes include the creation and deletion of files as well as edits to their contents. This history include the author, date and written notes on the purpose of each change. Having the complete history enables going back to previous versions to help in root cause analysis for bugs and it is crucial when needing to fix problems in older versions of software.
- **Branching and merging.**
  - Having team members work concurrently is a no-brainer, but even individuals working on their own can benefit from the ability to work on independent streams of changes. Creating a "branch" in VCS tools keeps multiple streams of work independent from each other while also providing the facility to merge that work back together, enabling developers to verify that the changes on each branch do not conflict.
- **Traceability**.
  - You are able to trace each change made to the files of a project. Having the annotated history of the code at your fingertips when you are reading the contents, trying to understand what it is doing and why it is so designed, can enable developers to make correct and harmonious changes that are in accord with the intended long-term design of the system.

What is Git

Performance Security Flexibility

By far, the most widely used modern version control system in the world today is Git. Git is a mature, actively maintained open source project originally developed in 2005 by **Linus Torvalds**, the famous creator of the Linux operating system kernel. A staggering number of software projects rely on Git for version control, including commercial projects as well as open source. Developers who have worked with Git are well represented in the pool of available software development talent and it works well on a wide range of operating systems and IDEs (Integrated Development Environments).

Having a distributed architecture, Git is an example of a DVCS (hence Distributed Version Control System). Rather than have only one single place for the full version history of the software as is common in once-popular version control systems like CVS or Subversion (also known as SVN), in Git, **every developer's working copy of the code is also a repository that can contain the full history of all changes**.

In addition to being distributed, Git has been designed with performance, security and flexibility in mind.

Let’s start with GIT

```
$ git config --global user.name "Your Name Comes Here"
$ git config --global user.email you@yourdomain.example.com
```

```
$ git clone https://github.com/moodle/moodle.git /path/to/your/htdocs
$ cd /path/to/your/htdocs
$ git checkout MOODLE_501_STABLE
$ git remote -v
```

```
$ git branch
$ git status
$ git log
$ git log -p
$ git log --stat –summary
```

Managing branches (1)

A single Git repository can maintain multiple branches of development. To create a new branch named "experimental", use

```
$ git branch experimental
```

If you now run

```
$ git branch
```

you'll get a list of all existing branches.

Now switch to the newly created branch with:

```
$ git checkout experimental
```

Managing branches (2)

Now edit some files in the experimental branch

```
$ git commit -a
$ git checkout master
```

Check that the change you made is no longer visible, since it was made on the experimental branch and you're back on the master branch.

You can make a different change on the master branch, so edit some other file or the same but in a different way:

```
$ git commit -a
```

at this point the two branches have diverged, with different changes made in each. To merge the changes made in experimental into master, run

```
$ git merge experimental
```

Managing plugins

Now add a mod plugin found on the moodle plugin repository

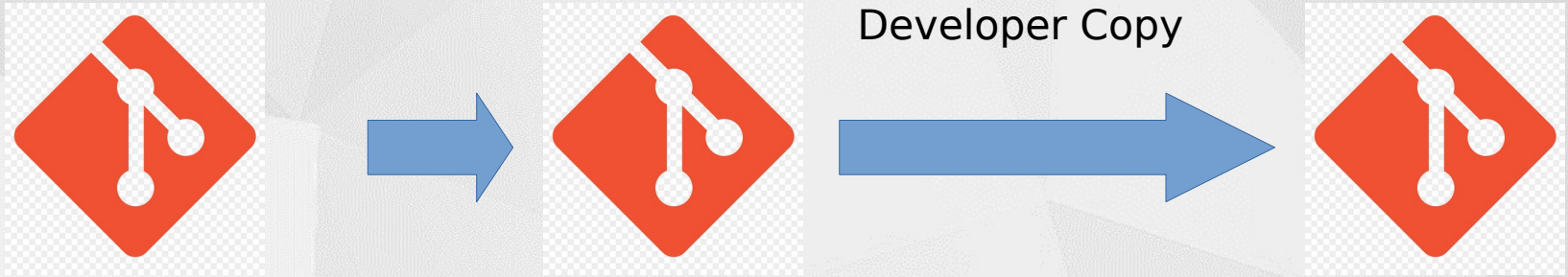
```
$ git status  → better if all is clean !!
$ cd /xxx/www/public/mod
$ git clone plugin-url pluginname
$ cd pluginname
$ git checkout xxxxx  ...the tags you think it's right ...
$ cd /xxx/www; vi .git/info/exclude  → add the path /xxx/www/public/mod/pluginname

$ now install it into your moodle either via cli or gui.
```

Obviously, repeat this for each plugin you want to add.
Moodle git is now separated from plugin gits folders...

Next step ? Write a simple script (bash/python/php...) to manage them if they are many...

Running Example...

A way to manage the complexity and continous upgrade/fast bugfixing



Developer Copy

https://github.com/moodle/moodle.git

https://your.gitlab.it/moodle/moodle.git

Production Copy

Customizations
Temporary Fixes
Develop

E quindi ?