Università degli Studi di Cagliari
Corso di Laurea DSBAI

Web Analytics e Analisi Testuale

http://agilegroup.eu

A.A. 2017/2018

Ing. Marco Ortu
Via Porcell 4, primo piano
mail: marco.ortu@diee.unica.it

github

SOCIAL CODE HOSTING

git-scm.com


git
the fast version control system
Home
About Git
Documentation
Download
Tools & Hosting
Wiki
The Git User's Survey 2010 is up! Please devote a few minutes of your time to fill it out, so we can improve Git!
Git is...
Git is a free & open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
Every Git clone is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server.
Branching and merging are fast and easy to do.
Git is used for version control of files, much like tools such as Mercurial, Bazaar, Subversion, CVS, Perforce, and Visual SourceSafe.
Projects using Git
Git
Linux Kernel
Perl
Gnome
Qt
Ruby on Rails
Android
PostgreSQL
Wine
Fedora
Debian
X.org
Download Git
The latest stable Git release is
v1.7.2.3
release notes (2010-09-03)
Windows
Mac OSX
Source
Other Download Options
Git Source Repository
Git Quick Start
Cloning and Creating a Patch
$ git clone git://github.com/git/hello-world.git
Creating and Commiting
$ cd (project-directory)

Git Reference

Reference About § Site Source

Getting and Creating Projects

- init
- clone

Basic Snapshotting

- add
- status
- diff
- commit
- reset
- rm, mv

Branching and Merging

- branch
- checkout
- merge
- log
- tag

Sharing and Updating Projects

- fetch, pull

INTRODUCTION TO THE GIT REFERENCE

This is the Git reference site. This is meant to be a quick reference for lea
Git commands. The commands are organized into sections of the type of
options and commands needed to accomplish these common tasks.

Each section will link to the next section, so it can be used as a tutorial. E
as the offical manual pages and relevant sections in the **Pro Git book**, so
with thinking about source code management like Git does.

HOW TO THINK LIKE GIT

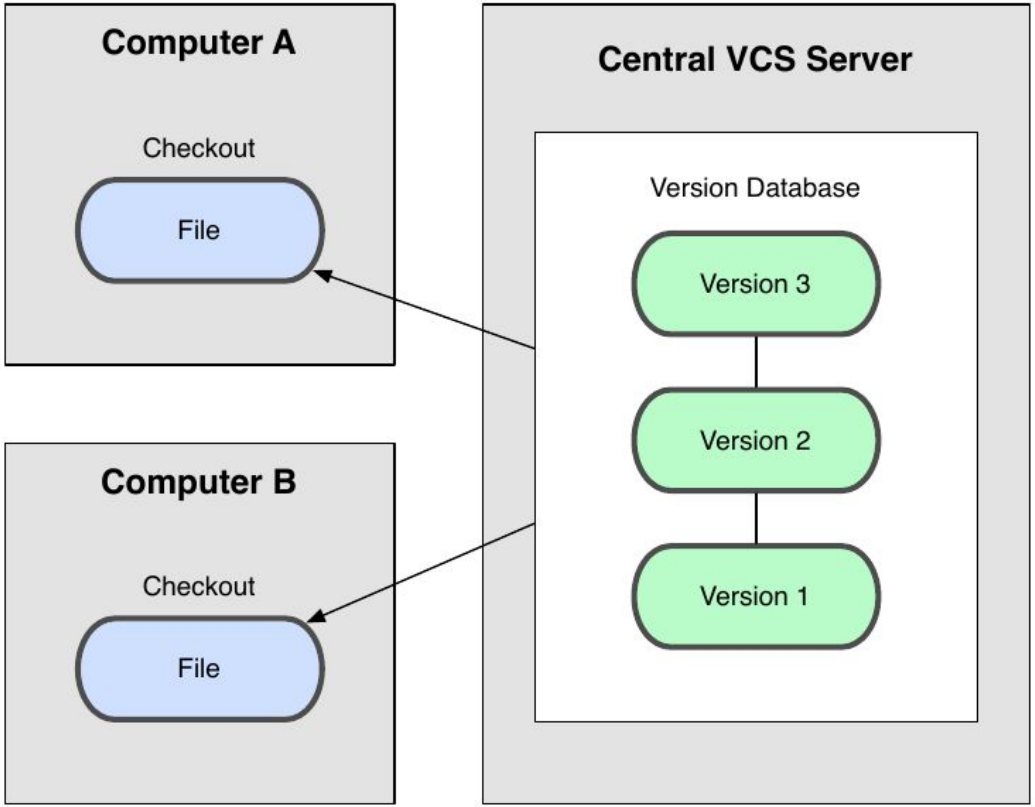
This first thing that is important to understand about Git is that it thinks abo
whatever SCM you may be used to. It is often easier to learn Git by trying
try to think about it in the Git way.

Let's start from scratch. Assume you are designing a new source code ma
you used a tool for it? Chances are that you simply copied your project di

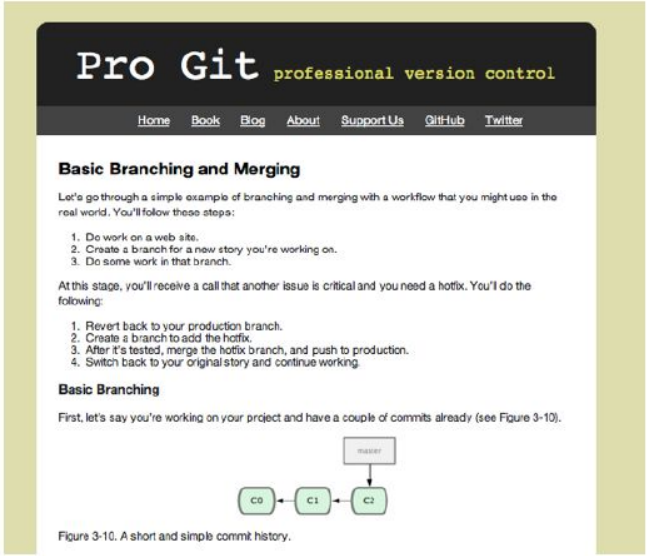
```
$ cp -R project project.bak
```

http://gitref.org

Piattaforma distribuita



Importanza dei branch



Pro Git professional version control

Home Book Blog About Support Us GitHub Twitter

Basic Branching and Merging

Let's go through a simple example of branching and merging with a workflow that you might use in the real world. You'll follow these steps:

1. Do work on a web site.
2. Create a branch for a new story you're working on.
3. Do some work in that branch.

At this stage, you'll receive a call that another issue is critical and you need a hotfix. You'll do the following:

1. Revert back to your production branch.
2. Create a branch to add the hotfix.
3. After it's tested, merge the hotfix branch, and push to production.
4. Switch back to your original story and continue working.

Basic Branching

First, let's say you're working on your project and have a couple of commits already (see Figure 3-10).

Figure 3-10. A short and simple commit history.

http://progit.org

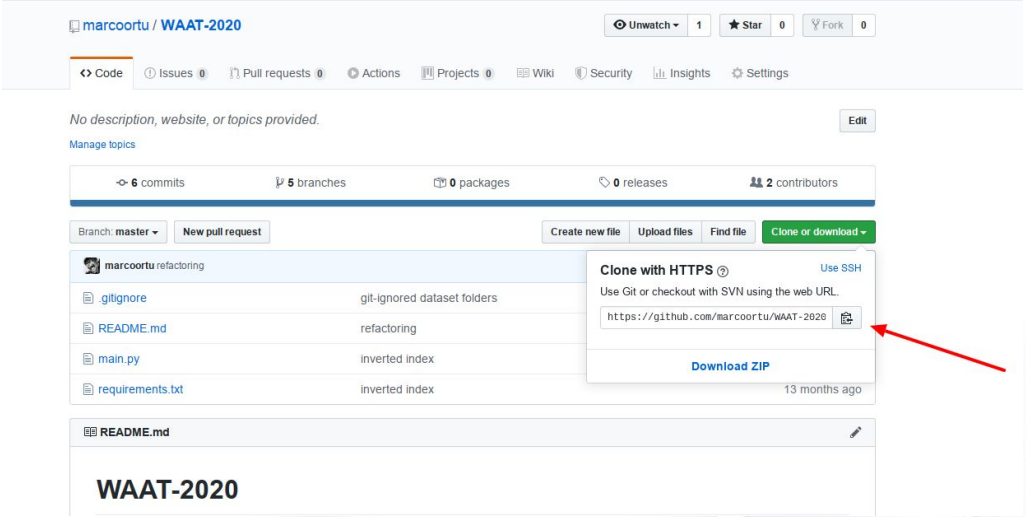
Comandi Git: configurazione

➔ Se utilizziamo per la prima volta Git dobbiamo specificare l'utenza predefinita per effettuare le operazioni

```
$ git config --global user.name “Scott Chacon”

$ git config --global user.email “schacon@gmail.com”
```

Comandi Git: clonare un progetto da un repository remoto



➔ Copiamo la URL del progetto da clonare ed eseguiamo:

➔ ***git clone https://github.com/marcoortu/WAAT-2020.git***

Comandi Git: commit locali

- ➔ Partiamo da una cartella contenente un progetto versionato con Git, per esempio WAAT-2020
- ➔ Immaginiamo di avere solo due file:

README.txt

```
This is Scott's
Hello project.

Licensed under GPL.
```

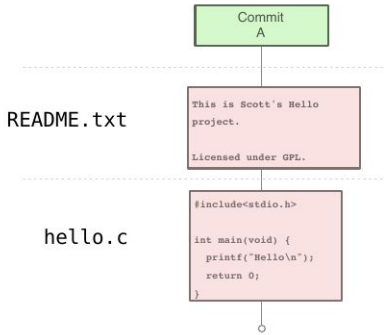
hello.c

```
#include<stdio.h>

int main(void) {
  printf("Hello\n");
  return 0;
}
```

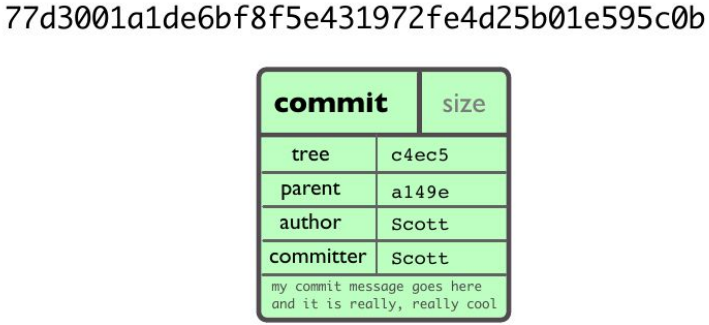
Comandi Git: commit locali

- ➔ Questi file non sono sotto versionamento e per poterli rendere tracciabili da Git (tracked) bisogna eseguire un commit.
- ➔ **git add .**
- ➔ **git commit -m “primo commit”**
- ➔ Il primo comando prepara i file che dovranno essere *committati,* il punto “.” indica “all” cioè tutti i file nuovi o che hanno subito una modifica



Comandi Git: commit locali

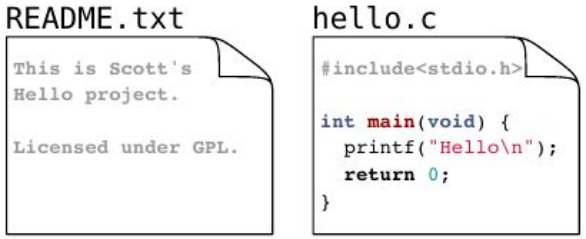
Un commit rappresenta le modifiche effettuate e nel repository locale viene salvato un oggetto commit identificato da un codice hash univoco.



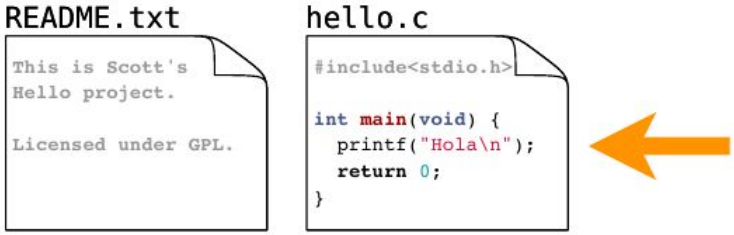
Comandi Git: commit locali

➔ Effettuiamo una modifica ad un file tracciato (tracked) da Git, per esempio hello.c

Prima



Dopo



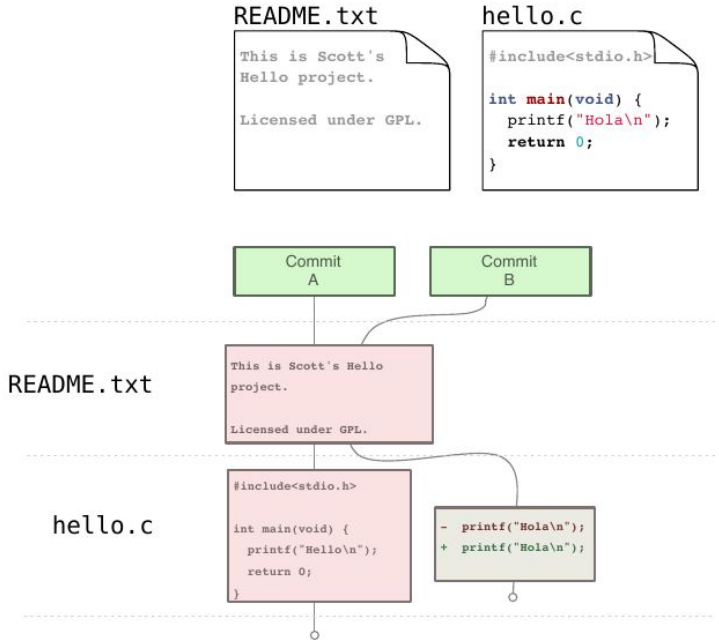
Comandi Git: commit locali

Eseguiamo i seguenti comandi:

➔ **git add hello.c**

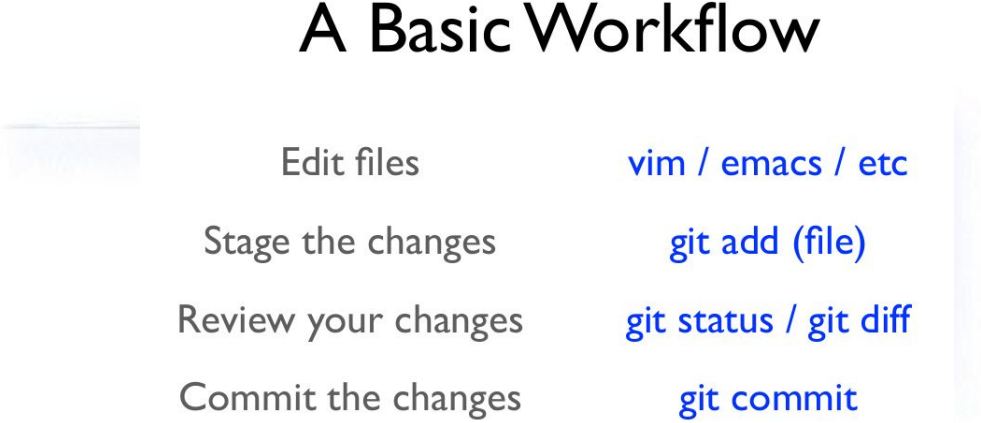
➔ **git commit -m “modifica printf”**

Abbiamo ora due commit, Git registra solo le differenze tra il file modificato e la versione precedente, in questo caso il commit B registrerà solo la modifica della riga ***printf(“Hola\n”);***



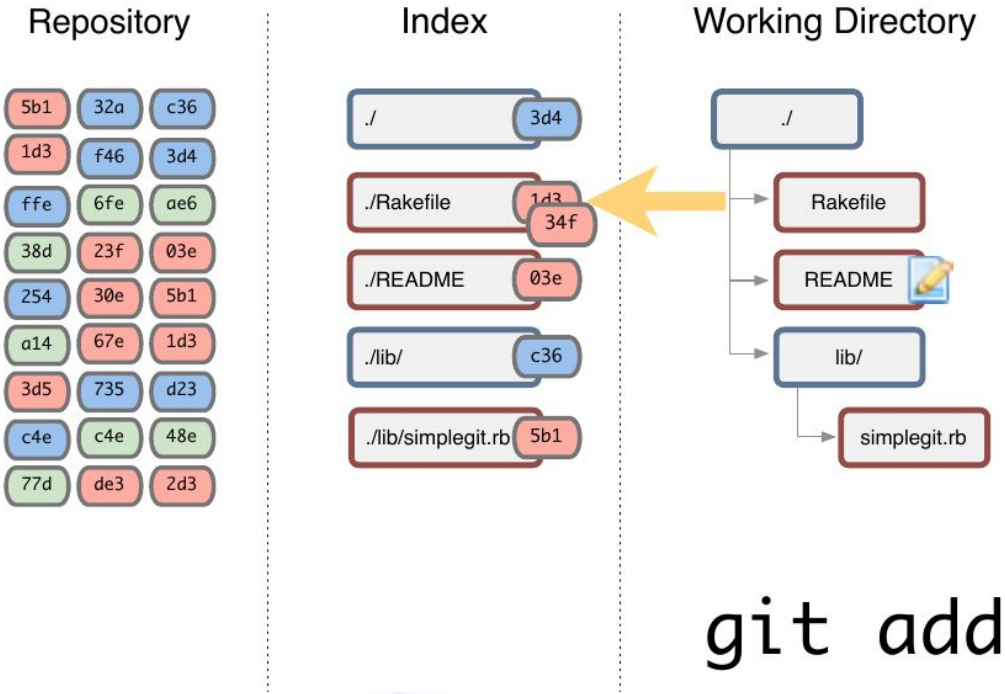
Comandi Git: commit locali

Riassumendo il workflow base per salvare le proprie modifiche locali:



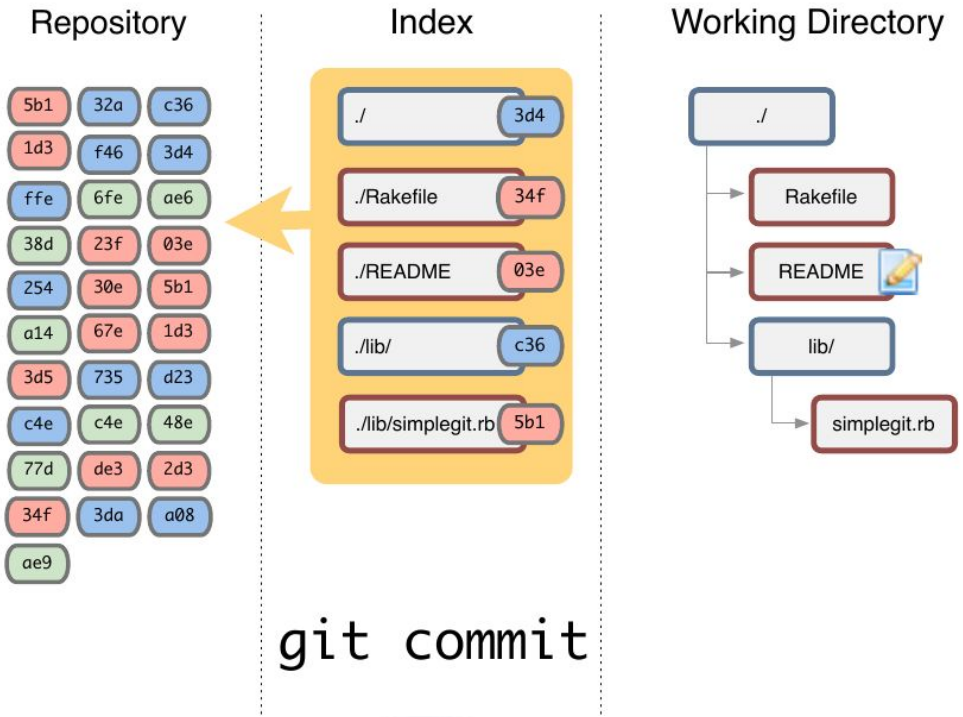
Comandi Git: commit locali

Il comando **git add** aggiunge le modifiche fatte ai file nella working directory all'indice dei file da salvare nel repository Git locale.



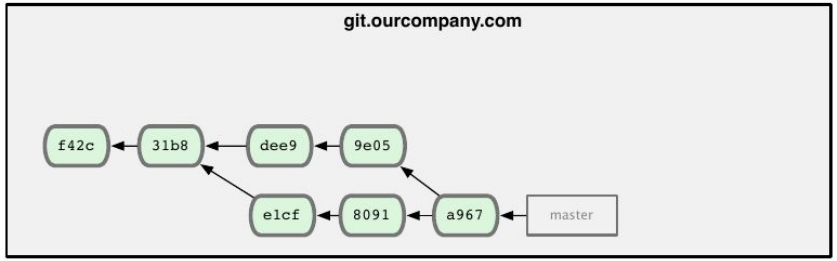
Comandi Git: commit locali

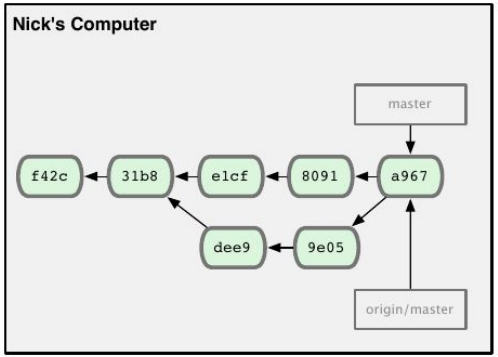
Il comando **git commit** aggiunge le modifiche contenute nell'indice delle modifiche al repository locale rendendole permanenti

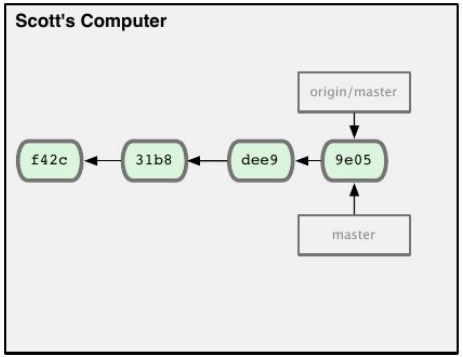


Comandi Git: branches

I ***branch*** non sono nient'altro che dei puntatori a dei particolari commit.
Immaginiamo di avere un repository remoto e due persone fanno il checkout del branch ***master.***
I due repository locali puntano tutti al commit ***a967*** e sono allineati con il repository remoto identificato da ***origin/master.***







Comandi Git: branches

Quando vengono modificati dei file in locale, queste modifiche vanno committate in locale prima di poter aggiornare il repository locale con quello remoto o per poter fare il checkout di un nuovo branch.