



Finanziato
dall'Unione europea
NextGenerationEU



*Ministero dell'Istruzione
e del Merito*



Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA



FINTECH DEVELOPER

Fondamenti di version control

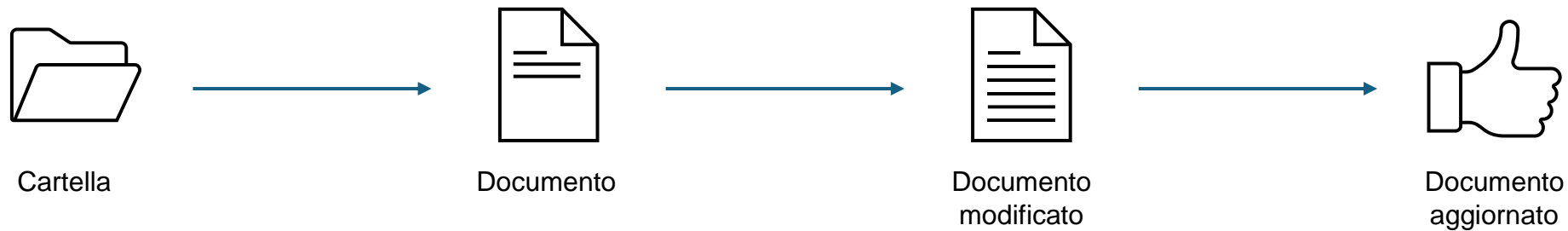
Docente: Loredana Frontino

Titolo argomento: Ripasso finale

Parte 1

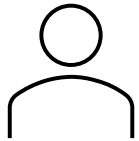
Fondamenti di Version Control – Introduzione al version control

AGGIORNAMENTO DI UN FILE IN UNA CARTELLA SUL PC

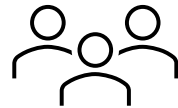


Fondamenti di Version Control – Introduzione al version control

Sviluppo collaborativo



Sviluppo per me



Sviluppo collaborativo

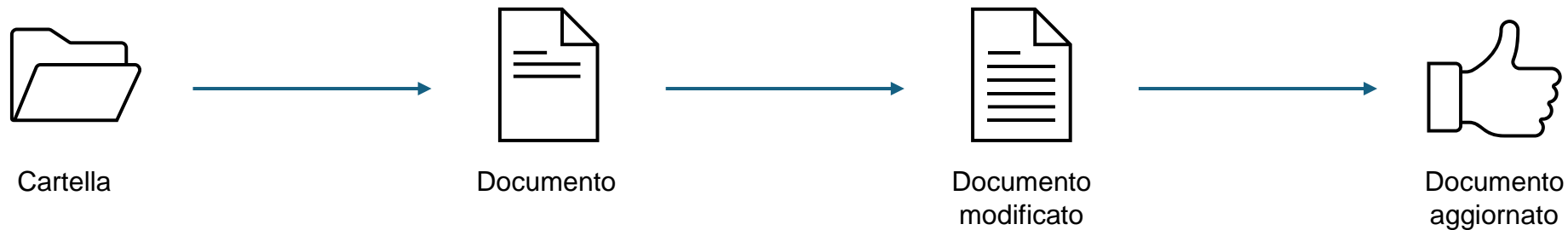
Collaborando

In contemporanea

Si potrebbe lavorare anche sullo stesso file!!!

Fondamenti di Version Control – Introduzione al version control

AGGIORNAMENTO DI UN FILE IN UNA CARTELLA SUL PC



- Dov'è lo storico cambiamenti?
- Come riportare un file allo stato precedente?
- Se voglio condividere il mio progetto con altri come gestisco le modifiche?
- Chi ha modificato quel punto?

Cos'è il version control?

Fondamenti di Version Control – Introduzione al version control

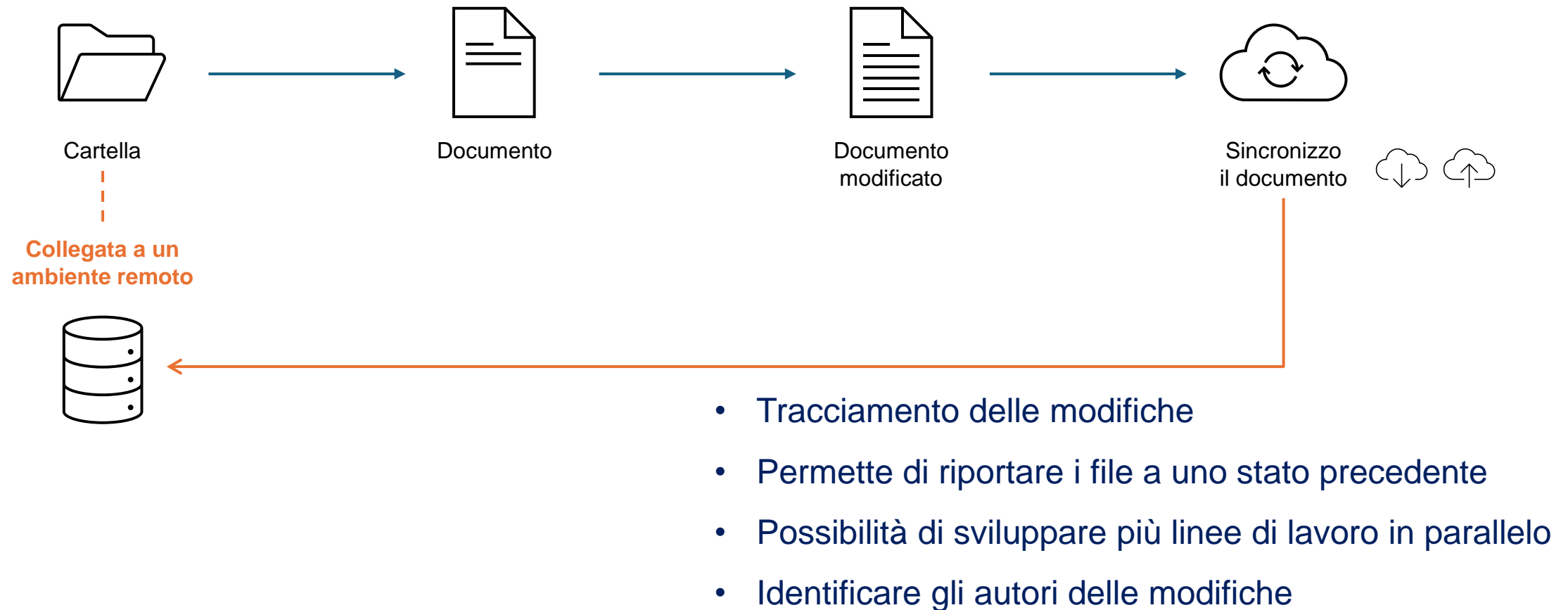
Definizione

Il **Version Control** o versionamento consente di **tracciare i cambiamenti** di un file o a un insieme di file. Permette, tra le altre cose, di **riportare i file** o l'intero progetto **a uno stadio precedente**, **visualizzare le modifiche nel corso del tempo**, sviluppare **più linee di lavoro in parallelo** e **identificare gli autori** delle modifiche.

docs.italia.it

Fondamenti di Version Control – Introduzione al version control

AGGIORNAMENTO DI UN FILE CON IL VERSION CONTROL



Fondamenti di Version Control – Introduzione al version control

Sviluppo collaborativo

Nel momento in cui il codice che scrivo non è più solo “mio”, bisogna:

Fondamenti di Version Control – Introduzione al version control

Sviluppo collaborativo

Nel momento in cui il codice che scrivo non è più solo “mio”, bisogna:

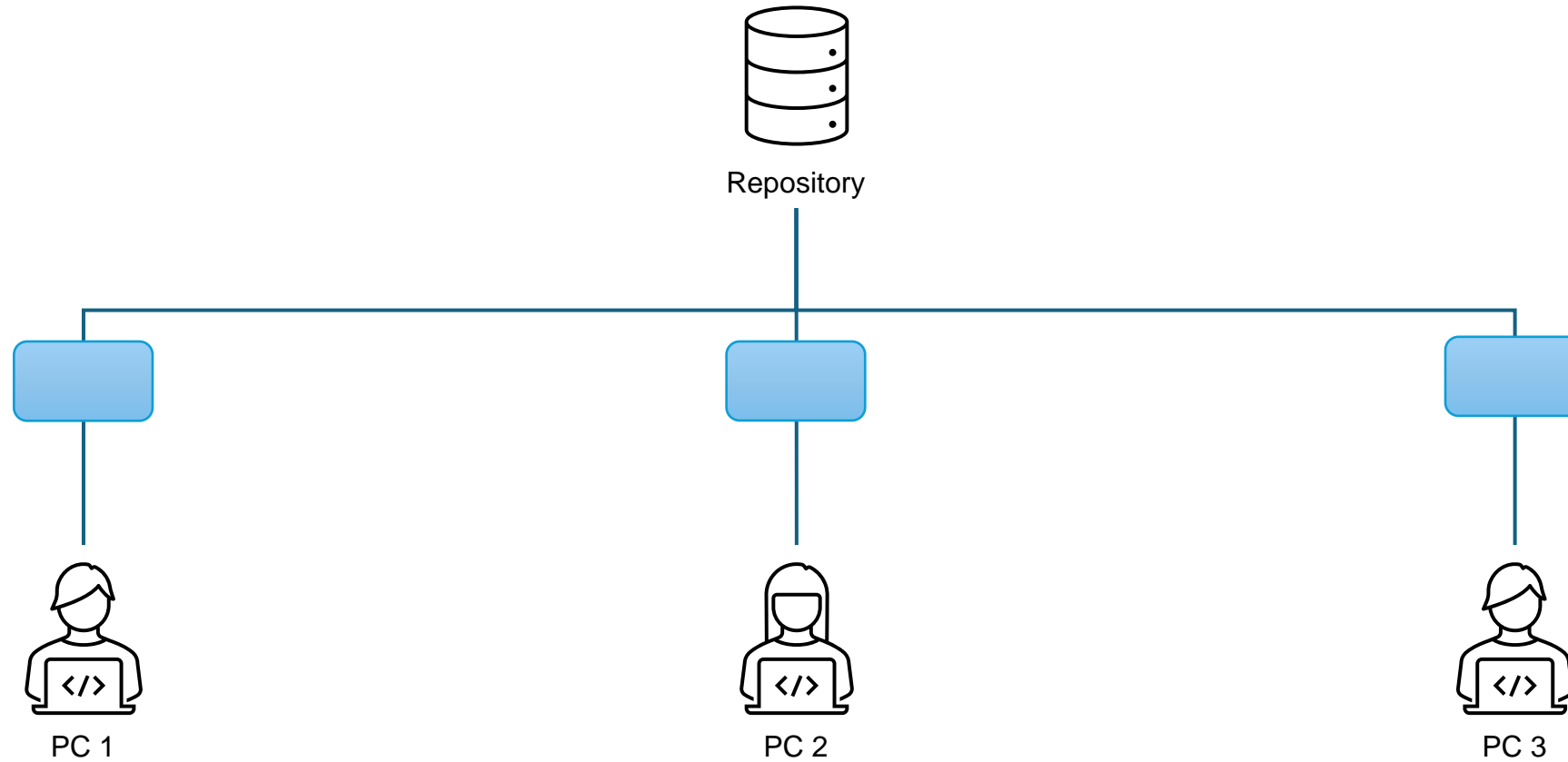
- Stabilire delle regole per la condivisione di questo codice
- Utilizzare degli strumenti che permettano di applicare queste regole
- Questi strumenti sono detti **Version Control Systems**

Fondamenti di Version Control – Introduzione al version control

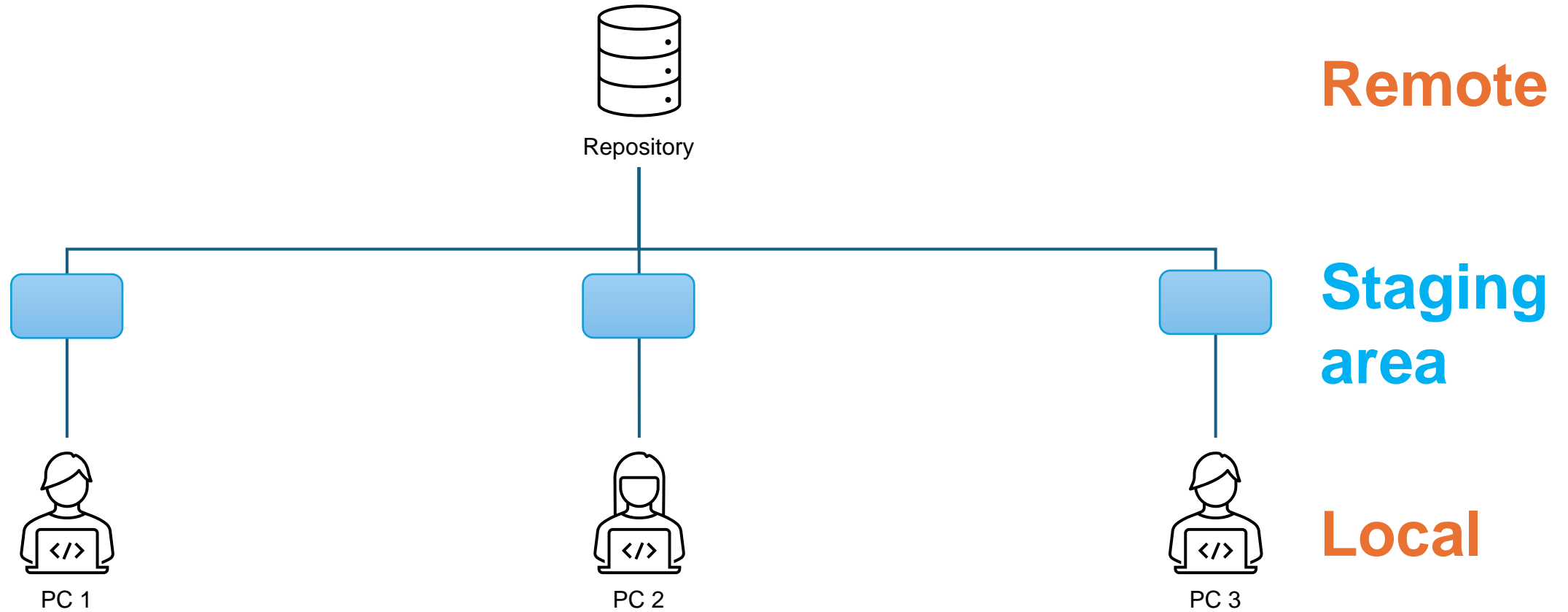
Version Control System (VCS)

- Fornisce supporto alla memorizzazione dei codici sorgenti
- Fornisce uno storico di ciò che è stato fatto
- Può fornire un modo per lavorare in parallelo su diversi aspetti dell'applicazione
- Può fornire un modo per lavorare in parallelo senza intralciarsi a vicenda

Fondamenti di Version Control – Ripasso laboratorio



Fondamenti di Version Control – Ripasso laboratorio

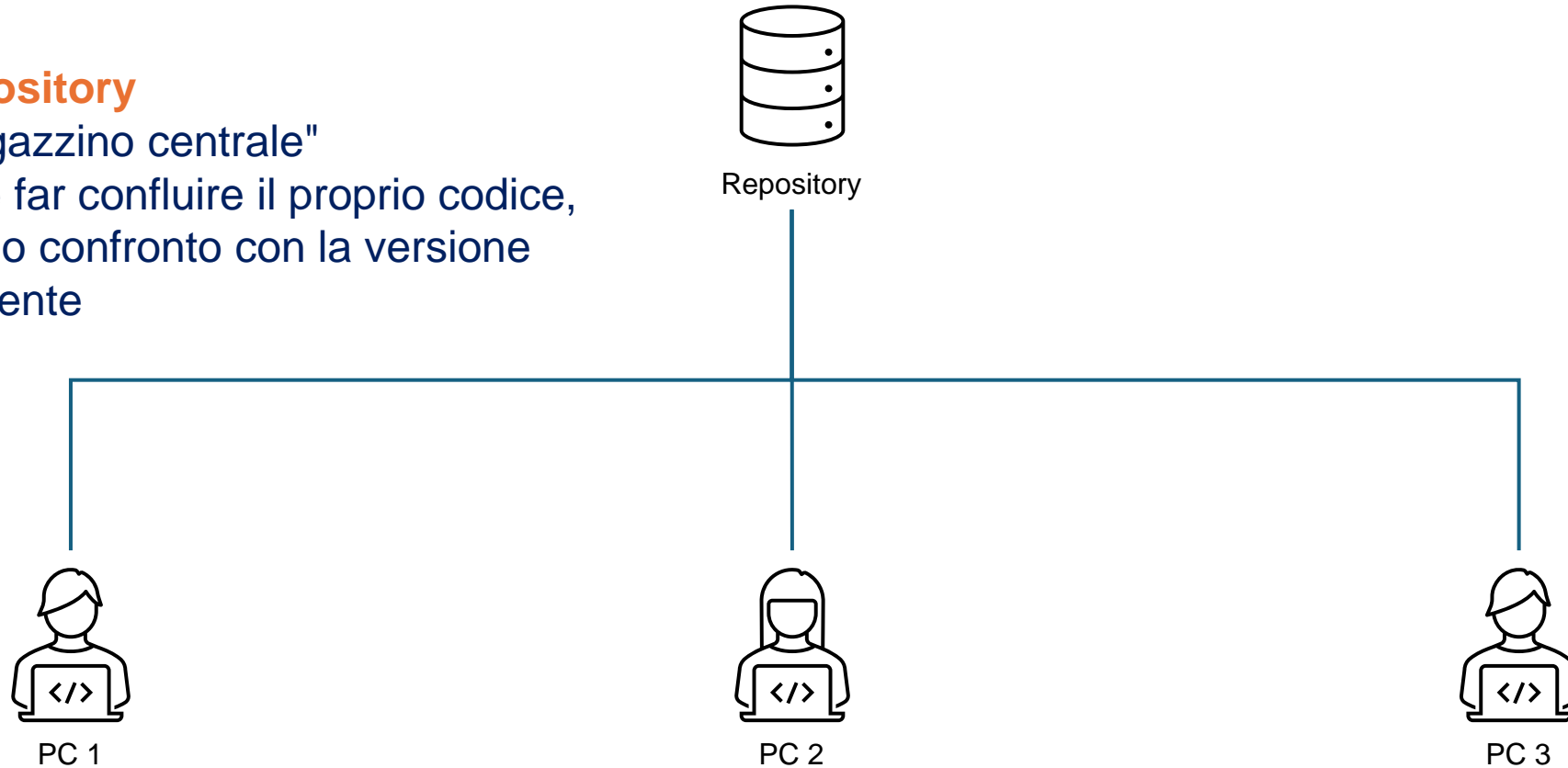


Fondamenti di Version Control – Introduzione al version control

Version control systems

Repository

"magazzino centrale"
dove far confluire il proprio codice,
previo confronto con la versione
esistente



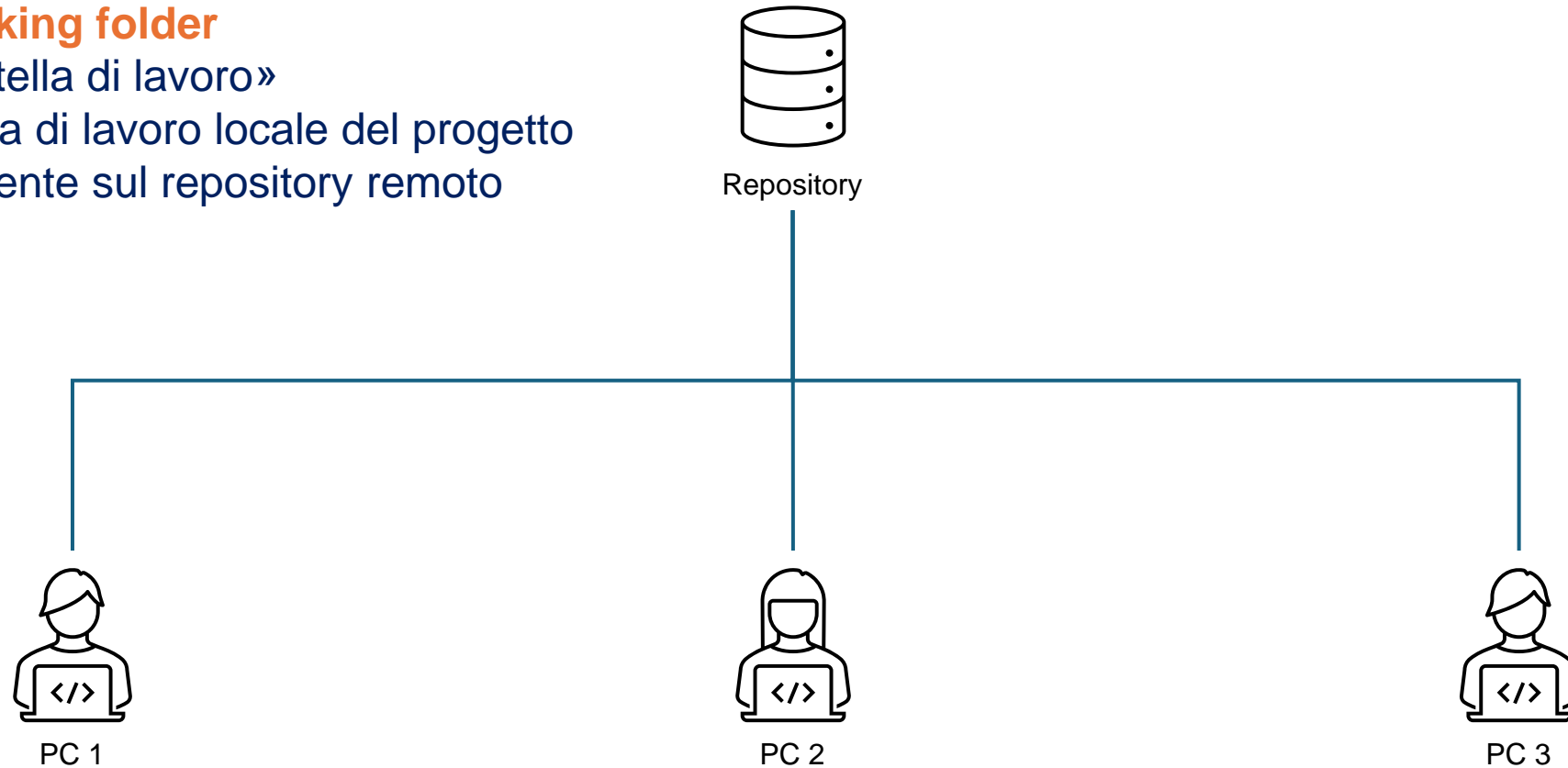
Fondamenti di Version Control – Introduzione al version control

Version control systems

Working folder

«cartella di lavoro»

Copia di lavoro locale del progetto
presente sul repository remoto



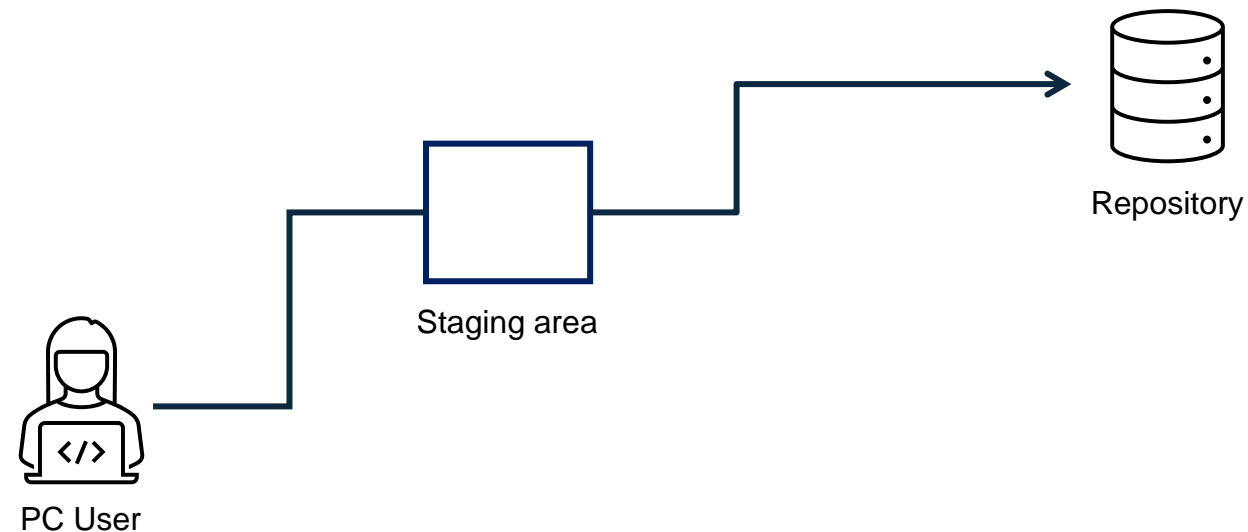
Stato di avanzamento di un file

Cos'è la staging area in git?

Fondamenti di Version Control – Ripasso concetti git

Cos'è l'area di staging

La **staging area in Git** è il luogo “virtuale” a cui aggiungere le modifiche presenti nella working copy che si intende salvare come commit.



Fondamenti di Version Control – Introduzione al version control

Come funziona

1. Ho appena finito una modifica al progetto e ho il mio codice pronto



Cosa faccio?



Fondamenti di Version Control – Introduzione al version control

Come funziona

1. Ho appena finito una modifica al progetto e ho il mio codice pronto
- ~~2. Non lo lancio a caso nel mucchio di codice già esistente (il repository)~~
3. Confronto il mio codice con la versione presente sul repository, verifico se nel frattempo qualcun altro ha "inviato" modifiche sui miei stessi sorgenti, se queste sono compatibili con le mie ecc...

sincronizzazione con il repository sorgente



Fondamenti di Version Control – Introduzione al version control

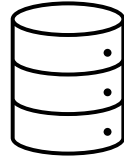
Come funziona - Esempio

1. 3 di voi lavorano su un progetto
2. Avete apportato modifiche tutti alla homepage del progetto

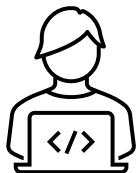


Fondamenti di Version Control – Introduzione al version control

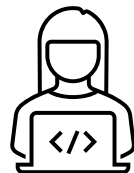
Come funziona - Esempio



Versione remota



PC 1



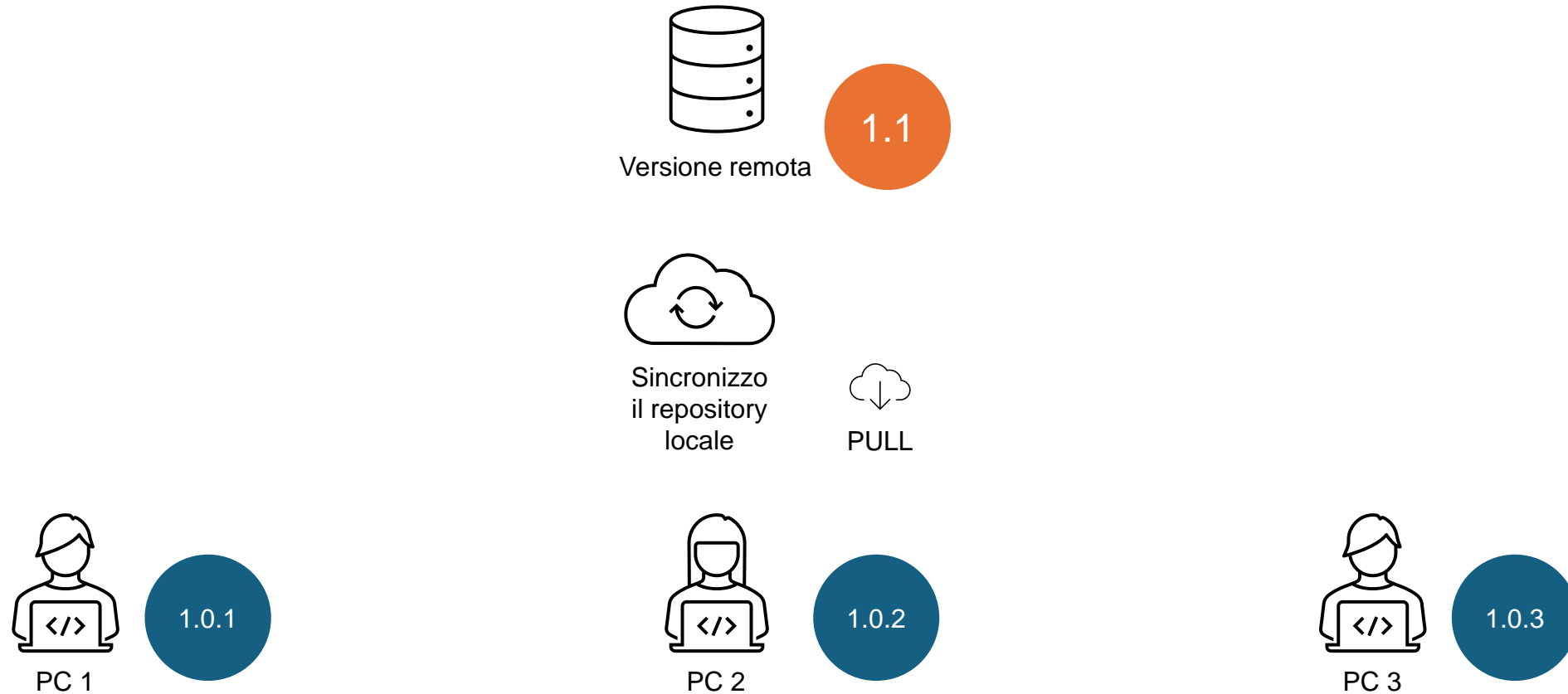
PC 2



PC 3

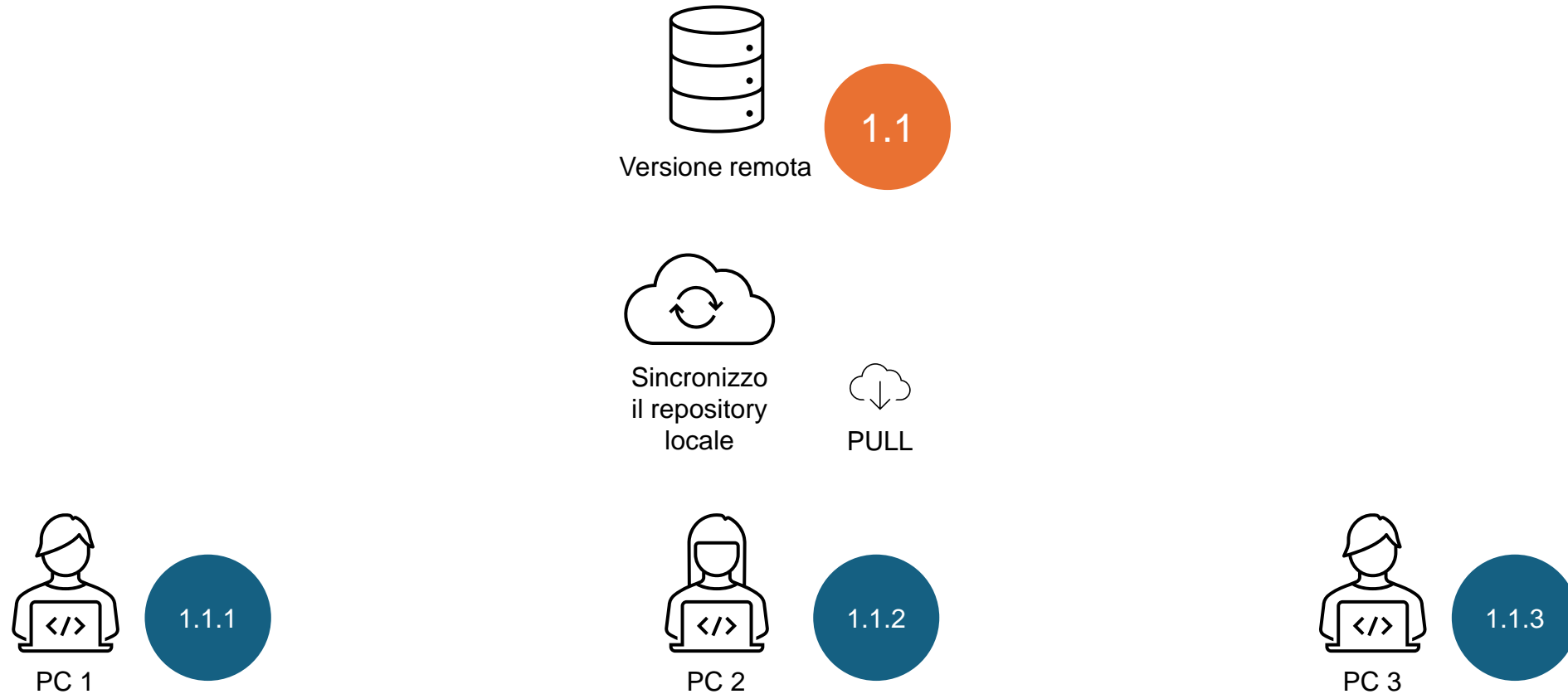
Fondamenti di Version Control – Introduzione al version control

Come funziona - Esempio



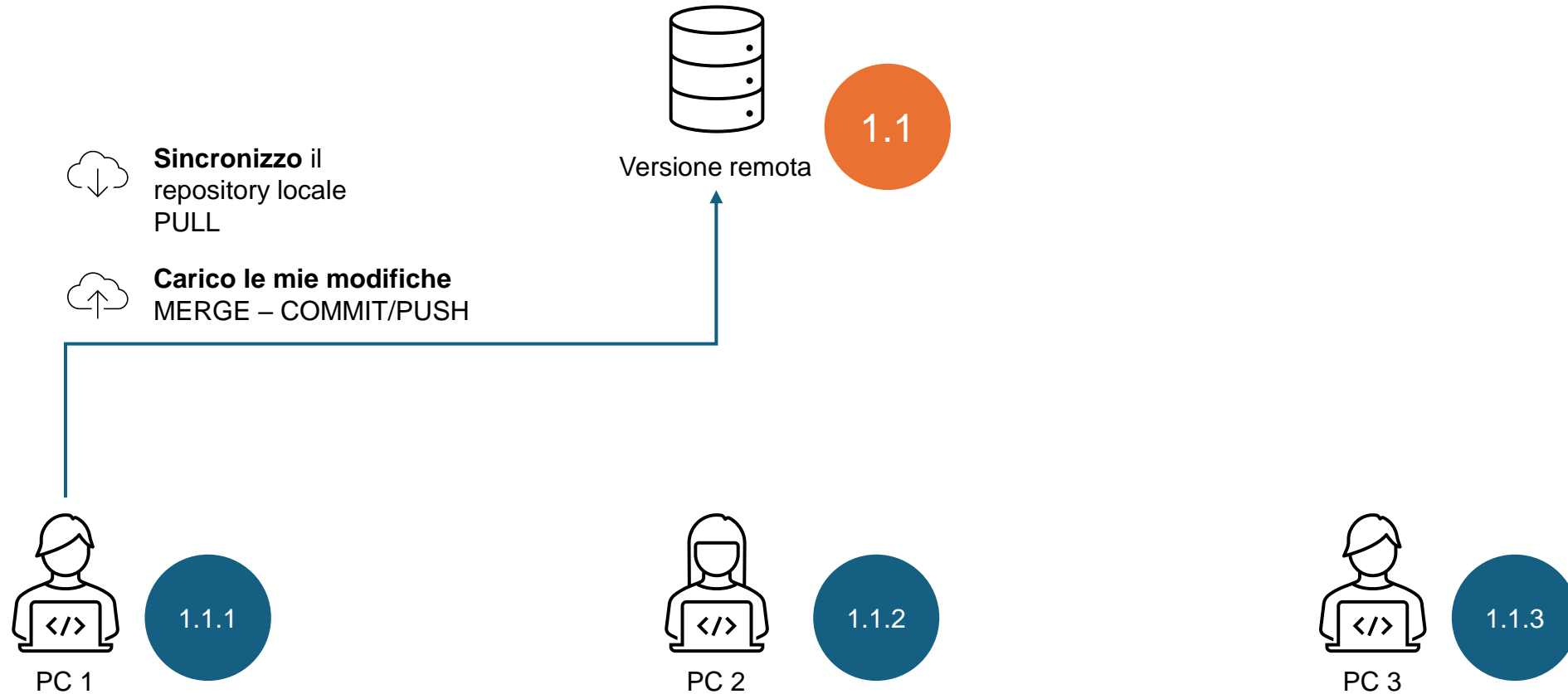
Fondamenti di Version Control – Introduzione al version control

Come funziona - Esempio



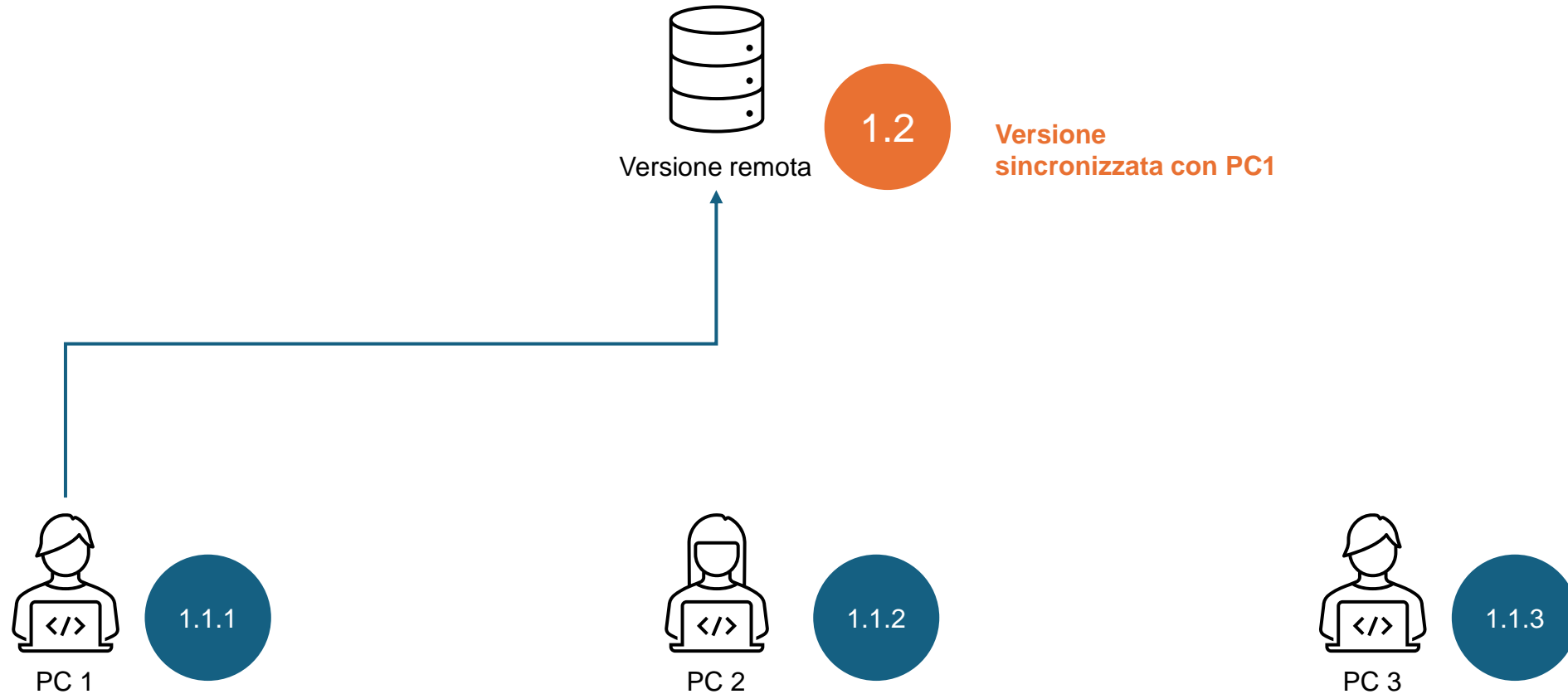
Fondamenti di Version Control – Introduzione al version control

Come funziona - Esempio



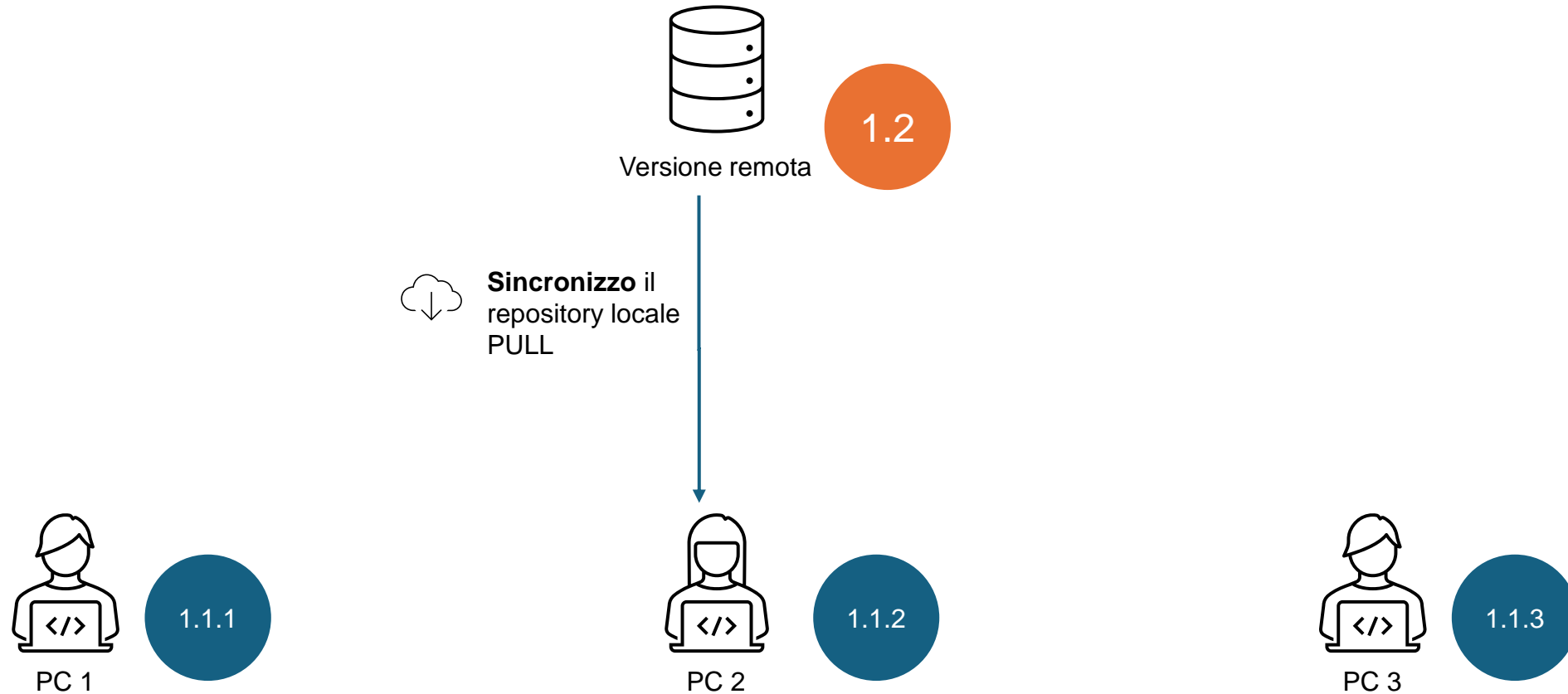
Fondamenti di Version Control – Introduzione al version control

Come funziona - Esempio



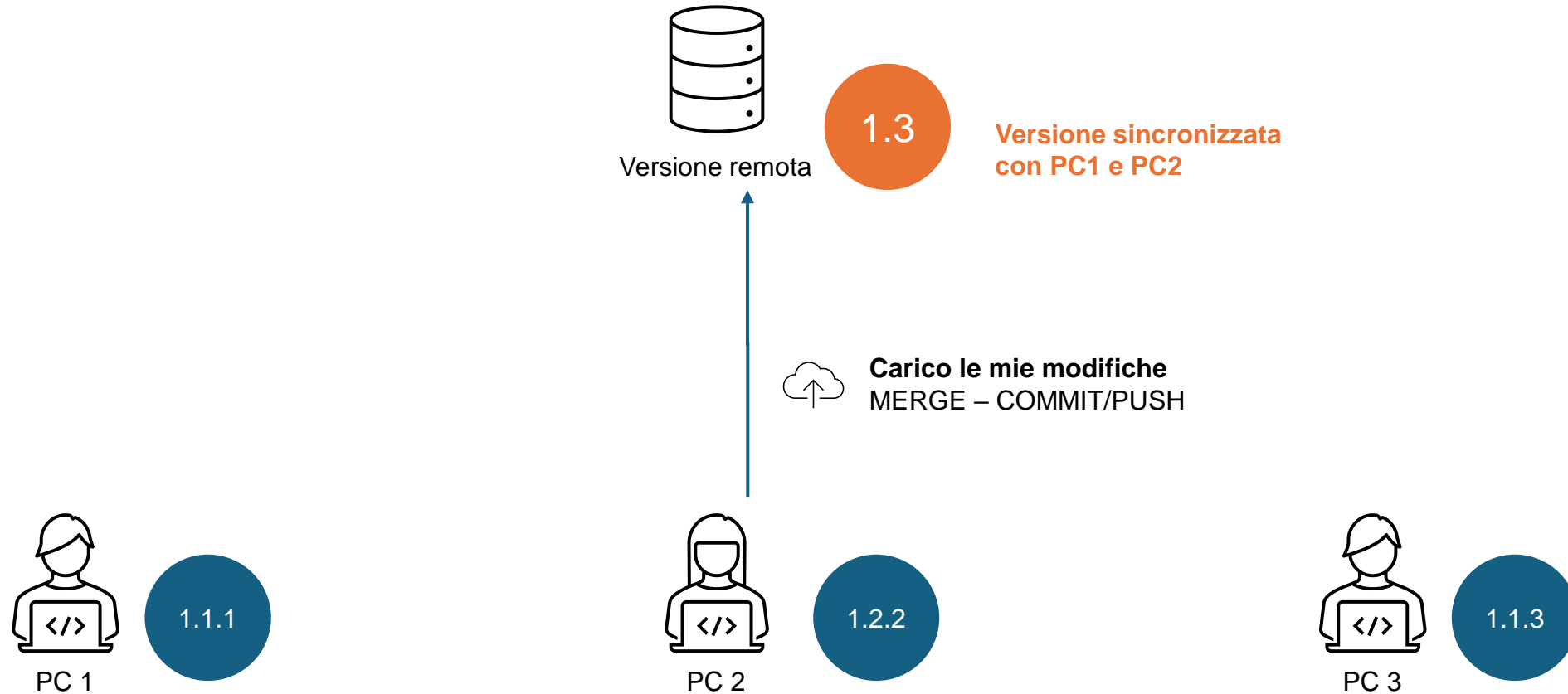
Fondamenti di Version Control – Introduzione al version control

Come funziona - Esempio



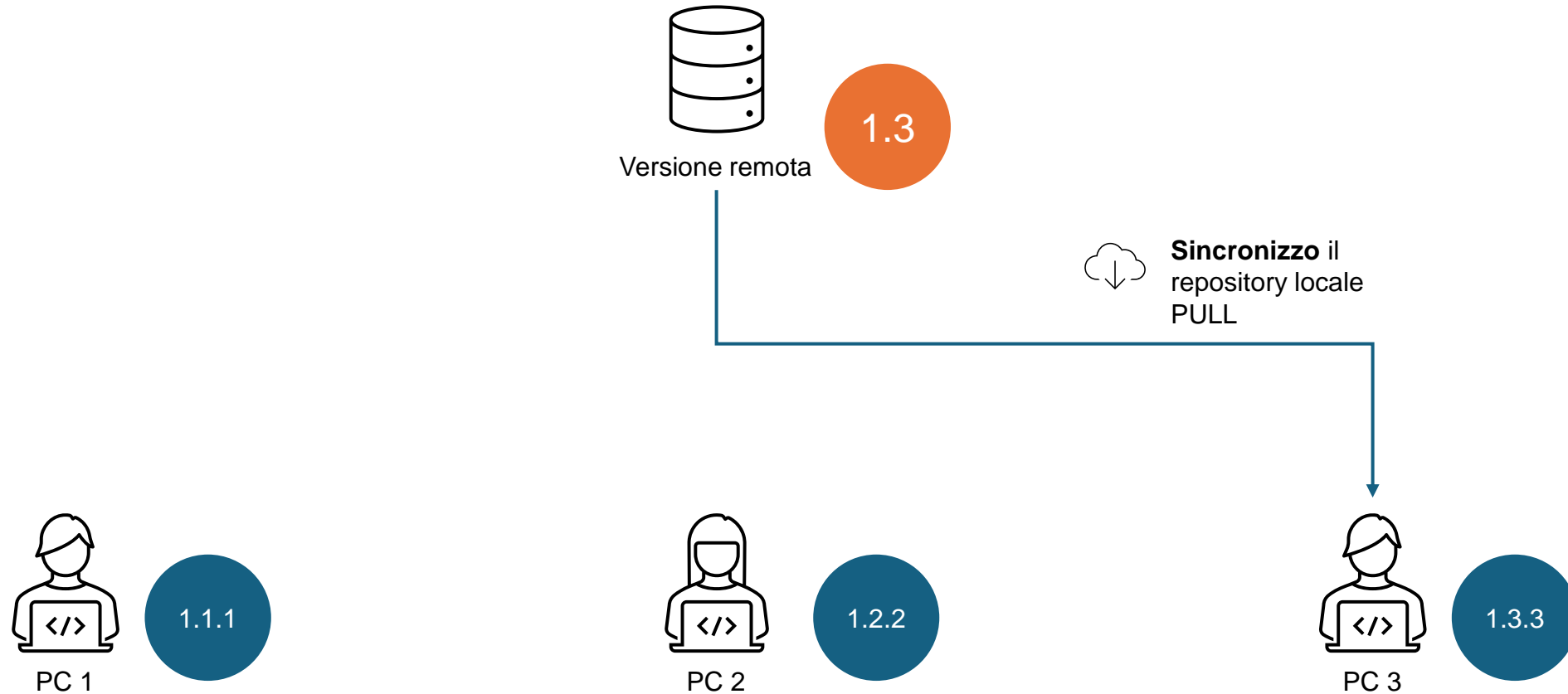
Fondamenti di Version Control – Introduzione al version control

Come funziona - Esempio



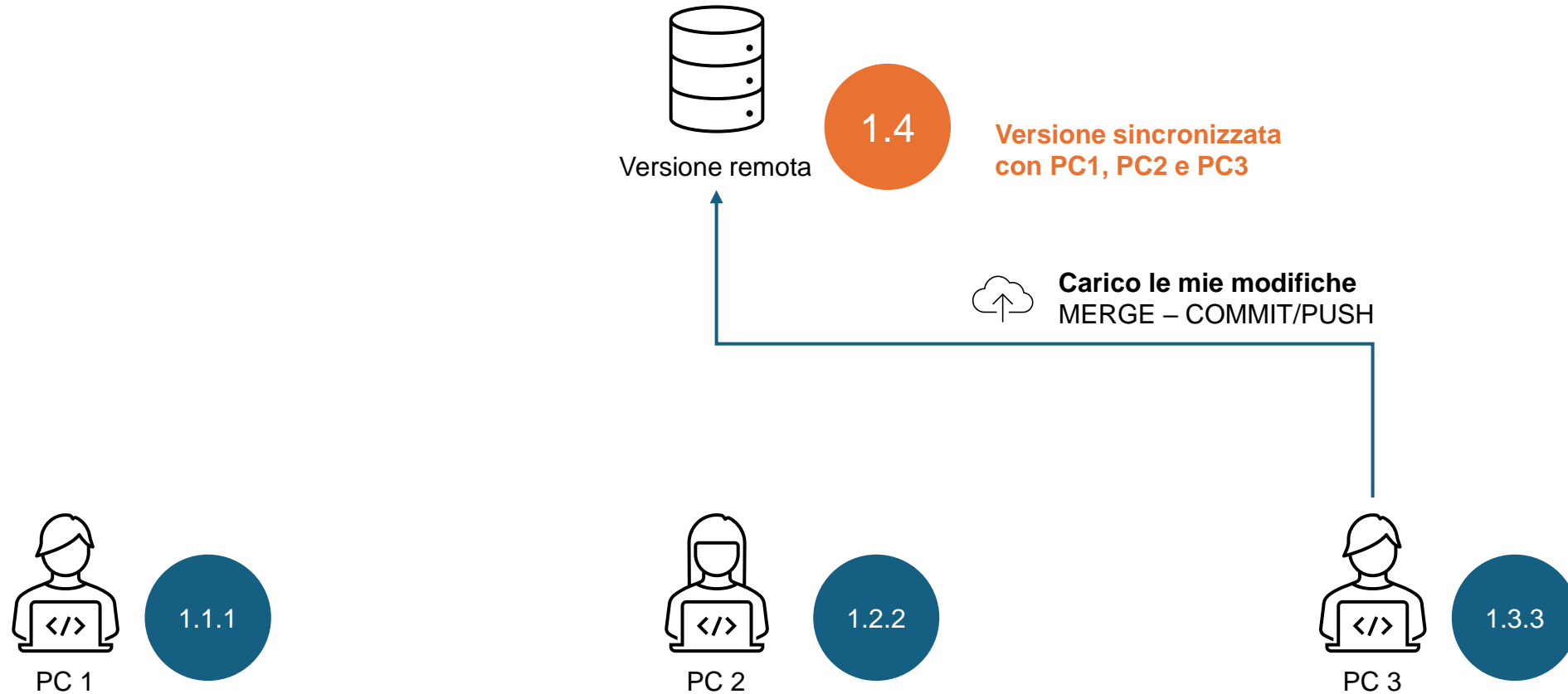
Fondamenti di Version Control – Introduzione al version control

Come funziona - Esempio



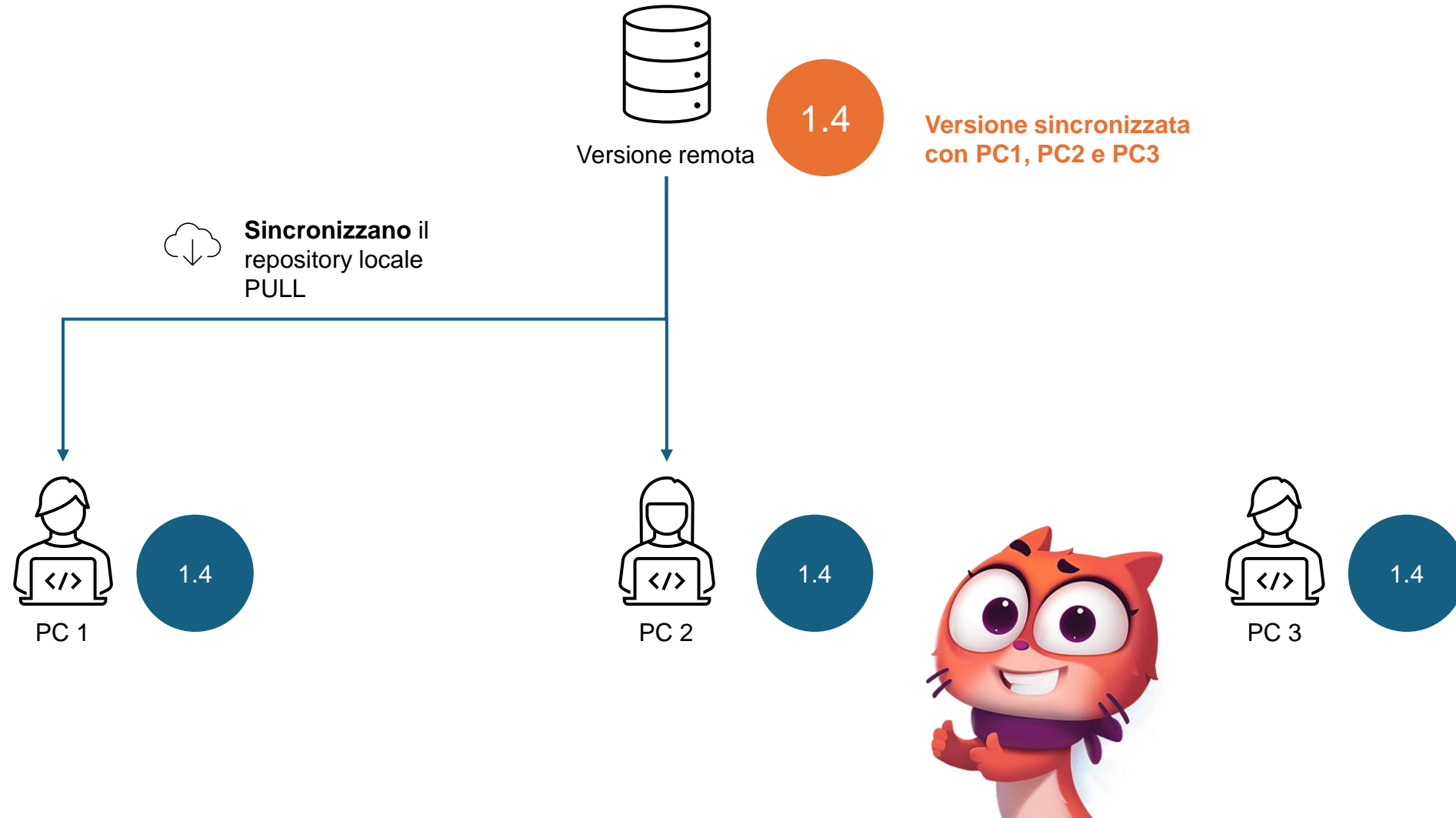
Fondamenti di Version Control – Introduzione al version control

Come funziona - Esempio



Fondamenti di Version Control – Introduzione al version control

Come funziona - Esempio



Fondamenti di Version Control – Introduzione al version control

Come funziona - Esempio

1. 3 di voi lavorano su un progetto
2. Avete apportato modifiche tutti alla homepage del progetto
3. ~~PANICO!!!~~
Bisogna fare in modo che si giunga a **una versione unica** che **comprenda tutte le modifiche** effettuate e **non escluda nulla** né di ciò che già esisteva, né delle novità



Fondamenti di Version Control – Introduzione al version control

Come funziona - Esempio

1. 3 di voi lavorano su un progetto
2. Avete apportato modifiche tutti alla homepage del progetto
3. **PANICO!!!**
Bisogna fare in modo che si giunga a **una versione unica** che **comprenda tutte le modifiche** effettuate e **non escluda nulla** né di ciò che già esisteva, né delle novità

4. Non buttiamo il codice sul repository sperando nella fortuna

Ma sfruttiamo i vantaggi del **versioning**

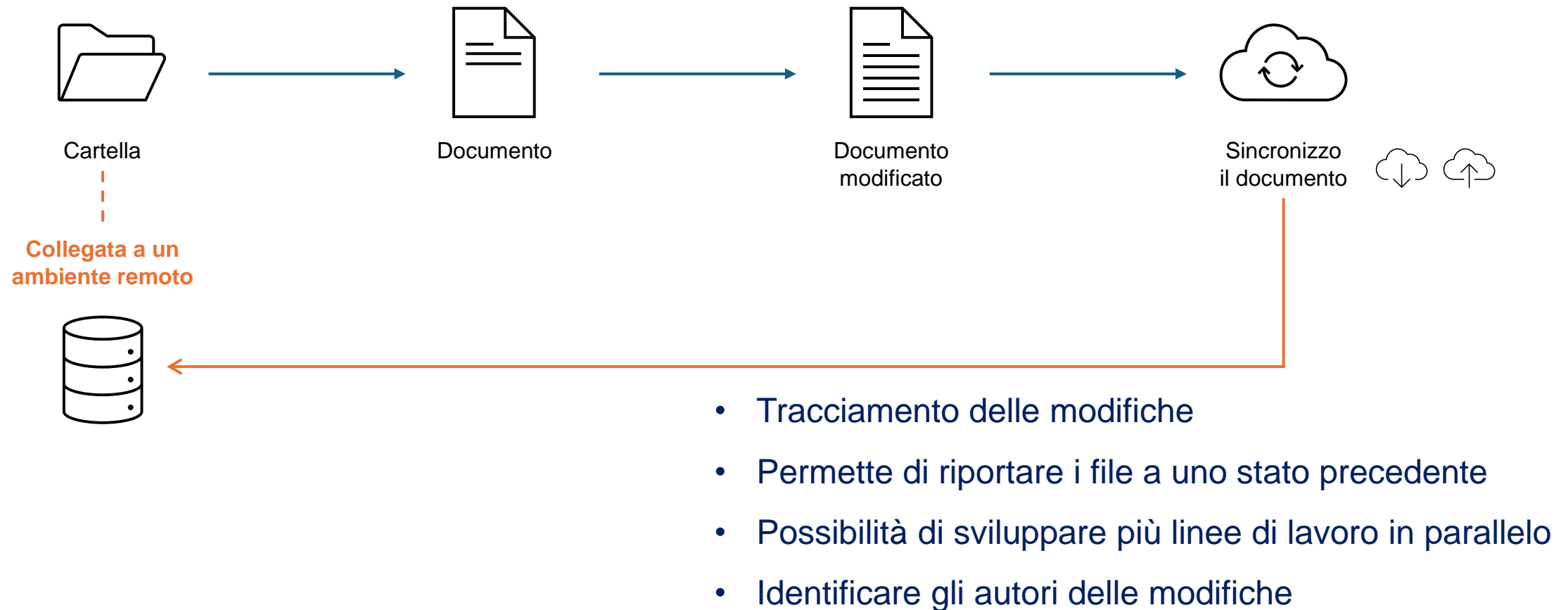
Versionamento del codice



Quali sono le caratteristiche principali del version control?

Fondamenti di Version Control – Ripasso generale

AGGIORNAMENTO DI UN FILE CON IL VERSION CONTROL



Fondamenti di Version Control – Ripasso generale

Definizione

Il **Version Control** o versionamento consente di **tracciare i cambiamenti** di un file o a un insieme di file. Permette, tra le altre cose, di **riportare i file** o l'intero progetto **a uno stadio precedente**, **visualizzare le modifiche nel corso del tempo**, sviluppare **più linee di lavoro in parallelo** e **identificare gli autori** delle modifiche.

docs.italia.it

**È utile per progetti personali
o per progetti in
collaborazione con altri?**

**È utile per progetti personali
o per progetti in
collaborazione con altri?**

Per entrambi!

Fondamenti di Version Control – Ripasso generale

Progetti personali:

- Tenere traccia dello storico
- Riportare i file ad uno stato precedente
- Poter lavorare al mio progetto scaricandolo da macchine differenti
- Creare un portfolio progetti

Progetti in collaborazione:

- Punti del personale
- Sviluppare con altri in parallelo
- Identificare gli autori delle modifiche

**Nello sviluppo collaborativo
cosa bisogna fare per
migliorare la condivisione del
codice e l'avanzamento del
progetto?**

Fondamenti di Version Control – Ripasso generale

Stabilire delle regole!

Organizzazione del codice (ad es. indentazione)

Struttura delle sezioni

Linguaggio comune per i commenti, le variabili e i commit

Definire il processo di aggiornamento del branch principale (Master)

Cos'è e cosa ci fornisce un VCS?

Fondamenti di Version Control – Ripasso generale

Version Control System (VCS)

- Fornisce supporto alla memorizzazione dei codici sorgenti
- Fornisce uno storico di ciò che è stato fatto
- Può fornire un modo per lavorare in parallelo su diversi aspetti dell'applicazione
- Può fornire un modo per lavorare in parallelo senza intralciarsi a vicenda

Parte 2

Fondamenti di Version Control – Storia dei VCS

Tipologie

Locale

Centralizzato

Distribuito

Fondamenti di Version Control – Storia dei VCS

Tipologie

Locale

Composti da un database di versionamento speciale, il cui compito è quello di memorizzare i cambiamenti dei file

Non permette la collaborazione perché lavora in locale sulla macchina

Centralizzato

Distribuito

Fondamenti di Version Control – Storia dei VCS

Tipologie

Locale

Composti da un database di versionamento speciale, il cui compito è quello di memorizzare i cambiamenti dei file

Non permette la collaborazione perché lavora in locale sulla macchina

Centralizzato

Il Database di versionamento viene spostato su un server. I computer si collegano al database e sincronizzano le proprie versioni

La presenza di un server non permette di lavorare offline e di recuperare le informazioni se questo viene distrutto.

Distribuito

Fondamenti di Version Control – Storia dei VCS

Tipologie

Locale

Composti da un database di versionamento speciale, il cui compito è quello di memorizzare i cambiamenti dei file

Non permette la collaborazione perché lavora in locale sulla macchina

Centralizzato

Il Database di versionamento viene spostato su un server. I computer si collegano al database e sincronizzano le proprie versioni

La presenza di un server non permette di lavorare offline e di recuperare le informazioni se questo viene distrutto.

Distribuito

È un sistema ibrido che comprende entrambe le tipologie viste in precedenza.

Prevede sempre un server con il database di versionamento, ma i client hanno una replica del database e la versione finale dei vari file.

Ogni collaboratore ha la propria copia di lavoro e, se una delle macchine non è più accessibile, una copia di lavoro si può recuperare. Permette di lavorare offline perché esiste una copia locale del progetto.

Fondamenti di Version Control – Storia dei VCS

Esempi di centralizzato e distribuito

GIT	SVN
Git è open source; è stato sviluppato da Linus Torvalds nel 2005. ha la sua forza nella rapidità e nell'integrità dei dati	Apache Subversion è open source sotto Apache license .
strumento Distribuito	strumento Centralizzato
ogni utente ha la propria “copia” dei sorgenti in locale	esiste un repository centrale al quale bisogna collegarsi per scaricare il progetto, confrontare il codice, committare

Fondamenti di Version Control – Storia dei VCS

Esempi di centralizzato e distribuito

GIT	SVN
NON serve una connessione per effettuare le operazioni	è necessaria una connessione per effettuare le operazioni
più difficile da imparare (ha una curva di apprendimento Maggiore), ha più concetti e comandi di SVN	molto più semplice da padroneggiare rispetto a git.
non ha un'interfaccia utente nativa	ha un'interfaccia utente semplice

Fondamenti di Version Control – Riassumendo

Step per avviare un aggiornamento di progetto

Ogni PC ha un repository **LOCALE** che viene generato **CLONANDO** il repository remoto (**PRODUZIONE**)

Le modifiche personali vengono verificate e committate sul repository locale

A questo punto, la situazione in **LOCALE** è definita, bisogna portare le modifiche sul repository remoto (**PRODUZIONE**)

L'operazione di "riversare" il repository **LOCALE** con quello **REMOTO** viene detta **PUSH**

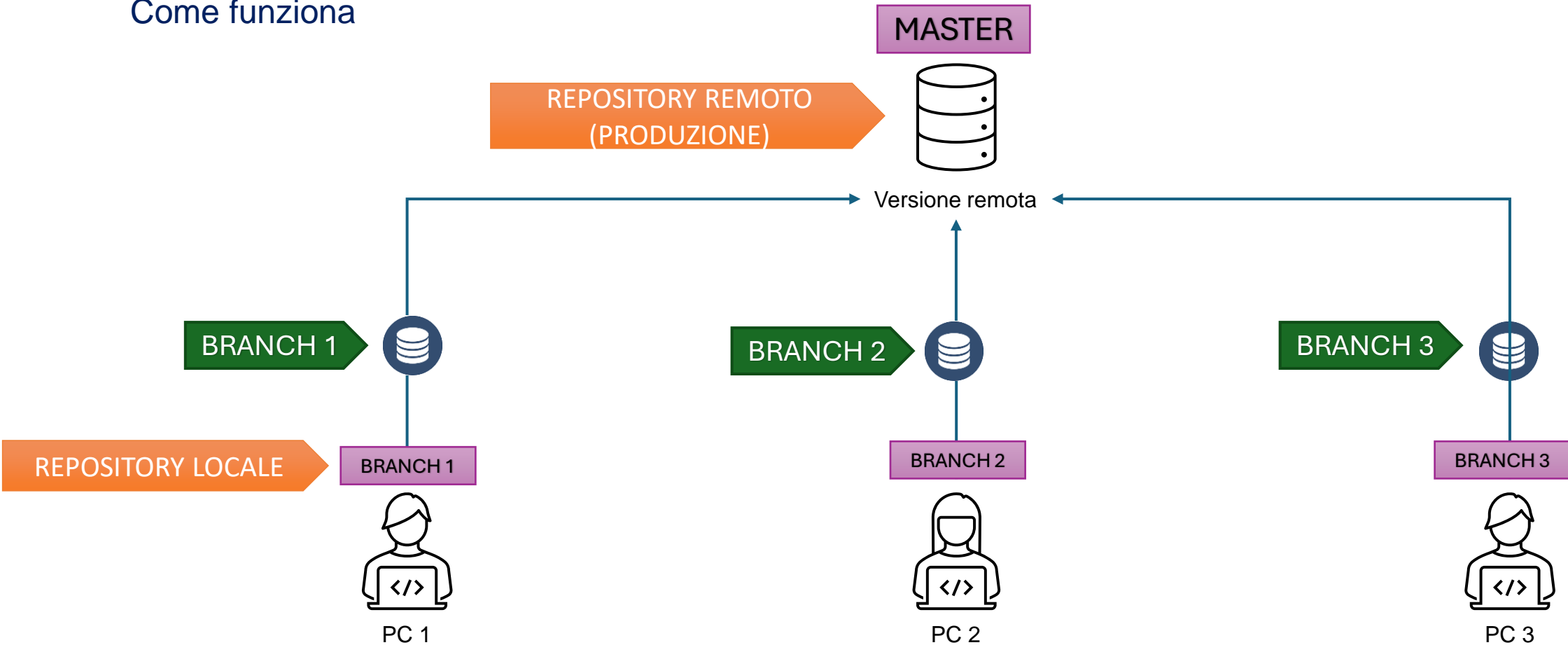
Può (anzi DEVE) essere effettuata anche l'operazione inversa, cioè "riversare" il repository **REMOTO** in quello **LOCALE**; questa operazione viene detta **PULL**

Fondamenti di Version Control – Branches

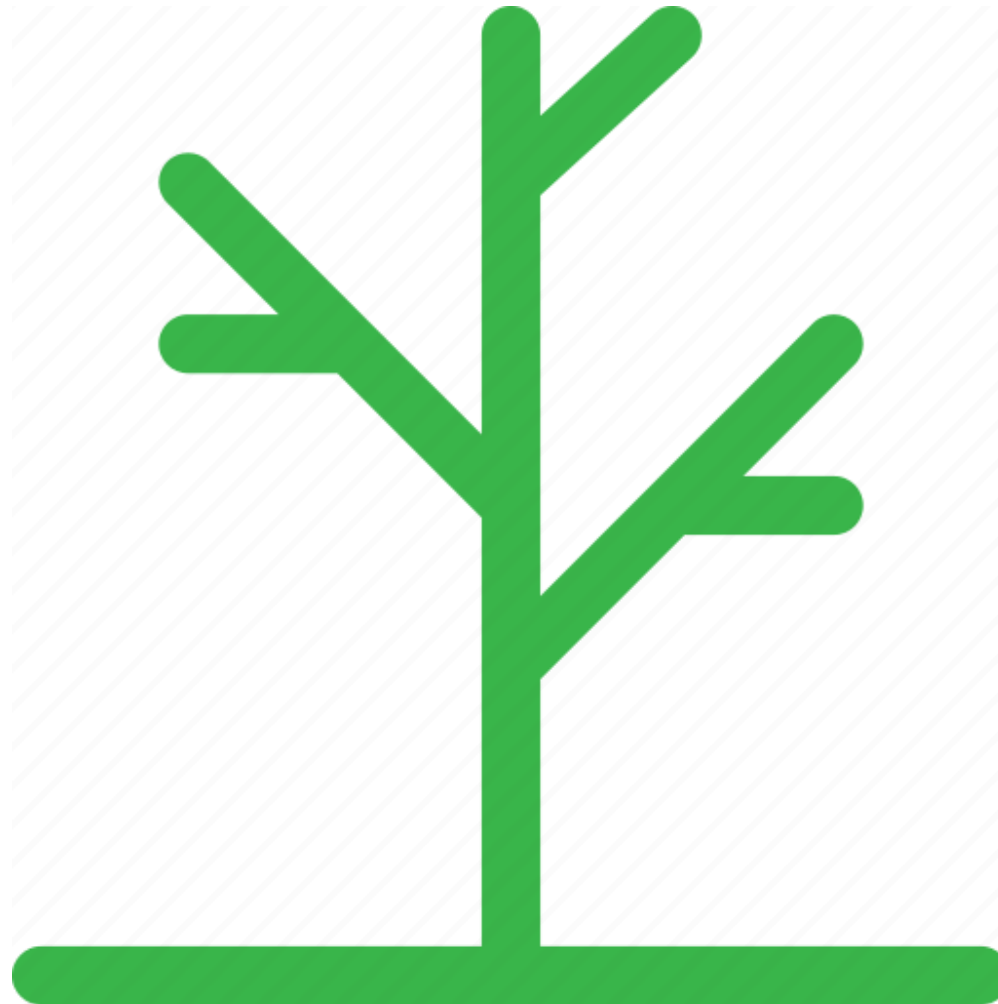
Git introduce il concetto dei **branches** (letteralmente "rami") che permettono di creare delle **ramificazioni del progetto**

Fondamenti di Version Control – Branches

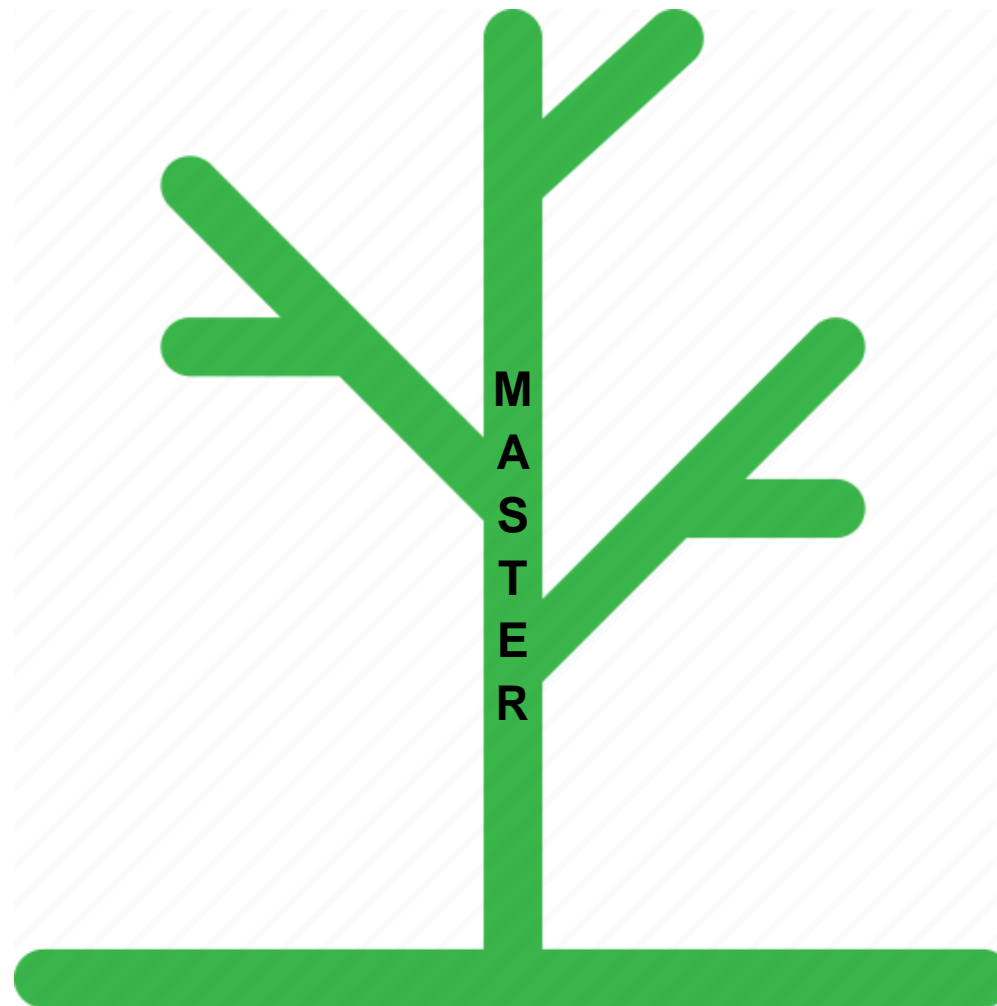
Come funziona



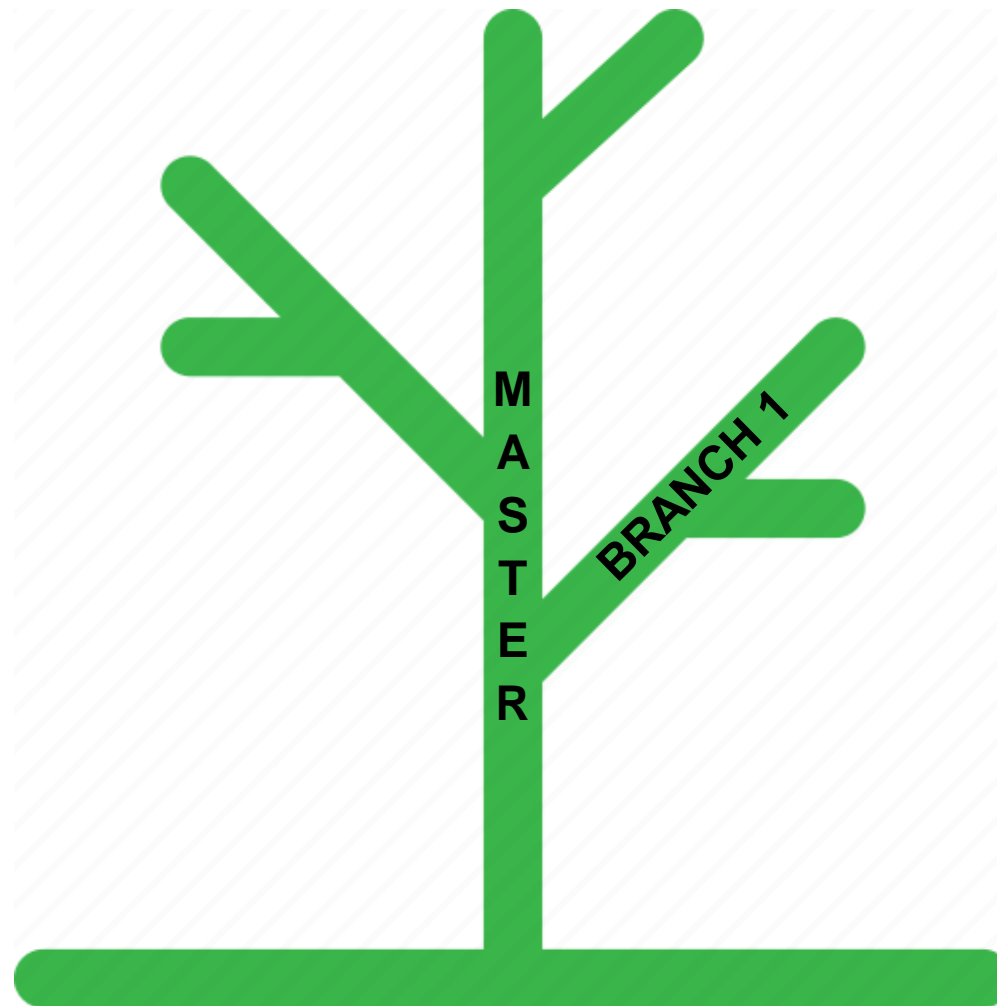
Fondamenti di Version Control – Branches



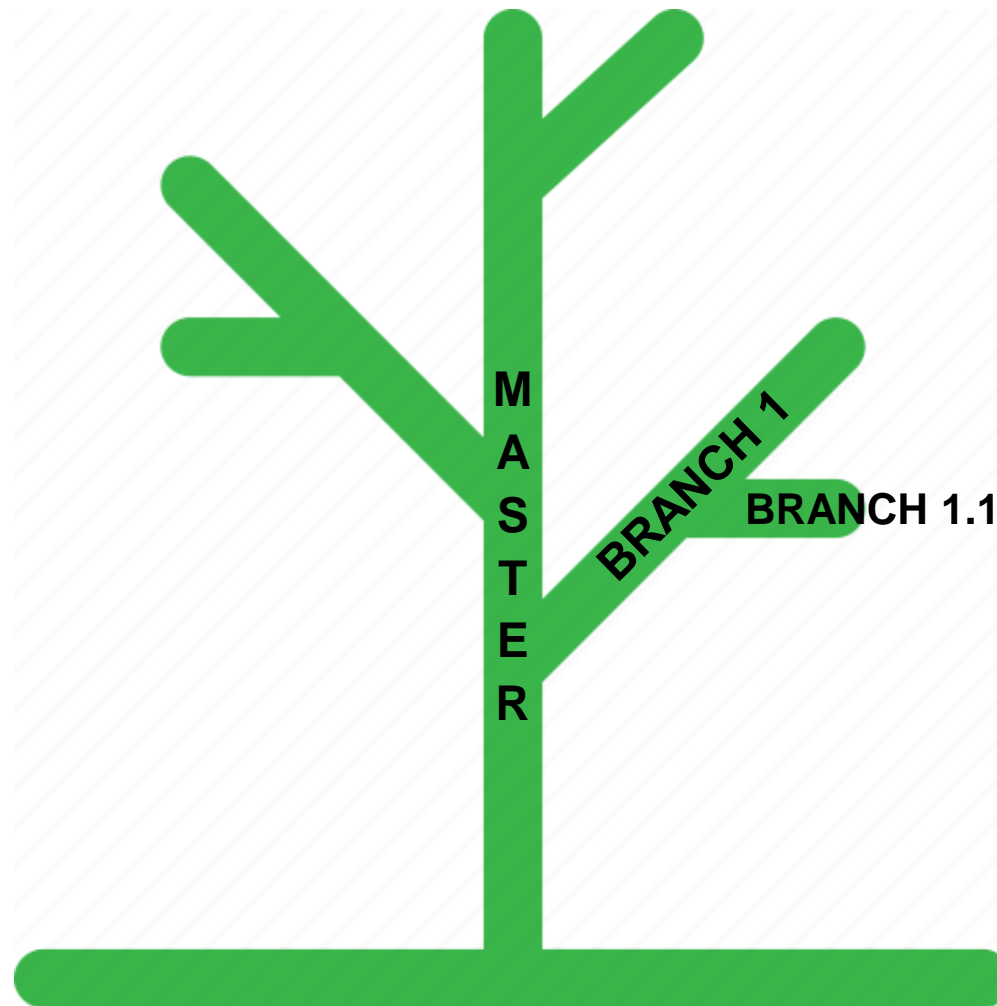
Fondamenti di Version Control – Branches



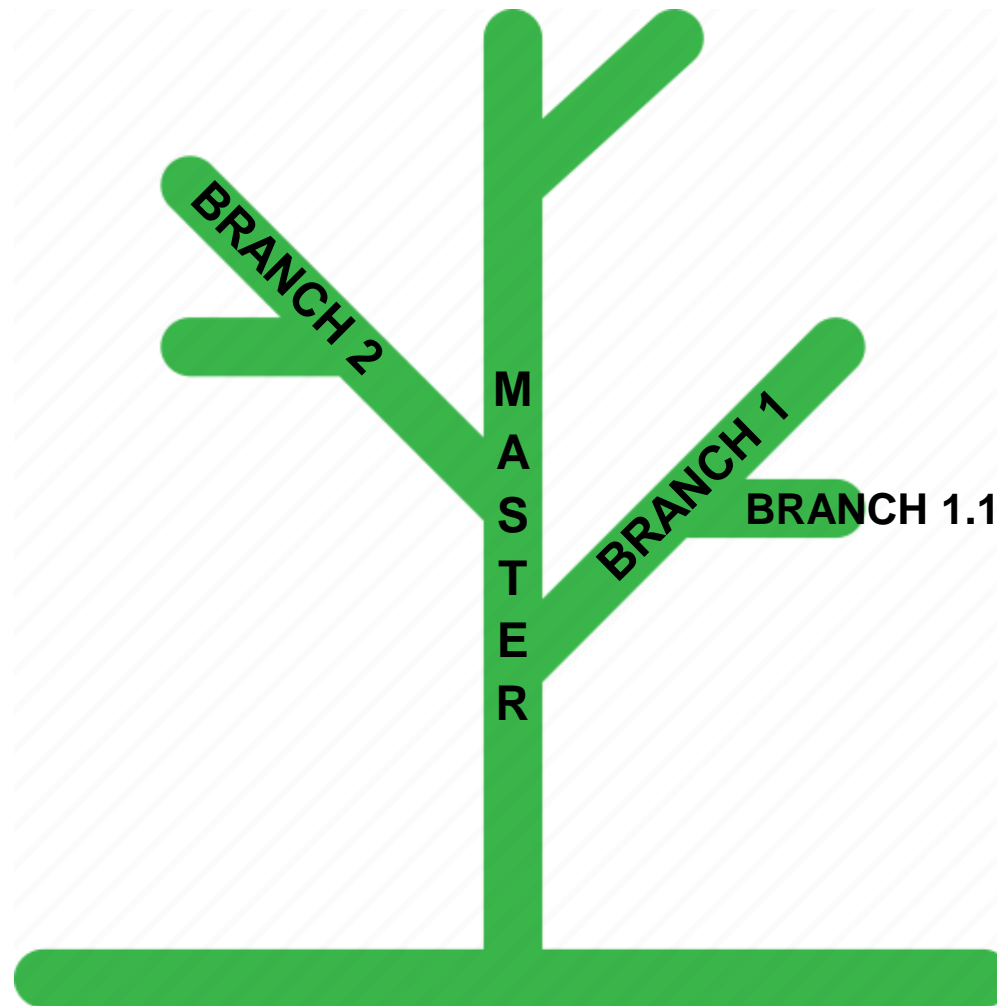
Fondamenti di Version Control – Branches



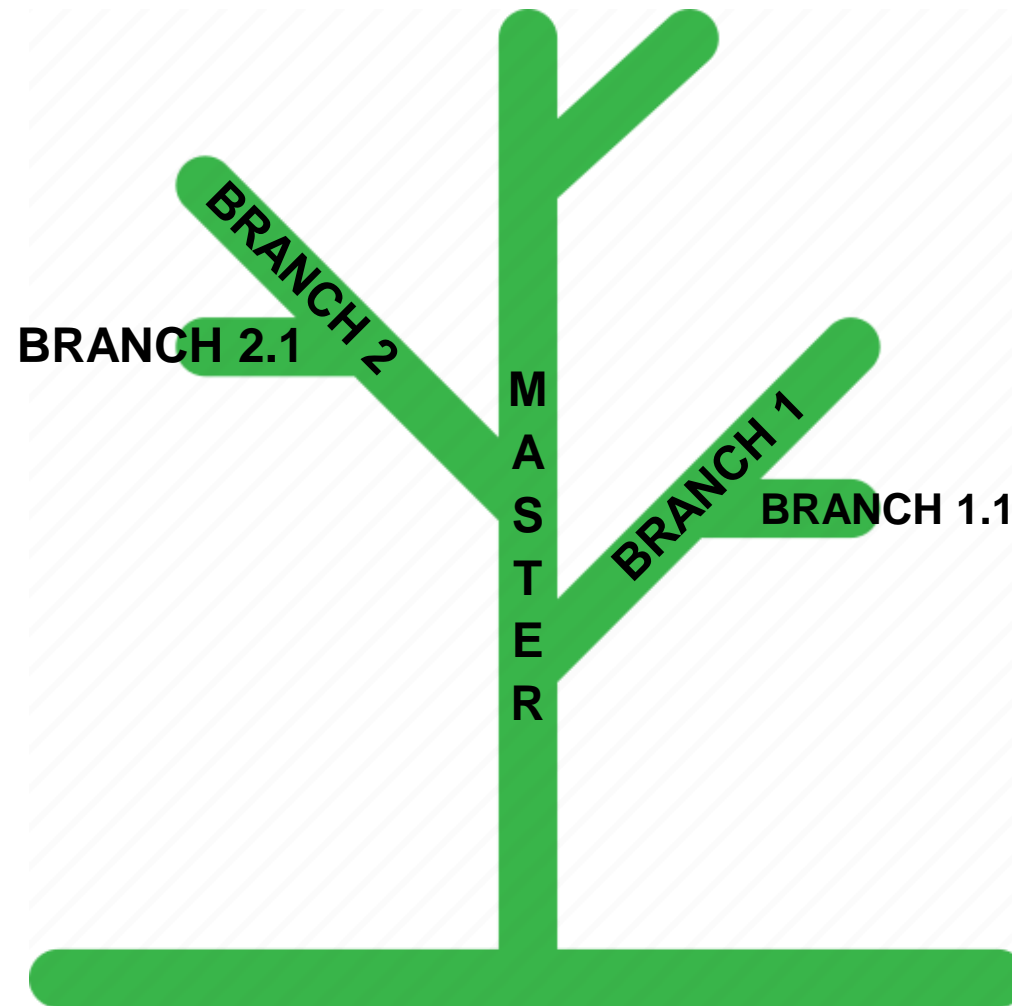
Fondamenti di Version Control – Branches



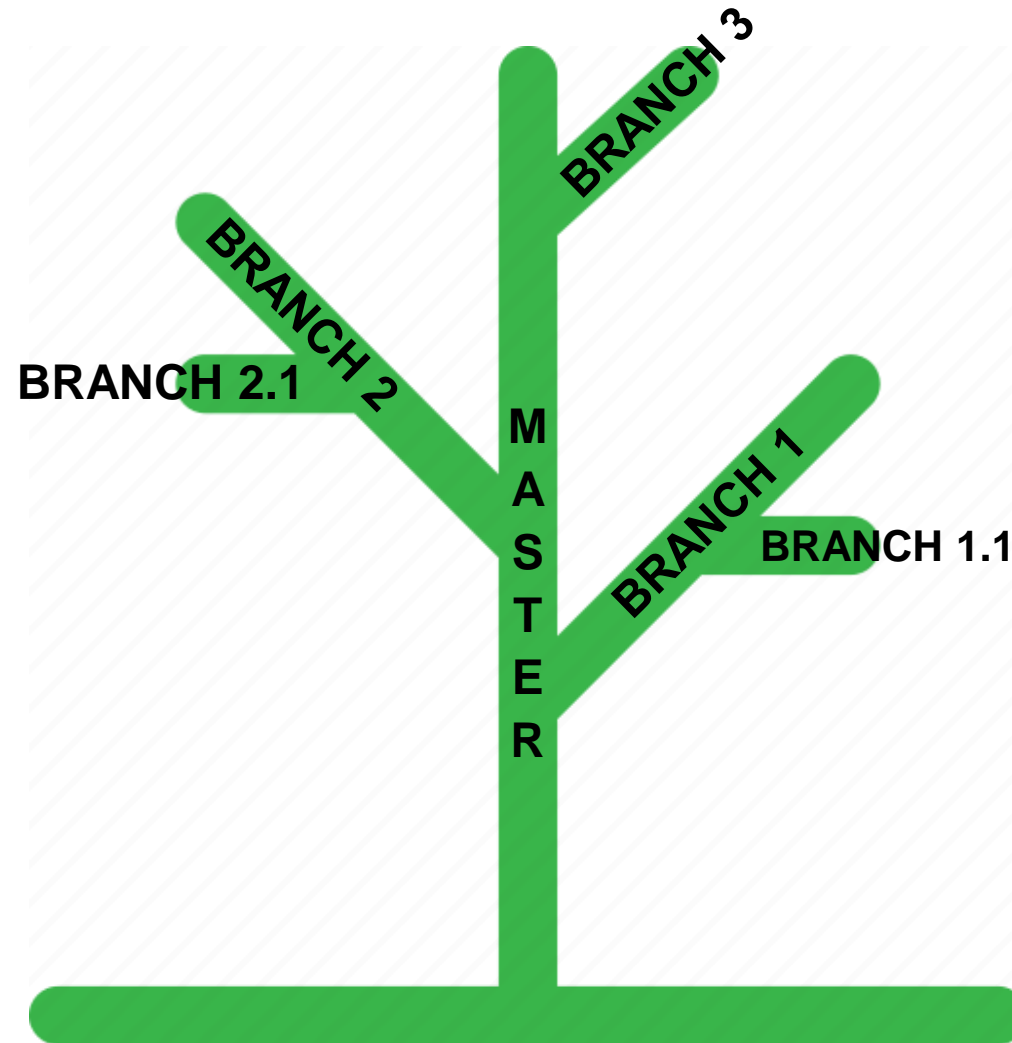
Fondamenti di Version Control – Branches



Fondamenti di Version Control – Branches



Fondamenti di Version Control – Branches



Fondamenti di Version Control – Branches

Git introduce il concetto dei **branches** (letteralmente "rami") che permettono di creare delle **ramificazioni del progetto**

La struttura va vista proprio come un albero

- Le radici sono il master
- Dal master si diramano i branches, 1 per ogni sviluppo (sviluppatore) diverso
- Capita che si crei un branch da un branch e non da master, ma non è la norma

Quali tipologie di VCS esistono?

Quali tipologie di VCS esistono?

Locale

Centralizzato

Distribuito

Git di che tipologia è?

Locale

Centralizzato

Distribuito

Git di che tipologia è?

Locale

Centralizzato

Distribuito

SVN di che tipologia è?

Locale

Centralizzato

Distribuito

SVN di che tipologia è?

Locale

Centralizzato

Distribuito

Quale non permette la collaborazione?

Locale

Centralizzato

Distribuito

Quale non permette la collaborazione?

Locale

Centralizzato

Distribuito

Quale lavora collegato a un server fisico?

Locale

Centralizzato

Distribuito

Quale lavora collegato a un server fisico?

Locale

Centralizzato

Distribuito

Quale lavora creando una replica locale del database?

Locale

Centralizzato

Distribuito

Quale lavora creando una replica locale del database?

Locale

Centralizzato

Distribuito

Con quali si può lavorare offline?

Locale

Centralizzato

Distribuito

Con quali si può lavorare offline?

Locale

Centralizzato

Distribuito

Chi introduce il concetto di branches?

SVN

GIT

Salvataggio di
un file in locale

Chi introduce il concetto di branches?

SVN

GIT

Salvataggio di
un file in locale

Parte 3

Creazione repository su Github

Quali sono i due file iniziali che vengono creati?

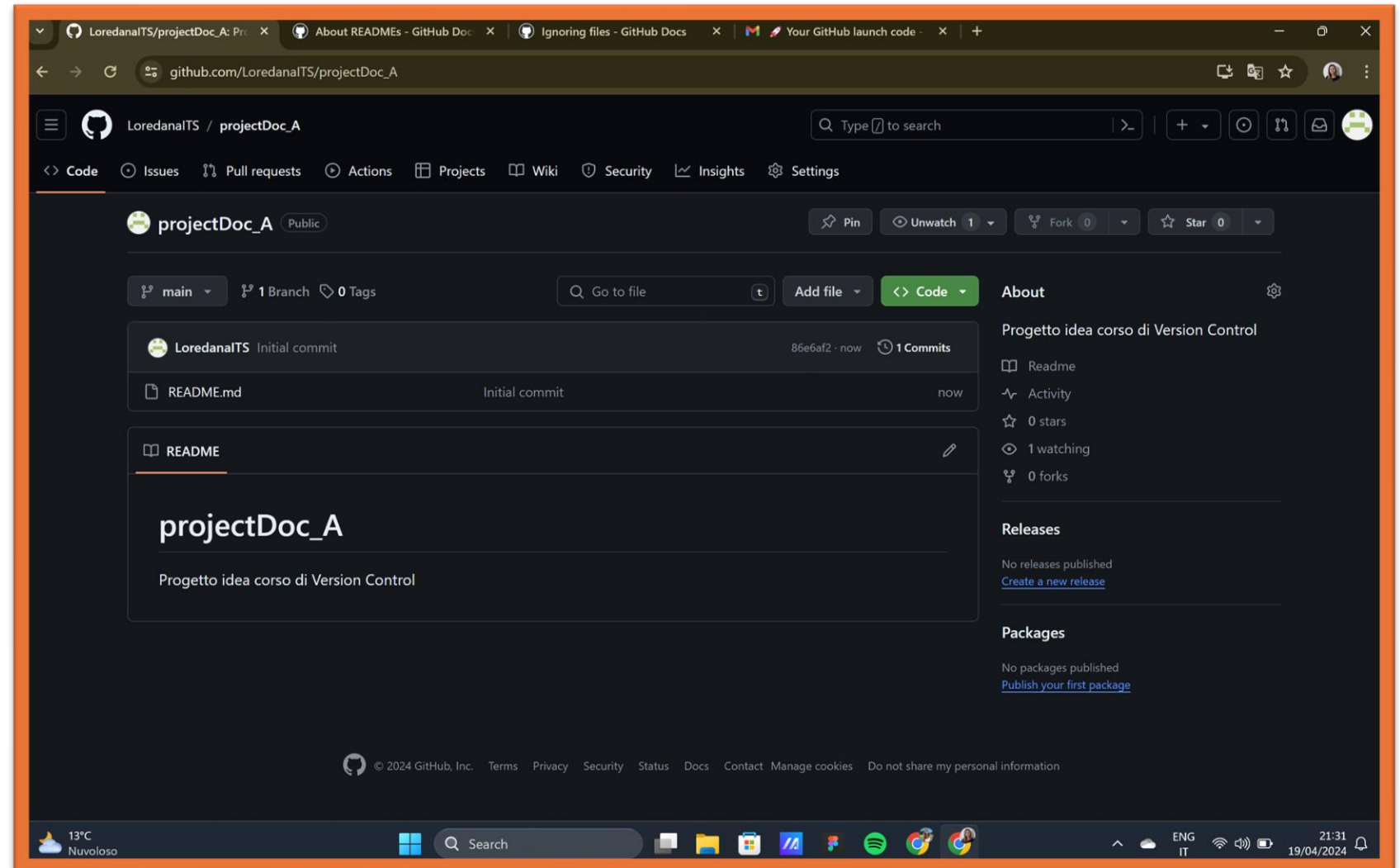
A cosa ci serve il file readme.md?

Fondamenti di Version Control – Ripasso concetti git

Readme file

File di markdown
www.markdownguides.org

- Descrizione del progetto
- Regole di collaborazione
- Indicazioni generali
- Documentazione



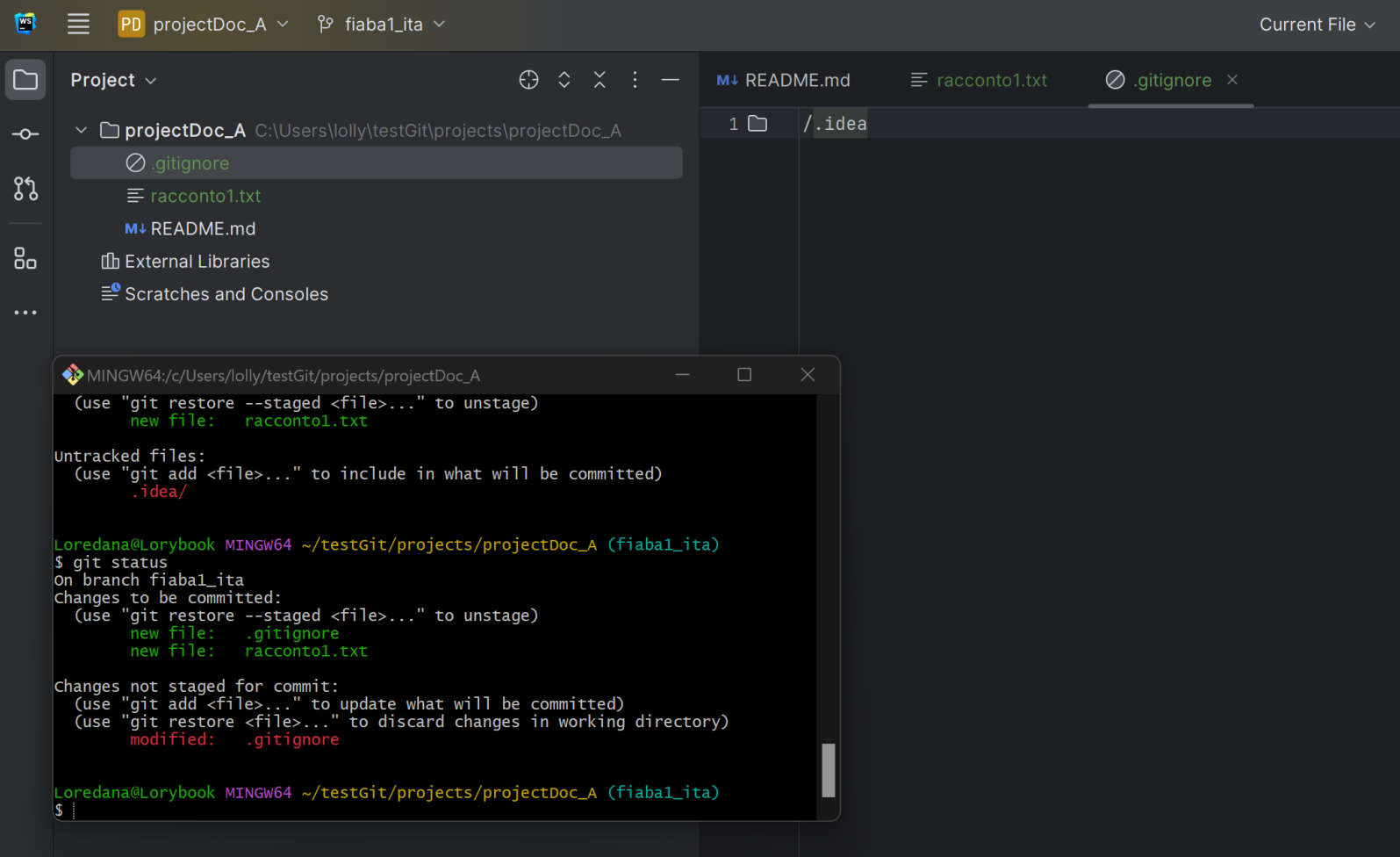
Creazione repository su Github

A cosa ci serve il file .gitignore?

Fondamenti di Version Control – Ripasso concetti git

.gitignore

Permette di far
ignorare dei file dallo
storico del repository



The screenshot shows an IDE interface with a project named 'projectDoc_A'. The file explorer on the left lists files: `.gitignore`, `racconto1.txt`, `README.md`, `External Libraries`, and `Scratches and Consoles`. The editor on the right shows the `.gitignore` file with the content `/.idea`. Below the editor, a terminal window displays the output of the `git status` command:

```
MINGW64:/c/Users/lolly/testGit/projects/projectDoc_A
(use "git restore --staged <file>..." to unstage)
new file:   racconto1.txt

Untracked files:
(use "git add <file>..." to include in what will be committed)
.idea/

Loredana@Lorybook MINGW64 ~/testGit/projects/projectDoc_A (fiaba1_ita)
$ git status
On branch fiaba1_ita
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore
    new file:   racconto1.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   .gitignore

Loredana@Lorybook MINGW64 ~/testGit/projects/projectDoc_A (fiaba1_ita)
$
```

Quale comando uso per copiare un repository sulla mia macchina?

Fondamenti di Version Control – Progetto 1

Clone del repository nella propria macchina

```
git clone 'url'
```

Clone: operazione con cui viene creato un **repository locale** facendo una sorta di "**copia-incolla**" dal **repository remoto**; questo permette di avere il **collegamento tra repository locale e remoto** senza dover fare altri passaggi (e quindi poter fare push, pull, eccetera)

Quale comando uso per conoscere lo stato delle mie modifiche locali rispetto all'area di staging?

Fondamenti di Version Control – Progetto 1

Git status

Ti permette di vedere quali modifiche sono state sottoposte a staging, quali no e quali file non vengono tracciati da Git.

Fondamenti di Version Control – Progetto 1

Alternativa a git status

`git diff <nome-file>`

- Permette di visualizzare quali sono le modifiche di un file in locale rispetto al suo stato nell'area di staging

`git diff --staged`

- Permette di visualizzare quali sono le modifiche in locale rispetto allo stato nell'area di staging di tutti i file

Quale comando uso per conoscere gli aggiornamenti delle modifiche del repository remoto rispetto al mio ultimo pull?

Fondamenti di Version Control – Progetto 1

Git fetch

- scarica i contenuti remoti senza alterare lo stato del repository locale
- permette di conoscere i cambiamenti fatti nel branch/repo remoto dal tuo ultimo pull

Caricamento file

Qual è il procedimento per caricare un file modificato in locale sul repository remoto?

Fondamenti di Version Control – Progetto 1

Inserire un file di testo ed effettuare il primo commit

Procedimento per il caricamento di un file

`Git add <nome-file>`

- Aggiunge i file modificati alla staging area
- Si potrebbero aggiungere tutti i file presenti con `git add *`, però potrebbe aggiungere file che non dobbiamo portare in stage

`Git commit -m «informazioni sul commit»`

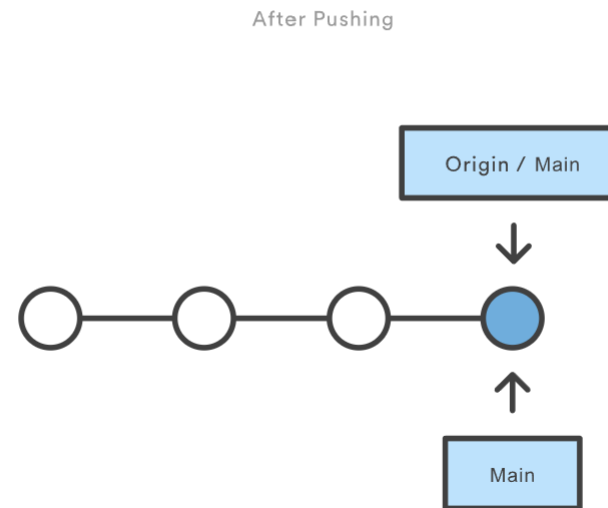
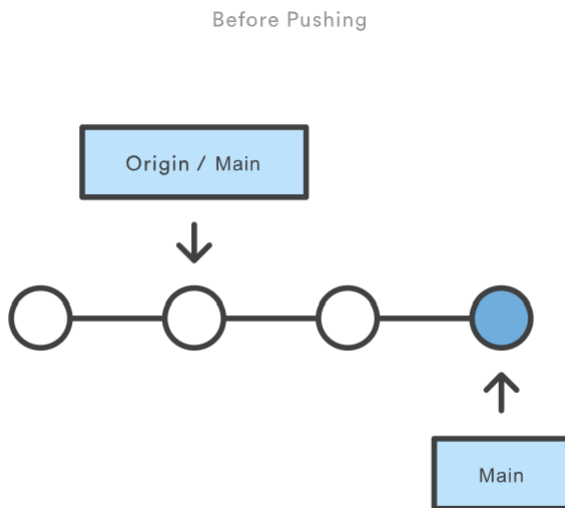
- Associa un messaggio alla modifica inserita in staging area e la preparo per il caricamento sul branch remoto

Fondamenti di Version Control – Progetto 1

Inserire un file di testo ed effettuare il primo commit

Procedimento per il commit

```
Git push origin <nome-branch>
```



Caricamento file

Perché è un'azione da evitare il push diretto su master?

Fondamenti di Version Control – Progetto 1

Caricare un file sul repository

Azione da evitare



- Il codice non viene controllato prima di essere portato sul ramo principale
- Le modifiche potrebbero generare problemi in produzione

Caricare un file sul repository

Come posso evitare che si effettui un push diretto su master?

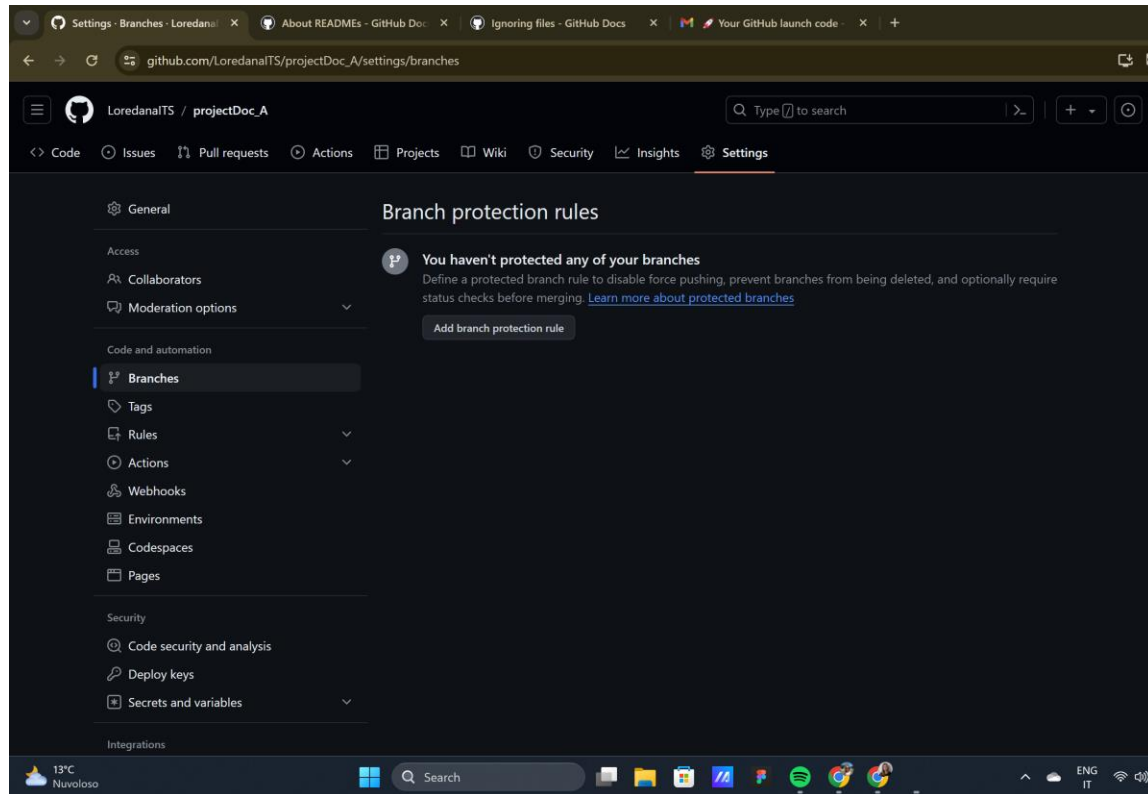
Fondamenti di Version Control – Progetto 1

Caricare un file sul repository

- Inserendo regole di protezione del branch principale, o anche di branch secondari
- Lavorando su branch che vengono creati con una logica condivisa da tutto il team
- Lavorando su branch secondari e portando la modifica su master solo quando è funzionante

Fondamenti di Version Control – Progetto 1

Regole di protezione del branch master



Branch protection rule



Protect your most important branches

[Branch protection rules](#) define whether collaborators can delete or force push to the branch and set requirements for any pushes to the branch, such as passing status checks or a linear commit history.

[Your GitHub Free plan](#) can only enforce rules on its public repositories, like this one.

Branch name pattern *

master

Protect matching branches

☒ Require a pull request before merging

When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

☒ Require approvals

When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.

Required number of approvals before merging: 1

☐ Dismiss stale pull request approvals when new commits are pushed

New reviewable commits pushed to a matching branch will dismiss pull request review approvals.

☐ Require review from Code Owners

Require an approved review in pull requests including files with a designated code owner.

☐ Require approval of the most recent reviewable push

Creazione di un branch

Quale comando uso per creare un nuovo branch?

Fondamenti di Version Control – Progetto 1

Creazione di un branch

Lavorare su un branch

Git Branch

- Permette di visualizzare la lista di branch esistenti
- Creare nuovi branch (senza spostarsi su di essi)

`git branch <nome-branch>`

- Eliminare branch con `git branch -d <nome-branch>`

Creazione di un branch

Quando creo un nuovo branch in locale questo è già visibile nel repository remoto?

Fondamenti di Version Control – Progetto 1

Creazione di un branch

Lavorare su un branch

Git Branch

- Permette di visualizzare la lista di branch esistenti
- Creare nuovi branch (senza spostarsi su di essi)

`git branch <nome-branch>`

- Eliminare branch con `git branch -d <nome-branch>`

NOTA

I branch creati in locale non saranno visibili sul repository fin quando non si effettuerà il primo commit su quel branch

**Con quale criterio scelgo
quando creare/eliminare un
branch?**

Con quale criterio scelgo quando creare/eliminare un branch?

- **Creo** un nuovo branch quando inizio a sviluppare una nuova funzionalità dell'applicazione o quando **lavoro su una proprietà specifica** (es traduzione in francese dei dialoghi)
- **Elimino** un branch quando ho **finito di lavorare alla funzionalità/proprietà** e la mergio nel master del progetto

Creazione di un branch

Quale comando uso per spostarmi su un branch?

Fondamenti di Version Control – Progetto 1

Creazione di un branch

Git checkout

- Permette di spostarsi tra i branch creati `git checkout <nome-branch>`
- Con `git checkout -b <nome-branch>` è possibile creare dei nuovi branch ed effettuare il passaggio al branch appena creato

Pull request

Cos'è una pull request?

Fondamenti di Version Control – Progetto 2

Cos'è una pull request?

La *Pull Request* è una richiesta, fatta all'autore originale di un software, di includere modifiche al suo progetto.

Quando una nuova Pull Request viene aperta, la piattaforma notifica al maintainer che è necessario affrontare le operazioni di revisione.

Quando faccio una pull request?

**Per integrare le mie modifiche
su un altro branch che
comandi dovrei dare?**

**Per integrare le mie modifiche
su un altro branch che
comandi dovrei dare?**

Merge

Comandi di merging

Fondamenti di Version Control – Progetto 2

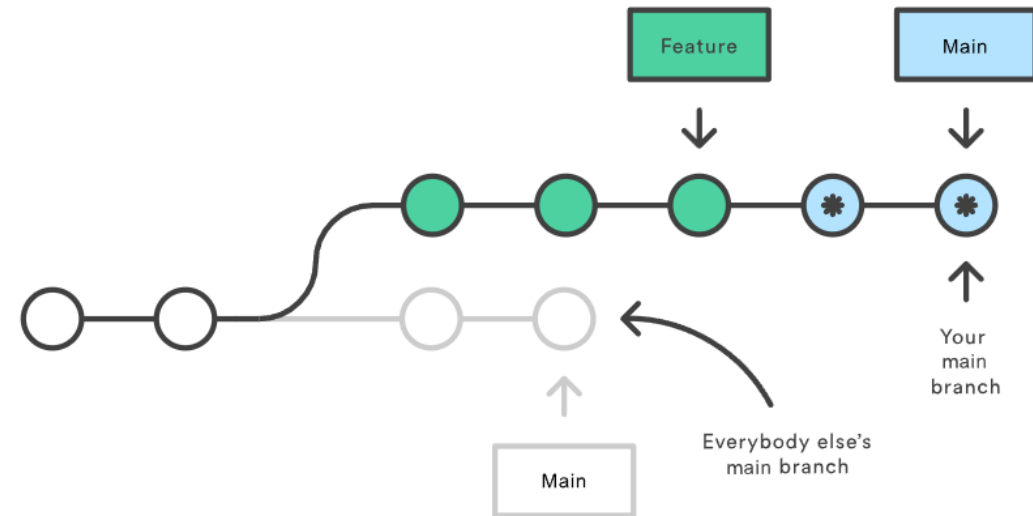
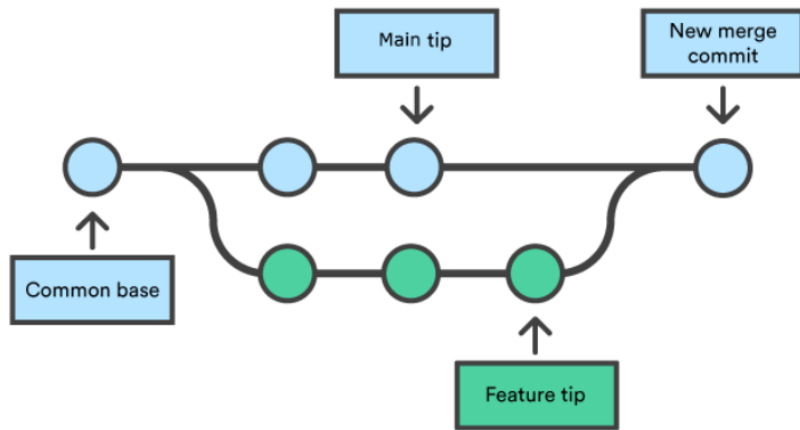
Effettuare un merge

```
git merge <branch-da-mergiare>
```

Permette di prendere le linee di sviluppo indipendenti create con i branch e integrarle in un unico branch

Fondamenti di Version Control – Progetto 2

Differenza tra merge e rebase



Fondamenti di Version Control – Progetto 2

Effettuare un rebase

```
git checkout <branch-base>
```

```
git rebase <branch-da-mergiare>
```

- Operazione simile al merge
- Non si porta dietro lo storico di un branch ma va ad effettuare un nuovo commit nel branch su cui si vuole spostare la modifica, senza portare dietro altre informazioni

NOTA

Effettua una riscrittura della cronologia, cosa che git merge non fa

Comandi per la cronologia

Fondamenti di Version Control – Progetto 1

Alternativa a git status

`git log`

- Permette di visualizzare la cronologia dei commit fatti

`git log --stat`

- Permette di visualizzare la cronologia con l'aggiunta di alcune statistiche sui file committati

Fondamenti di Version Control – Progetto 2

Badge di segnalazione

`git tag`

Git ha la capacità di contrassegnare punti specifici nella cronologia di un repository come importanti. In genere, le persone utilizzano questa funzionalità per contrassegnare i punti di rilascio

NOTA

L'aggiunta del tag potrebbe visualizzarsi dopo qualche minuto sul repository

Fondamenti di Version Control – Progetto 2

Tag di segnalazione

`git tag`

I tag possono essere:

- **Lightweight**: Sono semplici badge che vengono associati a un commit

```
git tag v0.8
```

- **Annotated**: Vengono salvate le informazioni di tagger name, email e data

```
git tag -a v1.0 -m "versione 1.0"
```

Comando per l'accantonamento

Fondamenti di Version Control – Progetto 2

Accantonamento di un file

```
git stash save "add style to our site"
```

Si applica a file nell'area di lavoro locale

Permette di accantonare le modifiche locali e poterle recuperare quando necessario ed effettua un ripristino della copia di lavoro

Fondamenti di Version Control – Progetto 2

Accantonamento di un file

`git stash list`

```
$ git stash list
stash@{0}: On main: add style to our site
stash@{1}: WIP on main: 5002d47 our new homepage
stash@{2}: WIP on main: 5002d47 our new homepage
```

Fondamenti di Version Control – Progetto 2

Accantonamento di un file

```
git stash pop stash@{2}
```

Permette di recuperare una modifica portata in stash e reinserirla nel proprio flusso di lavoro locale

Rimuovendole dall'area di accantonamento

Riapplica lo stash più recente se questo non viene specificato con `stash@{numero del file in lista di stash}`

Fondamenti di Version Control – Progetto 2

Accantonamento di un file

```
git stash apply
```

Permette di recuperare una modifica portata in stash e reinserirla nel proprio flusso di lavoro locale, mantenendola comunque nell'area di stash

Questa procedura è utile se si desidera applicare le stesse modifiche accantonate a più branch

Fondamenti di Version Control – Progetto 2

Accantonamento di un file

```
git stash clear
```

Rimuove tutti gli elementi portati in stash

```
git stash drop stash@{2}
```

Rimuove solo lo specifico elemento della lista portato in stash

Comandi di modifica

Fondamenti di Version Control – Progetto 2

Modifica del commit

Quando il commit è ancora nell'area di staging e si vuole aggiornare

```
git add <nome-file>
```

```
git commit --amend -m «messaggio di  
sovrascrittura»
```

Permette di unire nuove modifiche presenti nella staging area
all'ultimo commit appena salvato

Comandi di ripristino/ cancellazione

Fondamenti di Version Control – Progetto 2

Reset di un commit

```
git reset <file>
```

Rimuove i file dalla staging area.

Utilizza questo comando per annullare un git add *file*.

Si può anche usare per rimuovere tutto dallo stage (non specificando il file).

Fondamenti di Version Control – Progetto 2

Reset di un commit

```
git reset --soft [<commit>]
```

Sposta la head al commit specificato, mantenendolo comunque nell'area di staging

```
git reset --hard [<commit>]
```

Torna al commit precedente rimuove tutte le modifiche che non sono nell'area di staging

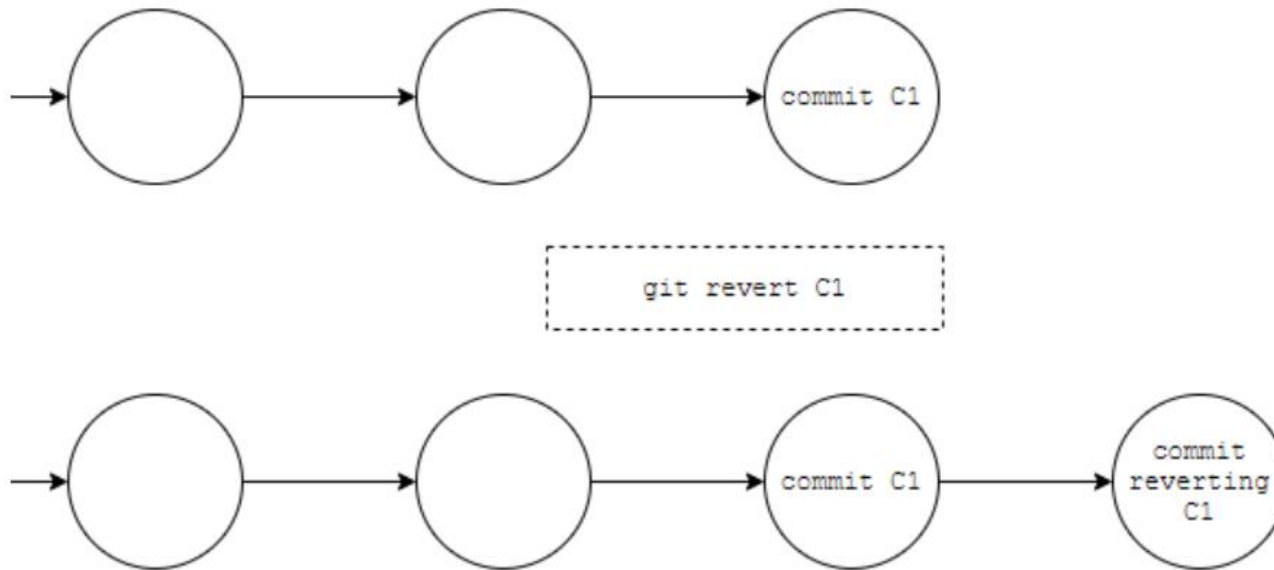
NOTA

Git reset cambia la storia dei commit quindi usarlo solo quando strettamente necessario

Fondamenti di Version Control – Progetto 2

Revert di un commit

`git revert <commit da annullare>`



Fondamenti di Version Control – Progetto 2

Revert di un commit

```
git revert <commit da annullare>
```

Permette di rimuovere le modifiche fatte con un commit, creando un nuovo commit.

Con `git log` o andando a cercare il nome del commit su github, è possibile trovare il nome dell'ultimo commit utile a cui si vuole ritornare

Fondamenti di Version Control – Progetto 2

Remove file da staging area

```
git rm <file>
```

Permette di rimuovere un file dal repository.
Nella staging area si può visualizzare la rimozione del file.

```
git rm --cached <file>
```

Permette di mantenere una copia locale del file

Fondamenti di Version Control – Progetto 2

Prendere una modifica da un commit precedente

```
git cherry-pick <nome-commit>
```

"copia" un commit creandone uno nuovo nel branch corrente con lo stesso messaggio applicando le modifiche come se fosse un commit diverso