



UNIVERSITÀ DEGLI STUDI DI MILANO

Dept. of Computer Science

Lab 2: Git

Carlo Bellettini, Matteo Camilli

carlo.bellettini@unimi.it

matteo.camilli@unimi.it

Ingegneria del software a.a. 2016/17

Git: creare un repository

- **git init**

- Solitamente il primo comando ad essere eseguito per inizializzare un repository vuoto

- Uso

```
$ git init
```

- Trasforma la directory corrente in un repository Git. Viene creata una directory `.git` con i file di configurazione necessari

- **git clone**

- questo comando crea una copia (intera history) di un repository esistente

- Uso

```
$ git clone <repo> <directory>
```

- clona il repository locato in `<repo>` all'interno della directory (opzionale) `<directory>`

- Esempio

```
$ git clone ssh://john@example.com/path/to/my-project.git
```

Git: configurazione

- **git config**

- il comando git config consente di configurare l'installazione di Git (oppure un singolo repository) da linea di comando. E' possibile ad esempio definire informazioni dell'utente e preferenze nel comportamento di Git.

- Esempi comuni

```
# Tell Git who you are
```

```
$ git config --global user.name "John Smith"
```

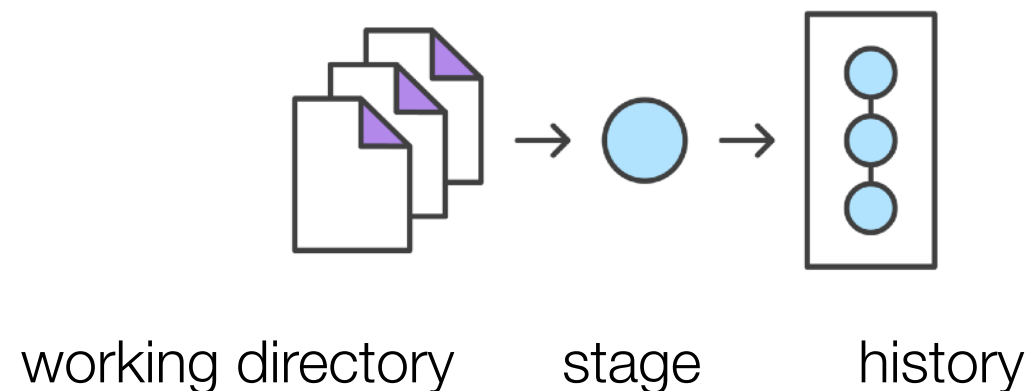
```
$ git config --global user.email john@example.com
```

```
# Select your favorite text editor
```

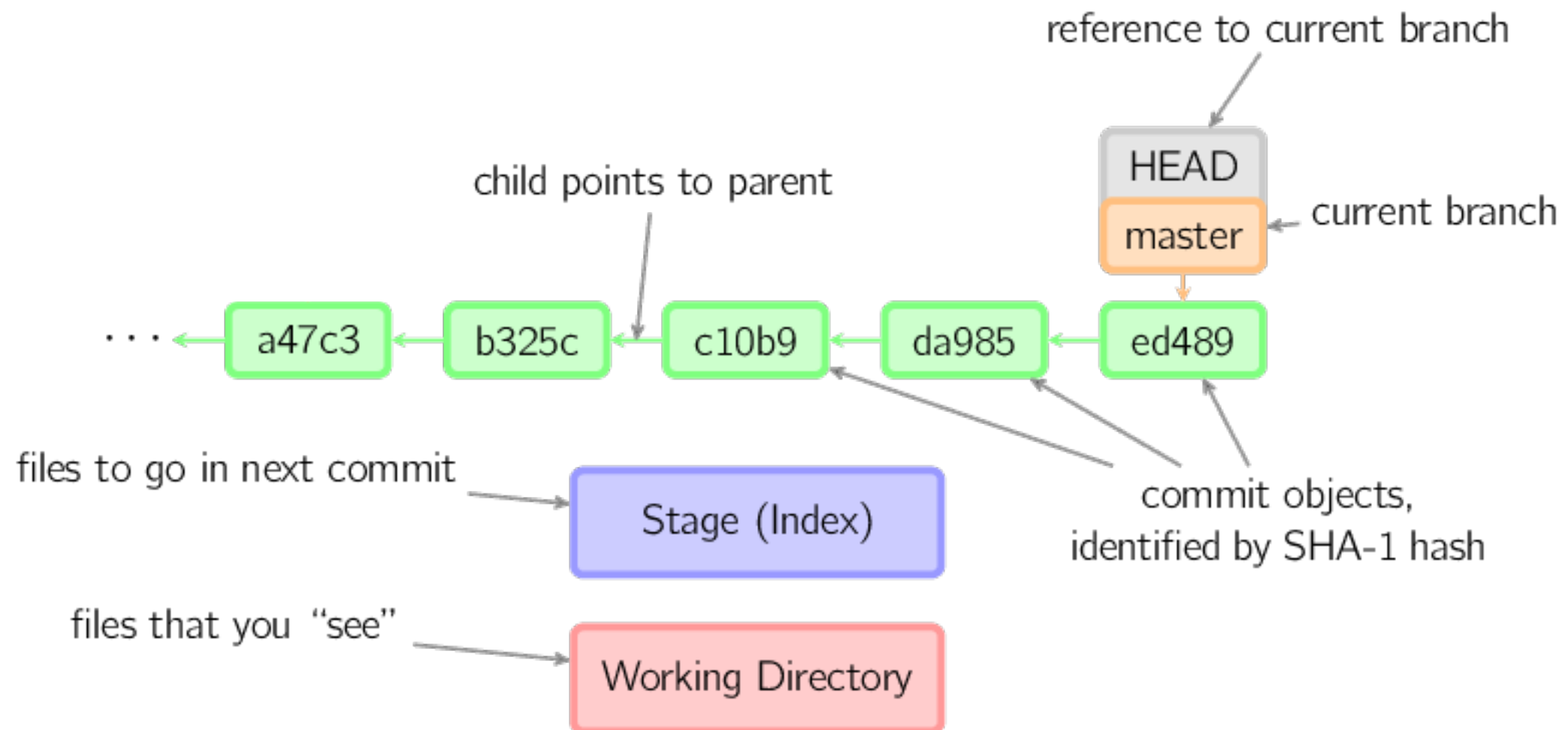
```
$ git config --global core.editor vim
```

Git: salvare cambiamenti

- **git add + git commit**
 - comandi fondamentali per l'uso di Git.
 - mezzo con cui le versioni vengono registrate nella history
- **edit/stage/commit pattern**
 - i file nella **working directory** vengono modificati
 - quando vogliamo registrare lo **stato corrente** del progetto, viene eseguito lo **stage** dei cambiamenti attraverso il comando **git add**
 - una volta che lo stage stato eseguito, lo **snapshot** del progetto viene salvato nella **history** attraverso il comando **git commit**



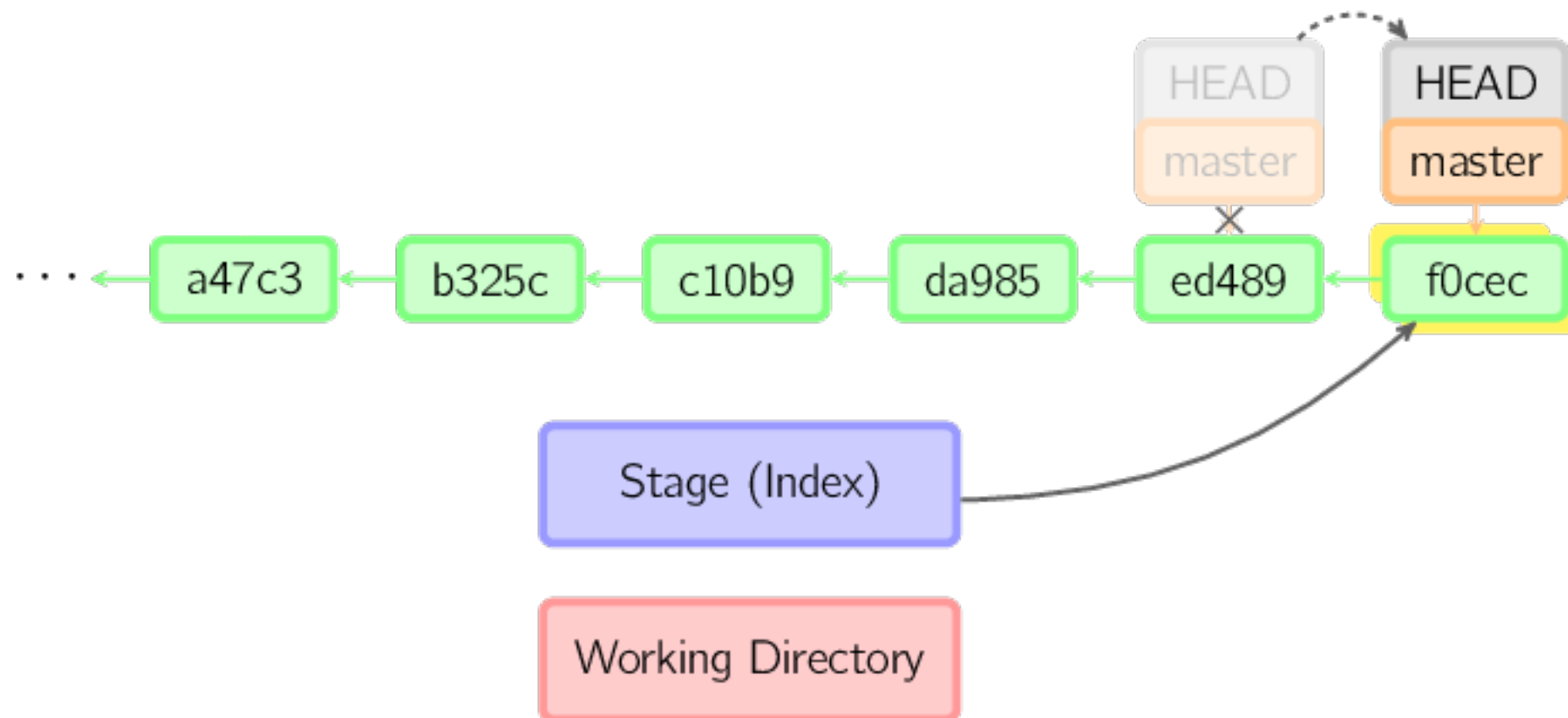
Git: struttura interna



- `git log --graph`, mostra la History del branch corrente
- `git log --oneline --graph --all`, mostra History di tutti i branch in modo compatto

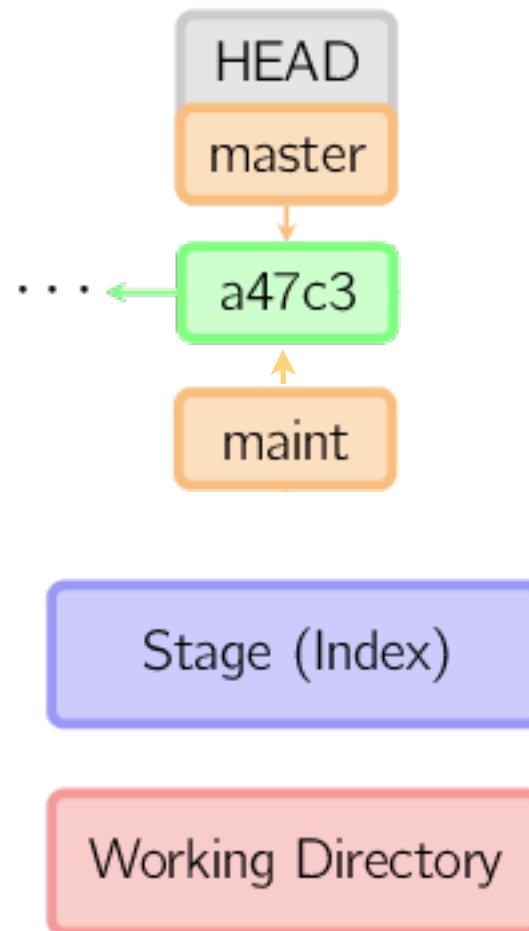
Git commit

`git commit`



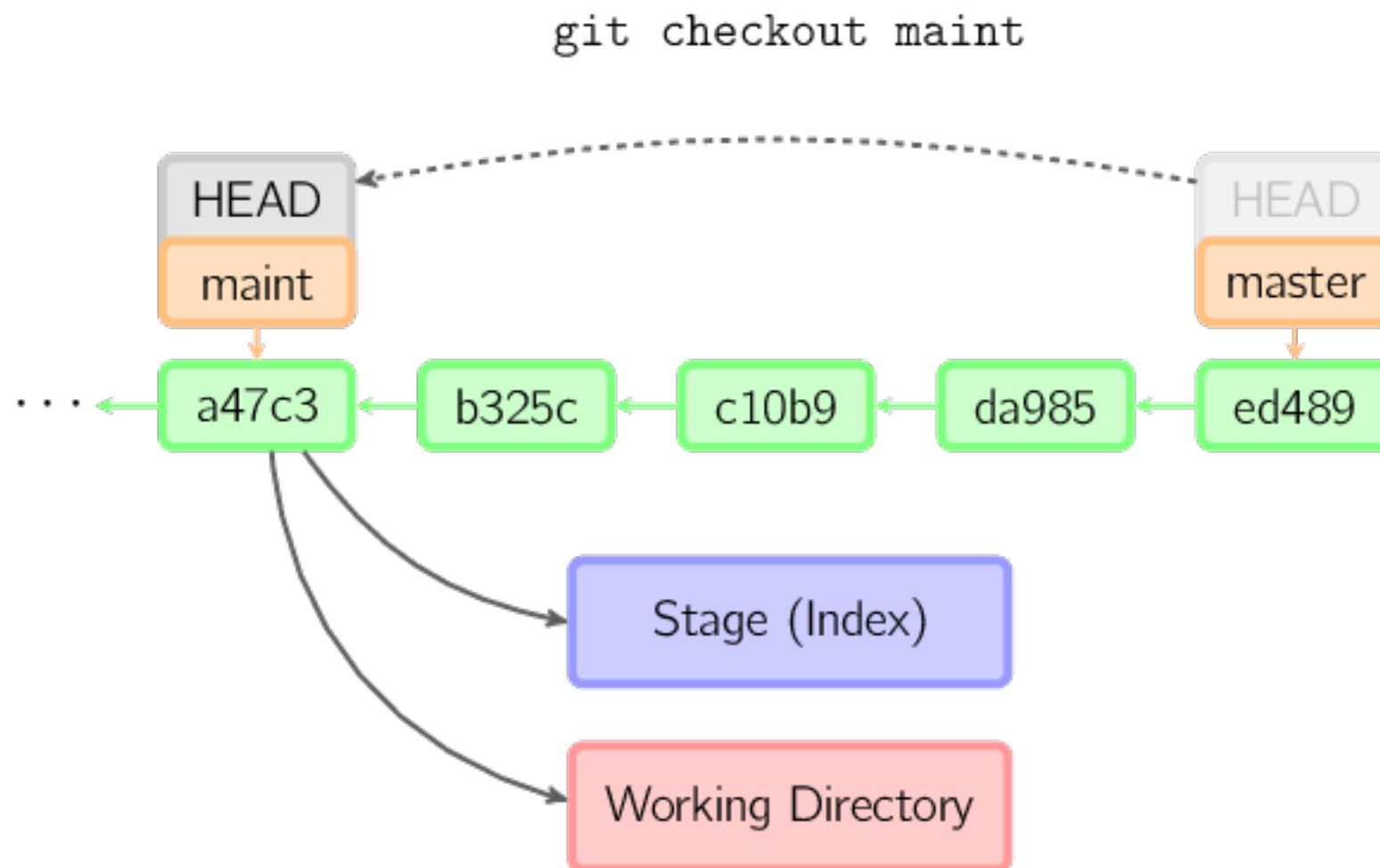
Git branch

`git branch maint`



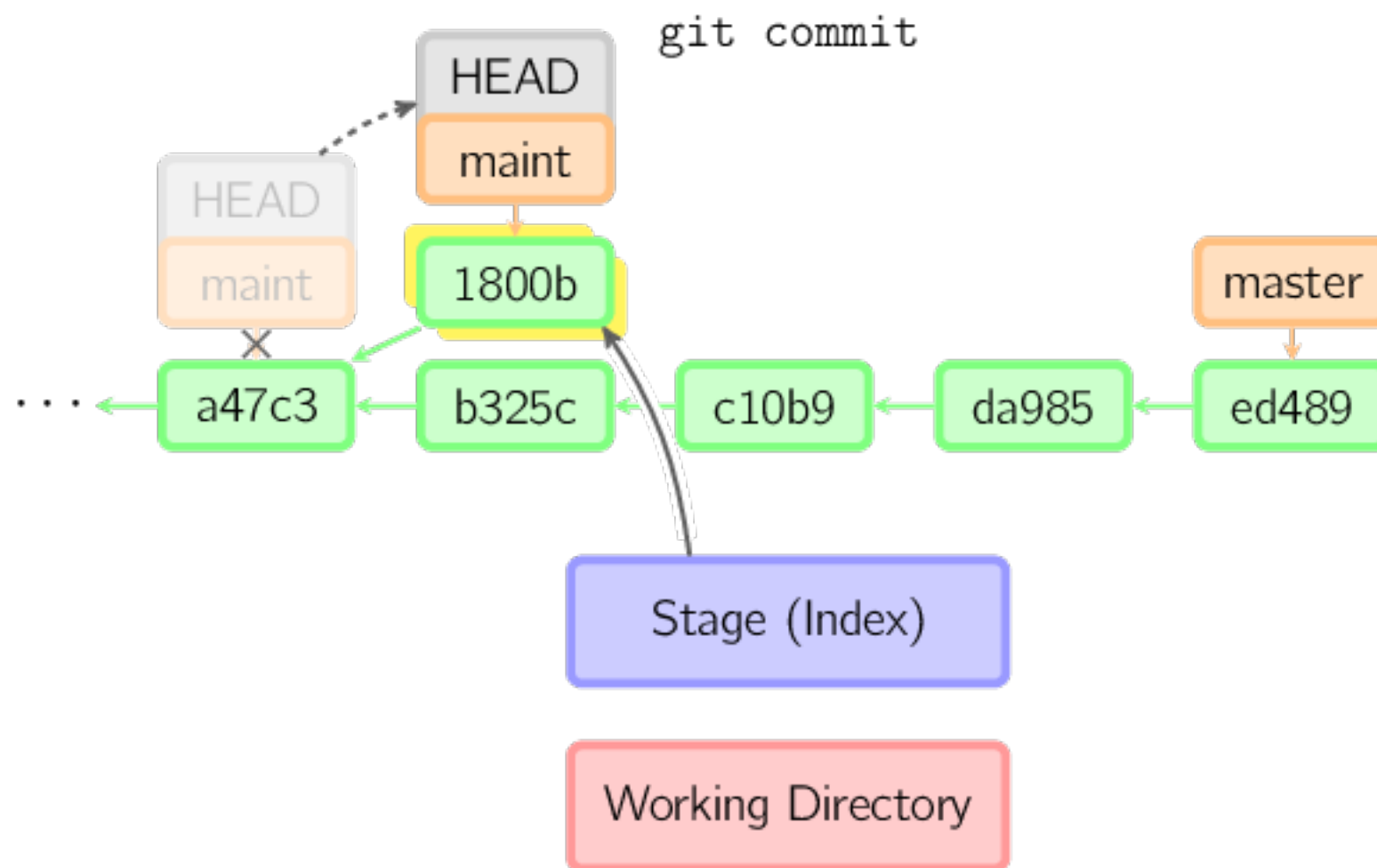
- `git branch [name]`, crea un nuovo branch
- `git branch`, lista i branch esistenti

Git checkout

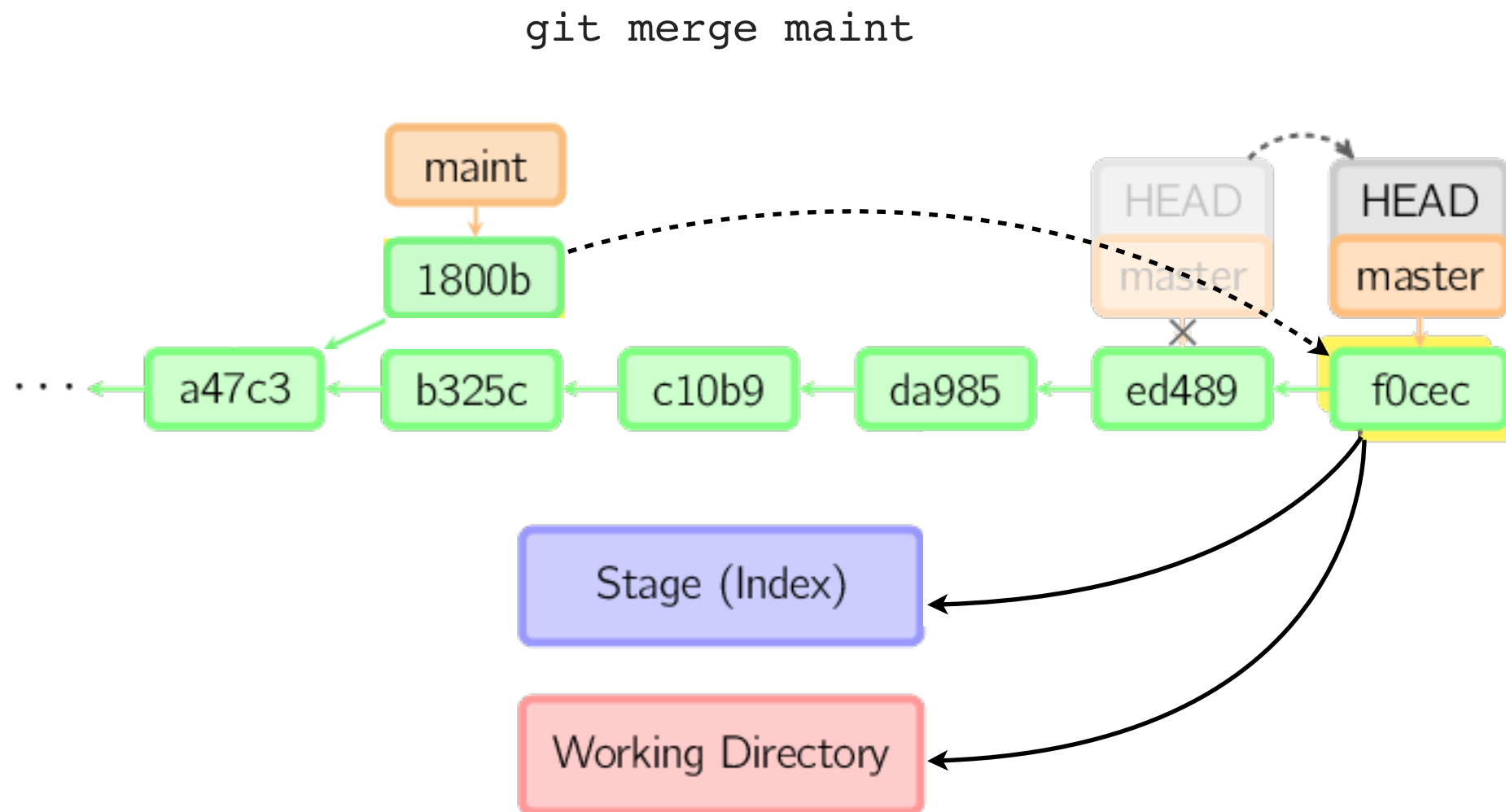


- `git checkout [name]`, sposta il puntatore HEAD al branch specificato. Cambiamo il contesto di lavoro.

Git commit (2)

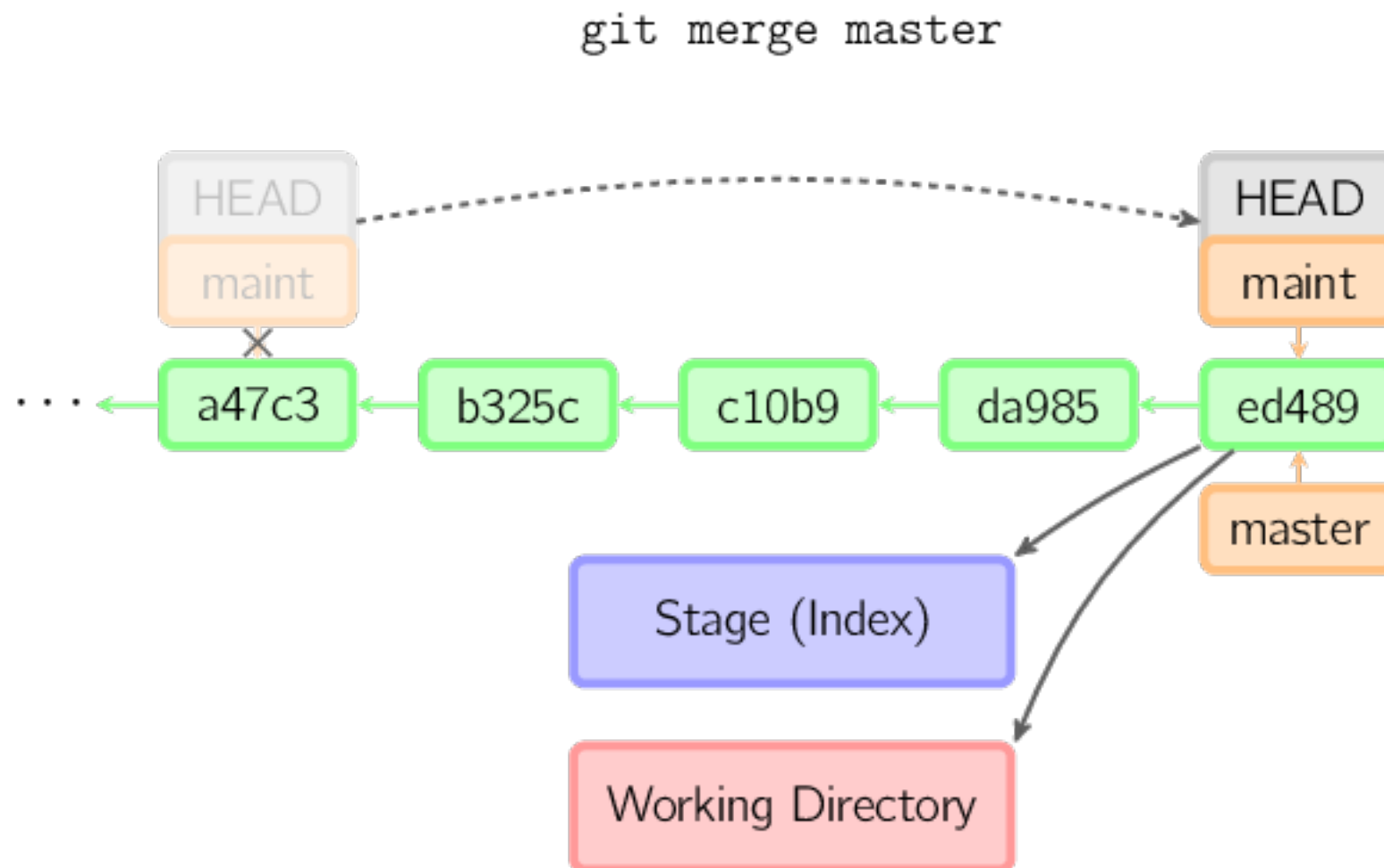


Git merge



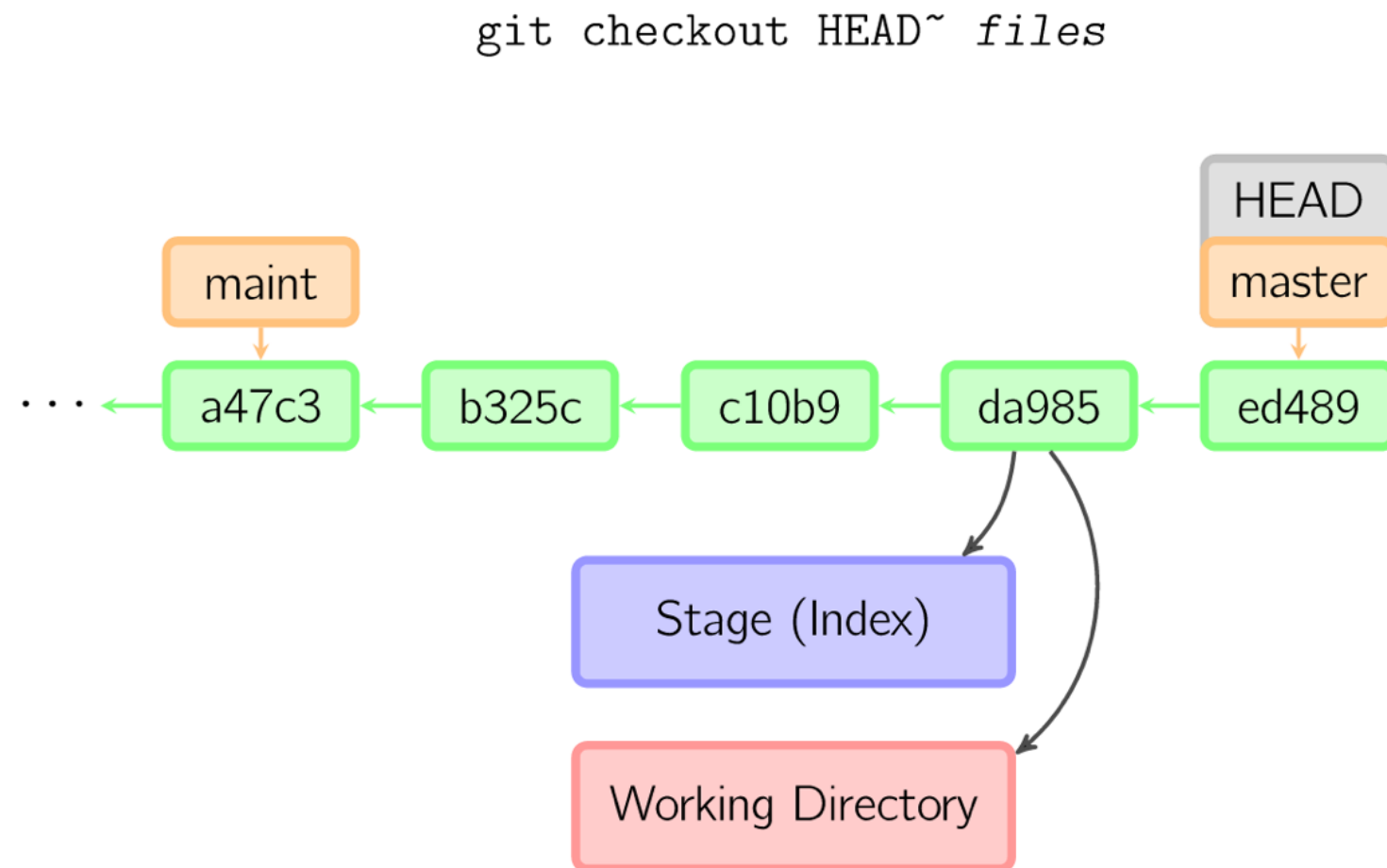
- `git merge [branch]`
 - **senza conflitti**: crea un nuovo commit che incorpora le modifiche effettuate nei commit del branch specificato .
 - **con conflitti**: risolvere i conflitti manualmente, poi `add` e `commit`

Trivial merge - fast forward



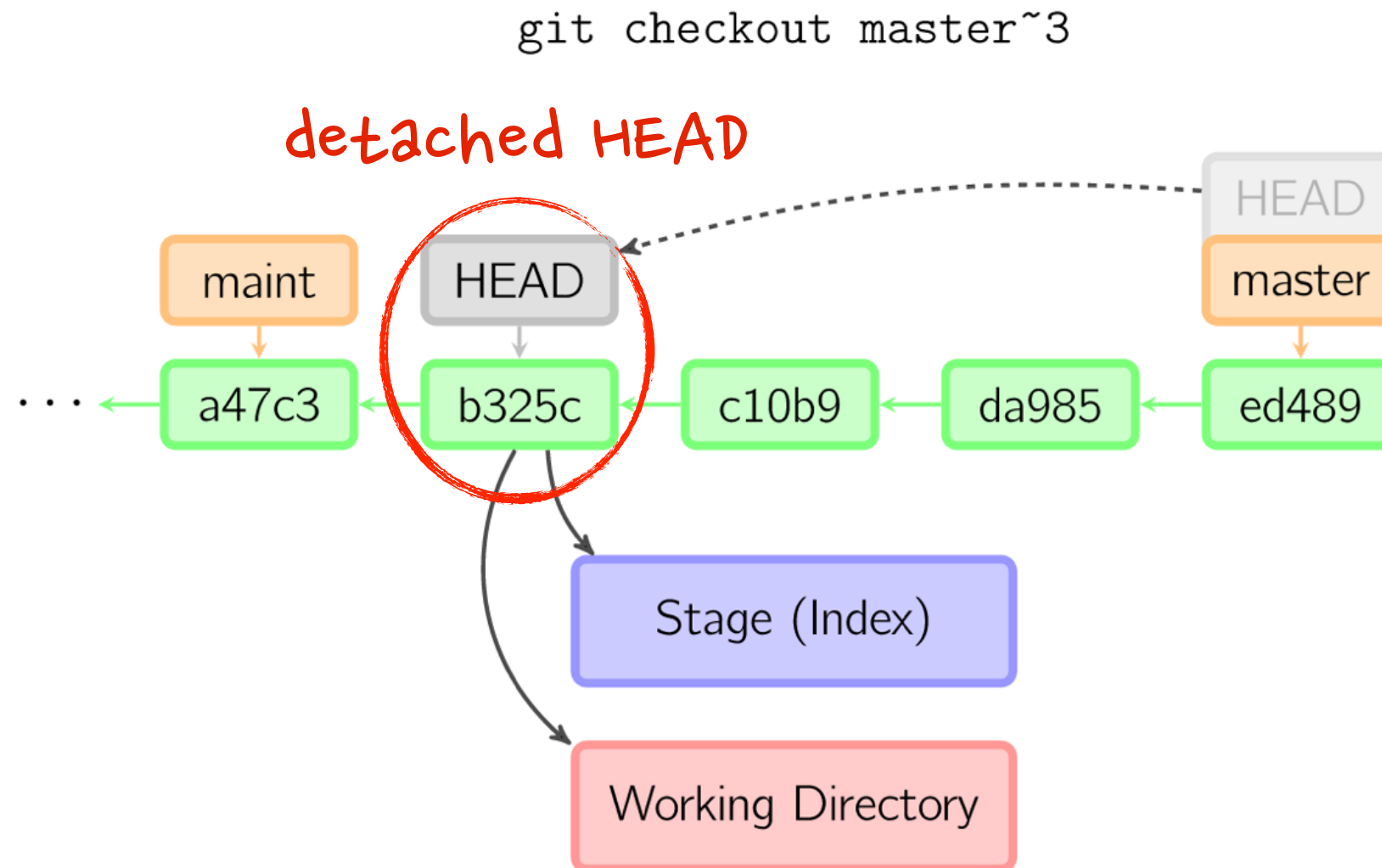
- Il commit corrente è un **predecessore** del commit target.
In questo caso viene semplicemente spostato il puntatore ed eseguito il checkout.

git checkout - riferimenti relativi



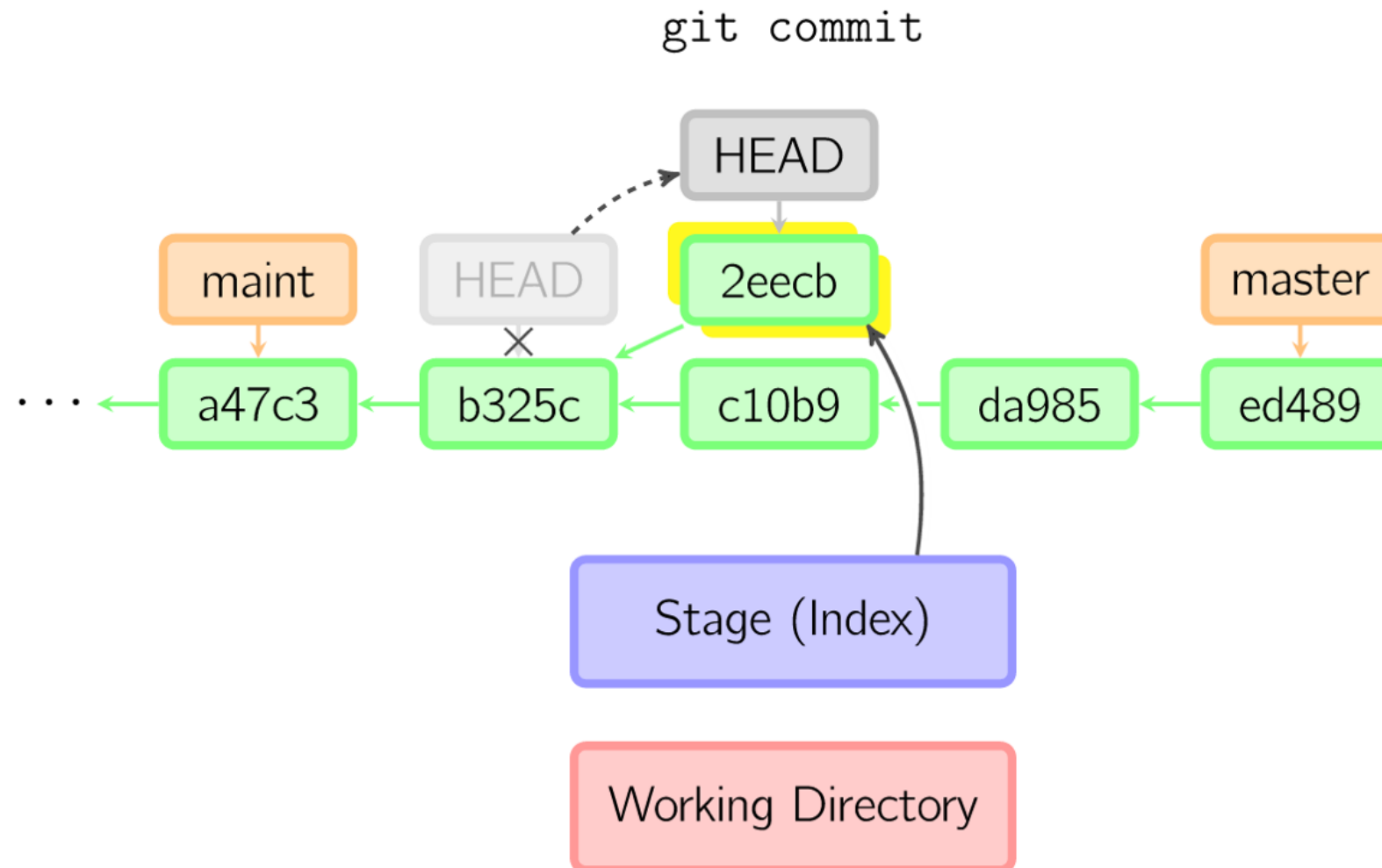
- Riferimenti relativi `HEAD~` (oppure `HEAD^`)

git checkout - riferimenti relativi



- Riferimenti relativi HEAD~X (risalgo la lista di X)
- HEAD~ = HEAD^

Commit con Detached HEAD



- In stato **detached HEAD** i commit funzionano normalmente, l'unica differenza è che il ramo su cui lavoriamo non ha un nome: **branch anonimo**
- I branch anonimi possono essere **eliminati** dal garbage collector di git

Git: sincronizzazione

- **REMOTE**

- il comando **remote** creare, vedere e cancellare collegamenti tra repository locale e repository remoti
 - listare i collegamenti con repository remoti
 - `git remote -v`
 - creare un nuovo collegamento remoto
 - `git remote add <name> <url>`
 - creare un nuovo collegamento remoto
 - `git remote rm <name>`



Git: sincronizzazione

- **PULL**

- esegue il **merge** con la versione remota del ramo specificato
 - `git pull <remote> <branch>`

- **PUSH**

- **trasferimento** di commit locali da repository locale a repository remoto
 - `git push <remote> <branch>`
- non è consentito il push se il risultato non è un merge di tipo fast-forward, in questo caso occorre eseguire prima il pull, oppure forzare il push:
 - `git push --force <remote> <branch>`

Riferimenti

- <https://www.atlassian.com/git/tutorials/>
- <http://learngitbranching.js.org/>