Università di Catania

Uni ct MATEMATICA E INFORMATICA

Git & Maven

Alessandro Midolo, PhD

Dipartimento di Matematica e Informatica

Università di Catania

alessandro.midolo@unict.it

https://www.dmi.unict.it/amidolo/

A.A. 2024/2025

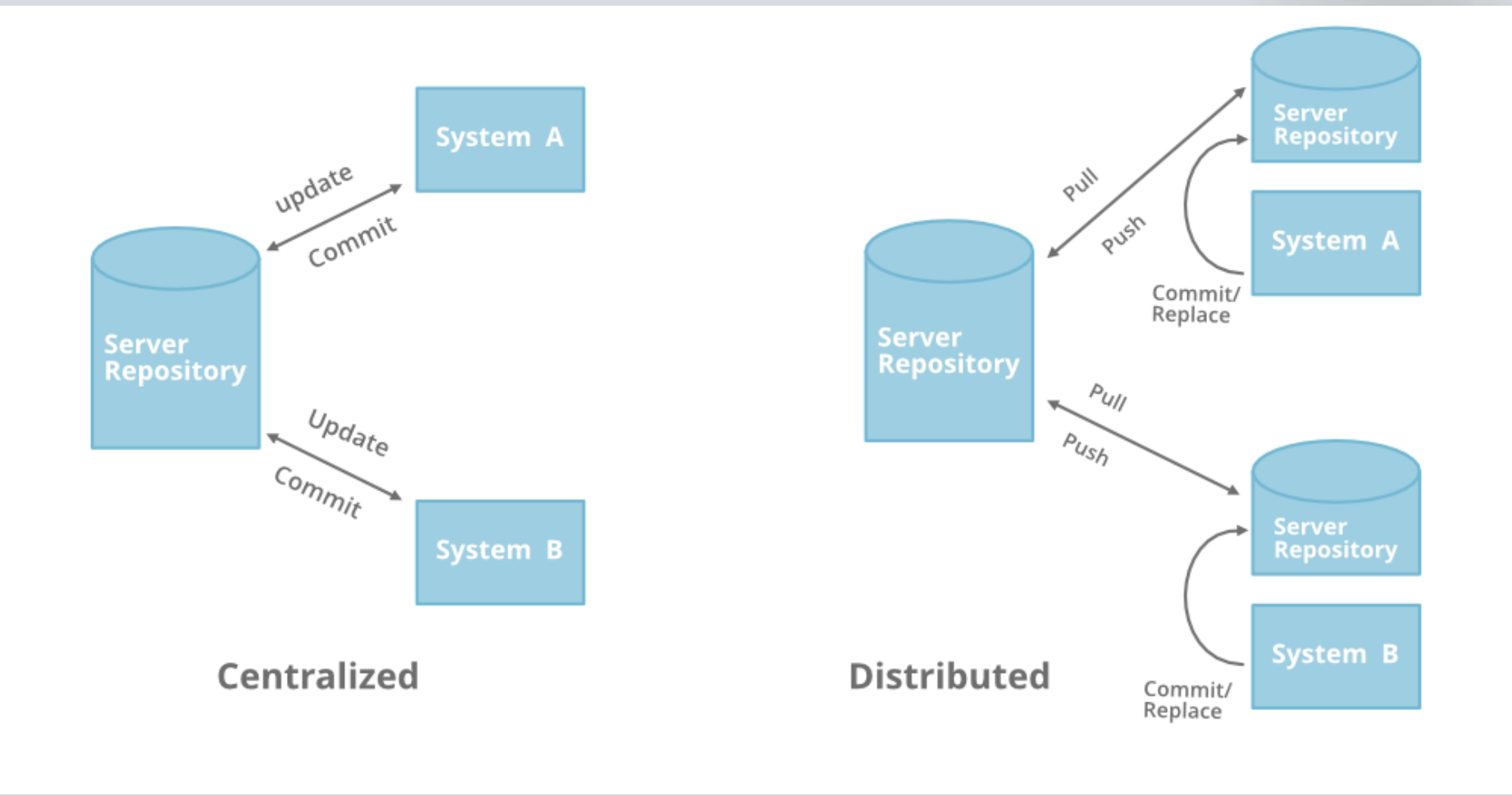
Version Control System (VSC)

- Permette di gestire e tenere traccia delle **modifiche** apportate al **codice sorgente** di un progetto nel tempo. Garantisce:
  - Code Versioning
  - Collaborazione tra sviluppatori
  - Tracciamento delle modifiche al codice
  - Creazione e gestione di Branches

GIT: A Distributed VCS (DVCS)

- Modifiche archiviate come patch e file archiviati nel **Kernel Linux** (1991)
- Il progetto Linux Kernel utilizza un DVCS proprietario, **BitKeeper** (2002)
- Linux Torval e la community di Linux creano il loro DVCS: **GIT** (2005)

What is a DVCS?



Source: https://www.geeksforgeeks.org/centralized-vs-distributed-version-control-which-one-should-we-choose/

Caratteristiche DCVS

- Ogni singolo sviluppatore o client ha il proprio «**server**»
- Si può lavorare «offline», dal momento che ognuno ha una **copia** del **repository** localmente
- Non è necessario affidarsi a un **server centrale**
- Questo favorisce la **produttività** e la **condivisione**

Obiettivi di GIT

- Supporto basato sullo sviluppo **non-lineare**
  - Supporta migliaglia di "**branches**" paralleli
  - La suddivisone in branch (branching) è facilitato
- Interamente **Distribuito**
- Può essere usato localmente e **condiviso** in remote con altri
- L'intera cronologia e informazioni sul codice sono salvati in **locale**
- La maggior parte delle funzioni sono **offline**
  - No latenza
  - Libertà di sperimentare
- Efficiente e gestibile per **grossi progetti**

Git Install?

- **Linux**
  - sudo dnf install git-all (Fedora, RHEL, CentOS)
  - sudo apt install git-all (Debian-based, e.g. Ubuntu)
- **MacOS**
  - brew install git
  - sudo port install git
- **Windows**
  - winget install --id Git.Git -e --source winget
  - TortoiseGit (https://tortoisegit.org)

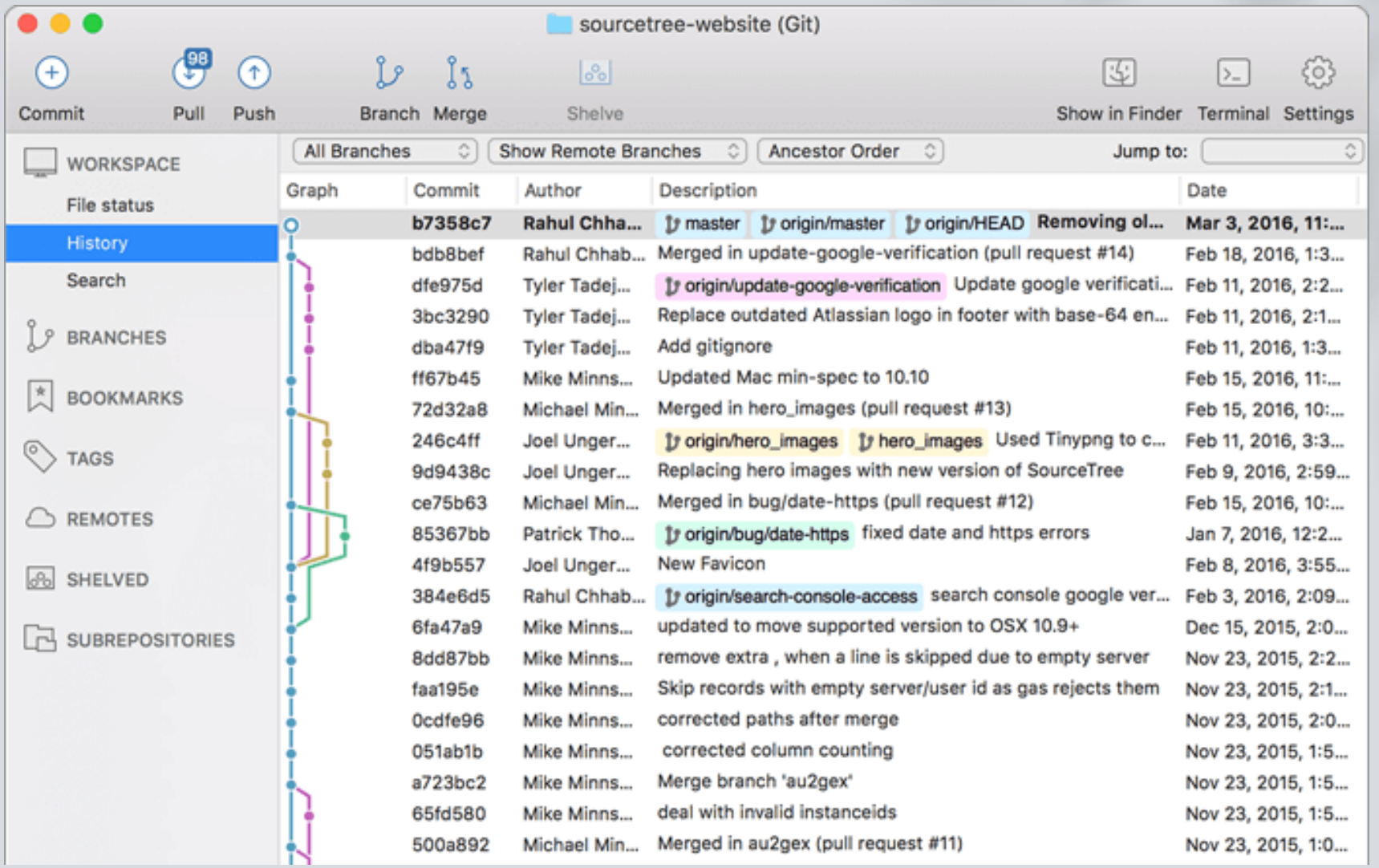
Source: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

First Steps..

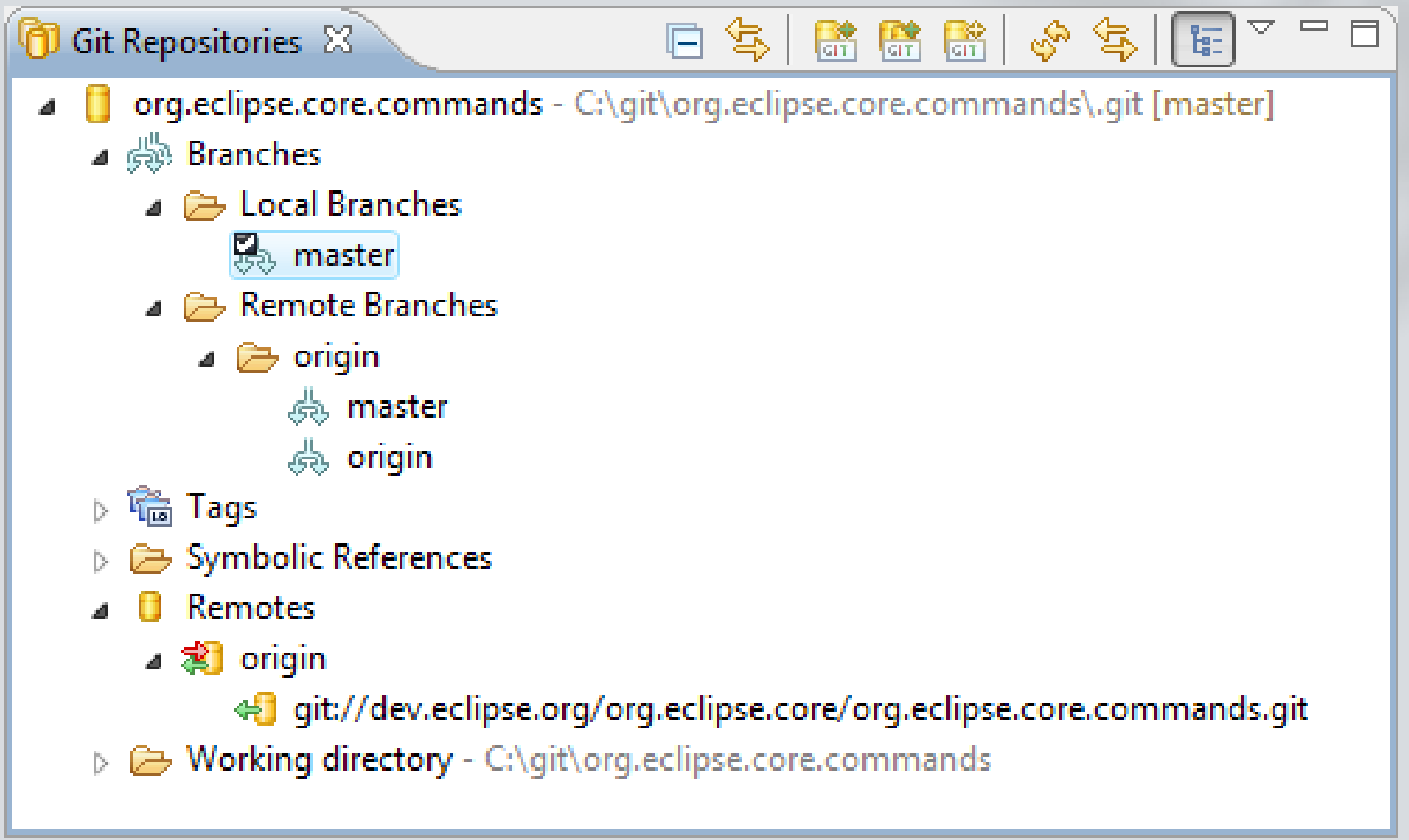
- git config --list --show-origin
- git config --global –edit
- git config --global user.name ”Alessandro Midolo”
- git config --global user.email alessandro.midolo@unict.it
- git config user.name

Atlassian

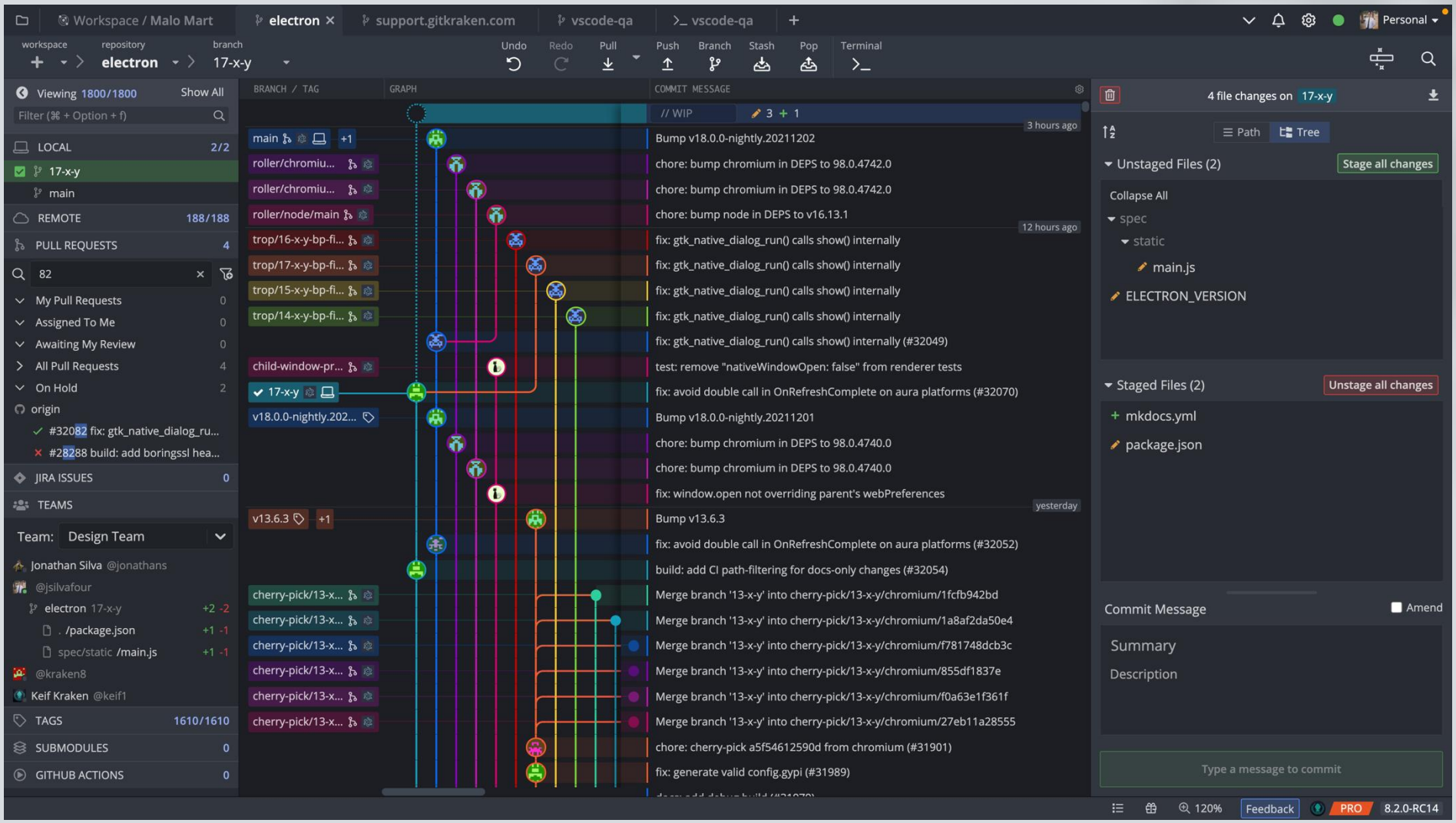
SourceTree




EGit


Git Repositories
org.eclipse.core.commands - C:\git\org.eclipse.core.commands\.git [master]
Branches
Local Branches
master
Remote Branches
origin
master
origin
Tags
Symbolic References
Remotes
origin
git://dev.eclipse.org/org.eclipse.core/org.eclipse.core.commands.git
Working directory - C:\git\org.eclipse.core.commands

GitKraken

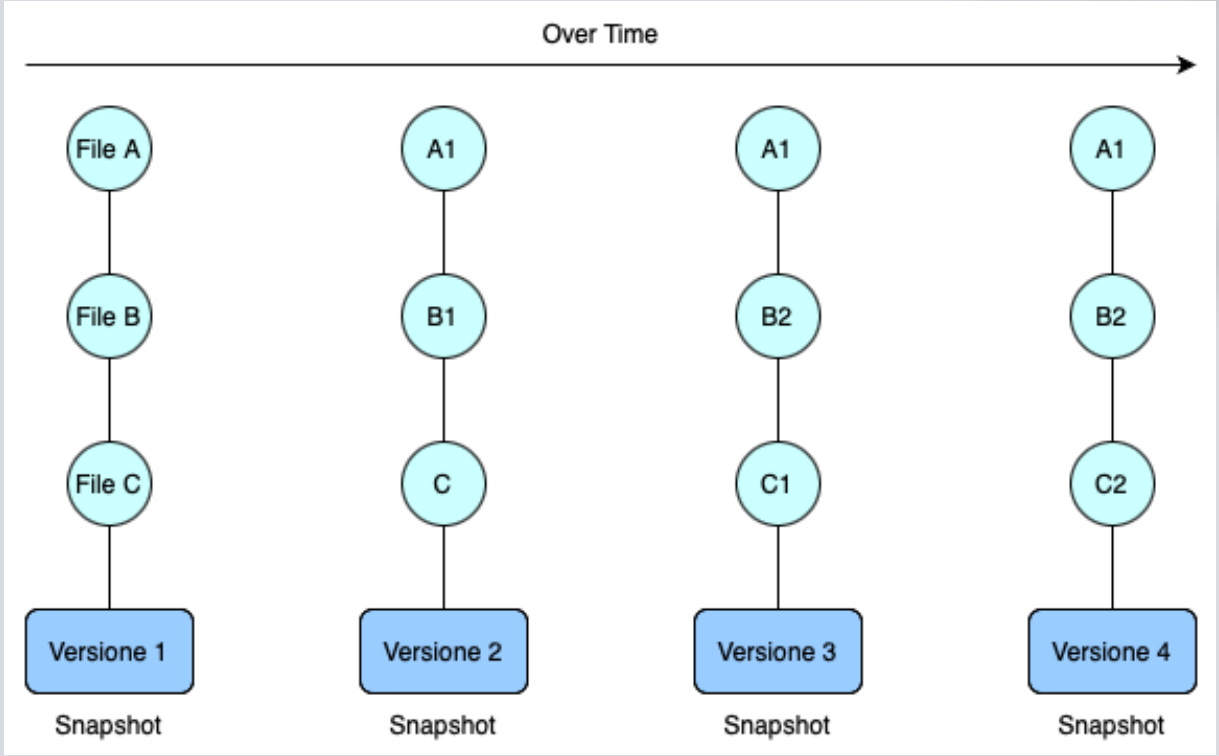

Workspace / Malo Mart
electron
support.gitkraken.com
vscode-qa
vscode-qa
Personal
workspace
repository
branch
electron
17-x-y
Undo
Redo
Pull
Push
Branch
Stash
Pop
Terminal
Viewing 1800/1800
Show All
Filter (⌘ + Option + f)
LOCAL 2/2
17-x-y
main
REMOTE 188/188
PULL REQUESTS 4
82
My Pull Requests 0
Assigned To Me 0
Awaiting My Review 0
All Pull Requests 4
On Hold 2
origin
#32082 fix: gtk_native_dialog_ru...
#28288 build: add boringssl hea...
JIRA ISSUES 0
TEAMS
Team: Design Team
Jonathan Silva @jonathans
@jsilvafour
electron 17-x-y +2 -2
. /package.json +1 -1
spec/static /main.js +1 -1
@kraken8
Keif Kraken @keif1
TAGS 1610/1610
SUBMODULES 0
GITHUB ACTIONS 0
BRANCH / TAG
GRAPH
COMMIT MESSAGE
// WIP 3 + 1
3 hours ago
main +1
Bump v18.0.0-nightly.20211202
roller/chromiu...
chore: bump chromium in DEPS to 98.0.4742.0
roller/chromiu...
chore: bump chromium in DEPS to 98.0.4742.0
roller/node/main
chore: bump node in DEPS to v16.13.1
12 hours ago
trop/16-x-y-bp-fi...
fix: gtk_native_dialog_run() calls show() internally
trop/17-x-y-bp-fi...
fix: gtk_native_dialog_run() calls show() internally
trop/15-x-y-bp-fi...
fix: gtk_native_dialog_run() calls show() internally
trop/14-x-y-bp-fi...
fix: gtk_native_dialog_run() calls show() internally
fix: gtk_native_dialog_run() calls show() internally (#32049)
child-window-pr...
test: remove "nativeWindowOpen: false" from renderer tests
17-x-y
fix: avoid double call in OnRefreshComplete on aura platforms (#32070)
v18.0.0-nightly.202...
Bump v18.0.0-nightly.20211201
chore: bump chromium in DEPS to 98.0.4740.0
chore: bump chromium in DEPS to 98.0.4740.0
fix: window.open not overriding parent's webPreferences
yesterday
v13.6.3 +1
Bump v13.6.3
fix: avoid double call in OnRefreshComplete on aura platforms (#32052)
build: add CI path-filtering for docs-only changes (#32054)
cherry-pick/13-x...
Merge branch '13-x-y' into cherry-pick/13-x-y/chromium/1fcfb942bd
cherry-pick/13-x...
Merge branch '13-x-y' into cherry-pick/13-x-y/chromium/1a8af2da50e4
cherry-pick/13-x...
Merge branch '13-x-y' into cherry-pick/13-x-y/chromium/f781748dcb3c
cherry-pick/13-x...
Merge branch '13-x-y' into cherry-pick/13-x-y/chromium/855df1837e
cherry-pick/13-x...
Merge branch '13-x-y' into cherry-pick/13-x-y/chromium/f0a63e1f361f
cherry-pick/13-x...
Merge branch '13-x-y' into cherry-pick/13-x-y/chromium/27eb11a28555
chore: cherry-pick a5f54612590d from chromium (#31901)
fix: generate valid config.gypi (#31989)
4 file changes on 17-x-y
Path
Tree
Unstaged Files (2)
Stage all changes
Collapse All
spec
static
main.js
ELECTRON_VERSION
Staged Files (2)
Unstage all changes
mkdocs.yml
package.json
Commit Message
Amend
Summary
Description
Type a message to commit
120%
Feedback
PRO
8.2.0-RC14

Commit

- Si crea un'istantanea (**snapshot**) del progetto in un determinato momento, includendo i cambiamenti effettuati nell'ultima versione
- Un commit è strutturato da:
  - **Autore**,(nome, email, data/orario) di chi ha fatto il commit
  - **Committer**,chi ha effettivamente caricato il commit nel repository
  - **Messaggio**, testo usato per commentare le modifiche effettuate
  - **Lista di commits "genitori"**(parents), il commit (0, 1 o 2) che precedono immediatamente prima il commit attuale

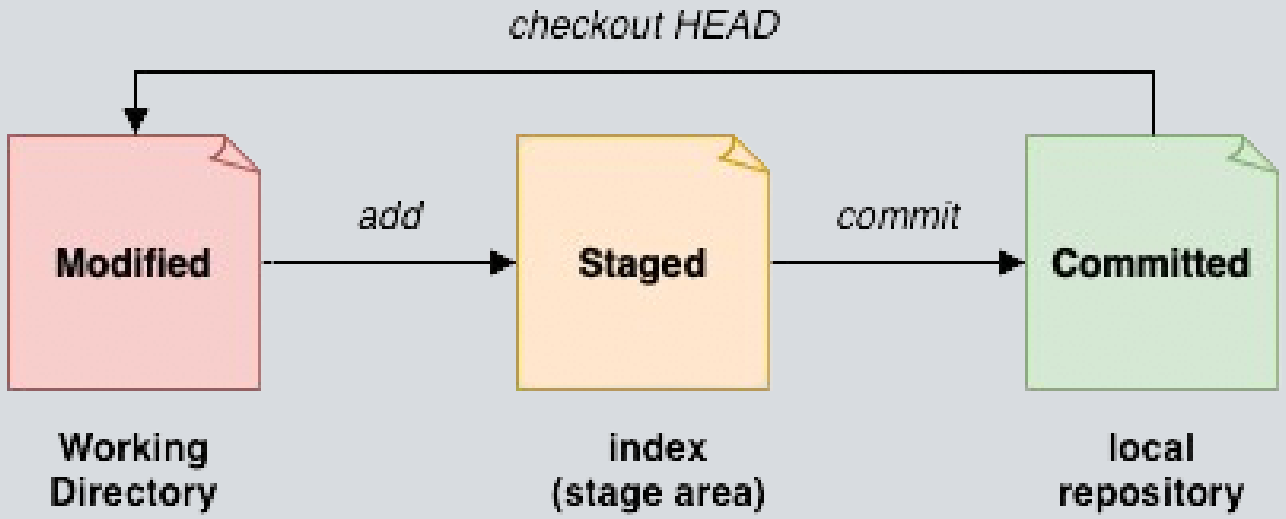
Snapshots

- All'interno di Git, i dati sono salvati come **snapshots** del filesystem, e non come una serie di **diffs**
- Ogni commit rappresenta una vera e propria istantanea dello stato di tutti i files in quell preciso momento
- Per questioni di efficienza, se un file non ha subito modifiche, questo non viene salvato di nuovo, bensì Git inserisce un collegamento all'identico file che è stato Salvato in precedenza
- Tutto questo rende le operazioni di branching più semplici

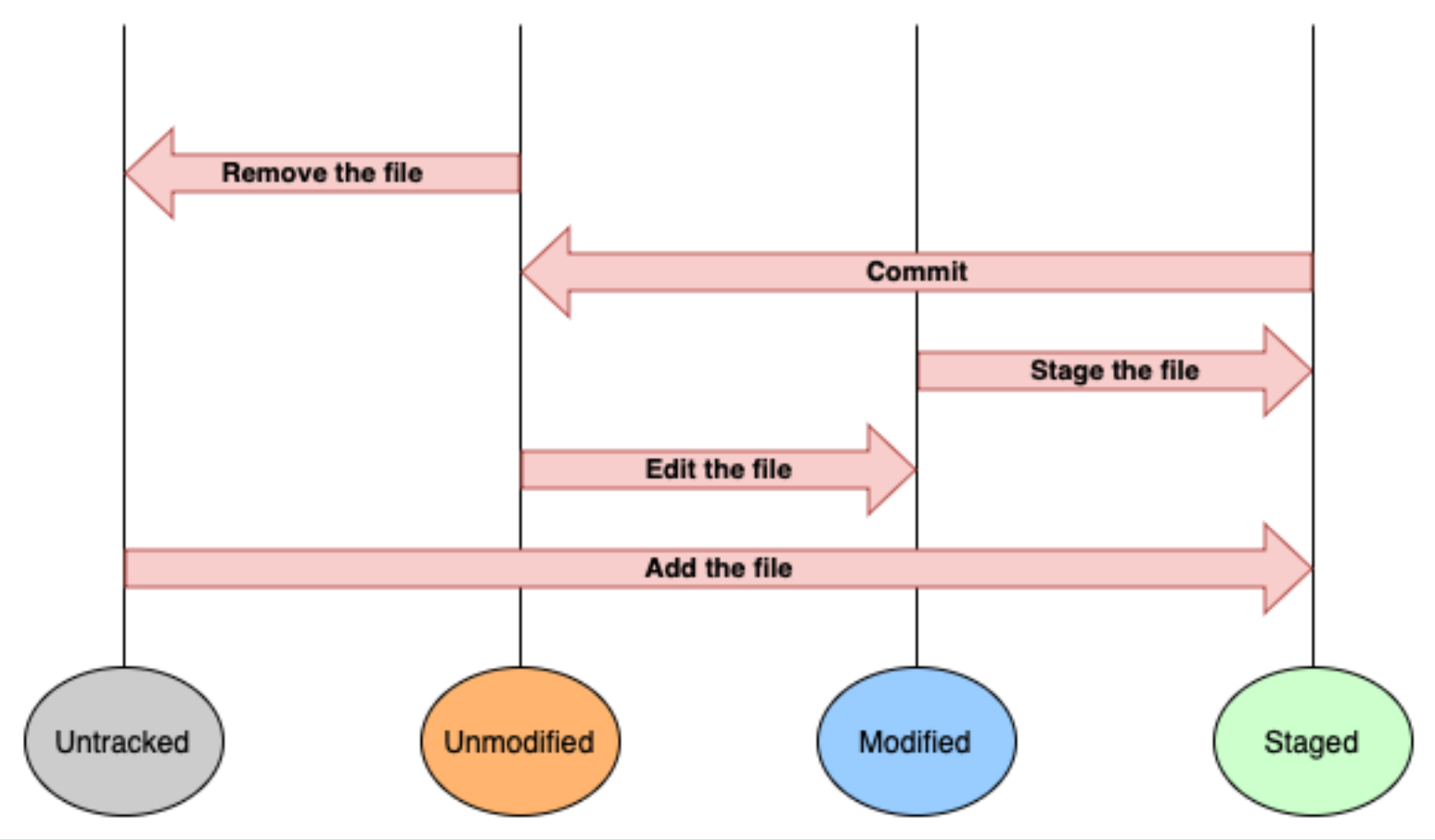


Git Workflow

- I file possono assumere tre differenti stati principali:
  - **Modified**: il file ha subito modifiche ma non è stato ancora "committed"
  - **Staged**: il file è marcato per essere inserito nel prossimo commit
  - **Committed**: il file è stato inserito all'interno del database locale



git status – lo stato dei propri files



git add & git commit

```
Alessandros-MacBook-Air:Lab_ISD midolo$ touch example.txt                    (untracked)
Alessandros-MacBook-Air:Lab_ISD midolo$ echo "file example" > example.txt    (untracked)
Alessandros-MacBook-Air:Lab_ISD midolo$ git add example.txt                  (new file staged for commit)
Alessandros-MacBook-Air:Lab_ISD midolo$ echo "added new text" > example.txt  (changes not staged for commit)
Alessandros-MacBook-Air:Lab_ISD midolo$ git add example.txt                  (new file staged for commit)
Alessandros-MacBook-Air:Lab_ISD midolo$ git commit -m "secondo commit"       (unmodified)
[main 49027c8] secondo commit
 1 file changed, 1 insertion(+)
 create mode 100644 example.txt
```

git log

```
Alessandros-MacBook-Air:Lab_ISD midolo$ git log
commit 49027c8cabc1fd55b7c9d1094692df066d93a2a6 (HEAD -> main)
Author: AleMidolo <alemidolo@gmail.com>
Date:   Tue Oct 22 18:17:05 2024 +0200

    secondo commit

commit 0d2210d26137b93db0db31a957e2c22703a4f0c9 (origin/main, origin/HEAD)
Author: Alessandro Midolo <alemidolo@gmail.com>
Date:   Tue Oct 22 16:34:52 2024 +0200

    Initial commit
```

Esempio di un Workflow

- **Modifica** dei file all'interno del proprio ambiente di lavoro
- **Seleziona** solo quelle modifiche che vuoi siano parte del prossimo commit, in modo da inserirle all'interno dell'area di **staging**
- Esegui il **commit**, quindi i file in precedenza inseriti nell'area di staging verranno permanentemente caricati nella cartella git, salvando così uno **snapshot** dello stato attuale

Alcuni consigli sui commit

- Esegui commit **frequenti**, che siano **coesi** e **chiari**
  - Non aspettare che si accumulino, in un primo momento sono salvati solo localmente
- Non perdere la **sincronia** con i tuoi colleghi/collaborator
- Esegui il commit **solo** dei file sorgenti, non i file generati o derivati
- Usa il file **.gitignore** per ignorare i files che non ti interessano

Source: https://www.conventionalcommits.org/en/v1.0.0/
https://gist.github.com/qoomon/5dfcdf8eec66a051ecd85625518cfd13

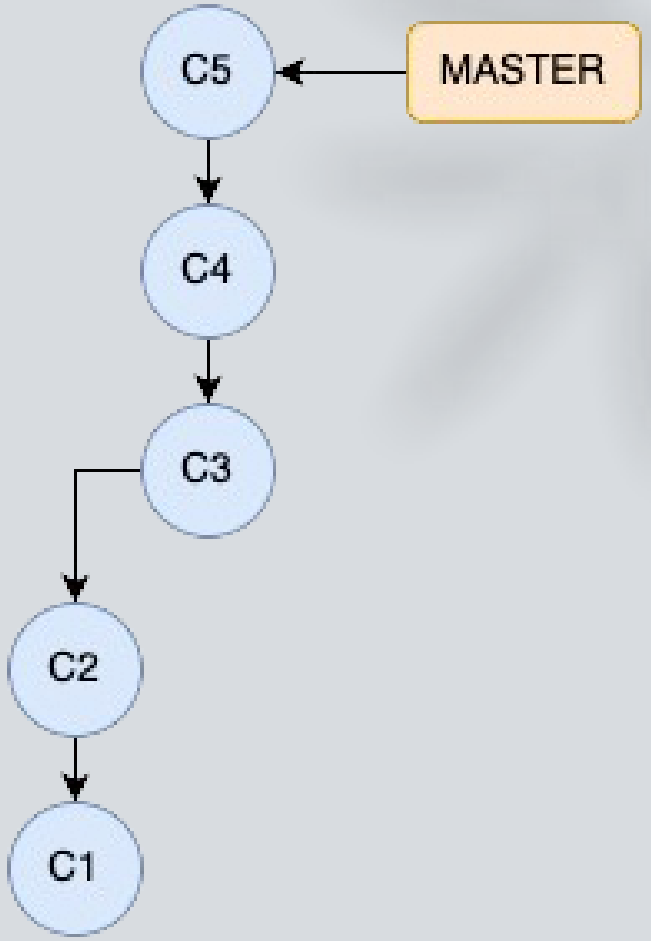
AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Branching & Merging

- Un **branch** (ramo) è una linea di sviluppo separata. Ti consente di deviare dal codice principale (spesso chiamato main o master) per lavorare su una nuova funzionalità, correggere un bug o fare esperimenti senza influenzare il progetto principale
- Il **merge** è il processo di integrazione delle modifiche di un branch in un altro. Una volta terminato il lavoro su un branch (es. feature-branch), potresti voler combinare quelle modifiche nel branch main.

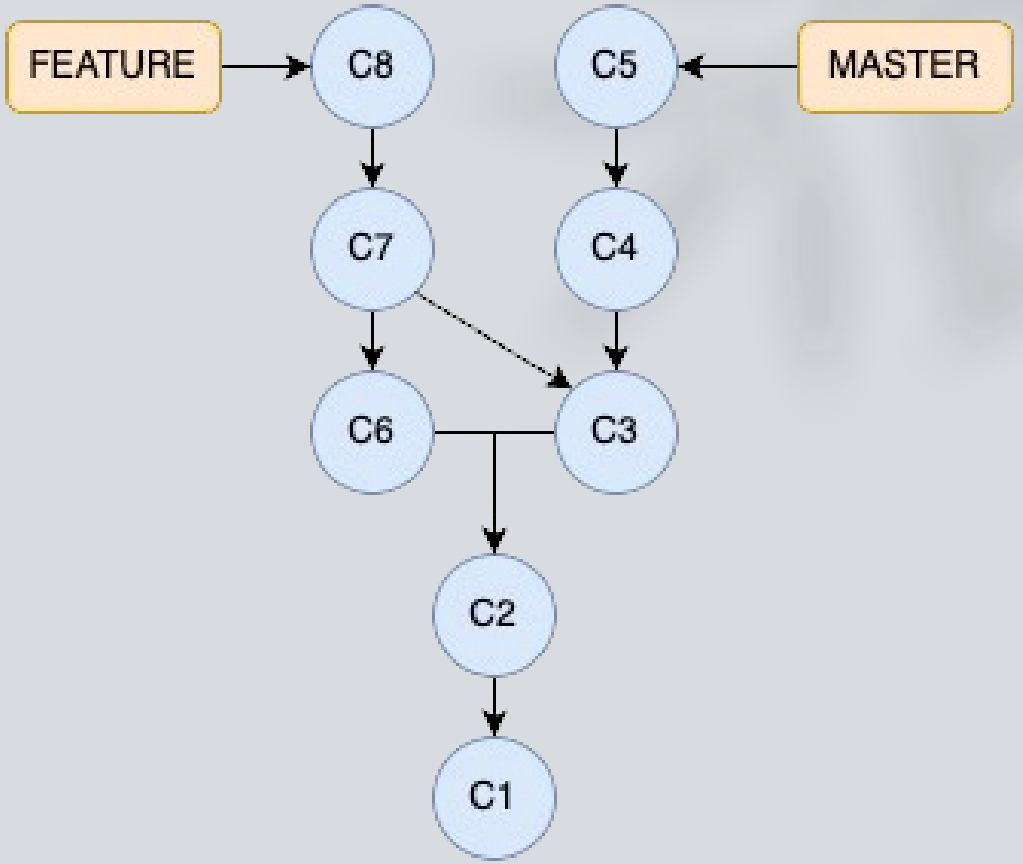
Linear History Branch

- Ogni repository inizia con un commit (root) che non ha predecessori (C1)
- Di base, ogni repository viene creato con il branch principale **master**
- Il branch punta all'ultimo commit di quel branch, questo viene chiamto "**branch tip**" (e.g. master -> C5)
- Un branch è l'insieme di tutti i commit raggiungibili dal **tip** fino alla **root** (radice) (e.g. master = {C1 ... C5}
- Un nuovo commit può essere inserito nel branch come un nodo che punta al **tip**, quindi la referenza del branch viene spostata al nuovo tip



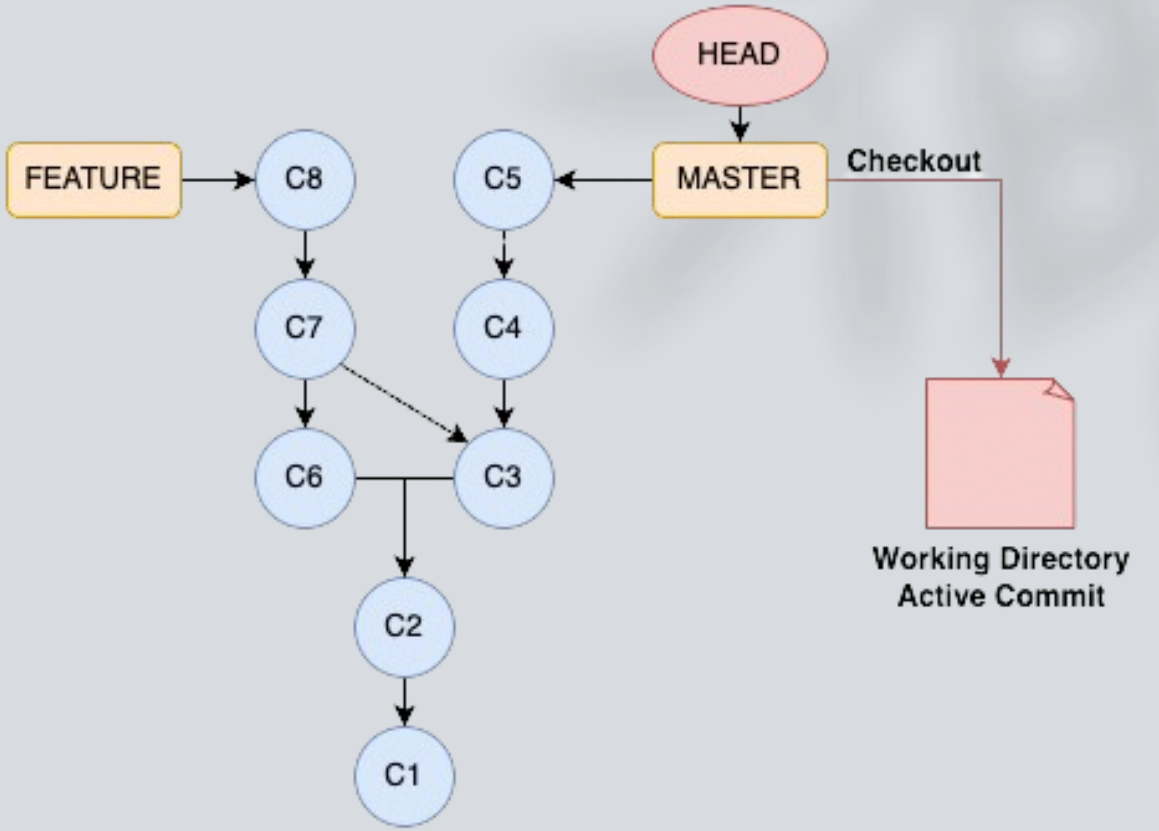
Non Linear History Branch

- Git supporta e incoraggia lo sviluppo non lineare. I Commit possono essere rappresentati come **DAG** (Directed Acyclic Graph)
- Un commit può avere
  - 0 predecessori (primo commit, root)
  - 1 predecessore (commit normale)
  - 2 predecessori (merge commit)
- Si possono avere diversi branches. In questo esempio:
  - master {C1, C2, C3, C4, C5}
  - feature {C1, C2, C3, C6, C7, C8}
- I branches possono sovrapporsi
- Le referenze non funzionano in modo **bidirezionale** e, se elimini un branch, potresti perdere l'accesso ai commit che quel branch stava puntando, a meno che non ci siano altre referenze che puntano agli stessi commit



Active Branch

- E' possibile **cambiare** branch senza perdere il lavoro svolto
- HEAD è una referenza speciale che punta al branch **attivo**
- La directory di lavoro nel proprio file system conterrà lo **snapshot** dell'ultimo **commit** attivo



git checkout

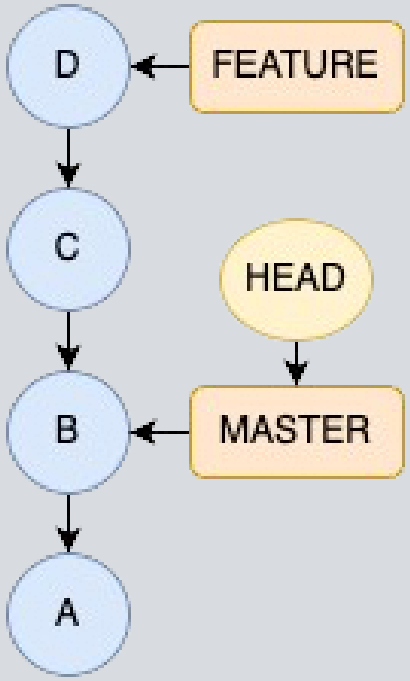
- Ha diverse funzionalità
  - **git checkout branch_name** : permette di passare da un branch all'altro all'interno del repository (e' chiaro che il branch deve esistere)
  - **git checkout –b branch_name**: permette di creare e passare direttamente al nuovo branch
  - **git checkout nomefile** / **git checkout commit-id -- nomefile**: annulla le modifiche locali su uno o più file, ripristinandoli alla versione dell'ultimo commit (o di un commit specifico)
  - **git checkout commit-id**: passa a un commit specifico per esplorarlo

git merge

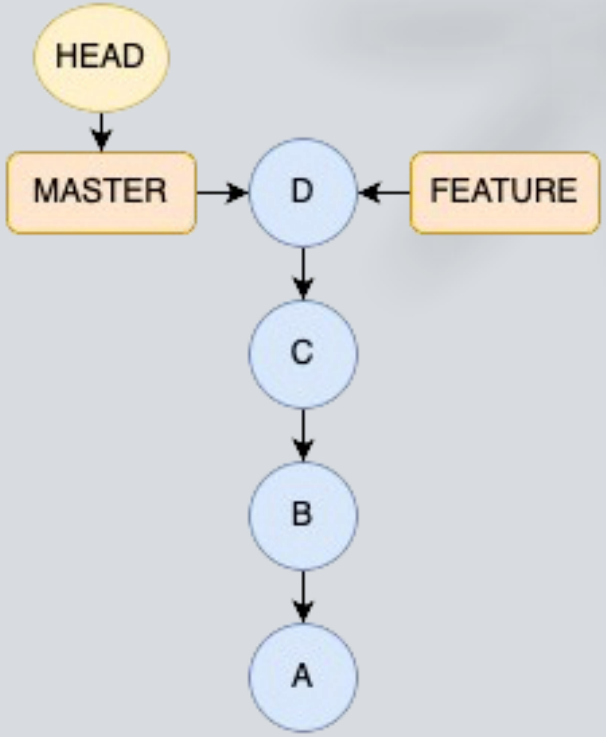
- **git merge <branch>**
- Unisce il branch sorgente con il branch attivo in quel repository
- Unisce le modifiche effettuate ai file nel branch target all'interno del branch attivo
- Può essere visto come un'operazione tra insiemi
  - Al termine dell'operazione di merge, il branch attivo dovrà contenere tutti i commits del branch sorgente

Fast Forward Merge

master = {A,B}; feature = {A, B, C, D} ==> master = {A, B, C, D}



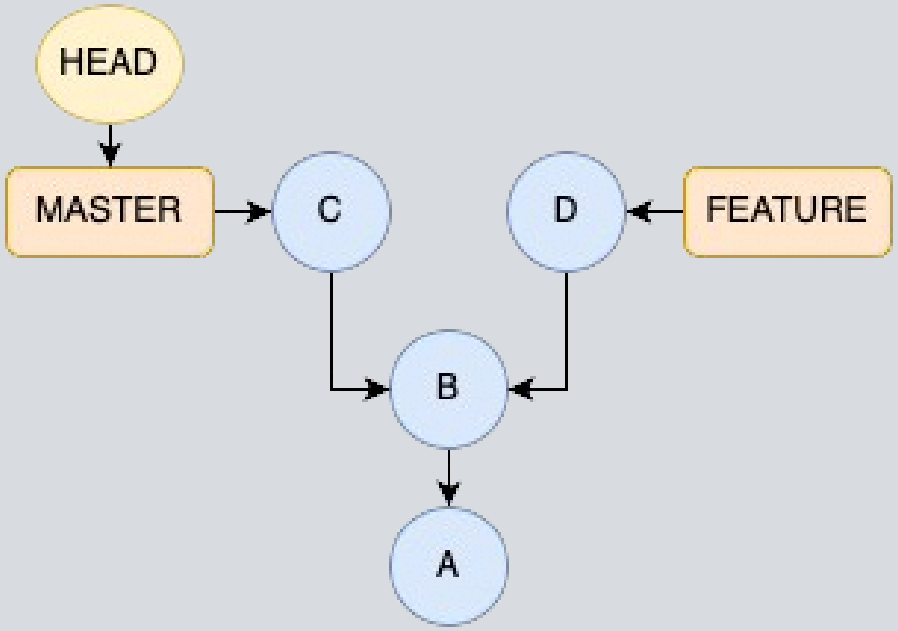
PRIMA



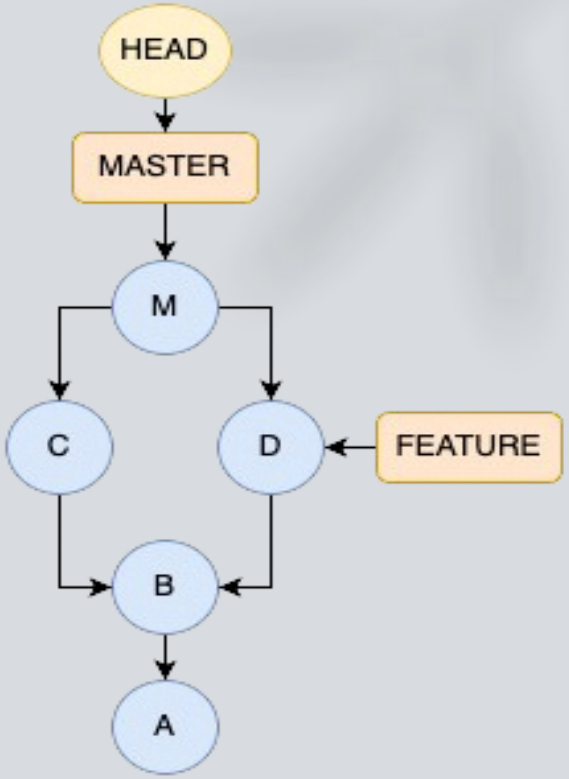
DOPO

Commit Merge

master = {A,B,C} ; feature = {A, B, D} ==> master = {A, B, C, D, M}



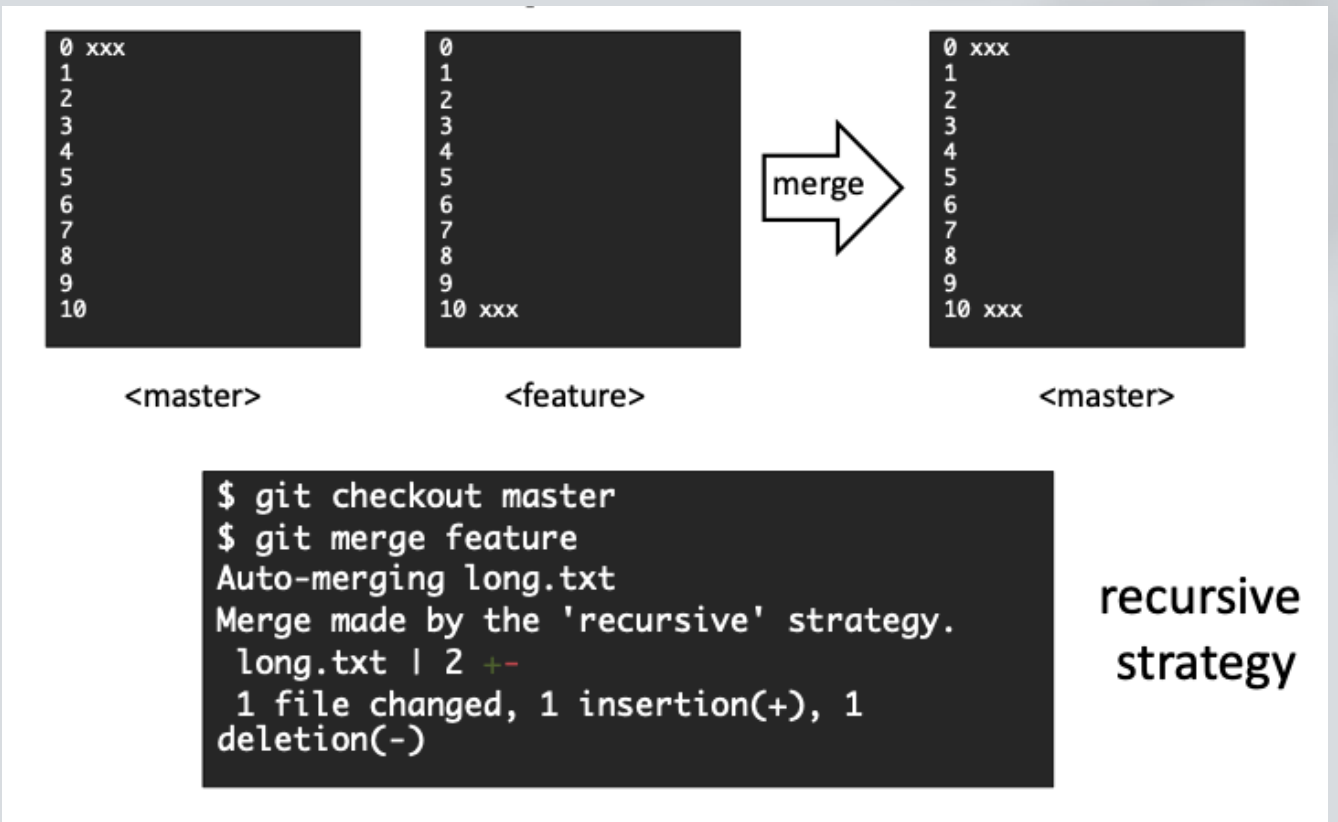
PRIMA



DOPO

Automatic Merge

- Git esegue in **automatico** il merge dei branches, cercando di unire le modifiche fatte ai files in modo da **riunire** il tutto nel branch master

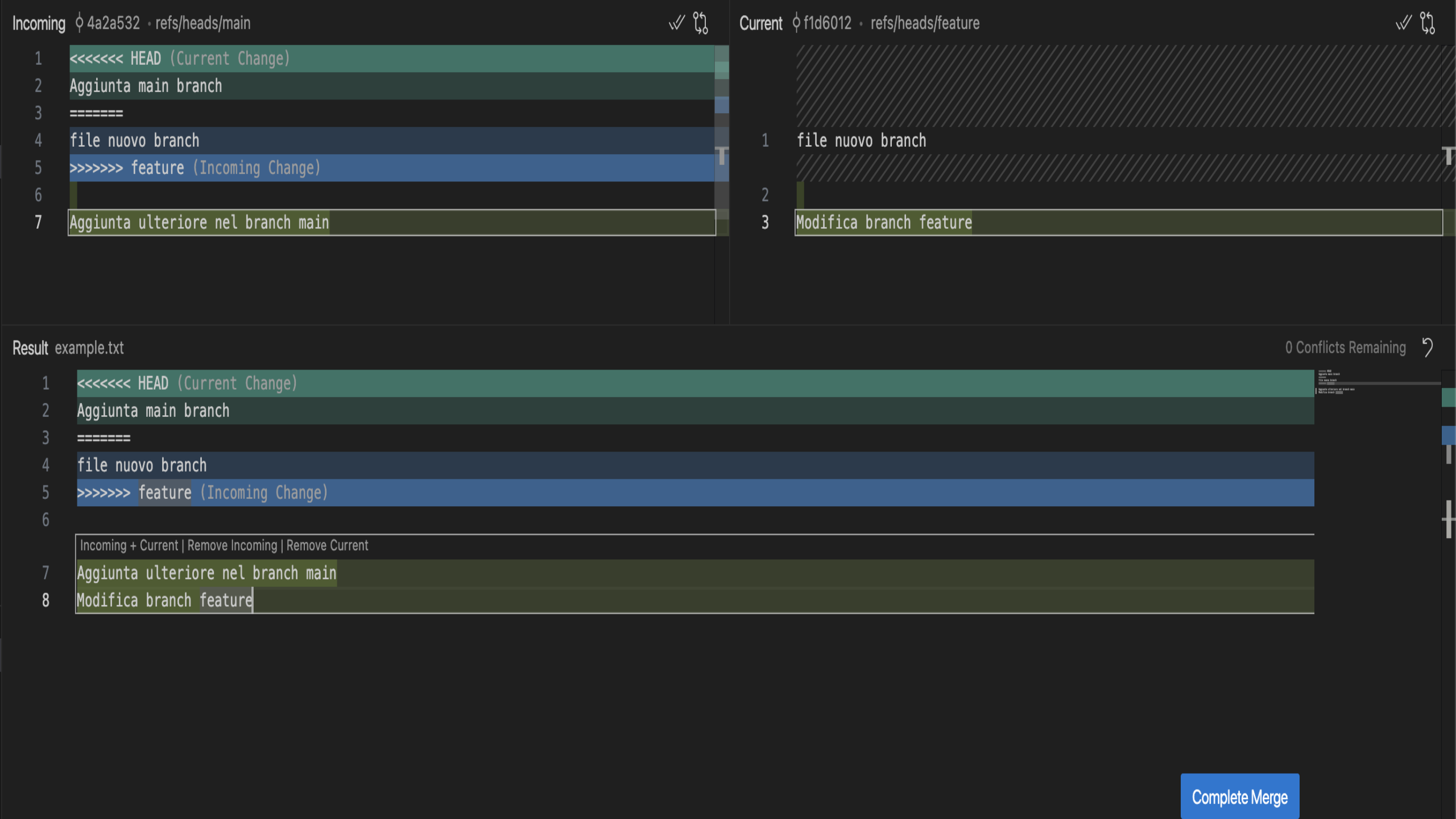


Merge Conflict

- Il merge automatico può **fallire** se Git trova modifiche effettuate all'interno dello stesso file, nella stessa posizione
- Quale delle due **modifiche** selezionare? La prima o la seconda?

```
Alessandros-MacBook-Air:Lab_ISD midolo$ git merge feature
Auto-merging example.txt
CONFLICT (content): Merge conflict in example.txt
Automatic merge failed; fix conflicts and then commit the result.
```

- Bisogna quindi guidare git nella scelta di come effettuare la modifica al file condiviso tra i due branch, così da definire una versione **comune**


Incoming 4a2a532 · refs/heads/main
<<<<<<< HEAD (Current Change)
Aggiunta main branch
=======
file nuovo branch
>>>>>>> feature (Incoming Change)
Aggiunta ulteriore nel branch main
Current f1d6012 · refs/heads/feature
file nuovo branch
Modifica branch feature
Result example.txt
0 Conflicts Remaining
<<<<<<< HEAD (Current Change)
Aggiunta main branch
=======
file nuovo branch
>>>>>>> feature (Incoming Change)
Incoming + Current | Remove Incoming | Remove Current
Aggiunta ulteriore nel branch main
Modifica branch feature
Complete Merge

SOURCE CONTROL

SOURCE CONTROL

Aggiunta main

Sync Changes 5 ↑

example.txt | 321681d7 - New merge (1 files)

example.txt

```
1  file nuovo branch
2
3  Modifica branch feature
```

```
1+ <<<<<<< HEAD (Current Change)
2+ Aggiunta main branch
3+ =======
4  file nuovo branch
5+ >>>>>>> feature (Incoming Change)
6
7+ Aggiunta ulteriore nel branch main
8  Modifica branch feature
```

SOURCE CONTRO... Auto

- New merge AleMidolo feature
- Aggiunta main AleMidolo
- Merged AleMidolo origin/main
- Conflitto aggiunto AleMidolo
- Feature commit 2 AleMidolo
- Nuovo commit ... origin/feature
- secondo commit AleMidolo
- Initial commit Alessandro Midolo

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

bash

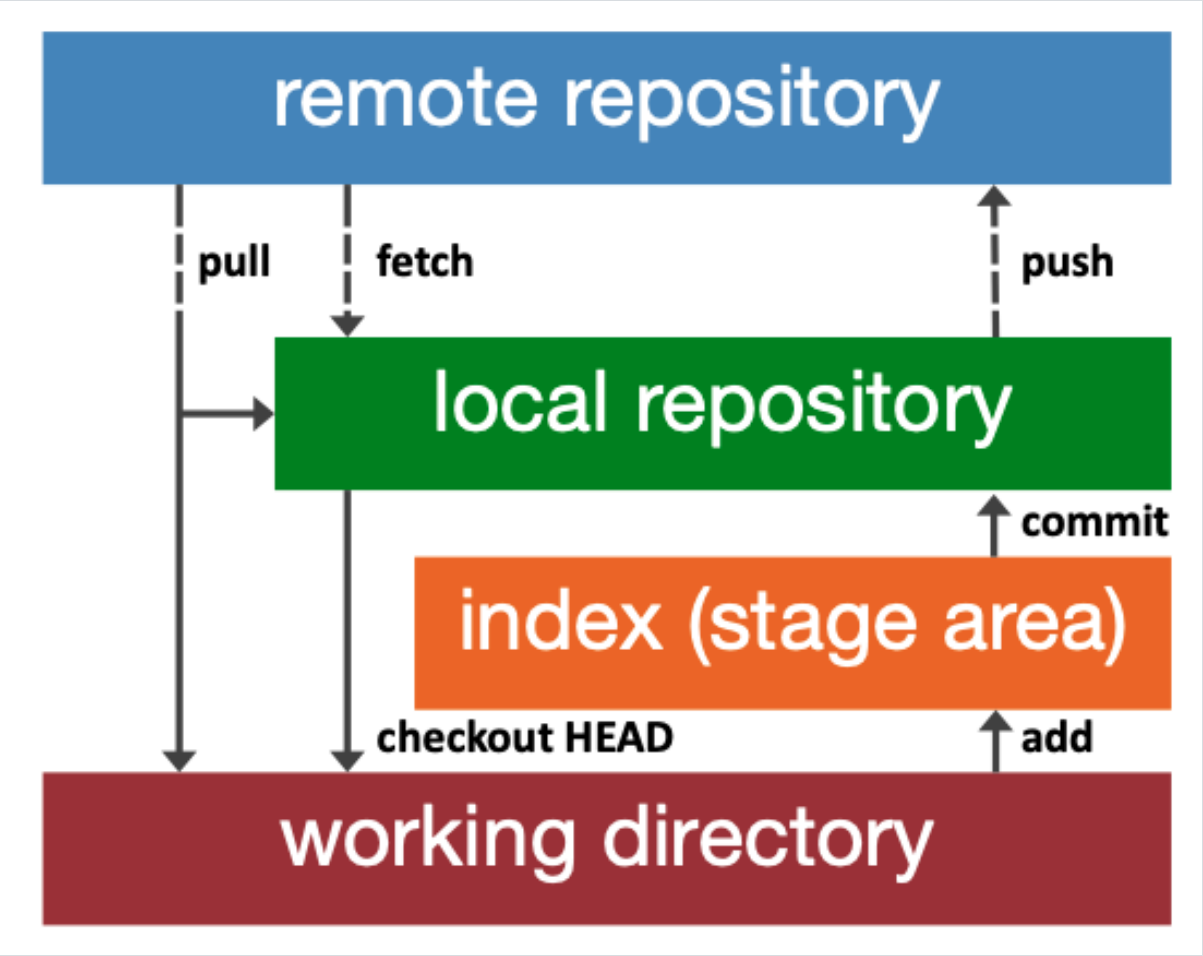
```
bash: status: command not found
Alessandros-MacBook-Air:Lab_ISD midolo$ git status
On branch feature
Your branch is ahead of 'origin/feature' by 1 commit.
  (use "git push" to publish your local commits)

All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
        modified:   example.txt

Alessandros-MacBook-Air:Lab_ISD midolo$ git commit -m "New merge"
[feature 321681d] New merge
Alessandros-MacBook-Air:Lab_ISD midolo$
```

Remote Repository



GitHub

GitLab

Bitbucket

Comandi Remote Repositories

- **git clone url**: crea una copia completa di un repository remoto sul tuo filesystem, compresi files, commits e branches
- **git pull**: recupera le modifiche dal repository remoto e le unisce automaticamete al branch corrente sul tuo computer
- **git push**: carica le modifiche dal tuo branch locale al repository remoto, aggiornando il branch corrispondente
- **git fetch**: recupera le modifhce dal repository remoto, ma **non** le unisce automatiamente al branch corrente. Scarica solo gli aggiornamenti dai branch remoti e li rende disponibili per l'unione manuale

Source: https://education.github.com/git-cheat-sheet-education.pdf

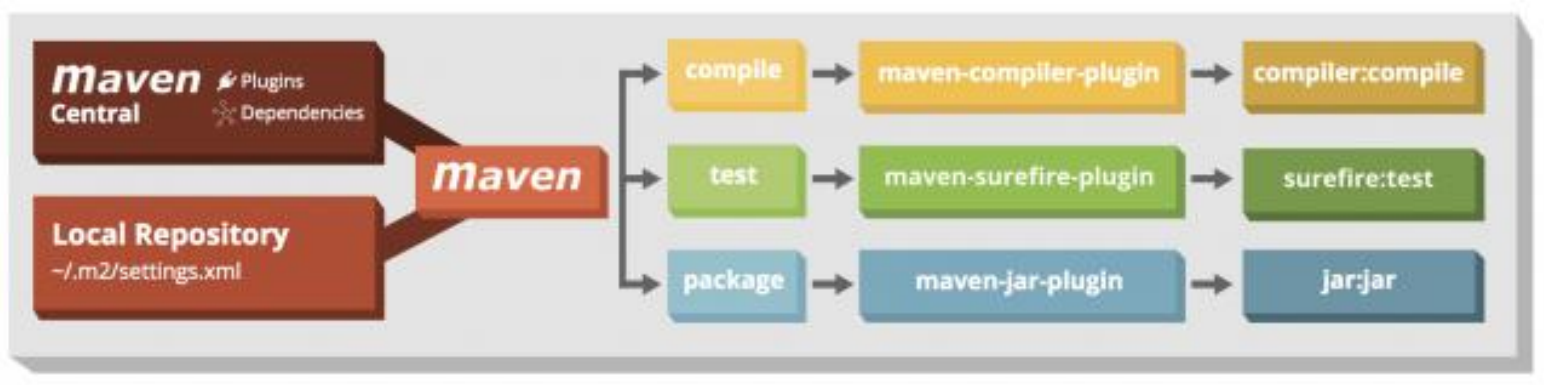
MAVEN

Gestione di un Progetto Java

- Un progetto Java può contenere delle **dipendenze** al suo interno (vere e proprie librerie in formato .jar). Questo comporta:
  - i .jar delle librerie devono essere **reperiti** e **scariati** in locale
  - il **CLASSPATH** deve essere modificato
  - i moduli del progetto possono essere **dipendenti** tra di loro e devono essere **compilati** nel giusto ordine
  - la **lista delle dipendenze** dovrebbe essere condivisa assieme al codice sorgente
- La creazione e la gestione di un progetto dovrebbe essere indipendente dall'IDE utilizzato
- La compilazione di un'eseguibile (artifact) richiede varis fasi da **automatizzare** quanti più possibile
- Aggiunta facilitata di strumenti utili di **reportistica** per facilitare i processi di gestione
  - Generazione automatica della documentazione
  - Misura della copertura del codice da parte dei test
- Dovrebbe essere facile **condividere versioni aggiornate** ed eseguibili
  - Quando viene creata una nuova release, l'artifact dovrebbe essere caricato in un repository, così da rendere disponibile ad altri la versione aggiornata

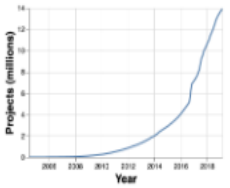
Maven™

- **Software Project Management Tool** per Java
- Permette di automatizzare diversi processi
  - Gestione delle **dipendenze**
  - **Compilazione** del codice
  - Esecuzione dei **test**
  - Generazione **documentazione**
  - Creazione pacchetti **software**
- Le caratteristiche del progetto sono definite all'interno di un file **XML** (pom.xml)
- Basato su **plugin** per estederne le funzionalità
- Utilizza **repository remoto** e **locale** (.m2) per fornire le dipendenze



Source: https://www.jrebel.com/blog/maven-cheat-sheet

Indexed Artifacts (43.9M)



Popular Categories

- Testing Frameworks & Tools
- Android Packages
- Logging Frameworks
- Java Specifications
- JVM Languages
- JSON Libraries
- Language Runtime
- Core Utilities
- Mocking
- Web Assets
- Annotation Libraries
- HTTP Clients
- Logging Bridges
- Dependency Injection
- XML Processing
- Web Frameworks
- I/O Utilities
- Defect Detection Metadata
- Code Generators
- Configuration Libraries
- OSGi Utilities

What's New in Maven

Scala3 Library Bootstrapped
org.scala-lang » scala3-library » 3.6.2-RC1-bin-20241022-ec...
Standard library for the Scala Programming Language Version 3
Last Release on Oct 23, 2024
8,738 usages
Apache

Scala3 Library Bootstrapped
org.scala-lang » scala3-library » 3.6.2-RC1-bin-20241022-ec...
Standard library for the Scala Programming Language Version 3
Last Release on Oct 23, 2024
8,738 usages
Apache

Scala3 Compiler Bootstrapped
org.scala-lang » scala3-compiler » 3.6.2-RC1-bin-20241022-ec...
scala3-compiler-bootstrapped
Last Release on Oct 23, 2024
104 usages
Apache

Big Data Genomics: Avro Formats
org.bdgenomics.bdg-formats » bdg-formats » 1.0.3
Big Data Genomics: Avro Formats
Last Release on Oct 23, 2024
27 usages
Apache

Ikasan Test Harness
org.ikasan » ikasan-test » 4.0.3
The test library for the Ikasan Enterprise Integration Platform
Last Release on Oct 23, 2024
26 usages
BSD

Ikasan EIP Standalone
org.ikasan » ikasan-eip-standalone » 4.0.3
25 usages
BSD

Indexed Repositories (2782)

- Central
- Atlassian
- WSO2 Releases
- Hortonworks
- JCenter
- Sonatype
- KtorEAP
- JBossEA
- WSO2 Public
- Atlassian Public

Popular Tags

aar android apache api application arm assets build build-system bundle client clojure cloud config cran data database eclipse example extension framework github gradle groovy ios javascript jboss kotlin library logging maven mobile module npm osgi plugin resources rlang sdk server service spring sql starter testing tools ui war web webapp

About

Web site developed by @frodriguez

Powered by: Scala, Play, Spark, Akka and Cassandra

File pom.xml

- Il file pom.xml (Project Object Model) è il cuore di un **progetto Maven** e definisce la configurazione del progetto, comprese le dipendenze, le informazioni di build e i plugin
- È un file XML che segue una **struttura** ben definita, con elementi standard. I principali sono:
  - **<modelVersion>:** La versione del modello di POM (generalmente 4.0.0)
  - **<groupId>:** L'ID del gruppo o organizzazione che sviluppa il progetto (e.g., com.example)
  - **<artifactId>:** Il nome dell'artefatto, ovvero il nome del progetto (e.g., my-app)
  - **<version>:** La versione del progetto (e.g., 1.0.0)
  - **<packaging>:** Il tipo di pacchetto da generare (jar, war, pom, ecc.)

Campi principali del pom

- La sezione **<dependencies>** consente di specificare le librerie di cui il progetto ha bisogno. Maven scaricherà automaticamente queste dipendenze, incluse quelle transitive
- La sezione **<build>** permette di configurare la build del progetto, definendo i plugin necessari. I plugin estendono le funzionalità di Maven, automatizzando attività come compilazione, test e packaging
- La sezione **<profiles>** consente di definire configurazioni diverse per diversi ambienti (e.g., sviluppo, test, produzione). Ogni profilo può avere dipendenze e plugin specifici
- È possibile dichiarare proprietà **<properties>** personalizzate per centralizzare la configurazione, come numeri di versione o URL di sistema, che possono essere riutilizzati in tutto il file.

Struttura Standardizzata

- Maven promuove una struttura standard per i progetti, il che facilita l'organizzazione del codice e la gestione delle dipendenze
- **pom.xml**: Il file di configurazione principale del progetto Maven, dove sono specificate le dipendenze, i plugin e altre impostazioni del progetto
- **src/**: La directory principale del codice sorgente
  - **main/**: Contiene il codice sorgente principale dell'applicazione
    - **java/**: Directory per i file Java sorgenti, organizzata secondo la convenzione di naming dei pacchetti (e.g., com/example)
    - **resources/**: Contiene file di risorse, come file di configurazione, immagini o file di proprietà, che devono essere inclusi nel pacchetto finale
  - **test/**: Contiene il codice sorgente dei test
    - **java/**
    - **resources/**
- **target/**:Directory generata automaticamente da Maven durante il processo di build. Contiene i file compilati e i pacchetti creati (e.g., file JAR o WAR). Questa cartella è generalmente ignorata nel controllo di versione, poiché è ricreata ad ogni build

Gestione delle dipendenze

- Maven risolve le dipendenze in modo **transitivo**, gestisce non solo le dipendenze dirette di un progetto, ma anche quelle di cui tali dipendenze dirette hanno bisogno
- **Dipendenze dirette**: Queste sono le librerie specificate nel file pom.xml del tuo progetto. Ad esempio, se il tuo progetto dipende da library-A, questa è una dipendenza diretta
- **Dipendenze transitive**: Se library-A a sua volta dipende da library-B, allora library-B è una dipendenza transitiva del tuo progetto. Maven si occupa di scaricare e includere anche library-B automaticamente, senza che tu debba specificarla nel tuo pom.xml

Maven Plugin

- I plugin in Maven sono componenti modulari che estendono le funzionalità di base di Maven, consentendo di automatizzare diverse attività durante il ciclo di vita di un progetto
- Ogni plugin è progettato per eseguire compiti specifici, come la compilazione del codice, l'esecuzione di test, la creazione di pacchetti, la gestione delle risorse e altro ancora
- **maven-compiler-plugin**: Utilizzato per compilare il codice sorgente Java. Puoi specificare la versione del linguaggio Java da utilizzare
- **maven-surefire-plugin**: Utilizzato per eseguire test unitari. È comunemente usato per eseguire test scritti con JUnit o TestNG
- **maven-jar-plugin**: Utilizzato per creare file JAR a partire dal codice compilato

JaCoCo

- **JaCoCo** (Java Code Coverage) è uno strumento di analisi della **copertura** del codice per applicazioni Java
- Viene utilizzato per misurare quanto del **codice sorgente** è stato eseguito durante i **test**, aiutando gli sviluppatori a identificare le aree del codice che non sono state testate
  - **Misurazione** della copertura
  - **Supporto** per diversi strumenti di build
  - **Report** dettagliati
  - **Integrazione** con strumenti di test
  - Report a livello di **classe** e **metodo**

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.jacoco</groupId>
            <artifactId>jacoco-maven-plugin</artifactId>
            <version>0.8.2</version>
            <executions>
                <execution>
                    <goals>
                        <goal>prepare-agent</goal>
                    </goals>
                </execution>
                <execution>
                    <id>report</id>
                    <phase>test</phase>
                    <goals>
                        <goal>report</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

code-coverage-maven-jacoco

code-coverage-maven-jacoco

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines
com.asimio.demo		37%		n/a	1	2	2	3
com.asimio.demo.rest		100%		n/a	0	2	0	5
com.asimio.demo.service		100%		n/a	0	2	0	2
Total	5 of 37	86%	0 of 0	n/a	1	6	2	10

Università di Catania

Uni ct MATEMATICA E INFORMATICA

The End 

Alessandro Midolo, PhD

Dipartimento di Matematica e Informatica

Università di Catania

alessandro.midolo@unict.it

https://www.dmi.unict.it/amidolo/

A.A. 2024/2025