Introduzione a Git

Presentazione di Fabio Iotti

Git

È una macchina del tempo

Cos'è Git?

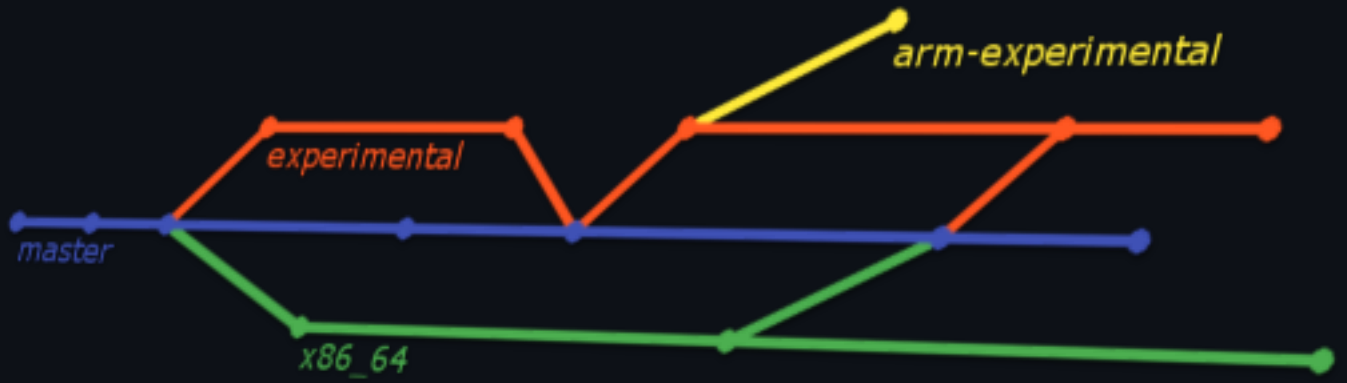
- È un sistema di controllo di versione distribuito (VCS)
- È nato come strumento per sviluppare Linux
- Permette di coordinare lo sviluppo su macchine diverse
- Permette a più sviluppatori di lavorare in contemporanea sulla stessa base di codice
- Mantiene una storia di tutte le versioni (chiamate *commit*)
- Git è uno strumento potente, ma anche complesso
- Git è una macchina del tempo*

*funziona solo sui file, purtroppo non permette di viaggiare nel tempo nel mondo reale

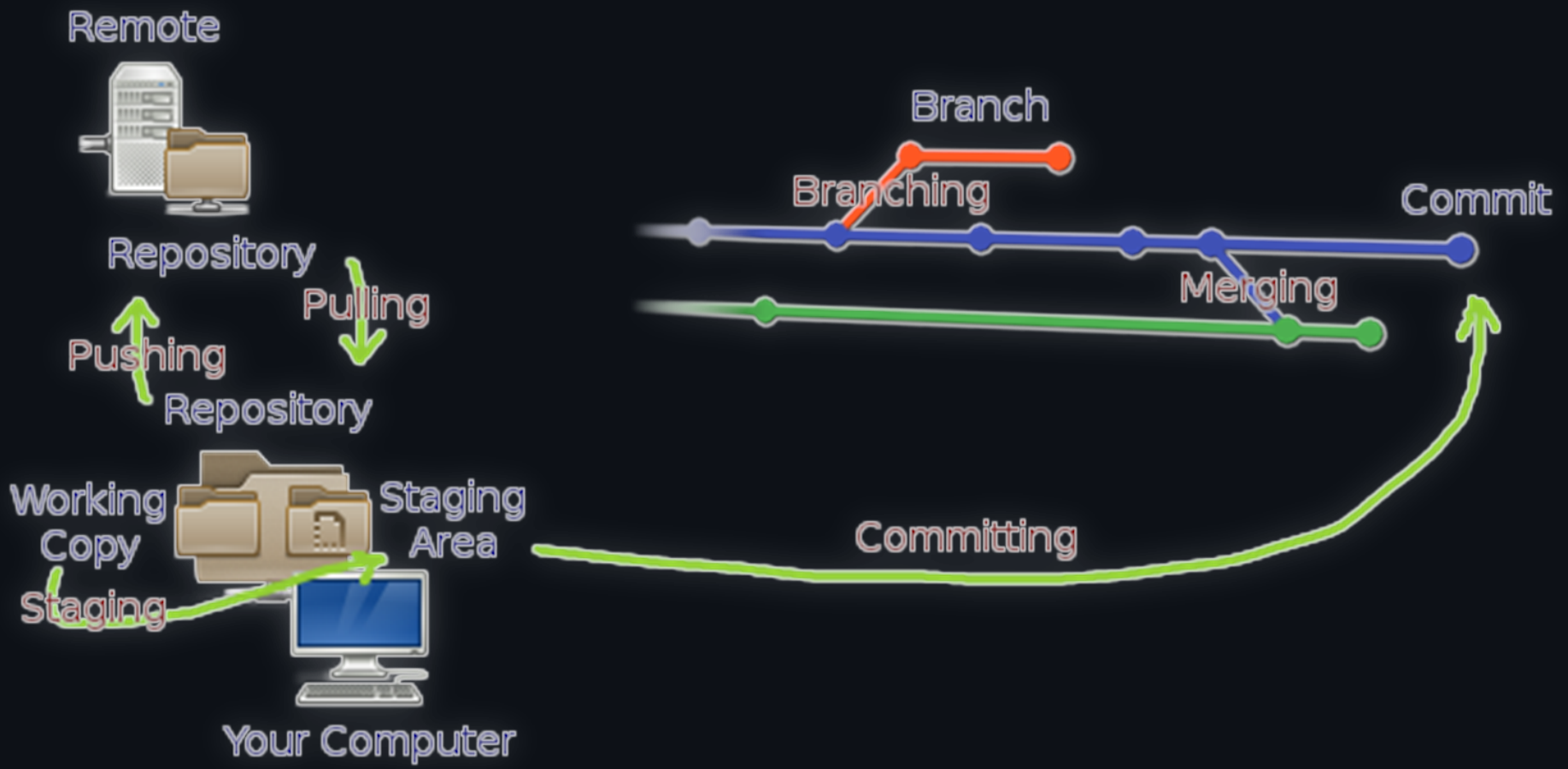
https://blog.fabioiotti.com/posts/git-windows-introduction/

Storia

- Git mantiene una storia di tutte le versioni (chiamate *commit*)
- È possibile utilizzare Git come meccanismo di backup
- In Git è comune sviluppare in parallelo su più *branch* separati
- Ogni branch ha la sua storia
- Ogni branch escluso il principale, nasce da un altro branch, ed eventualmente può morire con una *merge* in un altro branch



Terminologia



Terminologia

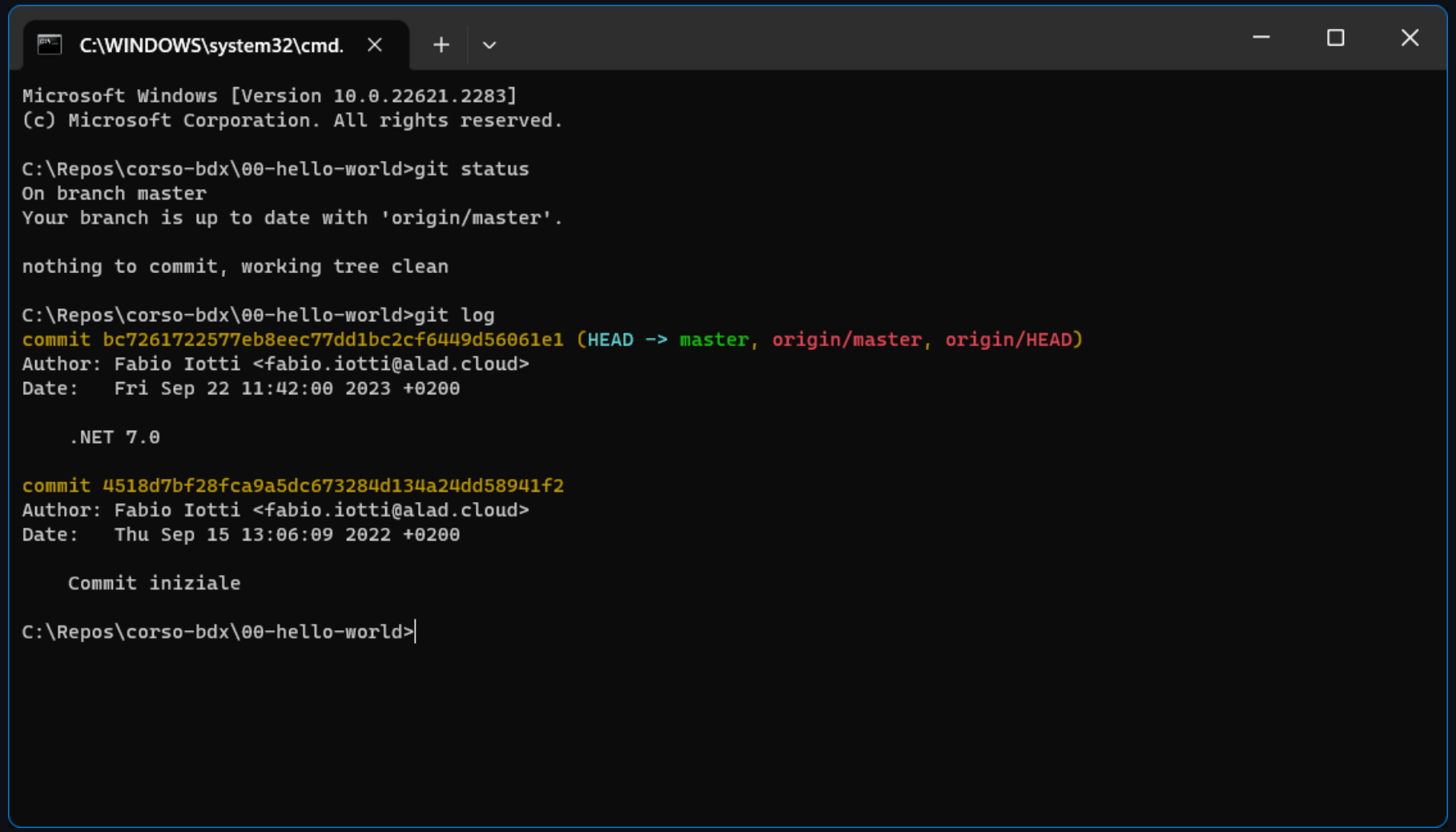
- **Repository:** contiene una serie di file, sia nella versione attuale che tutta la loro storia.
- **Working copy:** la versione dei file su cui stai attualmente lavorando.
- **Area di staging:** spazio temporaneo che contiene i file pronti per essere *«committati»*.
- **Staging:** prendere alcuni file dalla *working copy* e copiarli nell'*area di staging*.
- **Committing:** creare un nuovo *commit* da tutti i file nell'*area di staging*, svuotando poi l'*area di staging*.
- **Commit:** punto fisso nella storia che contiene una copia del codice in un momento particolare nella storia.

- **Remote:** un server remoto che ospita un *repository*.
- **Cloning:** scaricare una copia di un *repository* da un *remote* e salvarlo in una cartella in locale.
- **Fetching:** chiedere a un *remote* se c'è qualcosa di nuovo nella sua copia del *repository* rispetto alla versione locale.
- **Checkout:** spostarsi su una specifica *commit*.
- **Pulling:** equivale a fare *fetching* e poi *checkout* sull'ultima *commit* del *branch* corrente.
- **Pushing:** inviare il *repository* locale al *remote* e chiedere di unire la storia dei due *repository*.

- **Branch:** un percorso particolare nell'albero della storia di un *repository*, possono formarsi automaticamente quando due sviluppatori lavorano contemporaneamente sulla stessa versione del codice.
- **Branching:** creare un nuovo *branch*, ovvero una divergenza dal *branch* genitore nell'albero della storia del repository.
- **Merging:** prendere un *branch* e portare tutte le modifiche all'interno di un altro *branch* esistente.

Comandi

Git è l’unica macchina del tempo che si controlla da terminale

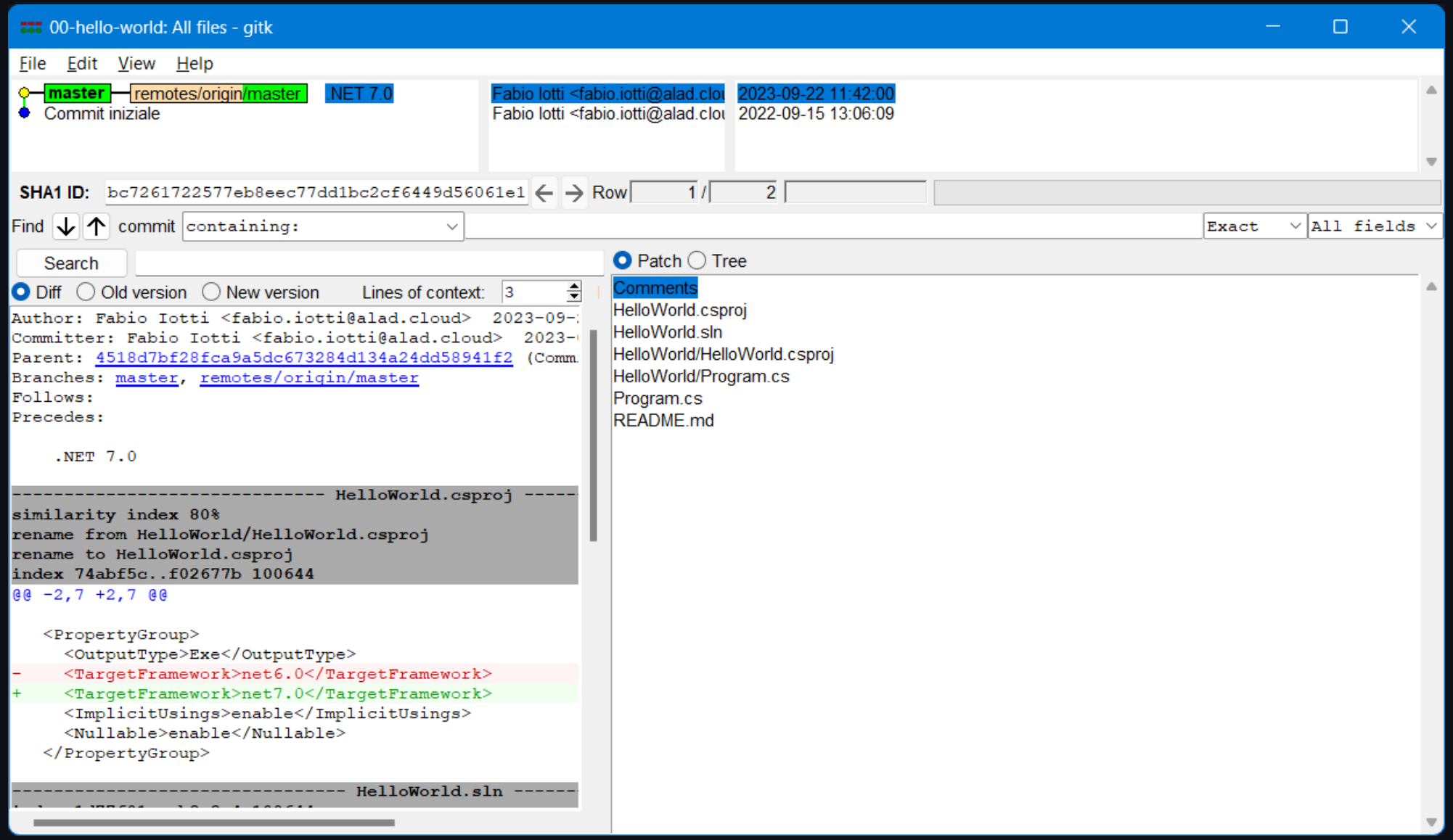

C:\WINDOWS\system32\cmd.
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.
C:\Repos\corso-bdx\00-hello-world>git status
On branch master
Your branch is up to date with 'origin/master'.
nothing to commit, working tree clean
C:\Repos\corso-bdx\00-hello-world>git log
commit bc7261722577eb8eec77dd1bc2cf6449d56061e1 (HEAD -> master, origin/master, origin/HEAD)
Author: Fabio Iotti <fabio.iotti@alad.cloud>
Date: Fri Sep 22 11:42:00 2023 +0200
.NET 7.0
commit 4518d7bf28fca9a5dc673284d134a24dd58941f2
Author: Fabio Iotti <fabio.iotti@alad.cloud>
Date: Thu Sep 15 13:06:09 2022 +0200
Commit iniziale
C:\Repos\corso-bdx\00-hello-world>

Clone

- Vogliamo clonare https://github.com/corso-bdx/00-hello-world
- Installare Git https://git-scm.com/
- Aprire un terminale (premere *WinKey* + *R* e digitare `cmd`)
- Spostarsi nella cartella interessata digitando per esempio:
  `cd C:\Users\MyUsername\Desktop`
- Fare clone del repository digitando:
  `git clone https://github.com/corso-bdx/00-hello-world`

Cronologia e stato

- La cronologia del repository è visibile con il comando `git log`
- È possibile controllare in qualsiasi momento lo stato di un repository locale con il comando `git status`
- Volendo è disponibile anche `gitk` che apre una semplice interfaccia grafica per vedere log e stato


00-hello-world: All files - gitk
File Edit View Help
master remotes/origin/master .NET 7.0
Commit iniziale
Fabio Iotti <fabio.iotti@alad.clou 2023-09-22 11:42:00
Fabio Iotti <fabio.iotti@alad.clou 2022-09-15 13:06:09
SHA1 ID: bc7261722577eb8eec77dd1bc2cf6449d56061e1
Row 1 / 2
Find commit containing: Exact All fields
Search
Patch Tree
Diff Old version New version Lines of context: 3
Author: Fabio Iotti <fabio.iotti@alad.cloud> 2023-09-
Committer: Fabio Iotti <fabio.iotti@alad.cloud> 2023-
Parent: 4518d7bf28fca9a5dc673284d134a24dd58941f2 (Comm
Branches: master, remotes/origin/master
Follows:
Precedes:
.NET 7.0
---------- HelloWorld.csproj -----
similarity index 80%
rename from HelloWorld/HelloWorld.csproj
rename to HelloWorld.csproj
index 74abf5c..f02677b 100644
@@ -2,7 +2,7 @@
<PropertyGroup>
<OutputType>Exe</OutputType>
- <TargetFramework>net6.0</TargetFramework>
+ <TargetFramework>net7.0</TargetFramework>
<ImplicitUsings>enable</ImplicitUsings>
<Nullable>enable</Nullable>
</PropertyGroup>
---------- HelloWorld.sln ------
Comments
HelloWorld.csproj
HelloWorld.sln
HelloWorld/HelloWorld.csproj
HelloWorld/Program.cs
Program.cs
README.md

Cheatsheet dei comandi di Git

- `git clone https://...`
- `git status`
- `git log`
- `gitk`
- `git add --all`
- `git commit`
- `git fetch`
- `git pull`
- `git push`

Commit

- Provare a modificare qualche file, per esempio in *Program.cs* sostituire «Hello, World!» con «Ciao, mondo!»
- Controllare lo stato sia con `git status` che con `gitk`, il file *Program.cs* dovrebbe risultare modificato
- Controllare il log con `git log`, non dovrebbe esserci nulla di nuovo
- Spostare il file dall'*area di lavoro* all'*area di staging* tramite comando `git add Program.cs`
- Controllare nuovamente lo stato ed il log
- Fare una *commit* con il comando `git commit` ed inserire un messaggio a piacere
- Controllare nuovamente lo stato ed il log, ora nel log dovrebbe apparire la nuova *commit*, con il relativo messaggio

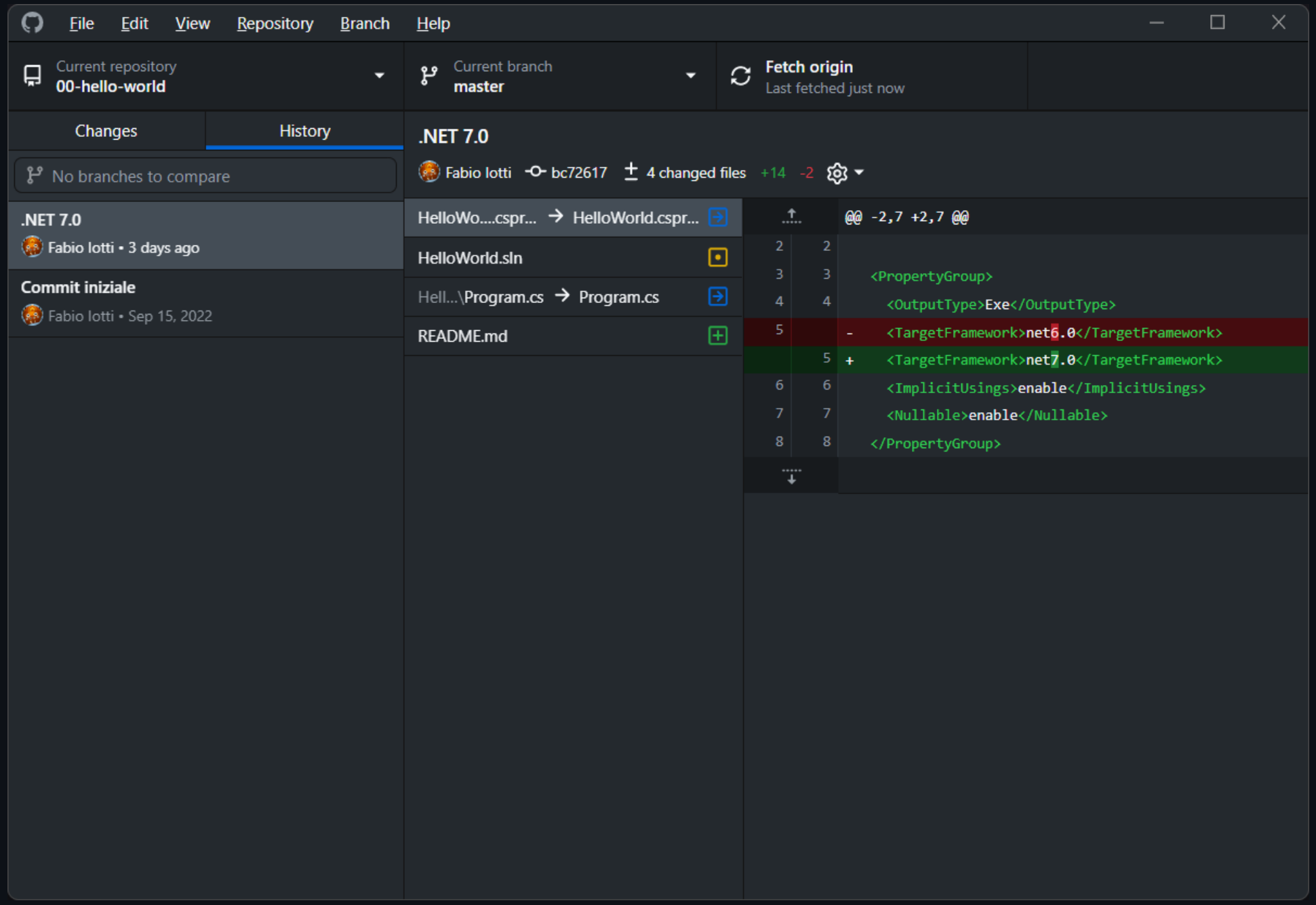
Push e pull

- Una volta fatta una commit, è possibile fare push per condividerla con il resto del team di sviluppo
- Il comando è `git push`
- Per verificare se sono presenti modifiche non scaricate si utilizza il comando `git fetch`
- Per scaricare le modifiche in arrivo si usa `git pull`

https://xkcd.com/1597/

Interfaccia

Per i viaggiatori del tempo alle prime armi è disponibile un’interfaccia grafica


File
Edit
View
Repository
Branch
Help
Current repository
00-hello-world
Current branch
master
Fetch origin
Last fetched just now
Changes
History
No branches to compare
.NET 7.0
Fabio Iotti • 3 days ago
Commit iniziale
Fabio Iotti • Sep 15, 2022
.NET 7.0
Fabio Iotti
bc72617
4 changed files
+14
-2
HelloWo....cspr... → HelloWorld.cspr...
HelloWorld.sln
Hell...\Program.cs → Program.cs
README.md
@@ -2,7 +2,7 @@
<PropertyGroup>
<OutputType>Exe</OutputType>
- <TargetFramework>net6.0</TargetFramework>
+ <TargetFramework>net7.0</TargetFramework>
<ImplicitUsings>enable</ImplicitUsings>
<Nullable>enable</Nullable>
</PropertyGroup>

GitHub Desktop

- Grazie a GitHub Desktop è possibile fare le più comuni operazioni supportate da Git senza bisogno di utilizzare il terminale
- Esistono altri client grafici per Git…
- …ma probabilmente GitHub Desktop è uno dei più utilizzati
- https://desktop.github.com/

- Nota personale: GitHub Desktop è pensato come uno strumento semplificato per usare Git, che è uno strumento complesso; GitHub Desktop non permette di fare tutto quello che Git permette di fare.
- In alcuni casi GitHub Desktop non usa nomenclatura standard, non permette di vedere più di un branch per volta, ed a volte si comporta in modo inaspettato.
- In alcuni casi potrebbe essere necessario risolvere eventuali problemi tramite terminale, oppure utilizzando altri client grafici.
- Personalmente, il mio client grafico per Git preferito è Git Graph per Visual Studio Code https://marketplace.visualstudio.com/items?itemName=mhutchie.git-graph

non è

Git, √GitHub e GitHub Desktop

- Git è il nome del sistema di controllo di versione
- `git` è il software da linea di comando
- GitHub è un servizio di hosting su Git
- GitHub è *uno dei tanti* servizi di hosting su Git…
- …ma è probabilmente il più usato
- GitHub Desktop è un client grafico per Git
- Git **non è** GitHub e GitHub Desktop

E ora un esempio pratico….