# **34 دستور کاربردی گیت**

اگر یک توسعه دهنده تازه کار یا حتی باتجربه هستید باید از کنترل منبع استفاده کنید. احتمال استفاده از گیت (Git) برای مدیریت کد منبع زیاد است. برای استفاده از Git باید دستورات آن را بشناسید. در این آموزش با مفیدترین دستورات گیت آشنا خواهید شد و می توانید سطح خودتان را بهبود دهید. در اینجا برای راهنمایی آسان و مفیدتر، دستورات Git را به سه بخش مختلف تقسیم کرده ایم: دستورات پایه، متوسط ​​و پیشرفته.

**دستورات سطح پایه Git**

در بخش اول دستورات پایه Git را یاد خواهید گرفت. این دستورات، پایه و اساس یادگیری دستورات پیشرفته تر هستند.

در اینجا 9 دستور کاربردی گیت آورده شده است:

1- git config

دستور git config بسیار مفید است به خصوص وقتی برای اولین بار از گیت استفاده می کنید یا یک گیت جدید نصب کرده باشید. این دستور هویت شما را با نام و آدرس ایمیل تنظیم می کند. بعد از این هر اطلاعات یا تغییری با این اعتبار ثبت خواهد شد.

تنظیم نام:

$ git config –global user.name “Your name”

تنظیم ایمیل:

$ git config –global user.email “Your email”

2- git version

همانطور که از اسمش مشخصه برای بررسی نسخه Git استفاده می شود.

$ git version

3- git init

احتمالاً اولین دستوری است که برای شروع یک پروژه جدید در Git استفاده می کنید. این دستور یک مخزن خالی جدید ایجاد می کند که در ادامه می توانید کد منبع خود را در داخل آن ذخیره کنید.

$ git init

همچنین می توانید نام مخزن را در دستور git init بیاورید.

$ git init <your repository name>

4- git clone

از دستور git clone برای کپی یک مخزن موجود استفاده می شود. یک تفاوت اساسی بین git init و git clone وجود دارد.

از git clone زمانی استفاده می کنید که نیاز به یک کپی روی مخزن موجود دارید. دستور git clone ابتدا از دستور git init استفاده می کند سپس تمام محتوای آن را بررسی خواهد کرد.

git clone <your project URL>

5- git add

دستور git add همه فایل های جدید کد یا فایل های ویرایش شده را به مخزن اضافه می کند. این دستور گزینه های مختلفی را برای اضافه کردن فایل ها و پوشه ها ارائه می دهد.

دستور زیر یک فایل خاص را به ناحیه stage اضافه می کند.

$ git add your\_file\_name

دستور زیر همه فایل های ویرایش شده و جدید را به ناحیه stage اضافه می کند.

$ git add \*

6- git commit

این یک دستور ضروری در Git است. در واقع دستور git commit تغییرات را به مخزن محلی اضافه خواهد کرد.

اگر از این دستور به درستی استفاده نکنید، کیفیت پروژه تحت تاثیر قرار می گیرد.

$ git commit -m “your useful commit message”

7- git status

git status برای بررسی وضعیت فایل ها استفاده می شود و می توانید تشخیص دهید کدام فایل ها نیاز به توجه دارند. این دستور در هر زمانی قابل اجرا است.

می توانید از آن در بین دستورات Git add و Git commits برای دیدن وضعیت استفاده کنید.

$ git status

7- git branch

اغلب چند شاخه در مخزن Git وجود دارد. شاخه یا branch یک خط مستقل از توسعه کد است.

با دستور git branch می توانید شاخه ها را به طور موثر مدیریت کنید. گزینه ها و سوئیچ های مختلفی از Git branch وجود دارد.

i) لیست کردن همه شاخه ها

$ git branch

ii) ساخت یک شاخه جدید

$ git branch <branch\_name>

iii) حذف یک شاخه

$ git branch -d <branch\_name>

9- git checkout

از این دستور برای جابجایی بین شاخه ها استفاده می شود. یکی از دستورات قدرتمند git است و می تواند به عنوان یک ابزار چند منظوره استفاده شود.

رفتن به شاخه دیگر:

$ git checkout <branch\_name>

همچنین می توانید یک شاخه جدید بسازید و روی آن سوییچ کنید.

$ git checkout -b <your\_new\_branch\_name>

**دستورات سطح متوسط ​​ Git**

بعد از دستورات پایه دستورات سطح متوسط Git ​​را بررسی می کنیم. این دستورات برای همکاری با تیم و به اشتراک گذاری کد خود با دیگران بسیار مفید هستند. دستورات دیگری مانند ورود به سیستم نیز وجود دارد که به شما کمک می کند تاریخچه commitهای قبلی را مشاهده کنید.

11- git remote

این دستور مانند یک مرز اطراف مخزن عمل می کند. در صورت نیاز به ارتباط با دنیای خارج از مخزن باید از دستور git remote استفاده کنید. این دستور مخزن محلی را به ریموت متصل می کند.

$ git remote add <shortname> <url>

**مثال**

$ git remote add origin https://dev.azure.com/aCompiler/\_git/DemoProject

11- git push

پس از اتصال به مخزن از راه دور (با کمک دستور git remote ) وقتشه که تغییرات را به مخزن بفرستید.

$ git push -u <short\_name> <your\_branch\_name>

**مثال**

$ git push -u origin feature\_branch

باید قبل از استفاده از دستور git push منبع و قسمت بالایی Git تنظیم شود.

$ git push –set-upstream <short\_name> <branch\_name>

**مثال**

$ git push –set-upstream origin feature\_branch

12- git fetch

در صورت نیاز به دانلود سایر تغییرات اعضای تیم یا دیگر مخزن ها باید از دستور git fetch استفاده کنید.

این دستور همه اطلاعات مربوط به commitها، رفرنس ها و … را دانلود می کند بنابراین می توانید قبل از اعمال این تغییرات در مخزن محلی خود آنها را مرور کنید.

$ git fetch

13- git pull

دستور git pull محتوا را دانلود کرده (نه متاداده ها) و بلافاصله مخزن محلی را با جدیدترین مطالب به روزرسانی می کند.

$ git pull <remote\_url>

14- git stash

این دستور فایل های ویرایش شده را به طور موقت ذخیره می کند. stash در لغت به معنی ذخیره کردن است. در گیت نیز هر چیزی که کامیت نشده رو ذخیره می کند.

می توانید با دستور git stash به صورت زیر کار کنید.

$ git stash

دستور زیر همه stashهای را لیست می کند.

$ git stash list

به سادگی یک stash به شاخه اعمال می کند.

$ git stash apply

15- git log

با کمک دستور git log می توانید همه کامیت های قبلی با شروع از آخرین کامیت را مشاهده کنید.

$ git log

به طور پیشفرض همه کامیت های شاخه فعلی را نشان می دهد اما می توانید از آن برای دیدن همه کامیت های کل شاخه ها با همه گزینه ها استفاده کنید.

$ git log –all

16- git shortlog

دستور git shortlog خلاصه ای از دستور git log را نشان می دهد. اگر فقط علاقه یا نیاز به یک خلاصه کوتاه دارید، این یک دستور مفید است.

این دستور برای دیدن اینکه چه کسی روی چه چیزی کار کرده نیز کاربرد دارد.

$ git shortlog

17- git show

در مقایسه با دستور git log، دستور git show جزئیات مربوط به یک کامیت خاص را نشان می دهد.

$ git show <your\_commit\_hash>

18- git rm

بعضی اوقات باید چند فایل را حذف کنید، در اینجا از دستور Git rm استفاده می شود.

این دستور می تواند فایل های ردیابی شده را از ایندکس و دایرکتوری کار حذف کند.

$ git rm <your\_file\_name>

19- git merge

دستور git merge کمک می کند تا تغییرات از دو شاخه را در یک شاخه واحد ادغام کنید.

$ git merge <branch\_name>

این دستور، <branch\_name> نوشته شده در دستور را در شاخه انتخاب شده فعلی ادغام می کند.

**دستورات سطح پیشرفته Git**

یک سطح جلوتر می رویم. در این بخش با دستورات پیشرفته Git آشنا خواهید شد. این دستورات به زمان و تمرین بیشتری نیاز دارند اما وقتی اصول آنها را یاد بگیرید استفاده از آنها آسانتر خواهد شد.

20- git rebase

git rebase شبیه دستور git merge است. دو شاخه را با یک استثنا در یک شاخه واحد ادغام می کند. یک دستور git rebase تاریخچه کامیت را بازنویسی می کند.

اگر چند شاخه خصوصی دارید باید از دستور git rebase برای ادغام در یک شاخه استفاده کنید. این باعث می شود که یک تاریخچه کامیت خطی بسازید.

$ git rebase <base>

21- git bisect

دستور git bisect کمک می کند تا کامیت های بد را پیدا کنید.

I ) شروع git bisect

$ git bisect start

II) به git bisect درباره یک کامیت خوب اطلاع دهید.

$ git bisect good a123

III) به git bisect درباره یک کامیت بد اطلاع دهید.

$ git bisect bad z123

با استفاده از git bisect می توانید کد خراب را در عرض چند دقیقه تعیین کرده و میزان کدهای خراب را کم کنید.

22- git cherry-pick

git cherry-pick یک دستور مفید و قوی است که امکان انتخاب هر کامیتی را از هر شاخه و اعمال آن به هر شاخه دیگری را فراهم می کند.

$ git cherry-pick <commit-hash>

23- git archive

دستور git archive چند فایل را در یک فایل واحد ترکیب می کند. مانند ابزار zip است، یعنی می توانید فایل ها را از حالت فشرده خارج کنید و به هر کدام از فایل ها به طور جداگانه دسترسی داشته باشید.

$ git archive –format zip HEAD > archive-HEAD.zip

یک فایل zip از بازبینی فعلی ایجاد می کند.

24- git pull –rebase

بیشتر مواقع هنگام استفاده از git pull باید rebase (نه ادغام یا merge) انجام دهید.

در این صورت می توانید از گزینه زیر استفاده کنید.

$ git pull –rebase

این کمک می کند تا تاریخچه را تمیز نگه دارید. همچنین از ادغام های متعدد جلوگیری می کند.

25- git blame

برای بررسی سطر به سطر محتوای هر فایل از دستور git blame استفاده کنید. این دستور کمک می کند تا تشخیص دهید چه کسی تغییرات را در یک فایل ایجاد کرده است.

$ git blame <your\_file\_name>

26- git tag

در Git تگ ها مفید هستند و می توانید از آنها برای مدیریت انتشار استفاده کنید. می توانید تگ را مانند یک شاخه بدون تغییر در نظر بگیرید. اهمیت آن هنگام انتشار عمومی خیلی بیشتری است. از دستور git tag برای ایجاد یک تگ جدید استفاده می شود.

$ git tag -a v1.0.0

27- git verify-commit

دستور git verify-commit امضای gpg کامیت ها را بررسی می کند GPG یا “GNU Privacy Guard” است.

$ git verify-commit <commit>

28- git verify-tag

با این دستور می توانید تگ را تأیید کنید.

$ git verify-tag <tag>

29- git diff

بیشتر اوقات قبل از commit یا push باید دو فایل یا شاخه git را با هم مقایسه کنید. دستور مفید git diff برای این کار استفاده می شود.

I) مقایسه دایرکتوری کاری با مخزن محلی:

$ git diff HEAD <filename>

II) مقایسه دو شاخه:

$ git diff <source branch> <target branch>

30- git citool

git citool یک جایگزین گرافیکی برای کامیت Git است.

$ git citool

31- git mv

دستور git mv برای تغییر نام فایل git استفاده می شود. دو آرگومان دارد؛ نام فایل منبع و مقصد.

$ git mv <old-file-name> <new-file-name>

32- git clean

با استفاده از دستور git clean می توانید با فایل های ردیابی نشده (untracked) کار کنید. با این دستور می توانید همه فایل های ردیابی نشده را از دایرکتوری کار حذف کنید. برای کار با فایل های ردیابی شده (tracked) باید از دستور git reset استفاده کنید.

$ git clean

33- git help

دستورات زیادی در Git وجود دارد، برای به دست آوردن اطلاعات و راهنمایی بیشتر برای هر دستور می توانید از دستور git help در هر زمان از ترمینال استفاده کنید.

$ git help <git\_command>

34- git whatchanged

دستور git whatchanged همان کار git log را انجام می دهد اما در قالب خام.

$ git whatchanged

**گیت هاب**

### **create a new repository on the command line**

echo "# GitHubTest" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/saeidmot/GitHubTest.git

git push -u origin main

git push --all

### **push an existing repository from the command line**

git remote add origin https://github.com/saeidmot/GitHubTest.git

git branch -M main

git push -u origin main

git push –all

**Clone a repository from github on Pc:**

Git clone https://github.com/saeidmot/GitHubTest.git

**Update pc file from github files:**

git fetch

git merge

**Update GitHub files from pc:**

git push –-all

**Delete a remote:**

git remote rm <name>

**Git**

0>Create GitProject directory

1>git init

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git init

Initialized empty Git repository in C:/Users/Mottaghi\_Asus/OneDrive/Desktop/GitProject/.git/

2>Create a file named index.html in GitProject directory

3>git status

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status

On branch master

No commits yet

Untracked files:

(use “git add <file>…” to include in what will be committed)

index.html

nothing added to commit but untracked files present (use “git add” to track)

4> git add index.html

5>git status

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git add index.html

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status

On branch master

No commits yet

Changes to be committed:

(use “git rm –cached <file>…” to unstage)

new file: index.html

6> git config –global user.email “saeidmot@gmail.com”

7>git config –global user.name “saeidmot”

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git config –global user.email “saeidmot@gmail.com”

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git config –global user.name “saeidmot”

8>git commit -m “Create index file”

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git commit -m “create index file”

[master (root-commit) 831ec64] create index file

1 file changed, 10 insertions(+)

create mode 100644 index.html

9> we add change to index.html file

10>git status

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status

On branch master

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: index.html

no changes added to commit (use “git add” and/or “git commit -a”)

11>git add -A -A meens all or git add index.html or git add “\*.html”

12> git status

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status

On branch master

Changes to be committed:

(use “git restore –staged <file>…” to unstage)

modified: index.html

13> git commit -m “change title of index.html file”

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git commit -m “ change title of index.html file”

[master adbbe1d] change title of index.html file

1 file changed, 1 insertion(+), 1 deletion(-)

14>git status

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status

On branch master

nothing to commit, working tree clean

15>git log

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git log

commit adbbe1d4d40eb556a8471a26ea0cdba782323106 (HEAD -> master)

Author: saeidmot <saeidmot@gmail.com>

Date: Tue Apr 23 13:02:10 2024 +0330

change title of index.html file

commit 831ec64e949f3b1266d45955e65ad342b7238e12

Author: saeidmot <saeidmot@gmail.com>

Date: Tue Apr 23 12:44:32 2024 +0330

create index file

**ایجاد فایلهای جدید**

1. Create 3 file named page1.html,page2.html, page3.html

2>git status

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status

On branch master

Untracked files:

(use “git add <file>…” to include in what will be committed)

page1.html

page2.html

page3.html

nothing added to commit but untracked files present (use “git add” to track)

3>git add “page\*”

تمام فایلهایی که نام آنها با page شروع می شود در stage قرار می گیرند.

4>git status

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git add “page\*”

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status

On branch master

Changes to be committed:

(use “git restore –staged <file>…” to unstage)

new file: page1.html

new file: page2.html

new file: page3.html

5>git commit -m “ commit 3 pages”

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git commit -m “ commit 3 pages”

[master a7b5df5] commit 3 pages

3 files changed, 20 insertions(+)

create mode 100644 page1.html

create mode 100644 page2.html

create mode 100644 page3.html

**بررسی تغییرات**

1. Change title of page2.html (from Document to Page2)

2>git diff HEAD

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git diff head

diff –git a/page2.html b/page2.html

index 633d2d2..613f048 100644

--- a/page2.html

+++ b/page2.html

@@ -2,7 +2,7 @@

<head>

<meta charset=”UTF-8”>

<meta name=”viewport” content=”width=device-width, initial-scale=1.0”>

- <title>Document</title> 🡪 old

+ <title>Page 2</title> 🡪 New

</head>

<body>

3>git add -A

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git add -A

4>git status

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status

On branch master

Changes to be committed:

(use “git restore –staged <file>…” to unstage)

modified: page2.html

5>git diff –staged -> show changes for all staged files

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git diff –staged

diff –git a/page2.html b/page2.html

index 633d2d2..613f048 100644

--- a/page2.html

+++ b/page2.html

@@ -2,7 +2,7 @@

<head>

<meta charset=”UTF-8”>

<meta name=”viewport” content=”width=device-width, initial-scale=1.0”>

- <title>Document</title>

+ <title>Page 2</title>

</head>

<body>

بر گرداندن تغییرات یک فایل که در مرحله stage است(قبل از Commit) به آخرین Commit ای که انجام شده بود.(لغو تمام تغییرات):

1> git checkout – page2.html

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git checkout – page2.html

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status

On branch master

Changes to be committed:

(use “git restore –staged <file>…” to unstage)

modified: page2.html

2>git restore –staged page2.html

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git restore –staged page2.html

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status

On branch master

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: page2.html

3>git restore page2.html

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git restore page2.html

با این دستور تغییرات در فایل page2.html اعمال میشود و title از page2 به Document باز می گردد. کلا با دستور git restore <filename> فایل به آخرین commit ای که انجام شده بود باز می گردد.

مثال: در 3 فایل page1.html و page2.html و apge3.html تغییراتی داده شد.

1>C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status

On branch master

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: page1.html

modified: page2.html

modified: page3.html

no changes added to commit (use “git add” and/or “git commit -a”)

اکنون آنها را commit می کنیم تا تغییرات آنها در گیت اعمال و ذخیره شود.

2>C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git commit -a -m “update3 file with a tag”

[master e31f07a] update3 file with a tag

3 files changed, 3 insertions(+), 2 deletions(-)

اکنون این 3 فایل را مجددا تغییر می دهیم. می توانیم تغییرات جدید را نسبت به نسخه commit شد، مشاهده کنیم.

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git diff HEAD

diff –git a/page1.html b/page1.html

index 83854dd..bedbb2b 100644

--- a/page1.html

+++ b/page1.html

@@ -5,7 +5,7 @@

<title>page1</title>

</head>

<body>

- <a>Hello from 1</a>

+ <a>Hello from 1 2 times</a>

</body>

</html>

\ No newline at end of file

diff –git a/page2.html b/page2.html

index d218480..a495030 100644

--- a/page2.html

+++ b/page2.html

@@ -5,6 +5,6 @@

<title>document</title>

</head>

<body>

- <a>Hello from 2</a>

+ <a>Hello from 2 2 times</a>

</body>

</html>

\ No newline at end of file

diff –git a/page3.html b/page3.html

index 27c3f1f..059c397 100644

--- a/page3.html

+++ b/page3.html

@@ -5,6 +5,6 @@

<title>page3</title>

</head>

<body>

-<a>Hello from 3</a>

+<a>Hello from 3 for 2 times</a>

</body>

</html>

\ No newline at end of file

(END)

</body>

</html>

\ No newline at end of file

جهت حذف این تغییرات و برگشتن به نسخه commit شده:

ابتدا تمام فایلهایی که ابتدای نام آنها page است از مرحله stage خارج می شوند.

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git restore –staged page\*

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status

On branch master

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: page1.html

modified: page2.html

modified: page3.html

no changes added to commit (use “git add” and/or “git commit -a”)

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git restore page\*

با آخرین دستور، کلیه تغییرات کنسل می شود و همه فایلهایی که نام آنها با page شروع می شوند به نسخه Commit شده برمی گیردند.

شاخه ها در گیت:

یک شاخه درست می کنیم که در این شاخه تغییرات تگ a فایل های html انجام می شود.

**C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git branch aTagChange**

**C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status**

On branch master

nothing to commit, working tree clean

**C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git branch**

aTagChange

\* master

یعنی یک شاخه Master و یک شاخه aTagChange داریم و با علامت \* نشان می دهد که در شاخه master هستیم.

اکنون می خواهیم به شاخه aTageChange برویم تا در آن شاخه فایلها را تغییر بدهیم.

**C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git checkout aTagChange**

Switched to branch ‘aTagChange’

اکنون تغییرات مورد نظر را در تگ a فایلهای page1 و page2 و page3 انجام می دهیم و یک فایل بنام pageTage.html نیز در پوشه gitProject ایجاد می کنیم.

**C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status**

On branch aTagChange

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: page1.html

modified: page2.html

modified: page3.html

Untracked files:

(use “git add <file>…” to include in what will be committed)

pageTag.html

no changes added to commit (use “git add” and/or “git commit -a”)

اکنون انها را Add و Commit می کنیم.

**C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git add page\***

**C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status**

On branch aTagChange

Changes to be committed:

(use “git restore –staged <file>…” to unstage)

modified: page1.html

modified: page2.html

modified: page3.html

new file: pageTag.html

اکنون می بینیم که همه چیز تمیز و مرتب است

**C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git status**

On branch aTagChange

nothing to commit, working tree clean

اکنون اگر به شاخه master بر گردیم این تغییرات شامل تغییر در تگ a و ایجاد فایل pageTag در پوشه GitProject وجود نخواهند داشت.

**C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git checkout master**

Switched to branch ‘master’

**C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>dir**

Volume in drive C has no label.

Volume Serial Number is 4675-EF4C

Directory of C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject

04/24/2024 12:54 PM <DIR> .

04/24/2024 12:10 PM <DIR> ..

04/23/2024 12:49 PM 199 index.html

04/24/2024 12:54 PM 220 page1.html

04/24/2024 12:54 PM 217 page2.html

04/24/2024 12:54 PM 210 page3.html

4 File(s) 846 bytes

2 Dir(s) 283,060,912,128 bytes free

حتی در نرم افزار visual studio code نیز فایل pageTag حذف شده و فایلها بدون تغییر در تگ a می باشند. در صورتیکه مجددا به شاخه aTageChange برگردیم تغییرات در آن شاخه اعمال و فایل pageTag هم پدیدار می شود.

**C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git checkout aTagChange**

Switched to branch ‘aTagChange’

**C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>dir**

Volume in drive C has no label.

Volume Serial Number is 4675-EF4C

Directory of C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject

04/24/2024 01:01 PM <DIR> .

04/24/2024 12:10 PM <DIR> ..

04/23/2024 12:49 PM 199 index.html

04/24/2024 01:01 PM 225 page1.html

04/24/2024 01:01 PM 222 page2.html

04/24/2024 01:01 PM 215 page3.html

04/24/2024 01:01 PM 197 pageTag.html

5 File(s) 1,058 bytes

2 Dir(s) 283,126,431,744 bytes free

اکنون جهت اعمال تغییراتی که در شاخه aTageChange دادیم در شاخه master اعمال شود، باید در حالیکه در شاخه master هستیم اقدام به merge نمائیم.

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>**git status**

On branch master

nothing to commit, working tree clean

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>**git merge aTagChange**

Updating e31f07a..f99a9bf

Fast-forward

page1.html | 2 +-

page2.html | 2 +-

page3.html | 2 +-

pageTag.html | 10 ++++++++++

4 files changed, 13 insertions(+), 3 deletions(-)

create mode 100644 pageTag.html

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>**dir**

Volume in drive C has no label.

Volume Serial Number is 4675-EF4C

Directory of C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject

04/24/2024 01:16 PM <DIR> .

04/24/2024 01:03 PM <DIR> ..

04/23/2024 12:49 PM 199 index.html

04/24/2024 01:16 PM 225 page1.html

04/24/2024 01:16 PM 222 page2.html

04/24/2024 01:16 PM 215 page3.html

04/24/2024 01:16 PM 197 pageTag.html

5 File(s) 1,058 bytes

2 Dir(s) 283,122,487,296 bytes free

بعد از اینکه شاخه را merge کردیم میتوانیم آن شاخه را پاک نمائیم. در حایکه در شاخه master هستیم با دستور git branch -d اقدام به حذف شاخه aTagChange می کنیم.

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>**git branch -d aTagChange**

Deleted branch aTagChange (was f99a9bf).

گیت هاب

می توانیم پروژه های درون github را در روی سیستم خود داشته باشیم و انها را تغییر دهیم.

ابتدا یک دایرکتوری بنام آن پروژه می سازیم و آدرس یک Repository را در سیستم خود clone می کنیم.

>git clone <address>

>git status

می توانیم تغییرات را در فایل مورد نظر اعمال نماییم سپس

>git add -A

>git commit -m “text”

به پروژه ای که در گیت هاب قراردارشت والان آن را روی سیستم خود clone کردیم، origin می گویند به پروژه ای که الان روی آن کار می کنیم master می گوئیم الان با دستور زیر تغییرات master برروی origin اعمال می شود.

>git push origin master

با گرفتن یوزر نیم و پسورد تغییرات بر روی origin در پروژه اعمال می شود و یک commit به commit های قبلی در گیت هاب اضافه می شود.

حال اگر تغییراتی در گیت هاب بر روی پروژه (origin) اعمال شده باشد و بخواهیم آن را در master خود داشته باشیم.

>git pull origin master

دستور git remote

>git remote

>git remot add <name> <Adress>

Git remot origin <https://github/saeidmot/test>

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git remote add origin https://github.com/saeidmot/test

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git remote

Origin

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git remote -v

origin https://github.com/saeidmot/test (fetch)

origin https://github.com/saeidmot/test (push)

ارسال شاخه master به origin

C:\Users\Mottaghi\_Asus\OneDrive\Desktop\GitProject>git push -u origin master

remote: Repository not found.

fatal: repository 'https://github.com/saeidmot/test/' not found

ولی چون آدرس اشتباه بود دسترسی پیدا نکرد.

Tag زدن:

جهت شناسایی بهتر و اعمال ورژن از تگ زدن استفاده میشود.

$ git tag -a v2.1 -m "second version with a little chanes"

$ git tag

v2.0

v2.1

می توان با درج قسمتهایی از اول سریال یک کامیت قبلی، نیز تگ ایجاد کرد.

$ git tag -a v1.8 e31f07abc30cc -m "version 1.8 created"

دیدن اقدامات انجام شده در یک ورژن

$ git show v1.8

tag v1.8

Tagger: saeidmot <saeidmot@gmail.com>

Date: Mon May 6 08:29:43 2024 +0330

version 1.8 created

commit e31f07abc30cc253266e466387356cf46c9fd5ca (tag: v1.8)

Author: saeidmot <saeidmot@gmail.com>

Date: Wed Apr 24 11:51:01 2024 +0330

update3 file with a tag

**diff --git a/page1.html b/page1.html**

**index 80b4bee..83854dd 100644**

**--- a/page1.html**

**+++ b/page1.html**

@@ -5,6 +5,7 @@

<title>page1</title>

</head>

<body>

+ <a>Hello from 1</a>

</body>

</html>

\ No newline at end of file

**diff --git a/page2.html b/page2.html**

**index 481c598..d218480 100644**

**--- a/page2.html**

**+++ b/page2.html**

@@ -5,6 +5,6 @@

<title>document</title>

</head>

<body>

:...skipping...

tag v1.8

Tagger: saeidmot <saeidmot@gmail.com>

Date: Mon May 6 08:29:43 2024 +0330

version 1.8 created

commit e31f07abc30cc253266e466387356cf46c9fd5ca (tag: v1.8)

Author: saeidmot <saeidmot@gmail.com>

Date: Wed Apr 24 11:51:01 2024 +0330

update3 file with a tag

**diff --git a/page1.html b/page1.html**

**index 80b4bee..83854dd 100644**

**--- a/page1.html**

**+++ b/page1.html**

@@ -5,6 +5,7 @@

<title>page1</title>

</head>

<body>

+ <a>Hello from 1</a>

</body>

</html>

\ No newline at end of file

**diff --git a/page2.html b/page2.html**

**index 481c598..d218480 100644**

**--- a/page2.html**

**+++ b/page2.html**

@@ -5,6 +5,6 @@

<title>document</title>

</head>

<body>

- <a>Hello</a>

+ <a>Hello from 2</a>

</body>

</html>

\ No newline at end of file

**diff --git a/page3.html b/page3.html**

**index 084a1e2..27c3f1f 100644**