|  |  |
| --- | --- |
| 1. چک میکنیم کجاییم | pwd |
| 2. دایرکتی فعلی رو چک میکنیم | ~ |
| 3. ایجاد یک فولدر به اسم repos و داخلش یکی دیگه به اسم projects | Mkdir repos  Cd repos  Mkdir projects  Cd projects |
| 4. شروع گیت | Git init |
| 5. ساخت فایلی به اسم index، با تایپ html و کامیت کردن آن | Touch index.html  Git add index.html  Git commit –m ‘index’ |
| 6. دیدن وضعیت گیت | Git status |
| 7. تغییر آن فایل ( اضافه کردن html) و مجدد کامیت کردن آن با پیام html added | Code .  (change the file)  Git add index.html  Git commit –m ‘html added’ |
| 8. دیدن تمام فایل ها + hidden ها + توضیحاتشان | Ls -ltrha |
| 9. فایل هایی که داخل فولدر .git هست رو نگاه میکنیم | ls -ltrha .git |
| 10. همه چیزو پاک میکنیم و وضعیت گیت رو نگاه میکنیم | Clear  Git status |
| 11. سه تا فایل دیگه با نام های page1.html تا page3 درست میکنیم، بدون استیج کردن | Touch page1.html page2.html  Page3.html |
| 12. فایل index را تغییر میدهیم بدون استیج کردن | Code .  (change the file) |
| 13. حالا فقط اونایی که داخل اسمشون page هست رو کامیت میکنیم | Git add ‘page\*’  Git commit –m ‘3pages’ |
| 14. حالا فایل index رو جدا کامیت میکنیم | Git add index.html  Git commit –m ‘hi added’ |
| 15. دیدن تاریخچه ی کامیت ها و خارج شدن از آن | Git log  Q + enter |
| 16. حالا مجدد فایل index.html رو تغییر میدیم. دو بار تغییراتش رو چک میکنیم. یکی قبل از استیج کردن و  و یکی بعد از استیج کردن. چون بعد از کامیت کردن دیگه نمیشه جزئیات تغییرات داخل یه فایلو دید. ( نکته: کامند این دو متفاوت است) | Code .  (change the file)  Git diff HEAD  Git add index.html  Git diff --staged |
| 17. حالا دو تا فایل رو با هم تغییر داده. با هم استیج میکنیم. و بعد باهم زیر تغییراتشونو چک میکنیم. | (we again change the index.html and page1.html)  Git add –A  Git diff --staged  \*اگر الان بزنیم  Git diff HEAD  تغییرات اخرین فایلو برامون میاره نه هردو |
| 18. حالا هر دو فایل رو از استیج در میاریم | Git reset index.html  Git reset page1.html |
| 19. فایل index.html را یکی به کامیت قبلی برمیگردونیم. | git reset --hard HEAD~1 |
| 20. میریم داخل فایل projects و branch هامون رو چک میکنیم | Cd projects  Git branch |
| 21. همه ی فایل های موجود رو به صورت کوتاه شده چک میکنیم | Ls |
| 22. یک برنچ خالی به اسم fixpages درست میکنیم | Git branch fixpages |
| 23. چک میکنیم روی کدام برنچ هستیم.  وارد برنچ fixpages میشویم و دوباره شاخه ها را چک میکنیم که کجاییم. | Git checkout fixpages  Git branch |
| 24. حالا به تمام اون 3 تا فایل کلمه ی html را در h1 اضافه میکنیم و یک فایل به اسم footer.html هم درست میکنیم. | Code .  (We change all the 3 pages)  Touch footer.html |
| 25. وضعیت را چک میکنیم | Git status |
| 26. تمام فایل هارو باهم استیج میکنیم | Git add -A |
| 27. فقط فایل فوتر رو از استیج درمیاریم | Git reset footer.html |
| 28. اون 3 تا فایلو با هم کامیت میکنیم با پیام added html | Git commit –m ‘added html’ |
| 29. بعد فایل فوتر رو کامیت میکنیم با پیام hi | Git add footer.html  Git commit –m ‘hi’ |
| 30. حالا برمیگردیم به شاخه ی مستر | Git checkout master |
| 31. حالا تاریخچه ی کامیت هارو چک میکنیم که داخلشون هیچ footer و یا تغییراتی که داخل اون برنچ fixpages دادیم، نیست | Git log |
| 32. از این حالت تاریخچه خارج میشیم | Q + enter |
| 33. حالا برنچ fixpages رو با برنچ master یکی میکنیم | Git merge fixpages |
| 34. حالا فایل هارو نگاه میکنیم ( به طور مختصر فقط با اسمشون) | Ls |
| 35. تاریخچه ی کامیت را هم چک کرده و بعد از این قسمت خارج میشویم | Git log  Q + enter |
| 36. حالا تا اینجا ما برای index.html یه footer.html درست کردیم و فایل ها هم باهم مرج کردیم اما لینک نکردیم یعنی فوتر در قسمت پایانی صفحه ی index.html وجود نداره.  حالا برای لینک کردنشون به هم باید یه برنچ دیگه درست کنیم.  اول برنچ هارو چک میکنیم و یه برنچ به اسم  Linkingpages درست میکنیم و واردش میشیم. | Git branch  Git branch linking pages  git checkout linkingpages |
| 37. حالا فوتر رو پاک میکنیم | Git rm footer.html |
| 38. پیج های 1 و 2 و 3 رو داخل index.html لینک میکنیم | <br><a *href*="page1.html">1</a>      <br><a *href*="page2.html">2</a>      <br><a *href*="page3.html">3</a> |
| 39. میریم داخل برنچ مستر | git checkout master |
| 40. میریم داخل برنچ linkingpages | Git checkout linkingpages |
| 41. حالا باید فایل فوتر رو کامیت کنیم که یعنی پاک شده | git commit -m 'removed footer' |
| 42. حالا باید فایل تغییر داده شده ی index رو جدا اد و کامیت کنیم | git commit -m 'added page links to index.html' |
| 43. حالا میخوایم ببینیم کدوم تغییرات رو داخل کدوم برنچ دادیم | git log |
| 44. وارد برنچ مستر میشیم | Git branch master |
| 45. هنوزم فایل footer.html که در برنچ linkingpages پاکش کرده بودیم، وجود داره! داخل فایل index.html رو چک میکنیم | Cat index.html  نکته: میخواد نشون بده یعنی تا مرج نکنی تغییرات داخل برنچ مستر نمایش داده نمیشن. |
| 46. این هم همون همون ورژن اولیس که داخلش اون 3 تا پیج لینک نشده پس تغییرات اون برنچ رو به برنچ مستر میاریم. | Git merge linkingpages |
| 47. حالا بازم محتویات داخل index.html رو نگاه میکنیم و میبینیم که فایل جدید اد شده. | Cat index.html |
| 48. برنچ ها را چک میکنیم | Git branch |
| 49. حالا دو برنچ linkingpages و fixpages رو پاک میکنیم چون دیگه نیازی بهشون نداریم. | Git branch -d fixpages  Git branch -d linkingpages |
| 50. میخوایم یه remote repository رو به  Local repository خودمون اضافه کنیم پس | میریم داخل گیت هاب مثلا این اکانت  <https://github.com/jadijadi> |
| 51. حالا میخوایم یه پروژه رو کلون کنیم | میریم داخل قسمت repositories، بعد این  <https://github.com/jadijadi/titap_mystry> |
| 52. ادرس گیت رو کپی میکنیم | یه جا زده code و میشه هم دانلود کرد هم کلون کرد. یه ادرس گیت هم داره. اون لینک HTTPS رو کپی میکنیم.  <https://github.com/jadijadi/titap_mystry.git> |

|  |  |
| --- | --- |
| Cd ..  Cd ..  Git clone <https://github.com/jadijadi/titap_mystry.git>  نکته: حالا گیت میگه من یه repository به همون اسم درست کردم و کل پروژه رو کلون کردم اون تو | 53. از فایل repos میایم بیرون  و کلونش میکنیم. |
| Cd titap\_mystry | 54. میریم داخل فایل  Titap\_mystry یا هر اسم دیگه ای که داره |
| Code .  نکته: میشه با دستور vi هم داخل گیت ادیتش کرد ولی بیرون اومدن ازش سخته پس با همون vsc ادیت میکنیم. | 55. یکی از فایل ها مثلا  README.md رو تغییر میدیم. |
| Git status | 56. وضعیت گیت رو نگاه میکنیم. |
| Git add –A  Git commit –m ‘space added’ | 57. فایلی که تغییر داده بودیم رو کامیت میکنیم. |
| Git status  میگه شما یه پله از origin بالاتری. | 58. وضعیت گیت رو نگاه میکنیم. |
| Git push origin master | 59. حالا میخوام این تغییر رو به origin اضافه کنم |
| مال خودمون رو وارد میکنیم | 60. از ما یوزر و پسوورد میخواد و بعد اکی میده |
| git pull origin master | 61. حالا داخل خود گیت هاب ما در اوریجین و برنچ مستر یه تغییری می دیم و کامیت می کنیم. حالا اونو به گیت اضافه میکنیم. |
| Git remote  اگر در حال حاضر ریموتی رو کلون کرده باشیم میزنه origin. | 62. فایل )های( ریموت رو نگاه میکنیم |
| اگر هنوز هیچ فایل ریموتی نداشته باشیم، میزنیم:  Git remote add origin (link) | 63. روش دوم برای کلون کردن چیه؟ |
| Git remote -v | 64. بیشتر راجع به اون ریموت به ما توضیح بده |
| نکته: قبل پوش کردن باید تمام فایل هارو به کامپیوتر خودمون منتقل کنیم پس اول باید اینو بزنیم وگرنه ارور میده:  Git pull origin master  و بعد:  Git push -u origin master | 65. حالا میگیم برنچ مستر منو بفرست روی origin که الان آدرسش رو دادی یعنی وصل شو به github |
|  | 66. برای ایجاد conflict در گیت، ما اول یک فایل رو داخل گیت هاب تغییر داده و کامیت میکنیم. |
| Git status | 67. حالا وضعیت گیت رو در گیت بش چک میکنیم که به ظاهر الان از اون تغییر خبر نداره چون ما pull ش نکردیم. |
| Code .  Git add -A  Git commit -m ‘to create conflict’ | 68. قبل از پول کردن، داخل گیت بش میایم فایلمون رو باز میکنیم. همون خط رو تغییر میدیم. استیج میکنیم و کامیت میکنیم. |
| Git push origin master  ارور میده که rejected. من نمیتونم پوش کنم شاید بهتر باشه تو اول یه git pull بکنی و ببینی چه تغییری داده اون یارو بعد پوش کنی | 69. حالا پوشش میکنیم به مستر که ارور بده.  چرا تا قبلش نمیداد؟ چون هنوز این دوتا conflict همدیگرو meet نکرده بودن. مثلا مامانت و عمت باهم یه تضاد فکری دارن، خو تا وقتی که همدیگرو نبینن که نمیگیرن گیس هم! اینم مثل همونه چون ما تغییرات داخل گیت هاب رو هنوز pull نکرده بودیم! |
| Git pull  میگه اکی من سعی کردم گیج مرج کنم اما نتونستم! | 70. حالا اون تغییر اولیه که داخل گیت هاب دادیم رو میاریم داخل گیت بش که ببینیم اصلا چی باعث این conflict شده.  )وقتی میخوایم دعوا رو حل کنیم میگیم خب اقا یه طرف اول شروع کنه مشکلشو بگه تا با مال خودمون مقایسه کنیم و در نهایت یکی کوتاه بیاد تا مشکل حل شه (صلوات) !) |
| Git status  میگه both modified | 71. وضعیت گیت رو چک میکنیم |
| Code .  این >>>>>> تا ====== تغییراتی بوده که من دادم و نتونستم پوش کنم چون ارور داده. اون چیزی ک الان روی سروره و اون یکی یارو اول پوش کرده بوده از ====== تا <<<<<< بوده . حالا ما میبنیم مال اون بهتر بوده پس میام خط های خودمو بعلاوه ی اون نشانه های = و > رو پاک میکنم و فقط خط اون یارو رو میذارم. | 72. حالا فایل رو مجدد داخل ادیتور خودمون باز میکنیم و اون تغییری که دادیم رو میکنیم عین روز اولش یعنی عین همون تغییر اولی که داخل گیت هاب دادیم |
| Git add -A  Git commit –m ‘last change’ | 73. حالا تغییرات اخیر هم داخل گیت بش کامیت میکنیم. |
| Git push origin master  هم یعنی اکی شدهCompleted وقتی میگه | 74. حالا به ما میگه اوکیه فقط تغییراتت یه پله بالاتر از origin هست اینو درست کن لطفا. |
| اینو بلد نیستم کامدنش چیه واسه خودمم سواله. | 75. فایل برای مثال index.html را به اولین کامیت برمیگردونیم یعنی اولین ورژن قبل اینکه هیچ تغییری توش بدیم. |
| تگ میزنیم!  (خیلی تاثیر گذار بود واقعا) | 76. ما برای شناسایی ورژن یا نسخه ها در گیت چیکار میکنیم؟ |
| Git tag –d (the name of the tag) | نکته: برای پاک کردن تگ ها: |
| Git log | 77. تاریخچه ی کامیت هارو نگاه میکنیم |
| Git show + (the unique no. of the commit)  e.g. Git show 2aad4…………….b4cc9425d9b2 | 78. چجوری جزئیات کامیت هارو نگاه میکنیم؟ |
| با تگ ها | 79. ما چجوری با گیت، ورژن هارو کنترول میکنیم؟ |
| برای دیدن تگ ها میزنیم git tag و اگر رفت خط بعدی یعنی هنوز هیچ تگی نداریم. | 80. نگاه میکنیم تا الان چند تا تگ داریم |
| git tag -a Firstv -m 'this is the first version' | 81. برای یکی از کامیت ها یه تگ میزنیم |
| باید اول از همه بزنیم git log که تمام تاریخچه ی کامیت ها بالا بیاد. بعد برای انتخاب اون کامیت مد نظر باید بزنیم  git show فلان که مشخص بشه داریم برای کدوم کامیت تگ میزنیم بعد هم که خیلی راحت برای تگ زدن از این کد استفاده میکنیم.  git tag -a Firstv -m 'this is the first version' | 82. چجوری می فهمیم داریم برای کدوم کامیت تگ میزنیم؟ |
| اول git log میزنیم که تاریخچه ی کامیت ها بیاد بالا.  بعد اون کامیتی که میخوایم رو فقط چند تا کد اولشو کپی میکنیم بعد  $ git tag -a SecondV 2aad40b9ef6  بعدش ادیتور برای ما باز شده و میگه مسیج هم وارد کن و ادیتور رو ببند. | 83. روش دوم تگ زدن رو توضیح بدید. |
| این فقط برای اون کامیت هاییه که قبلا تگ نداشتن اگر به اونایی که قبلا تگ داشتن بگیم بیا باز این تگ ارور میده که این تگ already exists اما میشه به یک ورژن چند تا تگ هم داد ولی نمیدونم دقیقا چجوری. | 84. اگر به کامیتی که قبلا تگ داشته مجدد تگ بدیم چه اتفاقی میوفته؟ |
| Git tag -l “v\*”  نکته: دقت کنید که git در واقع case-sensitiveهست | 85. میخوایم بگیم هر چی تگ که اولش v هست رو بهم نشون بده |
| git show (the name of the tag) | 86. چجوری جزئیات یه ورژن رو ببینیم؟ |
| اگر بزنیم git status هیچی رو نشون نمیده و اگر git push هم بزنیم باز چیزی رو push نمیکنه چون کلا tag اینا روش اثری نداره. | 87. آیا git status و git push روی تگ تاثیر داره؟ |
| git push origin (the name of the tag)  اگر ارور داد اینو بزنیم درست میشع  git config --global http.proxy | 88. چجوری یه تگ رو پوش میکنیم به remote directory و اگر ارور داد چیکار میکنیم؟ |
| git push origin --tags | 89. چجوری همه ی تگ هارو یه جا push کنیم؟ |
| git checkout (the name of the tag) | 90. چجوری بریم داخل یه ورژن خاصی که قبلا بهش تگ دادیم؟ |
| برای اینکار ساختن برنچ ها خیلی بهتره و تگ زدن معمولا برای مرحله ی نهایی، یعنی همون release یه محصول هست. | 100. آیا میشه داخل تگ ها یه چیزی رو کامیت کنیم؟ |

**قسمت های پایانی دوره:**

|  |  |
| --- | --- |
| 1. هش چیه؟ | وقتی میزنیم گیت لاگ میبنیم که هر کامیت یه هش داره و اون هش نمیذاره که کامیت تغییر بکنه.  Commit hashes  The long string following the word commit is called the commit hash. It's unique identifier generated by Git. **Every commit has one**, and I'll show you what they're used for shortly. Note: The “commit hash” is sometimes called a Git commit “reference” or “SHA”.  SHA = Simple Hashing Algorithm |
| 2. ما برای امضا کردن تگ ها از چی استفاده میکنیم و چطوری؟ | برای امضا کردن تگ ها باید از رمزنگاری استفاده کنیم.  چجوری؟ از طرق pgp  **Pretty Good Privacy = PGP**  بعد اونو تغییر دادن کردن این  **GPG =** **GNU Privacy Guard** |
| 3. مفهوم رمزنگاری مبتنی بر کلید چیه؟ | کلا رمز نگاری های کلیدی اینطورین که دو تا کلید داریم که مبتنی بر جفت کلیدن. هر چیزی رو که تغییر بدین فقط با این کلید قابل باز شدنه یعنی ما یه جفت کلید درست میکنیم و کلید public یا عمومی رو میدیم به همه و کلید خصوصی یا private رو برای خودمون نگه میداریم. وقتی میخوایم کامیت کنیم کامیت رو با این امضا میکنیم بعد وارد میشیم. |
| 4. میگیم هر چی کلید دارم نشونم بده | gpg --list-keys  اگر gpg نصب باشه اینارو میاره. این یعنی بهمون میگه هیچ کلیدی نداریم. |
| 5. میگیم بیا کلید رو درست کن | gpg --gen-key  بعد یه سری اطلاعات رو میدی و o میزنی حالا میگه یه رمز درست کن براش  در نهایت میگه یه کلید برات درست کردم که این شکلیه:  pub ... |
| 6. حالا باز میگیم هر چی کلید دارم نشونم بده | gpg --list-keys  میگه آره یه کلید اینجاست: |
| 7. ما داخل git config یه عالمه اطلاعات داریم. چجوری اونارو ببینیم؟ | git config --global |
| 8. میگیم بگو یوزر نیمم چیه؟ | git config --global user.name |
| 9. میگم بگو ساینینگ کی من چیه؟ | git config --global user.signingkey  اگر در جواب این هیچی نیاره اوکیه. |
| 10. **حالا میگم همه ی کلیدای مخفی خودمو نشون بده** | gpg --list-secret-keys |
| 11. حالا میگیم راجع به کلید های مخفیم توضحات بیشتری بده | gpg --list-secret-keys --keyid-format LONG |
| 12. کد یونیک کلید ما میشه کدوم قسمت؟ | 824F96B4EC47F729 |
| 13. گلوبال یعنی چی؟ | یعنی در تمام repository هایی که من داخل کامپیوتر دارم. |
| 14. حالا چجوری میتونیم signingkey درست کنیم؟ | Git config --global user.signingkey  (the unique code of our key)  e.g.  git config --global user.signingkey 824F96B…..47F729  خروجی خاصیم نمیده |
| 15. حالا یه تگ با امضای دیجیتال درست می کنیم. | در حالت عادی برای تگ کردن:  Git tag –a V2.1 –m ‘this is a new one’  اما اگر بخوایم هم تگ کنیم و هم ساین کنیم همزمان به جای a ما s میذاریم:  Git tag –s V2.1 –m ‘this is newer’  *A = annotate (add notes)*  *S = sign*  بعد از ما پسوورد همون کلید اولی هرو میخواد که درست کردیم و گفتیم برای user.sigingkey هم باید استفاده بشه = |
| 16. حالا تاریخچه ی کامیت هارو نگاه میکنیم | Git log  مثل قبله و هیچ تغییری نکرده |
| 17. حالا تگ هارو دوباره چک میکنیم | Git tag |
| 18. حالا به گیت میگیم همون تگی که ساین داشت رو نشونم بده. | Git show V2.1  Tagger:  در اینجا امضام هم به مسیجم اضافه شده.  این چیزیه که فقط با رمز public خودمون باز میشه. |
| 19. حالا میگیم تست کن ببین باز میشه یا نه!  بیا وریفایش کن. | git tag -v V2.1  *V = verify*  حالا هر کس که رمز public رو داشته باشه میتونه بره و مطمعن بشه که من اینو امضا کردم. |
| 20. ما چه چیزهایی رو در گیت میتونیم امضا کنیم؟ | تگ ها و کامیت ها |
| 21. چجوری یه کامیتو امضا کنیم حالا؟ | اول طبق معمول یه تغییری میدیم داخل پروژه که نیاز به کامیت کردن داشته باشه.  روش بدون امضا این بود:  Git add –A  Git commit –m ‘…’  برای امضا کردنش:  Git add –A  Git commit –S –m ‘…’  از ما مجدد همون رمزو میخواد:  S = sign  نکته: این s در امضا کردن تگ ها کوچک و در امضا کردن کامیت ها بزرگ است. |
| 22. مثلا میخوایم ببینیم blame چیکار میکنه؟ | Git help blame |
| 23. اول یه فایلو توشو نگاه میکنیم | Cat index.html |
| 24. بعد میخوایم ببینیم فلان خطو مثلا خط 8 رو کی نوشته؟ | Git blame index.html –L8  اولین خطی که میاره یعنی اخرین نفری که اون تغییراتو داده |
| 25. میخوایم حالا از خط 8 تا 10 رو ببینیم کی نوشته | git blame index.html -L8,10 |
| 26. حالا میگیم بگو همرو کی نوشته اصلا؟ | Git blame (file name) |
| 27. نحوه ی دیباگ کردن به کمک گیت؟ | ما یه باگ میبینیم داخل پروژه اما نمیتونیم بفهمیم از کی این باگ ایجاد شده. میخوایم زمان ایجادشو بفهمیم یعنی از بعد از کدوم کد این ایجاد شد؟  Git bisect start  Bisect = binary search commit |
| 28. مراحل دیباگ کردن: | Git bisect start  رانش میکنیم  Git bisect bad  الان بده یعنی اینی که الان توش هستیم بده  Git log  یه کامیتی که اونجا درش باگ نبوده رو کپی میکنیم اولشو  بعد به گیت میگیم باگ داخل این کامیت وجود نداشته:  Git bisect good 97aa4a…….f317006  بعد میگه خب من یه سری کارا میتونم بکنم. خودش سوییچ میکنه به یه کامیت و میگه الان چک کن ببین خوبه یا بده. خود گیت جلو عقب میره که پیدا کنه مشکل کجاست. من اینجا میدونم بده و اینجا میدونم خوب بوده. گیتم گفت پس بیا اینو چک کن. حالا مثلا من برنامم رو کامپایل میکنم و ران میکنم ولی بازم بده پس میزنم  Git bisect bad  بعد گیت میگه این چی؟ یه کامیت میده چک کنیم. میگم مثلا نه این خوبه مشکلی نداره  Git bisect good  در نهایت میگه مشکل اینه پس. |
| 29. گیت لب چیه؟ | گیت لب هم یه سایته مثل گیت هاب. اجازه میده پروژه رو private تعریف کنیم.  میشه خود گیت هم نصب کرد مثلا روی سیستم شرکت نصب باشه.  مراحل:  <https://about.gitlab.com/get-started/>  بیشتر پروژه ها ولی داخل گیت هاب هستن. نباید فعلا بدون فیلتر شکن وارد گیت لب شد چون ظاهرا بخاطر تحریم ها امکان پرایویت کردن کدها وجود نداره. |
| 30. فورک یعنی چی؟ | اگر یه پروسه خودشو فورک کنه یعنی اون پروسه میشه 2 تا.  در گیت لب اگر پروژه ای رو فورک کنیم یعنی یدونه کپی بکن داخل اکانت خودم.  یه پروژه پیدا میکنیم و فورک رو میزنیم.  بعد که فورک کردیم داخل اکانت خودمون میایم فایل رید می رو ادیت میکنیم.  میشه به جای اینکه داخل گیت لب بیایم ادیت کنیم پروژه رو کلون کنیم و بعد با ایدتورمون بازش کنیم ولی الان داخل همون گیت لب ادیت میزنیم و بعد با کامیت مسیج، کامیتش میکنیم.  این یعنی انلاین ما یه ادیتی زدیم.  حالا میتونیم به اون سازندش بگیم بیا این کامیت منو لطفا pull کن.  میگم درخواست مرج بده:  Create merge request  میگه سورس برنچت چیه؟  مثلا  Jadidjadi/AI  بعد که  Compare branched and continue  رو بزنیم برای اون یارو پیام میاد که یکی درخواست داده شما یه چیزی رو pull کنی. |
| 31. برای تمرین: | <https://github.com/jadijadi/gittutorial> |