**گردش کار (Workflow) در Git**

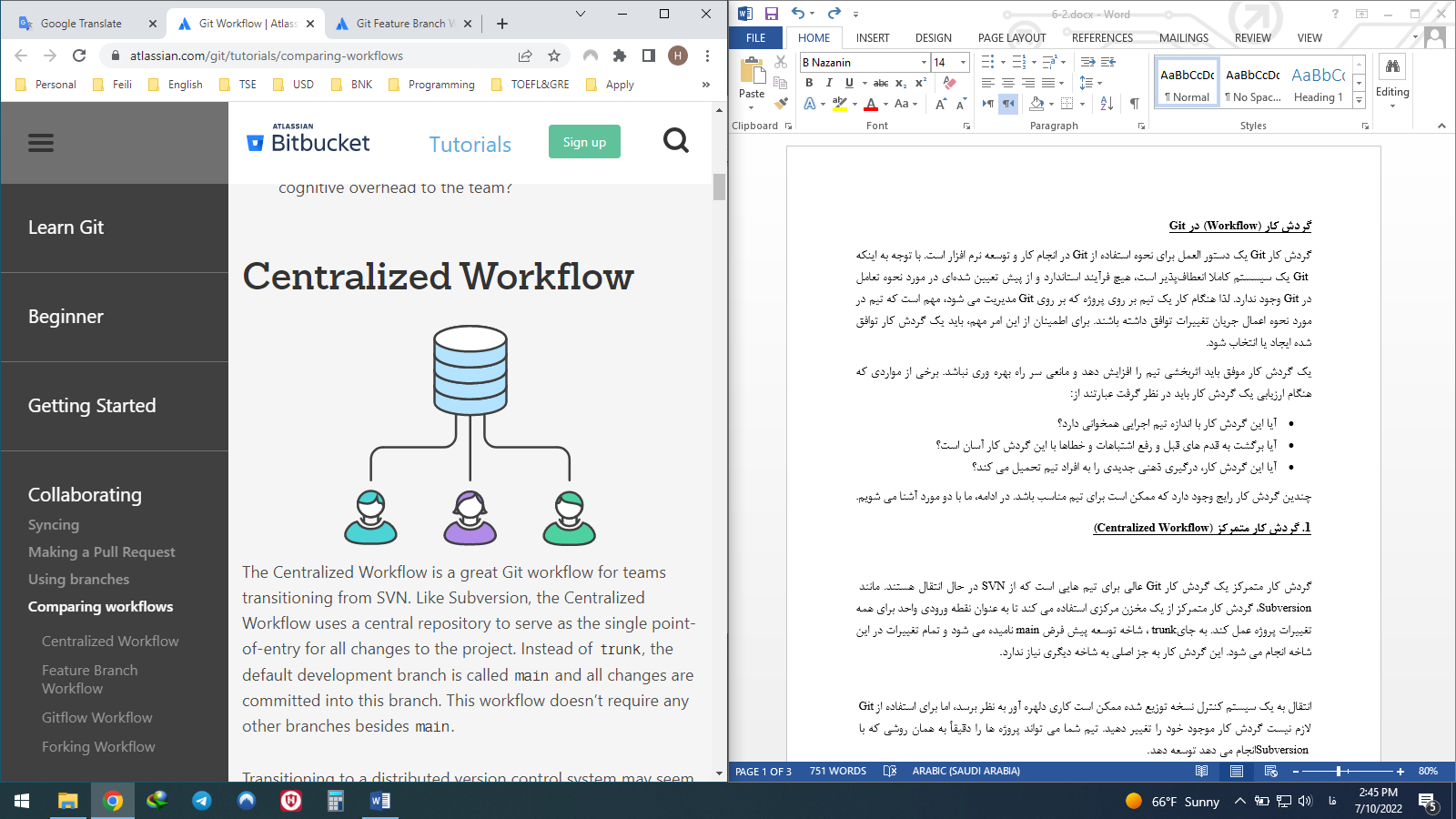
گردش کار Git یک دستور العمل برای نحوه استفاده از Git در انجام کار و توسعه نرم افزار است. با توجه به اینکه Git یک سیسستم کاملا انعطاف‌پذیر است، هیچ فرآیند استاندارد و از پیش تعیین شده­ای در مورد نحوه تعامل در Git وجود ندارد. لذا هنگام کار یک تیم بر روی پروژه که بر روی Git مدیریت می شود، مهم است که تیم در مورد نحوه اعمال جریان تغییرات توافق داشته باشند. برای اطمینان از این امر مهم، باید یک گردش کار توافق شده ایجاد یا انتخاب شود.

یک گردش کار موفق باید اثربخشی تیم را افزایش دهد و مانعی سر راه بهره وری نباشد. برخی از مواردی که هنگام ارزیابی یک گردش کار باید در نظر گرفت عبارتند از:

* آیا این گردش کار با اندازه تیم اجرایی همخوانی دارد؟
* آیا برگشت به قدم های قبل و رفع اشتباهات و خطاها با این گردش کار آسان است؟
* آیا این گردش کار، درگیری ذهنی جدیدی را به افراد تیم تحمیل می کند؟

چندین گردش کار رایج وجود دارد که ممکن است برای تیم مناسب باشد. در ادامه، ما با دو مورد آشنا می شویم.

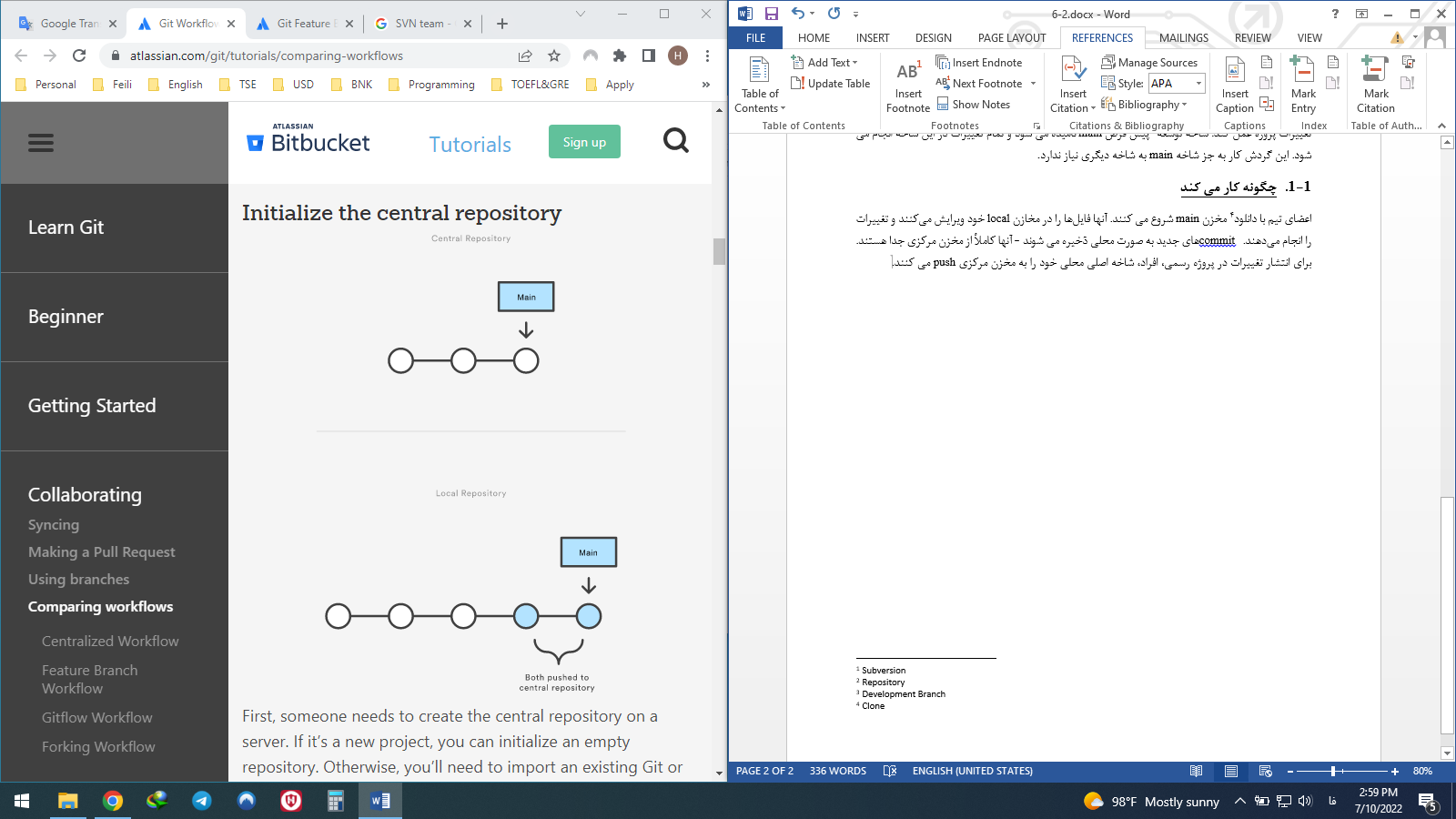
**1. گردش کار متمرکز (Centralized Workflow)**



یک گردش کار متمرکز معمولاً برای تیم هایی که از SVN[[1]](#footnote-1)به Git مهاجرت کردند و همچنین تیم های کوچک مناسب است. گردش کار متمرکز از یک مخزن[[2]](#footnote-2) مرکزی استفاده می کند تا به عنوان نقطه ورودی واحد برای همه تغییرات پروژه عمل کند. شاخه توسعه[[3]](#footnote-3) پیش فرض main نامیده می شود و تمام تغییرات در این شاخه انجام می شود. این گردش کار به جز شاخه main به شاخه دیگری نیاز ندارد.

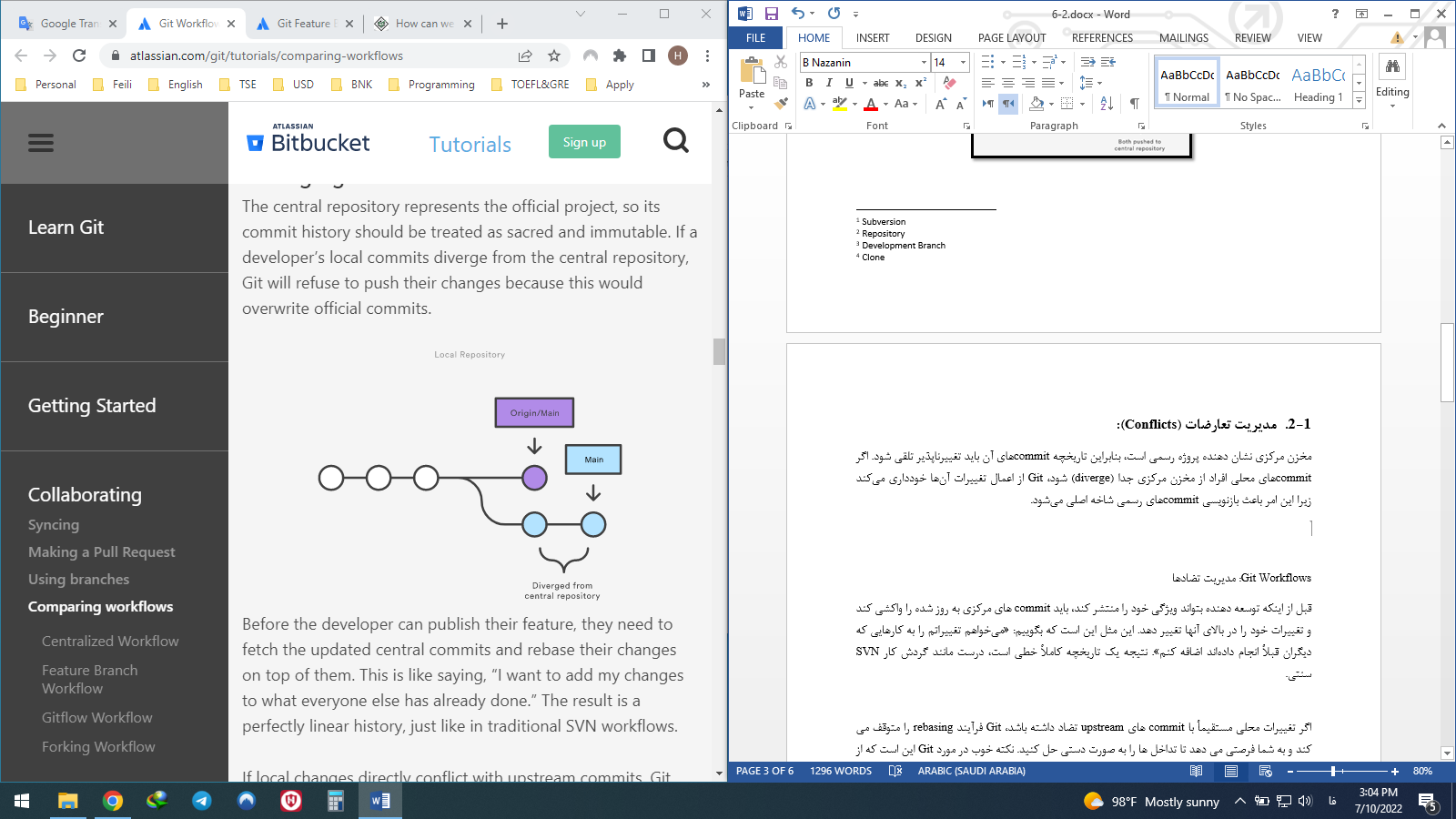
* 1. **چگونه کار می کند**

اعضای تیم با دانلود[[4]](#footnote-4) مخزن main شروع می کنند. آنها فایل‌ها را در مخازن local خود ویرایش می‌کنند و تغییرات را انجام می‌دهند. commit های جدید به صورت محلی ذخیره می شوند - آنها کاملاً از مخزن مرکزی جدا هستند. برای انتشار تغییرات در پروژه رسمی، افراد، شاخه اصلی محلی خود را به مخزن مرکزی push می کنند.



* 1. **مدیریت تداخلات (Conflicts):**

مخزن مرکزی نشان دهنده پروژه رسمی است، بنابراین تاریخچه commitهای آن باید تغییرناپذیر تلقی شود. اگر commitهای محلی افراد از مخزن مرکزی جدا (diverge) شود، Git از اعمال تغییرات آن‌ها خودداری می‌کند زیرا این امر باعث بازنویسی commit‌های رسمی شاخه اصلی می‌شود.



قبل از اینکه فرد بتواند کار خود را منتشر کند، باید commitهای مرکزی به روز شده را fetch کند و تغییرات خود را بالای آنها اعمال کند. یعنی بگوییم: «می‌خواهم تغییراتم را به کارهایی که دیگران قبلاً انجام داده‌اند اضافه کنم» که نتیجه آن یک تاریخچه کاملاً خطی است.

اگر تغییرات local مستقیماً با commitهای upstream تضاد داشته باشد، Git فرآیند rebasing را متوقف می کند و به شما فرصتی می دهد تا تداخل ها را به صورت دستی حل کنید. نکته خوب در مورد Git این است که از همان status git و دستورات git add برای ایجاد commit و حل تداخلات استفاده می کند. این امر مدیریت ادغام ها برای افراد جدید آسان می کند.

* 1. **مثال از گردش کار متمرکز**

یک مثال کلی در مورد نحوه همکاری یک تیم کوچک معمولی با استفاده از این گردش کار را مشاهده خواهیم کرد.

John روی feature خود کار می کند و می تواند در مخزن local خود ویژگی‌هایی را با استفاده از فرآیند Git commit توسعه دهد: edit، stage، و commit. به یاد داشته باشید از آنجایی که این دستورات commit های محلی را ایجاد می کنند، جان می تواند این فرآیند را هر چند بار که بخواهد بدون نگرانی در مورد آنچه در مخزن مرکزی می گذرد تکرار کند.

هم چنین Mary در حال کار بر روی feature در مخزن محلی شخصی با استفاده از همان فرآیند edit، stage، و commit است. او مانند John، اهمیتی نمی‌دهد که در مخزن مرکزی چه می‌گذرد، و واقعاً برایش مهم نیست که John در مخزن محلی‌اش چه می‌کند، زیرا همه مخازن محلی خصوصی هستند.

پس از آن John ویژگی خود را منتشر می کند. هنگامی که جان ویژگی خود را تمام کرد، باید commitهای محلی خود را در مخزن مرکزی منتشر کند تا سایر اعضای تیم بتوانند به آن دسترسی داشته باشند. او می تواند این کار را با دستور git push انجام دهد. از آنجایی که مخزن مرکزی از زمانی که John آن را دریافت کرده است، به‌روزرسانی نشده است، این انتشار منجر به هیچ تداخلی نمی‌شود و push همانطور که انتظار می‌رود کار خواهد کرد.

حال Mary سعی می کند feature خود را منتشر کند اما از آنجایی که تاریخچه محلی او از مخزن مرکزی جدا شده است، Git با یک پیام خطای نسبتاً واضح درخواست را رد می کند:

error: failed to push some refs to '/path/to/repo.git'  
hint: Updates were rejected because the tip of your current branch is behind  
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')  
hint: before pushing again.  
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

این مشکل Mary را از ارسال commit رسمی باز می دارد. او باید به‌روزرسانی‌های John را در مخزن خود بیاورد، آنها را با تغییرات محلی خود ادغام کند و سپس دوباره امتحان کند.

حال Mary باید commitهای خودر را در بالای تعهد(های) John اعمال کند. Mary می تواند با دستور زیر برای گنجاندن تغییرات بالادستی در مخزن خود استفاده کند:

git pull --rebase origin main

عبارت rebase به Git می‌گوید که تمام commitهای Mary را پس از همگام‌سازی با تغییرات مخزن مرکزی به ابتدای شاخه اصلی منتقل کند. اگر عبارت rebase را فراموش کنید، pull همچنان کار می کند، اما هر بار که نیاز به همگام سازی با مخزن مرکزی باشد، با یک "merge commit" اضافی مواجه می شوید. برای این نوع گردش کار، همیشه بهتر است به جای ایجاد یک "merge commit" ، rebase کنید.

https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow

1. Subversion [↑](#footnote-ref-1)
2. Repository [↑](#footnote-ref-2)
3. Development Branch [↑](#footnote-ref-3)
4. Clone [↑](#footnote-ref-4)