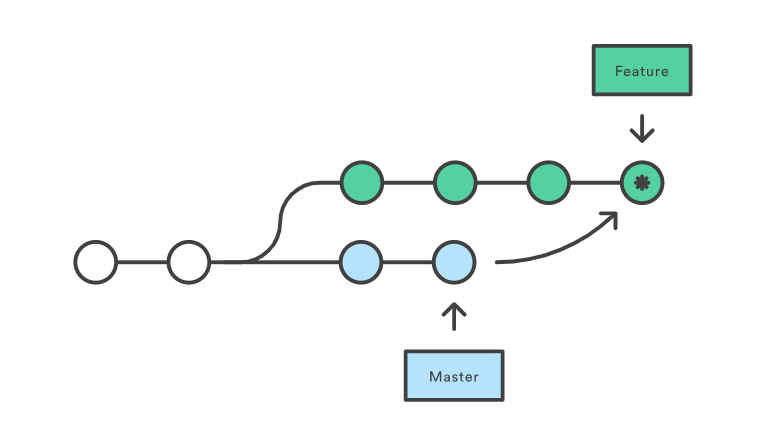
**Merge & Rebase**

Git Merge و Git Rebase برای هدف مشابهی کار می‌کنند. این دو طراحی شده‌اند تا تغییراتی از شاخه‌های مختلف را در یک شاخه ادغام کنند. با این که هدف نهایی آن‌ها یکی است، اما این دو متد به روش‌هایی مختلفی به این هدف می‌رسند و دانستن تفاوت آن‌ها، همینطور که تبدیل به یک توسعه دهنده بهتر می‌شوید خوب است

****

**Git Merge**

کردن یکی از شیوه‌های رایج برای استفاده از سیستم کنترل نسخه است. چه این شاخه‌ها برای آزمایش، چه بر طرف کردن باگ‌ها یا چه دلایل دیگر ساخته شده باشند، باز هم merge کردن محتویات یک شاخه منبع را می‌گیرد و آن‌ها را با یک شاخه هدف ادغام می‌کند. در این روند، فقط شاخه هدف تغییر کرده، و تاریخچه شاخه منبع به طور مشابه باقی می‌ماند.



نکات مثبت

* ساده و آشنا
* تاریخچه کامل و ترتیب وقوع را حفظ می‌کند
* زمینه یک شاخه را نگه می‌دارد

نکات منفی

* تاریخچه commitها می‌تواند با تعداد زیادی commitهای ادغام شده آلوده شود
* خطایابی با استفاده از git bisect می‌تواند سخت‌تر شود

نحوه استفاده از Merge با استفاده از دستورات زیر

|  |
| --- |
| $ git checkout feature |
|  |

|  |  |
| --- | --- |
| $ git merge master | |
|  |

|  |  |
| --- | --- |
| (یا) | |
|  |

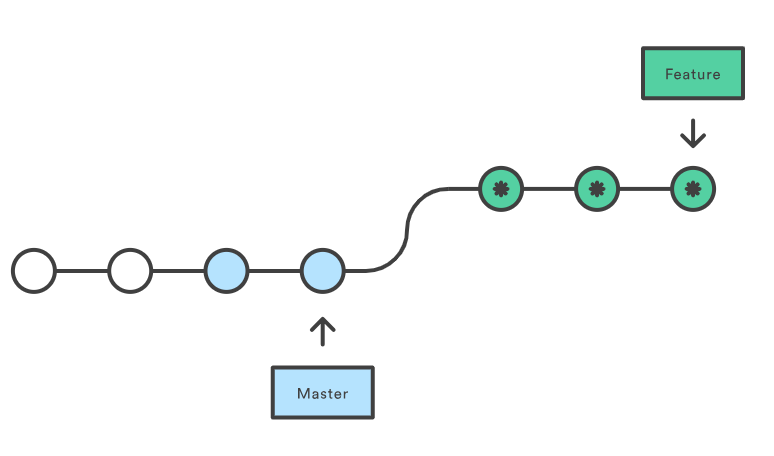
$ git merge master feature

این کار یک Merge commit جدید در branch feature خواهد ساخت که تاریخچه هر دو شاخه را نگه می‌دارد

Git Rebase

Rebase یک راه دیگر برای ادغام تغییرات از یک شاخه به یک شاخه دیگر است. Rebase تمام تغییرات را در یک patch فشرده‌سازی کرده، و سپس patch مورد نظر را در یک شاخه هدف ادغام می‌کند.Rebase‌ کردن بر خلاف merge کردن، تاریخچه را مسطح‌سازی می‌کند؛ زیرا این ابزار کار تمکیل شده را از یک شاخه به یک شاخه دیگر منتقل می‌نماید. در این روند، تاریخچه مذکور ناخواسته از بین می‌رود.

Rebaseها، نحوه منتقل شدن تغییرات از بالای سلسله مراتب به سمت پایین و mergeها هم نحوه جریان یافتن آن‌ها به سمت بالا هستند.



نکات مثبت

* یک تاریخچه پیچیده را ساده‌سازی می‌کند
* دستکاری یک commit تنها ساده است
* از وجود نویز در ropeهای مشغول که شاخه‌های مشغولی دارند، جلوگیری می‌کند
* با تبدیل کردن commitهای حد واسط به یک commit تنها که می‌تواند برای گروه‌های توسعه دهنده کاربردی باشد، آن‌ها را مرتب می کند

نکات منفی

* له کردن یک ویژگی در قالب چند commit، می‌تواند زمینه را مخفی کند
* Rebase کردن مخازن عمومی می‌تواند در هنگام کار به عنوان یک تیم، خطرناک باشد.
* کار بیشتری مورد نیاز است: استفاده از rebase برای بروز نگه داشتن شاخه ویژگی خود
* اگر شما به طور ناصحیحی rebase‌ می‌کنید و به طور غیر عمدی تاریخچه را بازنویسی می‌کنید، این موضوع می‌تواند به مشکلات جدی ختم شود؛ پس مطمئن شوید که می‌دانید چه کاری انجام می‌دهید

نحوه استفاده از Merge با استفاده از دستورات زیر:

Branch feature را با استفاده از این دستورات، به یک شاخه اصلی rebase کنید

|  |
| --- |
| $ git checkout feature |
|  |

$ git rebase master

ین کار branch feature را به کلی به بالای branch mastre منتقل می‌کند. این کار با بازنویسی تاریچه پروژه، و با ساخت commitهای جدید در شاخه اصلی انجام می‌شود.

**از کدام مورد استفاده کنیم**

عمومی کردن و تصمیم‌گیری درباره یکی از این موارد سخت است؛ زیرا هر کدام از آن‌ها متفاوت می‌باشند. اما بالاخره باید از جایی شروع کرد.

تیم‌های کاری وقتی که شروع به تنظیم کردن سیاست‌های rebase علیه merge خود می‌نمایند، چندین سوال را باید در نظر داشته باشند. زیرا به نظر می‌رسد که یکی از این دو استراتژی کاری، بهتر از دیگری نیست. این مسئله به تیم شما بستگی دارد.

سطح rebase کرد و رقابت Git را در سازمان خود در نظر بگیرید. درجه‌ای که بر حسب آن سادگی rebase کردن در مقایسه با قابل ردیابی بودن و تاریخچه merge کردن برایتان ارزش دارد را تعیین کنید.

در آخر، تصمیم‌گیری درباره merge کردن و rebase‌ کردن باید در زمینه یک استراتژی شاخه واضح در نظر گرفته شود. یک استراتژی شاخه موفقیت‌آمیز، بر حسب سازمان‌دهی تیم‌های شما طراحی شده است