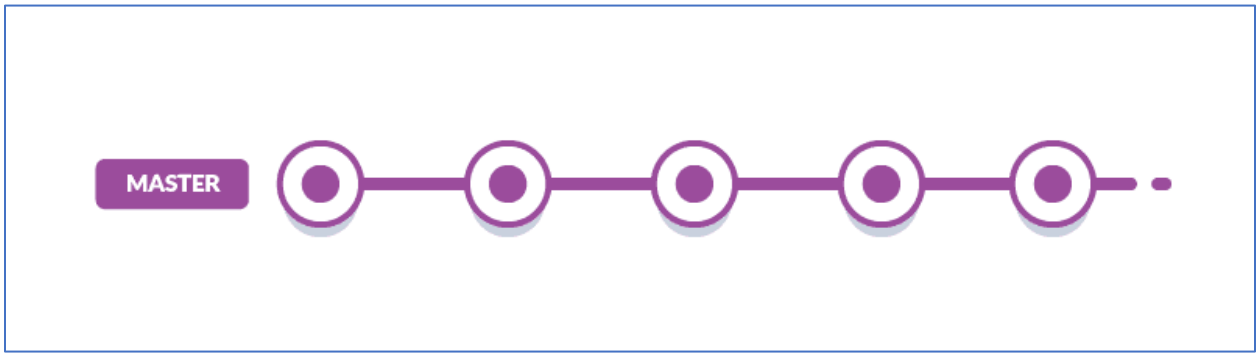
5 نوع از gitworkflow ها که میتواند به شما کمک کند که کد بهتری را ارائه دهید

همان طورکه میدانید، هر تیم workflow مخصوص به خود را دارد بسته به نوع پروژه، اندازه شرکت، ترجیحات تیم و خیلی موارد دیگر. هرچه تیم بزرگتر، کنترل کردن سخت تر است.

مشکلات با conflicts بیشترو تاریخ انتشار نسخه ها نیازمند به تعویق خواهد بود. اولویت ها در طول پروژه دائماً تغییر میکند.

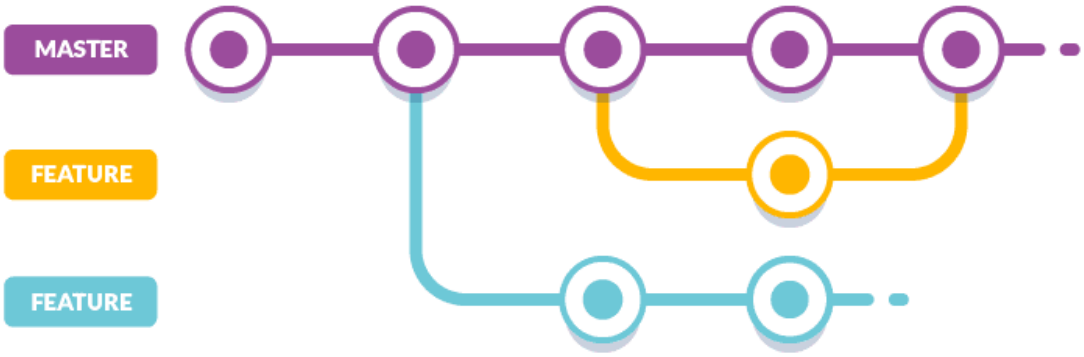
تطبیق Git اولین گام برای حل این مسائل است، زیرا می توانید آن را عملاً در هر نوع گردش کاری اعمال کنید.



شکل 1) Basic Workflow

محبوب ترین گردش کار توسعه Git و مرحله ورود هر پروژه.

ایده ساده است: یک مخزن مرکزی وجود دارد. هر توسعه دهنده مخزن را شبیه سازی می کند، به صورت محلی روی کد کار می کند، یک commit با تغییرات ایجاد می کند، و آن را به مخزن مرکزی push میدهد تا توسعه دهندگان دیگر آن را pull کنند و در کار خود استفاده کنند.



Feature branch workflow (شکل 2

گردش کار اصلی برای توسعه یک وب سایت ساده عالی است. با این حال، هنگامی که دو توسعه دهنده شروع به کار بر روی دو عملکرد جداگانه در یک پروژه می کنند، مشکلات ظاهر می شوند.

فرض کنید یکی از توسعه دهندگان عملکرد خود را به پایان رسانده و می خواهد آن را منتشر کند. با این حال، آنها نمی توانند این کار را انجام دهند زیرا ویژگی دوم آماده نیست. انتشار در این لحظه می تواند منجر به به هم ریختن همه چیز شود (حداقل).

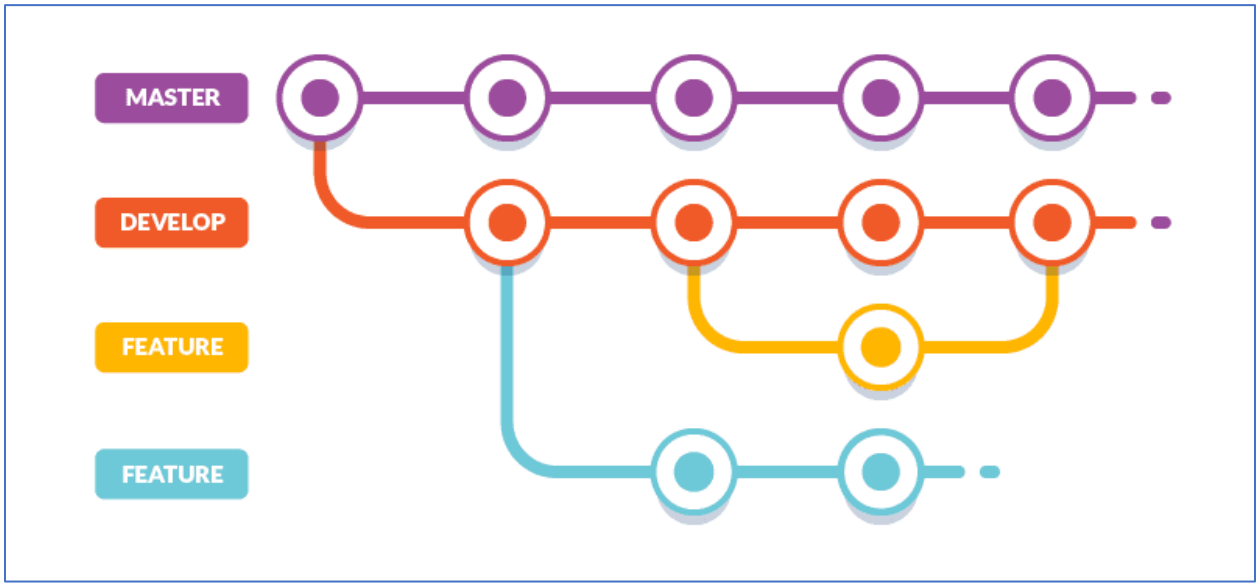
اینجاست که شاخه ها - ویژگی اصلی Git - به کار می آیند. شعبه ها مسیر مستقل توسعه پروژه هستند. برای هر عملکرد جدید، یک شاخه جدید باید ایجاد شود، که در آن ویژگی جدید توسعه یافته و آزمایش شود. پس از آماده شدن ویژگی، شاخه را می توان به شاخه اصلی ادغام کرد تا بتوان آن را به سرور تولید منتشر کرد.

شاخه های ویژه و درخواست های ادغام

گردش کار شاخه ویژگی فرض می کند که همه توسعه دهندگان در تیم از دانش و موقعیت یکسانی برخوردار هستند. با این حال، در تیم های بزرگتر، همیشه نوعی سلسله مراتب در شرکت وجود دارد.

در این مورد، می‌توانید از درخواست‌های ادغام و مجوزهای push استفاده کنید که به شما امکان می‌دهد push کردن را به شاخه‌های انتخابی در مخزن محدود کنید و کنترل کامل روی کد را حفظ کنید.

قبل از اینکه یک شاخه با Master ادغام شود، باید تأیید شود و برای خطا بررسی شود. توسعه دهندگان junior می توانند یک درخواست ادغام ایجاد کنند و آن را به یکی از Senior ها اختصاص دهند تا بتوانند کد را بررسی کنند و نظرات خود را بگذارند. اگر همه چیز درست باشد، درخواست پذیرفته می شود و شعبه ادغام می شود.



شکل 3) Gitflow workflow

هرچه پروژه بزرگتر باشد، به کنترل بیشتری بر روی زمان و زمان انتشار نیاز دارید. پروژه های شما به تست های واحد و ادغام بیشتری نیاز دارند که اکنون ساعت ها طول می کشد تا اجرا شوند. با این حال، شما چنین آزمایش‌هایی را در شاخه‌هایی که ویژگی‌ها در آن توسعه یافته‌اند، اجرا نمی‌کنید.

این را می توان با Gitflow حل کرد، یک گردش کار توسعه Git که توسط Vincent Driessen در سال 2010 ابداع و توصیف شد.

این گردش کار از دو شاخه موازی طولانی مدت استفاده می کند:

- Master

  فقط برای انتشار استفاده می شود

- Develop

  ایجاد شده از Master، این خانه تمام ویژگی های کامل و پایدار است که برای نسخه بعدی آماده شده است

وقتی روی یک ویژگی جدید شروع به کار می کنید، یک شاخه ویژگی جدید از Develop ایجاد کنید. هر تعداد شاخه ویژگی به صورت موازی که می خواهید و نیاز دارید ایجاد کنید. وقتی کار تمام شد و ویژگی تست شد، کد را دوباره با Develop ادغام کنید.

سپس، زمانی که زمان انتشار فرا رسید، ویژگی های جدید را از شاخه Develop در یک شاخه Release جدید جدا کنید. اطمینان حاصل کنید که انتشار به خوبی تست شده و پایدار است.

بسته به ویژگی پروژه شما، ممکن است ایده خوبی باشد که نسخه RC (Release Candidate) نرم افزار خود را برای عموم منتشر کنید. هنگامی که انتشار پایدار است و همه‌ی پیچیدگی‌ها برطرف می‌شوند، شاخه انتشار خود را به Master ادغام کنید و در مرحله تولید قرار دهید!

انشعاب گردش کار Forking workflow

درست مانند دریای آزاد، در متن باز همه چیز به کاپیتان بستگی دارد. در شرایط نرم افزاری، صاحب مخزن تصمیم می گیرد که چه کسی می تواند به مخزن push کند. اگرچه ایده منبع باز این است که همه می توانند در پروژه مشارکت داشته باشند، اما فکر کردن به اینکه لینوس تورووالدز دسترسی نامحدود به مخزن پروژه لینوکس را به هر کسی که مشتاق به دستکاری کدش است، سخت است، اینطور نیست؟

این مشکل با fork ها حل می شود: هر زمان که یک توسعه دهنده بخواهد چیزی را در یک پروژه منبع باز تغییر دهد، مستقیماً روی مخزن پروژه کار نمی کند. درعوض، آن را fork می‌کنند و در واقع یک کپی از کل مخزن ایجاد می‌کند. سپس، توسعه دهنده آزاد است تا بر روی ویژگی جدید به هر شکلی که می خواهد کار کند. شایان ذکر است که فورکینگ امکان ایجاد نسخه های سفارشی از اجزای خاص را که برای برنامه های خاص تنظیم شده اند نیز باز می کند. یک توسعه‌دهنده یا یک شرکت می‌تواند یک مخزن را جدا کند و کد را در مسیری کاملاً جدید که توسط مالک(ها) مجاز نیست، ببرد.

با این حال، در بیشتر موارد، وقتی کار انجام می‌شود، یک درخواست کشش ایجاد می‌شود که تغییراتی را که توسعه‌دهنده در فورک خود معرفی کرده است با وضعیت مخزن فورک شده مقایسه می‌کند. از آنجا، جامعه و صاحبان پروژه می توانند تغییرات را بررسی، بحث و آزمایش کنند. فراخوان نهایی در دستان کاپیتان پروژه و معاونان آنها باقی می ماند.

Website: https://buddy.works/blog/5-types-of-git-workflows