سناریو معمول گیت به این شکل هست: وقتی با گیت شروع میکنید روی برنچ **master** هستید و در ادامه میتونید چندین برنچ مختلف برای خودتون بسازید ، مثلا برنچ **update** یا **feature** و توی اون **branch** ها کار کنید و وقتی بعد از اتمام کار هم **merge** میکنید و دوباره به **master** و یا **development** میرسید.

مشکلاتی که این بین پیش میاد:  
اگه چندین نفر همزمان دارن روی یک پروژه کار میکنند یا زمانی که تعداد برنچ های کارتون زیاد میشه ، مدیریت و نگه داری گیت سخت میشه ! ممکنه جایی که کار میکنید یه قراردادهایی برای این کار داشته باشند و یا حتی یک نفر به عنوان **git** **manager** که کارش فقط کنترل گیت پروژه هست داشته باشند. اما با استفاده از ابزار گیت فلو میتونید خودتون به بهترین شکل، پروژه رو جلو ببرید.

سناریو کار گیت تو بیشتر شرکت ها به این شکل هست که:  
شاخه **master** : که کدهای پروژه (در حال اجرا) قرار دارند و ممکنه همه بهش دسترسی نداشته باشند و طبق قرارداد هم کسی اجازه کامیت و پوش کردن بهش رو نداره !  
شاخه **develop** :‌ کار بر روی پروژه از اینجا شروع میشه و توسعه دهنده ها میتونن کدهاشون رو به این قسمت پوش کنند اما این همه ماجرا نیست. اگه قرار باشه ۵ نفر همزمان روی این شاخه کار کنند ممکنه **conflict** به وجود بیاد پس شاخه دیگه ای هم لازم داریم:  
شاخه **feature**: فرض کنید نیما قراره رو **login** کار کنه و هادی روی **home** پس به راحتی میتونن هر کدوم شاخه های خودشون رو داشته باشند به این شکل که **feature/login** و **feature/home**. در نهایت وقتی کار تموم شد این شاخه با شاخه **develop** مرج میشه و در صورتی که لازم باشه بعدا با شاخه **master** هم **merge** میشه!  
شاخه **hotfix**:‌ فرض کنید هادی و نیما دارن روی فیچرهایی که گفته شده کار میکنن که مدیر پروژه یهو متوجه یه مشکلی تو پروژه در حال اجرا ( که طبیعاتا روی شاخه **master** هست) میشه و میخواد سریعا این مشکل حل بشه. پس به یه توسعه دهنده دیگه میگه این موضوع رو به سرعت حل کن!‌  
این جا شاخه **hotfix** به وجود میاد و بدون اینکه با فیچر ها دخالتی داشته باشه از مستر و یا دولوپ یه شاخه میگیره و شروع میکنه به رفع مشکل! بعد از اینکه مشکل حل شد با **master** و یا **develop** مرج میشه. هادی و نیما هم بدون مشکل به کار خودشون ادامه میدن !  
شاخه **release**: هم که از اسمش مشخصه! مربوط به کنترل تگ های برنامه است و مثلا وقتی پروژه به جایی رسید که قراره **release** بشه یک تگ بهش میخوره مثلا 2.4.1 و ...

خوب میبینید که مدیریت این همه شاخه و **merge** کردنشون کار ساده ای نیست!  
اینجاست که **git-flow** مثل سوپر من وارد میشه و کنترل ماجرا رو به دست میگیره. به این شکل که تو هر جایی از پروژه که باشید میتونید پروژتون رو با گیت فلو ادامه بدید !

همه شاخه هایی که لازم داریم و کار **merge & delete** شاخه های مختلف رو خود **git-flow** برای ما انجام میده !  
به این شکل که وقتی راه اندازیش میکنید ، روی **local** خودتون این شاخه هارو میسازه و بعد با چند تا دستور ساده میتونید به راحتی شاخه **fearture** دلخواهتون رو بسازید!

