عنوان: **#9 آموزش سیستم مدیریت کد Git : کار به صورت remote**
نویسنده: حسام امامی
تاریخ: ۱۳۹۱/۰۷/۱۶ ۲۱:۳۴
آدرس: www.dotnettips.info
گروه‌ها: Git

تا اینجا هر آنچه درباره git آموختیم در رابطه با عملکرد git به صورت محلی بود. اما یکی از ویژگی‌های سیستم‌های توزیع شده، امکان استفاده از آن‌ها به صورت remote می‌باشد.
در مورد git تفاوت چندانی بین سرورها و کلاینت‌ها وجود ندارد. تنها تفاوت، نحوه‌ی پیکربندی سرور است که این امکان را می‌دهد تا چندین کلاینت به صورت همزمان به آن متصل شده و با repository آن کار کنند. اما عملا تفاوتی بین repository موجود در کلاینت و سرور نیست.
تذکر ۱: در این مقاله از وب سایت github برای توضیح مثال‌ها استفاده شده است. github قدیمی‌ترین و قدرتمندترین وب سایت برای مدیریت repositoryهای git است. اما اجباری در انتخاب آن نیست؛ زیرا انتخاب‌های فراوانی از جمله bitbucket نیز وجود دارد.
تذکر ۲: نام مستعار origin اجباری نیست؛ اما از آن جهت که نام پیش فرض است، در اکثر مثال‌ها و توضیحات استفاده شده است.
قبل از شروع مبحث بهتر است کمی درباره‌ی پروتکل‌های ارتباطی پشتیبانی شده توسط git صحبت کنیم:
git از ۴ نوع پروتکل پشتیبانی می‌کند:
۱) http(s): پروتکل http با پورت ۸۰ و https با پورت ۴۴۳ کار می‌کند و معمولا فایروال‌ها مشکلی با این پروتکل‌ها ندارند. از هر دوی آن‌ها می‌توان برای عملیات نوشتن و یا خواندن استفاده نمود و می‌توان آن‌ها را به گونه‌ای تنظیم کرد که برای برقراری ارتباط نیاز به تائید هویت داشته باشند.
۲) git: پروتکلی فقط خواندنی است که به صورت anonymous و بر روی پورت ۹۴۱۸ کار می‌کند. شکل استفاده از آن به صورت زیر است و معمولا در github کاربرد فراوانی دارد:

```
git://github.com/[username]/[repositoryname].git
```

۳) ssh: همان پروتکل استفاده شده در یونیکس است که بر اساس مقادیر کلیدهای عمومی و خصوصی تعیین هویت را انجام می‌دهد. شکل استفاده از آن به صورت زیر است و بر روی پورت ۲۲ کار می‌کند و امکان نوشتن و خواندن را می‌دهد:

```
git@github.com:[username]/[repositoryname].git
```

۴) file: تنها استفاده محلی دارد و امکان نوشتن و خواندن را می‌دهد.

نحوه‌ی عملکرد git به صورت remote:
به طور کلی هر برنامه‌نویس نیاز به دو نوع از دستورات دارد تا همواره repository محلی با remote هماهنگ باشد:
۱) بتواند به طریقی داده‌های موجود در repository محلی خود را به سمت سرور بفرستد.
۲) این امکان را داشته باشد تا repository محلی خود را با استفاده از repository در سمت سرور آپدیت نماید تا از آخرین تغییراتی که توسط بقیه اعضای گروه صورت گرفته است آگاهی یابد.

طریقه رفتار git برای کار با repositoryهای remote به صورت زیر است:
هنگامی‌که کاربر قصد دارد تا repository یا شاخه‌ای از آن را به سمت سرور بفرستد، git ابتدا یک شاخه با نام همان شاخه به اضافه origin/ ایجاد می‌کند. مثلا برای شاخه master، آن نام به صورت زیر می‌شود:

```
origin/master
```

عملکرد این شاخه دقیقا مانند دیگر شاخه‌های git است؛ با این تفاوت که امکان check-in یا out برای این نوع شاخه‌ها وجود

ندارد. زیرا git باید این شاخه‌ها را با شاخه‌ها متناظرشان در remote هماهنگ نگه دارد.
از این پس این شاخه‌ی ایجاد شده، به عنوان واسطی بین شاخه محلی و شاخه راه دور عمل می‌کند.

cloning:

با استفاده از دستور clone می‌توان یک repository در سمت سرور را به طور کامل در سمت کلاینت کپی کرد. به عنوان مثال repository مربوط به کتابخانه jquery از وب سایت github به صورت زیر است:

```
git clone https://github.com/jquery/jquery.git
```

همچنین می‌توان با استفاده از دستور زیر پوشه‌ای با نامی متفاوت را برای repository محلی انتخاب نمود:

```
git clone [URL][directory name]
```

اضافه کردن یک remote repository:

برای آن‌که بتوان تغییرات یک remote repository را به repository محلی منتقل نمود، ابتدا باید آن را به لیست repositoryهای ریموت که در فایل config. ذخیره می‌شود به شکل زیر اضافه نمود:

```
git remote add [alias][URL]
```

در دستور فوق، برای repository باید یک نام مستعار تعریف کرد و در بخش URL باید آدرسی که سرور به وسیله آن امکان دریافت اطلاعات را به ما می‌دهد، نوشت. البته این بستگی به نوع پروتکل انتخابی دارد. به عنوان مثال:

```
git remote add origin https://github.com/jquery/jquery.git
```

اگر بخواهیم لیست repositoryهایی که به صورت remote اضافه شده‌اند را مشاهده کنیم، از دستور زیر استفاده می‌کنیم:

```
git remote
```

در صورتی‌که دستور فوق را با v- تایپ کنید اطلاعات کامل‌تری در رابطه با repositoryها مشاهده خواهید کرد.
همچنین برای حذف یک remote repository از دستور زیر استفاده می‌کنیم:

```
git remote [alias] -rm
```

در صورتی‌که بخواهید لیستی از شاخه‌های remote را مشاهده کنید کافیست از دستور زیر استفاده کنید:

```
git branch -r
```

همچنین می‌توان از دستور زیر برای نمایش تمامی شاخه‌ها استفاده کرد:

```
git branch -a
```

fetch:

برای دریافت اطلاعات از دستور زیر استفاده می‌کنیم:

```
git fetch [alias][alias/branch name]
```

در صورتی‌که تنها یک repository باشد می‌توان از نوشتن نام مستعار صرفنظر نمود. همچنین اگر شاخه یا شاخه‌های مورد نظر به صورت track شده باشند، می‌توان قسمت دوم دستور فوق را نیز ننوشت.
اگر بعد از اجرای دستور فوق، بر روی یک شاخه log بگیرید، خواهید دید که تغییرات در شاخه محلی اعمال نشده است زیرا دستور فوق تنها داده‌ها را بر روی شاخه origin/[branchname] ذخیره کرده است. برای آپدیت شدن شاخه اصلی باید با استفاده از دستور merge آن را در شاخه مورد نظر ادغام کرد.

pulling:

چون کاربرد دو دستور fetch و merge به صورت پشت سر هم زیاد است git دو دستور فوق را با استفاده از pull انجام می‌دهد:

```
pull [alias][remote branch name ]
```

اگر دو مقدار فوق را برای دستور pull تعیین نکنید، ممکن است در هنگام اجرای دستور فوق با خطایی مواجه شوید مبنی بر اینکه git نمی‌داند دقیقا شاخه ریموت را با کدام شاخه محلی باید ادغام کند. این مشکل زمانی پیش می‌آید که برای شاخه ریموت یک شاخه محلی متناظر وجود نداشته باشد. برای ایجاد تناظر بین دو شاخه ریموت و لوکال درگذشته باید فایل config. را تغییر می‌دادیم، اما نسخه جدید git دستوری را برای آن دارد:

```
git branch --set-upstream [local brnach][alias/branch name]
```

با اجرای این دستور از این پس شاخه محلی تغییرات شاخه remote را دنبال می‌کند.

pushing:

با استفاده از push می‌توان تغییرات ایجاد شده را به remote repository انتقال داد:

```
git push -u [alias][branch name ]
```

وجود u- در اینجا بدین معنا است که ما میخواهیم تغییرات repository در سمت سرور دنبال شود. در صورت استفاده نکردن از u- بایستی برای push هر بار مقادیر داخل کروشه‌ها را بنویسیم. در صورتی‌که بعدا بخواهیم، می‌توان توسط همان دستوری که در قسمت pull گفته شد دو شاخه را به هم وابسته کنیم.

tag:

همانطور که قبلا گفته شد تگ‌ها برای نشانه‌گذاری و دسترسی راحت‌تر به به commitها هستند. برای ایجاد یک تگ از دستور زیر استفاده می‌شود:

```
git tag [tag name]
```

همچنین می‌توان با a- برای تگ پیامی نوشت و یا با s- آن را امضا کرد. برای مشاهده تگ‌ها از دستور زیر استفاده می‌شود:

```
git tag
```

در حالت پیشفرض git تگ‌ها را push نمی‌کند. برای push کردن تگ‌ها باید دستور push را با اصلاح‌کننده tages-- به کاربرید.

نظرات خوانندگان

نویسنده: reza
تاریخ: ۱۳۹۱/۰۷/۲۰ ۱۵:۵۵

اگر بخواهیم بر روی سرور خودمان راه اندازی کنیم چه راه حلی وجود دارد ؟

نویسنده: حسام امامی
تاریخ: ۱۳۹۱/۰۷/۲۱ ۰:۲۲

میشه از git daemon استفاده کرد که در واقع یک سرور برای repositoryهای git است
توی این مقاله از CopSSH استفاده کرده codeproject
از scm-manager هم میتونید استفاده کنید
در ضمن همانطور که گفتم از پروتکل فایل هم پشتیبانی میشه یعنی میتویند در شبکه شیر کنید

نویسنده: neo
تاریخ: ۱۳۹۱/۰۸/۱۸ ۰:۰

با سلام خدمت شما آقای امامی.
من در قسمت pushing وقتی میخواهم این دستور را git push origin master اجرا کنم یه یوزر و پسورد میخواهد؟ این قسمت را میشه بگید چطوری است؟
دوستدار شما علیرضا از شهر ممقان.

نویسنده: حسام امامی
تاریخ: ۱۳۹۱/۰۸/۲۰ ۲۱:۵۴

با سلام
دقیقا متوجه منظور شما نشدم
اگر شما قصد داشته باشید که به یک سرور راه دور repository را push کنید بسته به نوع پروتکل از شما شناسه کاربری و رمز عبوری می‌خواهد که همان سایت به شما داده است
موفق باشید

نویسنده: سام ناصری
تاریخ: ۱۳۹۱/۱۲/۲۹ ۸:۲۱

فکر کنم بشود در محیط LAN با استفاده از آدرسهای شبکه محلی repositoryها را بین کاربران به اشتراک گذاشت. مثلاً من repository خودم را share کنم و همکارم هم repository خودش را. بعد هر کدام repository اون یکی را به remoteها اضافه کند.
همچنین میتوان یک repository از نوع bare در سرور شبکه LAN تعریف کرد و همه به آن به عنوان repository مادر متصل شوند.
البته من خیلی مطمئن نیستم. منتظرم تا ببینم نظر حسام در اینباره چیه؟

نویسنده: حسن
تاریخ: ۱۳۹۱/۱۲/۲۹ ۲۳:۵۰

بله. امکان تهیه Shared Repository هست. این مطلب هم مفید است.

نویسنده: kish
تاریخ: ۱۳۹۳/۰۳/۲۰ ۱۵:۳۱

سلام
در webstorm تحت LAN چه جوری می‌تونم از git استفاده کنم؟
با تشکر