Vad är Repository, Branch och Commit?

Repository

Ett repository är den huvudsakliga komponenten som spårar förändringar i ditt projekt. Den komponent hittar man under `./git` i roten av ditt projekt. Genom att titta på den här katalogen kan Git manipulera koden som du har valt att spåra. Tar du bort `./git` från ditt projekt förlorar du all spårning av ditt projekt i den instansen (om du har skickat förändringar till en server, finns de förändringarna som du skickat sparade där).

> Ett repo kan vara både lokalt på din maskin eller online på en server (remote) som https://git.ssis.nu/ och https://github.com/ du synkar dem genom push- och pull-kommandon

Att spåra förändringar

När man skapar ett nytt Git repository måste du manuellt lägga till de filer som du vill spåra. Det kan vara rimligt att inte spåra alla filer i ditt projekt. Om du använder en IDE till exempel finns det vissa filer som automatiskt sparas, som inte är relevanta till din kod. Dessa filer kanske inte är nödvändiga i ditt repository.

> Det vanligaste felet (som även professionella programmerare gör ibland) är att spåra förändringar av API-nycklar eller lösenord till databaser. Dessa får aldrig spåras, punkt.

För att förhindra att av misstag lägga upp filer som innehåller känsliga uppgifter kan du använda något som kallas `.gitignore`. Det är en fil som berättar för Git att du inte vill spåra vissa filer eller kataloger. Du skriver helt enkelt namnet (och sökväg om det behövs) på filen på var sin rad.

Exempelvis:

`.gitignore`

```
/node_modules
/public/hot
/public/storage
/public/js
/public/css
/public/mix-manifest.json
/storage/*.key
/vendor
.env
.env.backup
.phpunit.result.cache
docker-compose.override.yml
Homestead.json
Homestead.yaml
npm-debug.log
yarn-error.log
```

För att faktiskt spåra en förändring behöver du använda ett kommando (om du använder ett grafiskt gränssnitt är det här inte lika relevant) som kallas `git **add**` (alla kommandon i git måste prefixas såhär: `git **kommando**`).

```
du@dator:~/myFirstRepo$ git add minfil
```

Det innebär att git kommer börja hålla koll på den filen (om det är första gången). `add` lägger även till en fil till en s.k. commit och då är den *staged for commit.*

Commit

En commit är det huvudsakliga verktyget som du som programmerare segmenterar dina förändringar i mindre delar med. Det kan verka tradigt och lite poänglöst men det kommer ge utdelning i längden för dig och ditt framtida team (som ibland, är dig själv fyra månader i framtiden).

En commit ska göras varje gång du gör en förändring. Inte varje sekund, eller varje minut (det är oftast inte rimligt). Men när du som programmerare väsentligt förändrar hur din kod fungerar ska du commita det. Det innebär att när du är klar med något som fungerar, bör du commita det. Den förändringen vill du spara, för då kan du återgå och titta på förändringarna över tid om något går fel eller om det inte fungerar lika bra som förut.

Om du förändrat många olika saker bör du separera förändringarna till sina domäner. Gör du en förändring i hur din användare fungerar och sen i din databas, bör dessa två förändringar vara i två olika commits. Det är inte rekommenderat att förändra många olika saker (som inte påverkar

varandra) och lägga dem i samma commit.

Exempel:

Branch

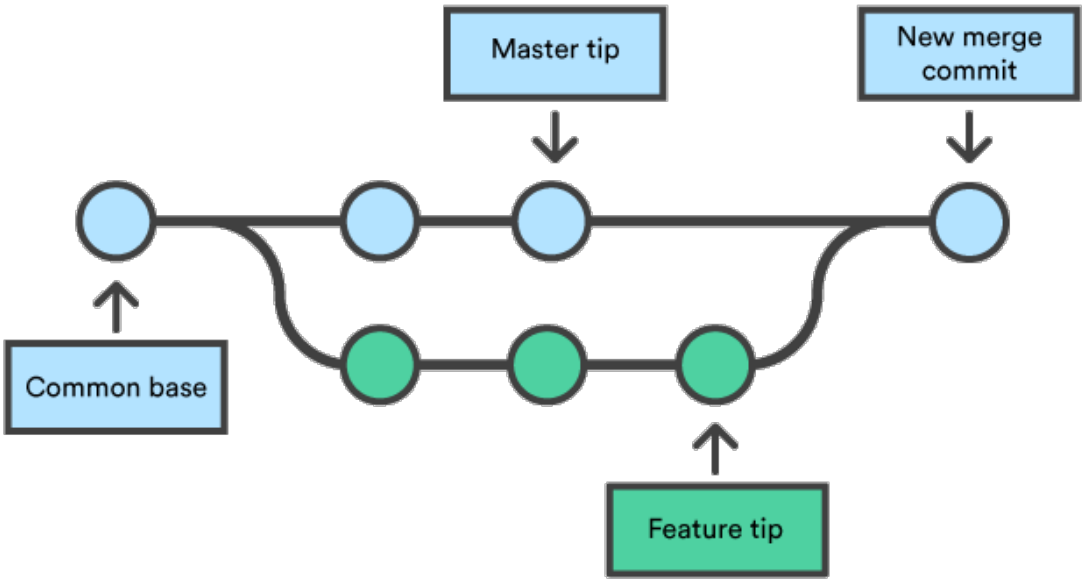
Git använder sig av något som kallas branches (eller grenar) för att spåra förändringar.

Huvudbranchen brukar kallas för `main` (eller `master` på aningen äldre system, läs mer här om varför). Det är det huvudsakliga projektet. Alla andra branches deriverar sin sanning från `main` vilket innebär att när du skapar en ny branch gör du det från main oftast.

Branching är främst till för att hålla main-branchen i ett fungerande stadie. Målet är att kunna klona ditt repo och veta att koden som laddas ner faktiskt fungerar.

Därför skapar du branches för din egen utveckling. Säg att du behöver lägga till en funktion (eller *feature*) till ditt program eller tjänst. Istället för att börja ändra i main-branchen skapar du en ny branch från main för att bevara mains integritet. På den här nya branchen kan du commita dina förändringar, utan att påverka main direkt. Därefter förenar du dem (*merge*) och låter main absorbera den nya branchen.

Det är lite som en tågräls där ett tåg fortsätter rakt fram och ett annat tåg tar av till ett annat spår för att få ny last och sen förenar sig med det tidigare tåget lite längre fram:



I mindre skolprojekt kan det vara så att branching inte är särskilt nödvändigt. Men i framtiden om du och dina skolkamrater ska arbeta tillsammans på samma projekt, är git ett otroligt bra sätt att kommunicera och åtgärda fel i kod som skrivs av många personer samtidigt.

Revisions #7
Skapad 3 maj 2021 15:28:41
Uppdaterad 18 augusti 2021 11:41:14