Öva på samarbete med GIT

Förberedelser

Vi ska nu jobba med git & GitHub enligt **Shared Repository Model**

Ni ska nu dela upp er i grupper. Grupper ska bestå av **3 personer**. En av personerna i gruppen ska var **ägaren** till repositoryt vi kommer att arbeta med. De andra två personerna kommer vara *Collaborators* som ska skicka upp ändringar till originalet. Ni får själva utse rollerna.

Även fast det står att ni ska ha olika roller får ni hjälpa varandra om ni fastnar på något, det är en gruppuppgift.

- GitHub Help
- git - the simple guide
- Understanding the GitHub Flow

Övning - Storyteller

Repot ni skapar ska innehålla textfiler som tillsammans bildar en saga

Alla som bidrar till repot ska därmed bidra till sagan

Sagan behöver inte vara speciellt bra. Det är `git`-delen vi ska fokusera på.

Sagan ska skrivas på engelska och ska vara barnvänlig.

1. Personen som är ägare i gruppen ska skapa ett nytt repo på **GitHub** och kalla det: `storyteller-by-fornamn-efternamn-klass`
2. Ägaren klonar ner repot till datorn: `git clone`
3. Ägaren skapar sedan följande tomma filer lokalt i den klonade mappen:
    - 1_tale_begins.txt
    - 2_villain.txt
    - 3_hero_enters_tale.txt
    - 4_epic_quest.txt
    - 5_journey_through_land.txt
    - 6_great_obstacle.txt
    - 7_hero_wins.txt
    - 8_moral_of_the_tale.txt
    - 9_end.txt

Fyll på filerna

1. Ni som grupp ska nu fylla på filerna **1, 3, 5-9** (**Inte 2 och 4**). Varje fil ska ha minst en mening. Här får ni hjälpas åt att skriva meningarna.
2. Ägaren lägger sedan till filerna till *Staging Area* via `git add`, sedan gör en ny commit med `git commit` och skriver ett lämpligt commitmeddelande.

Dela ut uppdrag

1. Se till så att samtliga deltagare är *Collaborators* genom att gå till ditt repo sen trycka på **Settings > Collaborators** och sedan bjuda in användarna till repositoryt. Se till så att alla användare har accepterat inbjudan, det borde komma ett mail.

2. Person **A** och **B** i din grupp ska klona ner repot lokalt till sin egen dator (`git clone`), sedan ska de skapa var sin ny branch lokalt `git checkout -b name-of-branch`

3. **A** och **B** får var sin uppgift:

    i. person **A** i din grupp ska på sin branch editera fil **2** och ändra meningarna i den filen
    ii. person **B** i din grupp ska på sin branch editera fil **4** och ändra meningarna i den filen.
    iii. Person **A** och **B** ska sedan pusha upp sin branch till GitHub med `git push -u origin name-of-branch`. De som är *collaborators* ska alltså skicka upp sina egna branches, inte `master`.

4. Gå in på repot på **GitHub** och se till så att branches har pushats.

Slå ihop alla samarbeten

Creating a Pull Request

Using Pull Requests

Merging a Pull Request

Closing a Pull Request

Nu har du som ägare fått en del input ifrån din grupp och det är dags att slå ihop alla branches in i din `origin/master` så du som ägare har en sammanslagen uppdaterad version av sagan.

1. Person **A** och **B** öppnar varsin **Pull Request** och begär att ägaren mergar deras redigerade branches med originalet (`master`). Detta kan inte allt göras samtidigt utan man måste göra **en merge i taget**

2. Ägaren mergar alla **Pull Requests**.

När allt ovan är klart och ni har den uppdaterade sagan i `master` går ni vidare till nästa steg.

Dags för lite konflikter

Du ska nu ge ut nya uppdrag till din grupp:

1. **A** får i uppgift att:

- i **fil 2** lägga till några adjektiv i beskrivningen av skurken (beskrivande ord som ugly, evil osv)
- i **fil 3** lägga till en kort beskrivning av hjältens side-kick .
- i någon av filerna mot slutet av sagan lägga till något som denna side-kick gör för att bidra till sagan.

2. **B** får i uppgift att

- i **fil 3** lägga till några adjektiv i beskrivningen av hjälten
- i **fil 2** lägga till en kort beskrivning av skurkens husdjur
- i någon av filerna mot slutet av sagan lägga till något som detta husdjur gör för att bidra till sagan.

Samtidigt som **A** och **B** arbetar på filerna **2** och **3** ska du som är ägare även gå in och lägga till att både skurken och hjälpten har något sorts vapen eller föremål. D.v.s. ändra samma filer som **A** och **B** ändrar på samtidigt.

När allt ovan är klart går du vidare till nästa steg.

`merge`

1. Nu har du fått en del input ifrån din grupp och det är dags att slå ihop all input till en version. Vi ska nu merga alla våra branches till `origin/master` så du har en sammanslagen uppdaterad version av sagan. (merge pull request)

2. **Whoops!** Eftersom flera användare har ändrat på filer på samma ställe vet inte `git` vilken version som `git` ska använda, all input är likvärdig. Ska vi använda ägarens, person A eller person B version? Detta måste vi ange själva och vi kan inte gå vidare med att *merge pull request* innan vi har löst våra konflikter.

3. Det finns olika sätt att lösa en konflikt. I många fall måste du först dra ner alla ändringar till din egen dator och göra det lokalt för att sedan pusha upp ändringar. **GitHub** har nu dock introducerat en web editor så att du kan lösa konflikter direkt från GitHub. Se dock alltid till så att dessa ändringar blir synkade med dina lokala ändringar. Du kan öppna web editorn genom att trycka på **Resolve Conflicts** när ni mergar era branches

Färdig?

Nu borde ni som grupp ha ett repo med en fin liten saga samtidigt som ni har lärt er att dela kod med git via GitHub. Här är några saker att tänka på för att underlätta för er själva:

- **Dra alltid ner ändringar innan du börjar arbeta med koden:** `git pull`
- **Försök undvika att jobba på samma kodrader, dela upp arbetet så att ni inte modiferar så mycket på samma ställe**
- **Kommunicera tydligt vilka delar ni jobbar på**
- **Använd en `.gitignore` -fil på filer som oftast skapar problem (konflikter) eller som innehåller känslig data. Detta kan t.ex. var `.css` -filer om man använder `.sass`**

Läs vidare: Markdown

Det kan vara en bra idé att läsa på om Markdown också. Alla projekt på *GitHub* har en välformatterad **README** där man kan läsa om vad projektet handlar om. Dessa **README**s är alltid i formatet **Markdown** och har filändelsen `.md` eller `.markdown`. *Markdown* är ett format som enbart fokuserar på strukturen, hur vi formatterar vår text. *GitHub* sköter sedan automatiskt om att konvertera detta till HTML och ge det en snygg design. Jag har skrivit den här övningen i Markdown t.ex.

Markdown används i stor utsträckning och är inte alls svårt att lära sig. Vill man t.ex. ha en rubrik i markdown så skriver man en hashtag framför: `# Rubrik`, vill man ha fetstilat så skriver har man två asterisk runt texten: `**fetstilat**`. En fullständig guide till att man kan göra i Markdown finns nedan:

Markdown Cheat Sheet