Laborationsrapport

Versionshantering & Git
DT173G, Webbutveckling III

Författare: Elias Eriksson, eler2006@student.miun.se
Termin, år: HT, 2021

Mittuniversitetet
MID SWEDEN UNIVERSITY

Campus Härnösand Universitetsbacken 1, SE-871 88. **Campus Sundsvall** Holmgatan 10, SE-851 70 Sundsvall.
Campus Östersund Kunskapens väg 8, SE-831 25 Östersund.
Phone: +46 (0)771 97 50 00, Fax: +46 (0)771 97 50 01.

Sammanfattning

Uppgiften går ut på att lära sig att använda versions hanterings systemet git och kunna använda git med en remote host som github, samt att lära sig att skriva .md filer.

Innehållsförteckning

1 Frågor

1.1 Fråga 1

1. Förklara kortfattat syftet med versionshantering.

Versionshanteringssystem låter utvecklare skapa snapshots utav sina projekt. Alla snapshots sparas och det är möjligt att hoppa emellan dessa snapshots. Det går att bara kolla hur en tidigare snapshot såg ut (git checkout) men även radera alla snapshots framför en viss snapchot (git reset) eller återställa till hur en tidigare snapshot såg ut (git revert). Det går även att skapa grenar (branches) som kan utvecklas parallellt. Detta är användbart för att bland annat ha en utveckling gren och en live gren eller om det är flera personer som utvecklar på samma projekt kan de ha olika grenar. Dessa grenar kan sedan bli sammanslagna (git merge eller git rebase) och på så sätt få in ändringarna till en och samma gren. [6]

Alla dessa snapshots och grenar sparas i ett repository och finns lokalt på en dator. Dessa repositorys kan också laddas upp på en server (remote host). Antingen privat hostad eller en mer public host t.ex. github. Ändringar som görs lokalt kan då laddas upp på servern (git push). Andra användare kan sedan ladda ner repositoryt från t.ex github (git clone) eller ladda ner de senaste ändringarna (git pull).[6]

Versionshanteringssystem ger os därmed en bra möjlighet att jobba med andra utvecklare samt att det även blir en bra backup ifall man vill gå tillbaka till tidigare versioner ifall allt till exempel slutar fungera som det ska efter någon ändring. [6]

2. Beskriv kortfattat tre stycken olika system för versionshantering ("version control systems") som är användbara för oss som webbutvecklare. En av dessa ska vara Git, de övriga två är valfria.

Beskriv syftet, utvecklare, utvecklingsår, licens (om det är baserat på öppen källkod eller ej) och annat du finner intressant.

Vid jämförelse av mercurial git och SVN (subversion), är det tydligt att de tre olika versions kontroll systemen har olika idéer när de skapades. Mercurial och git är båda två så kallade distribuerade versions kontroll system så mercurial och git ska vara väldigt lika. De skiljer sig dock lite granna. På grund utav att git har tillgång till fler olika kommandon och alternativ till dessa kommandon så är git mer komplex än vad mercurial är. [1] Mercurial utvecklades också med iden om att all historik är viktig och låter därmed användare endast backa i historiken ett steg i taget. Det går inte heller att radera utvecklingsgrenar. [2] Eftersom att mercurial inte låter en använda kommandon som typ efterliknar `git rebase` så gör det att en oerfaren användare inte kan åstadkomma så stor skada. Git till skillnad från mercurial (och även SVN) har ett så kallat staging area som låter användare lägga till exakt dom filer som ska vara med i en kommit istället för alla ändrade

filer. [1][3] SVN har förutom detta flera skillnader. SVN är ett så kallat centraliserat versions hanterings system. Detta innebär att istället för att hämta hela versions historiken som git och mercurial gör [4] så hämtas endast den senaste versionen som går att utveckla på och sedan pusha till den centraliserade servern. Tanken med detta är att projekt ledningen har en bättre överblick på vad som görs. [3]

1.2 Fråga 2

1. Förklara följande begrepp:

1. Workspace
   Workspace är den mappen som alla projekt filer och .git mappen ligger i. [5]

2. Staging area
   Staging area är det git använder för att veta vilka filer det är som ska vara med vid nästa commit. Om en fil inte är stagade med `git add` så kommer `git commit` inte att göra en snapshot.[6]

3. Local repository
   Ett local repository är ett repository som ligger lokalt på ens egna dator. Repositoryt innehåller alla projekt filer samt alla commits som har skett under repositoryts liv samt staging area. Vi en `git add` eller `git commit` så är det sin lokala repository ändringarna görs i.[6]

4. Remote repository
   Ett remote repository är ett repository som ligger på en server någonstans. Denna server kan antingen hostas privat på egen / företags server eller så används en tjänst som github. Ett lokalt repository kan sedan använda `git push` för att uppdatera serverns repository med ändringar som gjorts på det lokal. `git clone` eller `git pull` kan också användas för att hämta ändringar som gjorts på en annan dator eller annan användare.[6]

5. Branch
   En bransch är en förgrening i projektet förgreningen innebär att det går att göra ändringar på olika ställen i projektet. [6]Dessa grenar kan sedan ”slås ihop” med en `git merge`

6. Merge
   En merge är när en gren slås ihop med en annan gren till sig själv. [6]

2. Förklara följande begrepp:

1. git config
   `git config` gör det möjligt göra vissa konfigurationer till ens lokala repository eller globalt.[7] Dessa konfigurationer är till exempel att spara inloggningsuppgifter till remote repositorys. Vilket kan göras med `git

config credential.helper store` eller `git config --global credential.helper store`. Med --global sparas konfigurationen globalt och kan användas av andra repositorys som också har en global konfiguration.[8]

2. git init
   `git init` initialiserar ett nytt git repository som.[6] Detta repository finns bara lokalt tills det pushas upp till en remote host.

3. git status
   `git status` visar information om den nuvarande grenen är bakom eller före remote grenen samt om vilka filer som har ändrats och blivit tillagda i repositoryt. [6]

4. git add
   `git add` lägger till filer till ett ”staging area” som. Filer som blivit tillagda till ett staging area är de filerna som kommer att få en snapshot vid nästa commit.[6]

5. git commit
   `git commit` skapar en snapshot av de filerna som finns i staging area.[6]

6. git push
   `git push` laddar upp nuvarande grenens commits till remote host.[6]

7. git checkout
   `git checkout` gör det möjligt att kolla på andra branches eller kommits och se hur projektet ser ut vid en annan bransch eller såg ut vid en tidigare commit. Detta görs genom att flytta HEAD pekaren till en annan commit. [6]

8. git pull
   `git pull` hämtar ändringar från en remote bransch till det lokala repositoryt och mergar ihop dom.[6]

9. git merge
   `git merge` slår ihop den nuvarande grenen med en annan specificerad gren på ett sådant sätt att alla commits kommer i tidsordning.[6]

10. git fetch
    `git fetch` hämtar alla commits om den nuvarande branchens remote host men mergar inte ihop ändringarna. Ändringarna kan sedan inspekteras med `git checkout` innan en fullständig `git pull` körs för att merga ihop. [9]

11. git log
    `git log` visar en historik om vem det är som har gjort en viss commit. Committens meddelande och de branches som har committen som sin senaste commit.[6]

2 Slutsatser

Vi som utvecklare har stora fördelar om vi använder ett VSC då det bland annat låter oss lätt arbeta med en eller flera andra personer på samma projekt och det låter oss ha en historik på hur hela projektet har ändrats över tid och låter oss därmed gå tillbaka till tidigare snapshots av våran kod.

3 Källförteckning

Här följer exempel på hur en källförteckning kan utformas enligt Vancouver-systemet. Den är automatiserad enligt metoden numrerad lista och korsreferenser. Radera denna text, samt ersätt källorna med dina egna.

[1] Chuck Gehman ”Mercurial vs. Git: How Are They Different?” https://www.perforce.com/blog/vcs/mercurial-vs-git-how-are-they-different Hämtad: 2020-08-31

[2] Steve Losh ”Mercurial vs. Git: why Mercurial?” https://www.atlassian.com/blog/software-teams/mercurial-vs-git-why-mercurial Hämtad: 2020-08-31.

[3] Erica Chang ”SVN vs Git: Which One Is Best for Your Needs?” https://blog.hackbrightacademy.com/blog/svn-vs-git/ Hämtad: 2020-08-31.

[4] Diego Vergara ”VERSION CONTROL SYSTEMS: DISTRIBUTED VS. CENTRALIZED” https://www.oshyn.com/blog/version-control-systems-distributed-vs-centralized Hämtad: 2020-08-31.

[5] Mattias Dahlgren & Malin Larsson ”dt173g_introduktion_ht21” https://play.miun.se/media/dt173g_introduktion_ht21/0_h1v0ijk6 Hämtad: 2020-08-31.

[6] Mattias Dahlgren ”Versionshantering, GIT, Github och Github Pages” https://play.miun.se/media/Versionshantering%2C+GIT%2C+Github+och+Github+Pages/0_yfvidx92 Hämtad: 2020-08-31.

[7] Git-scm ”git-config - Get and set repository or global options” https://git-scm.com/docs/git-config Hämtad: 2020-08-31.

[8] Git-scm ”git-credential-store - Helper to store credentials on disk” https://git-scm.com/docs/git-credential-store Hämtad: 2020-08-31.

[9] Atlassian ”git fetch” https://www.atlassian.com/git/tutorials/syncing/git-fetch Hämtad: 2020-08-31.