Workshop 5 – GIT II

Innehållsförteckning

[Kollaboration med Git 2](#_Toc398901539)

[Introduktion till GitHub 2](#_Toc398901540)

[Uppgift 1 – Konfigurera Git och skapa ett repository 2](#_Toc398901541)

# Kollaboration med Git http://git-scm.com/images/logos/2color-lightbg@2x.png

I första delen av workshopen har vi bekantat oss med versionshanteringssystemet Git. Vi lärde oss exempelvis att installera GitBash för att kunna starta ett kommandofönster via filhanterarens högerklicksmeny, t.ex inne i xampp/htdocs/[projektnamn]. Fördelen med det är att det är enkelt att arbeta med serverscriptspråk som t.ex. php eftersom webbservern stödjer exekvering av dem i denna area. Behöver ni repetera lite eller vill lära mer hittar ni mycket nyttig information på Gits officiella webbplats: <http://git-scm.com/>.

Eftersom vi har lärt oss grunderna genom att öva mot ett lokal repository är nästa steg att arbeta med ett repository som ligger på webben. Fördelen med detta är att alla projektdeltagare kan arbeta mot samma repository och hela tiden följa varandras arbete. En annan fördel är att det blir en extra säkerhet mot förlust av data eftersom det hela tiden finns en giltig kopia på internet, t.ex. på GitHub. GitHub är gratis för publika projekt och passar utmärkt för att prova på i samband med att vi lär oss systemet. Det finns många andra hostar som också kan användas, men GitHub fungerar bra och är dessutom gratis.

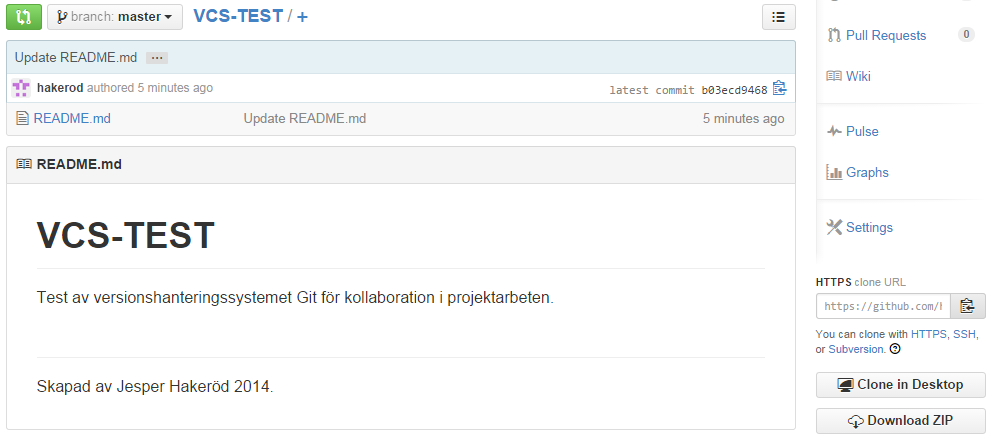
## Introduktion till GitHub

GitHub används tillsammans med lokala versioner av versionshanteringssystemet Git för att vi ska kunna placera vårt repository för projektet på webben så att alla kommer åt dem. Så fort ni bekantat er med att använda Git kan ni börja använda det i era projekt, men vi måste först av allt skapa ett konto och en area på Github, där vi därefter ska placera vår repository för projektet.

Figur 1: GitHub Bootcamp

### Uppgift 1 – Konfigurera Git och skapa ett repository på GitHub

1. Öppna webbläsaren och gå till urlen: <https://github.com/>.
2. Skapa ett konto på GitHub och logga in på GitHub.
3. Inne på GitHub kan du nu se att det finns en anvisning i 4 steg som kallas för GitHub BootCamp.
4. Börja med att starta steg 1 - *Set up a Git*, se figur 1. Följ guiden efter att ha klickat på steg 1 och dubbelkolla så att ni hamnar rätt i era val (se nedan):
   1. I *Set up a Git* har du redan installerat den lokala Git-miljön GitBash och kan därför hoppa över det steget.
   2. Det är viktigt att ni konfigurerar Git med dels ett namn, öknamn går också bra, och en korrekt mailadress, det ska vara samma epostadress som den i ditt GitHub-konto. Du använder följande kommandon i GitBash:   
      *git config --global user.name”ditt namn”* och *git config –global user.email ”din epostadress”*
   3. Du behöver även kunna koppla upp dig via HTTPS (rekommenderat) eller SSH. Vi väljer HTTPS och använder *credential helper* för att slippa skriva in lösenord varje gång vi kör *push* och *pull* mot GitHub. För att det ska fungera måste du göra ytterligare en konfigureringsinställning i GitBash:  
      *git config—global credential.helper wincred*
5. Efter konfigureringen av den lokala Git-miljön kan vi nu skapa en repository på GitHub. Starta steg 2 – *Create a Repository*. Följ guiden som dyker upp då ni klickade på steg 2 och dubbelkolla så att ni hamnar rätt i era val (se nedan).
   1. Tänk på att välja korta och träffsäkra namn på ert repository som är lätta att hantera och komma ihåg, i vårt fall kan vi namnge det som *VCS-TEST*. Beskrivningen skulle kunna vara *Test av versionshanteringssystemet Git för kollaboration i projektarbeten*.
   2. Eftersom vi ej vill betala så får vi välja att lägga det ***public***. Det går förståss att uppgradera sitt GitHub-konto, men det kostar 7$ i månaden.
   3. glöm ej att kryssa för *Initialize this repository with README.Ej sant*
6. Bra jobbat! Nu har du skapat ditt första repository på GitHub. :D
7. Testa att nu att ändra i filen readme.md genom klicka på den och sen på pennan.
   1. Skriv in tex ”Skapad av ditt namn 2014”. Klicka på preview och se hur det ser ut. Om du är nöjd så fyller du i en kort meningsfull beskrivning av de ändringar du gjort och därefter *commit changes*.
   2. Du känner säkert igen ändringsförfarandet då du committar i GitBash med:  
      *git commit –m ”Lade till namn och datum.”*

Figur : Resultatet efter att ha skapat ett repository på GitHub.

# Kollaborera

Initialt ska alla projektdeltagare skapa ett konto på GitHub. Användarnamnet behöver Person 1 som ska skapa repository på GitHub och därefter lägga till alla projektdeltagare.

Person 1 som skapar projektet på github gör följande:

1. Skapa ett konto på GitHub.
2. Skapa ett repository på GitHub. Du kan skippa init då du skapar repository på GitHub.
3. Ladda ner **GITII.zip** från workshop 5 på blackboard.
4. Packa upp filerna i en katalog, tex i *C:/xampp/htdocs/*. Fördelen med det är att ni kan köra php om ni vill i projektet eftersom det ligger i arean som webbservern hanterar.
5. Starta GitBash i samma katalog och kör en *git init* så att vi skapar ett .git-repository lokalt.
6. Konfigurera user.name och user.email till det som finns på github, se git cheat sheet.
7. Kör *git remote add origin [url-adressen till remote github-repository kopieras in].*Observera att du kan klistra in url-adressen i GitBash genom att högerklicka på ramen och välja *edit>paste*.
8. Vi lägger till alla filer med *git add .* och därefter kör vi *en git status* för att kontrollera att allt är med.
9. Vi kör nu en *git commit –m ”initial change”* så alla filer hamnar i ditt lokala repository.
10. För att föra över filerna till github skriver vi *git push origin master*.
11. Nu är vi klara och kan lägga till projektdeltagare i vårt projekt så de kan skicka upp filer till github i vårt projekt. klicka fram aktuellt repository inne på github. Nere till höger klickar vi på settings och sen collaborators. Lägg till de deltagarna som ska vara med i projektet. Tänk på att de måste ha konto på github de med innan du kan lägga till dem här. Lämpligen skickas länken till github ut till deltagarna via mail och i det mailet kan du även be om deras användarnamn på github. Vi skulle kunna fejka ett sådant mail och skicka iväg det till den som ska vara med och kollaborera (mailadressen behövs såklart). Skicka och vänta på svar.
12. När svar kommer lägger du till de deltagarnas användarnamn.
13. Då var det klart att lämna över stafettpinnen…

Person 2 som kollaborerar gör nu följande:

1. Öppna GitBash i rotbiblioteket på webserverbiblioteket, tex C:/*xampp/htdocs*.
2. Ladda ner en kopia av projektet med *git clone [url-adress till remote GitHub].*
3. När du laddade ner kopian skapas en mapp med namnet på aktuellt repository. Lista filer med *ls*. Hoppa in i mappen med *cd [namnet på mappen].* Kör *ls* igen för att lista filerna och se så allt verkar ha kommit med inklusive .git-mappen (repository).
4. Öppna index.html i webbläsaren för att kika hur det ser ut.
5. Vi skapar en ny branch genom kommandot: *git checkout –b [branchnamn, tex co-founderBio].*
6. Nu kan vi ändra i index.html. Vi öppnar den i sublime text och petar in information om co-founder (du). Spara ändringarna.
7. Nu kan vi kolla statusen med kommandot: *git status*.
8. Vi behöver lägga till de ändrade filerna med kommandot: *git add .*
9. Vi lägger till filerna i repository med kommandot: *git commit -m ”Change co-founder bio”*
10. Vi hoppar över till masterbranchen igen: *git checkout master*
11. Då var det dags att slå ihop brancherna: *git merge [namnet på branchen vi vill slå ihop] [ska det ej vara master här med?]*Vi för över filerna: *git push origin master* (du ombeds att knappa in användarnamn och lösenord)
12. Bra jobbat! Vänta nu en liten stund så nästa person hinner få ner sitt arbete genom pull-kommandot. Börja efter du ser att de laddat ner en aktuell kopia. Därefter ska du genomföra ändringar i samma paragraf så att vi åstadkommer en konflikt i *index.html* (två användare ändrar i samma fil på två olika håll). Kör punkterna 1-3, 5-10, 12 nedan.

Ni har nu sett att det går relativt smidigt att använda Git och ändra i filerna. Vi fortsätter med lite mer att göra för person 1:

1. Vi kör kommandot: *git pull origin master* för att ladda ner den senaste versionen av projektet.
2. Skapa en branch för att ändra i co-founder och i paragrafen : *git branch cofounderbio*
3. Hoppa in i branchen med kommandot: *git checkout cofounderbio*
4. Öppna *index.html* och lägg till någon rad om dig som co-funder. Ovanför paragrafen lägger du till en h3-rubrik med ditt namn, se hur föregående person gjort så det ser likadant ut.
5. Vi vill ändra i rubriken *Welcome To Your Website* på *index.html.* Skriv in en bra rubrik och ändra därefter även texten i paragrafen nedanför. Där behöver stå någon om de tjänster som vi erbjuder som säljer in förtroende till våra potentiella kunder.
6. Vi lägger till ändrade filer med: *git add .*
7. Kika med kommandot: *git status* så det ser riktigt ut.
8. Vi kör därefter: *git commit –m ”Added cofounder text and some selling text about company”*
9. Vi behöver därefter hoppa tillbaka till master branchen: *git checkout master*
10. Vi ska nu slå samman branchen *cofounderbio* med *mastern*: *git merge cofounderbio master*
11. **OBS!** **Vänta ikapp person 2 så denne har hunnit skicka iväg ändringarna med push-kommandot före dig**.
12. Då kör vi upp filerna till vårt repository på GitHub: *git push*
13. Eftersom en annan projektdeltagare har ändrat samma texter som vi gick det inte att ladda upp. Vi får därför ladda ner ändringarna från GitHub: *git pull origin master*
14. Det står om konflikt i *index.html* så vi öppnar den för att lösa konflikten inne i vår editor, sublime text.
15. I filen har Git märkt innehållet med *<<<<<<< HEAD* och skilljelinje *=======* mellan de olika ändringarna som är i konflikt. Avslutningsvis har git lagt in nåt i stil med  *>>>>>>> 5f207d8ef665e7572dd294f9aa6dece829ca6bdd.* Lös konflikten och spara *index.html.*
16. Kolla nu med: *git status*
17. Vi lägger till index.html: *git add .*
18. Vi kör nu: *git commit –m ”Resolved conflict”*
19. Vi provar nu på nytt med: *git push origin master*
20. Mycket bra jobbat! Allt fungerade fint efter vi löste konflikten.