**GitHUB / Git**

* Vad är Git?
* Vad är ­­­GitHub?
* Varför använda git?
* Registrering vid GitHub
* Installera Git for windows
* Att börja med GitHub
* Att skapa ett repositorie
* Att jobba aktivt med git

**Vad är Git?**

Git är ett versionhanteringssytem som är utarbetat främst för mjukvaruutvecklare och hantering av kod, speciellt i större projekt med flera utvecklare som jobbar mot samma kod. Med ett versionshanteringssystem menas att systemet hjälper användarna att hantera filer och spara dem i olika versioner så att man kan spåra förändringar och t.ex. gå tillbaka i historiken. Det ger också möjligheter för flera utvecklare att dela och kunna samarbeta kring gemensamma filer utan att riskera att man skriver över någon annans arbete. Ni kommer att använda er mycket av Git men även stöta på andra versionhanteringssytem t.ex. TFS (Team Foundation Server ), Subversion (svn).

**Vad är GitHUB?**

GitHub är en tjänst som ger oss utvecklare möjlighet att, via webben, skapa, hantera och lagra projekt och dess data via Git. GitHub ger oss också möjlighet att dela kod med andra, studera andras kod och använda andras kod för att bygga vidare på. GitHub är alltså en tjänst för "social coding". Man kan också använda GitHub för att dela utvecklingsfiler i privata grupper eller som ensam utvecklare hantera sina filer på ett smidigt sätt. Vi skulle kunna kalla det ett slags Facebook, fast för kod. På senare tid har dock allt fler upptäckt möjligheterna med Git och GitHub och nu för tiden hittar man inte bara mjukvaruprojekt på GitHub utan används även av författare, lärare och en mängd andra branscher.

**Varför använda git?**

Först och främst kommer Git och GitHub vara något du med säkerhet kommer stöta på i ditt kommande arbetsliv.

Ditt GitHub-konto kommer fungera som ett CV där du samlar dina projekt och där eventuella arbetsgivare kan se vad du kan och har producerat tidigare. Detta är något vi märker fler och fler arbetsgivare tycker är viktigt så att redan från början bygga upp sitt GitHubkonto bör vara en viktig punkt i din utveckling.

Du kommer enklare kunna hantera dina filer. Inga datorkrasher kommer få dina filer att försvinna (om du är noga med att synka dina filer med GitHub). Du kan också komma åt dina filer från flera olika datorer och enheter.

Du kan enkelt dela dina filer med dina lärare. Vid examinering av t.ex. koduppgifter vill läraren titta på din kod och kan då via GitHub enkelt komma åt denna. Vid t.ex. frågor kring eller problem med din kod kan läraren enkelt komma åt den senaste versionen.

GitHub är ett kraftfullt verktyg för att studera och lära sig av andras kod.

**Registrering vid GitHub**

<https://github.com/>

**Installera Git for windows**

<https://windows.github.com/>

Forsätt genom att ange dina uppgifter i konfigurationen för Git. Använd såklart ditt namn och e-postadress (din studentaddress)

$git config --global user.name "John Doe"

$git config --global user.email johndoe@example.com

Kontrollera dina inställningar via:

$git config --list

**Att börja med GitHub**

När man börjar jobba med Git och GitHub kommer man stöta på vissa begrepp som är viktiga att känna till. Vanliga ord som vi kommer använda är:

**Repositorie/Repository** - Kallar ofta förkortat för "repo" och är oftast ett projekt. Vi kan likna ett repositorie med en mapp/katalog innehållandes filer och undermappar som kan versionshanteras. Du skapar oftast ett repositorie per kurs eller ett repositorie per applikation.

**Commit** - Att "commit:a" sina ändringar talar om att man nu har uppdaterat sina filer och sparar dessa i en ny version, en commit.

**Forking** - Detta innebär att man hittar ett intressant projekt som någon annan lagt ut och som man själv vill bygga vidare på. "Forkar" man då projektet får man en kopia av alla filer som man kan fortsätta jobba vidare med på egen hand och kanske i slutändan föreslå dessa förändringar för orginalskaparen.

**Pull request** - Detta innebär att du har forkat ett projekt/repositorie och gjort förändringar i detta som du vill dela med dig av till orginalprojketet. Du kan då föreslå detta för repositorie-ägaren genom att göra en "pull request" som ägaren får godkänna.

Merge - Om ägaren av repositoriet tycker att dina ändringar är bra kan hon/han välja att baka in dessa i orginalkoden d.v.s. göra en "merge", en sammanslagning.

**Branch/Branching** - Du kommer att använda "branching" eller förgreningar i ditt repro för att undvika att ändra i projektets huvudgren. Detta är extra viktigt då man är flera personer som samarbetar i samma repro.

Vi kommer återkomma till vissa av dessa begrepp längre ner i texten.

**Att skapa ett repositorie**

Ett centralt begrepp inom Git är repositorie (repository) eller ofta förkortat som "repo". Detta kan förklaras som ett versionshanterat projekt. Du skapar oftast ett repositorie per kurs eller per applikation. Ett repositorie kan skapa på olika sätt. Antingen via att skapa ett helt nytt repo på din lokala dator eller genom att ladda ner ett redan påbörjat repo från t.ex. GitHub.

**Skapa ett nytt lokalt repositorie**

Börja med att använda kommandot cd i terminalen för att bege dig till någon lämplig katalog där du vill skapa ditt repositorie. Har du ingen erfarenhet av dessa typer av kommandon (UNIX-kommandon) kan vi rekommendera Ubuntu-manualens korta introduktion till de vanligaste. Väl i hemkatalogen skapar du en ny mapp genom att skriva:

**mkdir my-application**

Detta skapar en katalog med namnet my-application. För att skapa ett repositorie i denna mapp navigerar du dig in i mappen och där skriver

**git init**

Du har nu skapat ett helt eget lokalt repositorie och har förmodligen fått en mapp som heter ".git" (kan vara dold i filhanteraren). Den mappen innehåller all information om detta repositorie och de versionsförändringar vi kommer göra. Denna mapp ska du alltså inte bry dig om så mycket utan bara låta Git sköta om. Du kan nu testa statusen på ditt repo genom att skriva:

**git status**

Vi har ännu inte lagt till några filer i vår versionshanterade katalog (vårt repo). Så skapa en fil index.html och spara i katalogen du nyss skapade. Kontrollera nu statusen igen med git status och observera nu skillnaden.

Git har alltså upptäckt att vi lagt till en ny fil i mappen och talar om att denna inte är versionshanterad. Varje fil i en mapp som är Git-hanterad kan antingen vara "tracked" eller "untracked". För att lägga till filen index.html vi nyss skapade skriver du:

**git add index.html**

Ovanstående kommando lägger till just filen index.html. Har man skapat flera filer samtidigt som vill lägga till använder man:

**git add .**

Vilket också är det vanligaste fallet. Kolla nu status på ditt repositorie igen med git status.

Filen är nu tillagd i vad man kallar "staging area" men den är fortfarande inte helt versionshanterad. Vi måste också göra en så kallad "commit" för att få till detta. En "commit" gör man när man suttit och jobbat ett tag med något, kanske skrivit en deluppgift, implementerat en funktion i sin applikation eller kanske bara tar en paus och vill spara undan det man gjort hittils. Man brukar säga att man ska "commit:a" sin kod ofta men inte så att det blir absurt. En "commit" ska också innehålla en bra kommentar som beskriver för andra vad du gjort sedan sist. Även om du kommer jobba ensam i vissa repositorier så var noga med att skriva bra och förklarande kommentarer till dina commit:s. För att göra en commit skriver du i terminalfönstret:

**git commit -m "I have created the file index.html, but it is still empty"**

Du bör såklart byta ut kommentaren innanför citattecknen mot en kommentar som passar i ditt fall. Det kan vara bra att skriva kommentarerna på engelska då man ibland stöter på problem med svenska tecken. Kontrollera nu ditt repositories status

**git status**

Testa nu att göra en ändring i filen index.html samt skapa en ny fil och lägg i mappen. Kolla status, lägg till den nya filen med

**git add .**

och gör en ny commit. Du bör nu ha ett lokalt repositorie med två versionshanterade filer och två stycken "commits".

Skapa ett repositorie på GitHub och klona ner lokalt

Men om man har ett repositorie på GitHub och vill utveckla lokalt på sin dator och sedan skicka upp förändringarna. Hur gör man då?

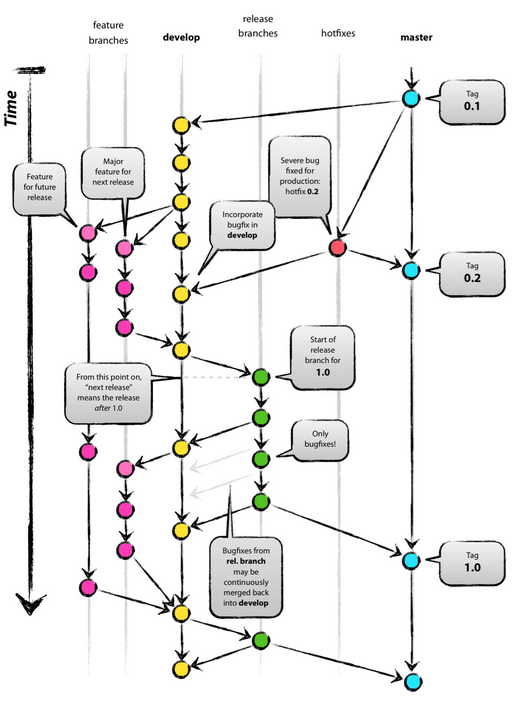
Börja med att skapa ett repositorie på ditt konto på GitHub. På ditt nya repositories första sida, nere till höger, kommer du se något som heter "HTTPS clone URL". Kopiera den sökväg som finns där. Den ska vi nu använda via terminalfönstret klona ner en kopia av repositoriet och få en koppling så vi kan skicka upp våra förändringar.

Gå till terminalfönstret och navigera dig till den mapp där du vill kopiera ner repositoriet och skriv:

**git clone https://github.com/xx222xx/myRepo.git**

där den sista delen är den address du kopierade. Du har nu sparat ner en exakt kopia av det som låg på GitHub. Alla filerna bör ha hamnat i en mapp som om man tittar på addressen ovan borde heta "myRepo". Du har nu ett lokalt skapat repositorie och kan jobba vidare med "add" och "commit" under arbetets gång. Observera dock att dessa förändringar bara sker lokalt än så länge.

**Att jobba Aktivt med Git**



**Hämta ett projekt från GitHub**

$ git clone <https://github.com/hamidsalehian/test.git>

cd test

**Skapa en feature bransch**

$ git checkout -b myfeature

Skapa ny branch "myfeature"

$ git commit -m “Skriv en kommentar”

Eller

$ git commit -am “Skriv en kommentar”

För att lägga till en skapad fil till repo och skriva kommentar

$ git status

För att se senaste ändringar

