1 Vad är Versionshantering?

Versionshantering (eller Version Control) är ett samlingsnamn för program som ger en användare möjlighet att komma åt tidigare versioner av dokument och spåra ändringar som gjorts. Den här funktionaliteten kan vara användbar i flera sammanhang, men den är särskilt värdefull inom mjukvaruutveckling, och versionshantering är idag ett centralt verktyg som du med stor sannolikhet kommer att stöta på i arbetslivet.

2 Git

Ett av de mest använda versionshanteringssystemen heter Git, och det är det ni kommer möta i den här uppgiften. Anledningen att vi valt Git är att det dels är vanligt, det är Free Software (fritt att använda, studera, och även modifiera och distribuera), och det är också förhållandevis enkelt och kraftfullt.

2.1 GitHub

GitHub är en tjänst som sitter ovanpå Git. GitHubs främsta förtjänst är att de ”host:ar” dina repository:s. Det vill säga, de ligger på GitHubs maskiner, så du slipper oroa dig för att förlora allt ditt arbete om du till exempel spiller kaffe på din dator. GitHub har också några andra användbara features som access control, verktyg för ärendehantering, och ett tjusigt webbgränssnitt.

KTH tillhandahåller en egen GitHub server där du kan lagra dina programmeringsprojekt. Den kommer att användas under dina programmeringskurser under det första året. Men det kan vara praktiskt att också skapa ett ”riktigt” GitHub-konto för eventuella sidoprojekt, och eftersom du är student kan du utnyttja GitHub Student Developer Pack.

För att komma åt KTHs GitHub så går du in på `https://gits-14.sys.kth.se/` och loggar in med ditt KTH-id och lösenord.

3 Uppgiften

Börja med att logga in på KTH GitHub om du inte redan gjort det.

3.1 Repositories eller repo:n

En av grundstenarna inom versionshantering är ett repository. Ett repository är den struktur som innehåller alla filer och all metadata (data om data) som används av versionshanteringssystemet. I Git är ett repository en helt vanlig katalog, med en underkatalog som heter ".git" där all metadata lagras. Innehållet i ".git" är helt vanliga textfiler, i den teckenkodning som är default för systemet, och beskriver historiken för filerna i ett repository. För den som är nyfiken på hur det fungerar rent tekniskt finns det en utmärkt förklaring här.

Till att börja med ska du skapa ett nytt repository, det gör du genom att klicka på plus-tecknet uppe i högra hörnet och därefter på "New repository". Välj ett godtyckligt namn och skriv en kort beskrivning. I det här fallet är det inte så noga om du väljer Public eller Private. Men framtida uppgifter vill du se till att ha i ett privat repository. Välj "Initialize this repository with a README". En README är en mer utförlig beskrivning av ditt projekt. Det kommer skapas en fil i ditt repository som heter README.md, där md betyder Markdown, vilket är ett annat språk som används för att formatera text.

Du har också möjlighet att välja en fördefinierad gitignore. Gitignore är också en fil, och som namnet antyder så används den för att säga åt Git vilken typ av filer som inte ska hanteras av systemet. Det kan vara värdefullt att ta en titt på senare, men täcks inte in i den här uppgiften.

3.2 Commits

Git beskriver ditt repository som en serie av "commits". En commit ses som en följd av ändringar till filer (tilläggningar/borttagningar av filer, uppdateringar av innehåll) och ett meddelande. Meddelandet ska vara beskrivande av vilka ändringar som gjorts. Om du tittar på ditt repository i webgränssnittet kan du se att du har en commit, som beskriver ändringarna i din README-fil.

3.3 Kloning

För att kunna arbeta med ditt nya repository på din dator behöver du klona det. Men för att kunna göra det behöver du kunna autentiseria dig, det vill säga, visa för GitHub att du är vem du utger dig för att vara. För KTH GitHub görs detta med något som kallar SSH-nycklar.

Hela förloppet för att skapa nycklarna och koppla det till ditt konto beskrivs här. Under Step 4 ska du dock inte installera xclip, du får istället ersätta:

```
xclip -sel clip < ~/.ssh/id_rsa.pub
```

med

```
cat ~/.ssh/id_rsa.pub
```

och markera manuellt från (inklusive) ssh-rsa . . . fram till och med din e-mail, och kopiera med Ctrl+Shift+C.

Förhoppningsvis har du nu fixat med din nya nyckel och kan ansluta till KTH GitHub. Om du sen vill använda din egen dator behöver du följa samma steg (länken har instruktioner även för Windows och Mac, under huvudrubriken).

Om du nu går tillbaka till ditt repository i webbläsaren, och kollar i den nedra högra delen av din skärm, så ser du en SSH-länk. Kopiera den, och öppna sedan ett terminalfönster och skriv (för att klistra in använder du Ctrl+Shift+V):

```
git clone din-SSH-länk
```

Om du nu kör kommandot ”ls” bör du nu ha en mapp med samma namn som ditt repository. Om du går in i den med ”cd” och kör ”ls -a” bör du se mappen ”.git” och din README-fil.

Skapa sen en fil, och öppna den med någon texteditor och skriv något i den. Sen går du tillbaka till din terminal, och skriver:

```
git add filnamn
```

Det här lägger till din nya fil i vad som kallas för indexet. Indexet kan ses som ett mellansteg där innehåll som ligger i din mapp lagras inför en commit. Därefter behöver du paketera ihop det du gjort till en commit, det görs enklast med kommandot:

```
git commit -a -m "meddelande"
```

Där flaggan ”-a” innebär att git automatiskt add:ar filer som redan spåras, och ”-m” att vi ger meddelandet direkt i terminalen (annars öppnas en text-editor), notera att meddelandet måste stå inom citat-tecken!

Det sista steget är att se till att dina ändringar också hamnar på GitHub. Vilket görs med kommandot

```
git push
```

Om du vid ett senare tillfälle sitter vid en en annan dator som också har en klon av ditt repository som inte har de senaste ändringarna behöver du innan du kan push:a använda

```
git pull
```

för att dra ner den kanoniska versionen (den som ligger på KTH GitHub).

4 Arbetsflöde

När du arbetar med Git så gäller det framförallt att komma ihåg:

- Nya filer behöver läggas till med ”git add”.
- Om du inte använder ”-a” när du commit:ar behöver du även lägga till gamla filer igen.
- Försök i någon mån göra ”lagom” stora commit:s, som är lätta att beskriva i ett kort meddelande.
- Saker hamnar inte på GitHub förrän du har push:at.
- Om din lokala kopia ligger efter GitHub behöver du först spara dina eventuella lokala ändringar med ”git commit”, använda ”git pull”, och sen lösa en eventuell merge-konflikt innan du kan push:a.

Något om Commit-meddelanden

Commit-meddelanden kan ibland vara svåra att skriva, särskilt om en arbetat på något ett tag, och commit:en blivit väldigt stor. En tumregel kan vara följande citat:

'If the title of your commit contains the word and, then the commit is too large.'

Ett bra sätt att motverka för stora commits är att kontinuerligt skapa nya commits i och med att du arbetar. Om du dessutom push:ar dina ändringar till GitHub har du dessutom en back-up på ditt arbete.