TDDC77 - Objektorienterad programmering

Laboration 0

Institutionen för datavetenskap
Linköpings universitet

Introduktion

Denna laboration kommer att gå igenom lite förkunskaper inför kursen, TDDC77, som också är väldigt viktiga och användbara att kunna för resten av era studier. Labb 0 fokuserar på Linux-terminalen och git, där ni kommer att få lära er hantera skoldatorerna och använda lite bra verktyg för att enkelt hantera er kod och inlämningarna av labbarna.

Terminalen

För att öppna terminalen på skoldatorerna så kan du använda kortkommandot CTRL+ALT+T. Det går även att högerklicka i en mapp eller på skrivbordet och sedan klicka på "Open terminal here". Sedan kommer en svart/vit ruta med text att komma upp. Från denna terminal kan du nu skriva kommandon för att navigera dig runt filsystemet.

För mer information kring Linux-terminalen, se gärna:

- https://ubuntu.com/tutorials/command-line-for-beginners#1-overview
- https://docs.cs.cf.ac.uk/notes/linux-shell-commands/

För mer information kring terminalen i andra platformer se gärna följande:

- https://riptutorial.com/cmd
- https://www.makeuseof.com/tag/mac-terminal-commands-cheat-sheet/

Linux-laborationen*

1. Läs igenom kapitel 2-6 i följande artikel:

   https://ubuntu.com/tutorials/command-line-for-beginners#1-overview

   Skriv alla kommandon som ligger i artikeln i en fil som heter `linux_article.txt`. Du ska beskriva kortfattat vad varje kommando gör.

2. Genomför följande scenario och sedan spara alla kommandon som använts i ordning i en fil som heter `linux_scenario.txt`. Filen ska lämnas in på samma sätt som beskrivits i nästa del, `Git`, steg 7.

   "Du ska skapa en mapp som heter `codebase` i hem-katalogen. Mappen ska innehålla en fil som heter `date.txt` som ska innehålla nuvarande tid och datum. Ni ska använda `date` kommandot och `>` symbolen för att skapa `date.txt` filen. Sedan ska ni använda `nano` editorn för att skapa en fil som heter `info.txt` och spara följande uppgifter i den:

Name: *ditt_namn*
Family name: *ditt_efternamn*
Name2: *ditt_namn*
Family name2: *ditt_efternamn*

Nu ska ni skapa en annan mapp i hemkatalogen som heter `codebase_v2`. Nu ska ni kopiera alla filer som ligger i `codebase` mappen till `codebase_v2`. Nu ska ni radera `codebase` mappen samt med alla filer som ligger i den."

Note: Om du använder ett annat operativsystem än Linux på egen dator då kan du göra den här uppgiften på våra datorer i su-salarna, via thinLinc eller i följande webbaserade Linux-terminalen: https://cocalc.com/doc/terminal.html

Git

Git är ett versionshanteringssystem som är väldigt populärt inom "datavärlden". Vi använder git för att effektivt kunna dela kod, samarbeta på ett projekt eller ha en backup på sin kod. Git är idag ett av de vanligaste verktygen som man behöver kunna på arbetsmarknaden som utvecklare. Det finns mycket att lära sig om git, men i denna labb kommer ni få testa grunderna som ni kommer behöva under resten av era studier och kanske resten av era arbetsliv.

GitLab

GitLab är en plattform för mjukvaruutvecklare som är git-baserad och tillhandahåller massor med verktyg för driften, delandet och utvecklingen av mjukvara. På GitLab går det att skapa projekt där all kod och tillhörande filer till ett projekt lagras. Det går även att se historiken på alla filer, alla ändringar som gjorts, vem som gjort vilka ändringar, och mycket mycket mer. Platsen där alla filer sparas kallas för ett repository och benämns ofta som "repo".

Ett typiskt flöde som man har när man arbetar med git är att man först klonar ett repo. Vad det innebär att klona ett repo är att man hämtar ner en kopia av ett repo med alla mappar och filer och sparar dem lokalt. Kloning sker med kommandot *`git clone URL`*, där *`URL`* är länken till ett repo. När man har klonat ett repo kan man fritt göra de ändringar man vill lokalt. När man är nöjd med en ändring lägger man till ändringarna genom att skriva *`git add filename`*. Det går bra att lägga till flera filer och mappar. Med ändringen sparad lokalt går det att göra en commit genom att skriva *`git commit -m "ett meddelande"`*. En commit innebär att man sparar de ändringarna man lokalt gjort till git-repot. Till varje commit behöver man skriva ett meddelande. Meddelandet ska förklara vilka ändringar som skett och vara kortfattad. Med sina ändringar sparade går det nu att ladda upp dem, till exempelvis GitLab. Detta gör man genom att använda kommandot *`git push`*. Med en push laddar man upp sina ändringar till ett repo. Mer information om git-kommandon och exempel finns i denna tutorial.

Förberedelser inför Labb 0(Linux)

SSH-nyckel

När man clone/pull/push till ett repo på GitLab kan man göra det via SSH, eller via HTTPS. Skillnaderna som berör er är att när man gör detta via HTTPS så kommer GitLab att fråga om ditt LiU-ID och lösenord varje gång du vill klona/pusha. Däremot, gör man det via SSH, och har lagt till en SSH-nyckel till sitt GitLab-konto, så behöver man inte skriva in sina inloggningsuppgifter varje gång.

En SSH-nyckel är en lång rad av siffror och bokstäver som används för att "logga in" utan användarnamn och lösenord på git. Nyckeln genereras lokalt på datorn och måste sedan läggas till på GitLab.

1. Öppna terminalen
2. Byt ut "liuid123" mot ditt LiU-ID och klistra sedan in kommandot i terminalen

   ```
   ssh-keygen -t rsa -b 4096 -C "liuid123@student.liu.se"
   ```

3. När du får upp prompten *"Enter a file in which to save the key"* tryck enter.
4. I terminalen startar du en ssh-agent genom att klistra in kommandot

   ```
   eval "$(ssh-agent -s)"
   ```

5. Sedan lägger du till nyckeln till agenten med kommandot

   ```
   ssh-add ~/.ssh/id_rsa
   ```

6. För att lägga till din nyckel på gitlab måste du först kopiera den. Det går lätt att göra med följande kommando

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

7. Logga sedan in på *gitlab.liu.se* med ditt LiU-ID. Navigera sedan till **"Settings"** → **"SSH Keys"** och klistra in din nyckel. Efter du skrivit in en lämplig titel klickar du **"Add key"**.

GitLab grupp

För att förenkla inlämningarna av labbarna kommer vi att bjuda in er till en grupp på GitLab. Gruppen kommer att heta **liuid1-liuid2**, där **liuid1** och **liuid2** är din och din labbpartners LiU-ID. Det är i denna grupp som ni kommer att skapa alla era projekt. Alla projekt i gruppen kommer du, din laborationspartner och din labbassistent ha tillgång till.

Git-laborationen*

Nu ska ni få testa på att använda git själva. Denna labb kommer lägga grunden till hur ni senare jobbar med git resten av labbserien. **Skriv ner alla kommandon ni använder för att lösa denna labb och vad de gör!**

1. Ställ dig, student 1, först där du vill ha den nya mappen med alla filer från git-repot. Clonea sedan repot https://gitlab.liu.se/tddc77material/lab0. Dubbelkolla att den nya mappen har skapats!
2. Student 1 ska fylla sina namn och LiU-ID i filen `source_file.java` på valfritt sätt.
3. Skapa ett eget repository i GitLab och pusha upp ändrade filen. Följ instruktionerna på GitLab för att pusha upp en existerande fil.

   För att skapa ett nytt projekt i er grupp klickar ni på **New project**. Döp projektet till något vettigt, exempelvis **Laborationer**. Det är i detta projekt som alla era labbar kommer att laddas upp till. Skapa en ny mapp för varje ny labb och döp mappen till lab0, lab1 etc.

4. Student 2 ska ladda ner repot, som innehåller lab0, och fylla också sina namn och LiU-ID i filen `source_file.java` på valfritt sätt.
5. Student 2 ska pusha upp uppdaterade filen.
6. Student 1 ska ladda ner senaste versionen av filen `source_file.java`
7. Skapa en ny fil som heter `git_scenario.txt`. Uppdatera filen med kommandona som ni använde för att genomföra steg 1-6, samt med beskrivning om vad de gör, i ordning och sedan pusha upp den.

Inlämning

Ingen redovisning behövs för den här labben och det är bara följande filer som ska lämnas in:

- `linux_scenario.txt`
- `linux_article.txt`
- `git_scenario.txt`
- `source_file.java`

Se instruktionerna för inlämning på kurshemsidan.