Instruktioner till Gustav

WSL

Installera

1. Öppna Windows Terminal
2. Skriv:

   ```
   wsl --install
   ```

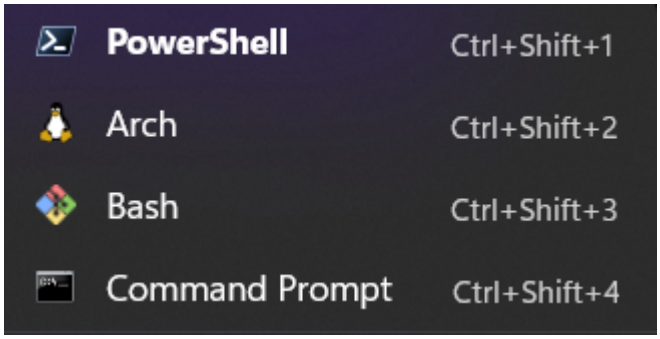
3. Vänta på installation och fyll sedan i konto-uppgifter

Använda

WSL används precis som ett vanligt Linux-distro

Öppna igen

1. Öppna WSL igen genom att klicka på pilen uppe vid ny tabb-knappen:
   

2. I menyn väljer du ditt distro, Ubuntu är standard, jag har Arch (BTW :FarFaceGrinBeam:)
   

3. Vänta på att den virtuella maskinen startar och så är WSL igång

Jekyll

Installera

1. Skriv in:

```
sudo apt install ruby-full build-essential zlib1g-dev
```

2. Lägg till PATH-variabler i din .bashrc-fil:

```
echo '# Install Ruby Gems to ~/gems' >> ~/.bashrc
echo 'export GEM_HOME="$HOME/gems"' >> ~/.bashrc
echo 'export PATH="$HOME/gems/bin:$PATH"' >> ~/.bashrc
```

3. Ladda om bash med antingen:

```
exec bash
```

   Eller

```
source ~/.bashrc
```

4. Installera till sist "gems":

```
gem install jekyll bundler
```

Använda

1. Ladda ner och öppna GitHub repon:

```
git clone https://github.com/Hardbergs-ljud-UF/hardberg-ljud-
uf.github.io.git
cd hardberg-ljud-uf.github.io
```

2. Installera eller uppdatera
    1. Installera...

    ```
    bundler install
    ```

    2. eller uppdatera

    ```
    bundler update
    ```

3. Starta jekyll-servern:

```
bundler exec jekyll serve
```

4. För att automatiskt uppdatera sidan vid ändringar:

```
bundler exec jekyll serve --livereload
```

Mer info

Git

Hur det funkar

Git är ett program för att hantera versioner och samarbeten med kod. När man laddar ner eller skapa ett "git-repo" gör man en mapp. I den mappen finns det en dold mapp som heter `.git`, `.git` innehåller infon som berättar för git att mappen är ett repo, och hur den ska hanteras.

Det finns också något som kallas för "branch". Brancher är olika versioner av kod och de kan ha olika förändringar. På GitHub ligger "origin/main", den branch som är chefen, finns koden inte där finns det inte på hemsidan. ***Men*** För att skydda den koden **måste** man göra en ny branch, ladda upp sin kod där och göra en "merge-request" för att fråga om man får lägga in sin kod i origin/main.

Man kan göra en branch antingen på GitHub eller lokalt via git. (Visar hur längre ner)

Hur man använder det

Git har en del kommandon som man behöver veta:

Command	Options	Function
`pull`		Ladda ner ändringar
`add`	`<file>`	Lägg till ändringar till lokala trackern (oftast kan `<file>` bara vara `.` för nuvarande mappen)
`commit`	`-m "<message>"`	Commita filerna du lagt till med `git add` så att de kommer laddas upp till GitHub
`push`		Ladda upp filerna tillagda med `git commit`
`checkout`	`[-b] <branch>`	Byt branch, med `-b` kan du skapa en ny branch att börja arbeta i. Gör du det behöver du också skriva ett annorlunda `push`-kommando:
`push` (igen)	`--set-upstream origin <ny-branch>`	Sätt upp så din nya branch pushas rätt, behöver bara göras första gången

Hur *vi* använder det

Planering

Här kan du se planeringen med öppna issues som behöver fixas, fixa dem så är vi klara.

Den första vyn är "board", det är en Kanban-bräda där man kan se vad om behöver göras, vad som jobbas på, vad som är klart, och så har jag lagt till en till kategori som visar vad som faktiskt ligger på hemsidan.

Under vyn "Development" ser du allt som inte är klart, och vad dess prioritering är. Att göra saker i prioriteringsordning är inte jätteviktigt då allt ändå **måste** göras.

Den sista vyn är "Overview" och den visar alla issues, både öppna och stängda.

Gör så här

Vi hostar vår sida med hjälp av GitHub Pages. GH Pages uppdaterar hemsidan när det märker en skillnad i `main` branchen, så din kod kommer inte påverka hemsidan förens du gjort en merge-request (som helst ska bli approved av mig men jag tror du kan tvinga en merge oavset (gör inte det)).

När du ska göra en ändring ska du göra en ny branch, antingen på GitHub eller via `git checkout -b <ny-branch>`. Döp den gärna till något som beskriver vad den är till för, särskilt om det är en liten ändring. När du har din nya branch kan du börja ändra kod och spara den med hjälp av `git add .`, och lägga till den i förändringarna med `git commit -m "<message>"`. I commit-meddelandet är det viktigt att åtminstone någorlunda beskriva vad du ändrat så att det är lätt att spåra och ändra om det skulle behövas.

När du gjort dina ändringar laddar du upp koden till GitHub med `git push`. Nu kan alla komma åt koden från vilken dator som helst, så länge de gör `git pull` och `git checkout <branch>`. Om du känner dig klar med din ändring för den här branchen kan du skapa en merge request på GitHub.

Gå in på GitHub så kommer det stå nåt i stil med "`<branch>` is x commit ahead of main". Bredvid bör det vara en grön ruta som säger "compare and create merge request". Klicka på den och fyll i så tydlig info du kan. Om du löser ett specifikt issue bör du lägga till ett "closes `#<issue-name>`" i din beskrivning, då det automatiskt stänger issuet och flyttar det i planeringen.