

Lab setup

Eenmalige setup

R. Grouls, 3 februari 2025

Het instellen van je labomgeving

Iedereen heeft een eigen Virtual Machine (VM) waar we op gaan werken. Iedereen heeft standaard 100 uur beschikbaar. Mocht dat te weinig blijken, dan kan dat op aanvraag worden uitgebreid. Zet de VM uit als je hem niet gebruikt, het kost 0.55€ per uur per persoon dus dat tikt op een gegeven moment wel aan. Elke avond om 23:00 sluit de VM automatisch af, om te voorkomen de VM vergeten en hem dagenlang aan laten staan.

Deze instructies leren je:

- Een ssh-keypair aan te maken
- Via VS Code een ssh-verbinding te maken met de VM
- De code van de repository te clonen naar je server
- Een environment te maken via uv

Installeer VS code

We gaan met VS code werken.

Dit kun je installeren via <https://code.visualstudio.com>

Registreren voor de VM

- A. Zorg dat je een microsoft account hebt, en dat je de uitnodiging voor het lab op je microsoft account ontvangt.
 - 1. Als je nog geen microsoft account hebt, kun je er een aanmaken met het e-mailadres waarop je gemaild bent
 - 2. Als je een al een ander microsoft account hebt dat je graag wilt gebruiken, geef dat e-mailadres dan door, dan stuur ik een nieuwe uitnodiging.
- B. Open de email van azure-noreply@microsoft.com met de titel "Registreer u voor het lab " en klik op de link.
- C. Als je de melding krijgt dat je geen toegang hebt: check dat je ingelogde microsoft account en het e-mailadres waarop je bent uitgenodigd dezelfde zijn.

Lokaal - Vind je terminal

- Voor Mac: zoek naar “terminal” (bv met spotlight) en open hem
- Voor Windows:
 - Installeer git bash <https://gitforwindows.org/>
 - Soms is het nodig OpenSSH te installeren om het ssh commando te gebruiken. Op moderne machines staat het meestal wel. Als je twijfelt, check dit via start > settings > apps > Optional features > zoek OpenSSH. Installeer indien nodig
 - Zoek en open “git bash”
 - Als ik het in dit document over de terminal heb, dan bedoel ik daarmee voor windows de “git bash”

Lokaal - Maak een ssh-keypair

- Een ssh-key is een manier om veilig toegang te krijgen tot een machine.
- Het bestaat uit twee bestanden: de public key (die je met iedereen mag delen) en de private key (die je voor jezelf houdt)
- Zonder ssh-key moet je elke keer als je bijvoorbeeld van folder wisselt, een wachtwoord invoeren.
- Een ssh-key is misschien wat ingewikkelder aanmaken, maar scheelt je later wel het gedoe van continue dat wachtwoord invoeren.

Lokaal - Maak een ssh-keypair

- Als je al een ssh-key hebt, mag je dit overslaan!
- Ga in je terminal naar de `~/.ssh` folder via ``cd ~/.ssh`` en geef het ``ls`` comma

- Op mijn machine ziet dat er zo uit:

```
~/.ssh  
> ls  
config          id_rsa          known_hosts.old  scep  
id_ed25519      id_rsa.pub      podman-machine-default  scep  
id_ed25519.pub  known_hosts     podman-machine-default.pub
```

- Je ziet dat ik een config bestand heb, en meerdere bestanden waar een `.pub` versie van bestaat. De `id_ed25519` en `id_rsa` bestanden zijn bijvoorbeeld ssh-keypairs
- Op jouw lokale machine is deze folder waarschijnlijk leeg.

Lokaal - Maak een ssh-keypair

- Als je al een ssh-keypair hebt, mag je dit overslaan!
- Geef in je terminal het commando:

`ssh-keygen t ed25519 -C "naam@email.com"`

Waarbij je je eigen email invult (zie afbeelding)

- Je mag je wachtwoord leeg laten, dat is wel zo handig. Mocht je een ssh-key hebben met een wachtwoord, dan kun je gewoon een nieuwe aanmaken zonder wachtwoord. Verwijder dan eerst de oude (tenzij je hem ergens voor nodig hebt, dan maak je een nieuwe met een andere naam). De terminal vertelt je waar je key wordt opgeslagen. Standaard in

`~/.ssh/id_ed25519`

- wat prima is. Let op! Check wat de locatie is. Je hebt deze locatie later nodig!!

```
azureuser in 🌐ML-RefVm-335705 in ~ took 7s
> ssh-keygen -t ed25519 -C "raoul.grouls@han.nl"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/azureuser/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/azureuser/.ssh/id_ed25519
Your public key has been saved in /home/azureuser/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:Lc9NXpjP0f0fh6P2bb0Ba06LNsHqgagdNYt01Y8jQKc raoul.grouls@han.nl
The key's randomart image is:
+--[ED25519 256]--+
|
|   .
|  . o .
|   E . .
|    o. oo
|   . +S.oo+o.
|  . = ++.*.+o+
|   + o .+ ++Bo+
|  o .   ..o*..+B
|  . . . .oo+.o*
| .....
```


Lokaal - Copy de public key

- Open lokaal (op je laptop) een tweede terminal, en navigeer naar je lokale `~/.ssh` folder met behulp van `cd`
 - Lokaal staan in je `~/.ssh` folder de ssh-keys. We gaan de public key kopiëren. Geef lokaal het commando

```
cat ~/.ssh/id_ed25519.pub | clip
```

Voor windows, en voor Mac/Unix:

```
cat ~/.ssh/id_ed25519.pub | pbcopy
```

- Als jouw computer zegt “No such file or directory”, dan... wel... je raadt het al... is er geen keybestand op de plek die jij aanwijst. Dubbelcheck of de key inderdaad `id_ed25519.pub` heet, en staat waar je denkt dat hij staat. Je kunt de inhoud van je `.ssh` folder checken via `ls ~/.ssh`
- Je gaat deze key straks plakken in je VM
- Toelichting bij het commando:
 - In nederlands lees je dit als: print de inhoud van de public key, en stuur de output daarvan door naar mijn clipboard.
 - `cat file` print de inhoud van het bestand `file`, we printen dus de inhoud van de public key met `cat ~/.ssh/id_rsa.pub`
 - door daar `| clip` of `| pbcopy` achter te zetten sturen we de output van `cat` door naar `clip`, wat je clipboard is. Nu kun je de inhoud van de file ergens anders plakken.
 - Je kunt natuurlijk ook het doorsturen weghalen, in dat geval moet je met de muis de tekst kopiëren nadat je `cat` hebt gebruikt.

VM - Start je VM

- Ga naar <https://labs.azure.com/virtualmachines>
- Klik op de button waar hier rechts “gestopt” bij staat
- Je VM wordt opgestart, dit duurt ongeveer twee minuten
- Als je “Running” of “Actief” ziet staan, klik op het kleine computer icon rechts onderin en kies “Connect via SSH”
- De eerste keer zul je een wachtwoord moeten aanmaken. Azure stelt bepaalde eisen aan je wachtwoord (hoofdletters, cijfers, tekens).
- Elke keer dat je dingen wilt aanpassen op je VM, zul je dit wachtwoord moeten invoeren. [Sla het op!](#)



VM - Log in via ssh

Als je op “Connect via SSH” op <https://labs.azure.com/virtualmachines> klikt zie je een lang commando. Plak dat in je lokale terminal.

De eerste keer krijg je de vraag of je de key-fingerprint van de server wilt toevoegen. Geef yes als antwoord.

Het wachtwoord waar om wordt gevraagd is je VM wachtwoord. Je bent nu ingelogd, zie screenshot. Dit is de terminal van de VM.

Als het goed is zie je kleurtjes. Typ anders **zsh**


```
ED25519 key fingerprint is SHA256:I9ZfLL0Sf7Zn5ogNAs0XpUe7Jahsk8k6bpCRy2gAfiA.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '[ml-lab-499b0cc0-f0fa-444d-a905-b2d2bead7477.westeurope.cloudapp.azure.com]:57696' (ED25519) to the list of known hosts.  
mladmin@ml-lab-499b0cc0-f0fa-444d-a905-b2d2bead7477.westeurope.cloudapp.azure.com's password:  
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-1022-azure x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
System information as of Fri Apr 22 09:28:36 UTC 2022  
  
System load:  0.07               Processes:            144  
Usage of /:   11.1% of 28.9GB    Users logged in:     1  
Memory usage: 1%                IPv4 address for eth0: 10.0.0.12  
Swap usage:   0%  
  
0 updates can be applied immediately.  
  
Last login: Fri Apr 22 09:26:14 2022 from 62.166.163.14  
λ ML-RefVm-812132 ~ +
```

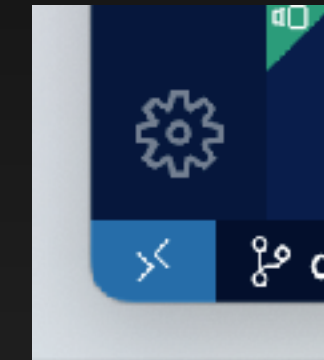
VM - Paste de public key

je kunt nu de inhoud van je public key gaan plakken naar de VM

- ga naar de terminal waarmee je op je VM bent ingelogd en navigeer naar de `.ssh` directory met het commando `cd ~/.ssh`
- In die folder staat een bestand, namelijk `authorized_keys`. Check met `ls`
- Open nu het `authorized_keys` bestand met een text editor, bijvoorbeeld nano:
`nano authorized_keys`
- Hier staat waarschijnlijk al een public key, namelijk de mijne. Laat die daar staan, dat is handig als ik je wil helpen.
- Je hebt een paar slides terug je key gekopieerd. Plak nu met `cmd+v` of `shift+insert` of `win+v` of je muis de inhoud van `id_ed25519.pub` in het `authorized_keys` bestand op de VM

VS Code - de VM toevoegen

- Ga naar VS Code en installeer Remote-SSH extension: ga naar extensions  zoek naar *remote ssh* en klik *install*.
- Linksonder in je VS code kun je nu verbinding maken via de blauwe pijltjes
- Kies “connect to Host... Remote-SSH”
- Kies “+ Add New SSH Host...”
- Copy-paste het commando om met ssh te verbinden dat je op <https://labs.azure.com/virtualmachines> via “Connect via SSH” krijgt (zie slide “VM - login via SSH”)



Enter SSH Connection Command

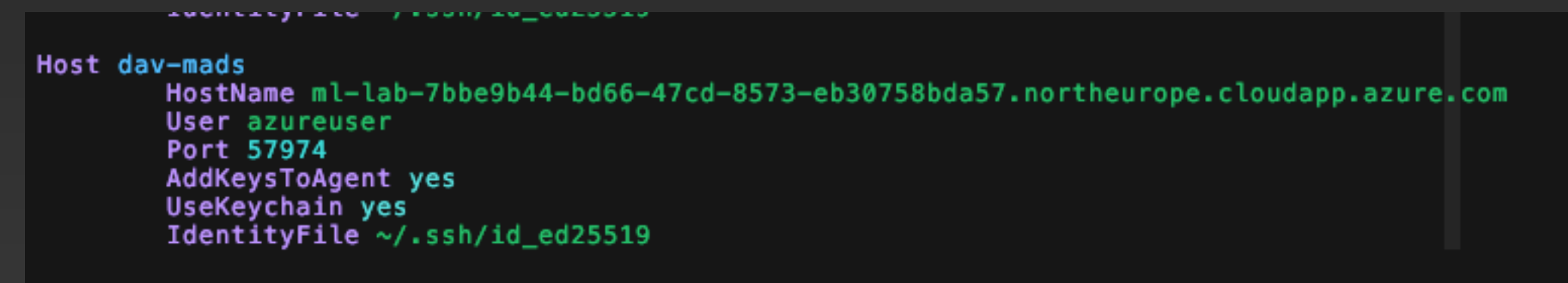
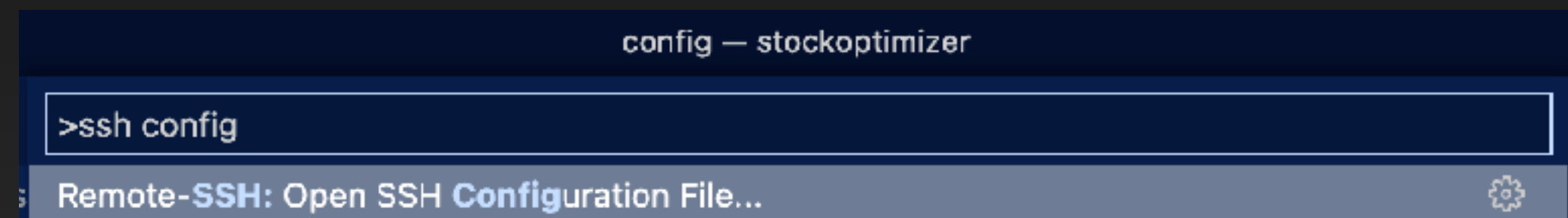
ssh hello@microsoft.com -A

Press 'Enter' to confirm your input or 'Escape' to cancel

- Bij “Select SSH configuration file to update”, kies je het .ssh/config bestand in je eigen gebruikersaccount

VS code verbinden

- Je kunt nu eventueel de naam wat makkelijker maken. Dat doe je door via VS code je configuratiefile aan te passen:
- Zoek door de commandos met **ctrl+shift+P** of **cmd+shift+P**
- zoek "Remote-SSH Open SSH Configuration File..."
- Pas de naam na Host aan naar een makkelijk te onthouden naam
- Als jouw key een andere naam heeft dan standaard, dan kun je dat in je config toevoegen via **`IdentityFile`**



Ok, dat was een heleboel!

Als je tot hier gekomen bent, en alles gelukt is, dan is de basis klaar!

De voorgaande paginas hoef je in principe niet meer te gebruiken.



In principe waren de meeste van deze stappen eenmalig, en is het inloggen voortaan een stuk eenvoudiger...

Workflow

Inloggen op je server is voortaan relatief simpel:

- Start je machine op via <https://labs.azure.com/virtualmachines>
- Verbind VS code via “Remote-SSH: Connect to Host...”
- Kies de naam van de server (de makkelijke naam die je net bedacht hebt)
- geef aan dat de VM een linux server is als daarom wordt gevraagd.
- Open een folder om in te werken; geef aan dat je de inhoudt vertrouwt als daarom gevraagd wordt
- Als je klaar bent, sluit de verbinding via “Close remote connection”
- Stop je machine weer via de website!

VM - Installeer de omgeving

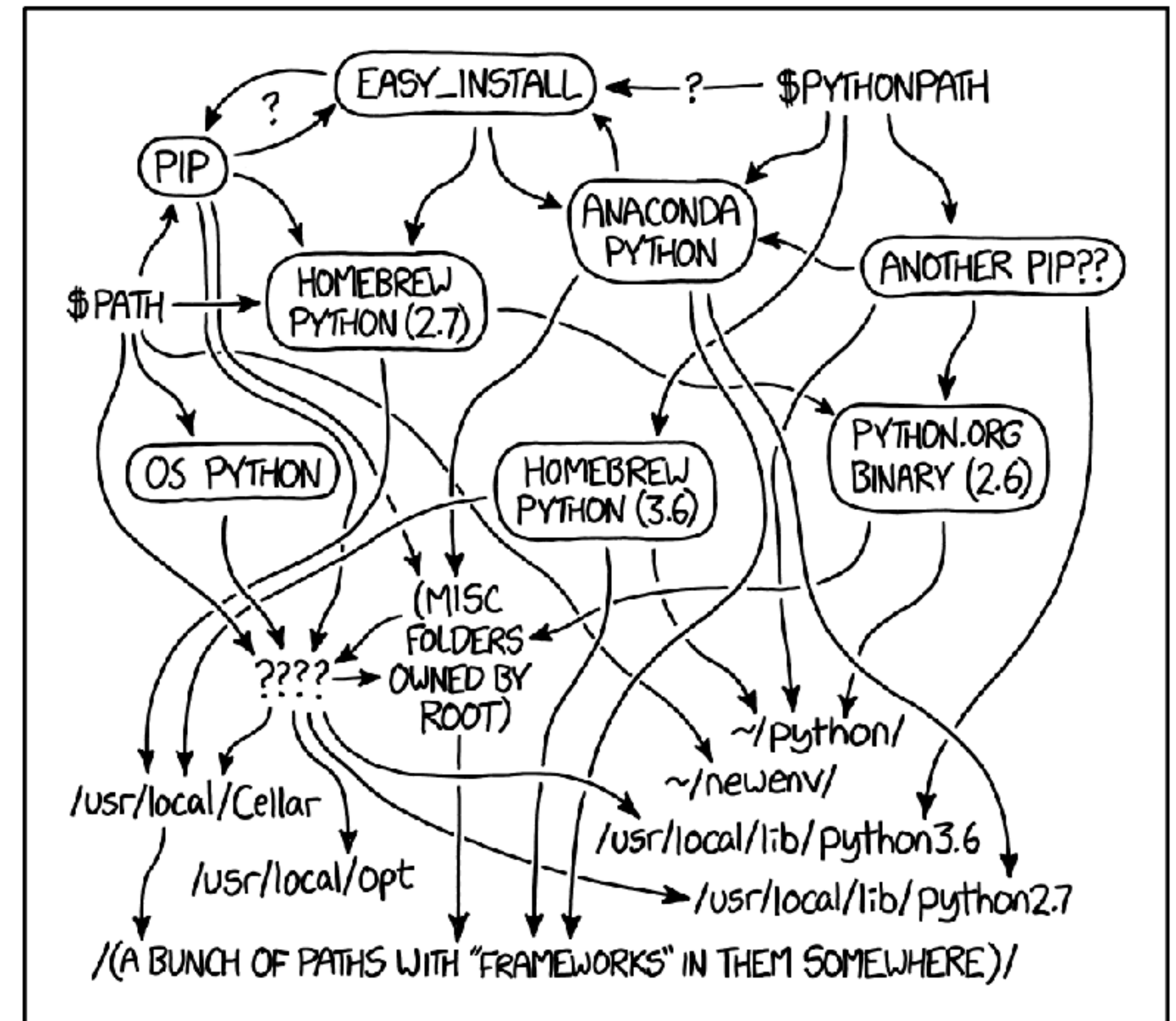
Veel van jullie zijn waarschijnlijk gewend om met conda of pip te werken.

Dat gaan wij niet doen:

We gebruiken uv: <https://docs.astral.sh/uv/>

Dit managed zowel je python versies eenvoudig, als je virtual environments

Zie <https://github.com/raoulg/codestyle> voor meer achtergrond & handleiding bij uv (start bijvoorbeeld bij de entry over `project.toml for dependencies`)



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

VM - setup

- In de root directory van de machine vind je de cursusfolder. Open VScode in die folder. **Dit is belangrijk!** Als je je project in de parent folder opent werkt VScode niet goed met je python omgeving samen. Elke cursus weer zijn er mensen die dit toch niet doen; de regel is dat je de volgende les trakteert als je probleem op te lossen is door VScode in de juiste folder te openen.
- In het geval dat je geen projectfolder hebt, kun je hem altijd clonen vanaf git:
 - Ga naar de repo op github.com en klik op de groen “code” knop. Kopieer de url die er bv zo uit ziet: <https://github.com/raoulg/MADS-DAV.git>
 - In je terminal doe je **git clone https://github.com/raoulg/MADS-DAV.git**
- In de projectfolder staat een **pyproject.toml** bestand. Dit beschrijft de dependencies die we gebruiken. Het is leesbaar, dus bekijk het gerust met vscode (of gebruik cat)
- Run **`git pull`** in de projectfolder (tenzij je hem net zelf hebt gecloned). Het is mogelijk dat ik nog wat laatste wijzigingen heb gedaan, die “pull” je dan van GitHub naar de VM.

VM - UV

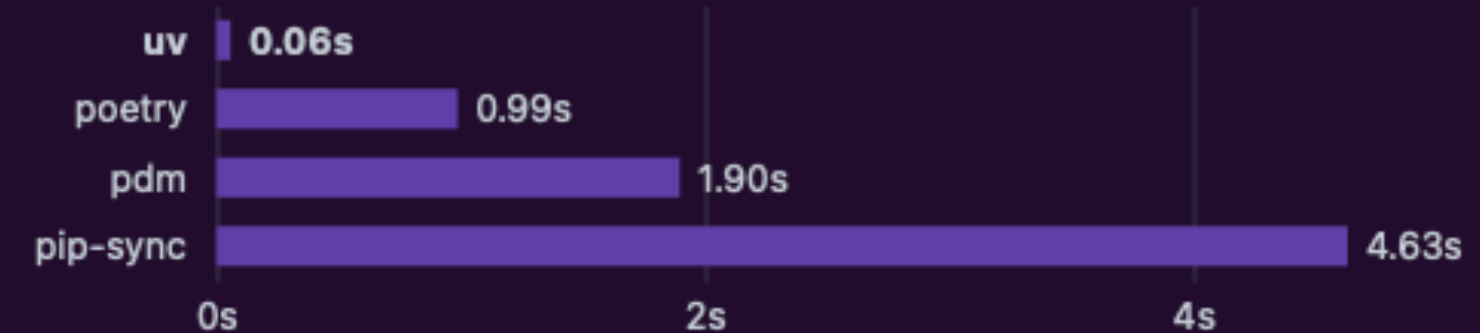
- Check of uv is geïnstalleerd met `which uv`, wat een locatie moet terugsturen, of `uv --help`, wat uitleg moet geven
- De `--help` van uv is sowieso erg goed en uitgebreid.
- Je kunt ook van subcommandos `--help` doen, bijvoorbeeld `uv sync --help` geeft je allerlei opties en uitleg over het sync commando.
- Als er geen uv op je machine is, volg de instructies op <https://docs.astral.sh/> om uv te installeren.

VM - UV

- Wat je hier links leest komt van de uv website af
- Vraag jezelf af of je in de supermarkt altijd de langste rij kiest.
- Aangezien dat waarschijnlijk niet zo is, ben je waarschijnlijk wel gemotiveerd om uv te gebruiken dat 10x-100x zo snel is als pip.

UV

An extremely fast Python package and project manager, written in Rust.



Installing *Trio*'s dependencies with a warm cache.

Highlights

- 🚀 A single tool to replace `pip`, `pip-tools`, `pipx`, `poetry`, `pyenv`, `twine`, `virtualenv`, and more.
- ⚡ 10-100x faster than `pip`.
- 🐍 Installs and manages Python versions.
- 🛠️ Runs and installs Python applications.
- ✨ Runs scripts, with support for inline dependency metadata.
- 📁 Provides comprehensive project management, with a universal lockfile.
- 🐍 Includes a `pip-compatible` interface for a performance boost with a familiar CLI.
- 🏠 Supports Cargo-style workspaces for scalable projects.
- 💾 Disk-space efficient, with a global cache for dependency deduplication.
- 📦 Installable without Rust or Python via `curl` or `pip`.
- 🖥️ Supports macOS, Linux, and Windows.

VM - UV

- Lees de introductie op de website van uv
- In plaats van ``pip install xyz`` doe je voortaan ``uv add xyz`` waarbij `xyz` de naam van de dependency is, bijvoorbeeld ``uv add numpy``.
- Alles wat je toevoegt wordt aan je `pyproject.toml` file toegevoegd, automatisch.
- Er is voor dit project al een `pyproject.toml` file, en je wilt geen dependencies aan mijn lesmateriaal toevoegen (doe dat ik je eigen project).
- Mogelijk heb je ook al een `.venv` folder met een virtual environment.
- Je kunt je `.venv` synchroniseren met de `pyproject.toml` file via `uv sync`, dan wordt alles geïnstalleerd wat in je `pyproject` staat beschreven. Gebruik `uv sync --all-extras` om ook de dependencies onder “optional-dependencies” te installeren, wat wel aanbevolen is.
- Als je je environment wilt gebruiken vanuit de shell, activeer dan je environment met `source .venv/bin/activate`

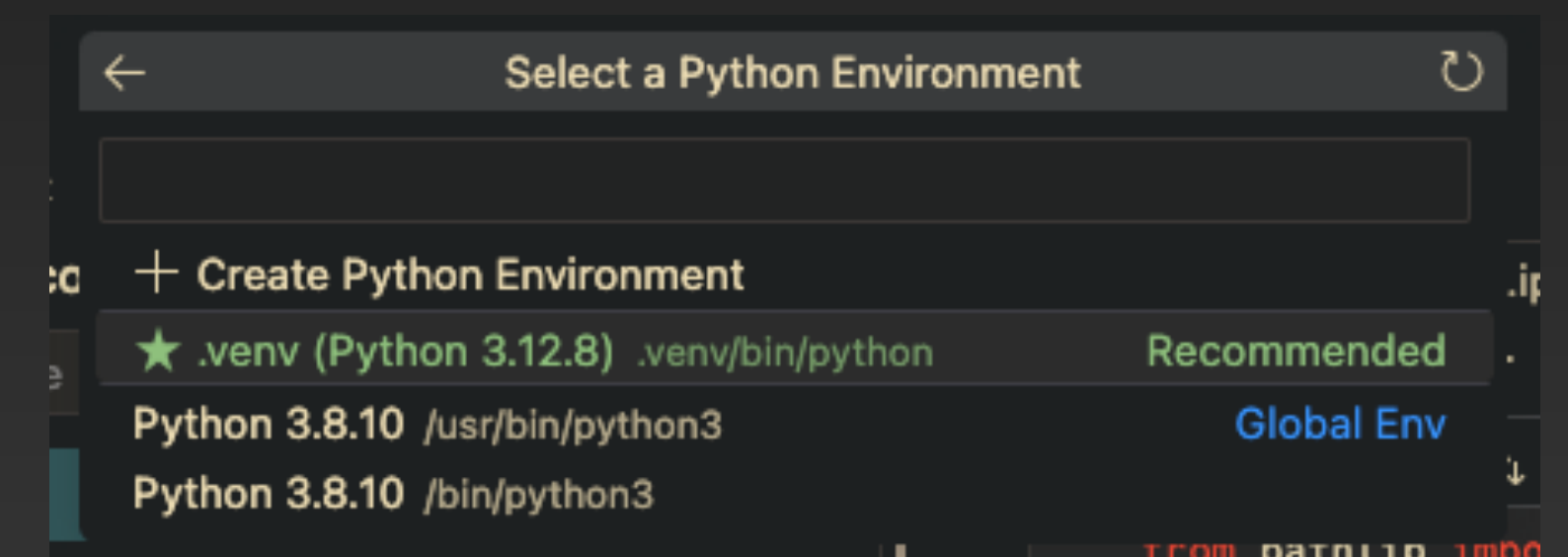
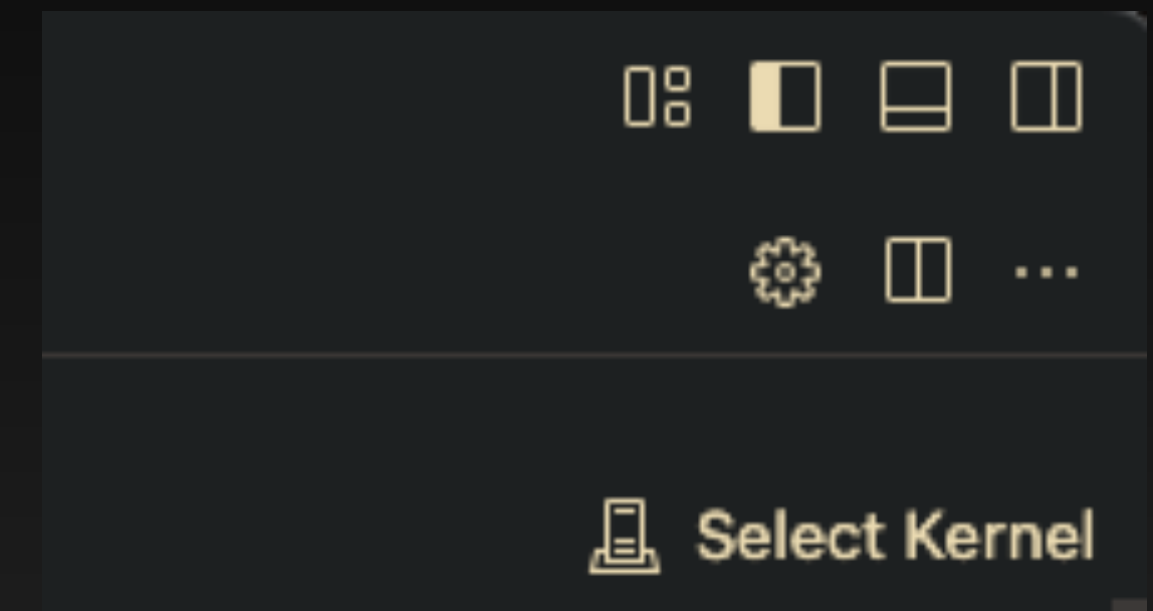
In jupyter een kernel selecteren

- Als je in VScode met python en Jupyter wilt werken, heb je de extensions nodig
- Ga naar extensions in VScode, en zoek “python” en “Jupyter”
- Soms staan ze al lokaal geïnstalleerd, maar moet je nog klikken op “install in SSH”



In jupyter een kernel selecteren

- Open nu een jupyter notebook van de cursus
- Je kunt nu de kernel selecteren in je jupyter notebooks: klik rechtsboven op “**select kernel**”, daarna mogelijk op **`Python environments`**.
- Dit werkt ALLEEN goed als je
 - 1) een **.venv** hebt aangemaakt
 - 2) VScode in de juiste folder hebt geopend (en anders trakteren :D)
- Waarschijnlijk herkent vscode je **.venv** folder automatisch en beveelt deze aan met een ster *
- Zo niet, reload VScode (ctrl+shift+p reload window)
- Je kunt nu notebook runnen vanuit de betreffende environment



Werken met git

- Git is in het begin verwarrend, als je daar nog nooit mee hebt gewerkt. Maar het is nu eenmaal (bijna) de enige reële manier als je met meerdere mensen tegelijkertijd aan code wilt schrijven.
- Jullie kunnen geen code pushen naar mijn repository. Maar je kunt wel branchen. Als je niet weet wat push of branch betekent, lees de git crash course in de references folder
- Als je al wilt beginnen met dingen testen, doe dat in je eigen branch:
 - Open de terminal van je VM
 - navigeer naar de lesfolder
 - maak een nieuwe branch genaamd les1 met `git checkout -b les1`
De `-b` heb je alleen de 1e keer nodig.
 - Je ziet nu dat je niet meer in master zit, maar in les1, je kunt nu veilig werken en nog steeds updates ontvangen van de lessen.

```
rename notebooks/{1_pytorch_intro => 2_convolutions}/06_resnet.ipynb (100%)
λ ML-RefVm-812132 ML22 → λ git master → git checkout -b les1
Switched to a new branch 'les1'
λ ML-RefVm-812132 ML22 → λ git les1 → poetry update
Updating dependencies
```


Werken met git

- **Life-Pro-Tip:** Als je mijn notebooks wilt aanpassen, en wijzigingen opslaan, geef je eigen notebooks een andere naam dan de mijne, in je eigen branch. Op die manier krijg je nooit conflicten:
 1. Copy een notebook en hernoem het, bijvoorbeeld `01_les1.ipynb` wordt `01_les1_WillemvO.ipynb`
 2. Werk in je notebook, sla de wijzigingen op via git commit.
 3. **Als alles is gecommit (check dit met `git status`)** kun je terug naar master, een git pull doen, weer terug naar je eigen branch en 'git merge master' doen. Alle updates worden dan geïntegreerd met je eigen branch.

Als dit abracadabra is, **lees de git crash course**.

Werken met git

- Ik maak regelmatig updates aan de code. Je kunt de laatste versie altijd binnenhalen via **git pull**. Minimale instructies:
 - Sla je werk op in je les1 branch:
git checkout les1
git add .
git commit -m "mijn eerste commit"
 - ga terug naar master en pull:
git checkout master
git pull
 - Je bent je werk niet kwijt. Dat zit in les1. Als je daarna
git checkout les1
- Voor meer instructies: bestudeer het materiaal in de references folder.

```
rename notebooks/{1_pytorch_intro => 2_convolutions}/06_resnet.ip
λ ML-RefVm-812132 ML22 → λ git master → git checkout -b les1
Switched to a new branch 'les1'
λ ML-RefVm-812132 ML22 → λ git les1 → poetry update
Updating dependencies
```


Crash course command line

De enige commando's die je hier kunt geven zijn tekst commando's. Ik heb een paar tweaks toegevoegd om het leven makkelijker te maken.

- `ls` : List. Geeft een overzicht van alle zichtbare bestanden. Blauw zijn directories. Probeer ook `lsd`
- `la` : shortcut voor `ls -A` , toont ook verborgen bestanden (die beginnen met een `.`)
- `cd` : Change directory
- `pwd` : Present Working Directory
- `mkdir` : Make directory.
- `df -H` : laat gebruikte schijfruimte zien
- `du -sh *` : laat de schijfruimte voor deze folder zien

```
λ ML-RefVm-812132 ~ → la
total 176K
-rw----- 1 mldadmin mldadmin 429 Apr 22 09:26 .bash_history
-rw-r--r-- 1 mldadmin mldadmin 220 Feb 25 2020 .bash_logout
-rw-rw-r-- 1 mldadmin mldadmin 67 Apr 19 14:00 .bashrc
-rw-r--r-- 1 mldadmin mldadmin 3.7K Feb 25 2020 .bashrc.bak
drwx----- 3 mldadmin mldadmin 4.0K Apr 19 14:08 .cache
drwxrwxr-x 4 mldadmin mldadmin 4.0K Apr 19 14:08 .local
drwxr-xr-x 12 mldadmin mldadmin 4.0K Apr 19 13:50 .oh-my-zsh
-rw-r--r-- 1 mldadmin mldadmin 904 Apr 19 13:50 .profile
drwxrwxr-x 14 mldadmin mldadmin 4.0K Apr 19 14:02 .pyenv
drwx----- 2 mldadmin mldadmin 4.0K Apr 22 09:32 .ssh
-rw-r--r-- 1 mldadmin mldadmin 0 Apr 19 13:48 .sudo_as_admin_successful
-rw----- 1 mldadmin mldadmin 2.9K Apr 22 09:32 .viminfo
-rw-rw-r-- 1 mldadmin mldadmin 48K Apr 19 13:54 .zcompdump
-rw-rw-r-- 1 mldadmin mldadmin 50K Apr 19 14:09 .zcompdump-ML-RefVm-812132-5.8
-rw-rw-r-- 1 mldadmin mldadmin 97 Apr 19 13:50 .zprofile
-rw----- 1 mldadmin mldadmin 1.9K Apr 22 09:36 .zsh_history
-rw-rw-r-- 1 mldadmin mldadmin 4.1K Apr 19 14:09 .zshrc
-rw-rw-r-- 1 mldadmin mldadmin 4.1K Apr 19 14:09 .zshrc.bck
-rw-rw-r-- 1 mldadmin mldadmin 23 Apr 19 13:50 .zshrc.pre-oh-my-zsh
drwxrwxr-x 3 mldadmin mldadmin 4.0K Apr 19 13:48 serverinstall
λ ML-RefVm-812132 ~ →
```

Codestyle

Heel veel code “werkt”. Helaas is maar weinig code zo geschreven dat hij goed te onderhouden en aan te passen is.

Beginners schrijven vaak code als een kaartenhuis; de geringste aanpassing kost uren, of laat alles in elkaar storten.

Je hebt waarschijnlijk al kennis gemaakt met de <https://github.com/raoulg/codestyle> repository; in deze cursus zul je daar meer gebruik van moeten gaan maken. Ik verwacht dat je minstens alles tot en met “make a proper module” leert beheersen aan het einde van deze cursus.

Topic	Explore	Consolidate	Cooperate	Deploy
never hardcode	💡	🏆	🏆	🏆
Prefer pathlib.Path over os.path	💡	🏆	🏆	🏆
pyproject.toml for dependencies	💡	🏆	🏆	🏆
organize your folders	💡	🏆	🏆	🏆
Add a README	💡	🏆	🏆	🏆
isolate your settings	💡	💡	🏆	🏆
Git	💡	💡	🏆	🏆
Use formatters and linting	🐌	💡	🏆	🏆
use logging	🐌	💡	🏆	🏆
Use typehinting	🐌	💡	🏆	🏆
Make a proper module	🐌	💡	🏆	🏆
Open-Closed Principle	🐌	💡	💡	🏆
Makefiles or shell scripts	🐌	💡	💡	🏆
Encapsulation, SRP	🐌	💡	💡	🏆
Abstract classes (ABC, Protocol)	🐌	🐌	💡	🏆
Write tests (pytest)	🐌	🐌	💡	🏆