Lab 1 - Installing and configuring Git

Last updated by | kees.vanloenen | Jun 29, 2025 at 10:15 AM GMT+2

Lab 1 - Installing and configuring Git

Most of this lab you already have done during the course. Just go through it step by step, and spot the new things.

Exercise 1

1. Check whether *Git* is installed and which version by executing this command:

```
git version
```

   - If a version number is printed then *Git* is installed correctly.
   - If the command doesn't work then download and install *Git* through the installer at: https://git-scm.com/downloads .
2. Make a new repository in `c:/git/demo` :
   - Open *Git Bash*
   - Run the next commands:

```
cd c:
mkdir git
cd git
git init demo   # handy one-liner for 'mkdir demo' and 'git init'
cd demo
```

   - `git init demo` creates a new Git repository in a new directory `demo`

Exercise 2

1. Take a look at the configuration settings of *Git*. There are 3 levels:
   - System-wide configuration
     - for all users / repos on this computer (seldomly needed)

```
git config --list --system
```

     system-wide,
   - Global user configuration ⭐
     - for all your repos (personal preferences: name, email adres, editor, line endings):

```
git config --list --global
```

   - Local repository configuration
     - for one specific repo (collaboration, special configurations using hooks etc, seldomly needed)

```
git config --list --local
```

2. Configure the `user.email` and `user.name` settings in the global user configuration:

```
git config --global user.email "<your e-mail address>"
git config --global user.name "<your full name>"
```

- Afterwards use the correct statement from the previous step to view the results.

3. By default *Git* uses *vim* as text editor. *Git* will open the configured editor when input is needed e.g. when a commit message is missing or during an interactive rebase. Through the `core.editor` setting this can be adapted.

   Let's use *vim* to adapt the standard text editor to *Visual Studio Code* (or *Wordpad* alternatively).

   - Check whether *VS Code* is installed and can be executed through the command line with the command:

     ```
     code
     ```

   - If this is not the case, install *VS Code* (https://code.visualstudio.com) or use *Wordpad*.
   - Check the configured editor:

     ```
     git config --global core.editor
     ```

   - Not *vim*? Change it to *vim* for this exercise:

     ```
     git config --global core.editor "vim"
     ```

   - Now configure *VS Code* as the standard text editor by editing the global config file with *vim*, our current standard text editor:

     ```
     git config --global -e
     ```

     vim opens, do next steps:
     - Press 'i' to get into the **insert** mode.
     - Move with the arrow keys to the [core] editor settings and change `vim` into `code --wait`.
     - Press ESC to leave the **insert** mode.
     - Type the 3 characters ':wq' and then ENTER to save the changes.
     - Check the new setting.
     - Try if *VS Code* is opened by Git when editing the global config file:

       ```
       git config --global -e
       ```

     - Close *VS Code* afterwards.

4. On Windows, it is recommended to set Git's newline handling to automatically convert line endings to Windows-style (CRLF) when files are coped to you working directory and to Unix-style (LF) on commit.

   CRLF = Carriage Return + Line Feed
   The setting `core.autocrlf` helps avoid issues when collaborating across different operating systems 😎

   - Check the current setting:

     ```
     git config --global core.autocrlf
     ```

- Apply the change:

```
git config --global core.autocrlf true
```

- Check setting again