DEMKIT

INSTALLATION GUIDE

Guide version 3.0

12 March 2023

Gerwin Hoogsteen

Contents

1. Introduction

This document serves as an installation guide of the DEMKit software. Furthermore, it provides some pointers to run the provided demo and how to extend this demo with data generated with the open-source artificial load profile generator.

The first chapter gives an overview of DEMKit resources (code and literature). Followed by a chapter describing the required software and packages to run DEMKit, such that advanced users know all requirements for the operating systems Windows, MacOS and Linux, followed by an extensive step-by-step instruction for Windows based systems for Docker (Chapter 4) and stand-alone (Chapter 5). Instructions on how to interpret the data are provided in Chapter 6. Chapter 7 presents the setup of a coding IDE and some general settings. The last chapter of this guide describes coding guidelines.

1.1 Changelog

Date	**Version**	**Remarks**
12-03-2023	3.0	Version to accommodate relicensing DEMKit
08-03-2023	2.1.0	Bump to include the shift to Github
19-08-2021	2.0.2	Typo’s and updated list of software versions and clarification
14-05-2020	2.0.1	Typo’s
10-05-2020	2.0	Dockerized version update
26-11-2018	1.0.1	Typo’s
07-11-2018	1.0	Initial guide

2. Resources and literature

2.1 Contact

The maintainers of DEMKit can be reached through e-mail for general contact about support, questions, features, etc. The email address is: demgroup-eemcs@utwente.nl
In case there is no prompt response, please contact the main DEMKit maintainer:

Gerwin Hoogsteen
email: g.hoogsteen@utwente.nl

2.2 Code

The code is hosted in Github and mirrored in the Gitlab instance hosted by the University of Twente, the Netherlands. Access to the repository is managed by the DEMKit maintainers.

Specific resources:

DEMKit @ Github:	https://github.com/orgs/utwente-energy/repositories
DEMKit @ UTwente:	https://gitlab.utwente.nl/utwente-energy
Project website:	https://www.utwente.nl/en/eemcs/energy

Github:
Access on Github is done manually by the maintainers who will add you to the appropriate group(s) to access DEMKit and project files.

GitLab:
Only employees and students can have access to the Gitlab instance hosted by the University of Twente. The DEMKit maintainers need to add your account to the right groups, for which you need to sign in once. Use your email username (e.g. hoogsteeng) and password to login.

2.3 Literature

For more background on the architecture of DEMKit and the implemented components and optimization algorithms, we refer to our publications. A comprehensive list of all our publications can be found on our website: https://www.utwente.nl/en/eemcs/energy/publications/

Specific information about the architecture of DEMKit and optimization methods are provided in: G. Hoogsteen, “A cyber-physical systems perspective on decentralized energy management”, PhD dissertation, 2017, University of Twente, the Netherlands.

Available at: https://doi.org/10.3990/1.9789036544320

Specific information about coordination and device level optimizations are provided in: T. van der Klauw, “Decentralized Energy Management with Profile Steering: Resource Allocation Problems in Energy Management”, PhD dissertation, 2017, University of Twente, the Netherlands.

Available at: https://doi.org/10.3990/1.9789036543019

Furthermore models for heating and ventilation systems match the models as presented by R.P. van Leeuwen. For more information on these models and control refer to: R.P. van Leeuwen, "Towards 100% renewable energy supply for urban areas and the role of smart control", PhD Thesis, University of Twente, Enschede, the Netherlands.

Available at: https://doi.org/10.3990/1.9789036543460

2.4 Third party documentation

DEMKit uses external tools and software, as also described in the requirements. The following links provide links to documentation of these tools:

- ALPG: https://github.com/utwente-energy/alpg
- Grafana: http://docs.grafana.org/
- InfluxDB: https://docs.influxdata.com/influxdb/v1.6/
- Python: https://docs.python.org/3/
    - Astral: https://astral.readthedocs.io/en/stable/index.html
    - Eve: http://docs.python-eve.org/en/latest/
    - Numpy: https://docs.scipy.org/doc/
    - Pytz: https://readthedocs.org/projects/pytz/
    - Requests: http://docs.python-requests.org/en/master/
    - ZMQ: https://pyzmq.readthedocs.io/en/latest/api/zmq.html

For project management, the following documentation is available:

- Git: https://git-scm.com/doc

2.5 Tutorials

There is no specific tutorial on using DEMKit yet, please refer to the DEMKit example to get familiar with the tool. However, tutorials on Python may be useful to learn how to adapt and create your own scripts as part of the configuration files. These interactive tutorials should provide a good starting point:
https://www.learnpython.org/
https://www.codecademy.com/learn/learn-python
https://www.datacamp.com/courses/intro-to-python-for-data-science

For Grafana, the getting started guide can be useful:
http://docs.grafana.org/guides/getting_started/

Furthermore, sometimes DEMKit workshops are organized. A recording of a previous workshop can be requested from Gerwin Hoogsteen.

3. Requirements

The DEMKit software makes use of third-party tools and software to simulate, interact, store and visualize data. This chapter gives an overview of the required tools and packages which are generally available for all major operating systems and also for Linux powered embedded systems using ARM processors. Advanced users should be able to install the software given this list and common knowledge of tools such as python, pip and Git. An extensive guide for Windows users is provided in the next chapter, which also presents the information given in this chapter. Instructions to run a simulation are given in Chapter 5.

3.1 Software

The following software, often running as services, is required for DEMKit and the interfacing with the version control system. A tested version number is provided, however we encourage everybody to test/run the most recent stable versions of these tools. Links to documentation, if needed, is provided in the previous Chapter.

- Git:
  - Download: https://git-scm.com/downloads
  - Tested version: 2.16.4
  - Notes: -

Software required for a Docker based installation

- Docker:
  - Download: https://www.docker.com/products/docker-desktop/
  - Tested version: 19
  - Notes: Make sure to share local disks and start-up on login
  - **Not needed when using the Stand-alone method (Chapter 5)**

Software required for a stand-alone setup without Docker

- Python 3:
  - Download: https://www.python.org/downloads/
  - Tested version: 3.9
  - Notes: Make sure you use Python 3.x instead of 2.x. On some operating systems this is achived by running using "python3" instead of just "python". The same applies to installing packages though pip ("pip3"). In Windows, do not forget to tick the add python to PATH-box during installation!
  - **Not needed when using the Docker method (Chapter 4)**

- InfluxDB:
  - Download: https://portal.influxdata.com/downloads
  - Tested version: 1.8.9 (Stick to 1.x versions, 2.x are incompatible)
  - Notes: On Windows, do not forget to start (or make it autostart) before running DEMKit
  - **Not needed when using the Docker method (Chapter 4)**

- Grafana:
    - Download: https://grafana.com/grafana/download
    - Tested version: 8.1.1
    - Notes: On Windows, do not forget to start (or make it autostart) before running DEMKit
    - **Not needed when using the Docker method (Chapter 4)**
- ALPG (recommended)
    - Download: https://github.com/utwente-energy/alpg
    - Tested version: 1.3.1
    - Notes: -
    - **Not needed when using the Docker method (Chapter 4)**

Additional and useful tools (for both setups)

In addition, for development of DEMKit and scenarios, we recommend the following tools, however you are free to use the tools you prefer. **These tools are optional.**

- Notepad++ (Windows only)
    - Download: https://notepad-plus-plus.org/downloads/
    - Tested version: 7.5.9
    - Notes: This is not the normal notepad and provides features like code highlighting. Ideal for quick changes to e.g. config files.

- PyCharm IDE
    - Download: https://www.jetbrains.com/pycharm/download/
    - Tested version: 2022.2
    - Notes: An academic license can be obtained free of charge for employees of most academic institutions. This lets you use the Professional version. Otherwise, the free Community version is likely to provide enough features.

3.2 Python packages

In addition, the following python modules are, which can be installed using "pip", e.g. using the following command in a command prompt:

```
python –m pip install <package>
```

Note to check if the right version of python (2.x or 3.) is used. Some operating systems require admin privileges, e.g. "sudo python …" or an "elevated prompt" (Windows) by running the console as admin (right-click on cmd.exe, "run as administrator").

Required Packages:

- Astral==1.10.1
- numpy
- requests
- pytz
- zmq

  Commandline:

  ```
  python –m pip install astral==1.10.1 numpy requests pytz zmq
  ```

Optional Packages:

- eve
- pyserial

  Commandline:

  ```
  python –m pip install eve pyserial
  ```

4. Installing and Running DEMKit with Docker

This is a step-by-step guide to install DEMKit. The guide is mostly focussed on Windows users, but similar steps can be executed by Linux and MacOS users as well.

For Linux and Mac users, equivalent scripts for Linux and MacOS users are available, execute the *.sh version instead of *.bat file. **NOTE**: When writing this document, no MacOS system was available to test the instructions.

4.1 Installing tools

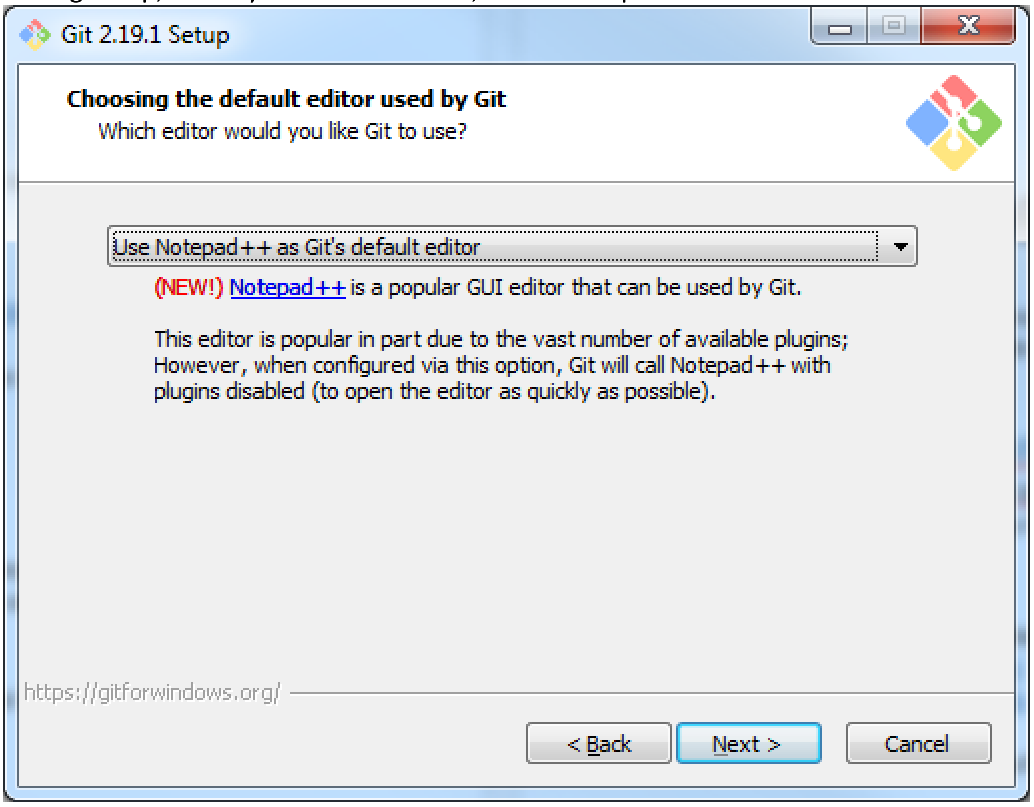
Git is required to download DEMKit. Most Linux distributions have Git installed by default as a commandline tool and can skip to 4.1.2.

4.1.1 Installing Git

WINDOWS

For Windows, we advise to install Notepad++ first from https://notepad-plus-plus.org/downloads/.
Then download and install Git. Executables are found here: https://git-scm.com/downloads .

During setup, select your default editor, select Notepad++


Git 2.19.1 Setup
Choosing the default editor used by Git
Which editor would you like Git to use?
https://gitforwindows.org/
< Back
Next >
Cancel

Use Notepad++ as Git's default editor

(NEW!) Notepad++ is a popular GUI editor that can be used by Git.

This editor is popular in part due to the vast number of available plugins; However, when configured via this option, Git will call Notepad++ with plugins disabled (to open the editor as quickly as possible).

The other default options ticked in the installation process are fine. Git does nto install bloatware, so it is safe to continue by clicking next.

LINUX

Linux users may prefer using their package manager instead.

MAC

No instructions available.

4.1.2 Generating an SSH-key

For security reasons, an SSH key is required to download the DEMKit code from the version control system. This key needs to be added on the version control system website. First, we obtain an SSH-key based on the operating system. Then, uploading this key is similar for all operating systems.

WINDOWS

- Start the tool called Git GUI
- Click "Help" and then "Show SSH Key"
- If there is no SSH Key, create one by clicking "Generate Key". It is advised not to set a password by just clicking "OK".
- Click "Copy to clipboard"

LINUX

- Open a terminal
- First check if you already have a key using this command:
  ```
  cat ~/.ssh/id_rsa.pub
  ```
  If it displays a key, then you can skip the next step.
- Generate a new key by:
  ```
  ssh-keygen -t rsa
  ```
  Follow the instructions. It is most convenient to skip the setting a passphrase part by simply pressing enter.
- After generation, copy the key by:
  ```
  cat ~/.ssh/id_rsa.pub
  ```
  Select and copy the output, which looks like:
  ```
  ssh-rsa AAAA … 4HQ1 user@systemname
  ```

MACOS

- See Linux.

Uploading the key

Assuming the key is copied correctly, go to https://github.com/settings/keys and follow these instructions:

- Click "New SSH Key"
- Enter a name/title, preferably one to identify your computer easily
  Paste the aforementioned SSH Key (e.g. `ssh-rsa AAAA … 4HQ1 user@systemname`) into the lower text box.
- Click "Add SSH key"
- You're done here

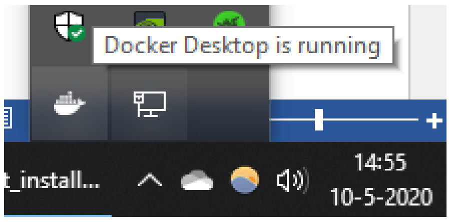
4.1.3 Installing Docker

Docker is used to setup virtual machines that run all necessary software for DEMKit. The advantage of using Docker is that the environment is equal for all users, and hence the support. Furthermore, a deployment of DEMKit using Docker can be automated, easing the process.

WINDOWS

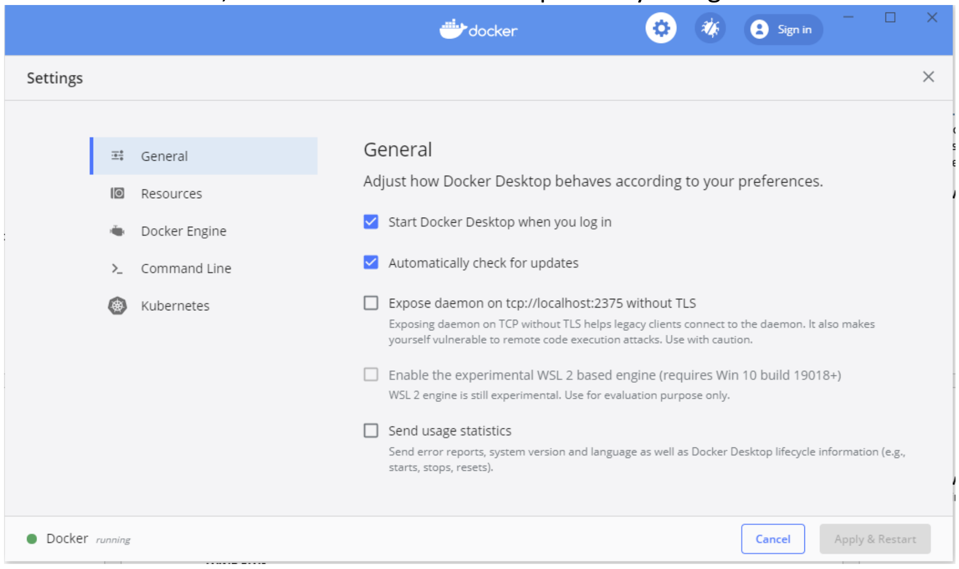
- Download and install Docker from here:
  https://hub.docker.com/editions/community/docker-ce-desktop-windows

After setup, start Docker Desktop. A ship-icon will appear in the task manager:



Right click the icon and select *settings*

In the tab **General**, Tick “Start Docker Desktop when you log in”



General

Adjust how Docker Desktop behaves according to your preferences.

- [✓] Start Docker Desktop when you log in
- [✓] Automatically check for updates
- [ ] Expose daemon on tcp://localhost:2375 without TLS
  Exposing daemon on TCP without TLS helps legacy clients connect to the daemon. It also makes yourself vulnerable to remote code execution attacks. Use with caution.
- [ ] Enable the experimental WSL 2 based engine (requires Win 10 build 19018+)
  WSL 2 engine is still experimental. Use for evaluation purpose only.
- [ ] Send usage statistics
  Send error reports, system version and language as well as Docker Desktop lifecycle information (e.g., starts, stops, resets).

Then click **Apply and Restart**

LINUX

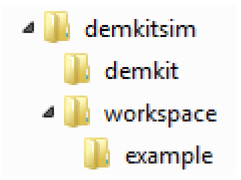
- Install docker using your distribution’s package manager and enable it to start-up by default.

MACOS

- Download and install Docker from here: https://hub.docker.com/editions/community/docker-ce-desktop-mac
- Make sure Docker starts when logging in
- Perhaps similar Settings as under Windows may be applicable

4.2 Installing DEMKit

Now we are ready to download the doe of DEMKit and an example scenario. For this setup, we assume that you have created a folder called “**demkitsim**” somewhere on your PC, e.g. in your “home” folder (Windows: C:\users\yourname\demkitsim\ or UNIX: ~/**demkitsim**/). We will refer to “demkitsim” as the folder in which we install all code and tools. At the end of this section you should end up with the folder structure depicted on the right.



Copy the **setup.bat** (or setup.sh for Linux and MacOS) to this demkitsim folder. Now we can install DEMKit.

WINDOWS

- Execute (double click) **setup.bat** and wait till it is done with executing

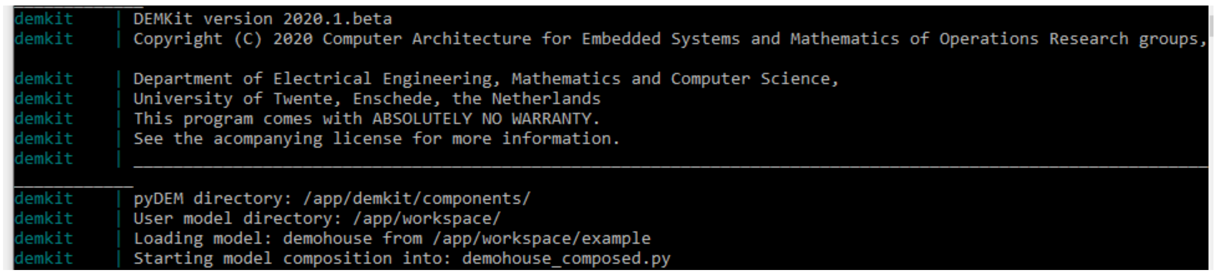
LINUX

- Open a terminal
- `cd  ~/demkitsim/`
- `chmod +x setup.sh`
- `./setup.sh`

MACOS

- See Linux

If everything goes well, a demo simulation should run at some point, e.g. you will see the following information passing by in the console:



```
demkit     | DEMKit version 2020.1.beta
demkit     | Copyright (C) 2020 Computer Architecture for Embedded Systems and Mathematics of Operations Research groups,

demkit     | Department of Electrical Engineering, Mathematics and Computer Science,
demkit     | University of Twente, Enschede, the Netherlands
demkit     | This program comes with ABSOLUTELY NO WARRANTY.
demkit     | See the acompanying license for more information.
demkit     | ________________________________________________________________________________________________________
demkit     | pyDEM directory: /app/demkit/components/
demkit     | User model directory: /app/workspace/
demkit     | Loading model: demohouse from /app/workspace/example
demkit     | Starting model composition into: demohouse_composed.py
```

If not, please seek contact (see contact details in Chapter 2)

4.3 Running a simulation

A simulation is started from a terminal. By default, an example model is provided, together with sample input data for a single house. This example can be extended by generating data for more houses (see Section 4.4) and serves as a template to create your own scenarios.

WINDOWS

First you will need to navigate to the right folder to execute a simulation

- Navigate to the demkit folder using the file explorer e.g.
  ```
  cd C:\Users\Yourname\demkitsim\demkit
  ```
- Execute (double click) **rundemkit.bat**

LINUX

First you will need to navigate to the right folder to execute a simulation

- Open up a terminal
- Navigate to the demkit folder, e.g.
  ```
  cd ~\demkitsim\demkit
  ```
- Navigate to the demkit folder.
  ```
  docker-compose up
  ```

MACOS

- See Linux.

Wait till the simulation is finished. You should now be able to visualize the results. The results of a simulation can be visualized with Grafana. Grafana can be accessed in a web browser at http://localhost:3000 . Default username is "admin", and password is "admin". You should see a "dashboard" called "**DEMKit Example**". Opening this example will show the results of the simulation just ran. For more info about visualizing the results, please refer to **Chapter 6**.

4.3.1 Running a specific model

To select a different model, the configuration needs to be changed. This can be done by modifying the configuration in **docker-compose.yaml** which can be found in the demkitsim\demkit-folder. This file can be edited using a text editor (e.g. notepad++).

From here, you can simulate a scenario using two paramers:

- **DEMKIT_MODEL:** The scenario input file, in the example *demohouse*. The model name corresponds to the filename of the scenario. Note that .py must not be included.
- **DEMKIT_FOLDER:** The folder where the above scenario is located. The path is relative to the *workspace*-folder (demkitsim/workspace/). Note that "example" is the default folder upon configuration.

4.4 Generating ALPG data

Larger scenarios can easily be generated using the open-source Artificial Load Profile Generator (ALPG) that can be obtained at https://github.com/utwente-energy/alpg . DEMKit has build-in functionality to load such a scenario. An example is given with the example "demostreet" model, which serves as a template for your own scenarios. To generate data using the ALPG and loading the demostreet simulation, follow these steps:

To make DEMKit generate and load data, follow these instructions:

WINDOWS

- Open Windows Explorer and navigate to the workspace folder, e.g.:
    - `C:\Users\Yourname\demkitsim\workspace`
- Then navigate to the model you want to add the ALPG to,
    - E.g. **example**
- Execute (double click) **setup_alpg.bat** and wait till it is done with executing

LINUX

- Open a terminal
- `cd  ~/demkitsim/workspace`
- `cd example`
- `chmod +x setup_alpg.sh`
- `./setup_alpg.sh`

MACOS

- See Linux

From here, a default setup of the ALPG is added. You will see that a folder named "**alpg**" is created. Also, you will see a file "**alpg.sh**". This default model can be modified however by changing the settings by opening and changing the file: **alpg/configs/example.py**

Refer to the ALPG documentation for more info: https://github.com/utwente-energy/alpg

To run the demostreet model with generated houses (10 is the default), you need to adapt **docker-compose.yaml** as indicated by changing **DEMKIT_MODEL=demohouse into DEMKIT_MODEL=demostreet**. Run DEMKit again using **rundemkit.py or docker-compose up** as instructed in 4.3.

Note: *After generation, the alpg.sh file in your workspace folder is renamed back to "**alpg_file.sh**". If you want to re-run ALPG data generation, you will need to rename this file to "**alpg.sh**" again. You may also edit this file to generate using different parameters. The command in this file equals the command in the ALPG documentation*

4.5 Creating own models

For your own creations, it is best to copy the example-folder and rename it. Accordingly, change all references to "example" with the name of your own scenario (i.e. modify the docker-compose.yaml file as described in 4.3.1). Change and alter the models as you wish, it is your workspace afterall! Tip, take small steps and see what happens.

From here on, Chapter 5 can be skipped and data can be visualized as instructed in Chapter 6.

5. Installing and Running DEMKit without Docker

This is a step-by-step guide to install DEMKit. The guide is mostly focussed on Windows users, but gives pointers for those running UNIX-based (Linux, MacOS) systems.

5.1 Retrieving DEMKit

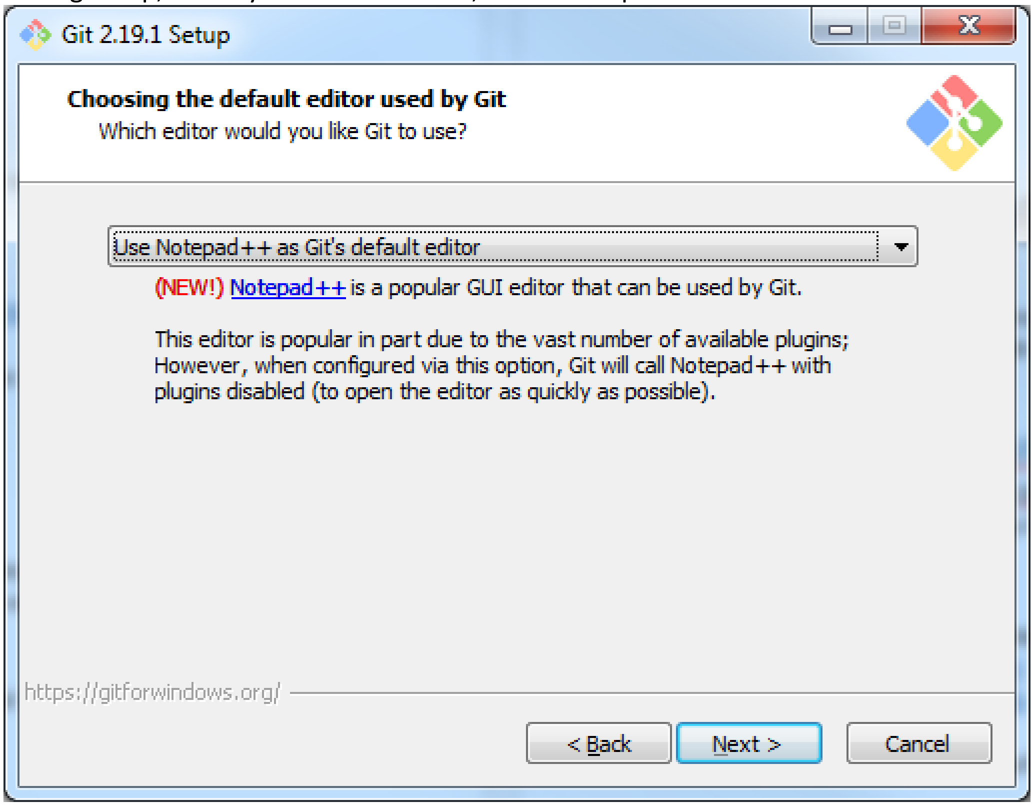
Git is required to download DEMKit. Most Linux distributions have Git installed by default as a commandline tool and can skip to 5.1.2.

5.1.1 Installing Git

WINDOWS

For Windows, we advice to install Notepad++ first from https://notepad-plus-plus.org/downloads/.
Then download and install Git. Executables are found here: https://git-scm.com/downloads .

During setup, select your default editor, select Notepad++



Choosing the default editor used by Git

Which editor would you like Git to use?

Use Notepad++ as Git's default editor

(NEW!) Notepad++ is a popular GUI editor that can be used by Git.

This editor is popular in part due to the vast number of available plugins;
However, when configured via this option, Git will call Notepad++ with
plugins disabled (to open the editor as quickly as possible).

The other default options ticked in the installation process are fine. Git does nto install bloatware, so it is safe to continue by clicking next.

LINUX

- Linux users may prefer using their package manager instead.

MAC

- No instructions available.

5.1.2 Generating an SSH-key

For security reasons, an SSH key is required to download the DEMKit code from the version control system. This key needs to be added on the version control system website. First, we obtain an SSH-key based on the operating system. Then, uploading this key is similar for all operating systems.

WINDOWS

- Start the tool called Git GUI
- Click "Help" and then "Show SSH Key"
- If there is no SSH Key, create one by clicking "Generate Key". It is advised not to set a password by just clicking "OK".
- Click "Copy to clipboard"
- Keep Git GUI open, but close the "Your OpenSSH Public Key" screen.

LINUX

- Open a terminal
- First check if you already have a key using this command:
  ```
  cat ~/.ssh/id_rsa.pub
  ```
  If it displays a key, then you can skip the next step.
- Generate a new key by:
  ```
  ssh-keygen -t rsa
  ```
  Follow the instructions. It is most convenient to skip the setting a passphrase part by simply pressing enter.
- After generation, copy the key by:
  ```
  cat ~/.ssh/id_rsa.pub
  ```
  Select and copy the output, which looks like:
  ```
  ssh-rsa AAAA … 4HQ1 user@systemname
  ```

MACOS

- See Linux.

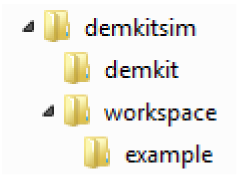
Uploading the key

Assuming the key is copied correctly, go to https://github.com/settings/keys and follow these instructions:

- Click "New SSH Key"
- Enter a name/title, preferably one to identify your computer easily
  Paste the aforementioned SSH Key (e.g. `ssh-rsa AAAA … 4HQ1 user@systemname`) into the lower text box.
- Click "Add SSH key"
- You're done here

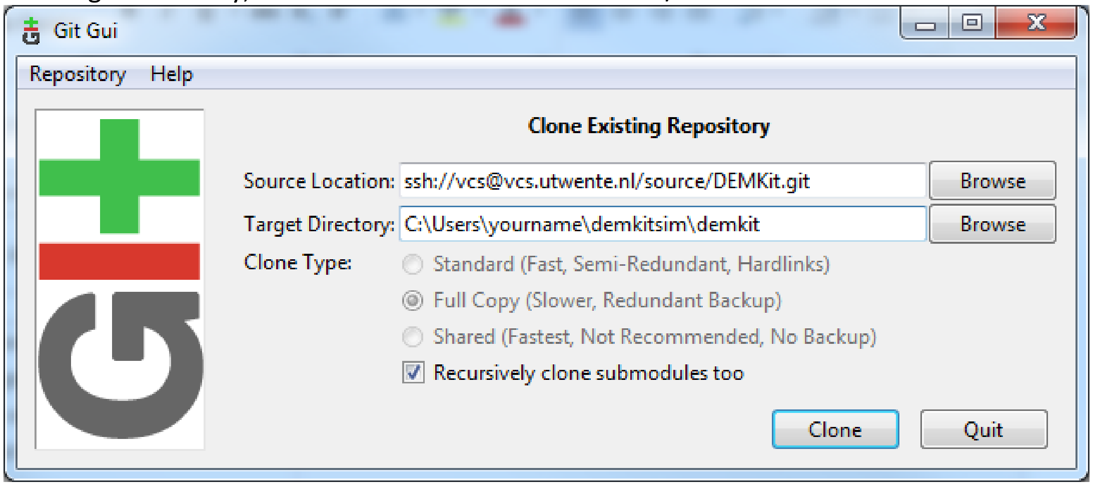
5.1.3 Pulling DEMKit

Now we are ready to download the doe of DEMKit and an example scenario. For this setup, we assume that you have created a folder called "**demkitsim**" somewhere on your PC, e.g. in your "home" folder (Windows: C:\users\yourname\demkitsim\ or UNIX: ~/**demkitsim**/). We will refer to "demkitsim" as the folder in which we install all code and tools. At the end of this section you should end up with the folder structure depicted on the right.



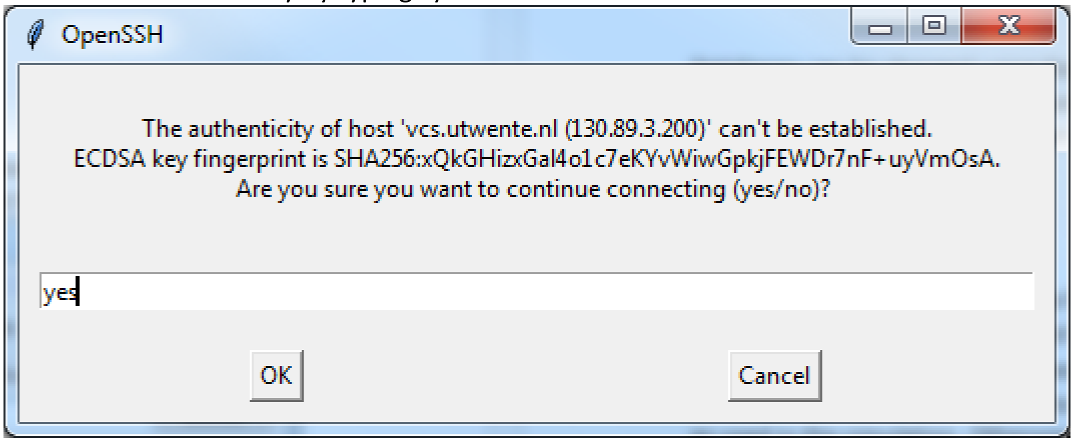
WINDOWS

- First, let us download the simulator software
- Click on “Clone existing repository”
- Use the following Source Location: ssh://git@github.com:utwente-energy/demkit.git
  For Target Directory, use the created folder “demkitsim/demkit”.



Source Location: ssh://vcs@vcs.utwente.nl/source/DEMKit.git
Target Directory: C:\Users\yourname\demkitsim\demkit
Clone Type: Standard (Fast, Semi-Redundant, Hardlinks) [✓ Full Copy (Slower, Redundant Backup)] Shared (Fastest, Not Recommended, No Backup)
[✓ Recursively clone submodules too]

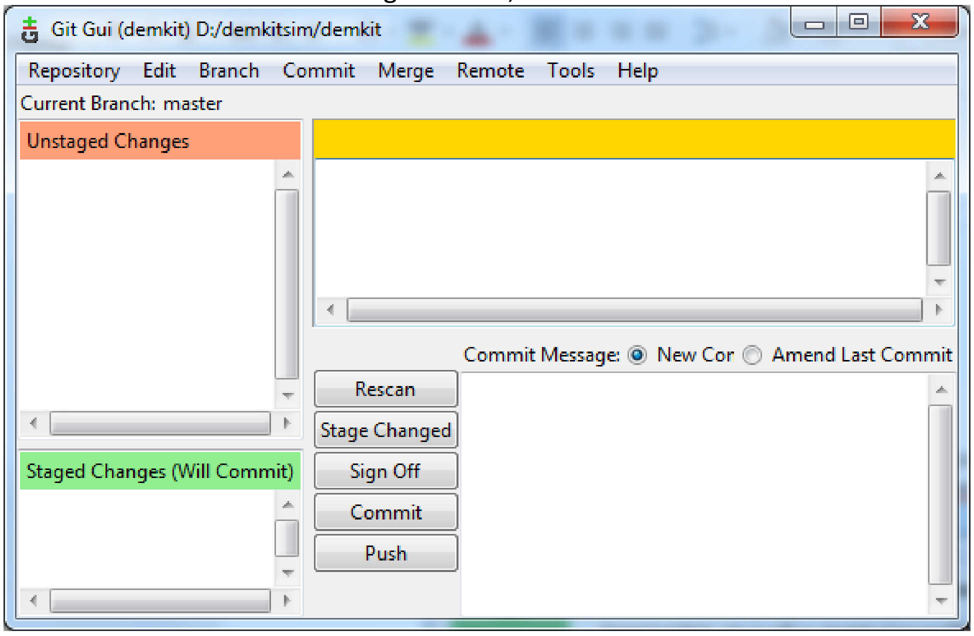
- Click “Clone”
- Confirm the authenticity by typing “yes” and click ok



The authenticity of host 'vcs.utwente.nl (130.89.3.200)' can't be established.
ECDSA key fingerprint is SHA256:xQkGHizxGal4o1c7eKYvWiwGpkjFEWDr7nF+uyVmOsA.
Are you sure you want to continue connecting (yes/no)?

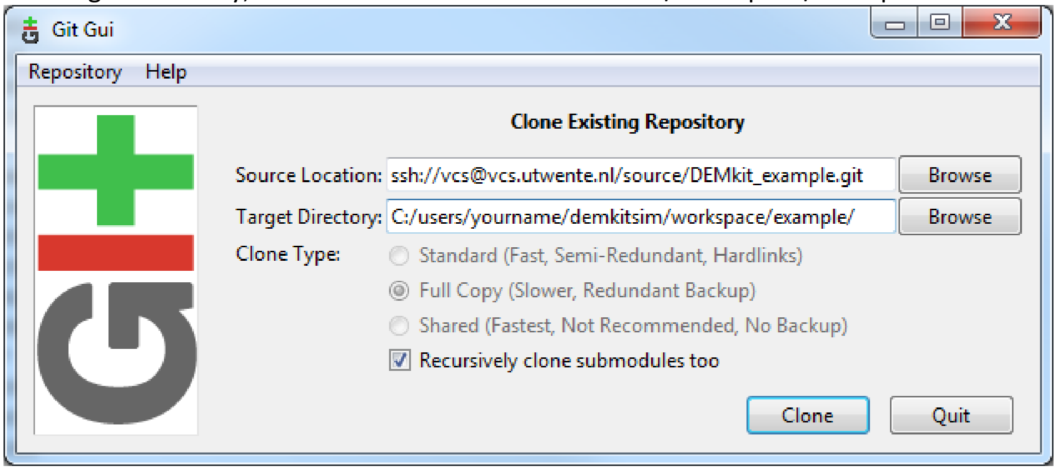
yes

- Wait till Git finishes downloading the code, then it is safe to close


Git Gui (demkit) D:/demkitsim/demkit
Repository Edit Branch Commit Merge Remote Tools Help
Current Branch: master
Unstaged Changes
Staged Changes (Will Commit)
Commit Message: New Cor Amend Last Commit
Rescan
Stage Changed
Sign Off
Commit
Push

- To download the example: open another Git GUI windows

- Click on “Clone existing repository”
- Use as Source Location: ssh:// git@github.com:utwente-energy/demkit-example.git
  For Target Directory, use the created folder “demkitsim/workspace/example”.



Source Location: ssh://vcs@vcs.utwente.nl/source/DEMkit_example.git
Target Directory: C:/users/yourname/demkitsim/workspace/example/
Clone Type: Standard (Fast, Semi-Redundant, Hardlinks) [✓ Full Copy (Slower, Redundant Backup)] Shared (Fastest, Not Recommended, No Backup)
[✓ Recursively clone submodules too]

- Click “Clone” and again, wait till Git completes cloning

LINUX

- Open a terminal
- Navigate to the created “demkitsim” folder, e.g.:
```
cd ~/demkitsim
```
- Now clone the demkit code
```
git clone ssh://git@github.com:utwente-energy/demkit.git
```
- Now clone the demkit example
```
mkdir workspace
cd workspace
git clone ssh://git@github.com:utwente-energy/demkit-example.git
cd ..
```

MACOS

See Linux.

5.1.4 Editing the DEMKit config

After downloading the DEMKit software, it is required to configure some environment variables in the DEMKit config:

- Open the file “demkitsim/demkit/conf/usrconf.py.misc” with a text editor, e.g. notepad++
- Change the following settings to reflect your setup. The componentPath and workspacePath are essential. Based on the example, the following values should be used:

```
demCfg['env']['path'] = 'C:/users/yourname/demkitsim/demkit/components/’
demCfg['workspace']['path'] = ‘C:/users/yourname/demkitsim/workspace/'
```

  The other values can be kept as they are. ***Make sure you end with a trailing slash***
  For Windows users, do use “/” instead of “\”, or use an escaped backslash “\\”
- Save the file as “**usrconf.py**”! ***Make sure the “.misc” is removed!***

5.2 Installing Python 3 and packages

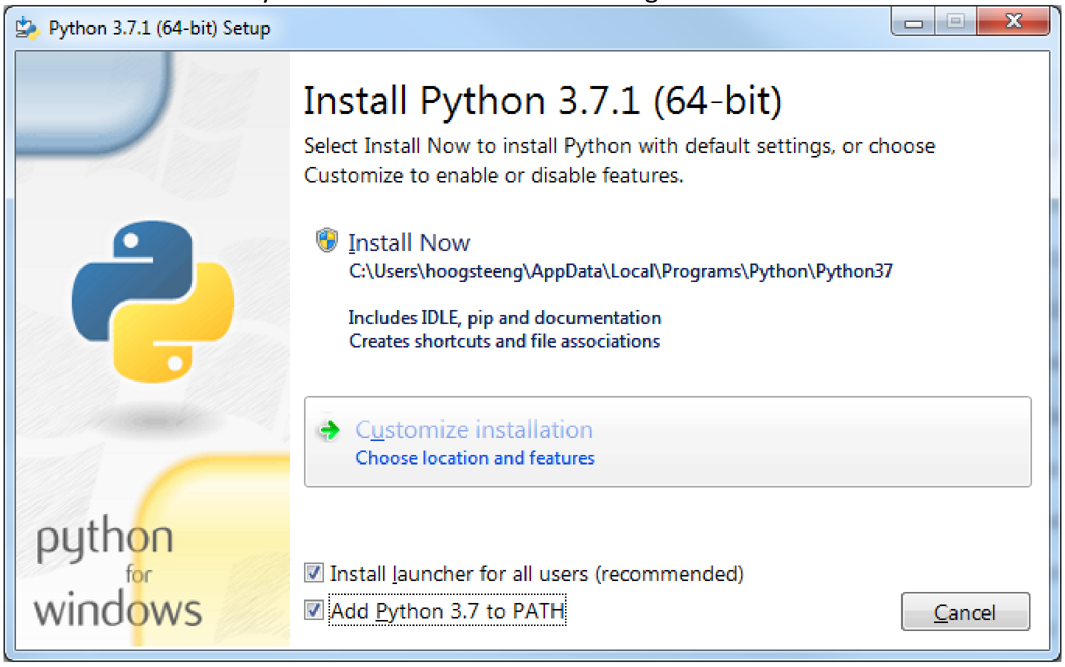
DEMKit is written in the Python 3 language, hence Pytohn 3 I required, together with some packages. This section first explains how to install Python 3 and then the required packages.

5.2.1 Installing Python

If you already have Python installed, make sure it is version 3!

WINDOWS

- Download the Python 3 installer from: https://www.python.org/downloads/release/
- Open the installer.
- Check the box "Add Python 3.x to PATH" as in the image below



Python 3.7.1 (64-bit) Setup

Install Python 3.7.1 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\hoogsteeng\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

[✓] Install launcher for all users (recommended)
[✓] Add Python 3.7 to PATH

- Then select "Install Now"
- Wait till the installer finishes.

LINUX

Usually, Python 3 is already installed. You can check this by opening a terminal and type "python" or "python3". In some distributions, "python" may start version 2.x, whereas "python3" starts python 3.x. If Python 3 is not yet installed, check your package manager.

MACOS

- Install the Command Line Tools for Xcode (required for python) by typing:
  ```
  xcode-select --install
  ```
  in a Terminal window.
- Install Homebrew from https://brew.sh. Homebrew is a package manager for Mac OS X, just as you might have for a Linux system.
- Install python3 using Homebrew, type:
  ```
  brew install python3
  ```
  in a Terminal window.

5.2.2 Installing Python packages

After installing Python, additional libraries are required.

WINDOWS

- Open a commandline as admininstrator:
  - Start, search for “cmd.exe”
  - Right-click cmd.exe, choose “run as administrator”
- For the required packages, type:
  `pip install astral==1.10.1 numpy requests pytz zmq`
- For additional (not required) packages, type:
  `pip install eve pyserial`
- Close the console

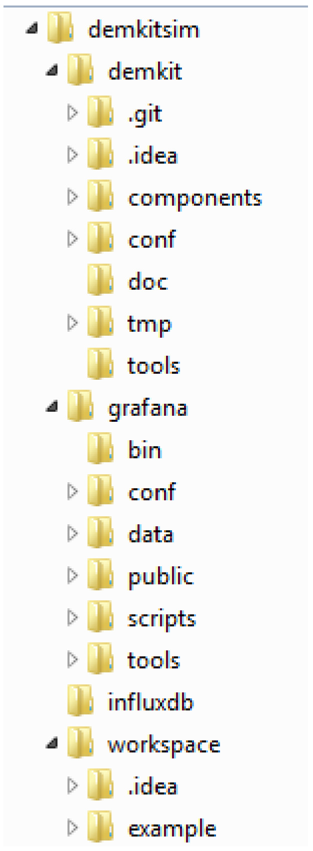
Note, optionally you may need to install a part of the Visual Studio software. Check the errors that the terminal provides

LINUX

- Open a terminal:
- For the required packages, type:
  `sudo pip3 install astral==1.10.1 numpy requests pytz zmq`
- For additional (not required) packages, type:
  `sudo pip3 install eve pyserial`
- Close the console

Note, pip3 may not exist on your distribution. Instead, try to replace “pip3” by “pip”, “python3 –m pip” or “python –m pip”. Make sure the packages are installed for Python 3.x. If no installation of pip exists, please install pip through your package manager.

MACOS
See Linux.



```
demkitsim
  demkit
    .git
    .idea
    components
    conf
    doc
    tmp
    tools
  grafana
    bin
    conf
    data
    public
    scripts
    tools
  influxdb
  workspace
    .idea
    example
```

5.3 Installing InfluxDB and Grafana

DEMKit uses an external database (InfluxDB) and web-based data viewer (Grafana) to store and present results of simulations. The following part will explain the full manual setup of these tools. However, a hybrid version is also possible in which Grafana and InfluxDB are running in docker.

5.3.1 Installing InfluxDB and Grafana through Docker

In this case we assume that Docker is already installed and up and running. If not, refer to Chapter 4 for details.

WINDOWS

First you will need to navigate to the right folder to execute a simulation

- Open up a terminal (Start, “cmd.exe”)
- Navigate to the demkitsim folder, e.g.
  `cd C:\Users\Yourname\demkitsim`
- Navigate to the demkit\services folder.
  `cd demkit\services`
- Execute the following command to load the docker compose file.
  `docker-compose up -d`

LINUX

First you will need to navigate to the right folder to execute a simulation

- Open up a terminal
- Navigate to the demkit\services folder.
  `cd demkit\services`
- Execute the following command to load the docker compose file.
  `docker-compose up -d`

MACOS
See Linux.

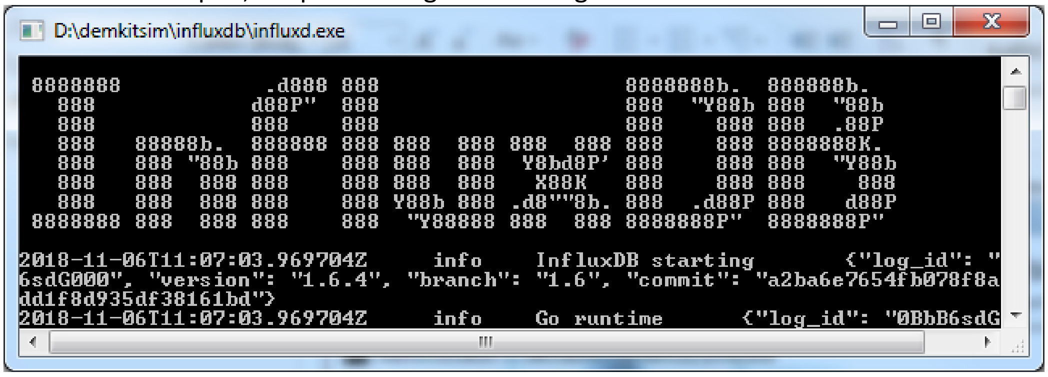
Sections 5.3.2 and 5.3.3 can be skipped.

5.3.2 Installing InfluxDB

For Windows, the final folder structure should look like the image on the right.

WINDOWS

- Download InfluxDB from: https://portal.influxdata.com/downloads
- Extract the contents to "demkitsim"
- Rename the influxdb-x.x.x folder to simply "influx"
- Navigate into the "influx" folder
- Start influxd.exe
  Click "Run" if a warning appears
  A terminal will open, keep it running in the background



```
 8888888           .d888 888                   8888888b.  888888b.
   888            d88P"  888                   888  "Y88b 888  "88b
   888            888    888                   888    888 888  .88P
   888   88888b.  888888 888 888  888 888  888 888    888 8888888K.
   888   888 "88b 888    888 888  888  Y8bd8P' 888    888 888  "Y88b
   888   888  888 888    888 888  888   X88K   888    888 888    888
   888   888  888 888    888 Y88b 888 .d8""8b. 888  .d88P 888   d88P
 8888888 888  888 888    888  "Y88888 888  888 8888888P"  8888888P"

2018-11-06T11:07:03.969704Z     info    InfluxDB starting       {"log_id": "
6sdG000", "version": "1.6.4", "branch": "1.6", "commit": "a2ba6e7654fb078f8a
dd1f8d935df38161bd"}
2018-11-06T11:07:03.969704Z     info    Go runtime      {"log_id": "0BbB6sdG
```

LINUX

- Install InfluxDB from your package manager or refer to instructions on https://portal.influxdata.com/downloads
- It is recommended to start InfluxDB on boot, e.g. through systemctl
- Alternatively, it can be started from the commandline:
  `influxd`

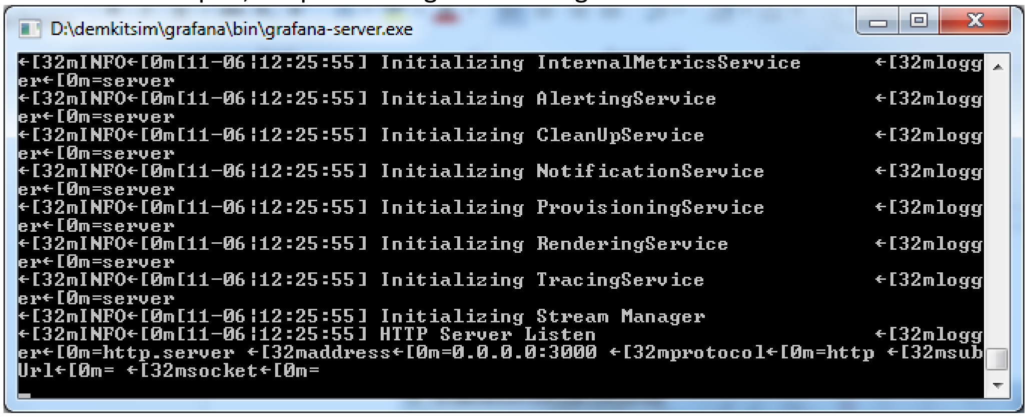
MACOS

- Install InfluxDB from the terminal using Brew:
  ```
  brew install influxdb
  brew services start influxdb
  ```

5.3.3 Installing Grafana

WINDOWS

- Download Grafana from: https://grafana.com/grafana/download
- Extract the contents to “demkitsim”
- Rename the grafana-x.x.x folder to simply “grafana”
- Navigate into the “grafana” folder, and subsequently into the “bin” folder
- Start grafana-server.exe
  Click “Run” if a warning appears
  A terminal will open, keep it running in the background



```
←[32mINFO←[0m[11-06|12:25:55] Initializing InternalMetricsService       ←[32mlogg
er←[0m=server
←[32mINFO←[0m[11-06|12:25:55] Initializing AlertingService              ←[32mlogg
er←[0m=server
←[32mINFO←[0m[11-06|12:25:55] Initializing CleanUpService               ←[32mlogg
er←[0m=server
←[32mINFO←[0m[11-06|12:25:55] Initializing NotificationService          ←[32mlogg
er←[0m=server
←[32mINFO←[0m[11-06|12:25:55] Initializing ProvisioningService          ←[32mlogg
er←[0m=server
←[32mINFO←[0m[11-06|12:25:55] Initializing RenderingService             ←[32mlogg
er←[0m=server
←[32mINFO←[0m[11-06|12:25:55] Initializing TracingService               ←[32mlogg
er←[0m=server
←[32mINFO←[0m[11-06|12:25:55] Initializing Stream Manager
←[32mINFO←[0m[11-06|12:25:55] HTTP Server Listen                        ←[32mlogg
er←[0m=http.server ←[32maddress←[0m=0.0.0.0:3000 ←[32mprotocol←[0m=http ←[32msub
Url←[0m= ←[32msocket←[0m=
_
```

LINUX

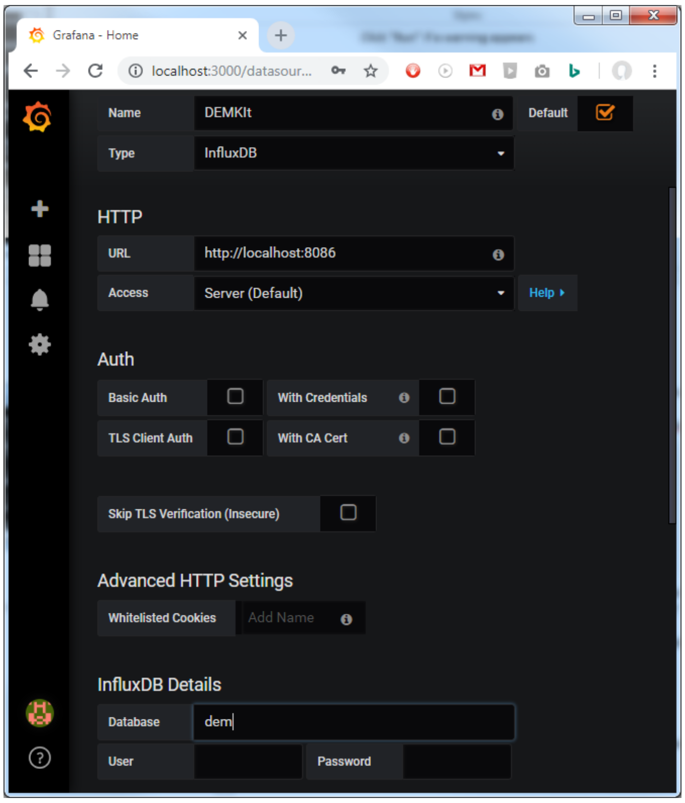
- Install grafana from your package manager or refer to instructions on https://grafana.com/grafana/download
- It is recommended to start Grafana on boot, e.g. through systemctl
- Alternatively, it can be started from the commandline:
  `grafana-server`

MACOS

- Install Grafana from the terminal using Brew:
  ```
  brew install grafana
  brew services start grafana
  ```

5.3.4 Setting up Grafana

- Open a browser and browse to the Grafana page located at http://localhost:3000
- Log in. Default username is “admin” and password “admin”.
- Set a new password if desired or skip.
- Go to the settings by hovering over the cogwheel on the left and clicking “Data Sources”
- Then select “Add Datasource”
- Fill in the following details:
    - Name: “DEMKit”
    - Default: Yes (tick)
    - Type: InfluxDB
    - URL:
        - Full manual setup: http://localhost:8086 (if it is grey, you still need to fill it)
        - Hybrid docker setup: http://demkit_influxdb:8086
    - Database: “dem”
    - The rest can be kept blank
- Click “Save and Test”
- Most likely, you will get an error as the database does not exist. This is not a problem as it will be created during the first simulation. If you have problems later, return to these steps and check if the source is working.



Name	DEMKIt	Default	✓
Type	InfluxDB		

HTTP

URL	http://localhost:8086
Access	Server (Default)

Auth

Basic Auth	☐	With Credentials	☐
TLS Client Auth	☐	With CA Cert	☐
Skip TLS Verification (Insecure)	☐		

Advanced HTTP Settings

Whitelisted Cookies	Add Name

InfluxDB Details

Database	dem		
User		Password	

5.4 Running a simulation

A simulation is started from a terminal. By default, an example model is provided, together with sample input data for a single house. This example can be extended by generating data for more houses (see Section 5.5) and serves as a template to create your own scenarios.

WINDOWS

First you will need to navigate to the right folder to execute a simulation

- Open up a terminal (Start, “cmd.exe”)
- Navigate to the demkitsim folder, e.g.
  `cd C:\Users\Yourname\demkitsim`
- Navigate to the demkit folder.
  `cd demkit`

LINUX

First you will need to navigate to the right folder to execute a simulation

- Open up a terminal
- Navigate to the demkitsim folder, e.g.
  `cd ~\demkitsim`
- Navigate to the demkit folder.
  `cd demkit`

MACOS

See Linux.

From here, you can simulate a scenario using two paramers:

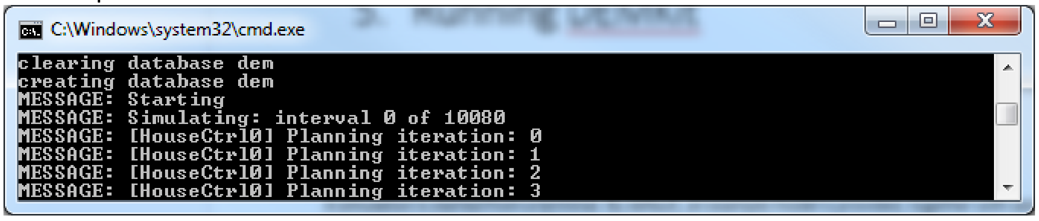
- **Model (`-m`):** The scenario input file, in the example *demohouse*. The model name corresponds to the filename of the scenario. Note that .py must not be included.
- **Folder (`-f`):** The folder where the above scenario is located. The path is relative to the configured path *workspacePath* in the usrconf.py file as described in Section 5.1.4.

To run the example scenario, issue the following command in the terminal (all operating systems):

```
python demkit.py –f example –m demohouse
```

The command above will load `demkitsim/workspace/`**`example`**`/`**`demohouse`**`.py`. Note that on some operating systems *python* should be replaced by *python3* to use Python 3.x.

The output should look like this:



```
clearing database dem
creating database dem
MESSAGE: Starting
MESSAGE: Simulating: interval 0 of 10080
MESSAGE: [HouseCtrl0] Planning iteration: 0
MESSAGE: [HouseCtrl0] Planning iteration: 1
MESSAGE: [HouseCtrl0] Planning iteration: 2
MESSAGE: [HouseCtrl0] Planning iteration: 3
```

5.5 Generating ALPG data

Larger scenarios can easily be generated using the open-source Artificial Load Profile Generator (ALPG) that can be obtained at https://github.com/utwente-energy/alpg . DEMKit has build-in functionality to load such a scenario. An example is given with the example “demostreet” model, which serves as a template for your own scenarios. To generate data using the ALPG and loading the demostreet simulation, follow these steps:

WINDOWS
First you will need to navigate to the right folder to execute a simulation

- Open up a terminal (Start, “cmd.exe”)
- Navigate to the demkitsim folder, e.g.
  ```
  cd C:\Users\Yourname\demkitsim
  ```
- Navigate to the demkit example folder.
  ```
  cd workspace\example
  ```

LINUX
First you will need to navigate to the right folder to execute a simulation

- Open up a terminal
- Navigate to the demkitsim folder, e.g.
  ```
  cd ~\demkitsim
  ```
- Navigate to the demkit example folder.
  ```
  cd workspace\example
  ```

MACOS
See Linux.

Now you can pull the ALPG and generate data. For all operating systems you can enter in the terminal:

```
git clone https://github.com/utwente-energy/alpg.git
cd alpg
python profilegenerator.py -c example -o demo –force
```

Now, and this takes a while, the data should be generated. When the ALPG is done, you can run a simulation. First navigate back to the demkit-folder:

```
cd ../../../demkit
```

Then simulate the demostreet

```
python demkit.py –f example –m demostreet
```

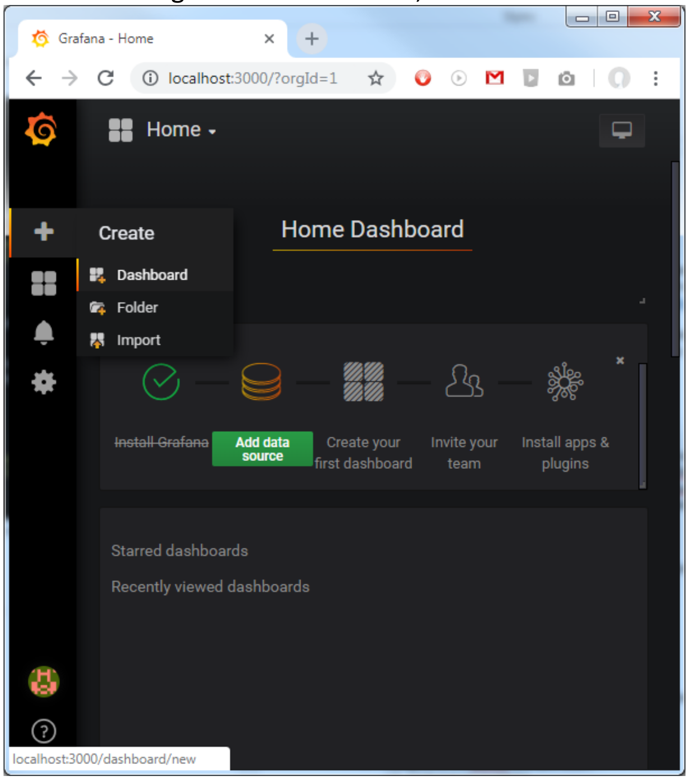
For your own creations, it is best to copy the example-folder and rename it. Accordingly, change all references to “example” with the name of your own scenario. Change and alter the models as you wish.

6. Visualizing data with Grafana

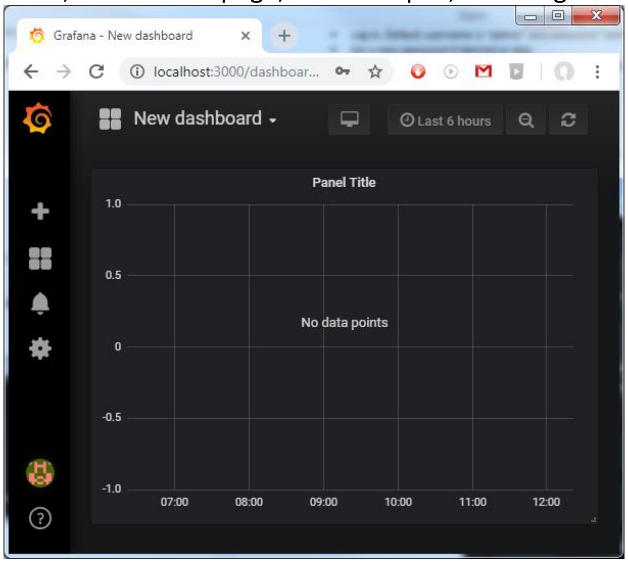
The results of a simulation can be visualized with Grafana. Querying data from the InfluxDB is made simple with this web-based tool. However, there are some tricks and pitfalls that you need to be aware with. Grafana can be accessed in a web browser at http://localhost:3000 . Default username is “admin”, and password is “admin”.

Data is visualized with a dashboard, which can be saved. A dashboard can contain multiple graphs as well. For the first time, a new dashboard needs to be set-up (except for a Docker install which comes with one preinstalled):

- Click on the big “+” on the left side, and choose “Create Dashboard”



- Then, on the next page, click “Graph”, resulting in an empty graph:



- Right click on “Panel Title” and choose “Edit”
- Now you can add queries at the bottom of this page to create plots.

QUERYING
Grafana is very flexible in what data you want to visualize and how. It is therefore important to carefully construct the query to only show the data you want to see. First we explain what can be plotted, followed by the tips and tricks on HOW it should be plotted.

A query starts with selecting the data in the "FROM default <measurement>" line
DEMKit writes the results of different components to different "measurements":

- devices (all device stats)
- controllers (control and optimization outcomes)
- flow (for load flow analysis of e.g. electricity networks)
- host (overall stats of the simulation)

Normally, this will result in data from all devices, controllers, flows or hosts. Usually, you want to target a specific type of device. This can be defined through filtering in the "WHERE"-block. Each component has a type (devtype for device type) to select all devices of the same type and a unique name to select a single component.

Regarding the types, we use abbreviations of the components described in literature:

- **meter**: MeterDevice, metering devices such as smart meters.
- **unc:** UncontrollableDevice, e.g. a base load
- **curt:** CurtailableDevice, static load/generation that can be curtailed / shed
- **buf**: BufferDevice, buffers such as electrical energy storage or heat tanks
- **conv:** Converter device, such as an heat pump
- **bufConv:** Combination of BufferDevice and ConverterDevice
- **ts:** TimeShiftableDevice, e.g. washing machines or dishwashers.
- **bts:** BufferTimeShiftable, e.g. an electric vehicle (buffer device with time limitations)

Lastly, one can select what field (value) to plot in SELECT. A long list is provided, but not all components will have data for the chosen field. The fields follow this naming convention:
Unit-type[.optional.number.of.specifiers][.c.commodity][.complex-part]

Examples:
Wh-energy-consumption.c.power
W-power.imported.c.power.real
W-power.imported.c.power.imag
C-temperature.indoor

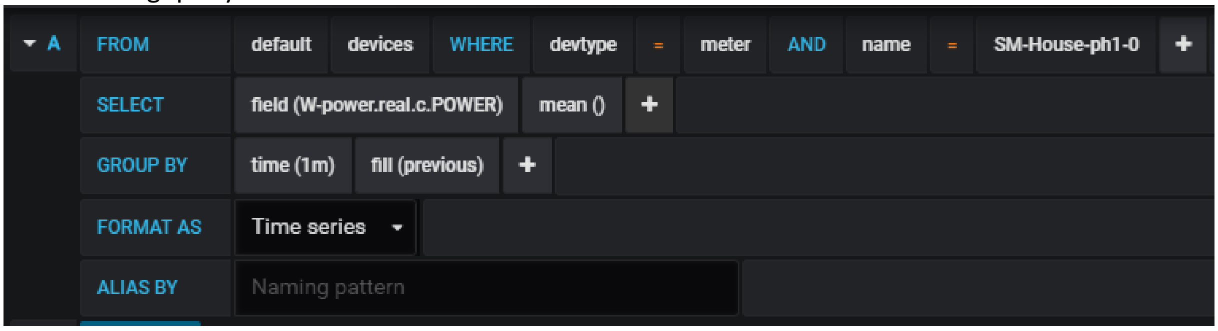
TIPS AND TRICKS

- By default a new plot only show the "Last 6 Hours" (see top right). DEMKit stores the data from the date specified in the simulation model, which by default is 1 Januari 2018. The fastest method is to display "This year" in the drop-down menu when clicking on "Last 6 hours". Dragging can be used to zoom.
- The data plotted in Grafana is often resampled through the "GROUP BY" settings. The time herein is variable by default ("$__interval"), which may result in unexpected outcomes. To have full control over how the data is plotted, preferably set the time grouping to 1m (m for minute) or the time interval configured in the model.
- Note that the aggregation type in SELECT (by default "mean") also influences how data is shown. Grouping by e.g. 1 hour will result in a graph that shows the average value for all samples, 60 in this case. Even more importantly, when plotting the data of multiple

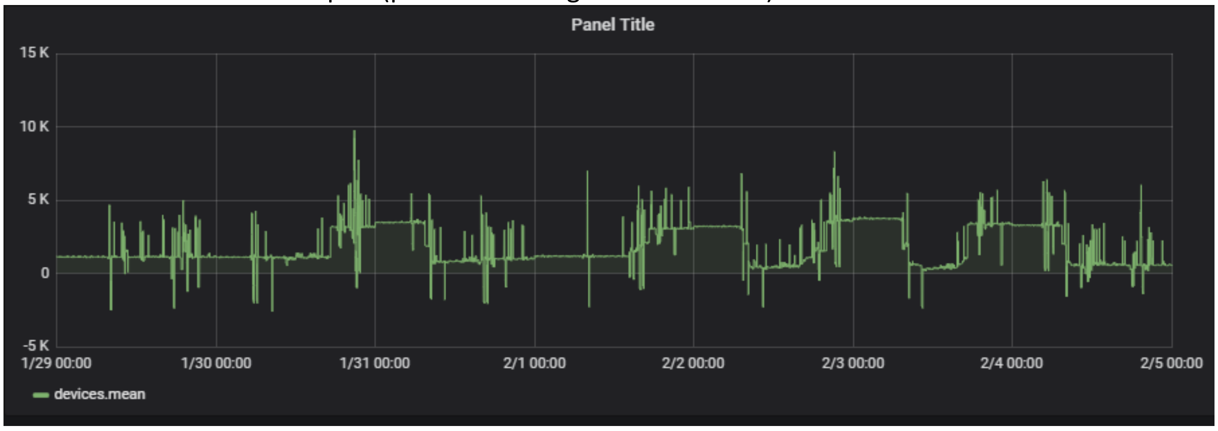
components (e.g. all meters, say 100 in total), you would get the average value of all samples again, so in this case 100 components times 60 minutes = the average of 6000 values.

- Selecting a small aggregation time, but plotting a lot of data may sometimes result in no data to be shown at all at first sight. This is often due to the fact that not enough samples exist, resulting in “null” to be displayed. It is recommended to set, in “GROUP BY” the “fill” to “previous”
- Data can be aggregated when desired. E.g. to see the total, summed, power consumption of all meters, it better to select all meters in the FROM-line, and then select the “sum” “aggregation” in the “SELECT” line.
- ALIAS BY can be used to set a legend
- CSV files (e.g. for plotting in Excel or PGFPlots for Latex) can be downloaded by right-clicking the panel title and then choosing “More”, then “Export CSV”.

As an example, to plot the total energy consumption (as measured by the meter) of the demo house, the following query is the result:


```
A  FROM       default  devices  WHERE  devtype = meter  AND  name = SM-House-ph1-0  +
   SELECT     field (W-power.real.c.POWER)  mean ()  +
   GROUP BY   time (1m)  fill (previous)  +
   FORMAT AS  Time series
   ALIAS BY   Naming pattern
```

Which should result in this plot (provided the right dates are set)



For more information, we refer to the Grafana documentation: http://docs.grafana.org/

7. Setting up tools

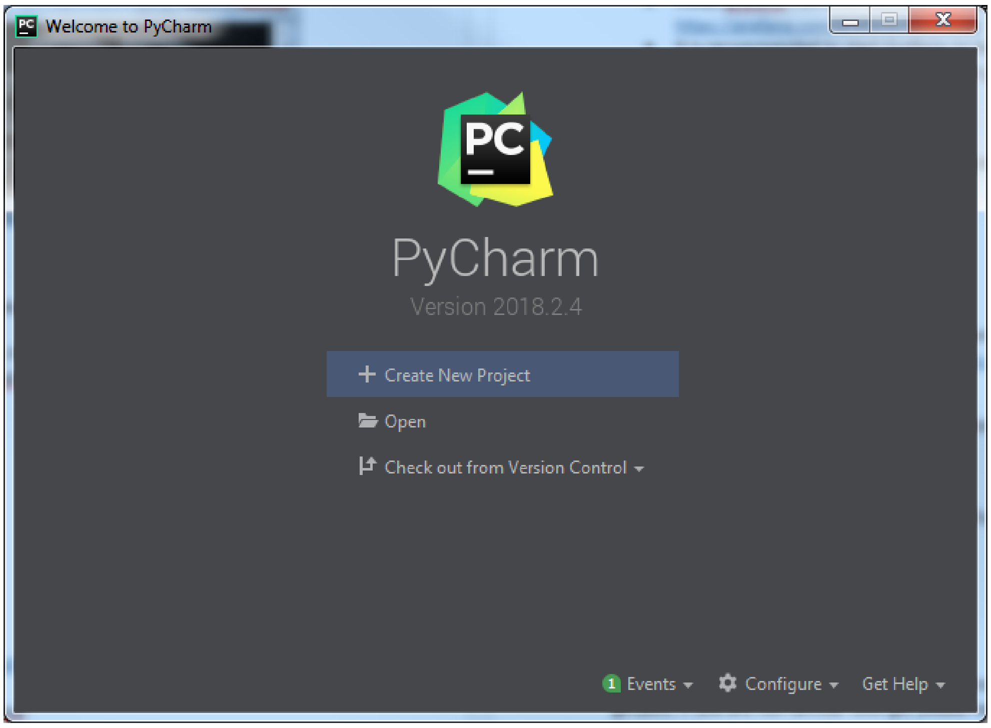
7.1 Setting up PyCharm

We recommend PyCharm as IDE for development. Installation of PyCharm should be trivial. PyCharm can be downloaded from https://www.jetbrains.com/pycharm/download/ (Note: there is an academic license for PyCharm Professional).

The remainder of this Section details how to import the code, change some settings and how to update.

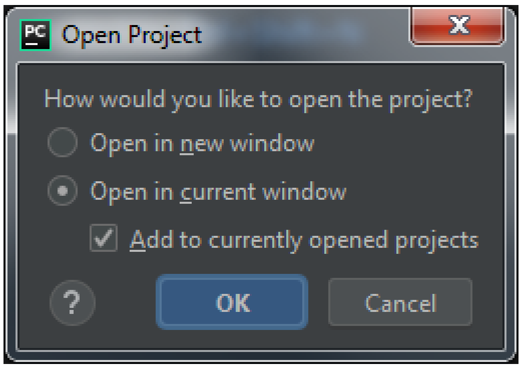
7.1.1 Loading DEMKit as a project

After installing and setting up PyCharm for the first time (e.g. preferences and license), a Welcome screen is greeting you:



- In this screen, select Open.
- Then select the “demkitsim\demkit\”-folder
- Wait a while till it completes loading
- On the left pane, you should see the “DEMKit project, with all the source files if you collapse it.
- The next step is to also open the “workspace” directory for your scenarios
- Click “File”, then “Open”
- Select the workspace directory “demkitsim\workspace\”

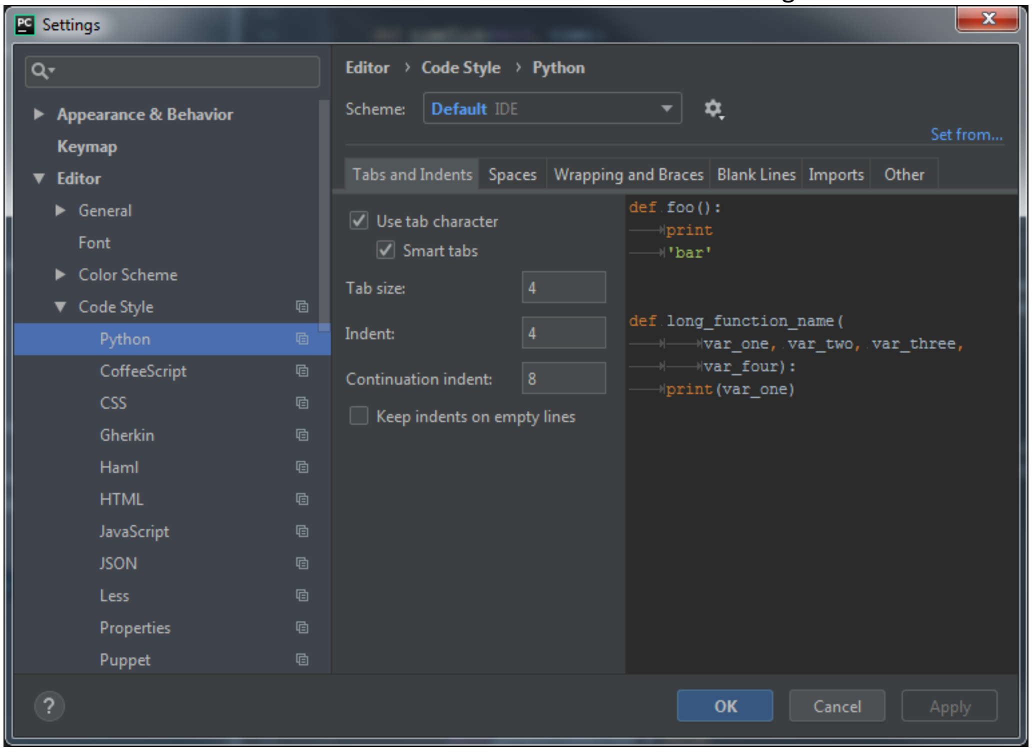
- If prompted how to open the project, choose “Open in current window” and also select “Add to currently opened projects”



How would you like to open the project?
( ) Open in new window
(●) Open in current window
[✓] Add to currently opened projects

By default, PyCharm uses Python coding style guidelines, which DEMKit does not completely follow. To ease life and remove ignoring warning, the default PyCharm settings need to be adapted.

- Open “File” and then “Settings”
- Go to “Editor”, then “Coding Style”, then “Python” and select the “Tabs and Indents” tab
- Now tick both “Use tab character” and “Smart tabs” as in the image below



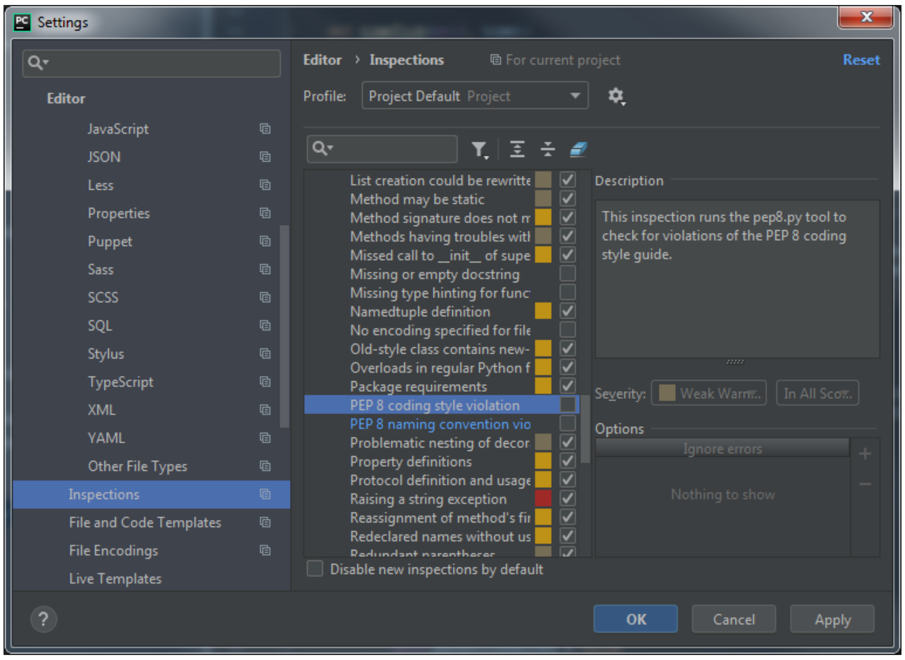
Scheme: Default IDE

[✓] Use tab character
[✓] Smart tabs
Tab size: 4
Indent: 4
Continuation indent: 8
[ ] Keep indents on empty lines

```
def foo():
	print
	'bar'

def long_function_name(
		var_one, var_two, var_three,
		var_four):
	print(var_one)
```

- Then go to “Editor”, then “Coding Style”, then “Inspections”.
- Select “Python”
- Now look for the following inspections and disable them:
  - PEP8 Coding style violation
  - PEP8 naming convention violation


Settings

Editor › Inspections For current project Reset

Profile: Project Default Project

List creation could be rewritte ✓
Method may be static ✓
Method signature does not m ✓
Methods having troubles wit ✓
Missed call to __init__ of supe ✓
Missing or empty docstring
Missing type hinting for func
Namedtuple definition ✓
No encoding specified for file
Old-style class contains new- ✓
Overloads in regular Python f ✓
Package requirements ✓
PEP 8 coding style violation
PEP 8 naming convention vio
Problematic nesting of decor ✓
Property definitions ✓
Protocol definition and usage ✓
Raising a string exception ✓
Reassignment of method's fir ✓
Redeclared names without us ✓
Redundant parentheses ✓

Disable new inspections by default

Description: This inspection runs the pep8.py tool to check for violations of the PEP 8 coding style guide.

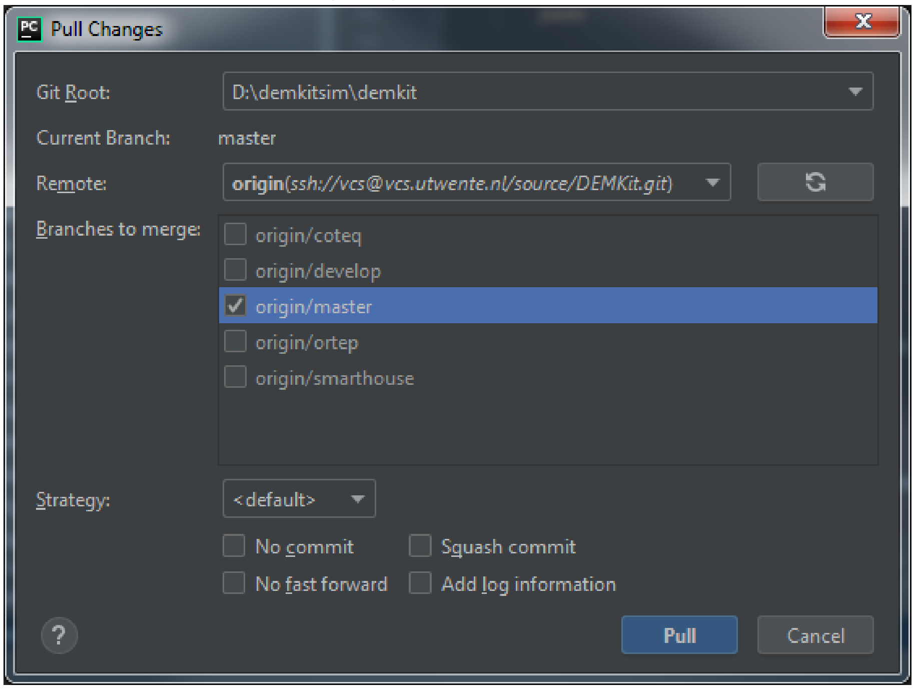
Severity: Weak Warni.. In All Scop..

Options: Ignore errors — Nothing to show

- Click “Apply” and then “OK”

The last trick with PyCharm is to easily update DEMKit and the example through Git. We only focus on getting updates, “Pulling” in Git, advanced users can also push and branch from Pycharm. Updating is done as follows:

- Right-click on the DEMKit project folder in PyCharm
- Then select “Git” , then “Repository” and then “Pull” (See picture on the nextpage)
- Then select your branch, e.g. origin/master and click “Pull”:


Pull Changes

Git Root: D:\demkitsim\demkit

Current Branch: master

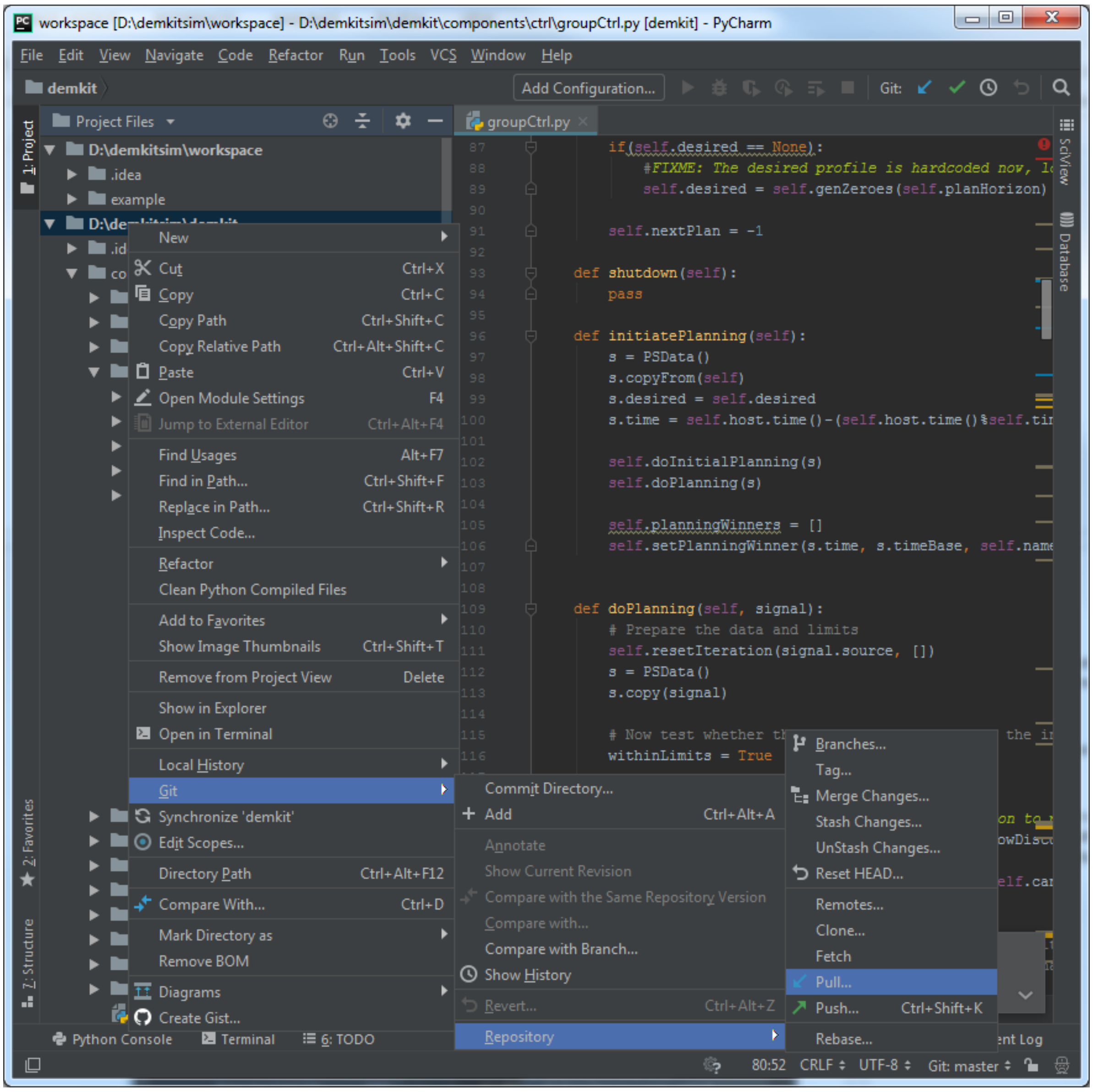
Remote: **origin**(*ssh://vcs@vcs.utwente.nl/source/DEMKit.git*)

Branches to merge:
- [ ] origin/coteq
- [ ] origin/develop
- [x] origin/master
- [ ] origin/ortep
- [ ] origin/smarthouse

Strategy: <default>

- [ ] No commit
- [ ] Squash commit
- [ ] No fast forward
- [ ] Add log information

- In case of errors, make sure to revert your changes first (read the docs on how to do this).



```
        if(self.desired == None):
            #FIXME: The desired profile is hardcoded now, l
            self.desired = self.genZeroes(self.planHorizon)

        self.nextPlan = -1

    def shutdown(self):
        pass

    def initiatePlanning(self):
        s = PSData()
        s.copyFrom(self)
        s.desired = self.desired
        s.time = self.host.time()-(self.host.time()%self.ti

        self.doInitialPlanning(s)
        self.doPlanning(s)

        self.planningWinners = []
        self.setPlanningWinner(s.time, s.timeBase, self.name

    def doPlanning(self, signal):
        # Prepare the data and limits
        self.resetIteration(signal.source, [])
        s = PSData()
        s.copy(signal)

        # Now test whether t
        withinLimits = True
```

8. Coding Guidelines

DEMKit has no strict coding guidelines, as long as things remain manageable. The reason for this is that most of the work concerns research, for which agility and creating a proof-of-concept are more important than long term stability. Hence, pruning of code and complete classes happens more often. Usually a yearly clean-up session of essential code is considered to keep a working base and release a new stable version of DEMKit, which uses proven and (nearly) published work. Due to the nature of research, most documentation is also provided in academic output (papers, journals and Master and PhD dissertations.

8.1 DEMKit framework

The DEMKit framework has a couple of coding styles:

- Use tabs instead of spaces
- CamelCase usage for function and variable names, start with small letter
- Clear variable names
- Classes start with capital letter
- Filenames start with small letter, also use CamelCase

Development takes place in the **development** branch, whereas the **master** branch contains the last stable(-ish) version of DEMKit. New features are developed both in separate branches as the **develop** branch. If development takes place in the development branch, make sure that new additions do not cause a broken simulator. Also, add warning statements (e.g. in the **startup** function of a component) that certain features are experimental.

Logging of data also follows a standard format for clarity. Please use metric numbers and follow SI conventions (except for energy, Wh is preferred). Abstain from using Kilo's, Mega's, Milli's, etc. to make calculations using the data more easily. Grafana already adds the prefixes in visualization as well.

Please follow this naming convention:
Unit-type[.optional.number.of.specifiers][.c.commodity][.complex-part]

Examples:
Wh-energy-consumption.c.power
W-power.imported.c.power.real
W-power.imported.c.power.imag
C-temperature.indoor

8.2 Scenarios

Scenarios (in the workspace folder) are not a part of maintained code and offer full flexibility to the user to create scenarios as they please. The only exception is the read-only example file, which is maintained. The example also the scenario that provides information about the parameters that can be set.