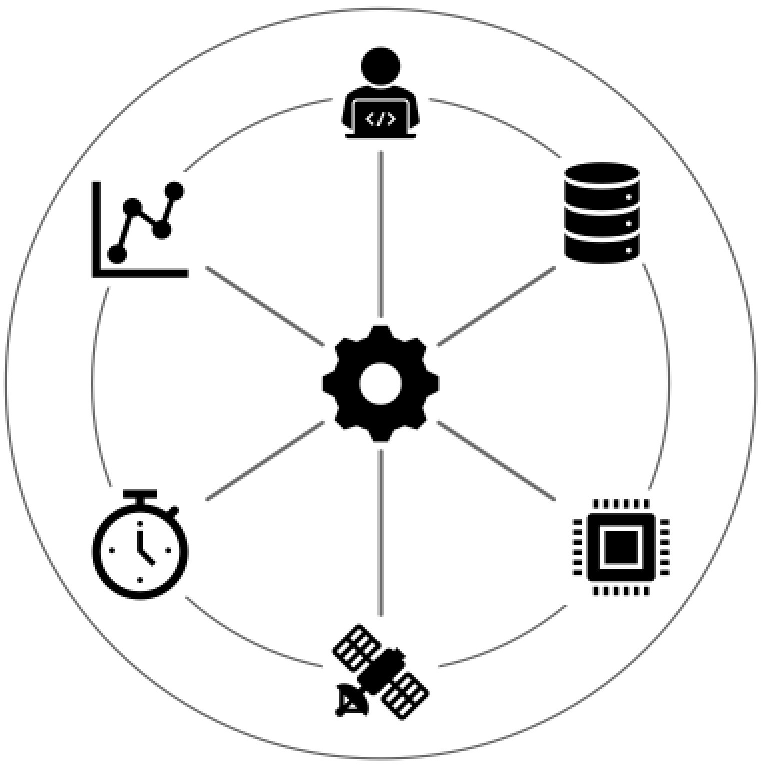


Common-EGSE

Installation Guide

Sara Regibo, Rik Huygen

Version 1.0, 20 May, 2022

Table of Contents

Chapter 1. TODO

- [ ] Where do we describe the deploy keys that are needed for updates of CGSE and TS, and for upload of Setups?
- [ ] .

Chapter 2. Introduction

This guide explains the installation and configuration of the components:

- The operating system on the egse-server and egse-client
- The installation of basic tools like git, python
- The installation of the Common-EGSE (CGSE)
- The installation of the Test Scripts (TS)

Chapter 3. Installing the Operating System

3.1. Server Installation

3.1.1. Server Hardware

- 1x Supermicro SYS-6019P-MT;
- 2x Intel Xeon Gold 5120 s3647 Skylake-SP 14 Cores 28 Threads 2.2GHz 2.6GHz Turbo;
- 4x Samsung 32GB DDR4-2666 2Rx4 (128GB main memory);
- 1x Intel SSD 240GB;
- 1x Intel SSD 2TB;
- 1x Seagate SATA 12TB;
- 2x Ethernet RJ45 1Gb/s.

3.1.2. Basic Installation of CentOS 8

Version: CentOS Linux release 8.1

Boot mode: UEFI

Perform a normal/default installation of CentOS-8.

3.1.3. User Administration

There are 4 specific PLATO users defined:

- `plato-admin`: has basically the same rights as root and is used for system installation and administration tasks. All root commands must be executed with `sudo`, no password will be asked. Do not usually login to this account.
- `plato-ops`: is used to administer and control the Common-EGSE core services. This user can start and stop the Common-EGSE control servers as a service with the systemd command `systemctl`. Log in to this account to monitor the systemd services with `systemctl` and `journalctl`. This user can also control and monitor the Prometheus and Grafana servers using `systemctl` and `journalctl`.
- `plato-data`: all services should run under `plato-data`, data locations will be writable by `plato-data` and readable by `plato-user`. You do not normally log into this account, but the services are started under this account. All the software (Common-EGSE and Test Scripts), and all the data created by the different processes and control servers can best be run and created by the `plato-data` user for consistency. If device control servers need to be started manually, use this account.
- `plato-user`: the generic user account for running the test scripts, start GUIs and analysing the data. This user has no `sudo` rights, but has read access to the `/data` directory. This is the account used to execute the test scripts. Don’t use this account on the `egse-server` unless you know what you are doing.

3.1.4. Disk and Storage

The following directories will be created on the server side:

- `/home`: used for the software installations and daily work;
- `/data`: used to store life data, i.e. image data from the FEEs, housekeeping data, logging information, metrics, etc.;
- `/archive`: used to archive all data. This data is transferred to the data archive in Leuven on a daily basis.

3.1.5. Installation of Python 3.8

We will install Python from the official Python website, as described in Installing Python. The procedure is as follows: We first install the development tools for CentOS and a number of `devel` packages that are needed for header files during compilation. We then get the Python source distribution from `www.python.org`, unpack, configure and compile. Use `altinstall` instead of `install` if you don't want previous installations of Python to be overwritten. This will install `python3.8` in `/usr/local/bin`. All the commands need to be executes as root.

```
yum -y groupinstall "Development Tools"
yum -y install openssl-devel bzip2-devel libffi-devel
yum -y install wget
wget https://www.python.org/ftp/python/3.8.7/Python-3.8.7.tgz
tar xvf Python-3.8.7.tgz
cd Python-3.8.7/
./configure --enable-optimizations
make altinstall
```

3.1.6. Installation of the Common-EGSE from GitHub

The installation will be done as the `plato-data` user. We will *clone* the IvS-KULeuven `plato-common-egse` repository. There is no need to *fork* because we will not develop from this account and will therefore not do any pull requests.

```
mkdir -p ~/git
cd git
git clone https://github.com/IvS-KULeuven/plato-common-egse.git
cd plato-common-egse/
```

The installation of the Common-EGSE will be done in a location that is accessible for `plato-data` and `plato-user`. Run the following commands as root.

```
mkdir -p /cgse/lib/python
chown -R plato-data:plato-data /cgse
ls -ld /cgse
drwxr-xr-x. 4 plato-data plato-data 4096 Sep 11 16:55 /cgse
```

Before actually installing the Common-EGSE, we have to set the `PATH` and `PYTHONPATH` environment variables.

```
PATH=/cgse/bin:$PATH
export PYTHONPATH=/cgse/lib/python/
python3.8 setup.py install --home=/cgse/
```

The above commands install the full Common-EGSE and all its dependencies in the `/cgse/lib/python` folder. Note that we have not used any Python virtual environment for this installation.

You might need to install the `wheel` package, which is needed to install binary Python distributions.

```
python3.8 -m pip install wheel
```

After a successful installation, you can check which packages are known to Python and where they are located:

```
[plato-data@localhost ~]$ python3.8 -m site
sys.path = [
    '/home/plato-data',
    '/cgse/lib/python',
    '/cgse/lib/python/Common_EGSE-2020.3-py3.8.egg',
    '/cgse/lib/python/ThorlabsPM100-1.2.2-py3.8.egg',
    '/cgse/lib/python/PyVISA-1.10.1-py3.8.egg',
    '/cgse/lib/python/PyVISA_py-0.4.1-py3.8.egg',
    '/cgse/lib/python/pylibftdi-0.19.0-py3.8.egg',
    '/cgse/lib/python/pyusb-1.0.2-py3.8.egg',
    '/cgse/lib/python/typing_extensions-3.7.4.3-py3.8.egg',
    '/cgse/lib/python/transitions-0.8.2-py3.8.egg',
    '/cgse/lib/python/transforms3d-0.3.1-py3.8.egg',
    '/cgse/lib/python/sshtunnel-0.1.5-py3.8.egg',
    '/cgse/lib/python/pyzmq-19.0.2-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/PyYAML-5.3.1-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/pytz-2020.1-py3.8.egg',
    '/cgse/lib/python/pyqtgraph-0.11.0-py3.8.egg',
    '/cgse/lib/python/PyQt5-5.15.0-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/psutil-5.7.2-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/numpy-1.19.2-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/matplotlib-3.3.1-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/jsonpickle-1.4.1-py3.8.egg',
    '/cgse/lib/python/ipython-7.18.1-py3.8.egg',
    '/cgse/lib/python/h5py-2.10.0-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/GitPython-3.1.8-py3.8.egg',
    '/cgse/lib/python/ginga-3.1.0-py3.8.egg',
    '/cgse/lib/python/distro-1.5.0-py3.8.egg',
    '/cgse/lib/python/deepdiff-5.0.2-py3.8.egg',
    '/cgse/lib/python/daiquiri-1.6.0-py3.8.egg',
    '/cgse/lib/python/click-7.1.2-py3.8.egg',
    '/cgse/lib/python/QLed-1.3.1-py3.8.egg',
    '/cgse/lib/python/six-1.15.0-py3.8.egg',
    '/cgse/lib/python/paramiko-2.7.2-py3.8.egg',
    '/cgse/lib/python/PyQt5_sip-12.8.1-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/python_dateutil-2.8.1-py3.8.egg',
    '/cgse/lib/python/pyparsing-3.0.0a2-py3.8.egg',
    '/cgse/lib/python/Pillow-7.2.0-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/kiwisolver-1.2.0-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/cycler-0.10.0-py3.8.egg',
    '/cgse/lib/python/certifi-2020.6.20-py3.8.egg',
    '/cgse/lib/python/importlib_metadata-1.7.0-py3.8.egg',
    '/cgse/lib/python/traitlets-5.0.4-py3.8.egg',
    '/cgse/lib/python/Pygments-2.6.1-py3.8.egg',
    '/cgse/lib/python/prompt_toolkit-3.0.7-py3.8.egg',
    '/cgse/lib/python/pickleshare-0.7.5-py3.8.egg',
    '/cgse/lib/python/pexpect-4.8.0-py3.8.egg',
    '/cgse/lib/python/jedi-0.17.2-py3.8.egg',
    '/cgse/lib/python/decorator-4.4.2-py3.8.egg',
    '/cgse/lib/python/backcall-0.2.0-py3.8.egg',
    '/cgse/lib/python/gitdb-4.0.5-py3.8.egg',
    '/cgse/lib/python/astropy-4.1rc1-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/QtPy-1.9.0-py3.8.egg',
    '/cgse/lib/python/ordered_set-4.0.2-py3.8.egg',
    '/cgse/lib/python/PyNaCl-1.4.0-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/cryptography-3.1-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/bcrypt-3.2.0-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/zipp-3.1.0-py3.8.egg',
    '/cgse/lib/python/ipython_genutils-0.2.0-py3.8.egg',
    '/cgse/lib/python/wcwidth-0.2.5-py3.8.egg',
    '/cgse/lib/python/ptyprocess-0.6.0-py3.8.egg',
    '/cgse/lib/python/parso-0.7.1-py3.8.egg',
    '/cgse/lib/python/smmap-3.0.4-py3.8.egg',
    '/cgse/lib/python/cffi-1.14.2-py3.8-linux-x86_64.egg',
    '/cgse/lib/python/pycparser-2.20-py3.8.egg',
    '/usr/local/lib/python38.zip',
    '/usr/local/lib/python3.8',
    '/usr/local/lib/python3.8/lib-dynload',
    '/usr/local/lib/python3.8/site-packages',
]
USER_BASE: '/home/plato-data/.local' (doesn't exist)
USER_SITE: '/home/plato-data/.local/lib/python3.8/site-packages' (doesn't exist)
ENABLE_USER_SITE: True
[plato-data@localhost ~]$
```

3.1.7. Update the Common-EGSE to the latest release

XXXXX: This must be updated !!

At some point you will be asked to update to a specific release. As an example we take release 2021.2. Execute the following commands:

```
git fetch upstream
git checkout tags/2021.2 -b 2021.2-branch
```

You have now checked out that specific release in a new branch. The next step is to update the installation:

```
python3.8 setup.py clean --all
python3.8 setup.py install --force --home=/cgse/
```

3.1.8. Open Ports on the Firewall

By default CentOS-8 has the Firewall enabled. When your system is installed in a save environment without external connectivity, you could consider to disable the Firewall altogether.

```
systemctl status firewalld
systemctl stop firewalld
systemctl disable firewalld
systemctl mask firewalld
```

When you need the Firewall to be enabled, open up some ports that are used by the Common-EGSE core services.

Open the following ports:

SM	CM	PM	DPU	Hexapod	Huber	Description
6100	6000	6200	6600	6700	6800	Commanding
6101	6001	6201	6601	6701	6801	Monitoring
6102	6002	6202	6602	6702	6802	Services
6103	6003	6203	6603	6703	6803	Heartbeat

We shall do this by introducing a new service on the server. The example below opens up the ports for the Hexapod PUNA Control Server. The commands to set up the service on the `firewalld` are:

```
sudo firewall-cmd --permanent --new-service=puna-conrol
sudo firewall-cmd --permanent --service=puna-control --set-description="Hexapod PUNA
Control Services"
sudo firewall-cmd --permanent --service=puna-control --add-port=6700-6703/tcp
sudo firewall-cmd --permanent --zone=public --add-service=puna-control
sudo firewall-cmd --reload
```

Repeat the same sequence for the other control service.

3.1.9. Setup Services for Control Servers with `systemd`

The control servers for this project that run on the `egse-server` are all managed by the `systemd` service manager. For information on **systemd** check out the documentation on the Redhat System Administration Site at [RHEL7](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/chap-managing_services_with_systemd).

The service files for each of the core control servers are located in the `server` directory at the root of the project. You will have to adapt the services, especially the absolute paths, to your needs and setup. Then copy the service files into the `/etc/systemd/system` directory:

```
sudo cp sm_cs.service /etc/systemd/system
sudo cp cm_cs.service /etc/systemd/system
sudo cp pm_cs.service /etc/systemd/system
sudo cp log_cs.service /etc/systemd/system
```

The following code lists the entire service for the Storage Manager Control Server. The text `EnvironmentFile` and `WorkingDirectory` need special attention for your specific setup.

```
[Unit]
Description=Storage Manager Control Server
After=network-online.target

[Service]
Type=simple
Restart=always
RestartSec=3
User=plato-data
Group=plato-data
EnvironmentFile=/cgse/env.txt
WorkingDirectory=/home/plato-data/workdir
ExecStart=/cgse/bin/sm_cs

[Install]
Alias=sm_cs.service
WantedBy=multi-user.target
```

The service starts the specific control server from a script that was created during the `setuptools` installation, in our example in the `/cgse/bin` folder. Check the services files for the Configuration Manager and Process Manager also, they contain a specific delay time of 3s to ensure the Storage manager had enough time to start up and process registrations.

```
[Service]
ExecStartPre=/bin/sleep 3
```

You also might need to create the `/home/plato-data/workdir` folder for the user `plato-data`.

Once the services file is correct, start the service as follows:

```
sudo systemctl start sm_cs
```

and to automatically start the service on boot:

```
sudo systemctl enable sm_cs
```

The counter parts of the above commands are **stop** and **disable** where the former just stops the service and the latter prevents the service to start at boot time.

Whenever you have made a change to the services file and copied it back into the `/etc/systemd/system` directory, reload the daemons as follows:

```
sudo systemctl daemon-reload
```

If you need to know the status of one of the control services, use the following command, e.g. for the Process manager:

```
sudo systemctl status pm_cs.service
```

This prints out the status info on the service plus the last few messages that were send to stdout or stderr.

When you want to check and follow the output in `/var/log/messages` for the specific service, you can use the following journal command:

```
sudo journalctl -f -u pm_cs
```

When you run into a authentication error while starting the control servers, you might need to disable SELinux (Security-Enhanced Linux). The error will look something like this (excerpt from `/var/log/messages`):

```
Sep 11 17:59:46 localhost systemd[1]: sm_cs.service: Service RestartSec=3s expired,
scheduling restart.
Sep 11 17:59:46 localhost systemd[1]: sm_cs.service: Scheduled restart job, restart
counter is at 369.
Sep 11 17:59:46 localhost systemd[1]: Stopped Storage Manager Control Server.
Sep 11 17:59:46 localhost systemd[1]: Started Storage Manager Control Server.
Sep 11 17:59:46 localhost systemd[22013]: sm_cs.service: Failed to execute command:
Permission denied
Sep 11 17:59:46 localhost systemd[22013]: sm_cs.service: Failed at step EXEC spawning
/cgse/bin/sm_cs: Permission denied
Sep 11 17:59:46 localhost systemd[1]: sm_cs.service: Main process exited, code=exited,
status=203/EXEC
Sep 11 17:59:46 localhost systemd[1]: sm_cs.service: Failed with result 'exit-code'.
Sep 11 17:59:47 localhost setroubleshoot[19162]: failed to retrieve rpm info for
/cgse/bin/sm_cs
Sep 11 17:59:47 localhost setroubleshoot[19162]: SELinux is preventing
/usr/lib/systemd/systemd from 'read, open' accesses on the file /cgse/bin/sm_cs. For
complete SELinux messages run: sealert -l a77af8c2-c91a-43cd-9b64-e7c0a5b24311
Sep 11 17:59:47 localhost platform-python[19162]: SELinux is preventing
/usr/lib/systemd/systemd from 'read, open' accesses on the file
/cgse/bin/sm_cs.#012#012*****  Plugin catchall (100. confidence) suggests
**************************#012#012If you believe that systemd should be allowed read
open access on the sm_cs file by default.#012Then you should report this as a
bug.#012You can generate a local policy module to allow this access.#012Do#012allow
this access for now by executing:#012# ausearch -c '(sm_cs)' --raw | audit2allow -M
my-smcs#012# semodule -X 300 -i my-smcs.pp#012
```

To disable SELinux, edit the `/etc/selinux/config` file and set `SELINUX=disabled`. Then reboot your system (this is a kernel setting, therefore we need to reboot).

3.1.10. Install the Prometheus server

Please note that in the developer documentation under the section [Monitoring](../dev-docs/monitoring.md) there is a description on *Installing Prometheus*. I will here only describe the setup for the `egse-server`. The best is to create a dedicated directory for the software installations, e.g. `~/software`. Then install Prometheus into that folder:

```
mkdir ~/software
cd ~/software
wget https://github.com/prometheus/prometheus/releases/download/v2.25.0/prometheus-
2.25.0.linux-amd64.tar.gz
tar xzvf prometheus-2.25.0.linux-amd64.tar.gz
ln -s prometheus-2.25.0.linux-amd64 prometheus
```

We want to automatically start the Prometheus server from the systemd services as we did with the core egse services. The service file, i.e. `prometheus.service`, can be copied from the `server` directory in the distribution to the `/etc/systemd/system` folder, same as for the core egse services. Make sure you update the locations if necessary. The configuration files for Prometheus, i.e. `prometheus-egse-`

server.yml and prometheus.rules.yml, should be copied from the metrics folder into the installation folder of Prometheus.

```
cd ~/git/plato-common-egse
cp ./server/prometheus.service /etc/systemd/system
cp ./metrics/prometheus-egse-server.yml ~/software/prometheus
cp ./metrics/prometheus.rules.yml ~/software/prometheus
```

Finally, create the metrics/data directory in the proper location, e.g. in /data. That is the location given with the --storage.tsdb.path option in the Prometheus service file.

```
mkdir -p /data/metrics/data
```

Then enable the service as user plato-admin and reload the systemd services daemon:

```
sudo systemctl enable prometheus
sudo systemctl daemon-reload
sudo systemctl start prometheus
```

3.1.11. Install the Grafana server

Please note that in the developer documentation under the section [Monitoring](../dev-docs/monitoring.md) there is a description on *Installing Grafana*. I will here only describe the setup for the egse-server. The best is to create a dedicated directory for the software installations, e.g. ~/software. Then install Grafana into that folder.[1]

```
wget https://dl.grafana.com/oss/release/grafana-7.4.2.linux-amd64.tar.gz
tar xzvf grafana-7.4.2.linux-amd64.tar.gz
ln -s grafana-7.4.2 grafana
```

Grafana doesn't need any further configuration. That is done in the dashboards that are loaded as explained in XXXXX Monitoring/Dashboard Configuration.

We also want the Grafana server to automatically start from the systemd services as we did for Prometheus. We currently use Grafana with the default configuration and have the database located in the installation directory. The service file is located in the server folder of the Common-EGSE project and should be copied to /etc/systemd/system. After that, enable Grafana with systemctl and reload the services daemon.

3.1.12. Check your services

A simple and quick way to check if the core services are still running together with Prometheus and Grafana is to check the running processes:

```
[plato-data@egse-server grafana]$ ps -ef|egrep "prometheus|grafana|_cs"
plato-d+    2338       1  0 Feb17 ?        00:00:39 /usr/bin/python3 /cgse/bin/log_cs
start
plato-d+    2344       1  0 Feb17 ?        00:21:53 /usr/bin/python3 /cgse/bin/sm_cs
start
plato-d+    3515       1  0 Feb17 ?        00:21:08 /usr/bin/python3 /cgse/bin/pm_cs
start
plato-d+   17538       1  0 Feb18 ?        00:07:01 /home/plato-
data/software/prometheus/prometheus --config.file /home/plato-
data/software/prometheus/prometheus-egse-server.yml --storage.tsdb.path
/data/metrics/data/
plato-d+   21225       1  0 Feb18 ?        00:01:20 /home/plato-
data/software/grafana/bin/grafana-server
plato-d+   37154       1  0 Feb18 ?        00:07:50 /usr/bin/python3 /cgse/bin/cm_cs
start
plato-d+   73223   29726  0 15:51 pts/0    00:00:00 grep -E --color=auto
prometheus|grafana|_cs
[plato-data@egse-server grafana]$
```

3.2. Client Installation

3.3. Setting up the users

3.4. Installation of Python

3.5. Installation of PyCharm

PyCharm is the IDE that we will use to execute test scripts. For this purpose the PyCharm Community edition is sufficient.

[1] Don't try to install Grafana using `yum`, because that will bring you into trouble with configuration files etc.

Chapter 4. Installing and configuring Git and GitHub

The `plato-common-egse` code is under version control in GitHub. To be able to get the latest version of the code on your local machine and to share possible contributions with other people in the project, you need to install Git on your computer. Git might already be installed on your system, check it with the `git --version` command in your terminal.

Installation instructions for your operating system can be found on the Git reference documentation.

4.1. Access to GitHub Repository

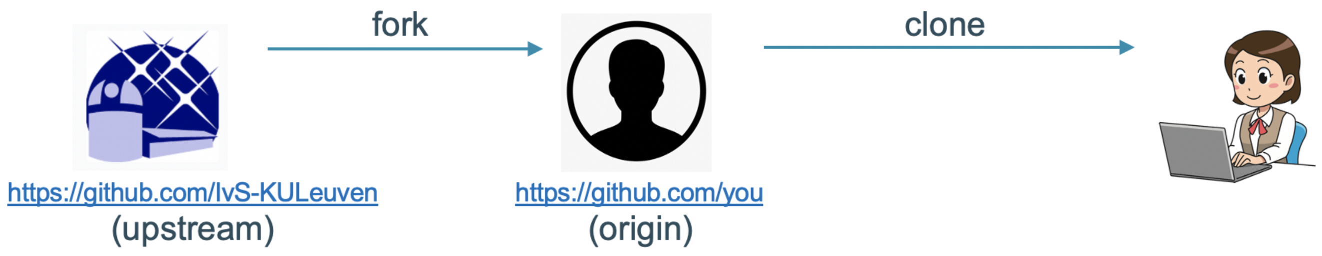
The `plato-common-egse` code is located in a private repository in GitHub. To be able to access it, we have to grant you access explicitly. Please, send your GitHub username to the development team and you will be granted read access to the repository. You will then get an invitation by email to join.

If you have not done so already, you can make an account on GitHub for free.

4.2. Forking & Cloning the Repository

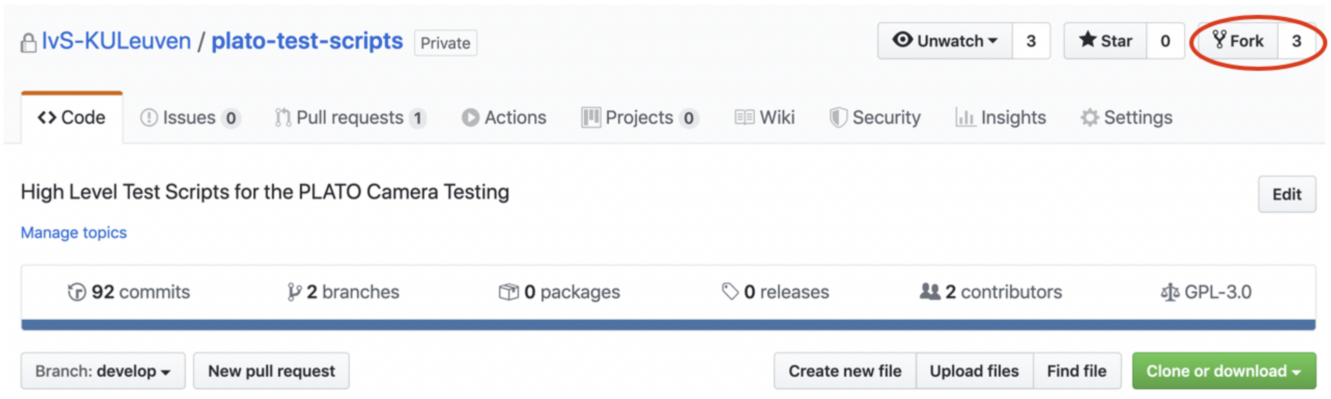
The `plato-common-egse` repository can be found on the IvS-KULeuven GitHub pages. This repository is referred to as **upstream**.

This section describes the process how to install a copy of the repository on your local machine. Note that this is a two step process as shown in the diagram below.



4.2.1. Fork a repository

When you not only want use the code but also contribute to it, you have to "fork" this repository. To do this, go to the upstream GitHub page, shown below.



High Level Test Scripts for the PLATO Camera Testing

92 commits · 2 branches · 0 packages · 0 releases · 2 contributors · GPL-3.0

Press the `"Fork"` button at the top right (encircled in red in the screenshot above) and follow the instructions. Your personal copy of the `plato-common-egse` repository will then show up on your personal GitHub pages. This copy is referred to as **origin**.

4.2.2. Clone a repository

To create a local copy of the repository, *clone* it to a designated directory on your local machine with the following command. This will create a folder `plato-common-egse` in the `~/git` directory.

```
$ cd ~/git
$ git clone https://github.com/<your GitHub username>/plato-common-egse.git
```

After executing these steps, you should see the following:

- on your personal GitHub page: the forked repository;
- on your local machine: a local copy (i.e. clone) of the repository in the `~/git` folder.

XXXXX: what's next?

Make sure the environment variables are set: PLATO_COMMON_EGSE_PATH, and PLATO_INSTALL_LOCATION

- install for develop → `python -m pip install -e .`
- install for ops → 'python setup.py install --force --home=$PLATO_INSTALL_LOCATION`

Chapter 5. Installing the Common-EGSE

5.1. First-time Installation

5.2. Update the Common-EGSE

Chapter 6. Installing the Test Scripts

6.1. First-time Installation

6.2. Update the Test Scripts

Chapter 7. Installating External Tools

7.1. Cutelog GUI

7.2. Textualog TUI