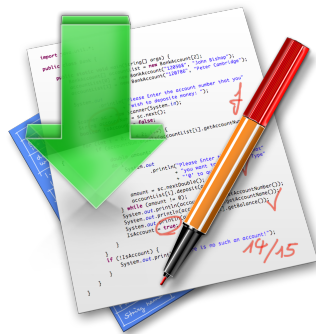


---

# Using the Grit Submission System

---

Official User Manual for Version 1.1



July 21, 2015

# Contents

<b>1</b>	<b>What is Grit?</b>	<b>3</b>
1.1	What does Grit do? . . . . .	3
1.2	When do you need Grit? . . . . .	4
1.3	What are the alternatives? . . . . .	4
<b>2</b>	<b>Getting Help</b>	<b>6</b>
2.1	Online Documentation . . . . .	6
2.2	Contact us . . . . .	6
<b>3</b>	<b>Getting Grit</b>	<b>7</b>
3.1	Required Software . . . . .	7
3.2	Using installation script . . . . .	8
3.3	Direct installation (by hand) . . . . .	8
3.4	Prerequisites . . . . .	9
<b>4</b>	<b>Running Grit</b>	<b>10</b>
4.1	How to run Grit . . . . .	10
4.2	First Startup . . . . .	10
4.3	Setting up a submission structure . . . . .	14
4.4	Configuration . . . . .	15
4.5	E-Mail transmission function . . . . .	16
4.6	E-Mail Submission system . . . . .	19
4.7	Data Source Specific Requirements . . . . .	21
<b>5</b>	<b>Frequently asked Questions</b>	<b>22</b>
<b>6</b>	<b>Credits</b>	<b>23</b>
6.1	Developers Team GRIT . . . . .	23
6.2	Developers Team VARCID . . . . .	23
6.3	Special Thanks . . . . .	24

# 1 | What is Grit?

Grit is a system, developed by TEAM GRIT and improved by TEAM VARCID, that makes the life of lecturers and tutors of mainly programming courses easier and more comfortable. With Grit, we implemented a server-side one-button solution for fetching and processing student submissions. This is done in such a way that the tutors will not have to worry about not-compiling programs and can therefore concentrate on the actually written code and its quality. Grit helps to maximize the teaching quality by ruling out unnecessary work and lets the tutors focus on the really important matter: Teaching.

## 1.1 What does Grit do?

With Grit, the students do not directly submit their solutions to our application. To be precise, they will not even notice that you are using Grit when submitting. Rather, Grit fetches the student submissions from whatever system provided for that purpose. As for now, Grit supports Subversion, and Email submissions, however, because of its high modularity, you can easily add your own *fetcher*. At the end of a previously set deadline, Grit fetches the submissions and processes them. As of now, we support Java, C and Haskell submissions, but like *fetchers*, additional modules can be added. After the processing, which includes compiling and testing the submission, using previously set unit tests, we will put that acquired information about the submission in a report, ready for you to download. For now that report will be either a PDF or a plain-text file, but, as you will have guessed, other formats can be implemented easily.

### What Grit does not.

Grit does not provide submission features, as it can only import submissions. Also, it is not possible to manipulate code in Grit, because this is not what it was intended for: offline correction on paper. Grit does not replace a lecturer or tutor by adopting all of their correction procedure, however Grit simplifies their work in supporting them with automatable checking routines. So, Grit does not offer any methods to control semantical errors in the program code like checking if the summation really does add the two summands. Furthermore Grit is not suitable to test if the submitted solutions of the students actually solves the given problem or a part of it. Grit cannot check any additional exercise restrictions specified in the assignment paper (e.g. using inherited methods for solving the exercise).

Grit will also not correct any programming faults automatically, but will present you the incorrect passage, the error, the passed tests as well as compiler error message.

## 1.2 When do you need Grit?

You are a lecturer/tutor of a general programming course, a java workshop, a C++ seminar, ..., in which the participants can or have to submit solutions to a given problem. If you are tired of searching for syntax errors, compiling errors, typos, etc. for hours at a time, then you will definitely appreciate our product to increase the efficiency of your correction by prechecking every submitted solution. All in all, if you are a university or college professor or lecturer, teacher, tutor, or corrector, Grit can help you automate collecting electronically submitted assignments concerning programming code from various resources. Compiling tests and predefined tests can be run so you can easily focus on the important things.

### When you do not need Grit.

You will not need Grit if you do not provide electronic submission of assignments for your course. Also, Grit is only suitable for fetching programming code assignments, however, other types of submissions can be added by oneself. Grit does not offer any features for students, as it only provides functionalities used by teachers and correctors.

## 1.3 What are the alternatives?

TEAM VARCID has compared Grit with the following alternatives. You can find it in another document.

- The Marmoset Project  
([marmoset.cs.umd.edu](http://marmoset.cs.umd.edu))  
Marmoset is a system for handling student programming project submission, testing and code review. It has been developing at the University of Maryland for over 5 years. It works in all different programming languages, and is designed to work well with both very small and very large projects, such as our OS course in which a submission consists of tens of thousands of lines of code.
- BOSS Online Submission System (Beta phase)  
([sourceforge.net/projects/cobalt](http://sourceforge.net/projects/cobalt))  
BOSS is a course management tool that allows students to submit assignments online in a secure manner. Staff can mark work and run automatic tests on submissions.
- Cafe grader  
([gitorious.org/cafe-grader](http://gitorious.org/cafe-grader))  
Cafe grader is a submission and grading system for programming contest and training. This software was used in APIO'08. It currently has mainly two components: the web submission system and the grader. The web app uses Ruby on Rails; while

the grader was written in plain Ruby. Current activity: process of migrating to git, and developing installation scripts.

- Praktomat

(<https://github.com/KITPraktomatTeam/Praktomat/>)

Praktomat is a quality control system for programming exercises. Students hand in their submissions directly into Praktomat which then compiles and tests them. It was first developed at the University of Passau in 1998 and was later migrated to the Karlsruhe Institute of Technology in 2008. The software is written in Python and uses the Django web framework. It is able to manage Java, other programming languages can be added.

## 2 | Getting Help

There are various resources available to get the most out of Grit.

### 2.1 Online Documentation

The online version of the documentation is essentially the same as the one you are reading now, however, you will have access to additional resources concerning troubleshooting. But online you will find the newest version.

### 2.2 Contact us

Do you have suggestions? Do you miss a feature? Contact TEAM GRIT or TEAM VARCID by mail. You will find our mail addresses at the end of the document.

## 3 | Getting Grit

As you are reading this documentation now, you have most likely already acquired Grit. It is either directly available by getting it from the developers, or found on the associated GIT repository, or found on the associated GIT repository.

Fundamental there are two ways to install Grit.

- direct on a Linux server system
  - with our installation script
  - by hand

### 3.1 Required Software

For installing Grit you need the current version of the following software. The packages will also be installed with the installation script (see 3.2).

- Java Development Kit (`jdk7` or higher)
  - The package contains the `javac` compiler, which you need to compile the students' Java submissions
- ~~TeX~~live (`texlive-full`)
  - you need Texlive to create your reports
- GNU-C-Compiler (`gcc`)
  - to compile the students' C submissions
- Glasgow Haskell Compiler (`ghc`)
  - to compile the students' Haskell submissions
- Subversion (`svn`)
  - Grit can fetch submissions from subversion,
- Secure Copy (`openssh` server and client)
- G++ Compiler (`g++`)

- Git
  - you need Git to get Grit.
- Gradle
  - you need Gradle to install Grit.

## 3.2 Using installation script

**Warning:** you need a Linux system with apt-get to use the script for a direct installation!

You have to take the script `install-GRIT.sh` and put it at the place, where Grit shall be installed. The required software will be installed by the script! Now you have to run script.

For Ubuntu the next step is to define the rights:

```
chmod u+x install-GRIT.sh
```

and run the installation script:

```
./install-GRIT.sh
```

Grit should now be installed in a folder `./grit/build/install/GRIT` relative to the directory where you run the install script.

## 3.3 Direct installation (by hand)

**Warning:** you need a Linux system to do a direct installation because Grit uses Unix shell commands.

The first step is to install all required files which are listed in 3.1. Next thing to do is to select the location where grit should be installed. Then you have to generate a git clone in to this location:

```
$ git clone https://github.com/VARCID/grit.git
```

Now there should be a folder called `grit`. You have to go inside the folder:

```
$ cd grit
```

In the next step you have to run gradlew:

```
% $ ./gradlew NOTE: not needed
$ ./gradlew install
```

Grit should now be installed in a folder `./grit/build/install/GRIT` relative to the directory where you run gradlew install.



## 3.4 Prerequisites

### 3.4.1 SVN repository setup

If you plan to use an SVN repository a "mapping file" must be provided. This file maps the name of the student folders to the student email address. Create a file `students.txt` in the top level of the repository. This file should contain only blank lines or lines formatted like `StudentFolder = StudentMail`. Lines beginning with the `#`-sign will be ignored.

The file might look like this:

```
# this is a comment
stud01 = max.mustermann@test.de
stud02 = elsbeth.musterfrau@test.de
```

## 4 | Running Grit

Now after successfully setting up your system you can finally begin using Grit. Grit runs as a Java application that sets up a web page. All administration and most of the configuration will run over that website. We tried to make your experience as intuitive as possible, but still there might be some ambiguities. For that we put an overview of the whole user interface at the end of this chapter. But first we will guide you through the typical workflow we designed and explain the elements step by step in the order you will most likely encounter them.

### 4.1 How to run Grit

You probably have installed Grit in a folder structure similar to `./grit/build/install/GRIT` relative to the directory where you run the install script. There you should find two important scripts, `runscript.sh` and `shutdownGrit.sh`.

Use this to start Grit:

```
$ ./runscript.sh
```

`Runscript.sh` will start Grit independently from your shell. Outputs are saved in `./log/grit.out` and deleted on every new startup. Use this to end Grit:

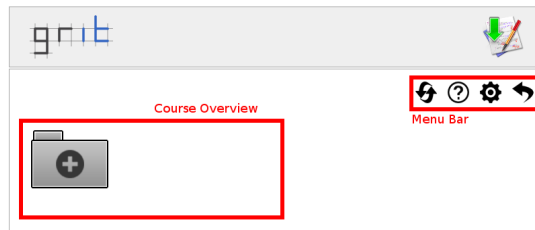
```
$ ./shutdownGrit.sh
```

`ShutdownGrit.sh` will terminate Grit. You can run this command from any shell-session. You can not restart Grit without running this script first. After `shutdownGrit.sh` the `grit.out` file in the log directory still exists, this file can be a great help in addition to `system.log` should any unknown errors occur.

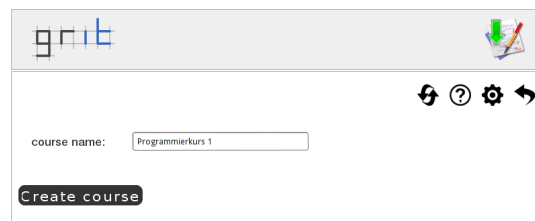
### 4.2 First Startup

After booting up, the web page will be available via the 8080 port of the Machine Grit is running on. You will need to activate Javascript in your Browser to properly use the Website. You will see a username and password prompt. The default username is `username`

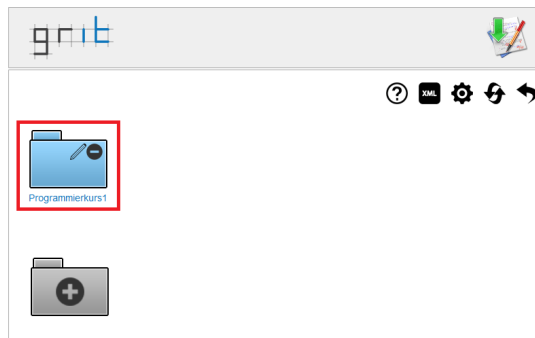
and the default password is `password`. Now it is important that you provide the system with all necessary configuration information. Essentially there are two ways you can do this. The first way is to create a valid fully configured `config.xml` before booting the system so it can load it while booting. The second way is to configure the system with the `edit xml` function on the webinterface. For further explanation see section 4.4. On your first visit of the site you will be presented an empty course overview and the menu bar.



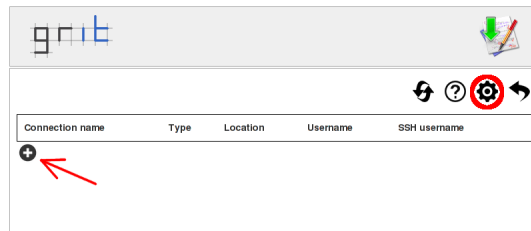
Before you start creating new courses you should read section 4.7 because you may need to fulfill external requirements. Now you will need to create a course. For that click on the folder with the plus sign.



You will be prompted to enter the course name, after that click on `Create course`. The created course will now be visible in the overview.



For creating an exercise we will first need to set up a connection so Grit knows where to get the students' submissions from. For that click on the "gear wheel" symbol in the menu bar. You get now presented the connection overview.



To add a new one click on the plus sign. You will get to a form where you can enter the connection information:

**Connection Name:** Name for you to identify the connection.

**Connection Type:** Either SVN or Mail is supported by Grit 1.1.

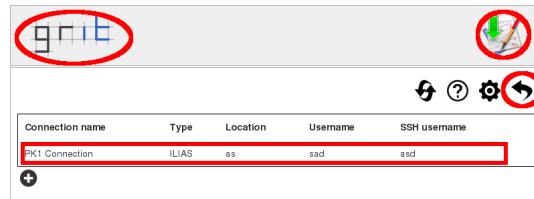
**Location:** The url of the root folder where the submissions will be. The root folder has to contain the `students.txt`

**Username:** The name of the SVN account.

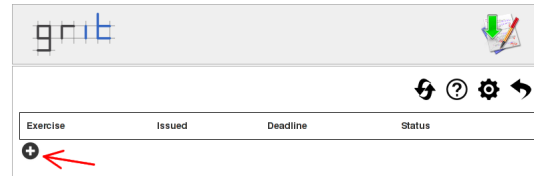
**Password:** The password for the SVN or Mail account.

**Structure:** The structure that Grit will encounter so it knows where to get the submission. For more see Section 4.3.

At the end click on **Create Connection**. And you will see the created Connection in the overview.



Now you go back to the course overview. Either by clicking the "back arrow" or by clicking the Grit logo. Now click on your created course and you will see the exercise overview.



Similar to the connection overview you can now create a new exercise by clicking the plus button and again will be prompted a form:

**Exercise Name:** The name of the Exercise that will also be printed on the output document (i.e. Exercise 1/Übung 1).

**Language:** The programming language for the Exercise. Java, C/C++ and Haskell are supported as of Grit 1.1.

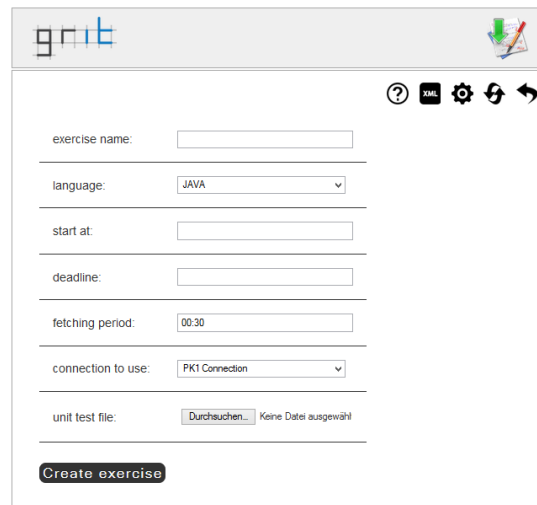
**Start at:** The date the exercise gets handed out to the students.

**Deadline:** The date when the students must have submitted their submissions.

**Connection to use:** Here you will select the connection, previously created, to be used with the exercise.

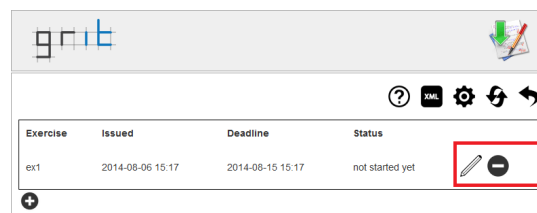
**Unit Test file:** The (optional) file that contains Unit Tests to test for. Only JUnit is supported by Grit 1.1.

Please mind that the fetching periods (the time interval between two fetches) have to fit into the interval between start time and deadline. So if you create an exercise e.g. between 10:00 and 18:00, the fetching period must not be 45 minutes because then the last fetch will be 15 minutes past the deadline. Also you should create the exercise *before* the start time, since otherwise the fetching period will be counted from the creation time.



The screenshot shows the Grit web interface with the 'Create exercise' form. The form includes fields for 'exercise name', 'language' (set to 'JAVA'), 'start at', 'deadline', 'fetching period' (set to '00:30'), 'connection to use' (set to 'PKI Connection'), and 'unit test file' (with a 'Durchsuchen...' button and the text 'Keine Datei ausgewählt'). A 'Create exercise' button is at the bottom.

Now you can click `create exercise` and you will see the just created exercise in the overview. If your settings made are correct everything is set up now and you can wait for the deadline. If you made some error you can click the pencil symbol to the right of the exercise entry and change the settings.



The screenshot shows the Grit web interface with the exercise overview table. The table has columns for 'Exercise', 'Issued', 'Deadline', and 'Status'. A red box highlights the pencil and minus icons next to the exercise entry.

Exercise	Issued	Deadline	Status
ex1	2014-08-06 15:17	2014-08-15 15:17	not started yet

Finally the end of the deadline we will process the submissions and when finished an additional symbol will be visible next to the exercise entry: With that you can download the PDF Document containing the students submission, the compiler output and - if existing - the test results.

### 4.3 Setting up a submission structure

In order to be able to correctly match student submission files to students in SVN repositories, Grit needs to know the structure of the directory tree it is supposed to look at. For this purpose you need to enter a comma separated list of regular expressions which match the path Grit needs to traverse to find submissions. The special tags `TOPLEVEL` and `SUBMISSION` indicate the repository root and the submission level respectively.

For example, your repository might have two folders at its root: `exercise/` and `lecture/`. While the latter contains no submissions, `exercise/` contains a folder for each student:

stud01/, stud02/, stud03/ and so on. Each of these students has a folder for each exercise: ex01/, ex02/, ex03/ and so on.

In order to tell Grit to look in a specific folder(e.g. ex01) you would specify the following list:

```
TOPLEVEL
exercise
stud[0-9][0-9]
ex01
SUBMISSION
```

TOPLEVEL stands for the root of the repository. Then Grit will only look at the `exercise` folder. In there, all folders from `stud01` upto `stud99` will be inspected. In each student folder Grit looks at the folder `ex01`. From here the system will retrieve the current submission. With the regular expression as seen above you can have up to 99 student folders. If you need more, you just need to add `[0-1]` between `stud` and `[0-9]`.

## 4.4 Configuration

The config file contains important data for the server, the mail notification and the admin account. This information is stored in an `.xml` file located at `config/config.xml`. If there is no config present at boot the standart failsafe config will be loaded. This config looks like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<config>
  <server>
    <port value="8080"/>
  </server>
  <email>
    <auth password="" adress="" host=""/>
  </email>
  <admin>
    <user name="username" email="" password="password"/>
  </admin>
</config>
```

The structure consists of the following items:

**<config>**: Wrapper tag which specifies when the config starts and when it ends.

**<server>**: Holds info about the server.

**<port>**: The port the server is running on.

**<email>**: Holds info about the email account where the notification mails are send from.

**<auth>**: All relevant authentication info for the mail account. The password, the mail address and the SMPT host server(e.g. smtp.gmail.com).

**<admin>**: Holds info about the admin account.

**<user>**: The data with which you want to log into the website and the email the admin want to get notifications on.

In order to customize the config you just change the values between the quotation marks. To do so please use the respective edit xml function on the webinterface. After the configuration was changed the system needs to be rebooted. This will be done automatically in the edit xml function. You should especially consider this if you changed the config file without the edit xml function on the webinterface.

## 4.5 E-Mail transmission function

The e-mail sending function for GRIT allows to set up the system so that e-mails will be sent automatically if any number of the following conditions are complied:

- The file which is delivered by the student is not plausible, i.e the file extension doesn't conform with the required one
- The file which is delivered by the student doesn't compile

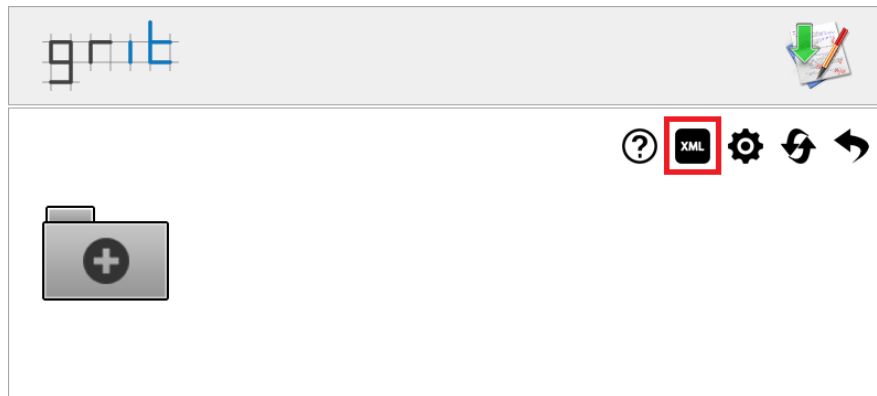
If the SVN-Submission system is used as Connection-Type e-mails will also be sent automatically if any number of the following conditions are complied:



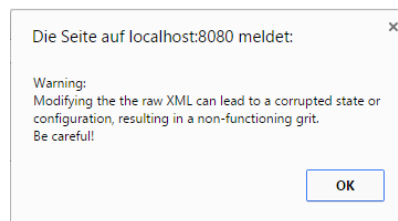
- The student hasn't committed anything 12 hours before the deadline is expired
- The student hasn't committed anything 6 hours before the deadline is expired

### Establishment of e-mail transmission function

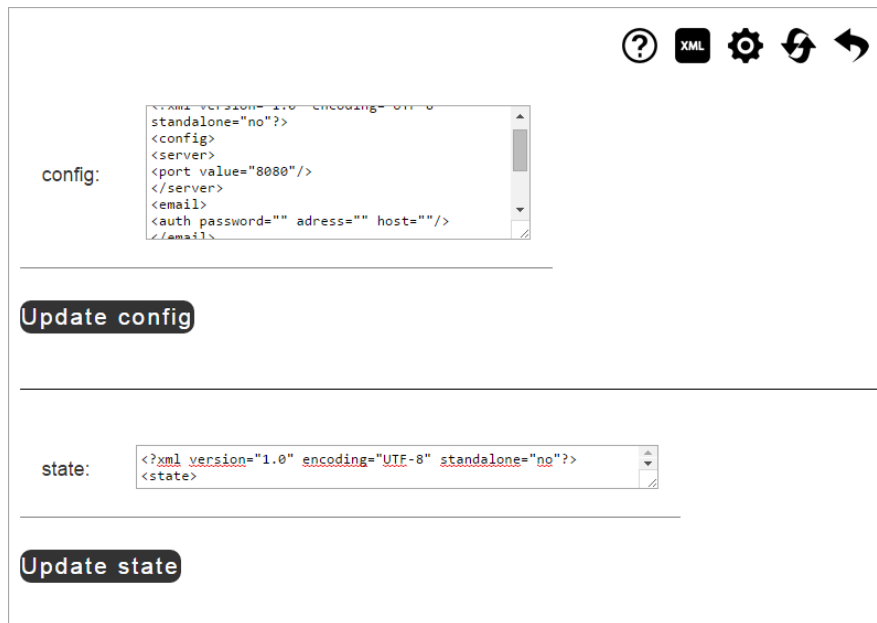
This section explains how to set up the e-mail transmission function. Therefore you have to press the button called "XML" of the web interface.



Thereupon it appears a warning that you have to be carefully when you want to change the XML-File.



By pressing the OK-Button you get to the actual XML-Configuration page:



The screenshot shows the GRIT configuration interface. At the top right, there are icons for help, XML, settings, refresh, and back. The 'config:' field contains the following XML code:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<config>
<server>
<port value="8080"/>
</server>
<email>
<auth password="" adress="" host=""/>
</email>
</config>
```

Below the 'config' field is a button labeled 'Update config'.

The 'state:' field contains the following XML code:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<state>
```

Below the 'state' field is a button labeled 'Update state'.

To set up the e-mail transmission function you need only the config-field:



The screenshot shows the GRIT configuration interface. At the top right, there are icons for help, XML, settings, refresh, and back. The 'config:' field contains the following XML code:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<config>
<server>
<port value="8080"/>
</server>
<email>
<auth password="" adress="" host=""/>
</email>
<admin>
<user name="username" email="" password="password"/>
</admin>
</config>
```

Below the 'config' field is a button labeled 'Update config'.

At the config-field the following fields are only important:  
<email>

- auth password="PasswordForTheMailsendingAccount"  
e.g.: auth password="password"
- adress="MailsendingAddress"  
e.g.: adress="testaccount@gmail.com"

- host="MailsendingHost"  
e.g.: host="smtp.gmail.com"

<admin>

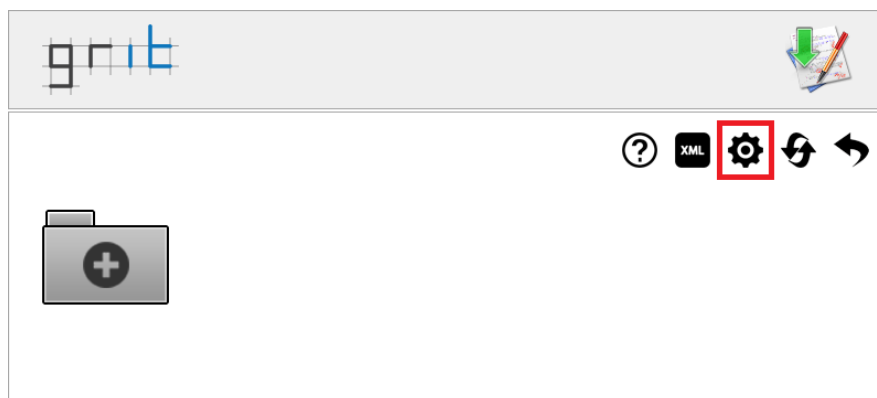
- email="MailaddressOfAdmin"  
This is the e-mail-address to which GRIT sends an e-mail to, as soon as the report-pdf is ready for download.  
e.g.: email="adminemail@uni-konstanz.de"

## 4.6 E-Mail Submission system

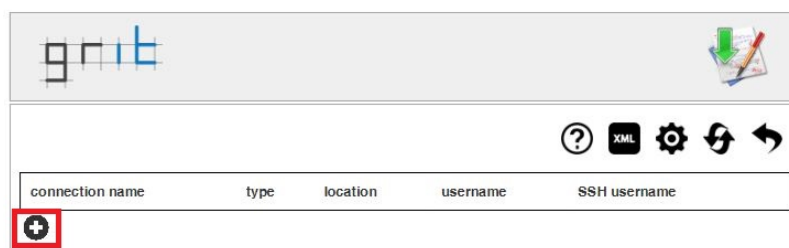
The e-mail submission system allows to set up GRIT so that students are able to submit their solutions by e-mail.

### Establishment of the e-mail submission system

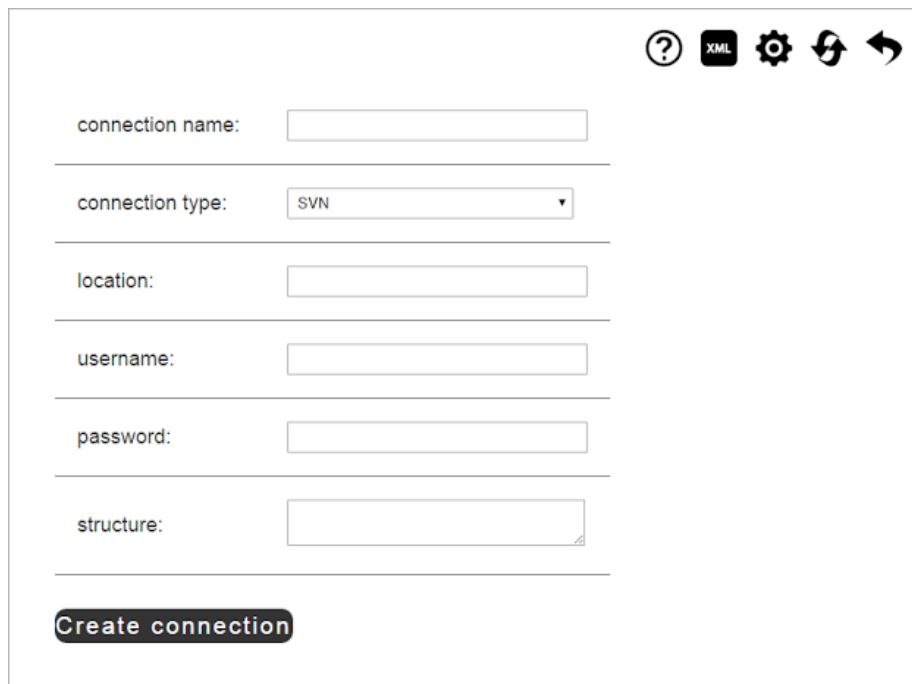
To set up the e-mail submission system you have to choose the "Connection type" "EMAIL" in a new connection. For this you have to press the button on the web interface which looks like a gear.



After this there appears the connection-interface on the web interface. There you have to press the button which looks like a circled plus symbol to set up a new connection:



Now appears the standard-entry mask:

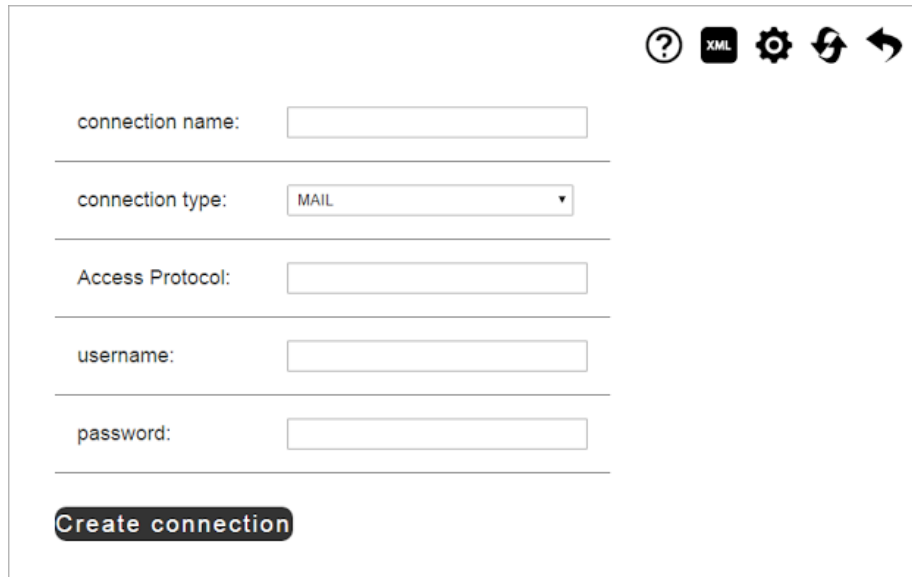


The screenshot shows a web form for creating a connection. At the top right, there are five icons: a question mark, a square with 'XML', a gear, a circular arrow, and a square arrow. The form fields are as follows:

- connection name:
- connection type:
- location:
- username:
- password:
- structure:

At the bottom left, there is a button labeled "Create connection".

There you have to set up the "Connection-Type" to "MAIL". Therefore only the necessary input fields for the e-mail submission system are visible:



The screenshot shows the same web form, but with the "connection type" set to "MAIL". The visible fields are:

- connection name:
- connection type:
- Access Protocol:
- username:
- password:

The "structure" field is no longer visible. The "Create connection" button remains at the bottom left.

Explanation of the fields:

- connection name: The name which identifies this connection.  
e.g.: EMAIL
- connection type: On which way the submission system fetches the submissions. Either "SVN", which is the standard type or "MAIL", which is selected in this case.
- Access Protocol: The network protocol of the used e-mail account  
Please notice that the network protocol should be able to access e-mails received by the e-mail-server  
e.g.: imap.gmail.com
- username: The e-mail-address to which the students will send their submissions.  
e.g.: git@gmail.com
- password: The password for the e-mail-account

## 4.7 Data Source Specific Requirements

There are some things you need to consider when creating courses and exercises.

### 4.7.1 SVN

For SVN you have to make sure to choose a different connection for every exercise because the submission structure in the connection specifies which exercise you want to fetch from the repository.

Besides you should guarantee that the submissions of the students must not contain whitespaces in the file name, otherwise the submission cannot be evaluated.

### 4.7.2 Email

If you decided to use email submissions, you already have specified the email address from which Grit should fetch email submissions. In order for this to work as intended, students have to name the email subject according to the following formula:

`[course name-exercise name]`

E.g. when the course name is "Programmierkurs 1" and the exercise name is "1" (for the first exercise), then the subject should be `Programmierkurs 1-1` so Grit can find the submission. Also consider that submissions handed in after the dead line are not imported.

## 5 | Frequently asked Questions

1. *What does Grit do?*

Grit assists you in running a programming course where it is required for students to hand in assignments done in code. The submissions can be automatically fetched from associated repositories and automatically tested. Grit delivers a well-structured and comprehensive document ready to be graded by a corrector as well as overall and detailed course statistics.

2. *What does Grit not do?*

Grit integrates between the steps assignment issuance, submission and correction. To be precise, it fills the formerly manual between submission and correction, as well as everything that comes after correction. Grit does not provide facilities to issue assignments, submit them, and, certainly cannot correct them.

3. *On what type of machine should I install Grit?*

It is possible to install Grit on different Linux systems (It is tested on Ubuntu, Debian and OpenSUSE)

4. *What is the default username and password?*

The default username is `username` and its password is `password`.

5. *Should I change the predefined passwords of the provided VM?*

Yes.

## 6 | Credits

### 6.1 Developers Team GRIT

Gabriel Einsdorf

Marvin Gülzow

Eike Heinz

Marcel Hiller

David Kolb

Fabian Marquart

Thomas Schmidt

Stefano Woerner

### 6.2 Developers Team VARCID

Verena Berg

(Verena.Berg@uni-konstanz.de)

Annelie Bruder

(Annelie.Bruder@uni-konstanz.de)

Robert Schmid

(Robert.Schmid@uni-konstanz.de)

Cedric Sehrer

(Cedric.Sehrer@uni-konstanz.de)

Iris Schmidt

(Iris.Schmidt@uni-konstanz.de)

Dirk Oehler

(Dirk.2.Oehler@uni-konstanz.de)

Onur Cakmak

(Onur.Cakmak@uni-konstanz.de)

## 6.3 Special Thanks

Arno Scharmann

Dr.-Ing. Ernst de Ridder

Sigmar Papendick

Simone Winkler

Tino Klingebiel

Stefan Brütsch