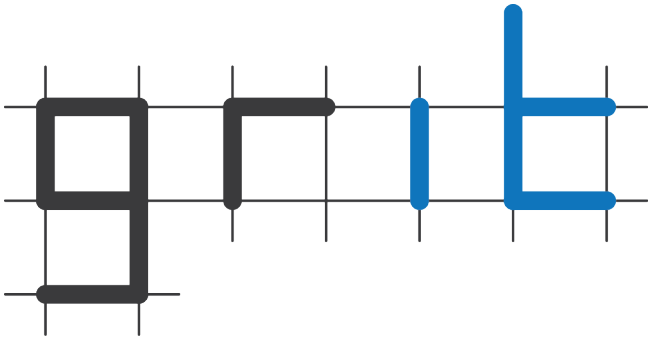
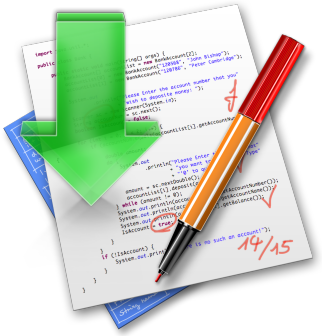


Using the Grit Submission System

Official User Manual for Version 1.0



July 8, 2014

Contents

1 What is Grit?

Grit is a system, developed by TEAM GRIT[1], that makes the life of lecturers and tutors of mainly programming courses easier and simpler. 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 and can 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: good programming.

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, Ilias 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 an by 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

[1] `https://team-grit.com`

summands. Furthermore Grit is not suitable to test if the submitted solutions of the students will dissolve the given problem or a part of it. Neither Grit can 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 recognized 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 self-dissolved solutions of a given problem statement and are tired of searching for syntax errors, compiling error, typing faults, etc. lasting for hours, then you will definitely need our product to increase the efficiency of your correction by prechecking every submitted solution by Grit automatically and therefore dull hours of automatable searching for compiling errors and similar faults will be omitted. 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 don't need Grit.

You won't 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?

- The Marmoset Project
  (`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`)
  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`)
  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.

2 | Getting Help

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

2.1 Our Website

On our website (`team-grit.com`), you will find links to latest builds of Grit as well as information on the new added features.

2.2 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.

2.3 Contact us

Do you have suggestions? Do you miss a feature? Contact us through our website `team-grit.com`.

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, which was not made public yet, but will be in the future.

4 Installing Grit

Grit is normally delivered as a virtual machine, thus, this chapter concentrates on running Grit via this VM. If you wish to run Grit natively or on another virtual machine, you will find the associated requirements in Section 3 of this chapter.
The advantage of using a pre-configured virtual machine is that Grit is able to run without making any changes. Only a minimum of configuration, for example setting up the network, is needed. All necessary components are already installed and Grit is ready to run.

4.1 Importing the Grit VM

How to import the Grit virtual machine depends on the used virtualization packages. This section will only cover VirtualBox Version 4.0.x since it is one of the most commonly used virtualization programs and is (mostly) free[1]. Still, the settings made for VirtualBox also apply if you use other applications, though details may differ.
First, the installation via command-line interface is encouraged since you will most likely run the VM on a server without any graphical user interface. After that, a short guide for the VirtualBox graphical interface is given.
Importing and applying the settings via the graphical interface might be a lot easier since you directly get to the VM login screen and don't have to login via `ssh` which might be not possible if the network setting aren't correct, for which we can't take care of.

4.1.1 Import into VirtualBox via Command-line

The Grit virtual machine comes in the Open Virtualization Format (OVF). First off, the VM needs to be imported using

```
$ VBoxManage import GritVM.ovf
```

This command imports the virtual machine with all its optimal settings and you are ready to start it using

[1] `https://www.gnu.org/philosophy/free-sw.html`

```
$ VBoxManage startvm GritVM --type headless
```

4.1.2 Import into VirtualBox via graphical Interface

To import the OVF file into VirtualBox using the graphical interface, go to "`File > Import Appliance`" and choose the OVF file downloaded previously. Next, launch the VM either by double-clicking on it or selecting "`Start`" on the menu bar.

4.2 First Startup

After the first boot, assuming you started the VM via command-line, it is necessary to log into the VM via `ssh`. If you used the graphical interface, you can skip this first part on how to obtain the VMs IP and immediately read about how to setup the static IP since you are able to just login via the graphical interface.

4.2.1 Get the current dynamic IP of our VM

The IP address is required for this task. Since by the VM is using a dynamic IP address by default (you will change that soon), you will not know this address beforehand. To find out, execute the following command on the host machine:

```
$ VBoxManage guestcontrol GritVM exec --image /sbin/ifconfig
--username grader --passwort grit!securedThisUser --wait-stdout
```

This executes the `ifconfig` command as the standard user called `grader` with the standard password `grit!securedThisUser` and waits for the command output to get printed. The output will look similar to the following:

```
eth0      Link encap:Ethernet  HWaddr 08:00:27:7c:36:06
inet addr:192.168.0.42  Bcast:192.168.255.255  Mask:255.255.0.0
inet6 addr: 2a02:8071:8385:2601:a00:27ff:fe7c:3606/64 Scope:↩
  ↪ Global
inet6 addr: fe80::a00:27ff:fe7c:3606/64 Scope:Link
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:47 errors:0 dropped:0 overruns:0 frame:0
TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:6212 (6.0 KiB)  TX bytes:3482 (3.4 KiB)

lo        Link encap:Local Loopback
inet addr:127.0.0.1  Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING  MTU:16436  Metric:1
RX packets:16 errors:0 dropped:0 overruns:0 frame:0
TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:1280 (1.2 KiB)  TX bytes:1280 (1.2 KiB)
```

```
```

You are concerned with the `eth0` interface. In this output, right after the entry `inet addr`, you can see that the VM is currently using the IP `192.168.0.42`. Now, you can log into this VM from a computer in the same network (or any computer if you already set up forwarding, which is not discussed in this document) using `ssh`:

```
$ ssh grader@192.168.0.42 # internal or external ip
```

4.2.2 Basic Security

Now that you are logged in, it is highly recommended to change the user and root passwords. As you may have noticed, the password for the user `grader` is `grit!securedThisUser`. The `root` password is `root`. To change these, type the following:

```
$ passwd # change password of logged in user (i.e. grader)
$ su # login as root
% *\#*) passwd # change root password
```

If you want to add additional security to the `ssh` server, you may want to generate a `ssh` key and change the settings of the server so that you can only login via a previously added `ssh` key.
WARNING: If you lose that key you can NOT login via `ssh` anymore. You will need direct access to the machine to disable the option and/or add the newly generated key!
To do this you need to generate you key on you local computer first and then copy that key into the VM using:

```
$ ssh-keygen -t rsa # create a rsa ssh-key
$ ssh-copy-id grader@192.168.0.42 # copy your ssh-key into the ↩
  ↪ VM
```

To disable password login on the virtual machine so that you can only login using your `ssh`-key, you now need to login into the VM, edit `/etc/ssh/sshd_config` and change the `PasswordAuthentication` entry to

```
PasswordAuthentication no
```

This entry will already be there, however, it is commented out. You only have to remove the # to disable password logins.

4.2.3 Setup a static IP address

After your first login, you will have to setup the network according to your local setup and (probably) set a static IP. For that, you need to edit `/etc/network/interfaces`. By default, the file looks like this:

```
# This file describes the network interfaces available on your ↩
  ↪ system
# and how to activate them. For more information, see interfaces↩
  ↪ (5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0

iface eth0 inet dhcp

# iface eth0 inet static
#  address 192.168.0.42
#  netmask 255.255.0.0
#  gateway 192.168.0.1
```

The important part - again - is the specification of the `eth0` interface: Here, you first comment out where the `eth0` interface is set up with a dynamic address (`dhcp`) and instead use the pre-outcommented template below that. Then assign a static IP address, set the net mask and tell the VM what IP address your router has.
After that, you need to reboot the VM by typing

```
$ sudo shutdown -r now
```

and you are done setting up the VM and are ready to launch Grit.

4.3 Infos about the VM

In this section, information is given about the VM Grit is shipped with and how you have to configure your own system if you chose not to use the VM provided. Also, the configuration already predefined for running the provided VM via VirtualBox, for example which network setting to use, are given.

4.3.1 The System

The provided virtual machine is running Debian Wheezy 7.5 64bit. It is a widely used, stable, performant, widely supported operation system, and very slim nevertheless. We did not provided a graphical interface since all configuration can be easily done via ssh. Furthermore, this saves memory and disk space. Tools like `sudo` and `apt-get` are already installed so that you are able to install additional tools and updates. The system settins made in the VM configuration set memory to 1GB of memory, 12MB of video memory, 8GB of virtual disk space and a bridged network adapter (e.g. Intel PRO/1000 MT Desktop).

4.3.2 Required Software

Grit depends on several packaged to be installed on the system for standard functionality. These are:

- OpenJDK 7 (`openjdk-7-jre`)
- TeXLive (`texlive-full`)
- GNU C Compiler (`gcc`)
  Note: You have to ensure, that gcc is part of the system's PATH variable. This only concerns systems running Windows.
- Glasgow Haskell Compiler (`ghc6`)

All of the above packages resemble *latest* releases. These might not be available through `apt-get` and need to be installed directly, this is, for example, the case for TeXLive.

4.4 Ilias Virtual Machine

5 | Running Grit

6 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?*

   As the provided virtual machine is very light-weight, any computer with constant internet access and at least 1GB memory for the VM does the job. However, we encourage you to use a dedicated server machine and not your personal computer to run the Grit back-end.

4. *Do I need a virtual machine for Grit?*

   It is possible to install the Grit back-end without any virtualization software or virtual machine. However, we strongly suggest using a virtual machine due to security concerns. Furthermore, we only provided comprehensive guides for using the virtual machine, this, if you wish to go without one, you are on your own.

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

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

6. *What is the* `root` *password?*

   The `root` password is `root`

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

   Yes.

8. *Is it possible to use a graphical user interface?*

   If you wish to run Grit natively or on another virtual machine, you will find the associated requirements in Section 3 of Chapter 4.

7 Credits

7.1 Developers

Gabriel Einsdorf

Marvin Gülzow

Eike Heinz

Marcel Hiller

David Kolb

Fabian Marquart

Thomas Schmidt

Stefano Woerner

7.2 Special Thanks

Simone *Last name*

Tino Klingebiel