

Hands on Version Control with Git

submission deadline: Oct. 20th 2017 11:59pm

In this sheet you get familiar with basic functionality of source version controlling with Git. This is a per-student exercise sheet, i.e., you are not allowed to submit a group solution. You must submit your solution in time via a pull request (see task 10) to the following Git repository:

<https://github.com/Arcade-Lecture/exercises.git>.

Prepare to present details of your solution during the tutorial.

Good Luck!

Exercise sheet No. 1

After working on this sheet you will have learned the following

- ✓ Setup Git and GitHub (via SSH)
- ✓ Fork repositories
- ✓ Clone your remote repository
- ✓ Basic operations on branches
- ✓ File checkout between branches
- ✓ Committing changes locally/remote
- ✓ Opening Pull Request
- ✓ Endurance for longer reads on tasks

You want to join an exiting new project. As it turns out, your project of interest is managed with the distributed version control system Git. Your project of interest is licensed as an Open Source project, and provides a public repository <https://github.com/Arcade-Lecture/exercises.git>. As stated by the maintainer of the project, contribution is welcome but public write access to the repository is not given. However, you are welcome to *fork* repository which means that you can create a personal copy on your own server. You are further welcome to notify the maintainer of your project of interest on changes that you did in your fork in order to integrate these into the official main code base.

Task 1 Choose an Environment

0.5 Point

The choice for a certain operating system and toolchain is sometimes a question by its own. However, due to the tight integration of Git (and the C language later on) into Unix-like operating systems and the powerful command line (bash) of Unix-like operating systems, it is recommended to use Linux, macOS or similar. The lecture will not cover a Windows port.

In case you choose Windows as your platform anyway, consider to install Cygwin (<https://www.cygwin.com>) that provides a lot of GNU and Open Source tools as well as several POSIX API functions for your Windows machine. Alternatively, feel free to run any Linux distribution (like Ubuntu, <https://www.ubuntu.com>) in a virtualized environment (e.g., by the use of VirtualBox, <https://www.virtualbox.org>)

Task 2 Install & Setup Git on your Machine

1 Point

To get started, you need to setup your environment in order to use Git. Check whether Git is installed on your development machine by typing the following into your bash terminal:

```
$ git --version
```

If you get an output notifying you on the version (e.g., `git version 2.7.4 (Apple Git-66)`), you have Git already installed on your system. Otherwise you get a message telling you that no command can be found (e.g., `-bash: git2: command not found`).

In case Git is not installed on your machine, install it with the following commands (see below). After installation, verify a successful installation by repeating the first steps of this task.

Git installation on Debian-based Linux distributions (like Ubuntu)

Type the following command into your bash

```
$ sudo apt-get install git-all
```

This will install Git by the use of the apt-get package manager. Alternatively, you visit <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git> for other installation options.

Git installation on macOS

Go to <http://git-scm.com/download/mac> and download the binary installer. Alternatively, feel free to install a package manager on your Mac, e.g., Homebrew. To install homebrew, type the following into your bash

```
$ /usr/bin/ruby -e "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

When Homebrew is installed successfully, install Git by typing the following into your bash

```
$ brew install git
```

When Git is installed, it is recommended that you add some specific information about you as developer for any action that you perform repositories. At least your name and a contact e-mail address should be provided. Therefore, type the following in your bash

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

Task 3 An Account at GitHub

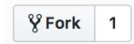
0.5 Points

GitHub is a version control internet hosting service on which your project of interest is hosted. You decide to use the built-in fork capabilities of GitHub to copy the official repository on your

own server (i.e., into your own GitHub account). For that purpose, you create a (free) account on GitHub by visiting <https://github.com>. After sign-in and log-in, you fork your repository of interest.

Task 4 Forking the Official Repository**0.5 Points**

You go to the repository URL of your project of interest, <https://github.com/Arcade-Lecture/exercises>. Afterwards you fork the repository by clicking the button for that action



Once the forking process is successfully completed, you see your personal copy of the repository in your account overview.

Task 5 Create SSH for your GitHub Account**1 Points**

You can use the SSH protocol to connect to remote services such as GitHub. SSH keys allow you to connect to GitHub without providing your username or password each time you pull or push changes.

You start with creation of a new SSH key (if not already present) by typing the following in your bash:

```
$ ssh-keygen -t rsa -b 4096 -C "<your e-mail address you used in GitHub>"
```

In case you are requested to enter a file name to store your key, press enter to accept the default location.

You are then prompted to enter a passphrase. A passphrase is similar to a password but used to derive an encryption key for SSH (i.e., to encrypt the private key). Choose a good passphrase that has at 15 to 20 characters and which is hard to guess. For more information, see here <https://www.ssh.com/ssh/passphrase>. Remember your passphrase.

Afterwards, add your private key to the ssh-agent and store your passphrase your keychain by typing:

```
$ ssh-add -K ~/.ssh/id_rsa
```

In case you used not the default location to store your key, you will have to change to location to your key by changing the path for the command `$ ssh-add -K`.

Now it's time to add your SSH key to your GitHub Account. For that purpose, copy the content of `~/.ssh/id_rsa.pub` into your clipboard in order to paste it into your GitHub settings. To copy the file contents on Linux, type

```
$ xclip -sel clip < ~/.ssh/id_rsa.pub
```

Note: On macOS you can type `$ pbcopy < ~/.ssh/id_rsa.pub` instead, or install xclip with Homebrew first. Alternatively, open `~/.ssh/id_rsa.pub` in a plain-text editor and copy the content to your clipboard manually.

Afterwards, open <https://github.com> in your browser and go to your account settings by clicking on “Settings” in the menu of your avatar in the top menu of the GitHub GUI. Navigate to “SSH and GPG keys” and click on “New (Add) SSH Key”. Add a descriptive title such as “My Student Developer Machine” and paste your SSH key into the text box below.

Task 6 Your local Repository

1 Points

In order to work on your repository, you must (partially) copy your newly created repository on your machine (“cloning”). Therefore, in your bash you change to a directory of your choice in which you will clone your repository. Then, type the following into your bash

```
$ git clone git@github.com:<your GitHub account name>/exercises.git
```

After success, type the following into your bash

```
$ cd exercises
```

to navigate into your repository root directory.

After a successful clone process, you type

```
$ git remote add mainline https://github.com/Arcade-Lecture/exercises.git
```

into your bash to add a new alias called mainline pointing to the official repository. With this, you can receive updates from <https://github.com/Arcade-Lecture/exercises> over the alias mainline while you can push/pull to your repository [git@github.com:<your github user name>/exercises.git](https://github.com/<your github user name>/exercises.git).

The repository contains several directories that are reserved for follow-up exercise sheets and material. However, the root directory contains a file called `hi_there`. Your task is to view that files content, and then copy it to a specific location. To view the content, type

```
$ cat hi_there
```

If you are happy with the result, you can proceed with exploring and modifying your repository. For the purpose of exploration, type

```
$ git branch -a
```

You will receive a list of all branches that are available (including remote ones that you not yet fetched).

Now it's time to checkout your first remote branch.

Type the following into your bash

```
$ git checkout -b sheet_01_material origin/sheet_01_material
```

Git will perform a download of the remote branch `sheet_01_material` (located at origin that points to your remote repository) and a creation of a new local branch `sheet_01_material`. Ensure, that you still are in the root directory of your project (`exercises/`) take a look at the file `rick`:

```
$ cat rick
```

Note that the file `rick` only exists in the branch `sheet_01_material` but not in the `master` branch from which you originally came from. However, you want to copy that file along with other material to your final submission. Therefore, create a new dedicated local branch for your new changes and submission. Type the following:

```
$ git checkout -b sheet_01_submission master
```

to create a new branch `sheet_01_submission` that branches from the `master` branch. Verify that the `rick` file does not exist in `master`. Type

```
$ ls -l
```

in your bash to see a list of the current directory's content. You should not see the file `rick`. Your task is to copy the file `rick` from the branch `sheet_01_material` into your new branch `sheet_01_submission`. For this, type

```
$ git checkout sheet_01_material rick && ls -l
```

You now should see the file `rick` as part of the directory. Run

```
$ git status
```

to verify that the file `rick` is staged in the index but marked as "new" (i.e., not committed). You want to move both the `rick` file and the `hi_there` file into a dedicated submission directory. Therefore, create a directory `<your last name>` below the directory `sheet_01/`, copy `hi_there` in that directory, and move `rick` into that directory. For this, type the following

```
$ mkdir sheet_01/<your last name>
$ cp hi_there sheet_01/<your last name>
$ mv rick sheet_01/<your last name>
```

After you moved the file into your newly created directory, you remember that changes to your file systems do not automatically imply a change into your repository. Therefore, you type the following into your bash

```
$ git status
```

This command will produce an output similar to this:

```
On branch sheet_01_submission
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   rick

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    deleted:    rick

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    sheet_01/<your last name>/
```

Note while `rick` is already tracked due to the checkout from another branch, everything below `sheet_01/<your last name>/` is not yet tracked which means not considered by Git since you created it in your file system without commanding Git to stage it. You will change this now.

Add the file to your index by typing the following into your bash:

```
$ git add sheet_01/<your last name>/
```

to command Git to remove the `rick` file from its original position, run

```
$ git rm rick
```

You verify that your submission directory `<your last name>` is added to the index by typing

```
$ git status
```

once again. In contrast to the first call, Git will output a message similar to this:

```
On branch sheet_01_submission
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
```

```
new file:   sheet_01/<your last name>/hi_there
new file:   sheet_01/<your last name>/rick
```

Task 8 **Commit your Changes**

0.5 Points

Now it's time to commit your changes via `git commit`. By default, a terminal-based editor like `vim` will start to accept your commit message. If you feel uncomfortable with `vim`, consider to change the default editor for Git first by typing

```
git config --global core.editor "<alternative>"
```

where `<alternative>` is the name of your alternative editor of choice, e.g., `nano`. However, commit your changes by typing

```
$ git commit
```

into your bash. A text editor will open to receive your commit message. For the current purpose, it's enough to add a single-line entry explaining the reader what your commit will do (e.g., "*Add submission*"). Alternatively, you can use `$ git commit -m "<message>"` for a one-liner commit message `<message>` without the need for an additional editor. Example

```
$ git commit -m "Add submission"
```

Task 9 **Push into *Your* Repository**

1 Points

It's time to end this longish exercise sheet. Push your branch `sheet_01_submission` to your remote repository by typing the following into your bash

```
$ git push origin sheet_01_submission
```


Open <https://github.com> in your browser, and navigate to your forked repository. If everything went smoothly, you will see the branch `sheet_01_submission` that was added just a few seconds before to your remote GitHub repository. Switch to your new branch in the GitHub viewer and verify that

(1) the `rick` file is not stored in the root:

Branch: `sheet_01_submi...`
New pull request

Create new file
Upload files
Find file
Clone or download


This branch is 1 commit ahead of Arcade-Lecture:master. [Pull request](#) [Compare](#)

 **pinnecke** Add submission Latest commit `c8a446c` 5 minutes ago

sheet_01	Add submission	5 minutes ago
sheet_02	Init	6 hours ago
sheet_03	Init	6 hours ago
sheet_04	Init	6 hours ago
sheet_05	Init	6 hours ago
README.md	Init	6 hours ago
hi_there	no message	2 hours ago

(2) That the directory `sheet_01/` contains a sub directory with your name, and

This branch is 1 commit ahead of Arcade-Lecture:master. [Pull request](#) [Compare](#)


 **pinnecke** Add submission Latest commit `c8a446c` 7 minutes ago

..

pinnecke	Add submission	7 minutes ago
README.md	Init	6 hours ago

(3) The sub directory contains both files `hi_there` and `rick`

This branch is 1 commit ahead of Arcade-Lecture:master. [Pull request](#) [Compare](#)

 **pinnecke** Add submission Latest commit `c8a446c` 8 minutes ago

..

hi_there	Add submission	8 minutes ago
rick	Add submission	8 minutes ago

Task 10 Submit via a Pull Request

5.0 Points

Open <https://github.com/Arcade-Lecture/exercises/pulls> in your browser and create a new pull request “across forks” from base fork: `Arcade-Lecture/exercises` base: `sheet_01_material` to head fork: `<your GitHub Account Name>/exercises` base: `sheet_01_submission`.

This pull request notifies us that you have changes that we can import into our main code base, i.e., you made your exercise submission.

For more information on pull requests in GitHub, follow these instruction <https://help.github.com/articles/creating-a-pull-request-from-a-fork/> to create a pull request in the official remote repository.