HOWTO Use the repository (Git)

Related courses: Robotica, PAS1 and PAS2

Team BORG

February 20, 2013

1 Introduction

This document introduces your to the BORG repository. Here we assume you are using a UNIX-like system (like GNU/Linux).

For more information on (distributed) revision control:

1. `http://en.wikipedia.org/wiki/Revision_control`
2. `http://en.wikipedia.org/wiki/Distributed_revision_control`

2 Git

Since November 2012, the BORG team uses Git as the main distributed revision control system.

For more information, see: `http://git-scm.com/`

3 Getting Access

In order to clone the repository (e.d. getting the actual data), you first need to acquire access to it. Follow the next instructions to acquire access.

1. If you haven't already, generate a RSA key (no need to enter a passphrase if asked to do so):

   ```
   # ssh-keygen -t rsa
   ```

   Use the default options (just hit enter everytime) and make sure to put the private key (`id_rsa`) and public key (`id_rsa.pub`) under the `~/.ssh` directory. Other systems can use your public key to encrypt data and you should use your own private key to decrypt the encrypted data. The reverse holds for the other end, this is all done automatically. *Do not publish your private key, keep it safe!*

2. Now *copy* your public rsa key (located under `~/.ssh/id_rsa.pub`), rename the *copied* public key to something like `<yourname>.pub` and send it to the repository administrator. Leave the original private and public keys as produced by `ssh-keygen` untouched in `~/.ssh/`.

3. Copy the public and private key to any of your accounts, or send the extra keys that you want to be added as well to the repository administrator (please rename them to something like `<yourname_2>.pub`.

4. Also, specify your name and email address in your `~/.gitconf` file in your home directory (create it if it does not exist), example:

   ```
   [user]
       name = Sint Nicolaas
       email = sint@kasteel.es
   ```

4 Cloning the repository

In order use the repository, you need to first clone it. To do so, execute the following command:

```
# git clone robocup@login.ai.rug.nl:sudo
```

5 Working with the repository

Once you have access to the repository, you can:

1. Pull and update changes from others:

   ```
   # cd <repository directory>
   # git pull
   ```

2. Add and remove files or directories:

   ```
   # git add <some_file_or_dir>
   # git rm <some_file_or_dir>
   ```

3. Git does not allow to add empty directories. If you really want to add a directory, you need to add a placeholder, for example:

   ```
   # mkdir somedir
   # touch somedir/placeholder
   # git add somedir/placeholder
   ```

4. Before commiting your changes, you need to mark the files first that need to be commited (included in the change):

   ```
   # git add <some_file_or_dir>
   ```

5. To add all files *that are and modified* in the underlying directory, you can do the following:

   ```
   # git add -u .
   ```

6. Commit changes of your own (once you changed, removed or edited one or more files) (no need to have network access):

   ```
   # git commit -m "Some␣small␣summary␣of␣what␣you␣changed."
   ```

7. And finally push the changes to the central repository[1]:

   ```
   # git push
   ```

8. In many cases you will need to "merge" two or more versions (for example if two persons made changes to the same file or if your own copy is out of sync) in order to incorporate all changes using the "merge" command:

   ```
   # git merge
   ```

9. You can lookup changes to the repository by using the log command, for example:

   ```
   # git log
   ```

[1] This is only possible if you have write access to the repository. Git is a distributed versioning system which means you can also push the repository to locations other than the one you pulled from. You can for example push changes to or pull changes from your teammembers directly (e.d. `git push <location>` or `git pull <location>`).

6 Guidelines

Please adhere the following guidelines while working with the repository:

1. Before committing files, make sure to first observe the changes you made using the "status" command:

   ```
   # git status
   ```

2. For more in depth differences of files:

   ```
   # git diff
   ```

3. Before pushing any changes to the repository, make sure to run the testrunner first and verify that all tests succeed (this will ensure that you did not break any essential systems while modifying the code):

   ```
   # cd brain/src
   # python testrunner.py
   ```

   Please report tests that fail by others on the mailinglist.

4. Try to pull and push as many times as possible during the day. This will ensure that you keep in sync with others.

5. *If you are doing experimental stuff that involves the core systems; create a new branch.*

6. *If you do not know what you are doing; do not push!*

7. *(Old) code that is not working, not documented, poorly written or has no unit tests, will be removed in time!*

7 More information

For more information about Git and it commands (the command is optional), type:

```
# git help <command>
```

or visit; `http://git-scm.com/`