Tasks and Assignments Course 02267

Hubert Baumeister

Version 1

1 Task (group): Continuous integration with Jenkins TASK_JENKINS

1.1 (group) Install Jenkins on the VM

Install Jenkins on the VM. You can follow the instructions here: `https://www.jenkins.io/doc/book/installing/linux/#debianubuntu`

Note that Jenkins requires to have a JDK installed.

Jenkins should run on port 8282 and not 8080. Port 8080 should be kept unused, so that we can later have Web services using port 8080.

- Once you have installed Jenkins using `sudo apt-install jenkins`, you can change the port Jenkins by using `sudo systemctl edit jenkins` and add the following as described in the instructions on the Web site

```
[Service]
Environment="JENKINS_PORT=8282"
```

- Restart jenkins on the new port using `sudo systemctl restart jenkins`.
- Then go to `http://<your-VM>:8282` and follow the instructions.
  - In the step, where you are asked about plugins to install, you should use the suggested plugins.
  - For our purposes it is okay to use the built-in node for running builds. So you should dismiss the respective security warning.

If you get some spurious Jenkins error messages, you may also need to install fontconfig via `sudo apt install fontconfig`

For Jenkins to work together with GitLab, one needs the GitLab plugin. This is installed through `Manage Jenkins::Manage Plugins`. This will allow GitLab to trigger Jenkins builds.

1.2 (single) Create a small Java program and let it build and run the tests via Jenkins

Everybody in the group should create a small Java program of your choice together with a test for that program, maybe just calling a function and checking that the return value is "hellow world" or calling a function that does some computation and check that the computation is correct, e.g., `assertEquals(7,calc.add(3,4))`. You want to create a small program with tests that you can build and execute using `mvn test`.

You can also use the Kata you have developed in the TDD exercise.

Everybody in the group should create a git repository for the program (git repositories are cheap, easy to create and easy to delete again :-) and create a Jenkins project pulling the source code from the git repository, building the Java programm and running the tests each time the source code is pushed to the git repository.

Here is a video[fn:±9] with instructions on how to create a Jenkins project. Note that the `./build_and_test.sh` shell script just executes `mvn test`. Although for more complex programs, more steps need to be done in the `./build_and_run.sh` shell script.

I suggest you use `gitlab.gbar.dtu.dk` for your git repositories. Any other git repository is fine, but for the instructions to work it should be a GitLab server. Other repositories have other kind of Webhooks, like GitHub which also requires different plugins. Finally, it also works if you let Jenkins poll every 10min for changes in your repository (this works for all kinds of git repositories).

1.3 A tip for debugging Jenkins projects

As shown in the video, on the VM, you can easily pose as the user jenkins by executing `sudo su -l jenkins`. Then you go to `cd workspace` and then to the project directory. Here you can run your build, deploy, and test scripts locally and see what is wrong to compared with what happens on your computer.

1.4 Create an Andon board

Create an Andon board to publicly show the status of your project using the Build Monitor View plugin of Jenkins.

1.5 Jenkins and Docker

If you have installed Docker before Jenkins, remember to add the user ‘jenkins‘ to the ‘docker‘ group to allow running docker commands without being superuser. Follow the respective instructions in the task for installing Docker on the Linux VM.