# Jenkins SIG – hands-on

### Needed for this hands-on

* Docker
* Docker Compose (already included with docker desktop)
* Git

## Setup Jenkins

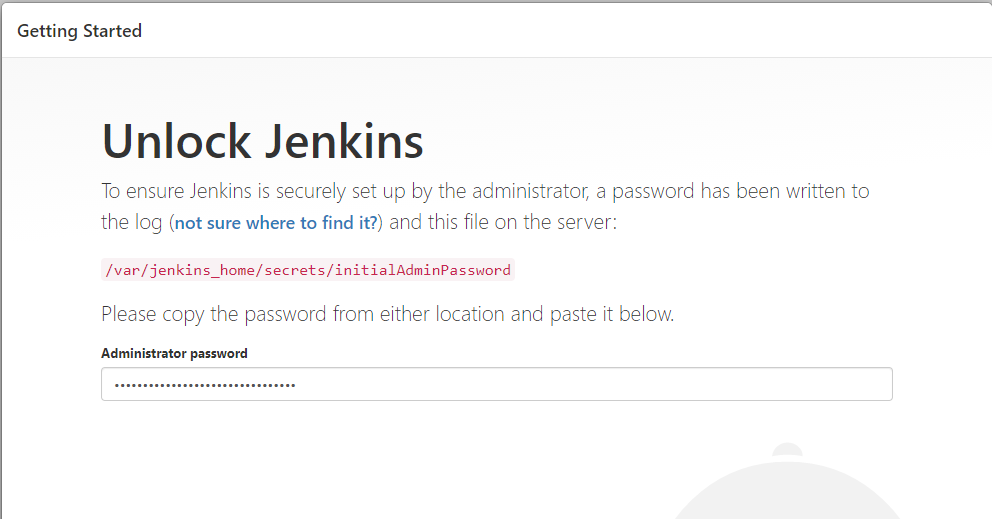
In this hands-on we will run Jenkins from docker. If you don’t want to use docker you can install Jenkins itself on your computer, but some if the provided solutions might not be correct then.

In the docker-compose.yml file everything needed to run Jenkins from docker is specified. Before running this file, you need to make one adjustment to the volumes section of the Jenkins-agent. The last volume maps the Jenkins home directory to a directory on your system. Replace C:/Users/EHERMAN91 with your own home directory.

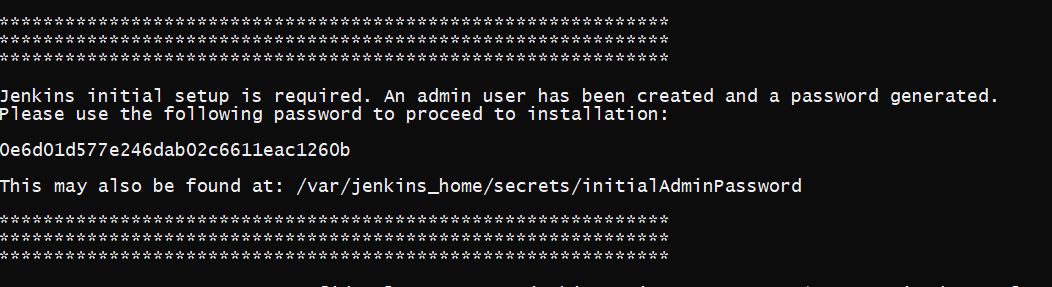
Run the following command to start everything up: “docker compose up” or “docker compose up -d” if you don’t want to see the output in the command line (-d means detached). This command needs to be run from the folder in which the docker-compose.yml file is placed.

These docker compose arranges a couple of things. It builds and runs the Jenkins docker container from the Dockerfile. In this dockerfile a couple of commands are run that add chrome to the container. Chrome is needed for running the angular tests in some of the pipelines in this hand-on. It also runs docker in a docker container and connects it to a network. Through this network the Jenkins container can access the docker container and use docker in the pipelines. As specified above, it maps the Jenkins home to a directory on your local machine. By doing this we can access local git repositories from Jenkins.

Once everything is running, go to localhost:8080, you will see a screen asking for the Administrator password.



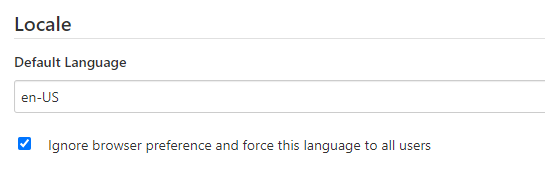
Run “docker logs jenkins-agent” to get the logging from the Jenkins container if you chose to run the detached command, otherwise the logging is already visible. The password will be printed here (between the \*\*).



Select “Install suggested plugins”. After the plugins are installed you need to create the first user. Remember the username and password, you will need this to sign in if the Jenkins docker container is restarted.

Note: the language which Jenkins displays depends on your browser language. If you want to change this language you can download the “locale” plugin:

* Go to “Manage Jenkins” > “Manage Plugins”.
* Click the “Available” tab.
* Search for “locale” and install without restart (restart is not necessary for this plugin)
* In “Manage Jenkins” > “Configure System” a locale options should now be available. Enter your preferred locale, for example “en-US”.
* Click the “Ignore browser preference ..” checkbox if you don’t want to use your browsers preferred language.



## Manually setup project/ pipeline

#### Assignment 1

For this assignment we will be creating a freestyle project. We will not actually build an application yet, but we will try some of the options Jenkins has to offer.

The job you will be creating has the following requirements:

* The workspace should be deleted before the build starts
* The build should automatically run every minute
* No more than three builds should be kept
* The build should print a greeting to the person (print the name) that started the build when run manually, or a greeting to “user” when run automatically. The user should be able to provide their name when starting the build.
* The preferred programming language of the user should be asked when starting the build. They should be able to pick from three choices. This preference should also be printed when the job is run. When running automatically, a default should be printed.
* A timestamp should be shown for each action

Try to create this freestyle project by yourself at first, if you get stuck some of the hints below might help. Check if the output of your project is as you expect when running manually and automatically, and that only 3 builds are saved. Don’t spend too long on this assignment, this is only meant to get acquainted with Jenkins.

#### Hints:

* Check the “Delete workspace before build starts” checkbox to delete the workspace.
* Create a new “Freestyle” project by clicking on “New item”. Enter a name for you project and click on “Freestyle project”.
* Check the “Build periodically” checkbox to specify a build schedule. \* \* \* \* \* means build every minute. If you click the “?” an explanation of the syntax is given.
* Check the “Discard old builds” checkbox to specify the number of builds to keep.
* Check the “This project is parameterized” checkbox to specify parameters for the build, that the user should fill. Choose “String Parameter” for the name of the user and “Choice Parameter” for the language.
* Choose “Add build step” > “Execute shell” and use the “echo” command to print the greeting and preferred language.
* In the dashboard, click on your project. The build history should be shown in the bottom left of the screen. Click one of the builds and then “Console Output” in the menu to see the output.
* Click “Build with Parameters” to do a manual build.
* Click the “Configure” option in the project to adjust the build.

Click the “Disable this project” checkbox to prevent the build from continuously running when you are done with the assignment

#### Assignment 2

Build a pipeline with some basic steps for the koa-angular-app located at <https://github.com/EmmyHermans/Jenkins-SIG/tree/main/koa-angular-app>. Do not use the Jenkinsfile in the app, but instead use the “pipeline script” option. In this script field you can put a declarative pipeline script. The pipeline should take care of the following requirements:

* It should check the npm version that will be used.
* It should install the dependencies for the frontend and backend of the app.
* It should run the tests for the frontend and backend.
* It should clean the workspace at the end.

#### Hints:

* There are multiple ways to use node.js from within the pipeline; install a node.js plugin in Jenkins or use a docker image.
  + If you use a node.js plugin, this needs to be configured in the “Global Tool Configuration” in “Manage Jenkins”.
  + A docker image can be listed in the agent section in a docker subsection. This image needs to include chrome, since this is needed for the frontend tests. You will need to install the docker pipeline plugin to be able to use the docker agent.
* The git repository has a “main” branch instead of a “master” branch. You need to check out this main branch in one of the stages.
* Be sure to execute the installation and test commands in the correct subfolders of the git repository.
* Since the Jenkins docker container runs linux, you can use sh in the Jenkinsfile to run commands.
* Installation can be done with “npm ci”.
* The backend tests can be run with “npm test”.
* Run the frontend tests with “npm run test:ci”. This ensures the tests are run in a headless browser.

## Jenkinsfile

#### Assignment 3

Use the Jenkins file located in <https://github.com/EmmyHermans/Jenkins-SIG/tree/main/koa-angular-app> to run a pipeline. The content of this Jenkinsfile is similar to what was put in the “pipeline script” in assignment 2.

#### Hints:

* Use “New Item” > “pipeline” to create the pipeline and select “Pipeline script from SCM” under the “Pipeline” header.
* Make sure you use the correct branch
* Make sure you provide the correct path to the Jenkinsfile, since this is not located at the root level

#### Assignment 4

Make a new job that runs every time something in the git repository changes for the main branch. This can be done by cloning <https://github.com/EmmyHermans/Jenkins-SIG> locally to your home-directory or by creating your own repository on github. You can just use the Jenkinsfile provided in the application as pipeline or create a whole new pipeline.

1. To do this locally you need to place the git repository inside the folder you mapped to the Jenkins home folder when starting the Jenkins docker container (this was done with: --volume <home-directory>:/home). The pipeline should run when a commit is done on the “main” branch locally. Linux and/or mac users could run into some user right issues here and will need to make sure the Jenkins user has reading rights to the home folder.
2. If you use your own repository on github, the pipeline should run when a change is pushed to the “main” branch of this repository.

#### Hints:

* The easiest way to do this is to use Poll SCM and poll for changes. You can enter this manually in the build trigger section of the pipeline.
* Use “New Item” > “pipeline” to create the pipeline and select “Pipeline script from SCM” under the “Pipeline” header.
* If you use a local repository, the repository url should start with “/home” and should be the path to your repository starting from the home folder you specified in the docker-compose.yml.
* If you use a git pull in your pipeline and you chose to use a local repository, make sure this git pull also points to your local repository.

#### Assignment 5

Add some extra steps/ options to JenkinsFile to:

* Discard old builds; after 5 days and keep max 5 builds. So instead of putting this in general options we will put this in the jenkinsfile.
* Make sure no builds are performed concurrently.
* Add the build polling to the Jenkinsfile instead of setting it manually in the build triggers section of the pipeline.
* Add a stage that is only performed for a specific build number (you can use the standard environment variable BUILD\_NUMBER for this) or thinks of another condition for which to perform a stage.
* Use (top level) environment variables for the path to the frontend directory and backend directory (koa-angular-app/angular-frontend and koa-angular-app/koa-backend) instead of writing the path in the different stages.
* Print a message to the logs only if the pipeline fails (you can test this by removing the x before the failing test in books.component.spec or movies.component.spec).
* Make sure the workspace is always cleaned, also if the pipeline fails at some point.

#### Bonus (addition to assignment 5)

You will need some knowledge of making and emptying folders in linux and moving files for this bonus assignment. You will build the frontend. Put the static files in a specific folder in the backend and start up the backend to serve the frontend folders and run the app.

* Build the frontend files and put them in a directory in the backend source: “../koa-backend/src/public”.
* Run the backend in prod mode to serve the static angular frontend; “npm run prod”.
* Write the process identification number (pid) of the backend process to a pid file.
* Use an input inside the steps section to wait for user input before moving on the next stage.
* Kill the process using the pid when the user has pressed “proceed” for the input.