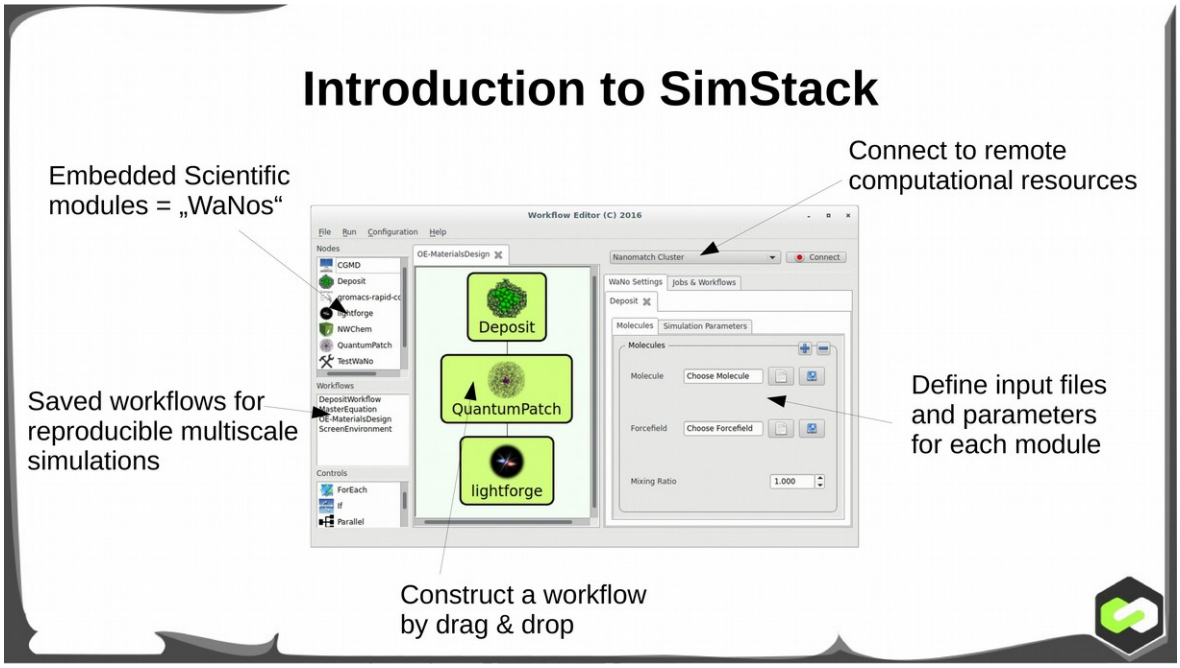
How to incorporate new modules into SimStack

The major benefit of SimStack is the rapid incorporation of software tools from any source. For any tool that can be executed via command, this can be achieved in a matter of minutes. Incorporating a new program into SimStack requires two steps:

1. The program needs to be installed/available at the remote (HPC) machine
2. A *Workflow active Node* (WaNo) for this specific program needs to be generated and added to the SimStack Client that is installed locally (on Laptops or Desktop PCs)

Using the SimStack client (picture below), WaNos are then combined into workflows by dragging them from the window on the top left into the middle pane, and automatically executed on HPC resources.



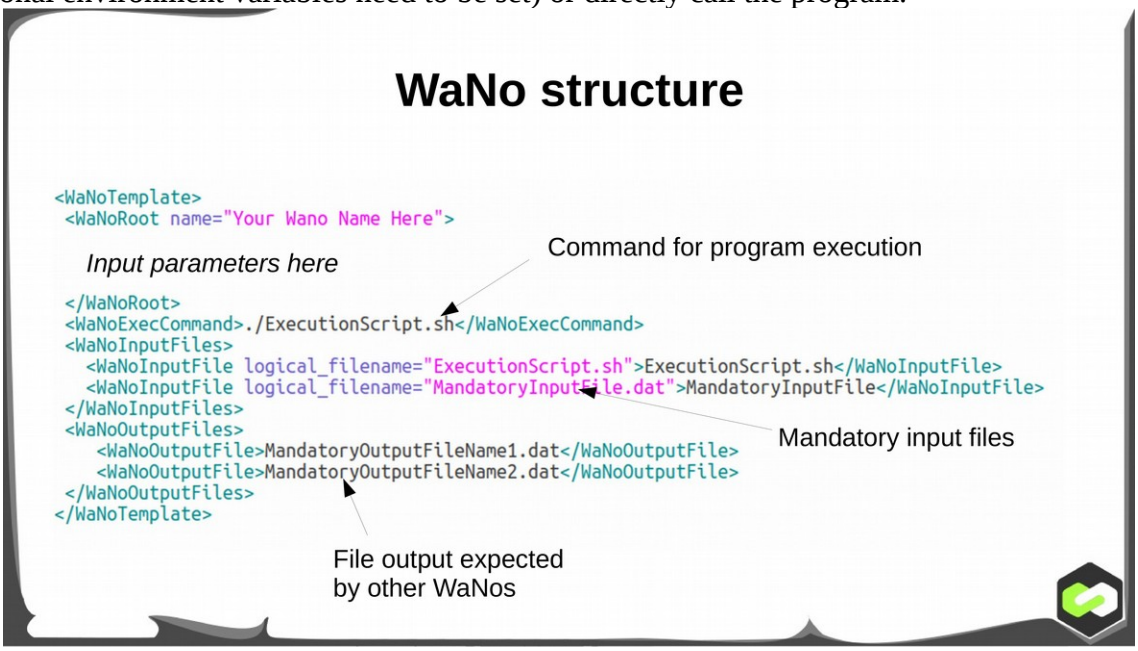
1. WaNos

1.1 The WaNo concept

WaNos are xml files that are interpreted by SimStack in two ways: First, the SimStack Client uses WaNos to automatically render a graphical user interface (GUI) for each tool, which is used to specify input parameters and files, and is displayed in the right half of the SimStack client. Second, the xml file defines parameters, input and output files of each tool and the execution command. This information is passed to the SimStack server on the remote machine when executing a workflow. WaNos are stored in the WaNo directory defined in the SimStack Client Setup.

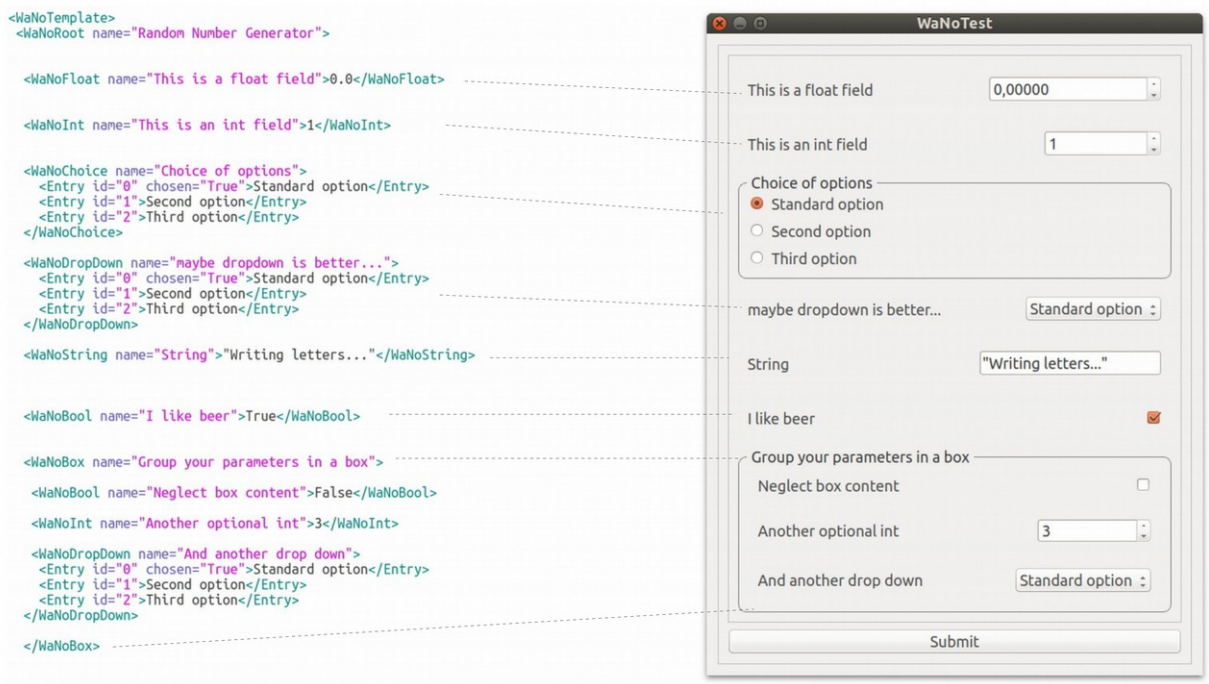
The general setup of a WaNo xml-file is depicted below. GUI elements and thereby also the program specific input parameters and files are specified within the <WaNoRoot> tags. The <WaNoInputFiles> section determines files that are necessary to execute the program (run script, mandatory input files) on the remote resources, and mandatory output is specified in the <WaNoOutputFiles> tag. In a workflow consisting of multiple modules, these mandatory output files can be specified as input in subsequent modules. During execution of the workflow, they are then automatically passed on between the modules. The execution command is defined in the

<WaNoExecCommand> tag. Here you can either call a run script (which is advantageous if additional environment variables need to be set) or directly call the program.



1.2 GUI elements

There are various GUI elements available to have input parameters specified by the end-user:



The name of the tags is used later to reference the parameters entered into the fields or the options of dropdown menues, and pass it on to the programs.

2. Example

Constructing a new WaNo is best illustrated using a simple example. In this case we want to include a python program into SimStack that plots an arbitrary number of Gaussian functions using

matplotlib, with mean and width specified by the user. Via the command line, the script is executed as follows:

python plot_gauss.py m0 s0 l0 m1 s1 l1 m2 s2 l2 …

Here mX, sX and lX are mean, width (sigma) and label of each gaussian, respectively. The program will plot as many gaussian functions as there are triples of arguments, into a plot called “gaussians.png”. It is best practice to execute programs like this on the remote resource using a shell script in order to set environment variables, e.g. the path to the program.

2.1 Starting a new WaNo project

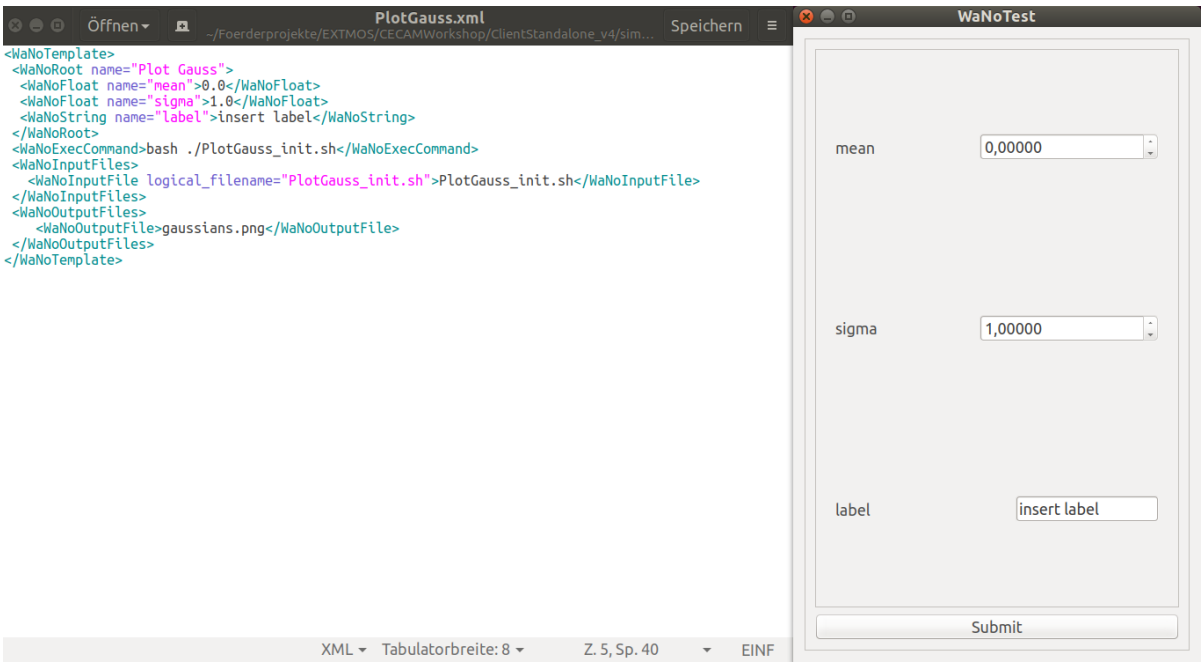
To incorporate a new tool, first set up a new subdirectory in the WaNo directory of your SimStack client (installed on your laptop). As a standard this WaNo directory is just the “wano” directory in the parent directory of the SimStack client. This is the point where you give your WaNo a unique name. In our example, we chose “PlotGauss”. Generate the directory “PlotGauss” in the WaNo directory, cd into …/wanos/PlotGauss/ and open a new file called “PlotGauss.xml”. In this xml file we will later specify the GUI, parameters and execution command of the new program.

2.2 Single Gaussian function

For a single Gaussian with mean 0.0 and sigma 1.0, the execution command (or rather the execution script) looks like this:

```
#!/bin/bash
python $NANOMATCH/PlotGauss/plot_gauss.py 0.0 1.0 StandardGauss
```

A WaNo for this simple single Gaussian plot would need three fields: two floats for mean and width and one string for the label. Furthermore, required input files are only the run script, the required output file is the “gaussians.png”. We include this in the XML file, save the XML file in the WaNo directory (.../wanos/PlotGauss/) and get the following WaNo:



Here, PlotGauss_init.sh is the execution script that is uploaded to the remote resource when the workflow is submitted. This execution script needs to be placed in the same folder as the WaNo xml file. In order to pass the parameters to the python program, we modify the run script as follows:

```
#!/bin/bash
python $NANOMATCH/PlotGauss/plot_gauss.py \
                    {{ wano["mean"] }} \
                    {{ wano["sigma"] }} \
                    {{ wano["label"] }}
```

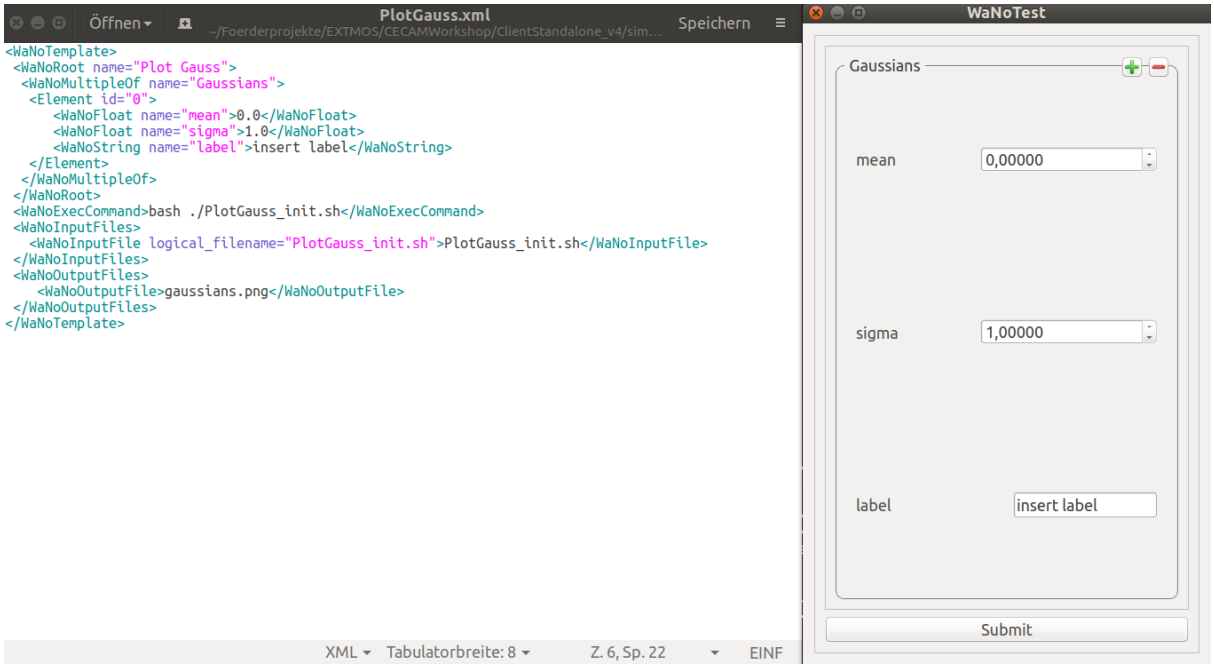
Note that “mean”, “sigma” and “label” were used as names in the XML-file (see above) and also appear as names in the rendered GUI for our program. This illustrates how variables that are set by the end-user in the gui are referenced in the submission script – via the brackets “{{ wano[VariableName] }}” including the name that was used in the WaNo xml file.

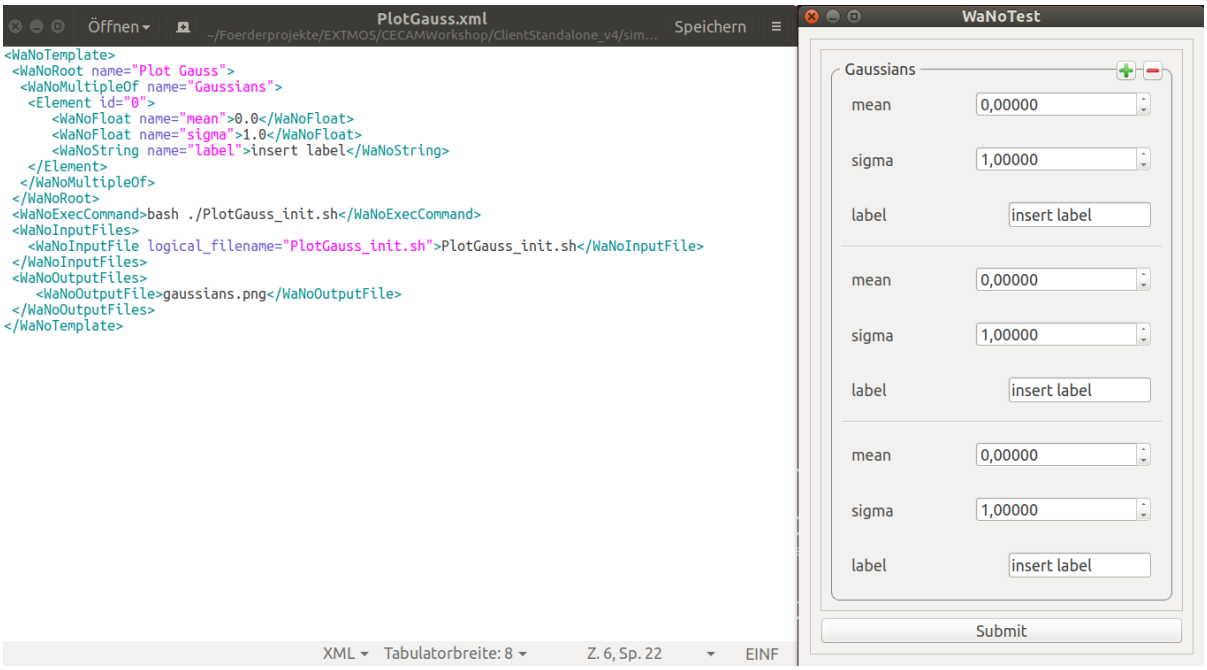
2.3 Plotting multiple Gaussians

Considering the capabilities of the program plot_gauss.py, we are still falling short of our goal. We need to give the end-user the option to plot as many gaussian functions as desired. This is achieved using the “MultipleOf” xml element:

```
<WaNoMultipleOf name="Gaussians">
 <Element id="0">
   <WaNoFloat name="mean">0.0</WaNoFloat>
   <WaNoFloat name="sigma">1.0</WaNoFloat>
   <WaNoString name="label">insert label</WaNoString>
 </Element>
</WaNoMultipleOf>
```

We include this in the WaNo XML file and get the following: A small + and – sign has appeared in the top right corner of the WaNo. By clicking +, new triples of mean, sigma and label are added to the rendered WaNo:





We now once more need to modify the run script to allow the flexible parsing of parameter triples. This is achieved by including an iterator in the parameter part of the submission script:

```
#!/bin/bash
python $NANOMATCH/PlotGauss/plot_gauss.py {% for element in wano["Gaussians"] %} \
                                          {{ element["mean"] }} \
                                          {{ element["sigma"] }} \
                                          {{ element["label"] }} {% endfor %}
```

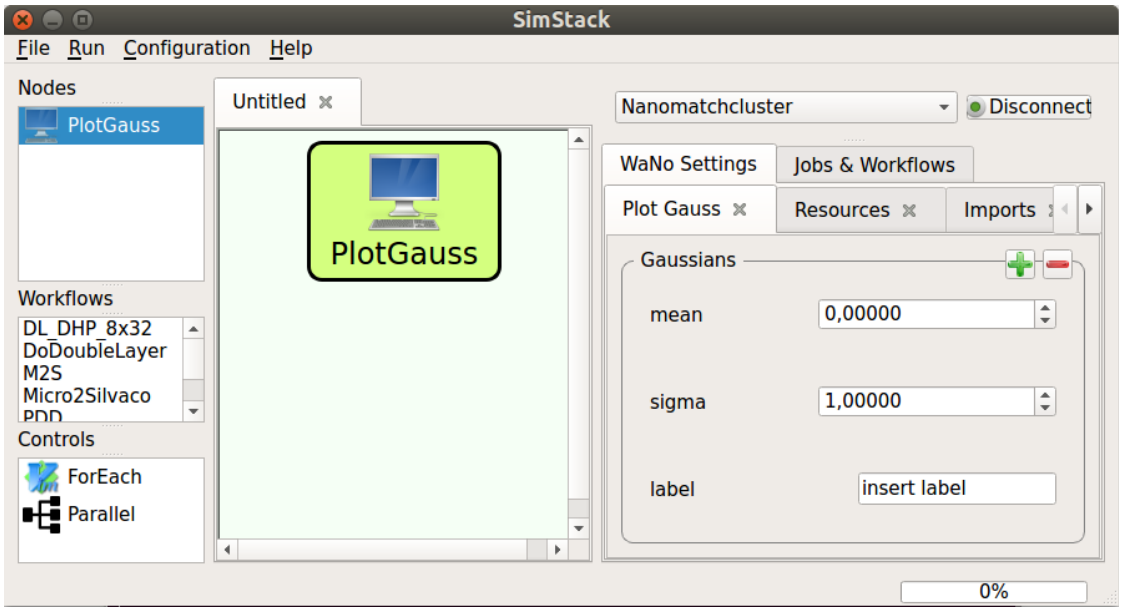
Note that, instead of having a separate execution script, we could have used the very same command and include it in the <WaNoExecCommand> tag.

2.4 Automated submission of plot_gauss.py

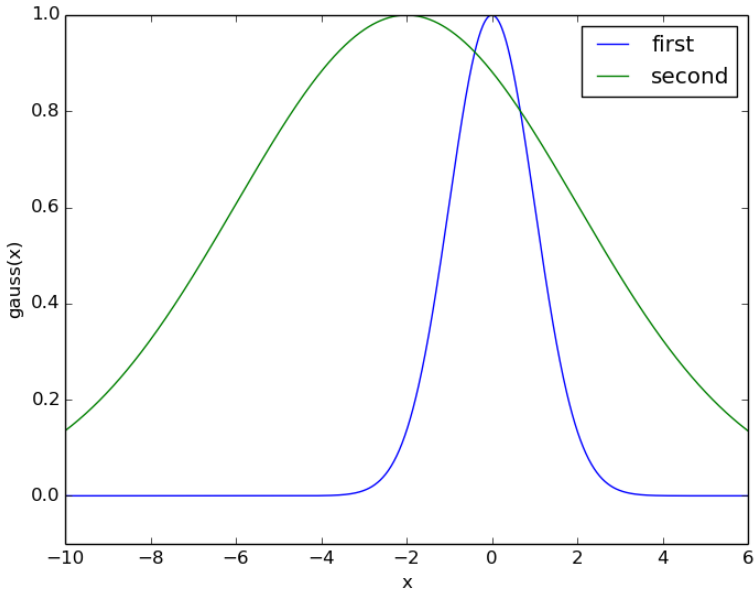
First, copy plot_gauss.py into your remote resource into any directory accessible by all users. Then modify the submission script in the WaNo/PlotGauss/ directory accordingly (adapt the path to the python code).

On your Laptop, make sure you have saved the PlottGauss.xml and the PlotGauss_init.sh in the simstack/wanos/PlotGauss/ directory. Now:

- Open Simstack and connect to your remote resource (top right button).
- Drag the PlotGauss WaNo from the top left menu (see picture below) into the middle pane
- Double click the PlotGauss element in the middle pane and set the parameters of your "simulation": number of gaussians to plot, means, sigmas and labels
- Save your "single-element-workflow" either by clicking File→ Save in the head menue or Ctrl+S. It will appear in the left column, second window from the top, and can be re-used at a later stage. This becomes relevant for more complex workflows consisting of more than a single element.
- Run the job by clicking Run → Run or Ctrl+R



You can monitor the workflow progress in the "Jobs & Workflows" tab in the right half of the SimStack Client. Select this tab and double-click on "Workflows". This opens up a list of all currently running and previously submitted workflows. The color in front of each workflow indicates a successfully finished workflow (green), running workflow (yellow) or failed workflow (red). Double clicking on each workflow opens another list of all modules in this workflow (in this case only the PlotGauss module). One more double click then shows a list of all files in the respective directory on the remote resource, including stdout and stderr for troubleshooting and the output file, in our case the gaussians.png. Double click each file to download.



3. Tips and tricks

- If you start a new WaNo for the first time, copy an existing WaNo and modify it step by step. This will make it easier to understand structure and functionality.
- To test a WaNo, use TestWaNo.py in the simstack/simstack directory. This will show you the rendered GUI.
- Many scientific codes have a variety of parameters that could, in principle, all be set by the user. However, for specific, reoccurring use-cases, only a specific set of parameters is needed and only a subset of those should be set by the user. When you start a new WaNo, instead of trying to construct a GUI where all parameters can be set by the user, ask yourself, what parameters you want the user to change, include those into the GUI and hard-code the rest. Depending on the tool/case, it may be beneficial to provide several separate WaNos for one program (e. g. a generic MD tool, where a typical simulation is structural optimization and another typical simulation is the computation of mechanical properties). Adaptions to the WaNo to allow more flexibility is a matter of minutes.