

Coding in the cloud

New ways to teach programming classes

Matthias Möller

Overview

- **Motivation:** OOSP-C++ course and lessons learnt last years
- **INGInious:** Overall concepts and adjustments for OOSP-C++ course
- **Conclusions:** Got interested? Then give it a try....

Motivation

Course: Object-Oriented Scientific Programming with C++ (OOSP-C++)

- Started as PhD seminar with 5-8 participants in 2015
- Has become a BSc/MSc/PhD course with about 80-100 participants
- Participants from all faculties within DCSE with very varying background
- **Hands-on course** (2h lectures + 4h lab sessions per week):
 - Need for sufficiently many and well-trained TAs (difficult to find!)
 - Reduce technical problems and use TAs for content-related support

Learning Objectives

1. Students will learn to **design, implement, and systematically validate well-structured and maintainable efficient computer programs in C++** for **solving scientific problems from their field of applications**.

This requires a good **knowledge of modern C++** features:

- OOP techniques: polymorphism, inheritance, encapsulation, abstraction
- Template meta-programming and compile-time optimisation techniques
- New (and really powerful) concepts introduced in C++11, 14, 17

together with **discipline-related knowledge** not taught in this course.

Learning Objectives

2. Students will learn to **use professional software development tools and workflows (version control systems, IDEs, build systems, debuggers, ...)** for **developing software projects in teams.**

This requires

- TU-wide availability of pre-defined software stack
- Willingness of students to use ICT infrastructure from TU
- Understanding of the need for it (not just among students)

Data Champions

TU Delft Cluster

Data Stewards

TU Delft ICT-services

DCSE D: Dream

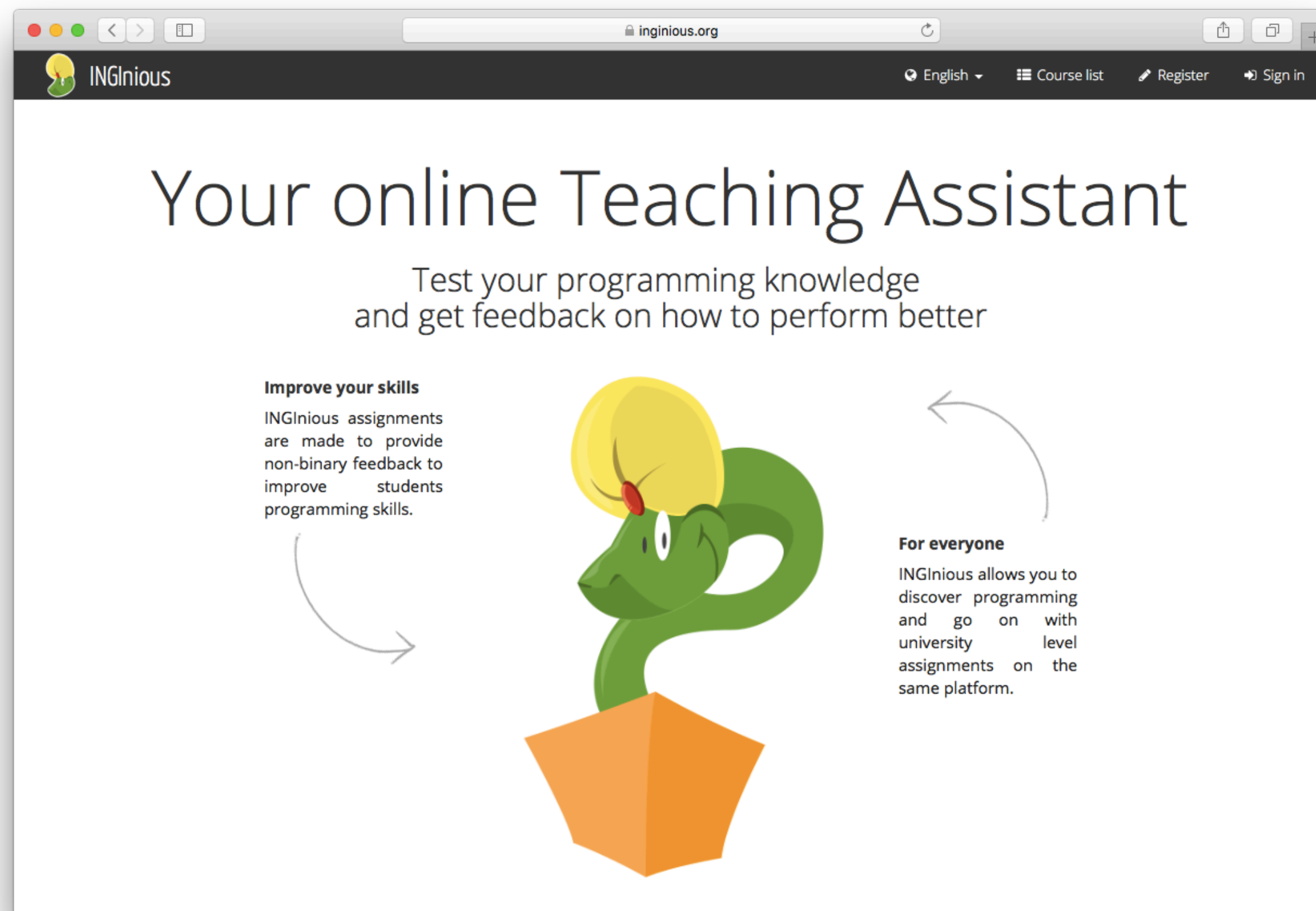


Lessons learned last years

1. Client-side solutions on students' computers are doomed to fail:
 - 1-2 weeks spent on giving installation support during lab sessions
 - $|S_{OSs} \times S_{Compilers}| \gg 1$, so one cannot test all possibilities before
 - C++ *is* platform dependent, so what is the reference for grading
 - New C++14 or 17 features not available in outdated compilers
2. Students are reluctant to use TU computers (even if they worked)

Conclusion: Web-based server-side solution (with admin rights) needed; step-by-step integration of software development aspects in next years

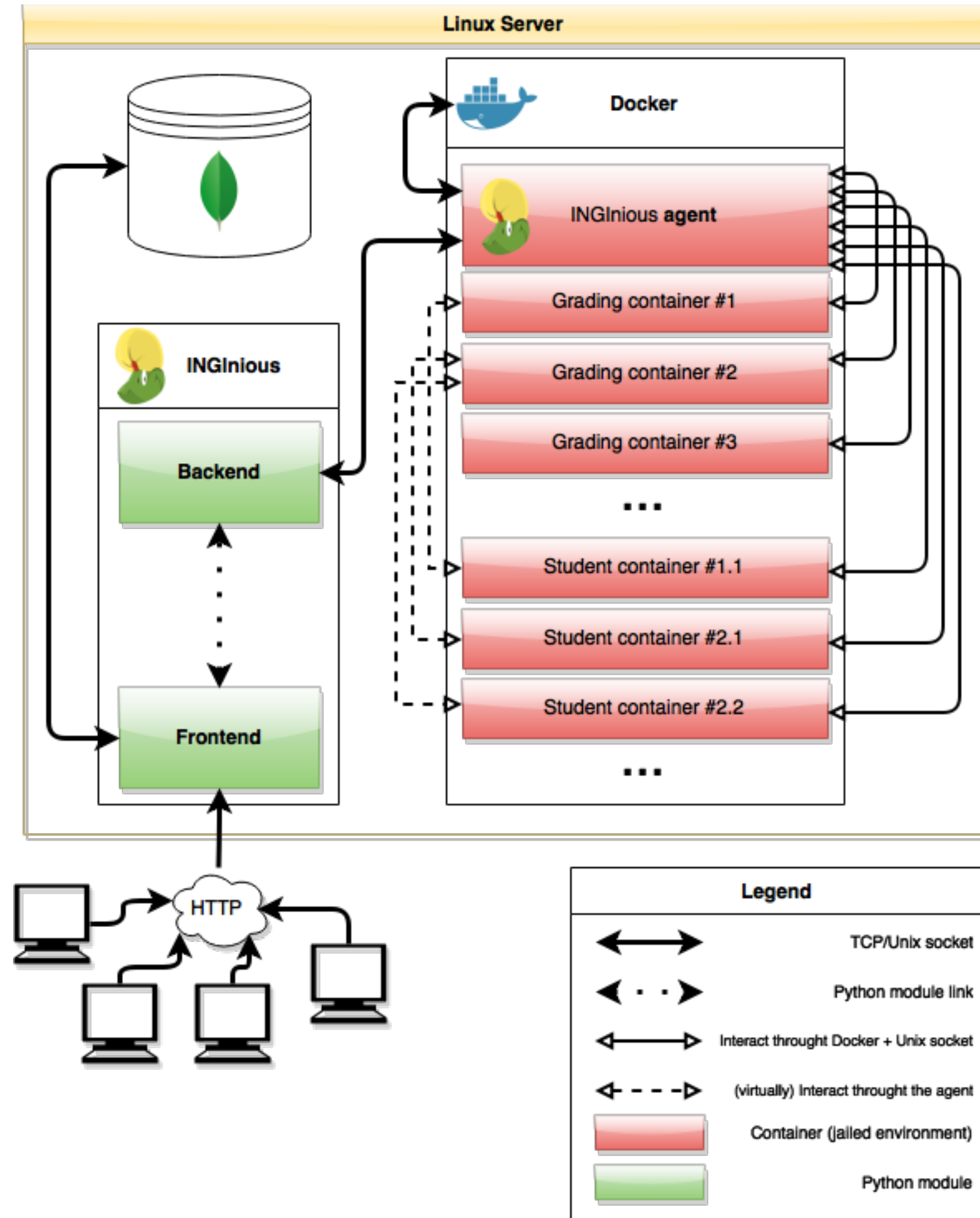
INGInious



- Open-source tool developed by the CSE department at UCLouvain/BE
- Used at UCL and for edX courses

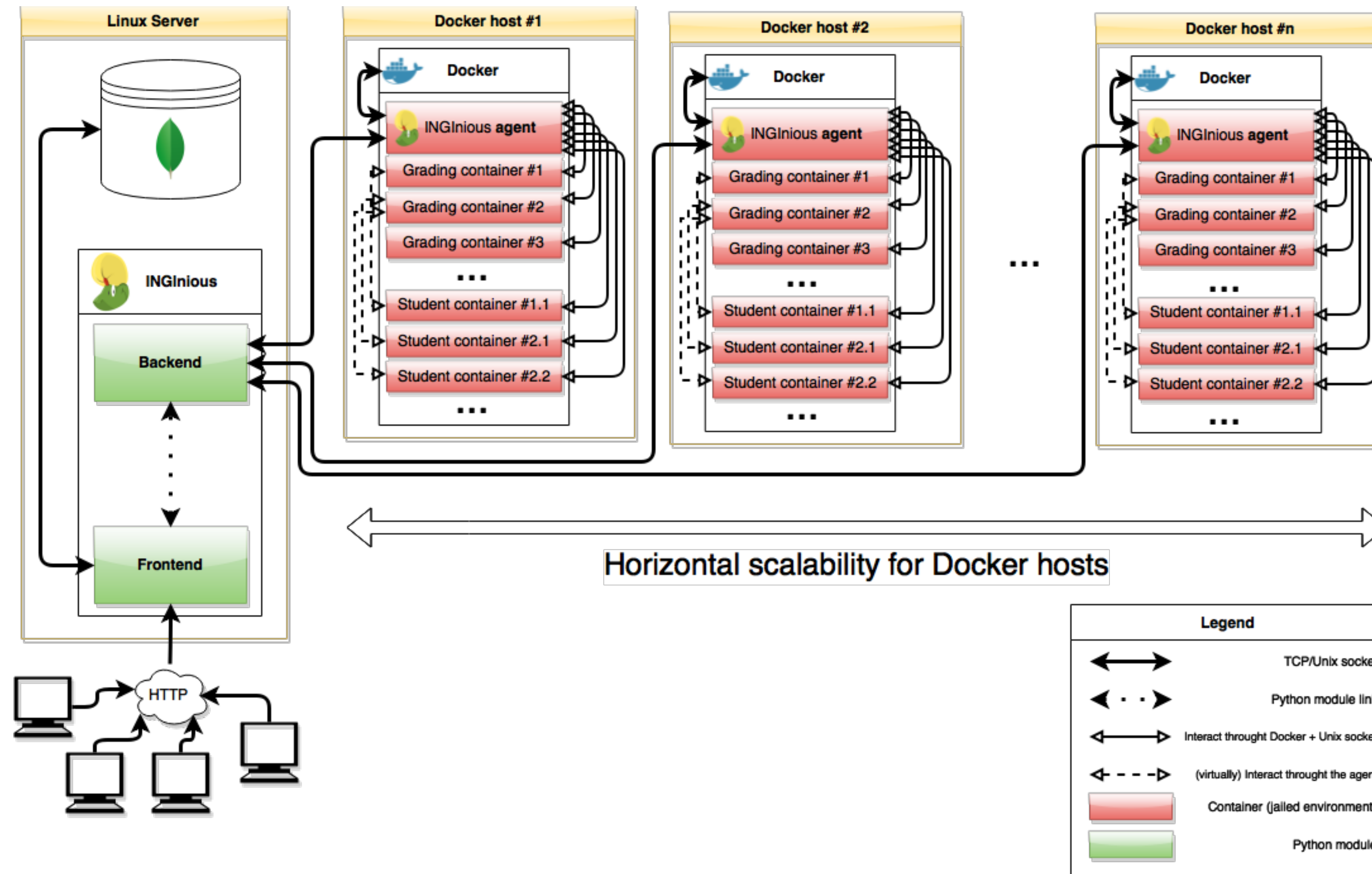


INGInious workflow

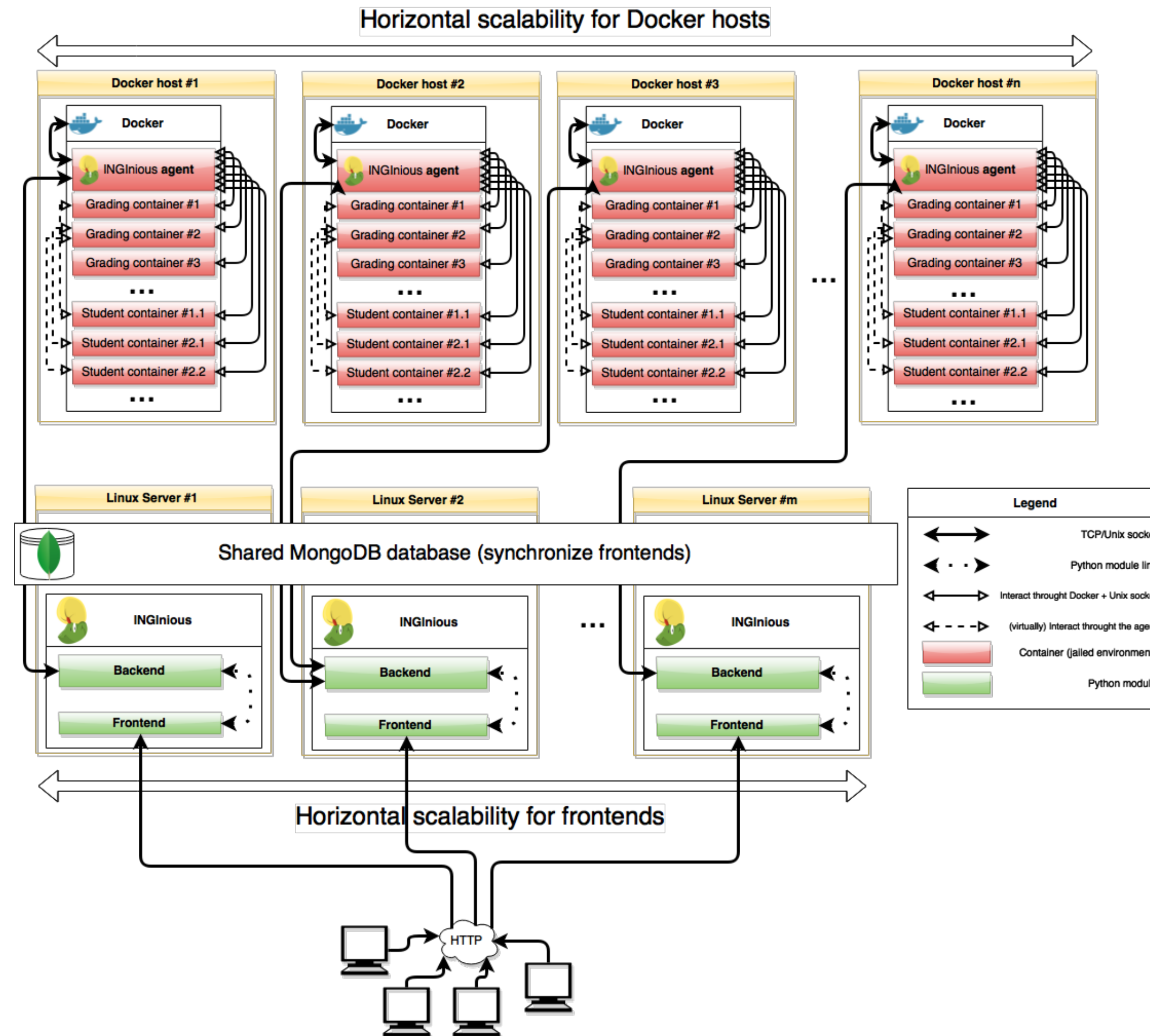


1. Student logs in to course web-site
2. Works on assignment in web formular
3. Submits the solution (history is stored)
4. *System assesses the solution in Docker container and returns feedback to student*
5. Student revises/accepts submission
6. *Final check by TAs/instructor for grading*

Scalability (near-term)

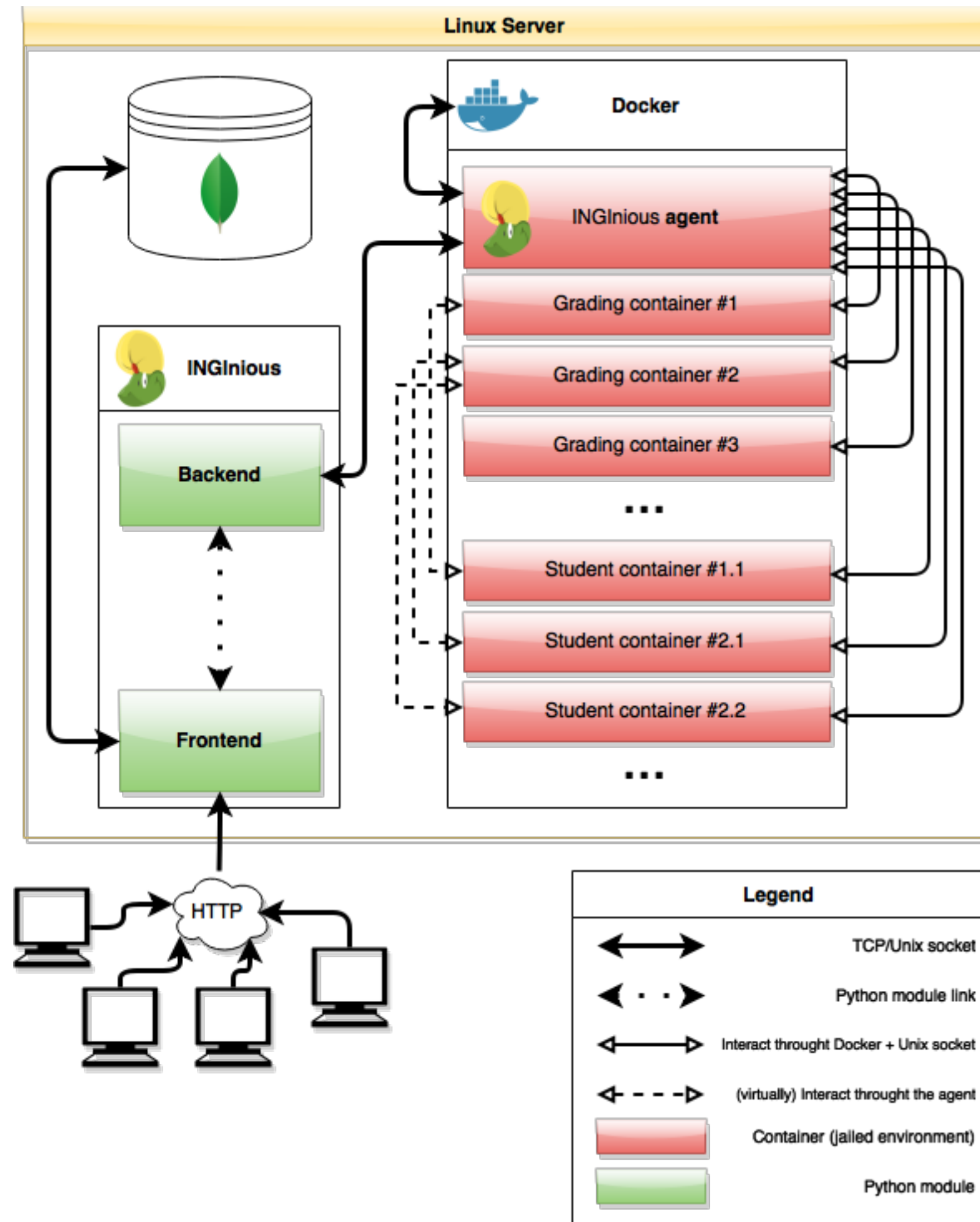


Scalability (long-term)



INGInious.EWI.tudelft.nl

- Runs on virtual server (CentOS 7) hosted, administrated and backed up by TUD-ICT
- First practical test in Q2 2018/19 for my OOSP-C++ course (130 registrations)



File-based course structure

- /var/www/INGInious
 - /Course1
 - /Task1
 - task.yaml
 - Description of task and assignment implemented in markdown language
 - run
 - Instructions for running/checking student submission and giving feedback
 - ...
 - /Task2
 - task.yaml
 - run
 - ...
 - /Course2
 - /Task1
 - ...

Good for backups and batch editing
(if you know what you are doing)

Direct SSH login to host needed

Web-based task editor

The screenshot shows the 'Edit task "01_demo"' page in the INginious system. The 'Basic settings' tab is selected and highlighted with a red box. The interface includes a navigation bar with 'Basic settings', 'Container setup', 'Subproblems', 'Tags', and 'Task files'. The main content area contains several form fields: 'Name' (Week 1] Demo 1 : Hello World!), 'Context' (a multi-line text area with content about the course and author), 'Author' (Matthias Möller), 'Grade weight' (0.0), 'Submission mode' (radio buttons for 'Individually' and 'Per group/team'), 'Submission storage' (radio buttons for 'All submissions' and 'Only the last' with a numeric input for '5'), 'Submission limits' (radio buttons for 'No limitation', 'Hard limit', and 'Soft limit' with numeric inputs for '5' and '5'), 'Evaluation submission' (radio buttons for 'Best submission', 'Last submission', and 'Student choice'), 'Accessible' (radio buttons for 'Never', 'Always', and 'Custom, from:'), and 'Random inputs' (a numeric input for '0'). A 'Delete task' button is visible next to the name field. A 'Save changes' button is at the bottom.

The screenshot shows the 'Edit task "01_demo"' page in the INginious system, with the 'Subproblems' tab selected and highlighted with a red box. The 'Problem id' is 'student_code'. The main content area shows the 'Context' field with a multi-line text area containing instructions: 'Calculating a result without using or outputting it somewhere is kind of useless...' and 'Interested in how you can print something on the screen? Try the code below and see how your changes affect the output!'. Below the context, there are fields for 'Language' (c), 'Type' (Multiline code), 'Optional?' (checkbox), and 'Default value' (a code editor with C++ code for a 'Hello World' program). A 'Save changes' button is at the bottom right. At the very bottom, there is a section for adding new subproblems with fields for 'New problem id' (new-problem-id), 'Problem type' (file upload), and an 'Add' button.

Web-based task editor

inginius.ewi.tudelft.nl

INGinius > [WI3720TU/WI4771TU Q2-2018/20... > Edit task "01_demo"

Edit task "01_demo"

Save changes View task

Basic settings Container setup Subproblems Tags Task files

Name [Week 1] Demo 1 : Hello World! Delete task

Context

```
1 This file is part of the course TW3720TU/WI4771TU
2
3 Object Oriented Scientific Programming with C++
4
5 Author: Matthias Möller
6
```

Author Matthias Möller

Grade weight (in comparison to other tasks) 0.0

Submission mode Individually Per group/team

Submission storage All submissions Only the last 5 submissions

Submission limits No limitation Hard limit: 5 submission(s) Soft limit: 5 submission(s) every 5 hour(s)

Accessible Never Always Custom, from: 2014-06-29 10:00 to: 2014-06-29 10:00

Random inputs 0 Regenerate input random

Save changes

Course settings

Students

Classrooms

Tasks

View submissions

Download submissions

Replay submissions

Danger zone

Contest

How to create a task?

Documentation

Save changes

inginius.ewi.tudelft.nl

INGinius > [WI3720TU/WI4771TU Q2-2018/20... > Edit task "01_demo"

Edit task "01_demo"

Save changes View task

Basic settings Container setup Subproblems Tags Task files

Problem id: student_code

Name Demo 1: Hello World!

Context

```
1 Calculating a result without using or outputting it somewhere is
2 kind of useless...
3 Interested in how you can use something on the screen? Try the
4 code below and see how your changes affect the output!
```

Language c

Type Multiline code

Optional?

Default value

```
1 // Include header file for standard input/output stream library
2 #include <iostream>
3
4 // The global main function that is the designated start of the
5 program
6 int main(){
7     // Write the string 'Hello World' to the default output stream
8     and
9     // terminate with a new line (that is what std::endl does)
10    std::cout << "Hello world!" << std::endl;
11
12    // Return code 0 to the operating system (= no error)
13    return 0;
14 }
```

New problem id new-problem-id Problem type file upload Add

Course settings

Students

Classrooms

Tasks

View submissions

Download submissions

Replay submissions

Danger zone

Contest

How to create a task?

Documentation

Save changes

Task execution and feedback

inginius.ewi.tudelft.nl

INGInious > [W3720TU/WI4771TU Q2-2018/20... > [Week 1] Demo 1 : Hello World!

[Week 1] Demo 1 : Hello World!

This file is part of the course TW3720TU/WI4771TU
Object Oriented Scientific Programming with C++
Author: Matthias Möller

Demo 1: Hello World!

Calculating a result without using or outputting it somewhere is kind of useless...
Interested in how you can print something on the screen? Try the code below and see how your changes affect the output!

```
1 // Include header file for standard input/output stream library
2 #include <iostream>
3
4 // The global main function that is the designated start of the program
5 int main(){
6
7     // Write the string 'Hello World' to the default output stream and
8     // terminate with a new line (that is what std::endl does)
9     std::cout << "Hello world!" << std::endl;
10
11     // Return code 0 to the operating system (= no error)
12     return 0;
13 }
14
```

Submit

Information

Author(s)	Matthias Möller
Deadline	No deadline
Status	Succeeded
Grade	100.0%
Grading weight	0.0
Attempts	13
Submission limit	No limitation
Category Tags	Demo, C++

Tags

Administration

- View submissions
- Edit task
- Debug information

For evaluation

- Best submission
- 02/11/2018 11:29:34 - 100.0%

Submission history

02/11/2018 11:29:34 - 100.0%
02/11/2018 11:29:27 - 100.0%
02/11/2018 11:29:20 - 100.0%
23/10/2018 13:03:44 - 0.0%
23/10/2018 13:03:22 - 100.0%
07/09/2018 09:42:03 - 0.0%

© 2014-2018 Université catholique de Louvain

INGInious is distributed under AGPL license

inginius.ewi.tudelft.nl

INGInious > [W3720TU/WI4771TU Q2-2018/20... > [Week 1] Demo 1 : Hello World!

[Week 1] Demo 1 : Hello World!

This file is part of the course TW3720TU/WI4771TU
Object Oriented Scientific Programming with C++
Author: Matthias Möller

Your answer passed the tests! Your score is 100.0%. [Submission #5bdee29735f3f3ffae2d7fd]

- Your code compiles.
- Your code passed all tests successfully.

Hello world!

Demo 1: Hello World!

- First output ever
- ⇒ Success (1/1) pts

Calculating a result without using or outputting it somewhere is kind of useless...
Interested in how you can print something on the screen? Try the code below and see how your changes affect the output!

```
1 // Include header file for standard input/output stream library
2 #include <iostream>
3
4 // The global main function that is the designated start of the program
5 int main(){
6
7     // Write the string 'Hello World' to the default output stream and
8     // terminate with a new line (that is what std::endl does)
9     std::cout << "Hello world!" << std::endl;
10
11     // Return code 0 to the operating system (= no error)
12     return 0;
13 }
14
```

Submit

Information

Author(s)	Matthias Möller
Deadline	No deadline
Status	Succeeded
Grade	100%
Grading weight	0.0
Attempts	14
Submission limit	No limitation
Category Tags	Demo, C++

Tags

Administration

- View submissions
- Edit task
- Debug information

For evaluation

- Best submission
- 04/11/2018 13:14:15 - 100.0%

Submission history

04/11/2018 13:14:15 - 100.0%
02/11/2018 11:29:34 - 100.0%
02/11/2018 11:29:27 - 100.0%
02/11/2018 11:29:20 - 100.0%
23/10/2018 13:03:44 - 0.0%
23/10/2018 13:03:22 - 100.0%

© 2014-2018 Université catholique de Louvain

INGInious is distributed under AGPL license

Task execution and feedback

The screenshot shows the INginious interface for a task titled "[Week 1] Demo 1 : Hello World!". The task description includes instructions and a code editor with the following C++ code:

```
1 // Include header file for standard input/output stream library
2 #include <iostream>
3
4 // The global main function that is the designated start of the program
5 int main(){
6
7     // Write the string 'Hello World' to the default output stream and
8     // terminate with a new line (that is what std::endl does)
9     std::cout << "Hello world!" << std::endl;
10
11     // Return code 0 to the operating system (= no error)
12     return 0;
13 }
14
```

The submission history table shows a successful submission on 04/11/2018 at 13:14:15 with a score of 100.0%.

Submission Date	Score
04/11/2018 13:14:15	100.0%
02/11/2018 11:29:34	100.0%
02/11/2018 11:29:27	100.0%
02/11/2018 11:29:20	100.0%
23/10/2018 13:03:44	0.0%
23/10/2018 13:03:22	100.0%

The screenshot shows the INginious interface for the same task, but with a failed submission. A red error message box is displayed, indicating a compilation error:

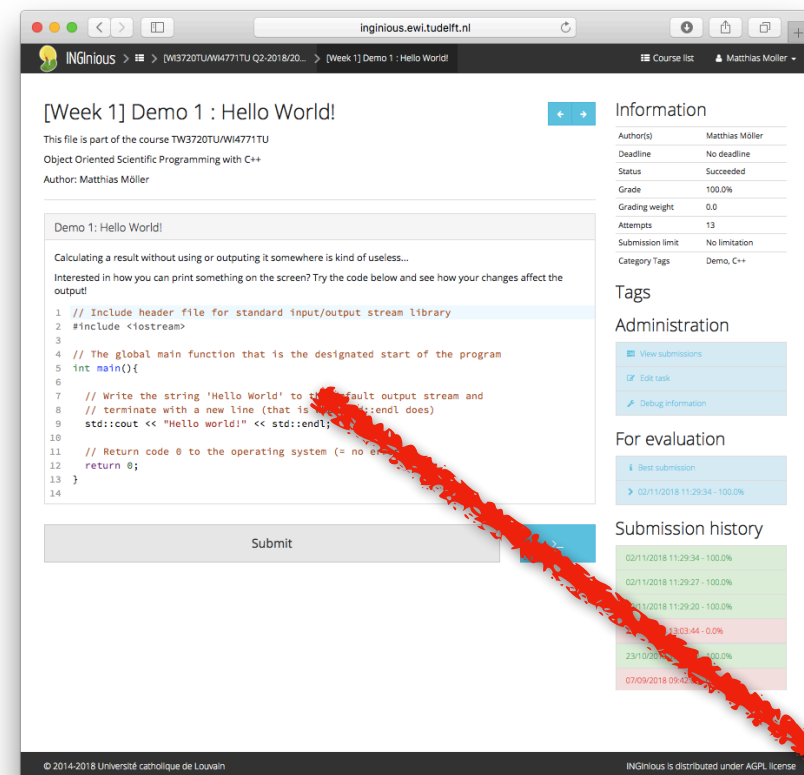
```
There are some errors in your answer. Your score is 0.0%. [Submission #5bdee30835f3f33ffae2d802]
Your code failed to compile. The make command returned the following error message:

In function 'int main()':
13:1: error: expected ';' before '}' token
  }
  ^
```

The submission history table shows a failed submission on 04/11/2018 at 13:16:08 with a score of 0.0%.

Submission Date	Score
04/11/2018 13:16:08	0.0%
04/11/2018 13:14:15	100.0%
02/11/2018 11:29:34	100.0%
02/11/2018 11:29:27	100.0%
02/11/2018 11:29:20	100.0%
23/10/2018 13:03:44	0.0%

Assessment workflow

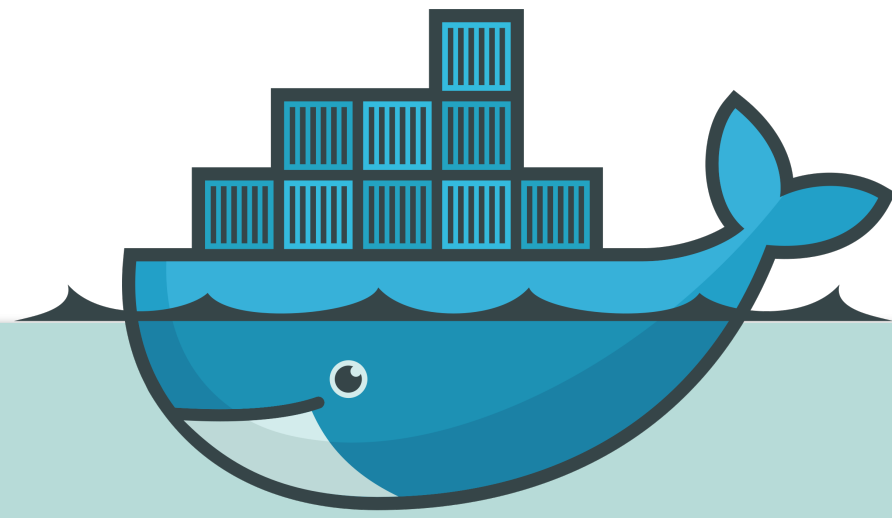
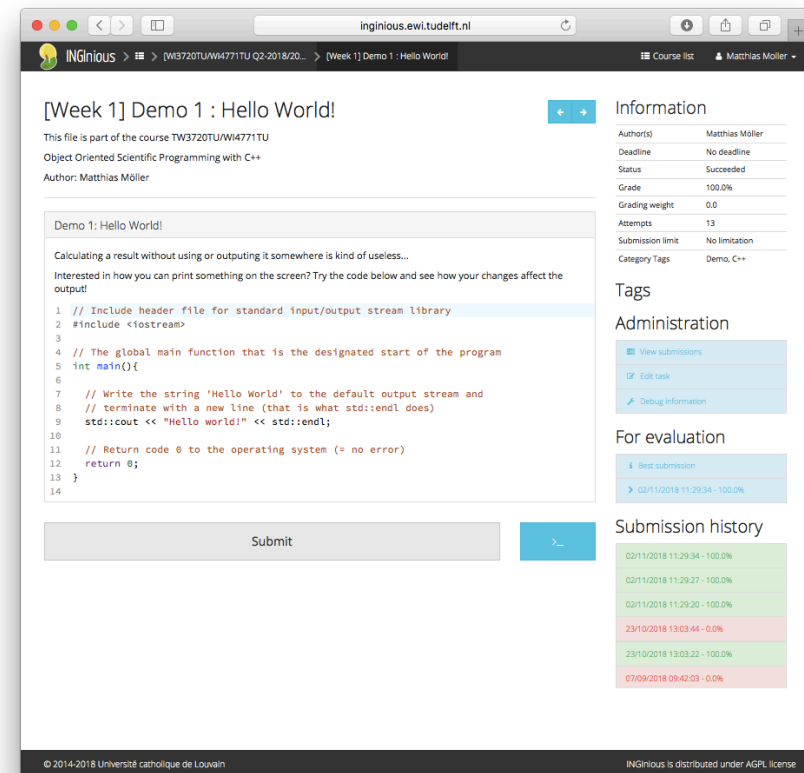


```
Template.cxx
1 // Include header file for standard input/output stream library
2 #include <iostream>
3
4 // The global main function that is the designated start of the program
5 int main(){
6
7     // Write the string 'Hello World' to the default output stream and
8     // terminate with a new line (that is what std::endl does)
9     std::cout << "Hello world!" << std::endl;
10
11     // Return code 0 to the operating system (= no error)
12     return 0;
13 }
14
```

1. Smart copy-and-paste:
Student input is parsed for banned 'expressions' and injected into template file.

```
File list /run x /student/config/banned x
1 # Input here the banned include names:
2 bannedIncludeList = thread, mutex
3
4 # Input here the banned function names:
5 bannedFunctionList = open, fprintf
6
7 # Input here the banned words (Not recommended to use this):
8 bannedWordList = fstream
9
10 # Note: Don't change the layout too much, it is read with a custom function.
11 # Expected layout: <listName> = <element1>, <element2>, <element3>
12 # The amount of elements are not limited to 3, it is just an example.
```

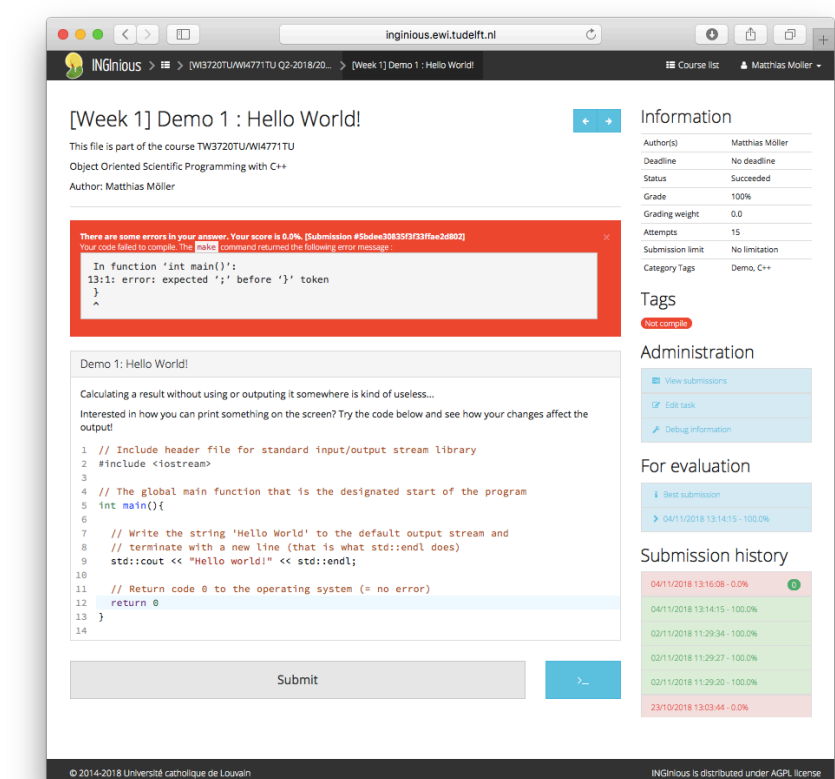
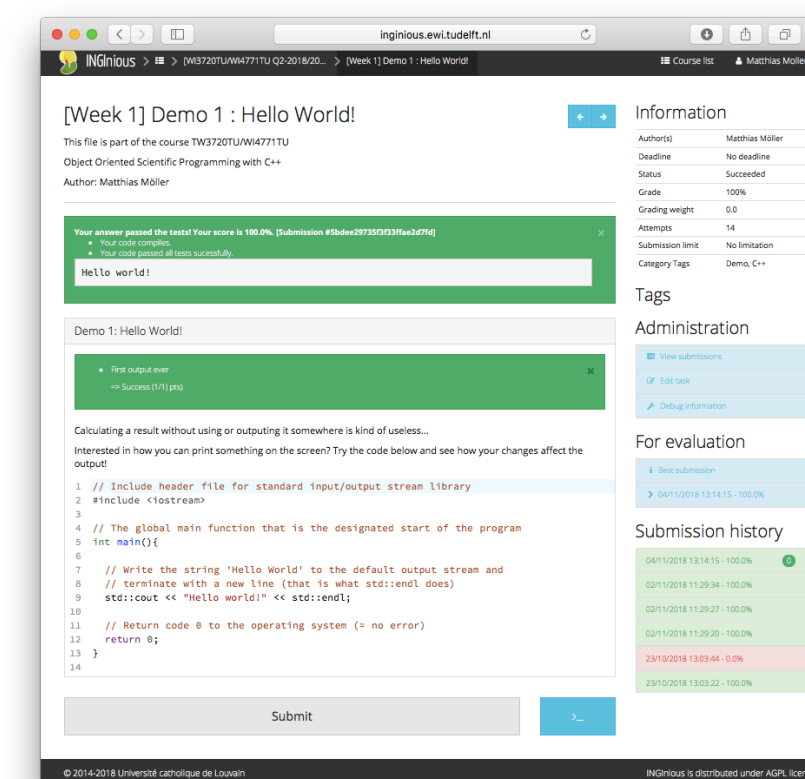
Assessment workflow



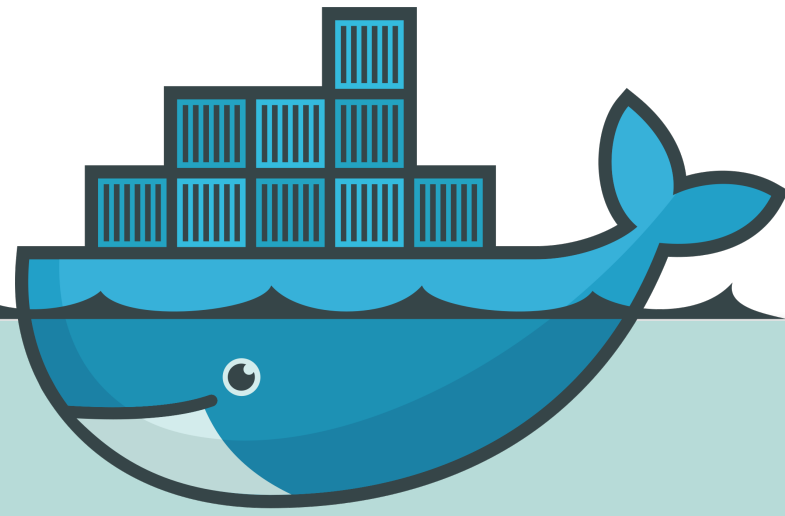
Test configuration
Makefile
Template.cxx

```
1 // Include header file for standard input/output stream library
2 #include <iostream>
3
4 // The global main function that is the designated start of the program
5 int main(){
6
7     // Write the string 'Hello World' to the default output stream and
8     // terminate with a new line (that is what std::endl does)
9     std::cout << "Hello world!" << std::endl;
10
11     // Return code 0 to the operating system (= no error)
12     return 0;
13 }
14
```

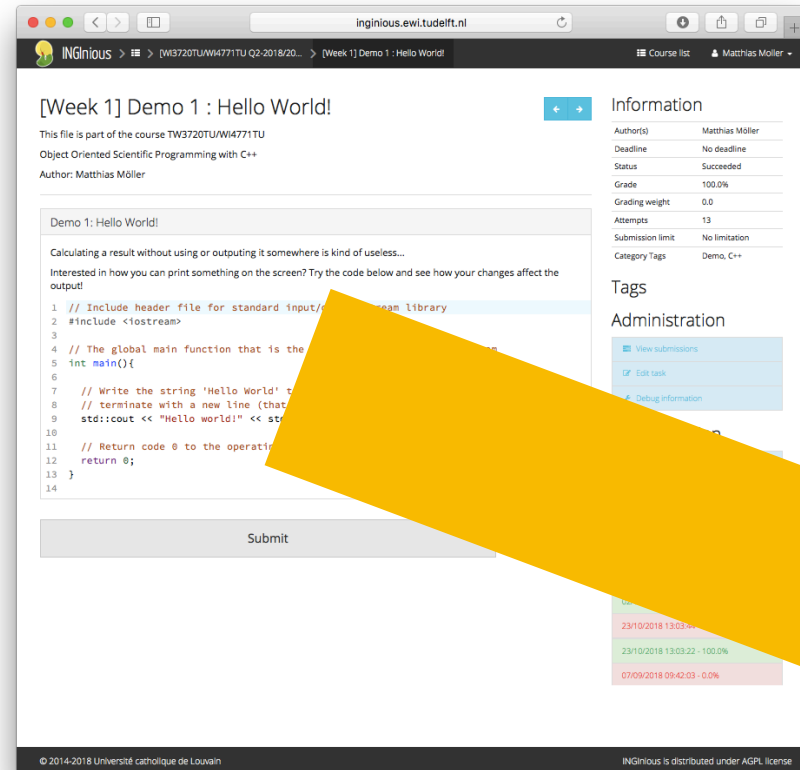
2. Docker-based execution: Input files are copied into new docker container that compiles submission based on Makefile. Compiler and execution output is captured, post-processed and presented to the student.



Assessment workflow

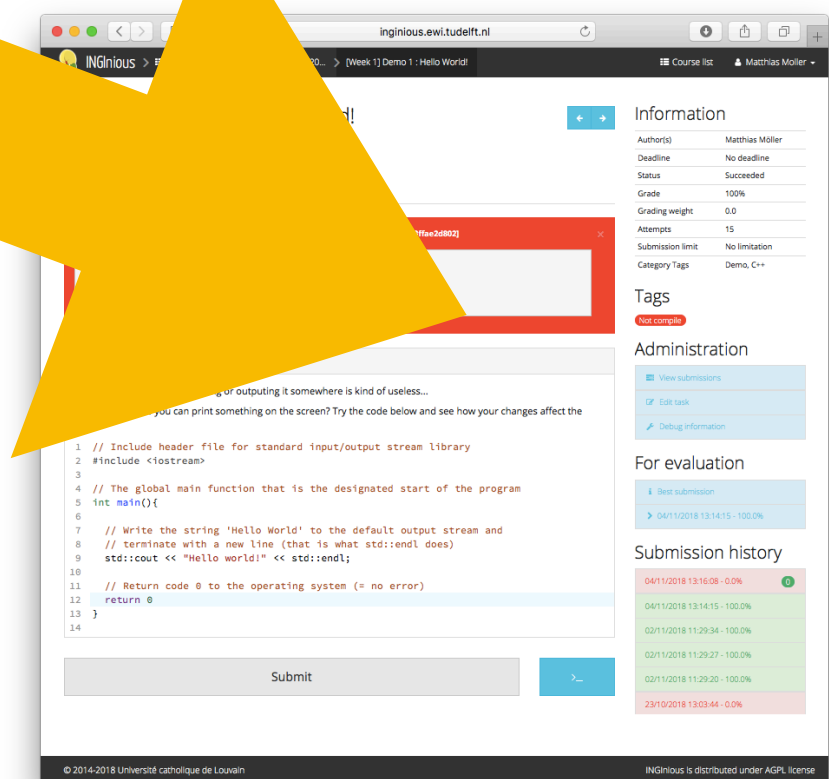


Flexible Python-based workflow developed by BSc student Sybold Hijlkema



Test configuration
Makefile
...ate.cxx

```
1 // Include header file for standard input/output stream library
2 #include <iostream>
3
4 // The global main function that is the designated start of the program
5 int main(){
6
7 // Write the string 'Hello World' to the default output stream and
8 // terminate with a new line (that is what std::endl does)
9 std::cout << "Hello world!" << std::endl;
10
11 // Return code 0 to the operating system (= no error)
12 return 0;
13 }
14
```



Python+Docker

RUN script:

- Shell scripts (from tutorial) for short answer and multiple-choice tasks
- Python script (by S. Hijlkema) for C++ (also for C) programming tasks

Docker container:

- Default containers with C/C++ (GCC 4.8.5), Python (2.7), ...
- Customised containers with GCC 7.x and 8.x for C++14, 17 support

Example Dockerfile

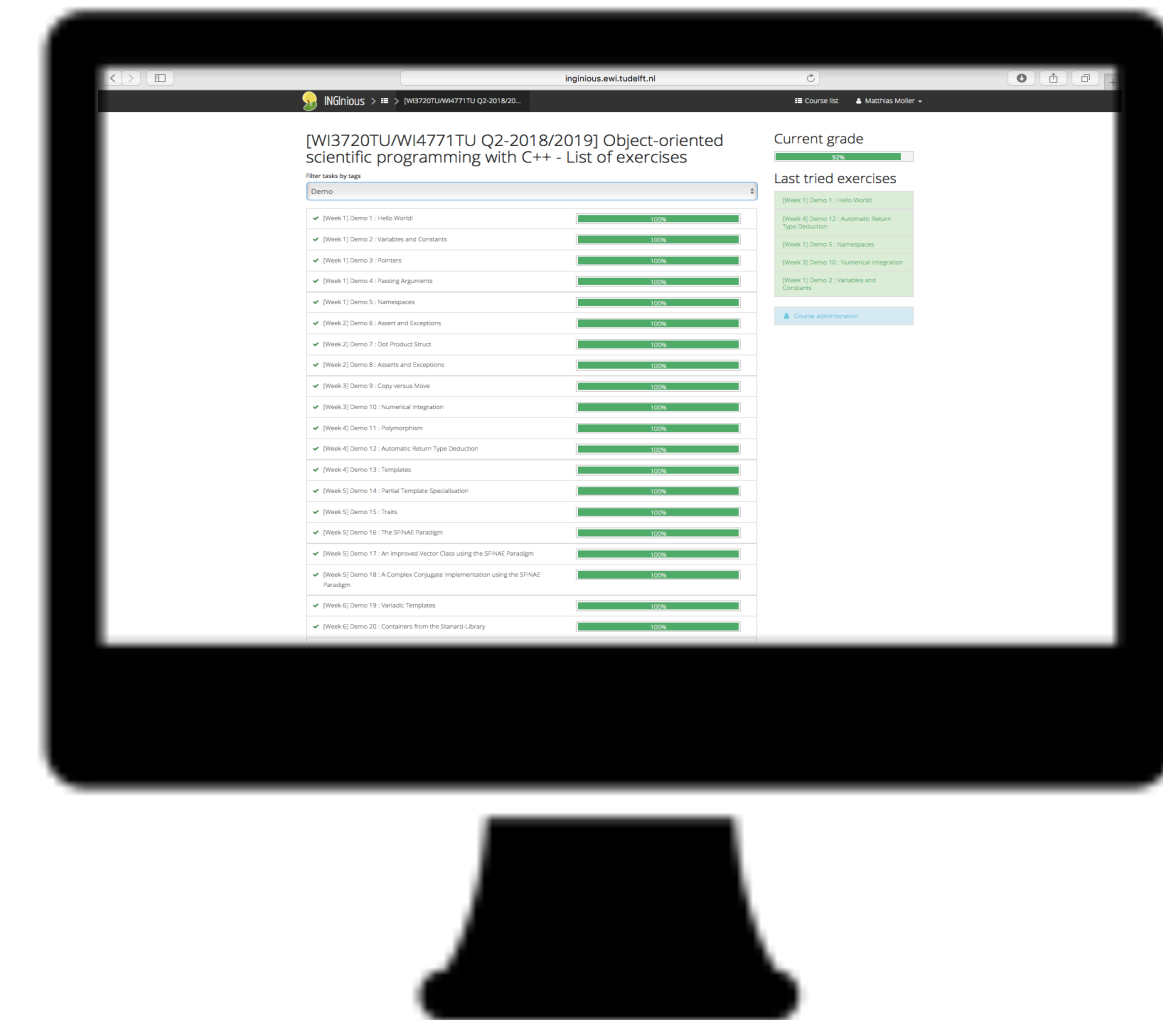
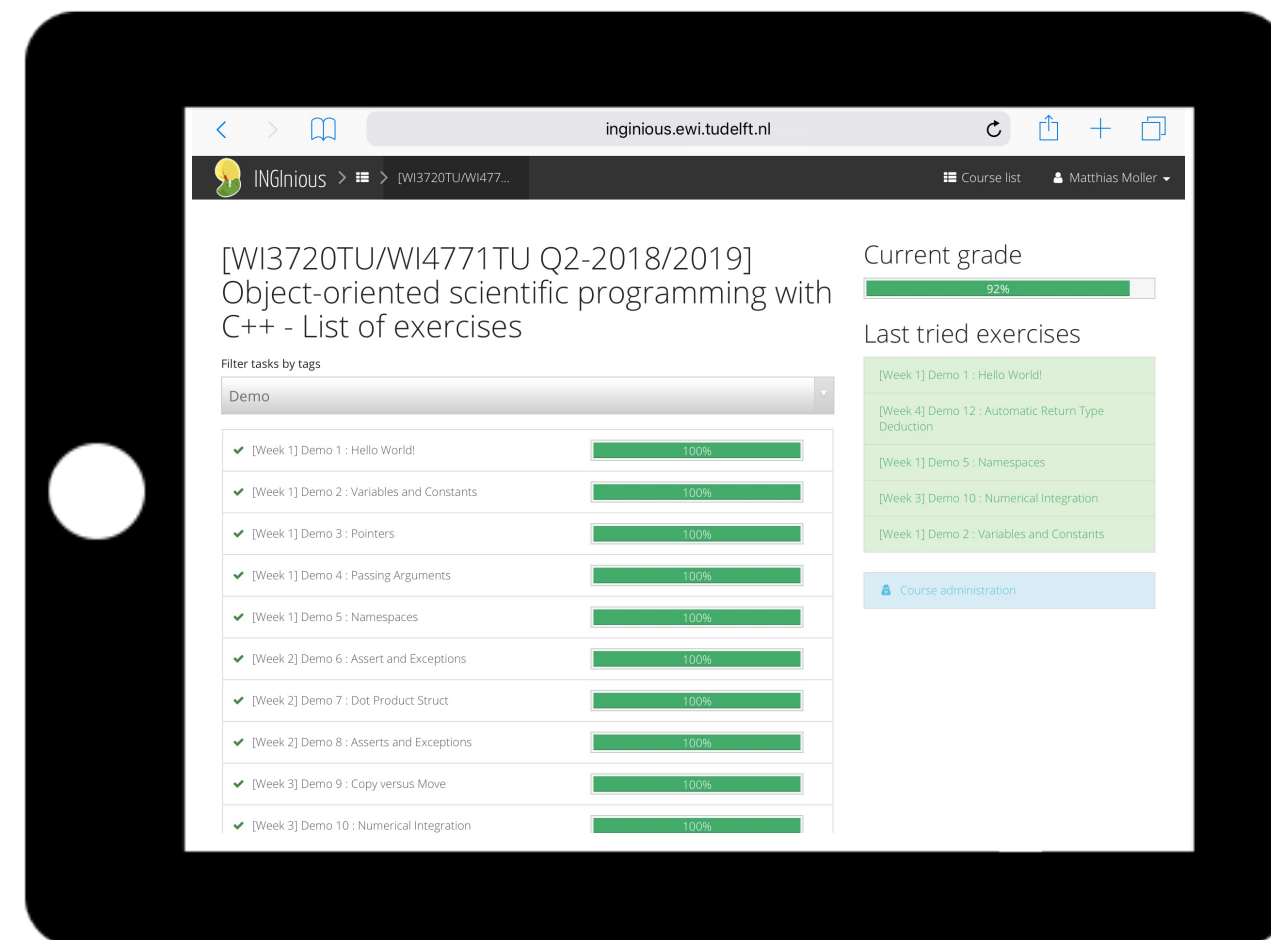
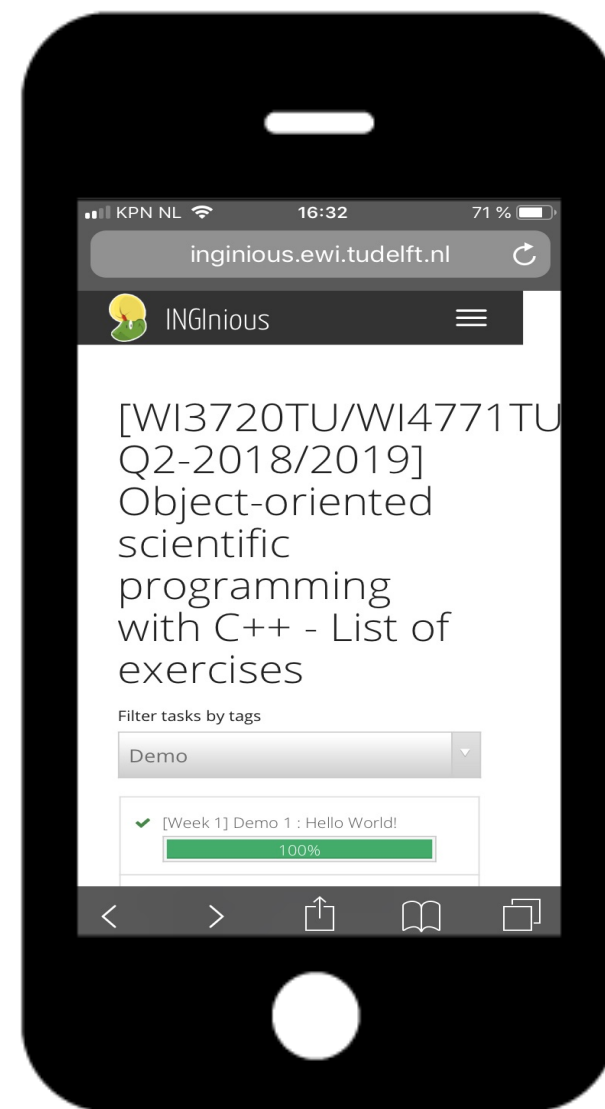
```
FROM      ingi/inginiuous-c-base
LABEL     org.inginiuous.grading.name="cpp-gcc7"
RUN       yum install -y centos-release-scl && \
          yum-config-manager --enable rhel-server-rhscl-7-rpms && \
          yum install -y yum install -y devtoolset-7-gcc devtoolset-7-
gcc-c++ devtoolset-7-gdb devtoolset-7-cpp devtoolset-7-make cmake
devtoolset-7-valgrind devtoolset-7-binutils libstdc++ clang clang-
analyzer clang-devel llvm automake check check-devel CUnit CUnit-
devel zlib-devel openssl-devel time jansson-devel graphviz graphviz-
devel cppcheck && \
          yum clean all

# Set PATH, LD_LIBRARY_PATH etcetera
ENV       PATH=/opt/rh/...
```

The sky is the limit

- Install commercial compilers, non-free test backends, simulators, ...
- Integrate proprietary libraries/tools (in binary form) in Docker image
- Use customised Docker container as
 - abstraction to special hardware (NVIDIA-CUDA, Maxeler-FPGA, ...)
 - communicator to external computer system (QuTech, Cluster, ...)
 - drivers for software testing (fuzzing, regression, ...)

It works on all platforms



Outlook

- First real-world test in Q2 2018/2019 with >130 participants (LObj. 1)
- Integration of web-based IDE and DevTools planned for 2019/20 (LObj. 2)
- If you are interested to give it a try for your course, let me know!



Acknowledgement: Sybold Hijlkema, Niels de Koeijer
Financial support by TU Delft Graduate School Kick-Start Funding