

Practical Lab

Cloud Systems Engineering

(cloud-lab)

Chair of Decentralized Systems Engineering
<https://dse.in.tum.de/>



Welcome to the cloud-lab!

The assignments

- 4 graded assignments
 - No exams, presentation, etc.
- About three weeks time for each assignment
 - Videos explaining tasks and background released on Mondays
 - Q&A sessions for assignments on Thursdays
- Each assignment comes with tests that we run to grade your solution
 - Tests run on our own CI runners whenever a commit is pushed
 - Passed tests result in points which determine your grade
 - Only the last commit before deadline is graded

The assignments



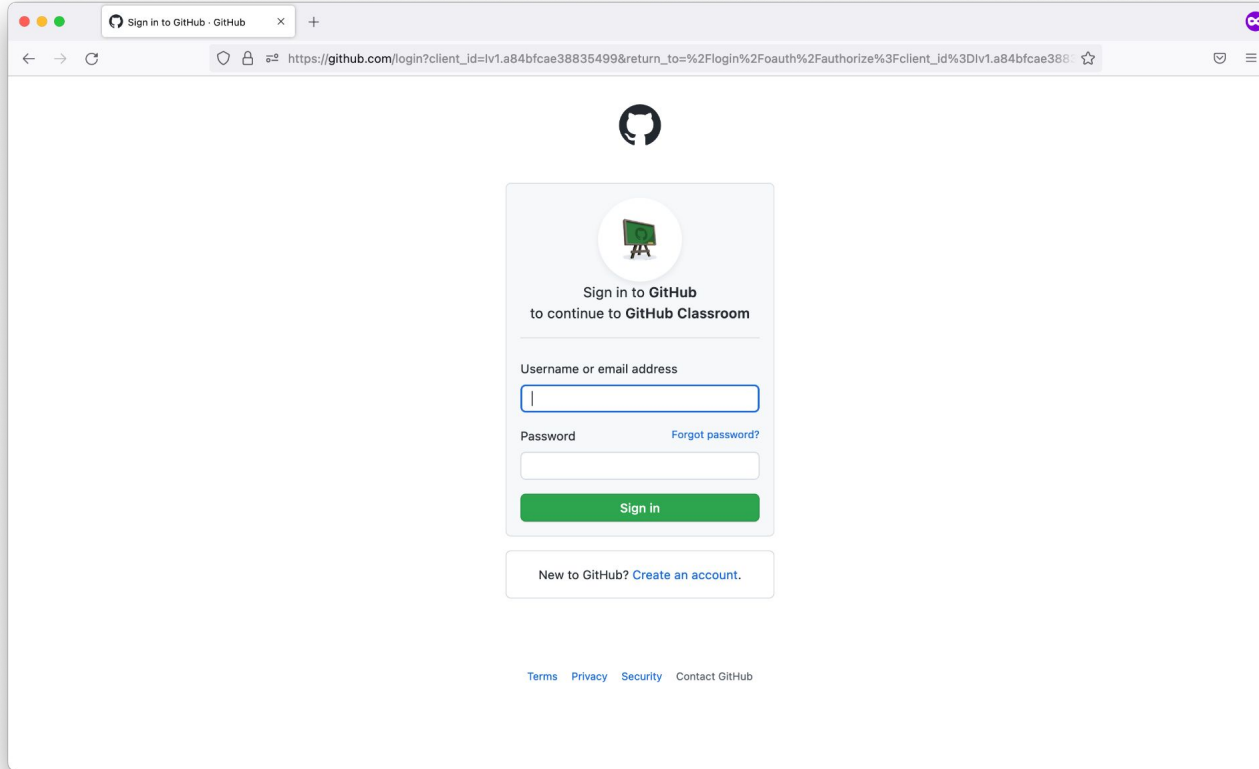
- Code should be written in C++
 - Other system programming languages like Rust may be used as well but there is no support by us
- In detail task description in [README.md](#) in the task folder
- Code is hosted on GitHub and will be auto-graded via GitHub classroom
 - You will need to create a GitHub account
 - We will send you invitation links for each assignment via Slack

Our CI runners

- Fancy kubernetes cluster
- Three servers, each equipped with
 - 2x Intel Xeon 5215
(10 cores / 20 threads each)
 - 128 GiB RAM
 - 10G NICs





Getting the assignment code



Sign in to GitHub · GitHub

https://github.com/login?client_id=lv1.a84bfcae38835499&return_to=%2Flogin%2Foauth%2Fauthorize%3Fclient_id%3Dlv1.a84bfcae38835499&return_to=%2Flogin%2Foauth%2Fauthorize%3Fclient_id%3Dlv1.a84bfcae38835499





Sign in to GitHub
to continue to GitHub Classroom

Username or email address

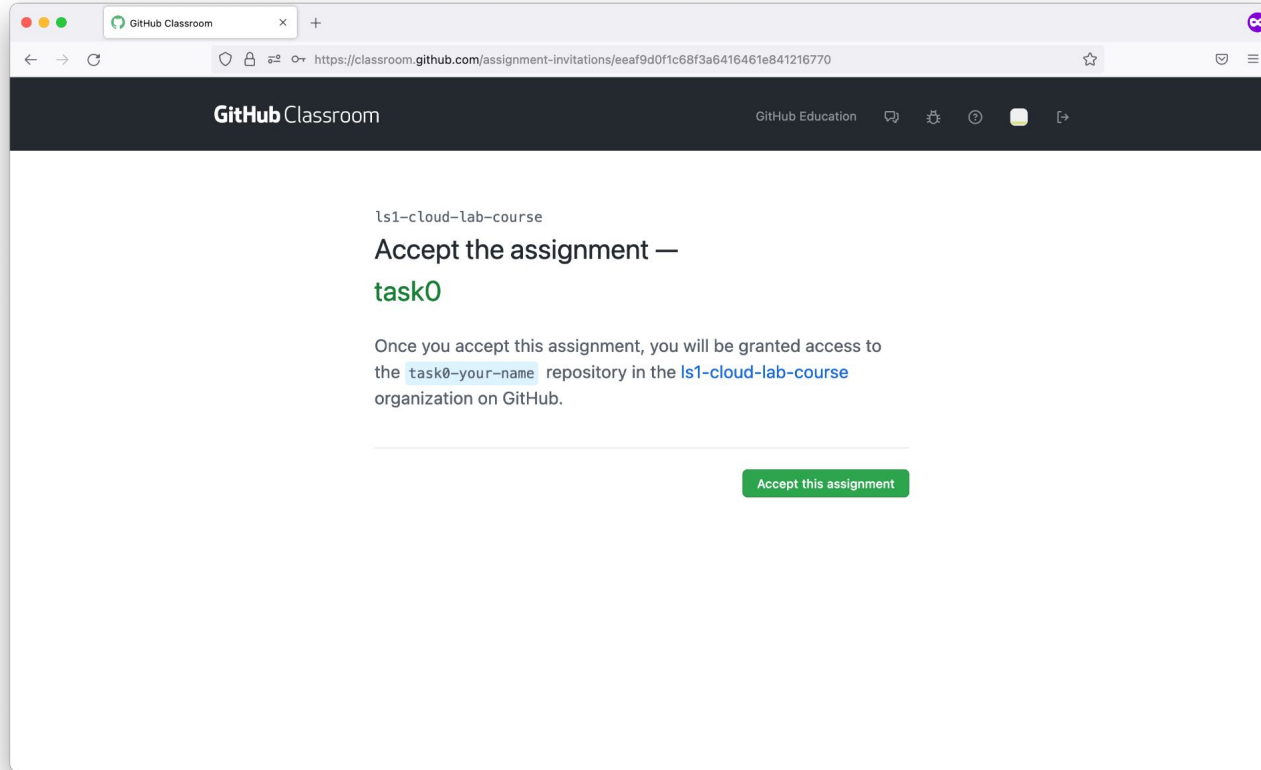
Password [Forgot password?](#)

[Sign in](#)

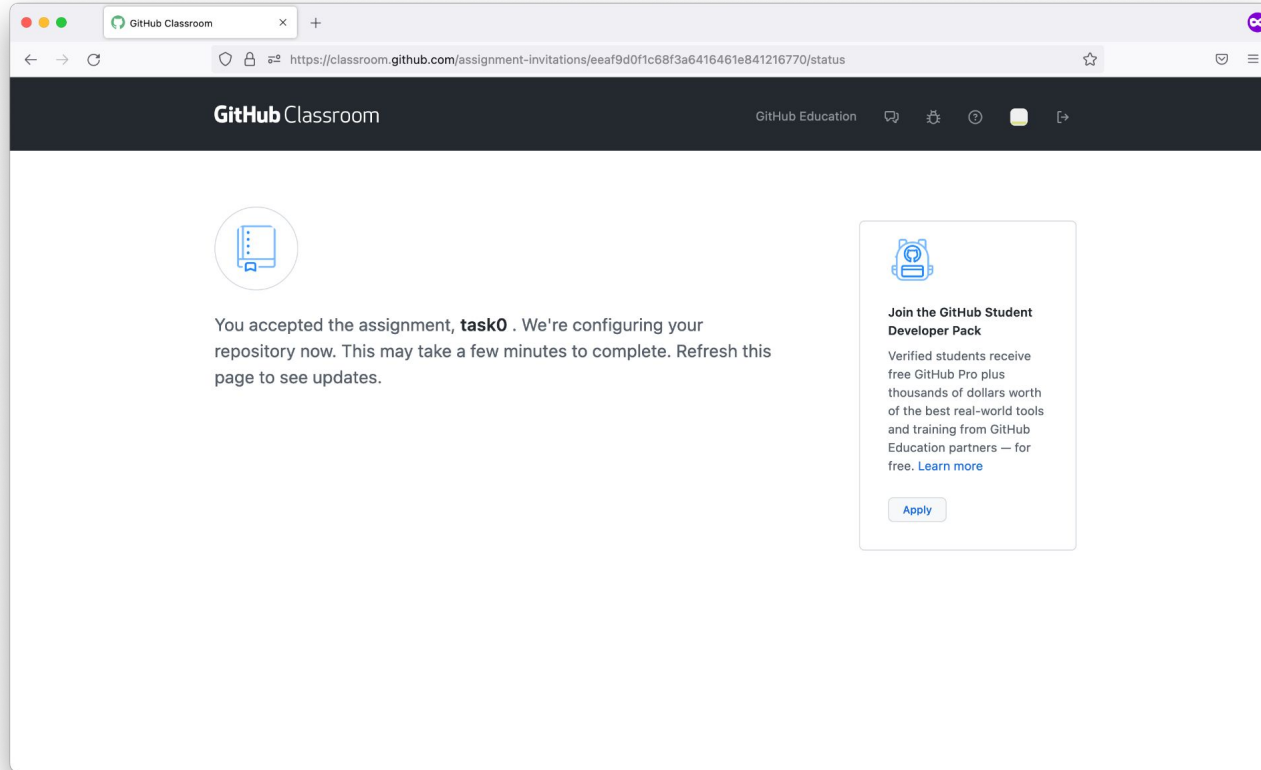
New to GitHub? [Create an account.](#)

[Terms](#) [Privacy](#) [Security](#) [Contact GitHub](#)

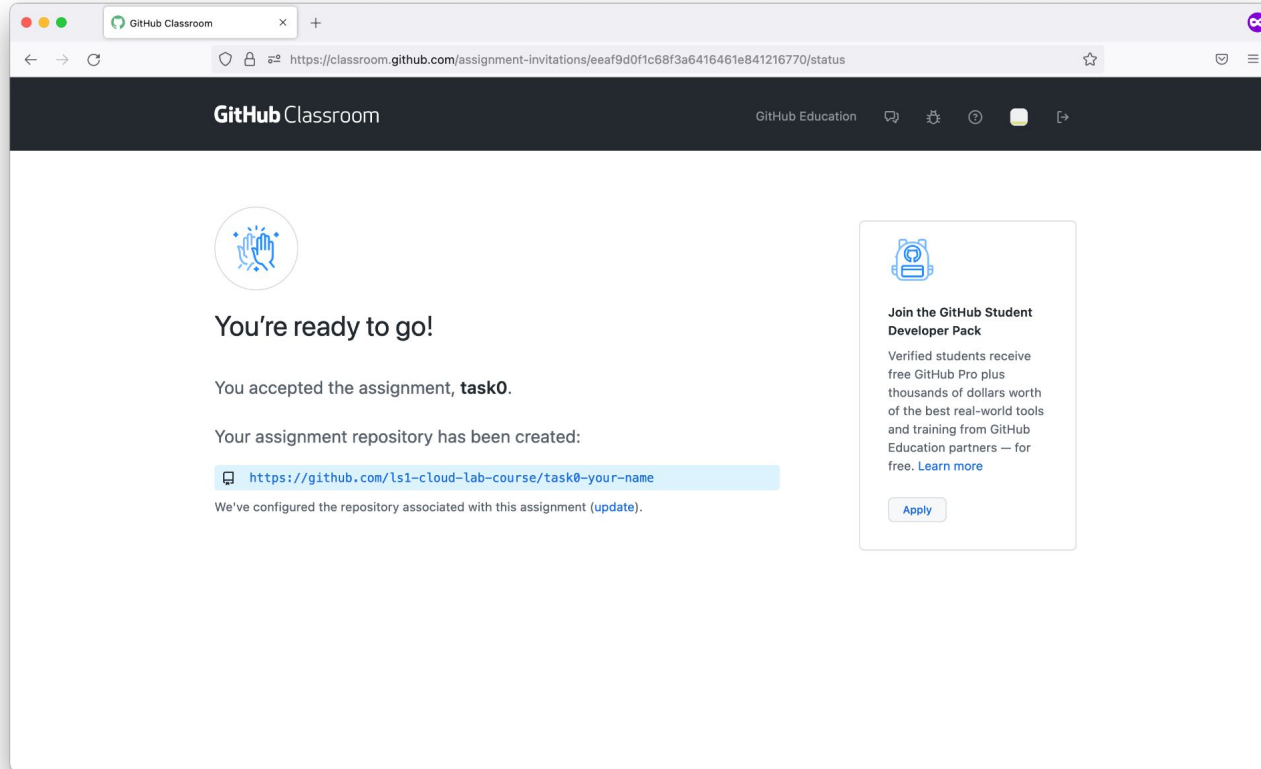
Getting the assignment code



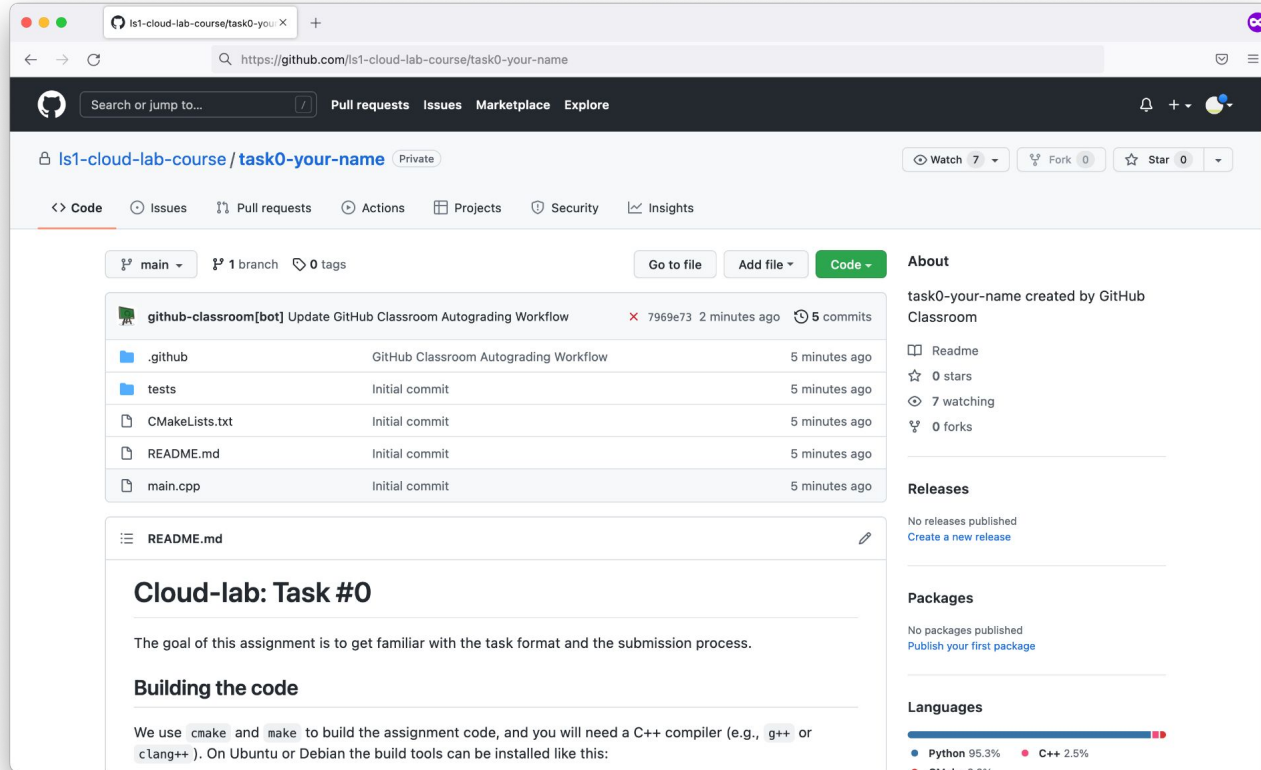
Getting the assignment code



Getting the assignment code

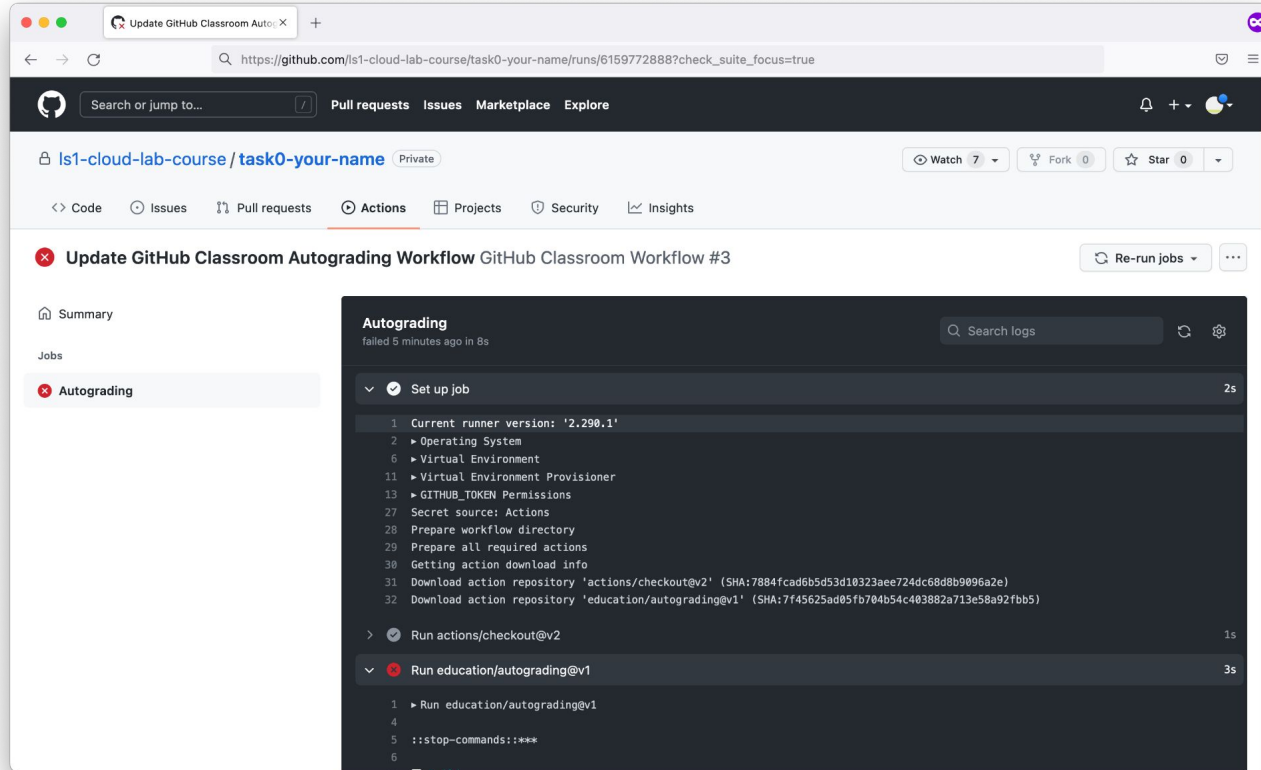


Getting the assignment code



The screenshot shows a web browser displaying a GitHub repository page. The address bar shows the URL `https://github.com/ls1-cloud-lab-course/task0-your-name`. The repository name is `ls1-cloud-lab-course / task0-your-name` and it is marked as `Private`. The page includes navigation tabs for `Code`, `Issues`, `Pull requests`, `Actions`, `Projects`, `Security`, and `Insights`. Below the repository name, there are buttons for `Watch` (7), `Fork` (0), and `Star` (0). The main content area shows the file list for the `main` branch, including `.github`, `tests`, `CMakeLists.txt`, `README.md`, and `main.cpp`. The `README.md` file is selected, showing the title **Cloud-lab: Task #0** and the text: "The goal of this assignment is to get familiar with the task format and the submission process." and "Building the code". The text continues: "We use `cmake` and `make` to build the assignment code, and you will need a C++ compiler (e.g., `g++` or `clang++`). On Ubuntu or Debian the build tools can be installed like this:". The right sidebar contains sections for **About**, **Releases**, **Packages**, and **Languages**. The **Languages** section shows a bar chart with `Python` at 95.3%, `C++` at 2.5%, and `CMake` at 2.2%.

Submitting your code

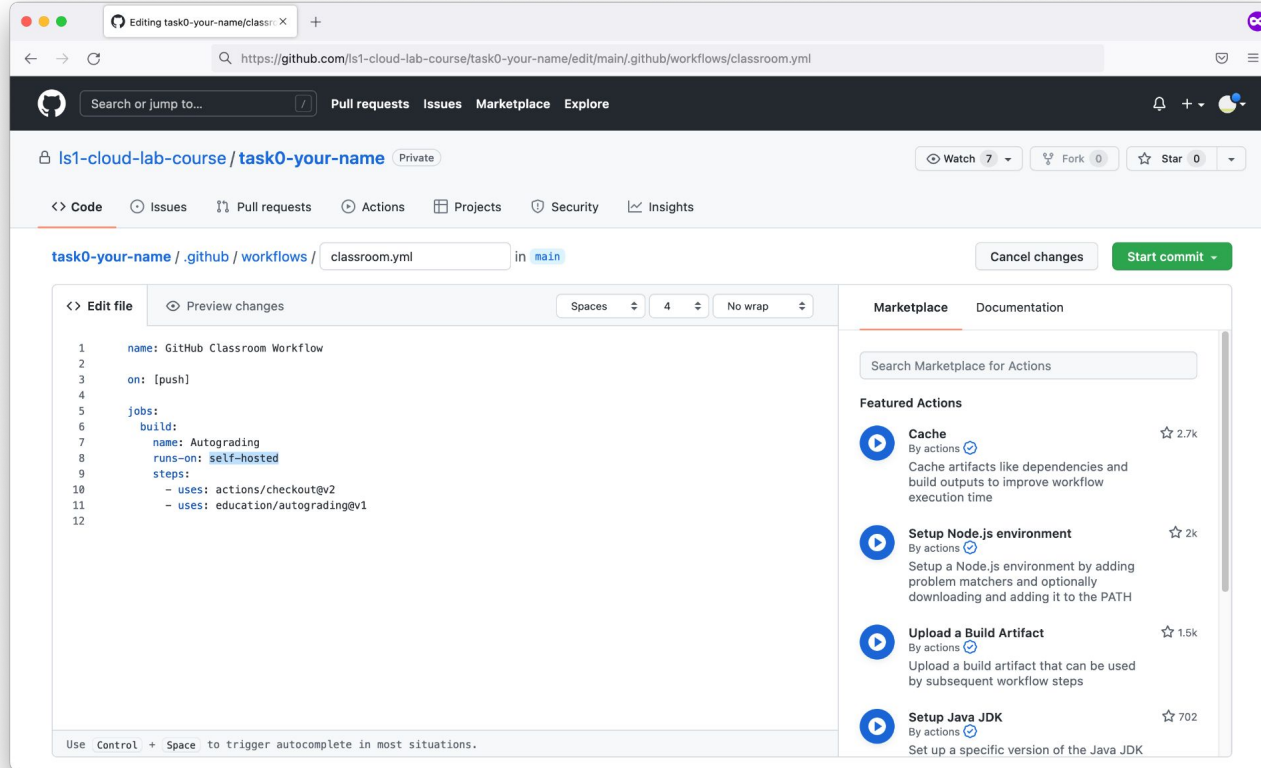


The screenshot shows a GitHub Actions workflow run page for the repository `ls1-cloud-lab-course/task0-your-name`. The workflow is named `Update GitHub Classroom Autograding Workflow`. The `Autograding` job has failed 5 minutes ago in 8s.

The workflow steps are:

- Set up job** (2s):
 - 1 Current runner version: '2.290.1'
 - 2 Operating System
 - 6 Virtual Environment
 - 11 Virtual Environment Provisioner
 - 13 GITHUB_TOKEN Permissions
 - 27 Secret source: Actions
 - 28 Prepare workflow directory
 - 29 Prepare all required actions
 - 30 Getting action download info
 - 31 Download action repository 'actions/checkout@v2' (SHA:7884fcad6b5d3d10323aee724dc68d8b9096a2e)
 - 32 Download action repository 'education/autograding@v1' (SHA:7f45625ad05fb704b54c403882a713e58a92fbb5)
- Run actions/checkout@v2** (1s)
- Run education/autograding@v1** (3s):
 - 1 Run education/autograding@v1
 - 4
 - 5 ::stop-commands::***
 - 6

Submitting your code



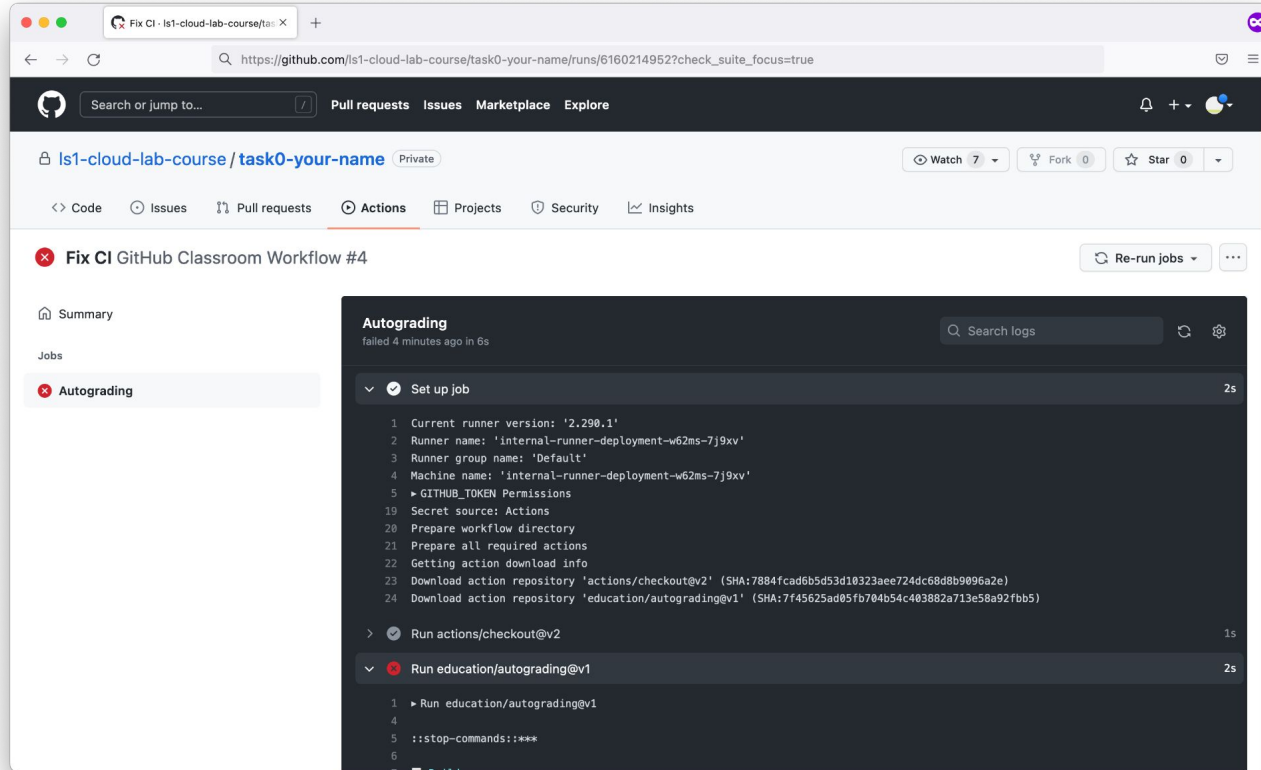
The screenshot shows a web browser window displaying a GitHub repository page. The address bar shows the URL: `https://github.com/lis1-cloud-lab-course/task0-your-name/edit/main/github/workflows/classroom.yml`. The repository name is `lis1-cloud-lab-course / task0-your-name`, marked as Private. The page has tabs for Code, Issues, Pull requests, Actions, Projects, Security, and Insights. The 'Code' tab is active, showing the file `task0-your-name / .github / workflows / classroom.yml` in the `main` branch. The file content is as follows:

```
1 name: GitHub Classroom Workflow
2
3 on: [push]
4
5 jobs:
6   build:
7     name: Autograding
8     runs-on: self-hosted
9     steps:
10       - uses: actions/checkout@v2
11       - uses: education/autograding@v1
12
```

At the bottom of the editor, a hint says: "Use Control + Space to trigger autocomplete in most situations." On the right side, there are buttons for "Cancel changes" and "Start commit". Below these, there is a "Marketplace" section with a search bar and a list of "Featured Actions":

- Cache** (2.7k stars): By actions. Cache artifacts like dependencies and build outputs to improve workflow execution time.
- Setup Node.js environment** (2k stars): By actions. Setup a Node.js environment by adding problem matchers and optionally downloading and adding it to the PATH.
- Upload a Build Artifact** (1.5k stars): By actions. Upload a build artifact that can be used by subsequent workflow steps.
- Setup Java JDK** (702 stars): By actions. Set up a specific version of the Java JDK.

Submitting your code



The screenshot shows a GitHub repository page for 'ls1-cloud-lab-course / task0-your-name'. The 'Actions' tab is selected, displaying a workflow named 'Fix CI GitHub Classroom Workflow #4'. The workflow has a red status icon and a 'Re-run jobs' button. On the left sidebar, the 'Autograding' job is highlighted with a red status icon. The main panel shows the logs for the 'Autograding' job, which failed 4 minutes ago. The logs are divided into three sections: 'Set up job' (2s), 'Run actions/checkout@v2' (1s), and 'Run education/autograding@v1' (2s). The 'Run education/autograding@v1' section shows a command to stop the process.

Fix CI GitHub Classroom Workflow #4

Summary

Jobs

- Autograding

Autograding
failed 4 minutes ago in 6s

Search logs

Set up job 2s

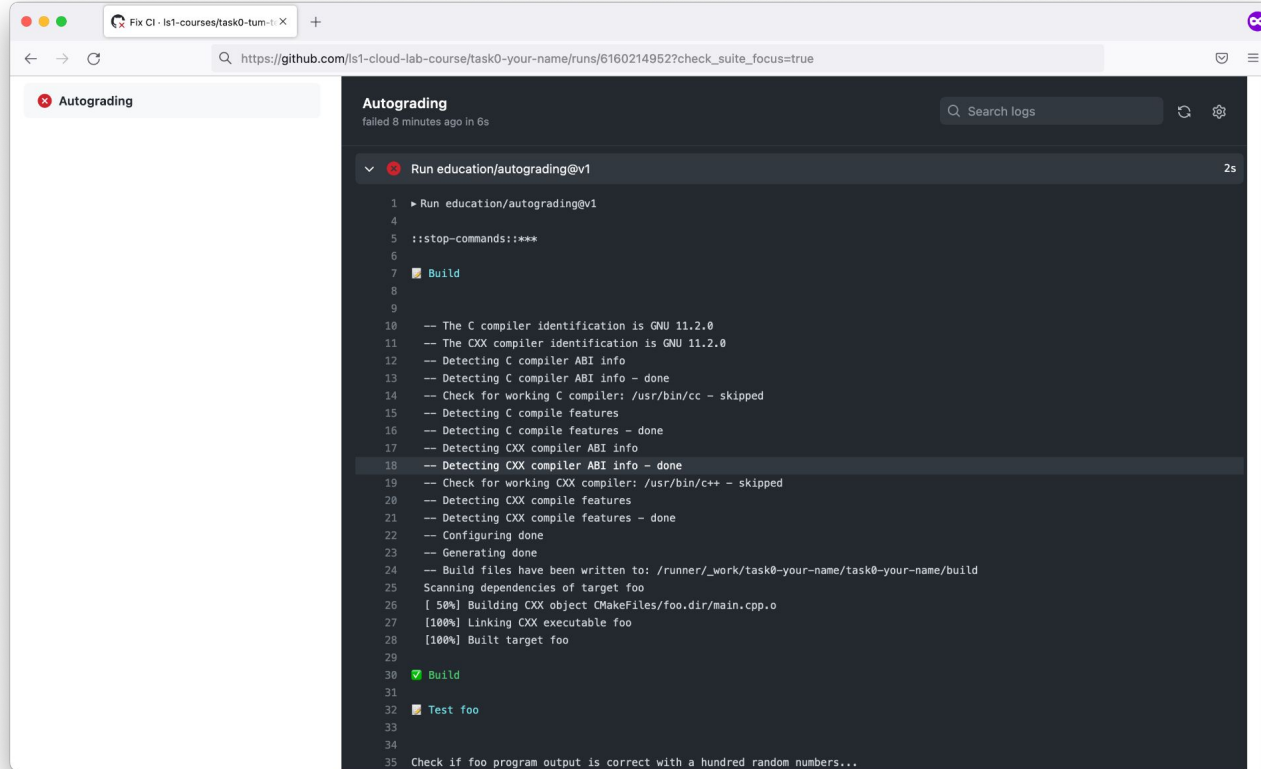
```
1 Current runner version: '2.290.1'
2 Runner name: 'internal-runner-deployment-w62ms-7j9xv'
3 Runner group name: 'Default'
4 Machine name: 'internal-runner-deployment-w62ms-7j9xv'
5 ▶ GITHUB_TOKEN Permissions
19 Secret source: Actions
20 Prepare workflow directory
21 Prepare all required actions
22 Getting action download info
23 Download action repository 'actions/checkout@v2' (SHA:7884fcad6b5d3d18323aee724dc68d8b9096a2e)
24 Download action repository 'education/autograding@v1' (SHA:7f45625ad05fb704b54c483882a713e58a92fbb5)
```

Run actions/checkout@v2 1s

Run education/autograding@v1 2s

```
1 ▶ Run education/autograding@v1
4
5 ::stop-commands::***
6
```

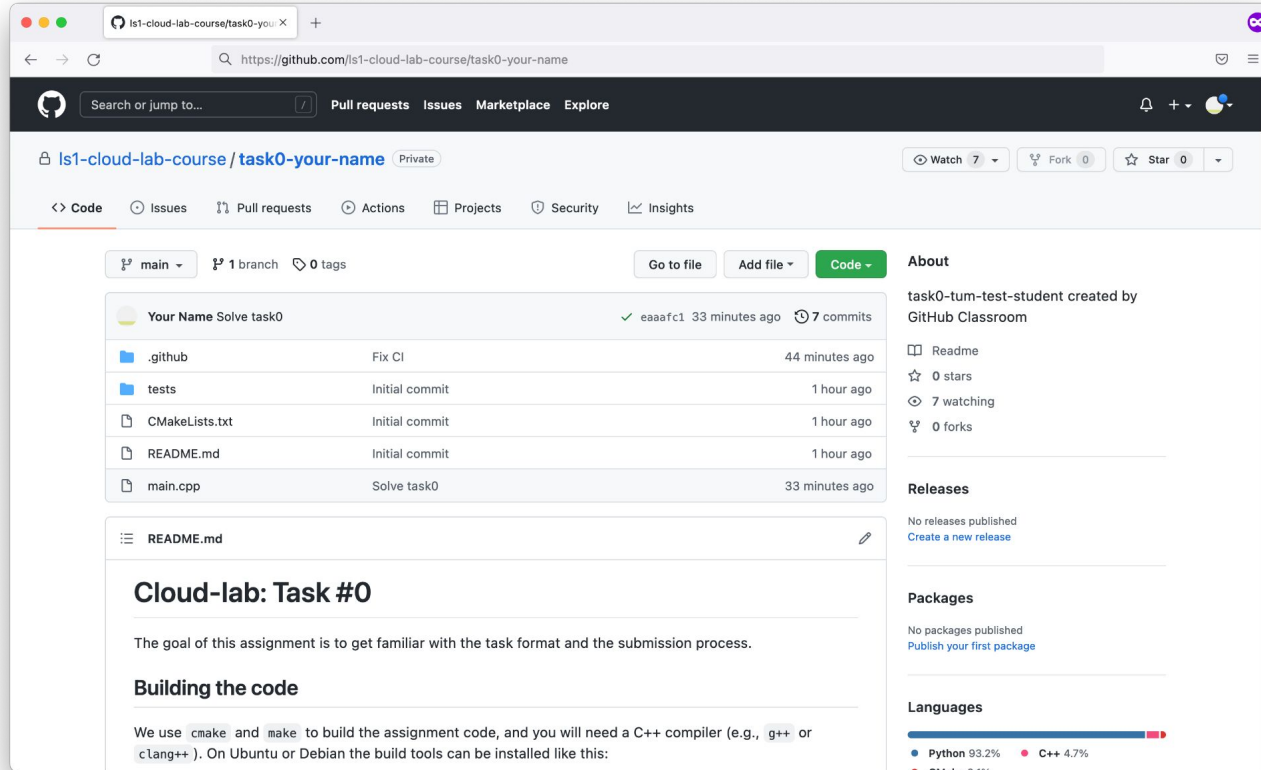
Submitting your code



The screenshot shows a web browser window displaying a GitHub Actions workflow run. The browser's address bar shows the URL: `https://github.com/lis1-cloud-lab-course/task0-your-name/runs/6160214952?check_suite_focus=true`. The page title is "Autograding" with a red status icon and the text "failed 8 minutes ago in 6s". On the left sidebar, there is a tab labeled "Autograding" with a red status icon. The main content area shows the workflow steps for "Run education/autograding@v1". The steps are listed with line numbers 1 through 35. The "Build" step (lines 7-30) is highlighted with a green checkmark, indicating it passed. The "Test foo" step (lines 32-35) is highlighted with a red status icon, indicating it failed. The output of the "Build" step shows the compilation of a C++ program named "foo".

```
1  ▶ Run education/autograding@v1
2
3  ::stop-commands::***
4
5  📦 Build
6
7  -- The C compiler identification is GNU 11.2.0
8  -- The CXX compiler identification is GNU 11.2.0
9  -- Detecting C compiler ABI info
10 -- Detecting C compiler ABI info - done
11 -- Check for working C compiler: /usr/bin/cc - skipped
12 -- Detecting C compile features
13 -- Detecting C compile features - done
14 -- Detecting CXX compiler ABI info
15 -- Detecting CXX compiler ABI info - done
16 -- Check for working CXX compiler: /usr/bin/c++ - skipped
17 -- Detecting CXX compile features
18 -- Detecting CXX compile features - done
19 -- Configuring done
20 -- Generating done
21 -- Build files have been written to: /runner/_work/task0-your-name/task0-your-name/build
22 Scanning dependencies of target foo
23 [ 50%] Building CXX object CMakeFiles/foo.dir/main.cpp.o
24 [100%] Linking CXX executable foo
25 [100%] Built target foo
26
27 🟢 Build
28
29 📦 Test foo
30
31 Check if foo program output is correct with a hundred random numbers...
```

Submitting your code



The screenshot shows a GitHub repository page for 'ls1-cloud-lab-course/task0-your-name'. The repository is private and was created by 'task0-tum-test-student created by GitHub Classroom'. It has 7 commits, 7 watchers, and 0 forks. The repository contains a file tree with the following files and their commit times:

File	Commit Time
.github	Fix CI 44 minutes ago
tests	Initial commit 1 hour ago
CMakeLists.txt	Initial commit 1 hour ago
README.md	Initial commit 1 hour ago
main.cpp	Solve task0 33 minutes ago

The README.md file is displayed below the file tree. It contains the following text:

Cloud-lab: Task #0

The goal of this assignment is to get familiar with the task format and the submission process.

Building the code

We use `cmake` and `make` to build the assignment code, and you will need a C++ compiler (e.g., `g++` or `clang++`). On Ubuntu or Debian the build tools can be installed like this:

The right sidebar of the repository page shows the following information:

- About:** task0-tum-test-student created by GitHub Classroom. Includes links for Readme, 0 stars, 7 watching, and 0 forks.
- Releases:** No releases published. Link: [Create a new release](#).
- Packages:** No packages published. Link: [Publish your first package](#).
- Languages:** A bar chart showing the language distribution: Python 93.2%, C++ 4.7%, and CMake 2.1%.

Containers

Motivation

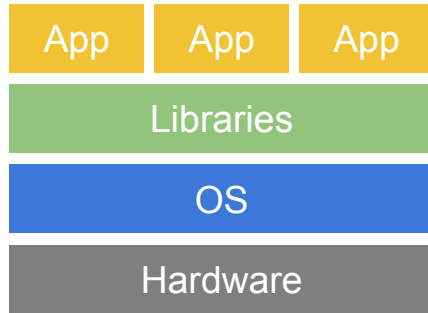
Share resources (IaaS, PaaS, ...) in a flexible and cost-effective way

→ split applications into microservices

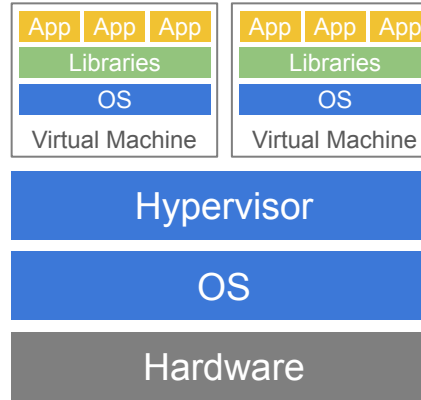
Advantages:

- Greater hardware resource-utilization
- Simply maintainable
- Scalable

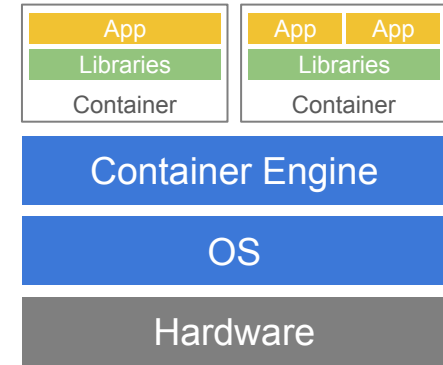
Containers



Traditional system



Virtual machines

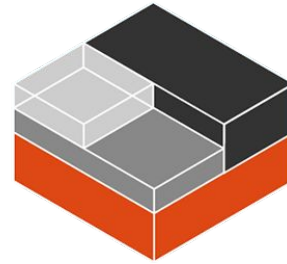
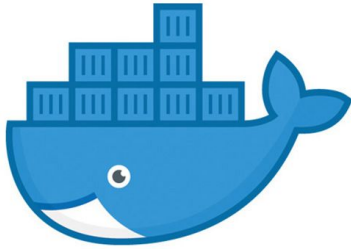


Containers

Unlike virtual machines ...

- containers often contain only one application
- containers do not include an operating system (OS)
- containers limit access to resources through host OS mechanisms

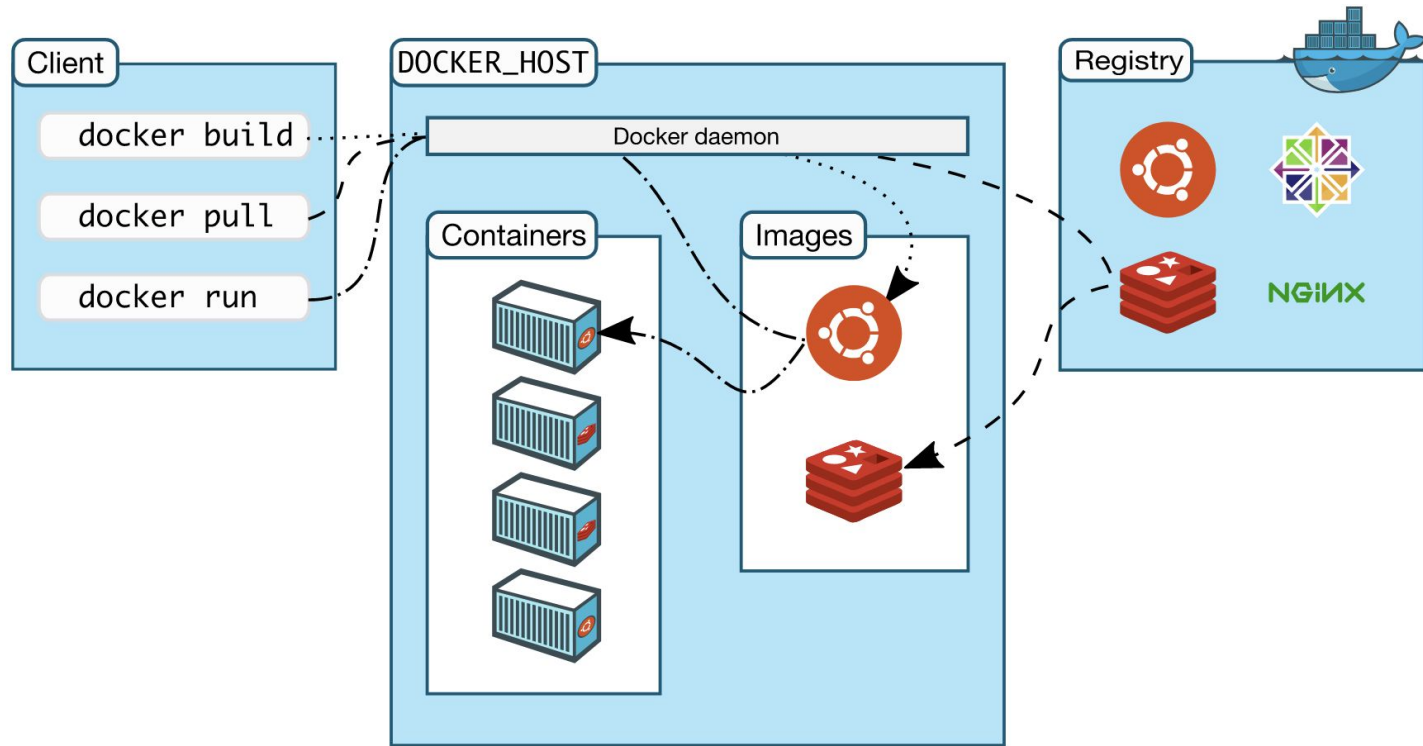
Container solutions



LXC

- **Application** container engine
 - Unlike LXC which is a **system** container engine
- Uses the resource isolation mechanisms of the Linux kernel
 - Namespaces
 - Cgroups
 - Layered filesystems
- Consists of three components: Software, objects and registry

Docker



Getting started

1. Install Docker on your system, e.g.: `# apt install docker-ce`
2. Run some container: `$ docker run -it ubuntu /bin/bash`

Hints:

- List all containers: `$ docker ps -a`
- Get a shell for a running container: `$ docker exec -it <container name> /bin/sh`

Getting started

1. Create a simple Dockerfile (see [Docker's docs](#))
2. Build an image based on the Dockerfile: `$ docker build -t <image name> .`
3. Run the container: `$ docker run -d <image name>`
4. Create and run multiple containers using [Docker compose](#)

Hints:

- Force clean rebuild of image: `$ docker build --no-cache -t <image name> .`
- To prevent a container from exiting early when debugging, add `CMD tail -f /dev/null` to end of Dockerfile

First assignment

Your assignment for the next two weeks

- Task to get familiar with the submission process
 - Will be graded like all other assignments
- Write a client/server application using Linux's socket API
 - Detailed description of what to do in the task repository
- Let the server run in a Docker container
 - Write a [Dockerfile](#) for the server
- Invitation link for the assignment will be handed out via Slack
 - Make sure that you are part of the channel [#ss-22-cloud-lab!](#)