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

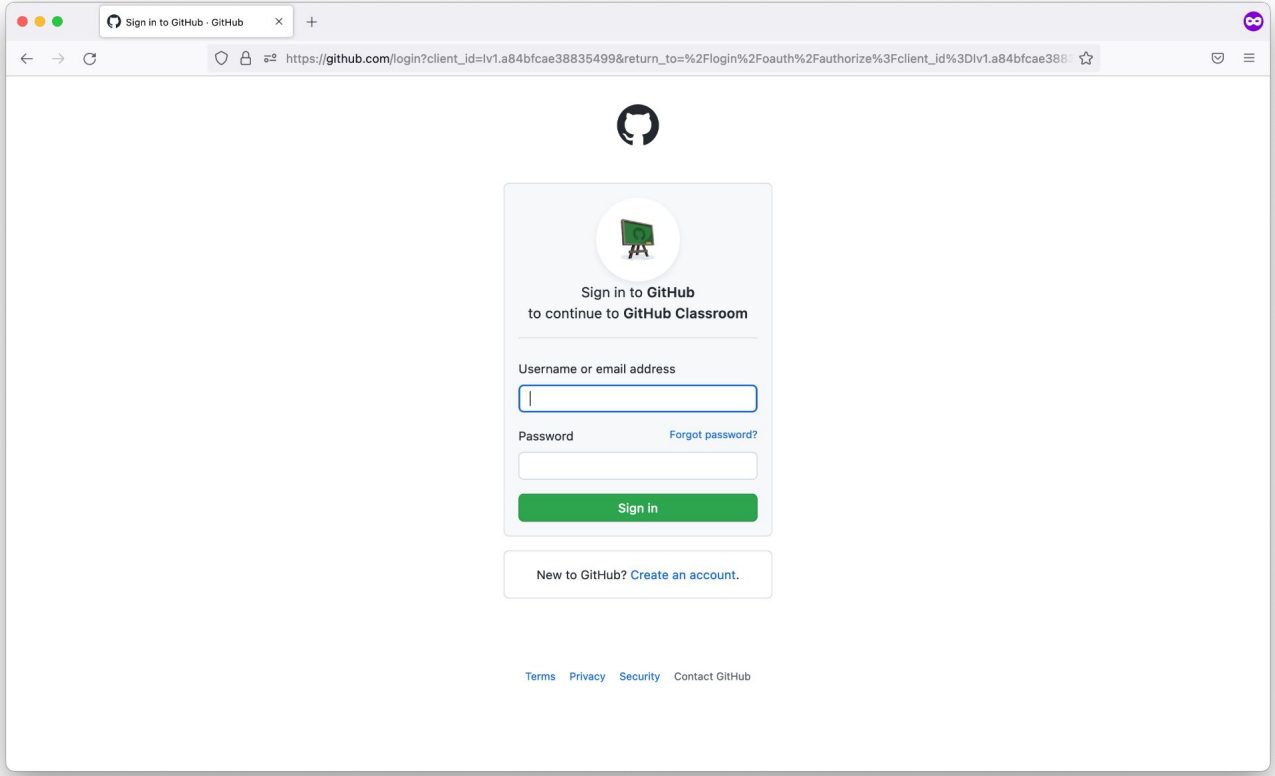
The assignments

- 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

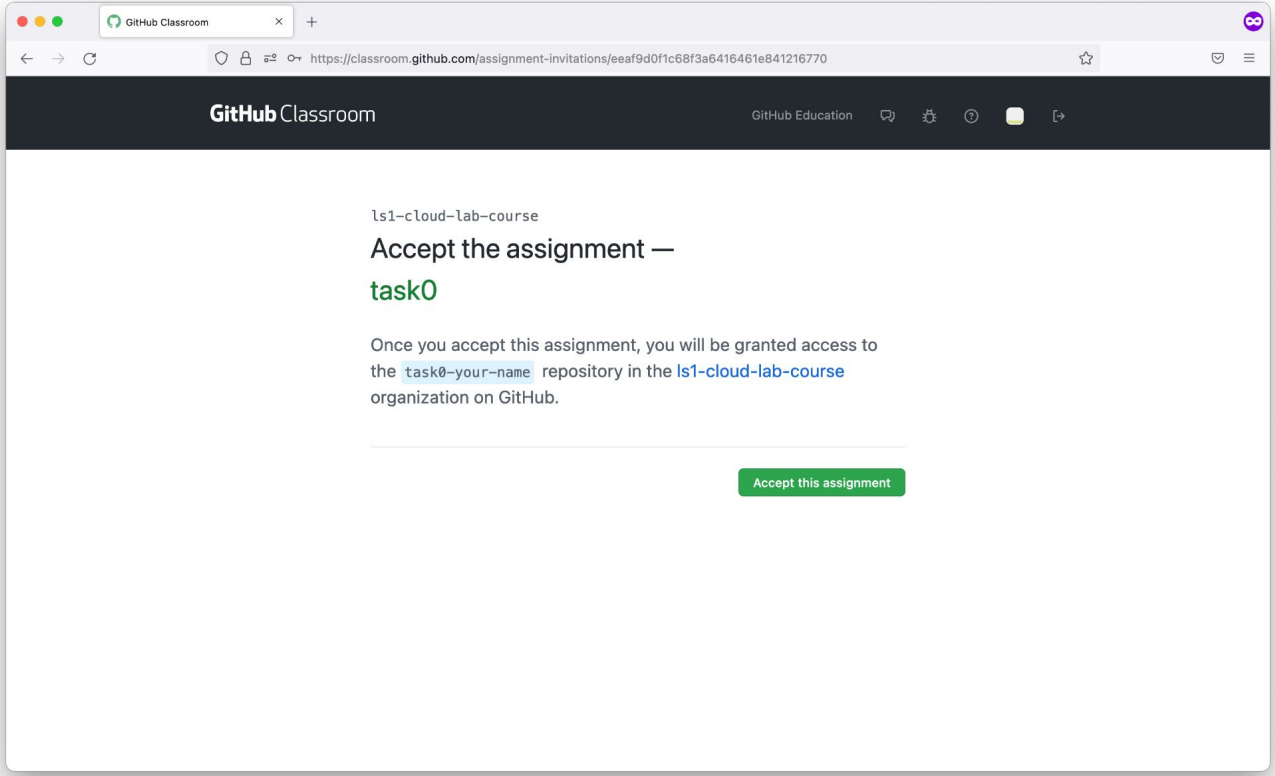
Our CI runners

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

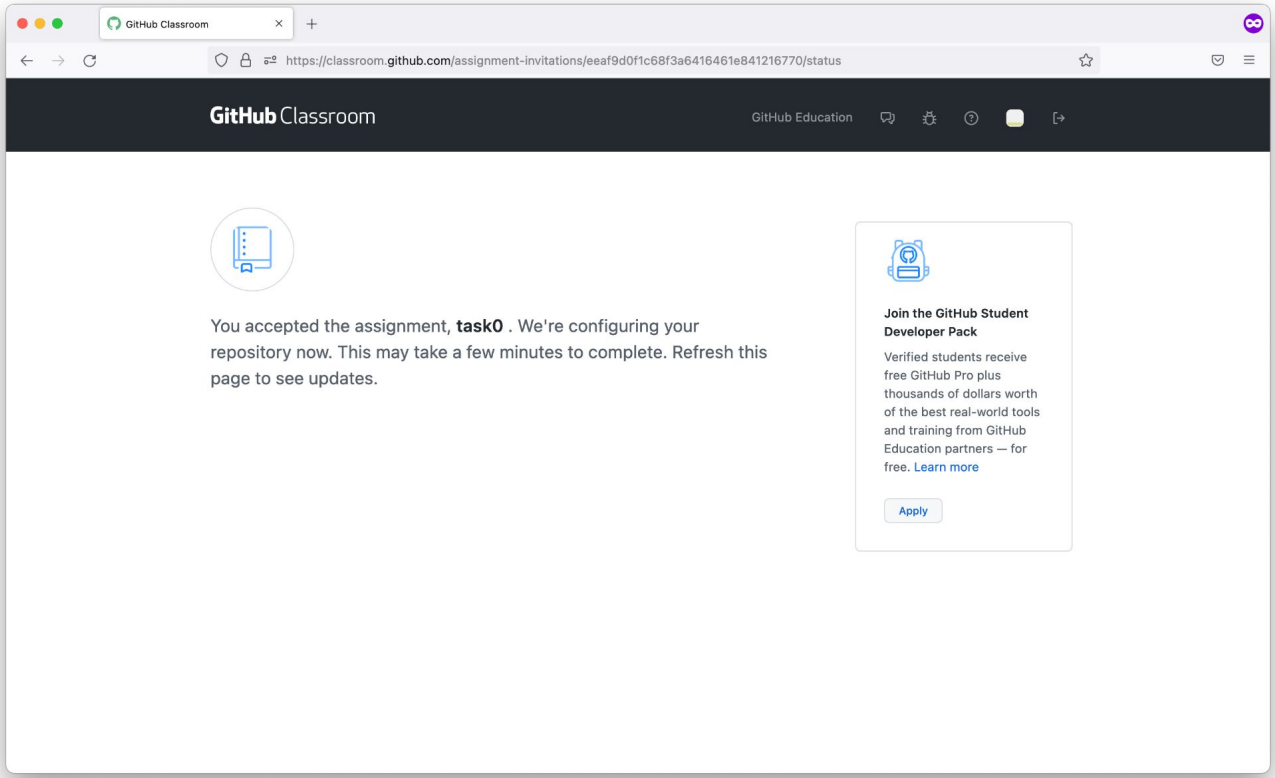
Getting the assignment code



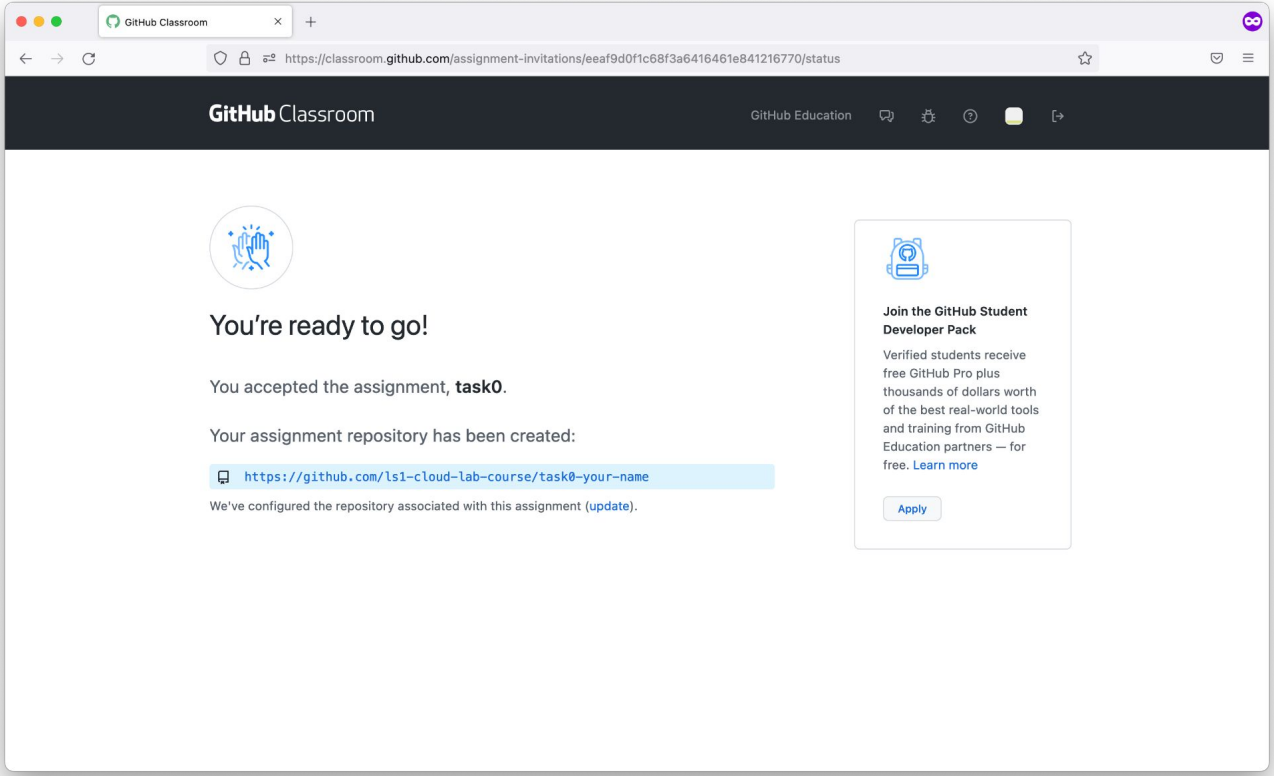
Getting the assignment code



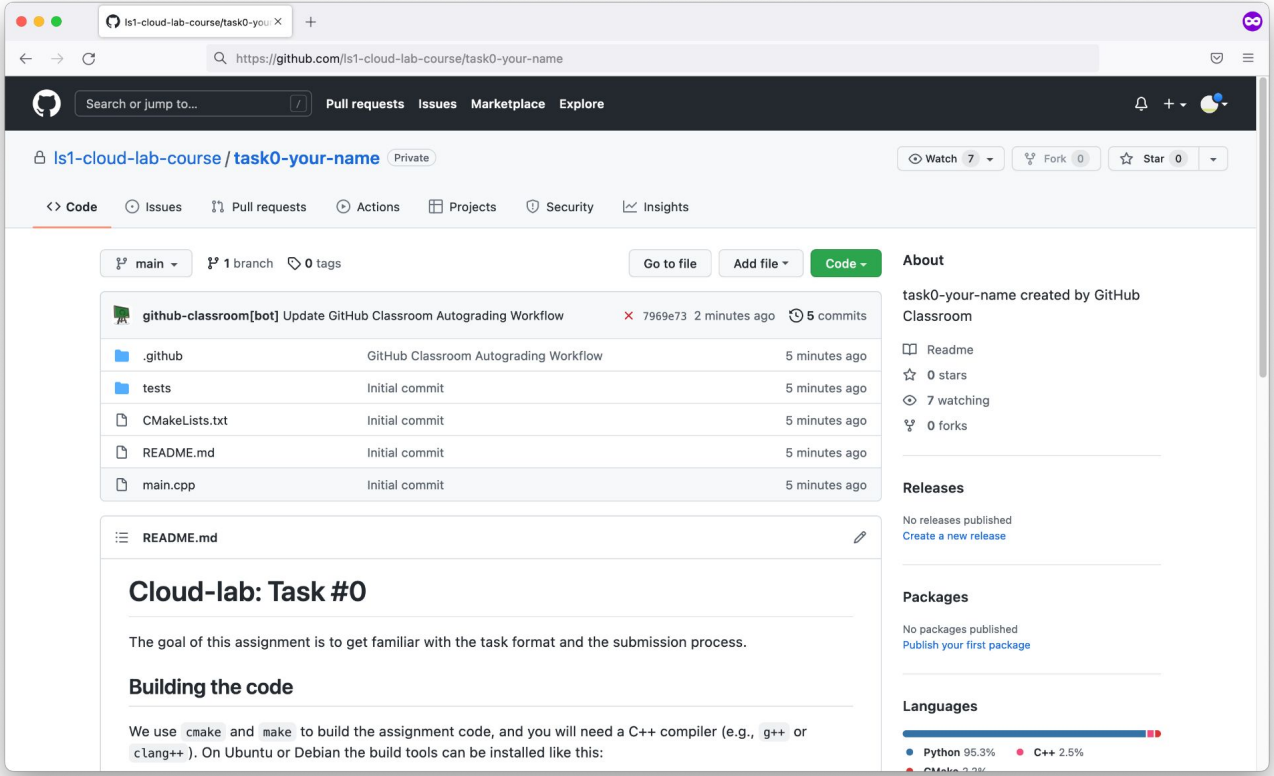
Getting the assignment code



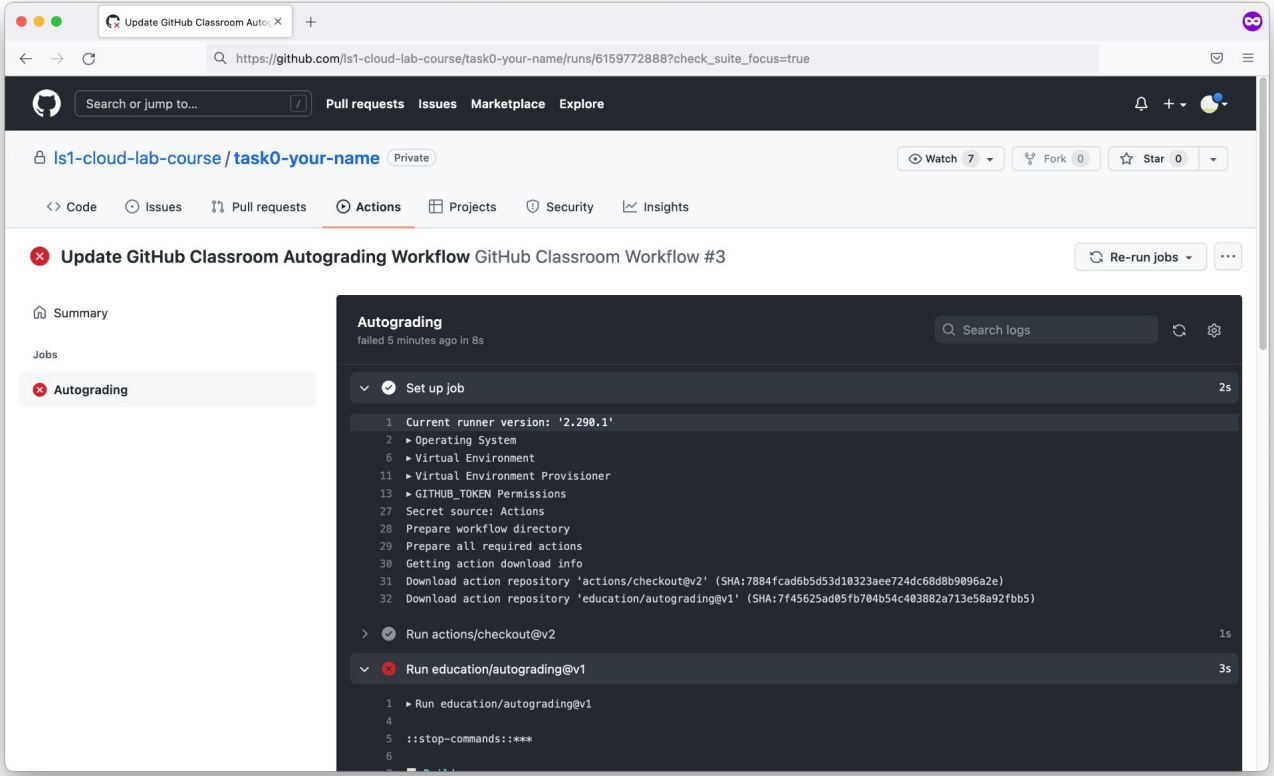
Getting the assignment code



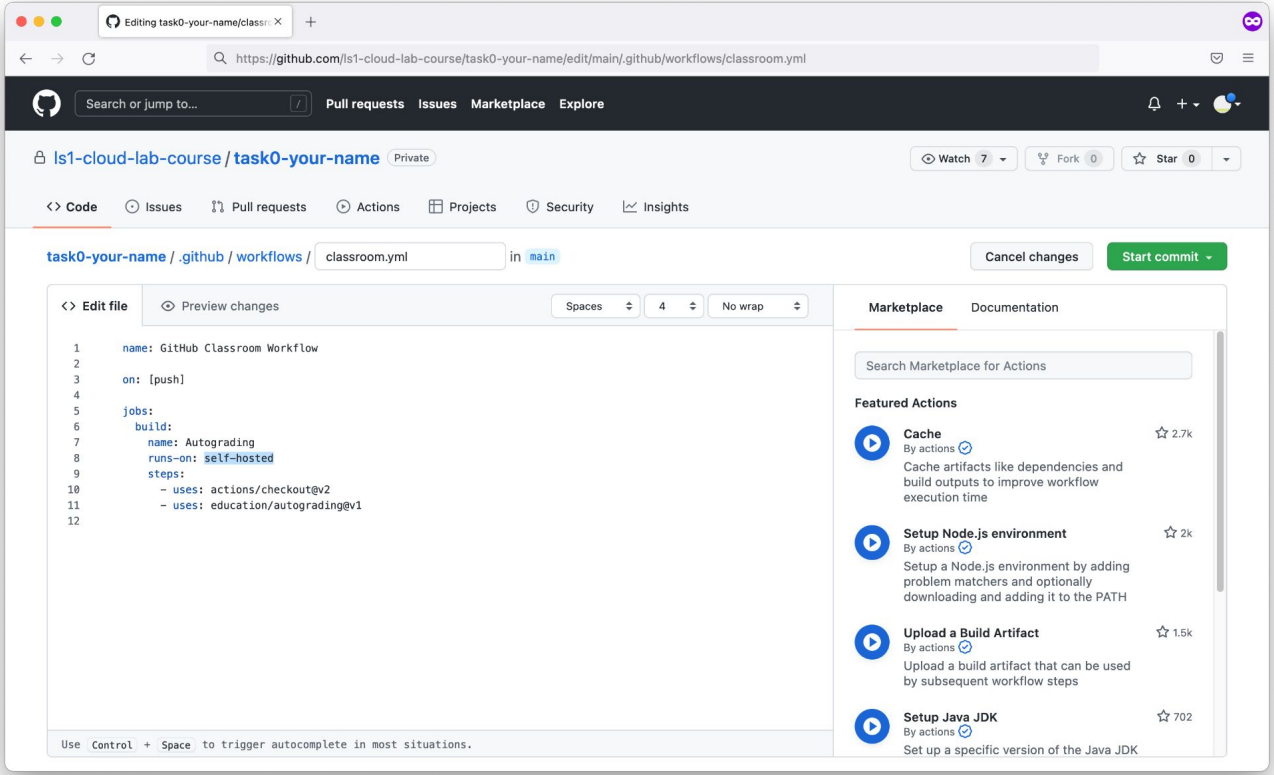
Getting the assignment code



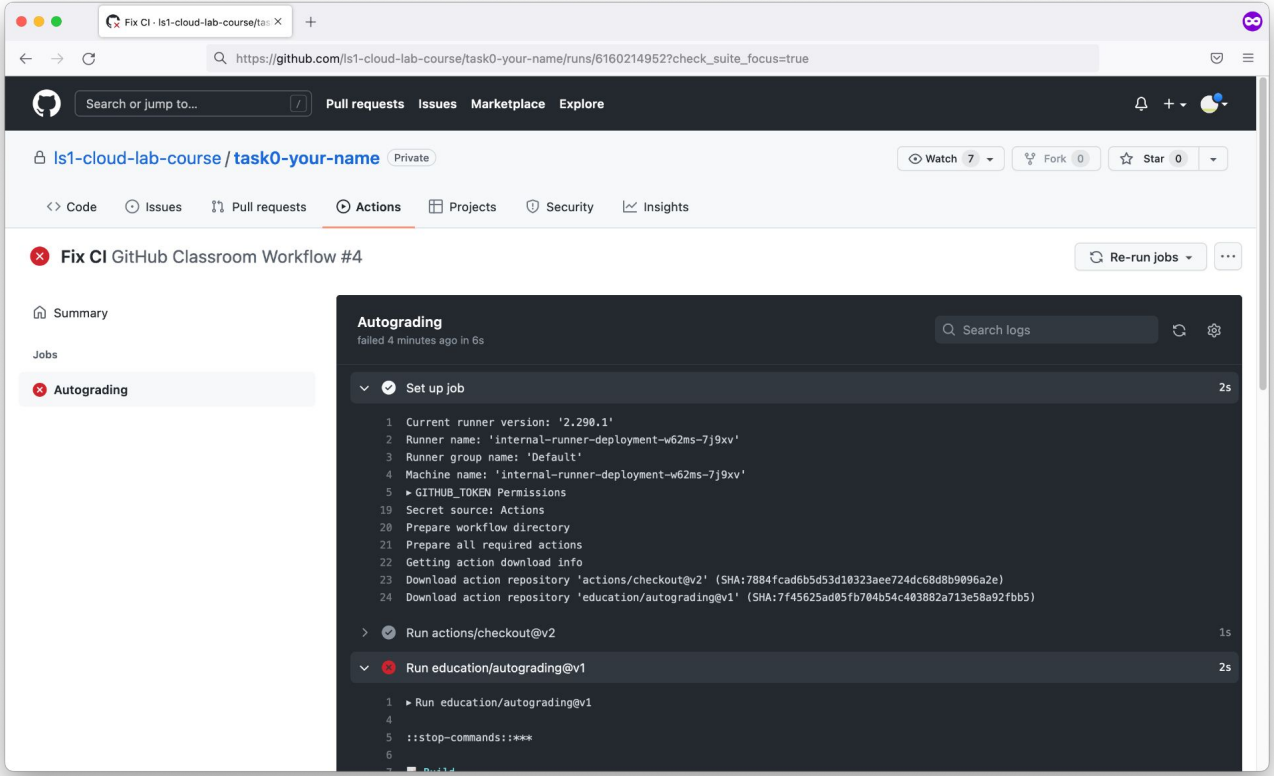
Submitting your code



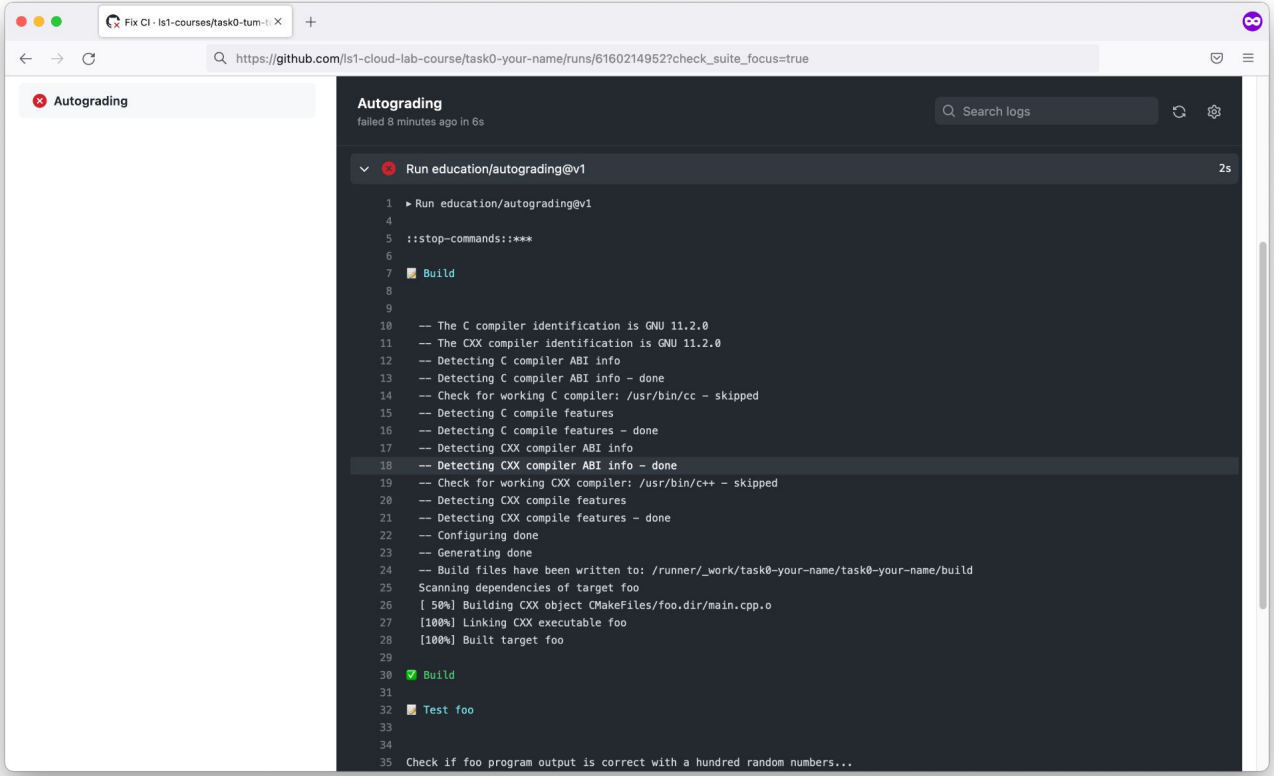
Submitting your code



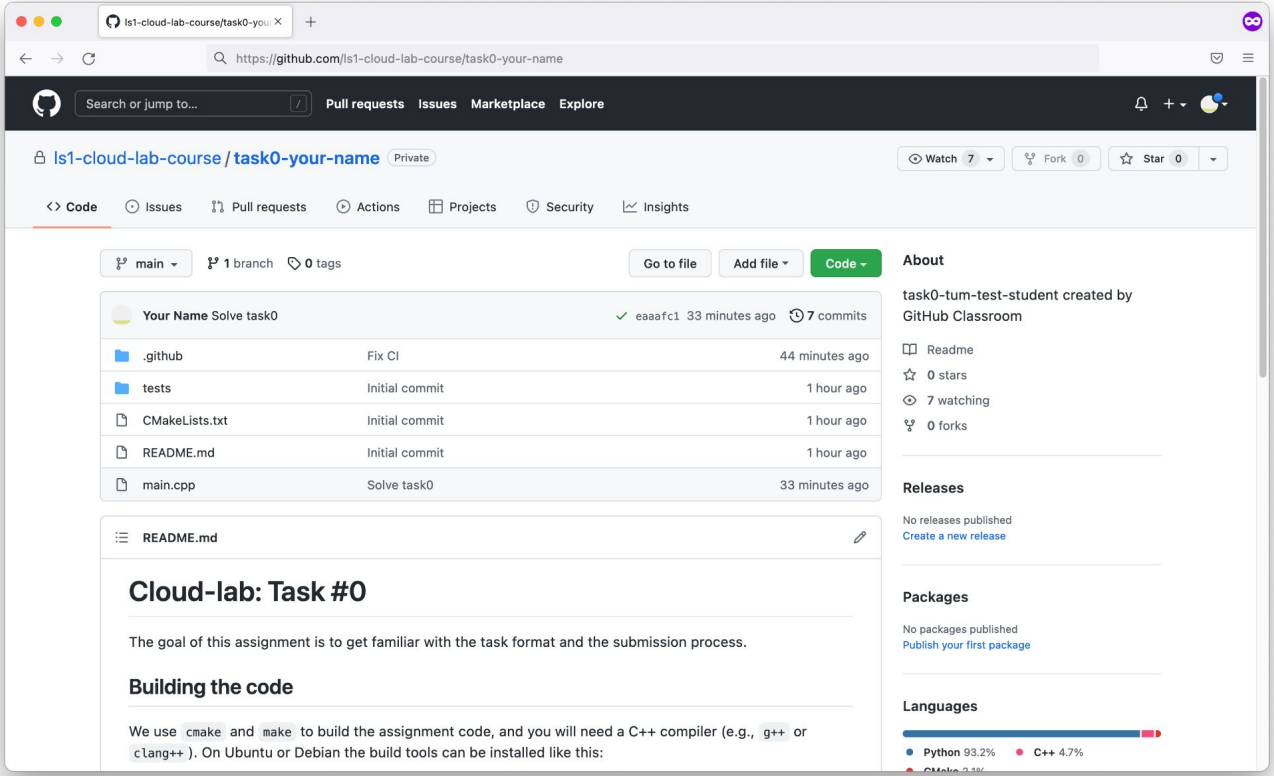
Submitting your code



Submitting your code



Submitting your code



Containers

Motivation

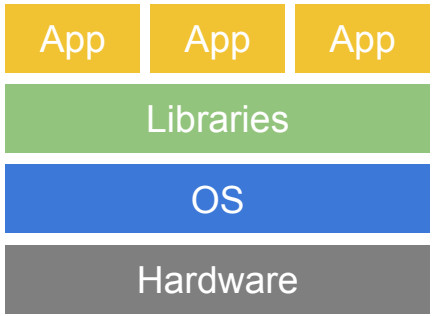
Share resources (IaaS, PaaS, …) in a flexible and cost-effective way

→ split applications into microservices

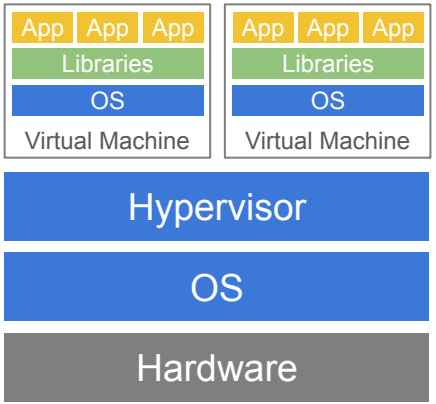
Advantages:

- Greater hardware resource-utilization
- Simply maintainable
- Scalable

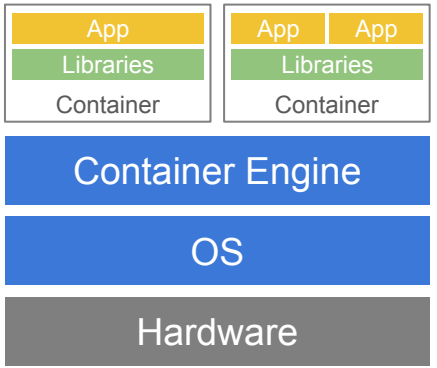
Containers



Traditional system



Virtual machines



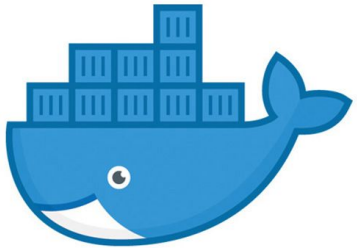
Containers

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

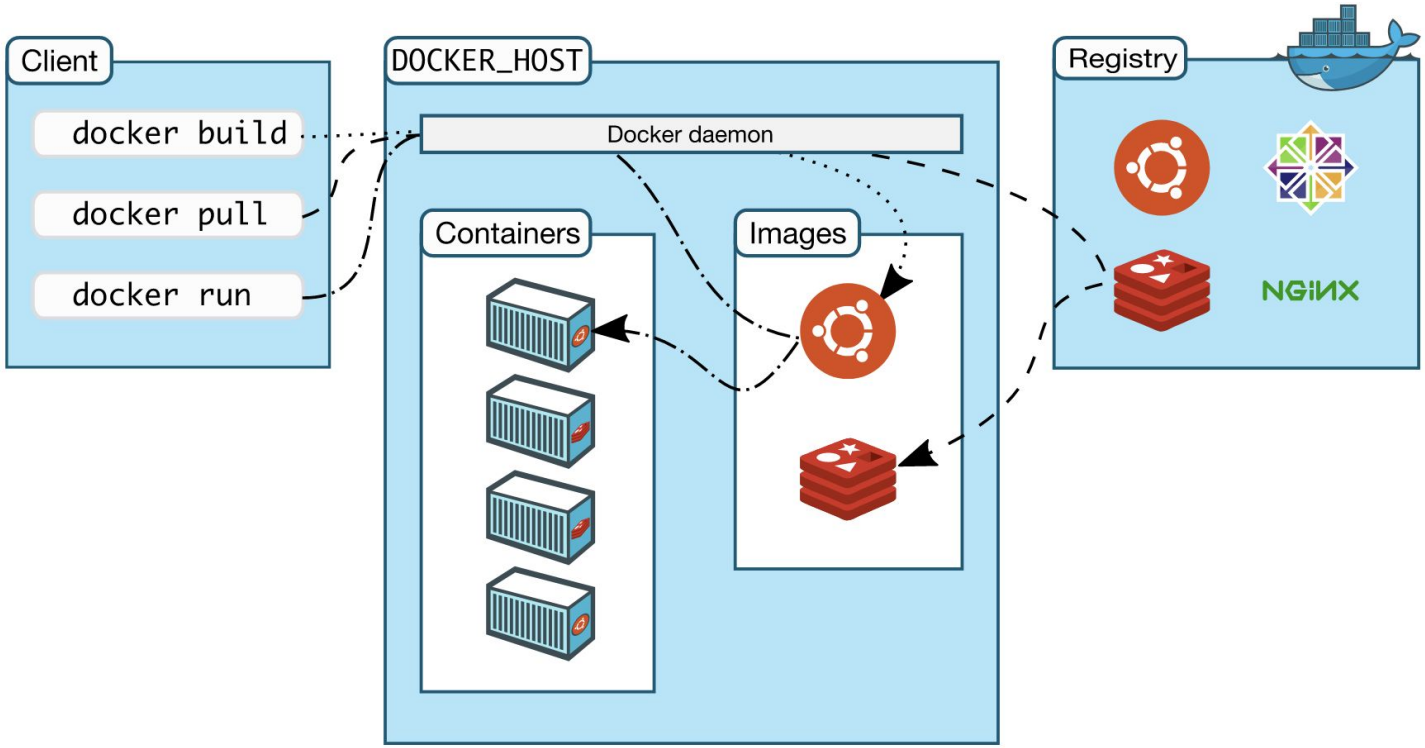
Container solutions



Docker

- 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


Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

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!