Virtualization II WS 2025/26
Dr. Torsten Harenberg – Dr. Marisa Sandhoff

Exercise 2 - multi-stage containers
Turn in: Monday 17.11.2025 until 08:00 **in Moodle**

Teamwork:

- form groups of two (2) to three (3) students.
- please file only **one** submission in Moodle for your whole group. Mark clearly the group member's names and student ID numbers on your submission.

(1) Writing your own container (20p)

You are tasked with containerizing a simple Monte Carlo simulation for calculating π. The application is written in C++ and should be deployed in the smallest possible Docker image.

The simulation code is given as follows:

```
#include <iostream>
#include <random>
#include <iomanip>

int main(int argc, char* argv[]) {
    long long iterations = 10000000;

    if (argc > 1) {
        iterations = std::stoll(argv[1]);
    }

    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_real_distribution<> dis(0.0, 1.0);

    long long inside_circle = 0;

    std::cout << "Running Monte Carlo simulation with "
              << iterations << " iterations..." << std::endl;
```

```
    for (long long i = 0; i < iterations; ++i) {
        double x = dis(gen);
        double y = dis(gen);

        if (x*x + y*y <= 1.0) {
            inside_circle++;
        }
    }

    double pi_estimate = 4.0 * inside_circle / iterations;

    std::cout << std::fixed << std::setprecision(10);
    std::cout << "Estimated π: " << pi_estimate << std::endl;
    std::cout << "Actual π:    " << M_PI << std::endl;
    std::cout << "Error:       " << std::abs(pi_estimate -
M_PI) << std::endl;

    return 0;
}
```

Tasks:

1. **Single-Stage Dockerfile (Baseline)** Create a simple Dockerfile that compiles the code and makes it executable. Use `gcc:latest` as the base image.

   Hand in: the Dockerfile and the output of the build and run commands. (5p)

2. **Multi-Stage Dockerfile** Optimize the Dockerfile using a multi-stage build:
   Build Stage: Compile the program with all necessary tools
   Runtime Stage: Use a minimal base image (e.g., `alpine:latest` or `debian:stable-slim`)

   Hand in: the Dockerfile and the output of the build and run commands. Comment difficulties that may have occurred. (9p)

3. Compare the image sizes of both variants (`docker images`) and document the size difference. What components remain/are removed in each stage.

   Hand in: commands you used and their output (2p)

4. Add a test stage that runs the program with 1000 iterations and checks if the result is within the range $2.5 < \pi < 4.0$

   Hand in: new Dockerfile. (4p)