Exercise 2 Report

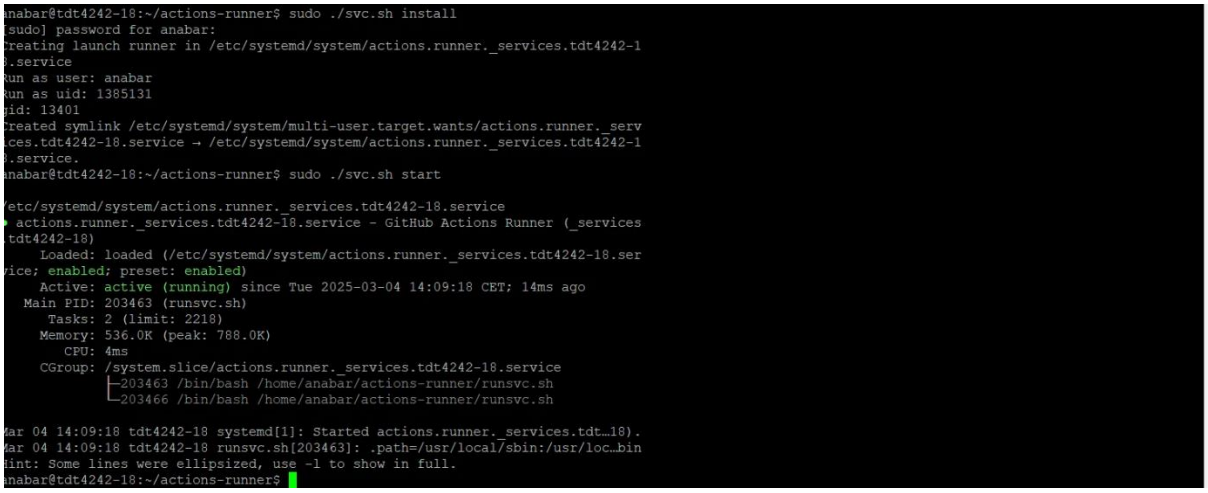
This is a thorough report of the process of familiarizing ourselves with DevOps and GitHub Actions by setting up the project in the server and configuring a workflow for continuous development

Task 1:

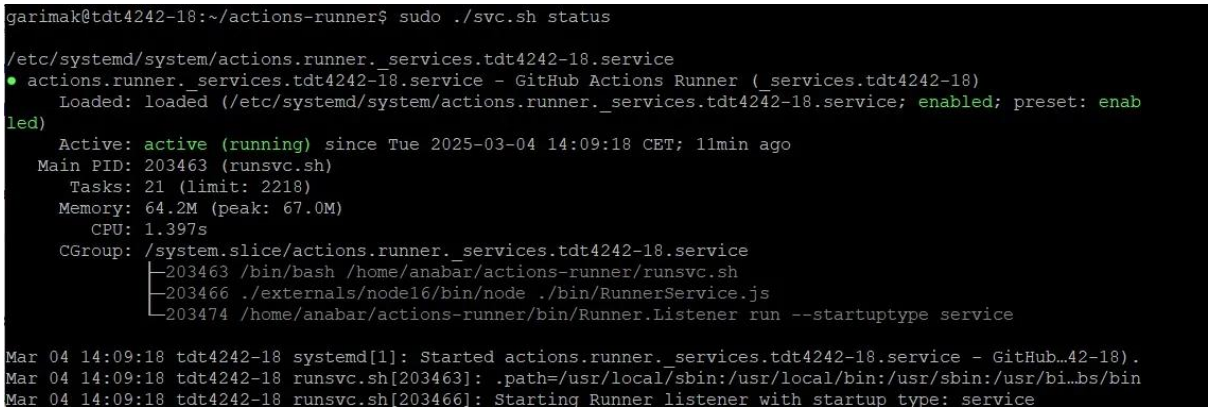
To begin with, we accessed the server by establishing an ssh connection with Putty. Once there we could set up a runner and run it:

```
Enter the name of runner: [press Enter for tdt4242-18]

This runner will have the following labels: 'self-hosted', 'Linux', 'X64'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip]

√ Runner successfully added
√ Runner connection is good

# Runner settings

Enter name of work folder: [press Enter for _work]

√ Settings Saved.

anabar@tdt4242-18:~/actions-runner$ ./run.sh

√ Connected to GitHub

Current runner version: '2.319.1'
2025-03-04 13:08:25Z: Listening for Jobs
^CExiting...
Runner listener exit with 0 return code, stop the service, no retry needed.
Exiting runner...
```

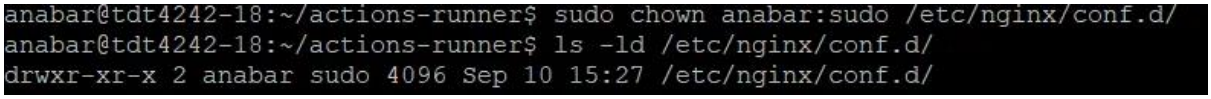
Once we checked it was properly running and could exit without problems, we configured the runner as a service to automatically start the runner application when the machine starts. To do so, we followed the steps in GitHub Actions:



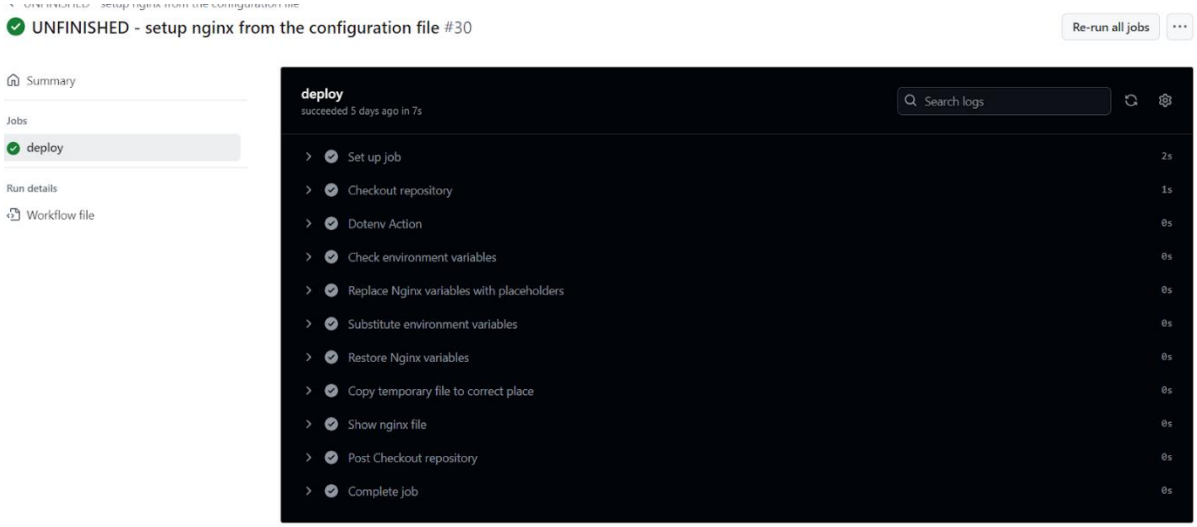
This took us a while because of some configuration mistakes that we were making, but after it worked we did a couple of tests to check if everything was working as it should, like stopping and restarting and checking the status with another user at the server:



Afterwards, we modified the file given as a template to test the deployment for it to actually work. The changes can be seen in the repository, but the two main changes were : 1. Related to the variable instances, that weren't handled correctly; and 2. Related to the permissions. We had to give the user permissions to be able to create the file where the contents were copied:



Once we figured all the errors that the deployment was giving we were able to have a proper configuration (checked that it could reload and the nginx configuration works):




```
gargik@tdt4242-18:~/actions-runner$ sudo systemctl reload nginx
[sudo] password for gargik:
gargik@tdt4242-18:~/actions-runner$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
gargik@tdt4242-18:~/actions-runner$ ^[[200~sudo systemctl reload nginx
sudo: command not found
gargik@tdt4242-18:~/actions-runner$ ~sudo systemctl reload nginx
sudo systemctl reload nginx
Command '~sudo' not found, did you mean:
  command 'sudo' from deb sudo (1.9.14p2-1ubuntu1)
  command 'sudo' from deb sudo-ldap (1.9.14p2-1ubuntu1)
Try: apt install <deb name>
gargik@tdt4242-18:~/actions-runner$
gargik@tdt4242-18:~/actions-runner$ sudo systemctl reload nginx
gargik@tdt4242-18:~/actions-runner$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: en>
     Active: active (running) since Wed 2025-02-26 06:23:10 CET; 6 days ago
       Docs: man:nginx(8)
    Process: 213858 ExecReload=/usr/sbin/nginx -g daemon on; master_process on;>
   Main PID: 116928 (nginx)
      Tasks: 2 (limit: 2218)
     Memory: 2.9M (peak: 5.3M swap: 728.0K swap peak: 1020.0K)
        CPU: 125ms
     CGroup: /system.slice/nginx.service
             ├─116928 "nginx: master process /usr/sbin/nginx -g daemon on; mast>
             └─213860 "nginx: worker process"

Mar 04 17:40:29 tdt4242-18 systemd[1]: Reloaded nginx.service - A high performa>
Mar 04 17:40:31 tdt4242-18 systemd[1]: Reloading nginx.service - A high perform>
Mar 04 17:40:31 tdt4242-18 nginx[213822]: 2025/03/04 17:40:31 [notice] 213822#2>
Mar 04 17:40:31 tdt4242-18 systemd[1]: Reloaded nginx.service - A high performa>
Mar 04 17:41:33 tdt4242-18 systemd[1]: Reloading nginx.service - A high perform>
Mar 04 17:41:33 tdt4242-18 nginx[213851]: 2025/03/04 17:41:33 [notice] 213851#2>
Mar 04 17:41:33 tdt4242-18 systemd[1]: Reloaded nginx.service - A high performa>
Mar 04 17:41:35 tdt4242-18 systemd[1]: Reloading nginx.service - A high perform>
Mar 04 17:41:35 tdt4242-18 nginx[213858]: 2025/03/04 17:41:35 [notice] 213858#2>
Mar 04 17:41:35 tdt4242-18 systemd[1]: Reloaded nginx.service - A high performa>
lines 1-23...skipping...
● nginx.service - A high performance web server and a reverse proxy server
```

Task 2:

Changes made:

We have made changes in the section defining the trigger events for our workflow. Since we worked within the main branch mostly, we only included the main branch. Therefore, whenever a change is committed to the main branch, the workflow will be triggered.

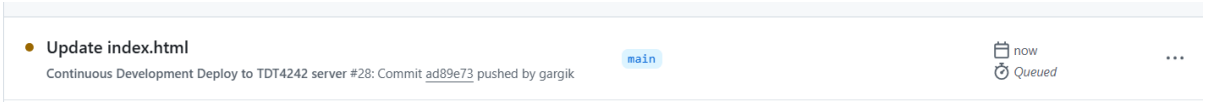
```yaml
name: Continuous Development Deploy to TDT4242 server

on:
  push:
    branches:
      - main # Or your development branch
  workflow_dispatch:
```

Proof of Working:

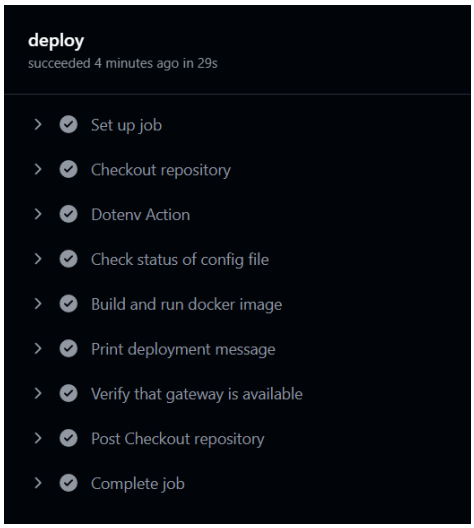
Step 1: We changed the title of the main html page from SecFit to SecFit- Group 18.

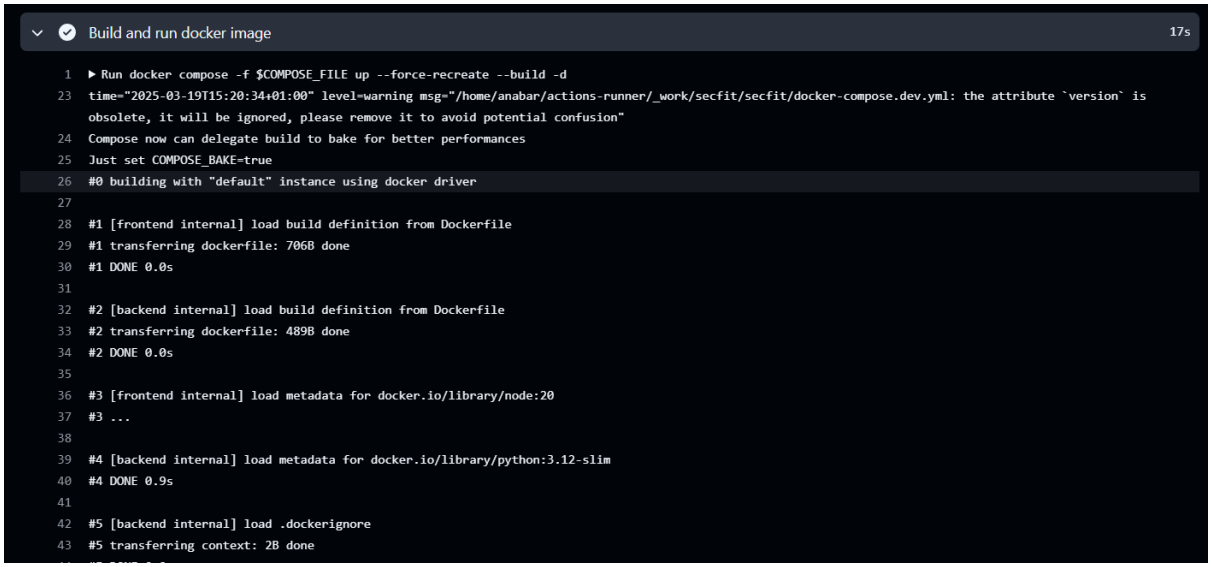
```
<title>SecFit-Group 18</title>
```

Step 2: Upon committing the change, we observed that the workflow was executed automatically.



Step 3: We checked the steps carried out- our docker images were built when updated. The docker image always ran the updated code.



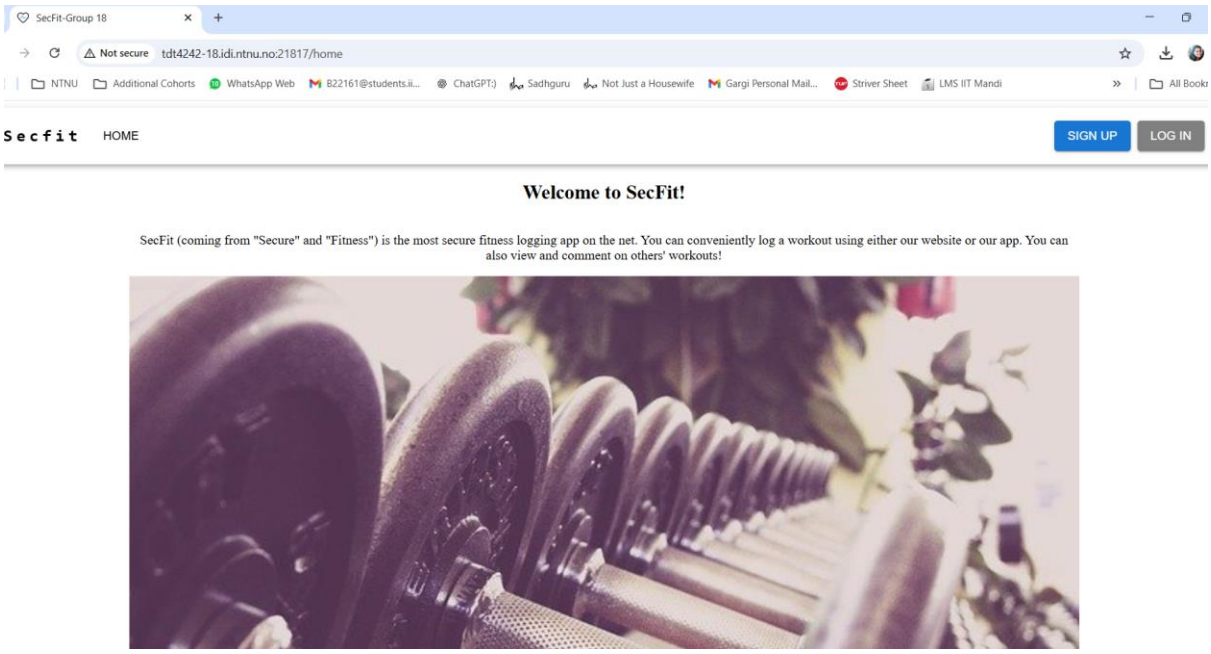



```
#24 [frontend] exporting to image
#24 exporting layers 0.2s done
#24 writing image sha256:4800927bd0652831f2e14c1c3b9800c693032cbaceb80333a15a2c92bf6c1989
#24 writing image sha256:4800927bd0652831f2e14c1c3b9800c693032cbaceb80333a15a2c92bf6c1989 done
 backend  Built
 frontend  Built
#24 naming to docker.io/library/secfit_dev-frontend done
#24 DONE 0.2s

#25 [frontend] resolving provenance for metadata file
#25 DONE 0.0s
 Container secfit_dev_gateway  Stopping
 Container secfit_dev_gateway  Stopping
 Container secfit_dev_gateway  Stopped
 Container secfit_dev_gateway  Stopped
 Container secfit_dev_backend  Recreate
 Container secfit_dev_frontend  Recreate
 Container secfit_dev_backend  Recreated
 Container secfit_dev_frontend  Recreated
 Container secfit_dev_gateway  Recreate
 Container secfit_dev_gateway  Recreated
 Container secfit_dev_frontend  Starting
 Container secfit_dev_backend  Starting
```

We checked the docker images in the Putty terminal. We find the docker image corresponding to our latest change in the frontend since the change was made to index.html in frontend directory.

```
gargik@tdt4242-18:~/actions-runner$ docker images
REPOSITORY            TAG       IMAGE ID       CREATED          SIZE
secfit_dev-frontend   latest    4800927bd065   12 minutes ago   2.08GB
<none>                <none>    6768e7175f3a   2 hours ago      2.08GB
```

Step 4: We checked the website and saw that the update in title is reflected there.



The server http://tdt4242-18.idi.ntnu.no:21817/home runs the updated code. This was achieved through the workflow which creates docker images whenever there is a push to the main branch.

Task 3:

In this part of the report we focus on theoretical workflows that could benefit the project. The explanation is done following the following format for each workflow:

Motivation …

Event → What to do and specific Jobs

1. Automated testing with specific test suites

Ensure that only the necessary tests run at each stage of development, reducing execution time and improving efficiency. Instead of running all tests every time, different test suites are triggered based on the type of change:

On push to feature branch → Run **unit tests** (fast-executing tests that focus on individual functions or components)

On push to development branch → Run **integration tests** (Test the interaction between different parts of the system or with external services)

On push to main branch → Run **end-to-end (E2E) tests** (full system testing, simulating real user interactions)

On pull requests → Run relevant tests based on changes (identify affected files and run only related tests)

Other jobs that can be used if needed:

- **Performance tests**: Measure speed and resource usage under load.
- **Visual regression tests**: Capture UI screenshots and compare them to previous versions to detect unintended design changes.

2. Automated testing with test coverage reports

Ensure that important parts of the code are tested and prevent untested code from slipping into production. Every time new code is pushed or reviewed, test coverage is analyzed:

- **On push to any branch** → Run all tests (gather data on how much of the code is executed by tests)
    - Generate coverage report with using tools like coverage.py for Python
    - Publish coverage reports to tracking platforms
    - Prevent merging if coverage falls below a percentage

- **On pull request** → Add comments to pull requests (if a pull request reduces code coverage, add a comment to the PR to notify the developer)

3. Automated testing with cross-browsing reports

Ensure the web application works properly in different browsers, avoiding compatibility issues. Since different browsers render web applications differently, this workflow runs automated tests across multiple browsers:

- **On push to main branch**
- **On pull request**

Jobs:

- Build the application: Ensure the latest version is tested.
- Run cross-browser tests: Use a tool like Selenium or Cypress to run tests in multiple browsers
- Capture screenshots or videos of tests running in each browser
- Publish test results: Store logs in services like BrowserStack or Sauce Labs for debugging.
- Report results: Provide immediate feedback to developers.

4. Automated Testing with Database Migrations Testing

Ensure database changes (migrations) don't cause data loss, corruption, or inconsistencies. Before applying new database migrations, this workflow sets up a controlled test environment and validates the process:

- **On push to main branch**
- **On pull request**

Jobs:

- Set up a test database: Create a temporary database for testing.
- Apply migrations: verify that schema changes execute successfully
- Run tests: execute tests that interact with the database
- Rollback migrations: Check if rollback restores the database correctly.
- Tear down test database: Clean up temporary test data after execution.