

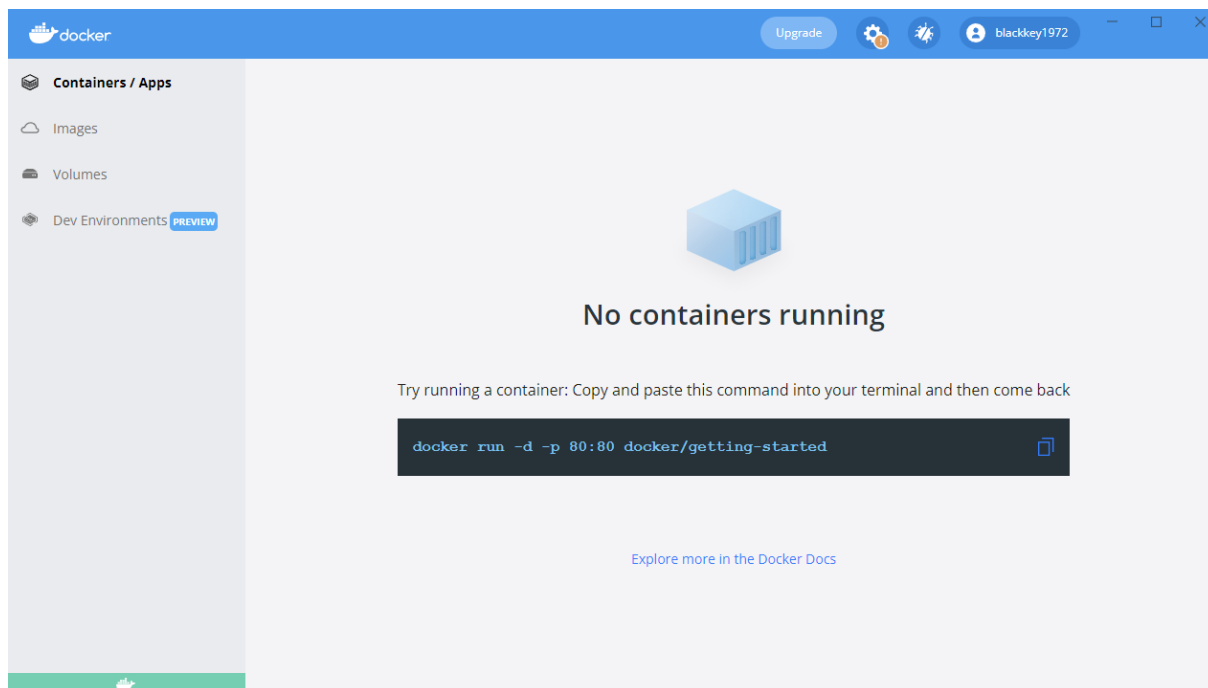
**FEI STU**  
**Používateľská príručka**  
**Unit testy v Linux kontajneroch**

**Tomáš Koštrna**  
**2021/2022**  
**3BC API MSUS**

# Docker

Docker je vhodné inštalovať priamo zo stránky [Developers - Docker](#). Jedná sa o oficiálnu stránku, ktorá poskytuje možnosť stiahnuť aktuálnu verziu aplikácie Docker. Verzie sú poskytované pre Windows a Mac. Inštalácia pre Linux prebieha pomocou príkazového riadku. Napríklad pre Ubuntu existuje samostatná stránka [Install Docker Engine on Ubuntu | Docker Documentation](#).

Je pravdepodobné, že používanie Dockeru vo Windowse alebo Macu bude vyžadovať registráciu. Pre registráciu možno využiť stránku [Docker Hub](#). Pri problémoch s inštaláciou alebo nefunkčností aplikácie na Windowse poskytuje Docker túto stránku [Logs and troubleshooting | Docker Documentation](#).



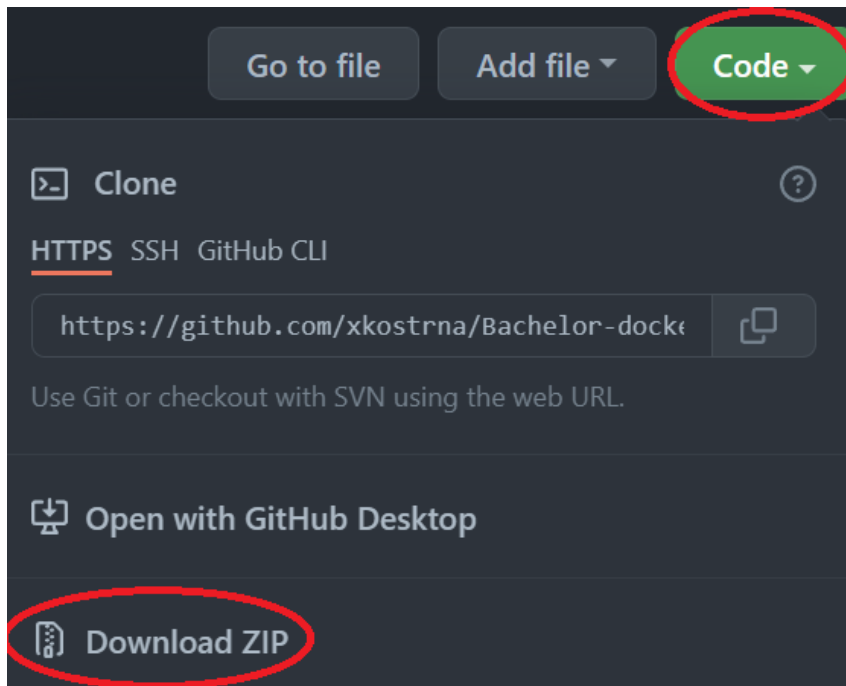
Obrázok 1 Docker po inštalácii

Pre Unix docker neponúka grafické rozhranie. Ovládanie Dockeru v rámci Unixových systémov je realizované pomocou príkazového riadku. Postup aj popis príkazov, ktoré Docker poskytuje je na spomenutej stránke pre Ubuntu. V rámci aplikácie Docker okrem samotnej inštalácie je už potrebné iba nainštalovať obraz.

## Inštalácia obrazu pre Docker

### Stiahnutie obrazu

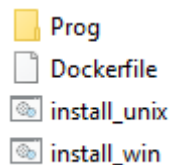
Celý repozitár s obrazom aj jeho inštaláciami je pripravený na tomto odkaze [xkostma/Bachelor-docker-image \(github.com\)](#). Po otvorení odkazu stačí kliknúť na zelené tlačidlo „Code“ a následne stiahnuť vo formáte ZIP (Obrázok 2 Stiahnutie obrazu pre Docker).



Obrázok 2 Stiahnutie obrazu pre Docker

## Inštalácia pre Windows

Po stiahnutí súboru ZIP ho stačí rozbaľiť a dvojklikom spustiť súbor `install_win.bat`.



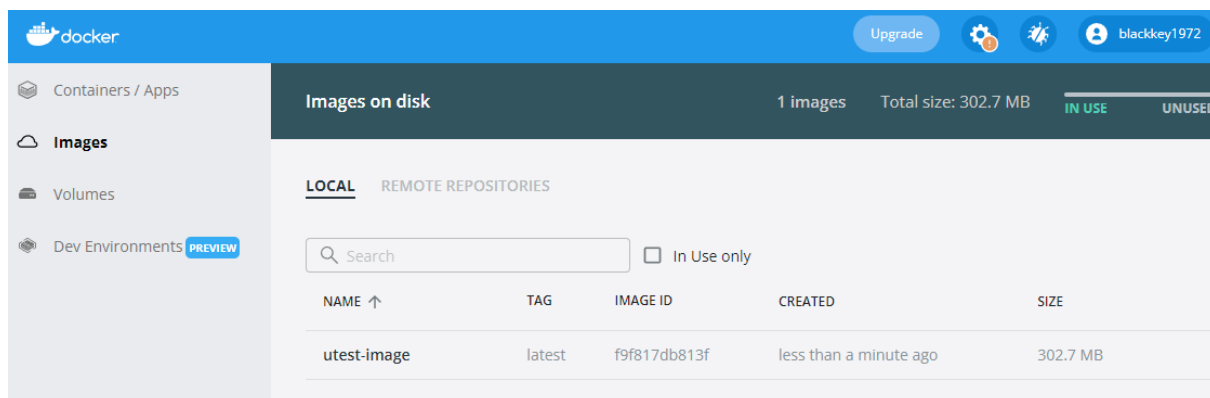
Obrázok 3 Obsah rozbaľeného súboru ZIP

Po spustení inštaláčného skriptu sa otvorí príkazový riadok a je možné pozorovať inštaláciu. Inštalácia prebieha pomocou súboru `Dockerfile`. Jedná sa o konfiguračný súbor, ktorý v rámci obrazu nainštaluje potrebnú funkcionálnosť. V rámci funkcionality sa jedná o nástroje `gcc`, `g++`, `python3` a `Makefile`.

```
C:\WINDOWS\system32\cmd.exe
C:\Users\kostrna\Downloads\Bachelor-docker-image-master\Bachelor-docker-image-master>docker build -t utest-image .
[+] Building 32.8s (6/10)
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 225B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/library/alpine:latest                  2.6s
=> [auth] library/alpine:pull token for registry-1.docker.io                    0.0s
=> [1/5] FROM docker.io/library/alpine:latest@sha256:4edbd2beb5f78b1014028f4fbb99f3237d9561100b6881aabbf5acce2c4 1.4s
=> => resolve docker.io/library/alpine:latest@sha256:4edbd2beb5f78b1014028f4fbb99f3237d9561100b6881aabbf5acce2c4 0.0s
=> => sha256:4edbd2beb5f78b1014028f4fbb99f3237d9561100b6881aabbf5acce2c4f9454 1.64kB / 1.64kB 0.0s
=> => sha256:a777c9c66ba177ccfea23f2a216ff6721e78a662cd17019488c417135299cd89 528B / 528B 0.0s
=> => sha256:0ac33e5f5afa79e084075e8698a22d574816eea8d7b7d480586835657c3e1c8b 1.47kB / 1.47kB 0.0s
=> => sha256:df9b9388f04ad6279a7410b85cedfdcb2208c0a003da7ab5613af71079148139 2.81MB / 2.81MB 1.2s
=> => extracting sha256:df9b9388f04ad6279a7410b85cedfdcb2208c0a003da7ab5613af71079148139 0.1s
=> [internal] load build context                                                  0.0s
=> => transferring context: 1.24kB                                              0.0s
=> [2/5] RUN apk update && apk add --no-cache python3 py3-pip g++ cmake make    28.8s
=> => # (25/63) Installing mpfr4 (4.1.0-r0)
=> => # (26/63) Installing mpc1 (1.2.1-r0)
=> => # (27/63) Installing gcc (10.3.1_git20211027-r0)
=> => # (28/63) Installing musl-dev (1.2.2-r7)
```

Obrázok 4 Inštalácia obrazu pre Docker

Po inštalácii je možné v aplikácii pozorovať nový obraz pripravený pre použitie. Názov obrazu je `utest-image`.



Obrázok 5 Pridanie obrazu do aplikácie

## Inštalácia obrazu pre Unix

Pri inštalácii obrazu pre Unix postupujeme rovnako. Taktiež je najprv potrebné stiahnuť súbor ZIP ako pri inštalácii vo Windowse. Inštalačný súbor `install_unix.sh` spustíme z príkazového riadku.

```
os2021@os2021: ~/Downloads/Bachelor-docker-image-master
os2021@os2021:~/Downloads/Bachelor-docker-image-master$ ./install_unix.sh
Sending build context to Docker daemon 6.656kB
Step 1/5 : FROM alpine:latest
--> c059bfaa849c
Step 2/5 : RUN apk update && apk add --no-cache python3 py3-pip gcc g++ cmake make libc-dev
--> Running in e5cf9e887b6d
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/community/x86_64/APKINDEX.tar.gz
v3.15.4-49-g2d9f01a977 [https://dl-cdn.alpinelinux.org/alpine/v3.15/main]
v3.15.4-51-g38b9d9d409 [https://dl-cdn.alpinelinux.org/alpine/v3.15/community]
OK: 15862 distinct packages available
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/community/x86_64/APKINDEX.tar.gz
(1/63) Installing libacl (2.2.53-r0)
(2/63) Installing libbz2 (1.0.8-r1)
(3/63) Installing expat (2.4.7-r0)
(4/63) Installing lz4-libs (1.9.3-r1)
(5/63) Installing xz-libs (5.2.5-r1)
(6/63) Installing zstd-libs (1.5.0-r0)
(7/63) Installing libarchive (3.6.1-r0)
(8/63) Installing ca-certificates (20211220-r0)
(9/63) Installing brotli-libs (1.0.9-r5)
(10/63) Installing nghttp2-libs (1.46.0-r0)
(11/63) Installing libcurl (7.80.0-r0)
```

Obrázok 6 Inštalácia obrazu pre Unix

Po inštalácii je možné pozorovať pridanie nového obrazu nasledovne.

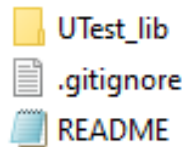
```
os2021@os2021:~/Downloads/Bachelor-docker-image-master$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
utest-image   latest    6a7cc57d6317   About a minute ago   303MB
```

Obrázok 7 Pridanie obrazu pre Unix

Teraz je obraz pripravený pre použitie.

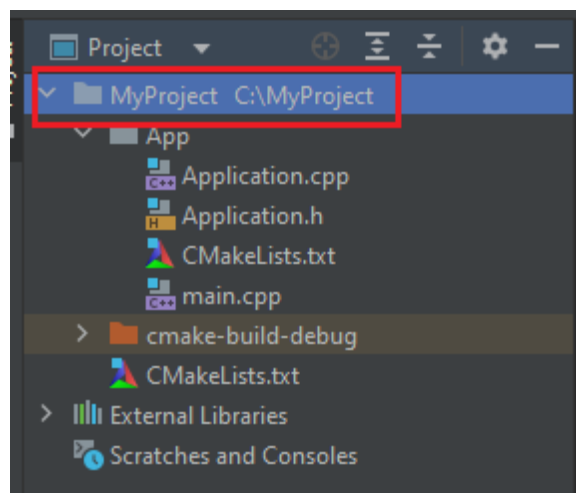
## Pridanie frameworku do používateľského projektu

Zdrojové súbory frameworku sú na odkaze [xkostma/Bachelor-UTest-lib \(github.com\)](https://github.com/xkostma/Bachelor-UTest-lib). Aj v tomto prípade stačí stiahnuť obsah repozitáru vo formáte ZIP a následne ho rozbaľiť.



Obrázok 8 Obsah ZIP súboru frameworku po rozbalení

Framework predpokladá určitú štruktúru používateľského projektu.



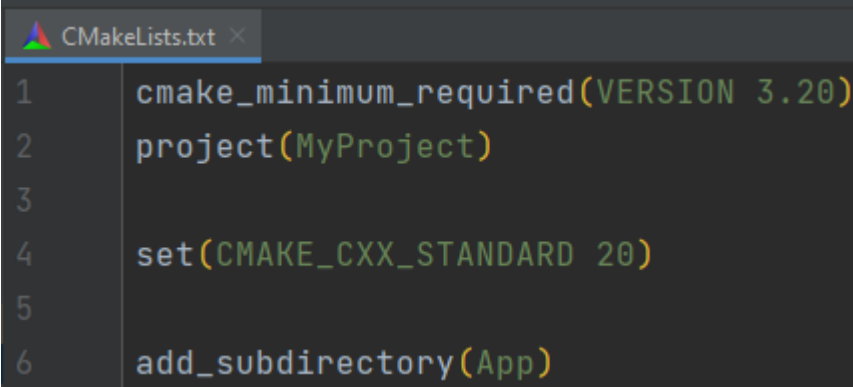
Obrázok 9 Štruktúra projektu

Štruktúra projektu (Obrázok 9) zobrazuje koreňový adresár s názvom MyProject. V tomto adresári sa nachádza podadresár App, ktorý obsahuje zdrojové súbory používateľskej aplikácie. Môže to byť napríklad kód, ktorý si používateľ želá testovať. Podadresár App obsahuje vlastný CMakeLists.txt, ktorý slúži na zostavenie zdrojového kódu. Môže vyzeráť nasledovne.

```
CMakeLists.txt x
1 cmake_minimum_required(VERSION 3.20)
2 project(MyProject)
3
4 add_executable(MyProject main.cpp Application.h Application.cpp)
```

Obrázok 10 Obsah súboru App/CMakeLists.txt

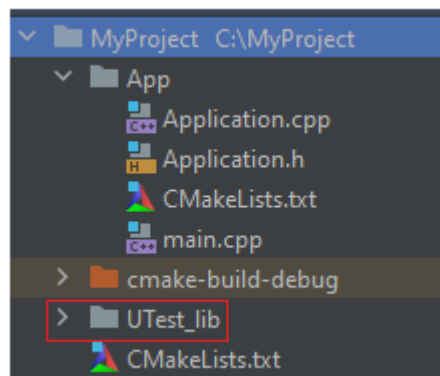
V koreňovom adresári sa nachádza ďalší súbor CMakeLists.txt. Jeho obsah je nasledovný.



```
1 cmake_minimum_required(VERSION 3.20)
2 project(MyProject)
3
4 set(CMAKE_CXX_STANDARD 20)
5
6 add_subdirectory(App)
```

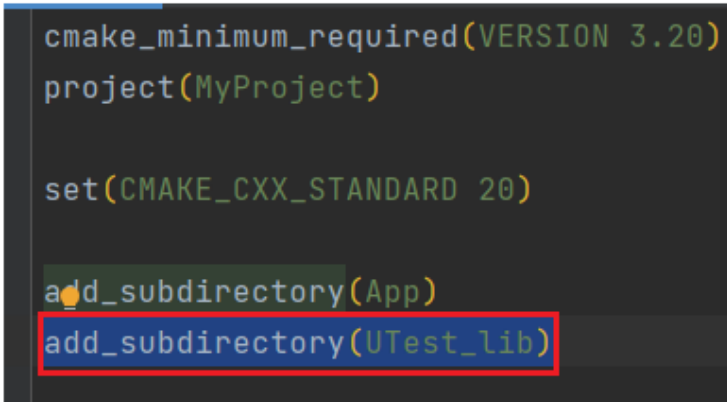
Obrázok 11 Obsah koreňového CMakeLists.txt

Jedinou úlohou tohto CMakeListu je pridať podpriechinok App, ktorý obsahuje vlastný CMakeLists čím sa zabezpečí jeho zostavenie. Adresár UTest\_lib z rozbaleného ZIP súboru repozitára je potrebné pridať do koreňového adresára projektu. V tomto prípade do adresára MyProject.



Obrázok 12 Pridanie frameworku

Adresár UTest\_lib už obsahuje vlastný CMakeLists.txt, ktorý zostaví súbory tohto frameworku. Do koreňového CMakeLists.txt stačí pridať nasledovný riadok kódu.



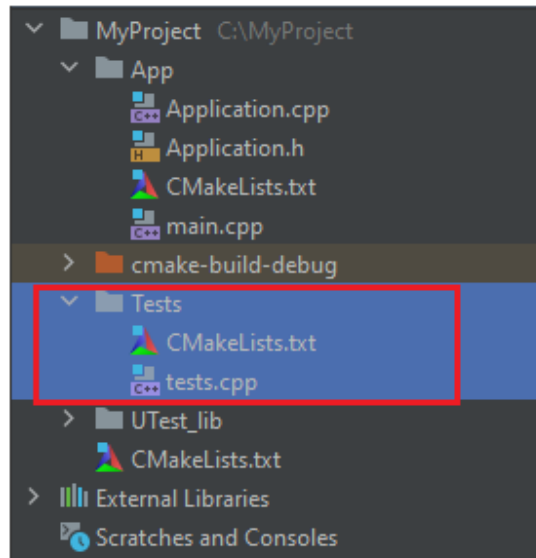
```
cmake_minimum_required(VERSION 3.20)
project(MyProject)

set(CMAKE_CXX_STANDARD 20)

add_subdirectory(App)
add_subdirectory(UTest_lib)
```

Obrázok 13 Pridanie riadku do koreňového CMakeLists.txt

Tento riadok zabezpečí zahrnutie CMakeListu z podadresára UTest\_lib, čím sa zostavia jeho súbory. Posledným krokom je vytvorenie podadresára určeného pre testovanie. **Názov tohto podadresára musí byť Tests.** V tomto podadresári vytvoríme zdrojový súbor .cpp pre unit testy a CMakeLists pre zostavenie unit testov.



Obrázok 14 Pridanie podadresára Tests

Obsah súboru Tests/CMakeLists.txt vyžaduje presný formát.

```
CMakeLists.txt
1 cmake_minimum_required(VERSION 3.20)
2 project(MyProject)
3
4 add_executable(UnitTests.exe # UnitTests.exe je názov spustiteľného súboru unit testov
5     tests.cpp                # súbor z obsahom unit testov
6     ../App/Application.h     #
7     ../App/Application.cpp) # súbory obsahujúce kód ktorý chceme testovať
8
9 target_link_libraries(UnitTests.exe UTest) # prepojenie spustiteľného súboru z funkčnosťou frameworku
10
```

Obrázok 15 Obsah súboru Tests/CMakeLists.txt

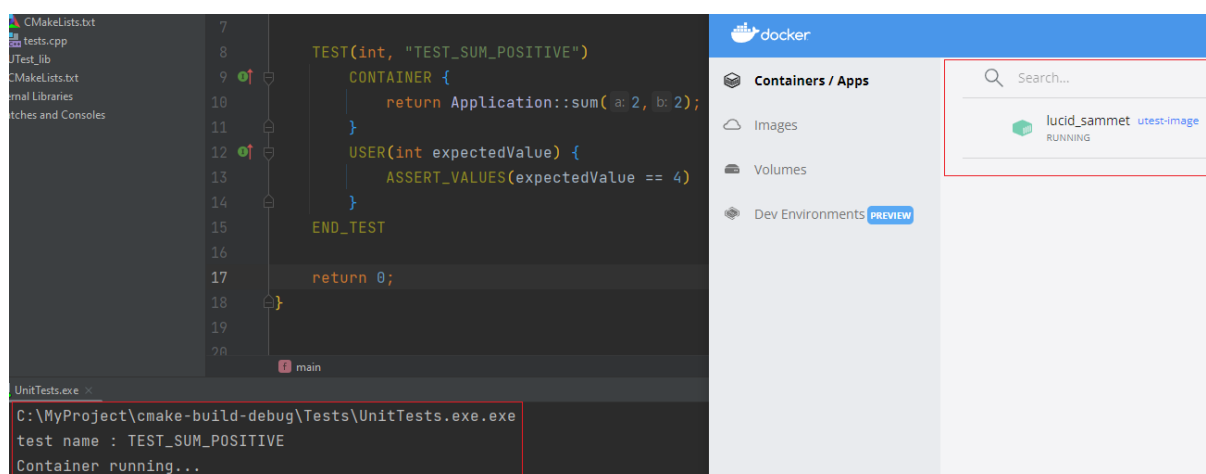
Názov spustiteľného súboru musí byť taktiež dodržaný. Pythonovský skript v kontajneri totiž spúšťa práve tento súbor. Jeho názov je teda vyhradený. Názov UTest na riadku č. 9 je názov knižnice, ktorá vznikne zostavením súborov v podadresári UTest\_lib.

## Testovanie

```
1 #include "../UTest_lib/UTest.h"
2 #include "../App/Application.h"
3
4 int main() {
5
6     Paths::setSharedFolder( path: "C:\\Shared");
7
8     TEST(int, "TEST_SUM_POSITIVE")
9     {
10         CONTAINER {
11             return Application::sum( a: 2, b: 2);
12         }
13         USER(int expectedValue) {
14             ASSERT_VALUES(expectedValue == 4)
15         }
16     }
17     return 0;
18 }
```

Obrázok 16 Obsah súboru Tests/tests.cpp

Prvý riadok zahrnie funkcionality frameworku a teda umožní písať unit testy pomocou makra. Druhý riadok pridá funkcionality aplikácie ktorú chceme testovať. Kód na riadku 6 určí cestu k zdieľanému priečinku v používateľskom systéme, ktorý framework použije pre komunikáciu. **Tento riadok je nutný.** Po spustení funkcie main() prebehne testovanie.



Obrázok 17 Spustenie unit testov



Po spustení unit testov možno pozorovať vytvorenie kontajneru v ktorom testovaný kód zbehne. Pre každý unit test je vytvorený samostatný kontajner. Každý kontajner vytvorený frameworkom bude po ukončení činnosti vymazaný. V konzole IDE je možné pozorovať výpis o názve aktuálneho unit testu a spustení kontajneru. Po zbehnutí každého unit testu je vypísaná informácia o jeho úspešnosti nasledovne.

```
test name : TEST_SUM_POSITIVE
Container running...
result : Passed
-----
```

Obrázok 18 Výpis úspešnosti testu

V prípade Unixu je možné pozorovať vytvorenie kontajneru príkazom `docker ps -a` spustením v príkazovom riadku.

## Zápis výsledkov unit testov

Pre používateľa je pripravená metóda `UTestResult::getInfo()`. Parametre metódy sú :

- `toFile` – hodnota typu `boolean`, ktorá určí či sa majú výsledky zapísať aj do súboru.
- `path` – reťazec určujúci cestu k súboru s výsledkami.
- `fileName` – reťazec určujúci názov súboru s výsledkami.

Ak používateľ neurčí tieto parametre, metóda bude volaná s prednastavenými hodnotami :

- `toFile` – `true`.
- `path` – cesta ku koreňovému adresáru projektu.
- `fileName` – „test\_results“.

```
4 ▶ int main() {
5
6     Paths::setSharedFolder( path: "C:\\Shared");
7
8     TEST(int, "TEST_SUM_POSITIVE")
9     CONTAINER {
10         return Application::sum( a: 2, b: 2);
11     }
12     USER(int expectedValue) {
13         ASSERT_VALUES(expectedValue == 4)
14     }
15     END_TEST
16
17     UTestResult::getInfo();
18
19     return 0;
20 }
```

Obrázok 19 Použitie metódy `UTestResult::getInfo()`