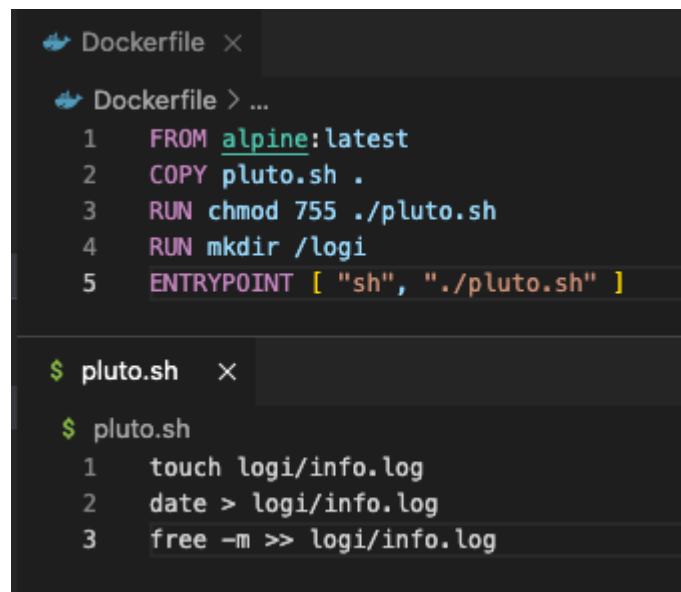


Sprawozdanie z laboratorium 10 - wolumeny

Wykonał: Mikołaj Starczewski

1. Opracować Dockerfile na podstawie obrazu bazowego *alpine*, który:

- zawierać będzie skrypt o nazwie *pluto.sh*, który generować będzie informacje o dacie utworzenia oraz ilości dostępnej pamięci a następnie zapisywać te dane do pliku tekstowego o nazwie *info.log*.
- pozwoli na umieszczenie pliku *info.log* na wolumenie, który podłączony ma być do systemu plików kontenera, w katalogu */logi*
- zdefiniuje sposób uruchomienia skryptu *pluto.sh* przy starcie kontenera.



```
Dockerfile x
Dockerfile > ...
1 FROM alpine:latest
2 COPY pluto.sh .
3 RUN chmod 755 ./pluto.sh
4 RUN mkdir /logi
5 ENTRYPOINT [ "sh", "./pluto.sh" ]

$ pluto.sh x
$ pluto.sh
1 touch logi/info.log
2 date > logi/info.log
3 free -m >> logi/info.log
```

Dockerfile korzysta z obrazu **alpine**. Kopiuje tam skrypt, po czym nadaje mu odpowiednie uprawnienia. Tworzony jest folder **/logi** do którego wygenerowany zostanie nasz log.

pluto.sh tworzy plik *info.log* w odpowiedniej ścieżce, po czym wrzuca tam datę oraz dane na temat dostępnej pamięci.

2. Zbudować obraz i nazwać go *lab10docker*

```
→ chmurylab10 docker build -t lab10docker .
Sending build context to Docker daemon 3.072kB
Step 1/5 : FROM alpine:latest
latest: Pulling from library/alpine
b3c136eddcbf: Pull complete
Digest: sha256:686d8c9dfa6f3ccfc8230bc3178d23f84eeaf7e457f36f271ab1acc53015037c
Status: Downloaded newer image for alpine:latest
----> 6e30ab57aeee
Step 2/5 : COPY pluto.sh .
----> 12a450bec2a9
Step 3/5 : RUN chmod 755 ./pluto.sh
----> Running in 6b6869c98729
Removing intermediate container 6b6869c98729
----> 4ad280c60d86
Step 4/5 : RUN mkdir /logi
----> Running in cc09a7bc24b7
Removing intermediate container cc09a7bc24b7
----> 4d898b368910
Step 5/5 : ENTRYPOINT [ "sh", "./pluto.sh" ]
----> Running in 93a2233f795c
Removing intermediate container 93a2233f795c
----> c71fd49f027f
Successfully built c71fd49f027f
Successfully tagged lab10docker:latest

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

Budowany jest obraz z odpowiednim tagiem.

3. Utworzyć volumen o nazwie *RemoteVol* wykorzystujący odpowiedni sterownik (plugin), by miejsce przechowywania danych znajdowało się na systemie macierzystym:

```
→ chmurylab10 docker volume create --name RemoteVol --opt type=None --opt device=/Users/mikolaj/Desktop/chmurylab10/logi --opt o=bind
RemoteVol
```

Przez korzystanie z systemu macOS na technologii M1, nie byłem w stanie znaleźć rozwiązania wykorzystujące podany plugin. Powyższe rozwiązanie z device wskazującym na ścieżkę lokalną działa tak jak chciało polecenie.

4. Uruchomić kontener o nazwie *alpine10* na bazie zbudowanego obrazu *lab10docker* w taki sposób, by:

a. podłączyć do niego utworzony volumen *RemoteVol* w miejsce katalogu */logi* w systemie plików kontenera.

```
→ chmurylab10 docker run --name alpine10 -v RemoteVol:/logi lab10docker
→ chmurylab10
```

Uruchomienie kontenera z flagą *-v* montująca volumen do odpowiedniego miejsca na kontenerze.

a. korzystając z informacji w podpunkcie E, dla tego kontenera ograniczyć ilość wykorzystywanej pamięci RAM do 512MB.

```
→ chmurylab10 docker run --name alpine10 -v RemoteVol:/logi -m 512m lab10docker
```

Flaga **-m** umożliwiła ograniczenie pamięci kontenera.

5. Za pomocą poznanych narzędzi docker plugin, docker inspect ..., docker stats ... itd. należy potwierdzić, że:

a. skrypt *pluto.sh* generuje wymagane dane i umieszcza je w pliku *info.log* na wolumenie, który znajduje się w systemie plików na maszynie macierzystej.

```
→ chmurylab10 docker inspect alpine10
[
  {
    "Id": "932d20431d3d0e3f6d51090e22e22e7f6443c26f78ec9a3e4ae1a06b0c4833ac",
    "Created": "2022-05-27T11:54:13.611850833Z",
    "Path": "sh",
```

Polecenie **docker inspect** pozwala na wyświetlenie wszystkich danych na temat kontenera.

```
    "WorkingDir": "",
    "Entrypoint": [
      "sh",
      "./pluto.sh"
    ],
    "OnBuild": null,
```

Pod kluczem **Entrypoint** wyświetla się zawartość naszego entrypoint, który uruchamia skrypt.

```
→ chmurylab10 ls
Dockerfile logi      pluto.sh
→ chmurylab10 cat logi/info.log
Fri May 27 11:54:13 UTC 2022

```

	total	used	free	shared	buff/cache	available
Mem:	7951	297	7050	320	604	7180
Swap:	1024	0	1024			

Wygenerowany plik przez kontener.

b. kontener *alpine4* ma ograniczoną ilość pamięci RAM zgodnie z treścią zadania

```
    "Isolation": "",
    "CpuShares": 0,
    "Memory": 536870912,
    "NanoCpus": 0,
    "CgroupParent": "",
    "BlkioWeight": 0,
```

Pod kluczem **Memory** znajduje się ilość dostępnej pamięci.