
ATTACK
DEFENSE
by PentesterAcademy

Name	Multi Container Setups
URL	https://attackdefense.com/challengedetails?cid=2271
Type	Docker Security : Container Basics

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Objective: Create the multi container setup with manual method and docker-compose!

Solution:

Manual method

Step 1: Pull the both Docker images.

Command: docker pull registry:5000/appserver

```
root@localhost:~# docker pull registry:5000/appserver
Using default tag: latest
latest: Pulling from appserver
d519e2592276: Pull complete
d22d2dfcfa9c: Pull complete
b3afe92c540b: Pull complete
9188c24067dc: Pull complete
eeee80a04dcf: Pull complete
d750d8ba83e3: Pull complete
fd98e9b0fb87: Pull complete
Digest: sha256:e6ee5e787f843a73837f6186c0e31ebd5c8f8c7029bfe8811800c98f4c112f19
Status: Downloaded newer image for registry:5000/appserver:latest
registry:5000/appserver:latest
```

Command: docker pull registry:5000/memcached-app

```
root@localhost:~# docker pull registry:5000/memcached-app
Using default tag: latest
latest: Pulling from memcached-app
7b8b6451c85f: Pull complete
ab4d1096d9ba: Pull complete
e6797d1788ac: Pull complete
e25c5c290bde: Pull complete
0dc12925fe9f: Pull complete
ccaf5b836780: Pull complete
32273d7680c8: Pull complete
669c646b5e6a: Pull complete
5cb09c1a71cb: Pull complete
6014ea3b58b7: Pull complete
5fbe26ba3dfb: Pull complete
e227ca8741ad: Pull complete
Digest: sha256:90b94c68175036149112aca11fc45338671798277826c457c914e0211946a354
Status: Downloaded newer image for registry:5000/memcached-app:latest
registry:5000/memcached-app:latest
```

Step 2: Check the locally present image list.

Command: docker images

```
root@localhost:~# docker images
REPOSITORY                   TAG                 IMAGE ID            CREATED             SIZE
registry:5000/appserver      latest              d80770e73034        37 hours ago        517MB
registry:5000/memcached-app  latest              15508dc64dc6        2 years ago         490MB
```

Step 3: Create a network named "test-net".

Command: docker network create test-net

```
root@localhost:~# docker network create test-net
5fcac9dc218b55dc2b1a5a2c220d1d7b3861f3dcc14b8a4b1f0fe7aaab8a1b3f
```

Verify if the network is created

Command: docker network ls

```
root@localhost:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
1c88421fd79c        bridge              bridge              local
2371877d0b47        host                host                local
282213cc5959        none                null                local
5fcac9dc218b        test-net            bridge              local
```

Step 4: Run the appserver images first and attach test-net to it.

Command: docker run --network test-net -d registry:5000/appserver

Now, similarly run the memcached image

Command: docker run --network test-net -d registry:5000/memcached-app

```
root@localhost:~# docker run -d --network=test-net registry:5000/appserver
c3681ed768facc8f7e550514248c80fd37e02caac9d5d2b342275c23ea42e2f6
root@localhost:~#
root@localhost:~#
root@localhost:~# docker run -d --network=test-net registry:5000/memcached-app
db073e7174f86fa0add38bd2eda2944adb1b0d1adc51f9e94b238f132e54ab3e
```

We deliberately started appserver container before memcached-app container because they will work only in this order as suggested by the challenge description.

Step 5: Check the running containers

Command: docker ps

```
root@localhost:~# docker ps
CONTAINER ID        IMAGE                          COMMAND             CREATED
db073e7174f8        registry:5000/memcached-app    "/startup.sh"       20 seconds ago
shi
c3681ed768fa        registry:5000/appserver        "/start.sh"         36 seconds ago
yabhata
```

Step 6: Check the IP address for the appserver container.

Command: docker inspect c3681ed768fa | grep IP

```
root@localhost:~# docker inspect c3681ed768fa | grep IP
            "LinkLocalIPv6Address": "",
            "LinkLocalIPv6PrefixLen": 0,
            "SecondaryIPAddresses": null,
            "SecondaryIPv6Addresses": null,
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "",
            "IPPrefixLen": 0,
            "IPv6Gateway": "",
                    "IPAMConfig": null,
                    "IPAddress": "172.19.0.2",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
```

Step 7: Make a curl request on this IP to see if we can access the HTTP API for memcached.

Command: curl 172.17.0.2

```
root@localhost:~# curl 172.19.0.2


Accessing Memcached Made Simpler!!

Method    Endpoint    Parameter        Description
-------------------------------------------------------------

 GET      /list                      List all memcached keys
 GET      /get          key        Retrive data stored in key
 GET      /set        key,value      Store key value pair
-------------------------------------------------------------
```

Step 8: Stop both containers

Command: docker stop db073e7174f8 c3681ed768fa

```
root@localhost:~# docker stop db073e7174f8 c3681ed768fa
db073e7174f8
c3681ed768fa
```

Also delete the test-net network

Command: docker network rm test-net

```
root@localhost:~# docker network  rm test-net
test-net
```

Docker Compose method

Step 9: Create a docker-compose.yaml and describe this scenario

docker-compose.yaml

```
version: "3.5"
services:
  appserver:
      image: registry:5000/appserver
      networks:
          - backend
  memcache:
      image: registry:5000/memcached-app
      networks:
          - backend
      depends_on:
          - appserver
networks:
   backend:
      name: test-net2
      driver: bridge
```

```
root@localhost:~# cat -n docker-compose.yaml
     1  version: "3.5"
     2  services:
     3     appserver:
     4           image: registry:5000/appserver
     5           networks:
     6                   - backend
     7     memcache:
     8           image: registry:5000/memcached-app
     9           networks:
    10                   - backend
    11           depends_on:
    12                   - appserver
    13  networks:
    14      backend:
    15           name: test-net2
    16           driver: bridge
```

Step 9: Use docker-compose to create the setup

Command: docker-compose up

```
root@localhost:~# docker-compose up
Creating network "test-net2" with driver "bridge"
Creating root_appserver_1 ... done
Creating root_memcache_1  ... done
Attaching to root_appserver_1, root_memcache_1
appserver_1  |  * Serving Flask app "app" (lazy loading)
appserver_1  |  * Environment: production
appserver_1  |    WARNING: This is a development server. Do not use it in a production deployment.
appserver_1  |    Use a production WSGI server instead.
appserver_1  |  * Debug mode: off
appserver_1  |  * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
memcache_1   | /usr/lib/python2.7/dist-packages/supervisor/options.py:297: UserWarning: Supervisord is running as root
its configuration file in default locations (including its current working directory); you probably want to specify a
an absolute path to a configuration file for improved security.
memcache_1   |   'Supervisord is running as root and it is searching '
memcache_1   | 2021-02-27 03:05:01,866 CRIT Supervisor running as root (no user in config file)
memcache_1   | 2021-02-27 03:05:01,871 WARN No file matches via include "/etc/supervisor/conf.d/*.conf"
memcache_1   | 2021-02-27 03:05:01,998 INFO RPC interface 'supervisor' initialized
memcache_1   | 2021-02-27 03:05:02,002 CRIT Server 'unix_http_server' running without any HTTP authentication checking
memcache_1   | 2021-02-27 03:05:02,008 INFO supervisord started with pid 1
```

Note: Make sure the docker-compose.yaml is present in the working directory.

Step 10: Open another terminal (by clicking the lab link button again) check the IP address for the appserver container.

Then, make a curl request on this IP to see if we can access the HTTP API for memcached.

Command: curl 172.20.0.2

```
root@localhost:~# curl 172.20.0.2


Accessing Memcached Made Simpler!!

Method    Endpoint    Parameter        Description
-----------------------------------------------------------------

 GET      /list                      List all memcached keys
 GET      /get          key        Retrive data stored in key
 GET      /set        key,value      Store key value pair
-----------------------------------------------------------------
```

Step 11: Use the curl request to store an item to the memcached and retrieve it

Store an term

Command: curl "172.20.0.2/set?key=a&value=test"

```
root@localhost:~# curl "172.20.0.2/set?key=a&value=test"
Key Value pair stored sucessfully
root@localhost:~#
```

Retrieve that item

Command: curl "172.20.0.2/get?key=a"

```
root@localhost:~# curl "172.20.0.2/get?key=a"
Key: a
Value: test
root@localhost:~#
```

Step 12: Destroy the setup

Command: docker-compose down

References:

1. Docker-compose (https://docs.docker.com/compose/)