Docker: Caddy

Caddy is a powerful full featured webserver, which is also easy to use and setup.

In this guide i will show how to use Caddy as SSL- Reverse- Proxy for your services to deliver them to the internet via name- based virtual hosting.

Prerequisites

You need a (Linux/rootless)- Dockerized System, as descirbed here: https://obel1x.de/dokuwiki/doku.php?id=content:serverbasics:docker|

DNS- Records

You will also need a domainname like `my.domain.tld` and hostnames, that are resolvable for each service, so that `service.my.domain.tld` can be resolved from out of - and in the internet to point to your Server. If you do not have them already, read here: network-dyndns

When your service sucessfully resolves, you can use it in Caddy as written beneath.

Create Caddy Yaml

log in to your docker- user using ssh e.g.: `ssh localhost -l docker`

Then go to where your yaml- folders are and create a new folder for your caddy service.

For me, this would be `mkdir /srv/docker-compose/caddy`
Change to that directory and create the following docker-compose.yml file in it, putting in the following:

```
services:
  caddy:
    image: caddy:alpine
    restart: always
    volumes:
```

```
      - $PWD/Caddyfile:/etc/caddy/Caddyfile:ro
      - caddy_certs:/certs
      - caddy_config:/config
      - caddy_data:/data
      - caddy_sites:/srv
#Test html
#      - $PWD/index.html:/usr/share/caddy/index.html:ro
    cap_add:
      - NET_ADMIN
    healthcheck:
      test: "wget --no-verbose --tries=1 --spider https://www.servername.domainname.tld || exit 1"
      interval: "60s"
      timeout: "3s"
      start_period: "5s"
      retries: 3
# Be sure, that docker daemon has access to unprivileged ports (beneath 1024)
# This can be archived by:
# setcap cap_net_bind_service=+ep /usr/bin/rootlesskit
#
# To access local services, take the hostname directly, maybe define it static and add it to /etc/hosts on the host
# Mind, that the ports must be pubilshed by the other containers to the host via ports or expose, or add them to the network
# e.g. pcserver:9000 !!! NOT: !!! localhost:9000 - this is prevented by docker in rootless- mode !
#Do NOT use networkmode: "host", this will fail (Acme: Connection refused)!
#    network_mode: "host"
    ports:
      - "80:80"
      - "443:443"
      - "443:443/udp"

volumes:
  caddy_data:
  caddy_config:
  caddy_certs:
  caddy_sites:

networks:
# Still needs to be defined while without it won't enable ipv6
  default:
```

```
    driver: bridge
    enable_ipv6: true
```

Also, check that your Firewall has those Port 80 + 443 open on your host and that Port Forwarding in your Router is enabled for ipv4 and for ipv6.

Caddy Configuration

if you omit the Caddyfile, the server will already work, but we can directly Skip those tests and create the file `Caddyfile` in that folder too with the following content:

```
https://portainer.my.domain.tld:443 {
        header Strict-Transport-Security max-age=31536000;
        reverse_proxy mylocalhostname:9000
}
```

replace *mylocalhostname* with your actual hostname (can be found out by calling hostname in your terminal).

Don't use localhost - see above. If you do not have a clue which hostname you have, better specify some fixed one which you can freely chose and edit `/etc/hosts` to have that name point to your local ip.

Fetch and run the Caddy Container

Thats all - use `docker-compose up -d` to start your container. In the Container- Logs you will see Caddy automagically create SSL- Certificates from lets encrypt if everything was setup the right way. Caddy will take care of renewal without the need to configure anything.

If the Caddy doe not return any Errors, you now have a powerful proxy, that can transparently deliver your Dockers to the world with SSL- encryption enabled.