LAMP docker container with adminer

I know, I know, we are supposed to only dockerize microservices and not a complete set of services like apache AND mysql in the same container, and I also know, that a container is not a VM ..

having said all that, let's forget it all and do the opposite :) .. what i want is a very simple docker image that i can start a new container which will give me a quick and dirty Linux + Apache + MySQL (MariaDB actually) + PHP envrionment to mess around with while I'm on the go with my notebook or on my workstation at work. I don't want to use docker compose to combine multiple containers into a whole environment etc. This is all very nice for production and such, but just to try a few lines of php or to give some open source php scripts a quick try it seems overly complicated.

so here i am, creating a docker image that includes the following:

- based on latest ubuntu
- apache2
- mariadb-server
- adminer (like phpMySQL only smaller and simpler)
- php
- openssh-server
- a non-privileged user `jdoe` with defualt password `jdpasswd`
- exposes ports `80` and `22`

the openssh server might come as a shock.. this is the part where we violate the idea, that a container is to be treated like a service, not like a VM :) through ssh we can login to the container and using i.e. apt-get we can then install additional dependencies that our php scripts may have.

once again: this is ment to be a quick and dirty way to try out some php stuff and then **THROW IT AWAY**, don't use this as a base for a complete service you want to run for a longer time in a productive environment, do it the proper way and use docker compose or k8s!

The files needed

I call this container my “uamp” (ubuntu apache mysql php) .. so I've created a folder “uamp” which contains the following structure:

```
.
├── build.sh
├── Dockerfile
├── files
│   ├── phpinfo.php
│   ├── preseed.txt
│   └── startup.sh
└── run.sh
```

here are the contents of those files:

Dockerfile

```
FROM ubuntu:latest
# for a specific version of ubuntu use a tag like ubuntu:20.04 or
similar
COPY files/preseed.txt /tmp/preseed.txt
RUN debconf-set-selections /tmp/preseed.txt &&\
  export DEBIAN_FRONTEND=noninteractive
DEBCONF_NONINTERACTIVE_SEEN=true &&\
  apt-get update &&\
  apt-get install -y openssh-server adminer mariadb-server sudo
RUN a2enconf adminer

# create default user "jdoe"
RUN useradd -rm -d /home/jdoe -s /bin/bash -G sudo,www-data jdoe
RUN echo 'jdoe:jdpasswd' | chpasswd

# set permissoins to /var/www and create symlink
COPY files/phpinfo.php /var/www/html/phpinfo.php
RUN chown -R jdoe.jdoe /var/www
RUN ln -s /var/www /home/jdoe/www

# clean up apt cache
RUN apt-get autoclean -y &&\
  apt-get autoremove -y &&\
  rm -rf /var/lib/apt/lists/*

# startup script
COPY files/startup.sh /usr/local/bin/startup.sh
CMD /bin/bash /usr/local/bin/startup.sh

EXPOSE 22 80
```

files/preseed.txt

```
tzdata tzdata/Areas select Europe
tzdata tzdata/Zones/Europe select Zurich
locales locales/locales_to_be_generated multiselect     en_US.UTF-8
UTF-8
locales locales/default_environment_locale      select  en_US.UTF-8
```

files/startup.sh

```
#!/bin/bash

# if there is no admin user in mysql yet, create it and run some other
preparations as this is the first start
# of this container
service mysql start
if ! echo "SELECT User FROM mysql.user WHERE User='admin';" | mysql |
```

```
grep -q "admin"; then
  if [ -z "$DEFAULTPW" ]; then
    DEFAULTPW='jdpasswd'
  fi
  # create admin user in mysql
  echo "CREATE USER 'admin'@'localhost' IDENTIFIED BY '${DEFAULTPW}';"
| mysql
  echo "GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost' WITH GRANT
OPTION;" | mysql
  echo "FLUSH PRIVILEGES;" | mysql
  # set password for jdoe user
  echo "jdoe:${DEFAULTPW}" | chpasswd
fi
service apache2 start
service ssh start
/bin/bash
```

build.sh

```
#!/bin/bash
docker image build --tag uamp:latest .
```

run.sh

```
#!/bin/bash
# to run in the foreground with an interactive shell:
docker run --name uamp -ti -p 8080:80 -p 2222:22 -e
DEFAULTPW="jdpasswd" uamp:latest

# to run in background:
# docker run --name uamp -td -p 8080:80 -p 2222:22 -e
DEFAULTPW="jdpasswd" uamp:latest
```

files/phpinfo.php

```
<?php phpinfo(); ?>
```

Usage

now simply build the container using the `./build.sh` script and then run it using the `./run.sh` script. once the container is started you are in a root-shell within the container. to detach from it press CTRL+p followed by CTRL + q. to attach to it again later use `docker attach uamp`

you can access the webserver via `http://localhost:8080` on your machine and you can ssh to `ssh -p 2222 jdoe@localhost` and login using the password `jdpasswd`

the user `jdoe` has full sudo privileges, so to do stuff as root simply use `sudo` as you would in a normal ubuntu installation

customizations

to set another default password upon the first start of the container, you can edit the line in `run.sh` where the `DEFAULTPW` environment variable is passed to the container. this is done so you can for example use a tool like pwgen to autmoatically generate a random password and then echo that upon starting the container if you prefer to have a new random password generated each time you spool up a new container.

you can also start the container in detached mode (aka in the background) right away, edit the `run.sh` to do so.. there is already an example for that in there.

as stated above, you can login to the container and then use apt to add additional tools and such. if you add a system service (like for example postgreSQL) you can't start it using `systemctl` as systemd is not working inside a container. Use `service xyz start` instead. if you want the service to be started each time you restart your container, add it to `/usr/local/bin/startup.sh`