0x0D. Web stack debugging #0

DevOpsSysAdminScriptingDebugging

- By: Sylvain Kalache, co-founder at Holberton School
- Weight: 1
- Project over - took place from Sep 19, 2022 6:00 AM to Sep 21, 2022 6:00 AM
- An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 3.0/3 mandatory
- **Altogether: 100.0%**
  - Mandatory: 100.0%
  - Optional: no optional tasks

Concepts

For this project, we expect you to look at these concepts:

- Network basics
- Docker
- Web stack debugging

Background Context

The Webstack debugging series will train you in the art of debugging. Computers and software rarely work the way we want (that's the "fun" part of the job!).

Being able to debug a webstack is essential for a Full-Stack Software Engineer, and it takes practice to be a master of it.

In this debugging series, broken/bugged webstacks will be given to you, the final goal is to come up with a Bash script that once executed, will bring the webstack to a working state. But before writing this Bash script, you should figure out what is going on and fix it manually.

Let's start with a very simple example. My server must:

- have a copy of the `/etc/passwd` file in `/tmp`
- have a file named `/tmp/isworking` containing the string `OK`

Let's pretend that without these 2 elements, my web application cannot work.

Let's go through this example and fix the server.

```
vagrant@vagrant:~$ docker run -d -ti ubuntu:14.04
Unable to find image 'ubuntu:14.04' locally
14.04: Pulling from library/ubuntu
34667c7e4631: Already exists
d18d76a881a4: Already exists
119c7358fbfc: Already exists
2aaf13f3eff0: Already exists
Digest: sha256:58d0da8bc2f434983c6ca4713b08be00ff5586eb5cdff47bcde4b2e88fd40f88
Status: Downloaded newer image for ubuntu:14.04
76f44c0da25e1fdf6bcd4fbc49f4d7b658aba89684080ea5d6e8a0d832be9ff9
vagrant@vagrant:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATU
S              PORTS               NAMES
76f44c0da25e        ubuntu:14.04        "/bin/bash"         13 seconds ago      Up 12
seconds                            infallible_bhabha
vagrant@vagrant:~$ docker exec -ti 76f44c0da25e /bin/bash
root@76f44c0da25e:/# ls /tmp/
root@76f44c0da25e:/# cp /etc/passwd /tmp/
root@76f44c0da25e:/# echo OK > /tmp/isworking
root@76f44c0da25e:/# ls /tmp/
isworking  passwd
root@76f44c0da25e:/#
vagrant@vagrant:~$
```

Then my answer file would contain:

```
sylvain@ubuntu:~$ cat answerfile
#!/usr/bin/env bash
# Fix my server with these magic 2 lines
cp /etc/passwd /tmp/
echo OK > /tmp/isworking
sylvain@ubuntu:~$
```

Note that as you cannot use interactive software such as `emacs` or `vi` in your Bash script, everything needs to be done from the command line (including file edition).

Installing Docker

For this project you will be given a container which you can use to solve the task. **If** you would like to have Docker so that you can experiment with it and/or solve this problem locally, you can install it on your machine, your Ubuntu 14.04 VM, or your Ubuntu 16.04 VM if you upgraded.

- Mac OS
- Windows
- Ubuntu 14.04 (Note that Docker for Ubuntu 14 is deprecated and you will have to make some adjustments to the instructions when installing)
- Ubuntu 16.04

Resources

man or help:

- `curl`

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be interpreted on Ubuntu 14.04 LTS
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project, is mandatory
- All your Bash script files must be executable
- Your Bash scripts must pass `Shellcheck` without any error
- Your Bash scripts must run without error
- The first line of all your Bash scripts should be exactly `#!/usr/bin/env bash`
- The second line of all your Bash scripts should be a comment explaining what is the script doing

Tasks

0. Give me a page!
mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Be sure to read the **Docker** concept page

In this first debugging project, you will need to get Apache to run on the container and to return a page containing `Hello Holberton` when querying the root of it.

Example:

```
vagrant@vagrant:~$ docker run -p 8080:80 -d -it holbertonschool/265-0

47ca3994a4910bbc29d1d8925b1c70e1bdd799f5442040365a7cb9a0db218021

vagrant@vagrant:~$ docker ps

CONTAINER ID        IMAGE                  COMMAND             CREATED             S
TATUS              PORTS                  NAMES

47ca3994a491        holbertonschool/265-0   "/bin/bash"         3 seconds ago       U
p 2 seconds        0.0.0.0:8080->80/tcp   vigilant_tesla

vagrant@vagrant:~$ curl 0:8080

curl: (52) Empty reply from server

vagrant@vagrant:~$
```

Here we can see that after starting my Docker container, I `curl` the port `8080` mapped to the Docker container port `80`, it does not return a page but an error message. Note that you might also get the error message `curl: (52) Empty reply from server`.

```
vagrant@vagrant:~$ curl 0:8080

Hello Holberton

vagrant@vagrant:~$
```

After connecting to the container and fixing whatever needed to be fixed (here is your mission), you can see that curling port 80 return a page that contains `Hello Holberton`. Paste the command(s) you used to fix the issue in your answer file.

Repo:

- GitHub repository: `alx-system_engineering-devops`
- Directory: `0x0D-web_stack_debugging_0`
- File: `0-give_me_a_page`

Done! Help Check your code Get a sandbox QA Review