** Task group: middleware **

Context

After developing some feature in your ASP.NET core 3.1 Web application, you'll want to test in on a platform that is much like the real Acceptance/Production environment.

The test Debian Linux server runs the dotnet core application server. The application server gets/sends its requests/repsonses via an Apache web server reverse proxy. The web application stores/retrieves its information on/from a SQLserver database server running in a docker container.

The test server runs a SDK version of the application server, because in case of failures you must be able to analyse bugs. The Acceptance server runs only a Runtime version: no unnecessary software allowed.

Normally you would separate the different services on different hosts (multi-tier). In this assignment we integrate all the services on one host.

To copy a working application from the Test to the Acceptance environment (a *promotion*) we use a secure SSH connection or we use a remote git service like gitlab.windesheim.nl, gitlab.com or github.

Deliverables

- Show a running smoketest application on the test environment and acceptance environment. This task can only be approved by your teacher after completing security level 4.

Your teacher will check if the smoketest application is running on your external ip-address or hostname.

Task

Install and configure the backend services on the test and acceptance server.

Subtask 1: install a database server

A database server is needed for the account management application. See http://docs.microsoft.com for a howto.

1. Let docker pull the container for SQLserver

   ```
   # docker pull microsoft/mssql-server-linux
   ```

2. Start the docker container

   ```
   # docker run ...
   ```

Optional, not recommended, to manage your database from the command line:

3. To access and modify your database install SLQ client. See https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-setup-tools

```
$ sudo apt-get install npm
$ sudo npm install -g sql-cli
```

4. Connect to the server using

```
$ mssql -u sa -p REALYSTRONGPASWORD123!
```

Subtask 2: install a web (proxy) server

The Apache webserver acts as a reverse proxy between browser and application server. It handles the termination of HTTPS traffic and communicates with the dotnet core server.

1. Install Apache server

```
$ sudo apt install apache2
```

2. Check if HTTP server is listening on all interfaces, port tcp/80

```
$ netstat -ant
```

3. Configure the proxy, so that incoming connections (tcp/80) are proxied to your application server (http://localhost:5000). See https://httpd.apache.org/docs/current/mod/mod_proxy.html

Subtask 3: install the dotnetcore application server

Use the complete dotnet core 3.1 SDK on your test server (virtualbox). Use the dotnet core 3.1 Runtime on your acceptance server (skylab).

Install the latest Microsoft dotnet core software development toolkit (SDK) on your Test environment and dotnet core Runtime om the Acceptance environment: Visit: https://docs.microsoft.com/en-us/dotnet/core/install/

1. Install the dotnet application server SDK (test platform on VirtualBox/debian) resp. Runtime (Acceptance platform on Skylab/debian)
   - Look for downloads and instructions at the Linux sections at https://dotnet.microsoft.com/download/dotnet-core/3.1

Subtask 4: Install OpenSSH for secure shell connections

A secure connection is used to copy your developed application to the application server (Acceptance).

1. Install openssh-server package

```
$ sudo apt install openssh-server
```

2. Check if SSH is listening on all interfaces, port tcp/22:

```
$ netstat -ant
```

Subtask 5: Configure the pfSense firewall

To communicate between a host on the Internet and your application environment you need NAT rules in the pfSense firewall. These rules translate the external IP address to the internal IP address of the application server, v.v.

1. Lookup documentation on how to configure the pfSense firewall
2. Find the IP of your Debian host

```
$ ip address
$ ifconfig
```

3. Add NAT rules to the pfSense firewall, redirecting port 22, 80 en 443 to the Debian application server
    1. Use a browser on the Kali host to connect to the pfSense firewall http://192.168.1.1/
    2. Menu -> Firewall -> NAT

Subtask 6: install and run the Smoketest application on the Test environment

To finally test the working of your infrastructure and middleware setup, a smoketest application is available.

In Virtualbox:

1. Install the entity framework packages. Look at https://docs.microsoft.com/en-us/ef/core/miscellaneous/cli/dotnet for inspiration.
2. Clone the SmokeTest application: `git clone https://gitlab.windesheim.nl/thomas/smoketest.git`
3. Find the file `appsettings.json` in `SmokeTest` and check or alter the database connection string.
4. Update your database using `dotnet ef database update` if this is not working run the dotnet-ef core 3.1 tool from your home directory: `~/.dotnet/tools/dotnet-ef database update`.
5. Run `dotnet run --project SmokeTest`
6. Surf with a browser to http://localhost:5000 and click 'Students' to check database connection

Subtask 7: optional, Make sure the dotnet application runs always as a service

See https://medium.com/@benmorel/creating-a-linux-service-with-systemd-611b5c8b91d6 It will show you how to run an application as a service after booting up your machine

1. Create a service script and think of a nice name f.i.: `/etc/systemd/system/myrevgame.service`
2. Enable the script on boot:

   ```
   $ systemctl enable myrevgame
   ```

   Read on. You might need to think a little bit harder. When should this service be started?

Done

You'll know that the environment is setup to start testing and deploying your own application.