Instituto Superior de Engenharia de Lisboa

BEng in Computer Science and Engineering

System Virtualization Techniques, Autumn/Winter 2023/2024

Third coursework assignment

For this assignment, we provide you with a minimal web application, developed on Node.js, with data stored in Elasticsearch. The application allows for the simultaneous launch of multiple instances on different ports, anticipating the use of a reverse proxy with load balancing to distribute requests across the various instances.

You will create a *systemd* service that receives control instructions on a Unix domain socket to:

- Launch one or more new instances of the Node.js application and add them to the reverse proxy.
- Terminate one or more instances of the Node.js application and remove them from the reverse proxy.
- Deactivate the application in the reverse proxy, stopping all instances, but keeping the configuration.
- (Re)launch all application instances and (re)activate the application in the reverse proxy.

Setup

- Install `npm`, `nodejs` e `nginx`:
  ```
  sudo apt update
  sudo apt install npm nodejs nginx
  ```

- Install `elasticsearch` (choose the appropriate version for your CPU):
  ```
  x86-64 ⇒  wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-8.11.0-amd64.deb
  arm64 ⇒  wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-8.11.0-arm64.deb
  sudo dpkg -i elasticsearch-8.11.0-*.deb
  sudo nano /etc/elasticsearch/elasticsearch.yml
      xpack.security.enabled: false
  ```

- Unpack the attached file `tvs-2324-1_cw3.tgz` (note the uppercase 'C', space, and forward slash /):
  ```
  sudo tar xzvf tvs-2324-1_cw3.tgz -C /
  cd /opt/isel/tvs/tvsapp/app
  npm install
  ```

Preparation

i. Confirm the proper operation of the provided web application:

   a. Launch only the web application, without the database:
   ```
   NODE_PORT=29900 node /opt/isel/tvs/tvsapp/app/tvsapp.js
   ```

   b. Use the browser to access http://localhost:29900
   (you should see PORT: 29900 and *database unavailable*)

   c. Start `elasticsearch`:
   ```
   sudo systemctl start elasticsearch
   ```

   d. Use the browser again to access http://localhost:29900
   (you should see PORT: 29900 and a counter that increments with each new request made)

   e. Terminate the web application (Ctrl-C), but keep `elasticsearch` running

ii. Install the web application as a service and start 4 instances in ports 29901, 29902, 29903, and 29904.
NOTE: see `/opt/isel/tvs/tvsapp/service/` and Service Templates.

iii. Add a configuration for a new site (`tvsapp`) in `/etc/nginx/sites-available` to expose it on port 12021, operating as a load balancer for the 4 local instances on ports 29901 to 29904. Activate the new configuration in `/etc/nginx/sites-enabled` along with the existing one (`default`). Use the browser to access http://localhost:12021 and check the load balancer's operation (PORT changes on refresh).
NOTE: see `/opt/isel/tvs/nginx/sites-available/` and nginx Configuration Control

Exercises

1. Write the following bash scripts to manage the configuration and operation of the proposed solution:
   - `tvsapp-reset.sh` arguments: `scale` (default = 1), `base` (default = 39000)
     Force an initial stopped configuration with `scale` instances in consecutive ports starting at `base`
   - `tvsapp-inc.sh` arguments: `delta` (default = 1)
     Add `delta` instances of Node.js running the web application, using more consecutive ports
   - `tvsapp-dec.sh` arguments: `delta` (default = 1)
     Remove `delta` instances of Node.js with the highest ports in use, leaving at least 1
   - `tvsapp-stop.sh` arguments: `-db` (optional)
     Deactivate the site from `nginx` and stop web app instances. If `-db`, also stop `elasticsearch`.
   - `tvsapp-start.sh`
     Start `elasticsearch`, all web app instances and (re)activate the site in `nginx`
   - `tvsapp-status.sh`
     Write a summary with the solution status, with one line per element (`nginx`, web apps, db)

   You may use the following command to confirm that the load balancer is distributing the requests:

   ```
   seq 32 | xargs -I{} curl -s http://localhost:12021/ | grep "PORT" |
     sed "s/<\/\?[a-z]\+>//g" | sed "s/^[[:space:]]*//" | sort | uniq -c
   ```

   Tag this exercise on the GitHub repository with: **`CW3-1`**

2. Using the C language, build a *systemd* service (`tvsctld`) to receive instructions on a Unix domain socket, to be placed in `/run/isel/tvsctld/request`, and a client program (`tvsctl`) with the following operations:
   - `tvsctl reset [`*`scale`* `[`*`base`*`]]`
   - `tvsctl inc [`*`delta`*`]`.
   - `tvsctl dec [`*`delta`*`]`
   - `tvsctl stop [-db]`
   - `tvsctl start`
   - `tvsctl status`

   The `tvsctld` service is socket activated and runs with *root* privileges. Both the socket and the client (`tvsctl`) will only be available to *root* and members of group `tvsgrp`. to which user `isel` will belong.

   For each execution of the client program (`tvsctl`), a request is sent to the service (`tvsctld`) through the socket. The service *daemon* then invokes the appropriate script (from exercise 1) to perform the requested action.

   Tag this exercise on the GitHub repository with: **`CW3-2`**

Do not submit binaries and other unneeded files to the repository.

For the absolute final version, use the *tag* **`CW3-DONE`** on the GitHub repository.

ISEL, November 10th, 2023

Submission last date: November 26th, 2023