Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Técnicas de Virtualização de Sistemas, Inverno de 2022/2023

Quarta série de exercícios

Tendo como base a mesma aplicação *web* da terceira série de exercícios (`tvsapp`), pretende-se atingir uma solução com o mesmo objetivo geral, mas usando contentores Docker para a distribuir, orquestrando a execução via Docker Compose.

No final pretende-se ter em execução:

- Um contentor com um nó único de `elasticsearch` (8.5.2)
- Um ou mais contentores com a aplicação *web* `tvsapp` a correr sobre Node.js.
  - O número de contentores com réplicas de `tvsapp` é alterável em tempo de execução.
- Um contentor com um *proxy* reverso (`nginx`) a distribuir os pedidos pelas várias réplicas de `tvsapp`.

Preparação

- Copiar os ficheiros `package.json` e `tvsapp.js` usados na série anterior para:
  `[diretoria base de trabalho]/se4/tvsapp/src/`

Exercícios

1. Modificar a aplicação *web* da série anterior nos seguintes dois aspetos:

   a. Quando a variável de ambiente `NODE_PORT` não estiver definida, usar o valor 4004.

   b. Alterar a representação HTML enviada como resposta a um pedido GET para o recurso / para que, em vez de:
   PORT: *número_do_porto*
   seja observado:
   HOST: *nome_do_host*
   PORT: *número_do_porto*
   no *browser* que opera como *user agent*, em que *nome_do_host* é obtido com `os.hostname()`

   Execute `node tvsapp.js` e confirme que em http://localhost:4004/ é visível:
   HOST: isel-virtualbox
   PORT: 4004

   Depois do teste, apague os ficheiros desnecessários e assegure-se que, durante todo o projeto, apenas existem os ficheiros `package.json` e `tvsapp.js` na diretoria `se4/tvsapp/src`

   ```
   rm -Rf package-lock.json node_modules
   ```

2. Criar um `Dockerfile` em `se4/tvsapp/` para criar uma imagem com as seguintes características:
   a. Imagem base: `node:alpine`
   b. `package.json` e `tvsapp.js` colocados em `/home/node/app` e `npm install` executado
   c. Todos os ficheiros e subdiretorias abaixo de `/home/node` devem pertencer a `node:node`
   d. A diretoria de trabalho da imagem é `/home/node/app` e deve ser exposto o porto 4004
   e. Deve-se definir `USER node` e também `CMD ["node","tvsapp.js"]`
   f. Minimize o número de *layers*, mas permitindo um bom reaproveitamento da *build cache* quando o ficheiro `tvsapp.js` for alterado.

   Crie a imagem `tvsapp` a partir do `Dockerfile` e use `docker run` para colocar quatro contentores em execução, sendo possível aceder à aplicação nos portos 4001, 4002, 4003 e 4004 da máquina *host*.

   Deverá conseguir explicar os valores observados em **HOST** e **PORT** nas respostas HTML a GET /

3. Crie um ficheiro `docker-compose.yml` em `se4/` com as seguintes características:
   a. Dois serviços, `tvssvc` e `elasticsearch`, a operarem numa rede (`tvsnet`) em modo *bridge*
   b. Para o serviço `elasticsearch` utilize a imagem correspondente com a *tag* 8.5.2, exponha os portos 9200 e 9300 diretamente para o *host* e defina as variáveis de ambiente:
      ```
      discovery.type=single-node
      xpack.security.enabled=false
      ```
   c. Para o serviço `tvssvc` comece por expor diretamente o porto 4004 para o *host* e defina o necessário para que a aplicação `tvsapp` (inalterada) encontre a instância de `elasticsearch`.

   Certifique-se de que terminou os contentores criados no exercício anterior e execute a solução com:
   ```
   docker compose up -d
   ```

   Com a solução em execução, deverá conseguir:
   - Demonstrar que está operacional
   - Consultar os *logs*
   - Executar um *shell* no contentor `elasticsearch` e mostrar a lista de processos
   - Executar um *shell* no contentor `tvssvc`, confirmar conetividade com o contentor `elasticsearch` e encontrar os endereços IP de ambos os contentores.

4. Modifique a especificação do porto do serviço `tvssvc` para ter apenas o porto do contentor, mas não o do *host*. Encontre duas formas de saber o porto atribuído no *host*, depois de pôr a solução a correr, e confirme que consegue aceder via *browser*. A seguir, utilize a opção `--scale` de `docker compose up` para aumentar o número de instâncias de `tvssvc` para quatro (sem interferir com a que já está em execução). Encontre os portos das quatro instâncias e confirme que está tudo operacional. Verifique que também consegue baixar o número de instâncias para duas.

5. Adicione a `docker-compose.yml` mais um serviço, `entry`, a expor no porto 8088, baseado numa imagem `nginx:alpine`, mas com o ficheiro `/etc/nginx/conf.d/default.conf` alterado. Pode utilizar como base a configuração usada na série de exercícios anterior, alterando:
   - apagar o `upstream` e em `server` mudar o porto para 80
   - dentro de `location`
     - definir uma variável `$TVSSVC` para http://tvssvc:4004
     - definir o `proxy_pass` para reencaminhar para `$TVSSVC`
     - definir: `resolver 127.0.0.11 valid=5s`

   Verifique que a solução final funciona, distribuindo os pedidos que chegam a 8088 pelas instâncias de `tvssvc` que estejam em operação, mesmo que este número seja entretanto alterado.

   Pode usar a seguinte linha de comandos, desde que a consiga explicar:
   ```
   seq 32 | xargs -I{} curl -s  http://localhost:8088/ | grep "HOST" |
     sed "s/<\/\?[a-z]\+>//g" | sed "s/^[[:space:]]*//" | sort | uniq -c
   ```

   Deverá demonstrar porque basta o endereço `tvssvc:4004` para ser realizada a distribuição de pedidos e explicar porque é necessária a opção `valid` no `resolver`.

6. [OPCIONAL] Altere **apenas** o ficheiro `docker-compose.yml` para que os dados do `elasticsearch` (presentes em `/usr/share/elasticsearch/data`) se mantenham entre execuções e para que as definições de rede(s) não permitam o acesso entre os serviços `entry` e `elasticsearch`.

Entrega

Entregue usando a *tag* **SE4** no repositório GitHub.

ISEL, 12 de dezembro de 2022

Data limite de entrega: 7 de janeiro de 2023