3.2. Exercício orientado Criação de imagens de contêiner para o Red Hat OpenShift

Criar imagens de contêiner para o Red Hat OpenShift usando as Red Hat Universal Base Images (UBI).

Resultados

- Criar uma imagem de contêiner com base em uma Red Hat Universal Base Image (UBI).
- Enviar por push a imagem ao container registry.
- Implantar a imagem criada no Red Hat OpenShift.
- Solucionar problemas de permissão na imagem.

Pré-requisitos

Com o usuário `student` na máquina `workstation`, use o comando `lab` para preparar seu sistema para este exercício. Esse comando cria o projeto do OpenShift no qual você deve implantar a imagem de contêiner.

```
[student@workstation ~]$ lab start images-ubi
```

Você encontra a solução deste exercício no diretório `~/DO288/solutions/images-ubi` e nos aplicativos https://github.com/RedHatTraining/DO288-apps/tree/OCP4.14/solutions/images-ubi.

Instruções

Crie e implante uma imagem de contêiner de um aplicativo Node.js, que fornece mensagens de saudação em idiomas aleatórios. O aplicativo escuta a porta `80` e usa um cache de conversão no diretório `/var/cache`.

1. Revise o Containerfile fornecido.
   1. Em uma janela de terminal, navegue até o diretório do exercício.
      ```
      [student@workstation ~]$ cd ~/DO288/labs/images-ubi/greetings
      ```
   2. Inspecione o conteúdo do `Containerfile`, incluído neste diretório. Esse arquivo define as instruções para criar a imagem de contêiner para o aplicativo Node.js.
      ```
      FROM registry.ocp4.example.com:8443/ubi9/nodejs-18-minimal:1-51 ❶

      ENV PORT=80
      EXPOSE ${PORT} ❷

      USER root ❸

      ADD . $HOME ❹

      RUN npm ci --omit=dev && rm -rf .npm ❺
      ```

```
CMD npm start ❻
```

1. ❶ Como a imagem de base, use a versão mínima da imagem do Node.js 18 UBI 9.0.
2. ❷ Faça com que o aplicativo escute na porta 80. O Containerfile define a variável `PORT` porque o aplicativo lê essa variável para determinar a porta.
3. ❸ Execute como o usuário `root` para usar a porta 80, que é uma porta privilegiada.
4. ❹ Copie os arquivos de origem do aplicativo do diretório de workstation `~/DO288/labs/images-ubi/greetings` para o diretório `HOME` do contêiner. As imagens UBI do Node.js definem a variável `HOME` como `/opt/app-root/src`.
5. ❺ Instale dependências para produção. O comando `npm ci` (instalação limpa) é semelhante ao comando `npm install`. Ele remove as dependências atuais e faz o download das versões especificadas das dependências do aplicativo. O comando `rm` remove os arquivos de cache que o NPM cria durante o processo de instalação.
6. ❻ Execute o servidor. O arquivo `package.json` declara o script `start`.

2. Crie e envie por push a imagem de contêiner.
   1. Faça login no container registry.

      ```
      [student@workstation greetings]$ podman login -u developer -p developer \
      registry.ocp4.example.com:8443
      Login Succeeded!
      ```

   2. Crie a imagem de contêiner com o Podman.

      ```
      [student@workstation greetings]$ podman build . \
      -t registry.ocp4.example.com:8443/developer/images-ubi-greetings:1.0.0
      ...output omitted...
      Successfully tagged registry.ocp4.example.com:8443/developer/images-ubi-greetings
      680...01ae
      ```

   3. Envie a imagem de contêiner ao registro de sala de aula interno.

      ```
      [student@workstation greetings]$ podman push \
      registry.ocp4.example.com:8443/developer/images-ubi-greetings:1.0.0
      ...output omitted...
      Writing manifest to image destination
      Storing signatures
      ```

3. Implante a imagem no cluster e verifique se o contêiner falha.
   1. Faça login no cluster como o usuário `developer`.

      ```
      [student@workstation greetings]$ oc login -u developer -p developer \
      https://api.ocp4.example.com:6443
      Login successful.

      ...output omitted...

      Using project "images-ubi".
      ```

   2. Certifique-se de que você esteja usando o projeto `images-ubi`.

```
[student@workstation ~]$ oc project images-ubi
Already on project "images-ubi" on server "https://api.ocp4.example.com:6443".
```

3. Crie um aplicativo no cluster usando a imagem que você acabou de criar. Chame o aplicativo `greetings`.

```
[student@workstation greetings]$ oc new-app \
--name greetings \
--image=registry.ocp4.example.com:8443/developer/images-ubi-greetings:1.0.0
--> Found container image ...

    Node.js 18 Micro
    ----------------
...output omitted...

--> Success
    Application is not exposed. You can expose services to the outside world by e
     'oc expose service/greetings'
    Run 'oc status' to view your app.
```

4. Verifique se o pod do aplicativo está com falha.

```
[student@workstation greetings]$ oc get pods
NAME                    READY   STATUS             RESTARTS   AGE
greetings-...           0/1     CrashLoopBackOff   0          2s
```

5. Confira os logs de implantação e descubra por que o aplicativo falha ao iniciar.

```
[student@workstation greetings]$ oc logs deployments/greetings

> greetings@1.0.0 start
> node index.js

Running with user ID: 1000690000, group ID: 0 ❶
Verifying file cache...
File cache does not work due to [Error: EACCES: permission denied, open '/var/cac
  errno: -13,
  code: 'EACCES',
  syscall: 'open',
  path: '/var/cache/translation_greeting_en-us'
}

...output omitted...

Error: listen EACCES: permission denied 0.0.0.0:80 ❷
    at Server.setupListenHandle
    ...output omitted...
```

1. ❶ O cluster executa o contêiner com um usuário não root e o grupo root.
2. ❷ O aplicativo não tem acesso de gravação a `/var/cache`.

3. ❸ O aplicativo não pode usar a porta privilegiada `80`.

6. Remova o aplicativo e seus recursos associados:

```
[student@workstation greetings]$ oc delete all \
--selector app=greetings
service "greetings" deleted
deployment.apps "greetings" deleted
imagestream.image.openshift.io "greetings" deleted
```

4. Reproduza o problema localmente executando o aplicativo como um usuário não root.

1. Execute a imagem de contêiner com o Podman. Verifique se o aplicativo está sendo executado na porta `80`, usando a ID do usuário `0`, que corresponde ao usuário root, e a ID do grupo `0`, que corresponde ao grupo root.

```
[student@workstation greetings]$ podman run --rm \
registry.ocp4.example.com:8443/developer/images-ubi-greetings:1.0.0

> greetings@1.0.0 start
> node index.js

Running with user ID: 0, group ID: 0
Verifying file cache...
Starting server...
Server listening at http://0.0.0.0:80/
```

O contêiner é executado com êxito porque o Podman está respeitando a instrução `USER root`. Em comparação, o Red Hat OpenShift não respeita a instrução `USER` e executa os contêineres usando um usuário não root arbitrário.

2. Interrompa o contêiner pressionando Ctrl + C.
3. Abra `Containerfile` e remova a instrução `USER root`. O arquivo deve ficar da seguinte maneira:

```
FROM registry.ocp4.example.com:8443/ubi9/nodejs-18-minimal:1-51

ENV PORT=80
EXPOSE ${PORT}

ADD . $HOME

RUN npm ci --omit=dev && rm -rf .npm

CMD npm start
```

4. Recrie a imagem do aplicativo e marque-a como a versão `1.0.1`.

```
[student@workstation greetings]$ podman build . \
-t registry.ocp4.example.com:8443/developer/images-ubi-greetings:1.0.1
...output omitted...
Successfully tagged registry.ocp4.example.com:8443/developer/images-ubi-greetings
680...01ae
```

Certifique-se de usar a nova tag de imagem, `1.0.1`, no restante do exercício.

5. Execute a nova imagem de contêiner. Verifique se o aplicativo é executado usando a ID do usuário `1001`. Ao executar localmente como um usuário não root, esse contêiner exibe os mesmos problemas que um contêiner executado no cluster.

```
[student@workstation greetings]$ podman run --rm \
registry.ocp4.example.com:8443/developer/images-ubi-greetings:1.0.1
...output omitted...

Running with user ID: 1001, group ID: 0
Verifying file cache...
File cache does not work due to [Error: EACCES: permission denied, open '/var/cac

...output omitted...

Error: listen EACCES: permission denied 0.0.0.0:80

...output omitted...
```

5. Corrija o erro da porta e recrie a imagem do contêiner.
   1. Altere a porta no `Containerfile` para `8080`. Essa porta não tem privilégios.

```
ENV PORT=8080
EXPOSE ${PORT}
```

   2. Recrie a imagem do aplicativo como a versão `1.0.1`.

```
[student@workstation greetings]$ podman build . \
-t registry.ocp4.example.com:8443/developer/images-ubi-greetings:1.0.1
...output omitted...
Successfully tagged registry.ocp4.example.com:8443/developer/images-ubi-greetings
680...01ae
```

   3. Execute a imagem de contêiner novamente. O problema da porta foi resolvido, e o aplicativo agora escuta na porta `8080`. O erro de permissão `/var/cache/` ainda persiste.

```
[student@workstation greetings]$ podman run --rm \
registry.ocp4.example.com:8443/developer/images-ubi-greetings:1.0.1
...output omitted...

Running with user ID: 1001, group ID: 0
Verifying file cache...
File cache does not work due to [Error: EACCES: permission denied, open '/var/cac
  errno: -13,
  code: 'EACCES',
  syscall: 'open',
  path: '/var/cache/translation_greeting_en-us'
}
Starting server...
Server listening at http://0.0.0.0:8080/
```

4. Interrompa o contêiner pressionando `Ctrl` + `C`.

6. Corrija o erro de permissão do arquivo, crie e envie a imagem de contêiner por push.

   1. Corrija as permissões do diretório `/var/cache`.

      Volte para o `Containerfile`. Como o usuário root, certifique-se de que o grupo atribuído ao diretório `/var/cache` seja o grupo root (`0`). Em seguida, conceda ao grupo root as mesmas permissões que o usuário proprietário desse diretório. Por fim, restaure `1001` como a ID do usuário que executa o aplicativo.

      ```
      FROM registry.ocp4.example.com:8443/ubi9/nodejs-18-minimal:1-51

      ENV PORT=8080
      EXPOSE ${PORT}

      ADD . $HOME

      RUN npm ci --omit=dev && rm -rf .npm

      USER root
      RUN chgrp -R 0 /var/cache && \
          chmod -R g=u /var/cache
      USER 1001

      CMD npm start
      ```

   2. Recrie a imagem do aplicativo como a versão `1.0.1`.

      ```
      [student@workstation greetings]$ podman build . \
      -t registry.ocp4.example.com:8443/developer/images-ubi-greetings:1.0.1
      ...output omitted...
      Successfully tagged registry.ocp4.example.com:8443/developer/images-ubi-greetings
      680...01ae
      ```

   3. Execute a imagem de contêiner novamente. O aplicativo é executado como o usuário `1001` na porta `8080`.

      ```
      [student@workstation greetings]$ podman run --rm \
      registry.ocp4.example.com:8443/developer/images-ubi-greetings:1.0.1

      > greetings@1.0.0 start
      > node index.js

      Running with user ID: 1001, group ID: 0
      Verifying file cache...
      Starting server...
      Server listening at http://0.0.0.0:8080/
      ```

   4. Pressione `Ctrl` + `C` para interromper o aplicativo.

   5. Envie a tag `1.0.1` ao registro.

```
[student@workstation greetings]$ podman push \
registry.ocp4.example.com:8443/developer/images-ubi-greetings:1.0.1
...output omitted...
Writing manifest to image destination
Storing signatures
```

7. Implante a imagem atualizada:
   1. Recrie o aplicativo usando a versão `1.0.1` da imagem.

```
[student@workstation greetings]$ oc new-app \
--name greetings \
--image=registry.ocp4.example.com:8443/developer/images-ubi-greetings:1.0.1
...output omitted...
--> Success
    Application is not exposed. You can expose services to the outside world by e
     'oc expose service/images-ubi-greetings'
    Run 'oc status' to view your app.
```

   2. Verifique se o pod do aplicativo inicia corretamente. Aguarde até que o pod esteja no estado `Running`.

```
[student@workstation greetings]$ oc get pods
NAME                    READY   STATUS    RESTARTS   AGE
greetings-...   1/1     Running   0          45s
```

   3. Encontre o pod do aplicativo e inspecione os logs. Verifique se os logs não mostram problemas.

```
[student@workstation greetings]$ oc logs deployments/greetings

> greetings@1.0.0 start
> node index.js

Running with user ID: 1000690000, group ID: 0
Verifying file cache...
Starting server...
Server listening at http://0.0.0.0:8080/
```

   4. Exponha o aplicativo.

```
[student@workstation greetings]$ oc expose svc/greetings
route.route.openshift.io/greetings exposed
```

   5. Obtenha a URL da rota.

```
[student@workstation greetings]$ oc get route greetings
NAME        HOST/PORT                                       ...
greetings   greetings-images-ubi.apps.ocp4.example.com     ...
```

   6. Faça uma solicitação para verificar se o aplicativo responde com êxito.

```
[student@workstation greetings]$ curl -s \
http://greetings-images-ubi.apps.ocp4.example.com | jq
{
```

```
    "message": "Guten tag"
}
```

Encerramento

Use o comando `lab` na máquina `workstation` para concluir este exercício. Essa etapa é importante para garantir que recursos de exercícios anteriores não afetem exercícios futuros.

```
[student@workstation ~]$ lab finish images-ubi
```
