Docker I: um prólogo prático

Convencido de que conteinerização é útil? Então vamos conteinerizar uma aplicação bobinha para entendermos como isso é feito na prática.

Nesse tutorial, irei criar um app superficial em node.js, e encapsulá-lo em um container de Docker, e logo em seguida executá-lo.

Parte I: Construindo nosso app bobo

primeiro, vamos instalar o que é necessário: `npm` e `docker`. Essas instruções são para instalar npm e docker em Debian/Ubuntu, confira como é feito no seu próprio sistema.

```
sudo apt update
sudo apt install npm docker
```

agora, crie uma nova pasta. vamos trabalhar nela o tempo todo.

```
mkdir docker_intro
cd docker_intro
```

agora comece um repositório de NPM (Node Packet Manager) dentro da pasta. O NPM será útil pra instalarmos pacotes de Javascript para o nosso programa.

```
npm init -y
```

Isso cria um arquivo chamado `package.json` com as seguintes informações dentro:

```
{
  "name": "docker_1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
```

```
  "license": "ISC"
}
```

Vamos alterar uma linha: trocar a linha `"test": "echo \"Error: no test specified\" && exit 1"` para `"start": "node index.js"`. Assim, quando rodarmos `npm start`, o NPM executará `node index.js` para nós.

O arquivo fica assim então:

```
{
  "name": "docker_1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Para esse exercício, vamos instalar o pacote ramda, apenas para provar que o container conseguirá instalar esse pacote também.

```
npm install ramda
```

Agora vamos criar o arquivo `index.js` e colocar o seguinte "Hello World" dentro dele:

```
const { applyTo } = require('ramda')

const withHelloWorld = applyTo('Hello, World!')

withHelloWorld(console.log)
```

Adendo: Se você quer entender o que está acontecendo nessas linhas, você pode pensar a função applyTo como

```
const applyTo = (values) => (func) => func(values)
```

Ou seja, uma função que recebe valores e devolve uma função que recebe uma função e devolve

ela aplicada nos valores. *Confuso, eu sei.* Se quiser saber mais, procure saber um pouco sobre **programação funcional**. Essa apresentação parece ser uma boa introdução ao assunto.

Voltando aos trilhos. Ao rodar `npm start` no nosso console, se tudo ocorreu certo, deveriamos ver isso:

```
$ npm start

> docker_1@1.0.0 start /home/raz/infosec/wiki/docker_1
> node index.js

Hello, World!
```

Ótimo! Fora algumas instruções de debug, vemos o nosso `Hello, World!`.

Parte II: Conteinerizando o app

Temos nosso hello world funcionando. Agora vem a parte de colocá-lo dentro de um container. E você vai ver que é *super* fácil.

Crie um arquivo chamado `Dockerfile` na pasta do projeto. Esse arquivo é uma "receita" de como construir um container com nosso app. Coloque isso dentro do arquivo:

```
# Herda da imagem do debian
FROM debian

# Instala o npm
RUN apt update
RUN apt install -y npm
# Atualiza o npm
RUN npm install -g npm

# Cria o diretório /app
# e instala as dependências do /app lá
# note que WORKDIR tanto cria o diretório
# quanto troca para ele (cd /app)
WORKDIR /app
COPY package.json .
COPY package-lock.json .
RUN npm install
```

```
# copia o nosso app para dentro da imagem
COPY index.js .
# Esse é o comando que será rodado quando você executar o `docker run`
CMD ["npm", "start"]
```

Não precisa pensar muito sobre o que tem dentro desse arquivo ainda não. Já vamos falar sobre ele.

Agora, rode o seguinte comando **(OBS: Você talvez precise rodar os próximos comandos com `sudo`)**:

```
docker build . -t meu-app
```

Leia esse comando como "Construa o container da pasta `.` e dê um nome de `meu-app` a ele". Esse comando vai demorar para terminar, e uma conexão boa com a internet ajuda bastante.

Com esse ultimo comando acabando com sucesso, é só rodar o container!

```
docker run meu-app
```

No meu caso, aconteceu o seguinte:

```
$ docker run meu-app

> docker_1@1.0.0 start /app
> node index.js

Hello, World!
```

Perfeito! Igualzinho ao nosso da parte I, só que esse comando rodou dentro de um container isolado!

Na próxima página, vamos discutir o que na verdade aconteceu por baixo dos panos.

Revision #4
Created Sun, Aug 25, 2019 8:18 PM by razgrizone
Updated Tue, Sep 10, 2019 12:33 PM by Cainotis