Configuração e publicação de projetos com Docker, Vue.js e Nuxt.

Autor: William Costa

Fala devs, blz?

Neste documento vou descrever como configurar, compilar e executar um projeto Vue.js + Nuxt com Docker em produção.
O ambiente de produção precisa ter Docker instalado e estar com a porta 80 exposta (configuração padrão de servidores da Infinite).

Qualquer dúvida é só avisar :D

1 – Configurações do projeto

O projeto deve possuir alguns arquivos na raiz do repositório para que o ambiente Docker possa ser compilado e que os comandos NPM sejam executados corretamente durante esse procedimento.

1.1 – Dockerfile

Arquivo responsável por configurar o ambiente Docker e indicar quais imagens serão carregadas.
Em nosso ambiente utilizaremos apenas uma imagem com NodeJS para realizar a compilação e servir o ambiente via NPM.

```
FROM node:12
```

```
ARG npm_token_deploy

ENV NUXT_VERSION=2.15.8

ENV NPM_TOKEN=$npm_token_deploy

ENV HOST=0

ENV PORT=80

WORKDIR /app

COPY . .
RUN : \
  && npm install \
  && npm run docker-build \
  && :

ENTRYPOINT ["npm", "start"]
EXPOSE 80
```

1.2 – .dockerignore

Da mesma forma que o .gitignore indica para o Git os arquivos e pastas que não devem ser considerados no projeto, o .dockerignore faz a mesma ação porém nesse caso é o docker que deve ignorar os arquivos e pastas.
Em nosso projeto utilizaremos para ignorar a pasta `.git` e `node_modules` para deixar a ação de build mais rápida.

```
node_modules
.git
```

1.3 – .npmrc

Este arquivo é responsável por gerenciar as regras dos pacotes NPM e em nosso projeto utilizaremos ele para aplicar o token de autenticação em todas as requests para o registry.npmjs.org.

```
//registry.npmjs.org/:_authToken=${NPM_TOKEN}
```

1.4 – package.json

O arquivo package.json já existe no repositório, porém é necessário criar o script "docker-build" que será executado em uma das instruções do Dockerfile para gerar a versão standalone a aplicação.

```
{
  [...]
  "scripts": {
     [...]
     "docker-build": "nuxt build --standalone"
   }
  [...]
}
```

Baixando o projeto manualmente no servidor

Acesse via SSH o servidor que deseja rodar o projeto e, em seguida, rode o comando abaixo para clonar o projeto na pasta desejada:

```
git clone https://gitlab.com/nome-do-projeto.git nome-da-pasta-do-projeto
```

Troque a URL e o nome da pasta pelos dados do repositório e nome do projeto que deseja utilizar.
Caso o projeto já esteja baixado no servidor, utilize o comando PULL do Git.

Compilando a imagem Docker

Para compilar a imagem Docker basta acessar a raiz do projeto (após fazer todas as configurações acima) e rodar o compando abaixo.
Importante: onde está descrito `nome-do-projeto` você pode indicar o nome que a imagem terá após ser compilada (esse nome será utilizado na execução) e onde está descrito como `NPM_TOKEN` você deve substituir pelo token da conta da web.art group no NPM para que os pacotes restritos possam ser baixados durante a compilação.

```
docker build --build-arg npm_token_deploy={NPM_TOKEN}  -t nome-do-projeto .
```

Executando a imagem compilada

Para executar a imagem compilada basta executar o comando abaixo alterando o `nome-do-projeto` para o nome definido durante a compilação.

```
docker run --rm -it -p 80:80 nome-do-projeto
```

Importante: Caso existam mais containeres rodando no servidor, será necessário alterar a porta do HOST. Dentro do comando RUN acima estão descritas duas portas 80, sendo que a porta da esquerda (antes dos dois pontos) é a porta do HOST (servidor) já a porta da direita (depois dos dois pontos) é a porta do container (será sempre 80).
Para alterar a porta do host, basta executar o comando assim:

```
docker run --rm -it -p 8080:80 nome-do-projeto
```

Executando a aplicação em uma screen

Quando executamos a imagem em um ambiente de produção é importante que ela continue funcionando após fecharmos o terminal. Para nos auxiliar nisso existem as screens que são sessões que podem ser abertas no terminal e que não precisam de um terminal conectado para continuarem funcionando.

Para acessar uma screen utilize o comando abaixo:

```
screen -R
```

Execute o comando de execução da imagem (ou `ctrl+c` para parar a execução atual, caso seja uma screen que já estava com a aplicação aberta).

```
docker run --rm -it -p 80:80 nome-do-projeto
```

Para sair da screen sem finaliza-la basta executar `ctrl+a d` (aperte `ctrl+a` e, sem soltar `ctrl`, aperte `d`).

Descobrindo o ID do container em execução

Após executar o projeto (docker run) um novo processo será criado e ele terá um ID único para essa execução.
Para obter detalhes dos processos em execução, basta utilizar o comando abaixo:

```
docker ps
```

Após executar esse comando, o resultado na tela trará todos os processos docker rodando no momento e basta encontrar a linha que tem o `nome-do-projeto` que você definidou e terá o ID de sua execução.

Importante: caso tenha executado o comando RUN dentro de uma screen é necessário sair da screen antes de executar o comando.