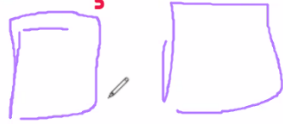


Aula 04 - Benefícios de Infra como Código

- 1) Redução de erro humano
- 2) Repetibilidade e previsibilidade
- 3) Redução de tempo e desperdício

3. Redução de tempo e desperdício



O responsável pela operação de nossa infraestrutura poderá fazê-lo em questão de minutos e sem a necessidade de instalação de nenhum componente extra. Sempre poderemos ativar o processo para fazer o seu trabalho!



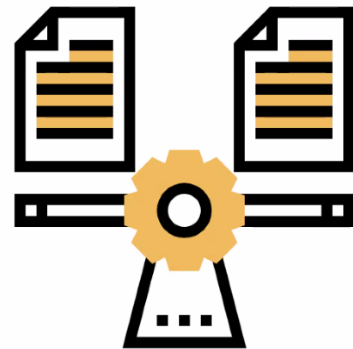
- 4) Controle de versão

4. Controle de versão

Nossa infraestrutura vai ser definida em arquivos, então poderemos versionar - assim como o código fonte de nossa aplicação - **em templates**.

Vamos extrair o conteúdo dos templates!

Podemos usar parâmetros para escrever nosso código da maneira mais genérica possível. Então, ao executá-lo, poderemos enviar diferentes dados na forma de parâmetros para que nosso código os receba.



- 5) Redução de custos

5. Redução de custos

Ao automatizar processos, podemos nos concentrar em outras tarefas e melhorar o que já foi feito. Isso fornece flexibilidade para que as equipes de infraestrutura cubram mais tarefas.



Docker File, Docker Compose acaba criando uma infra, uma estrutura. Ele sobe uma infra para a gente. Por exemplo, não preciso de 5 pessoas criando um container se eu tenho um Docker Compose, pois ele já vai subir sozinho.

- 6) Testes

6. Testes

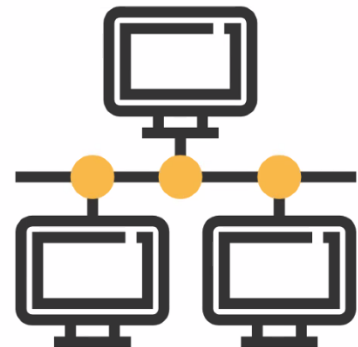
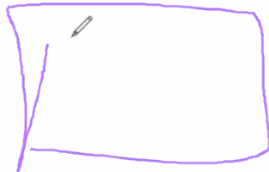
A infraestrutura como código permite que as equipes de infraestrutura **testem os aplicativos** em qualquer ambiente (incluindo produção) no início do ciclo de desenvolvimento.



- 7) Ambientes estáveis e escaláveis

7. Ambientes estáveis e escaláveis

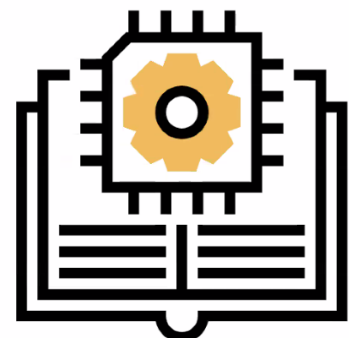
Ao evitar configurações manuais, a falta de dependências e obter o estado final da infraestrutura de que precisamos para nossas aplicações, oferecemos **ambientes estáveis e escaláveis**.



- 8) Padronização de configuração

8. Padronização de configuração

A padronização das configurações e a implantação da infraestrutura nos permite evitar qualquer problema de incompatibilidade com nossa infraestrutura e que as aplicações rodem com o **melhor desempenho possível**.



- 10) Documentação

9. Documentação

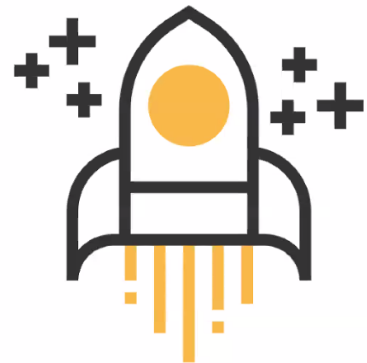
Ao contribuir para a documentação dos processos internos das nossas equipas vamos melhorar os tempos e custos. Como já vimos, podemos criar versões de nossas automações. Este recurso permite que **cada alteração seja documentada**, registrada pelo usuário e com um rápido rollback se encontrarmos erros nas implantações, assim como no código-fonte.



- 11) Mais rápido, sem negligenciar a segurança

10. Mais rápido, sem negligenciar a segurança

Ao melhorar nossa infraestrutura, nunca devemos parar de pensar na segurança que a compõe. Na hora de padronizar a execução da infraestrutura, podemos também **padronizar os grupos de segurança com as permissões mínimas**, mas necessárias para que todas as equipes possam trabalhar e evitar tarefas manuais pelas equipes de segurança.

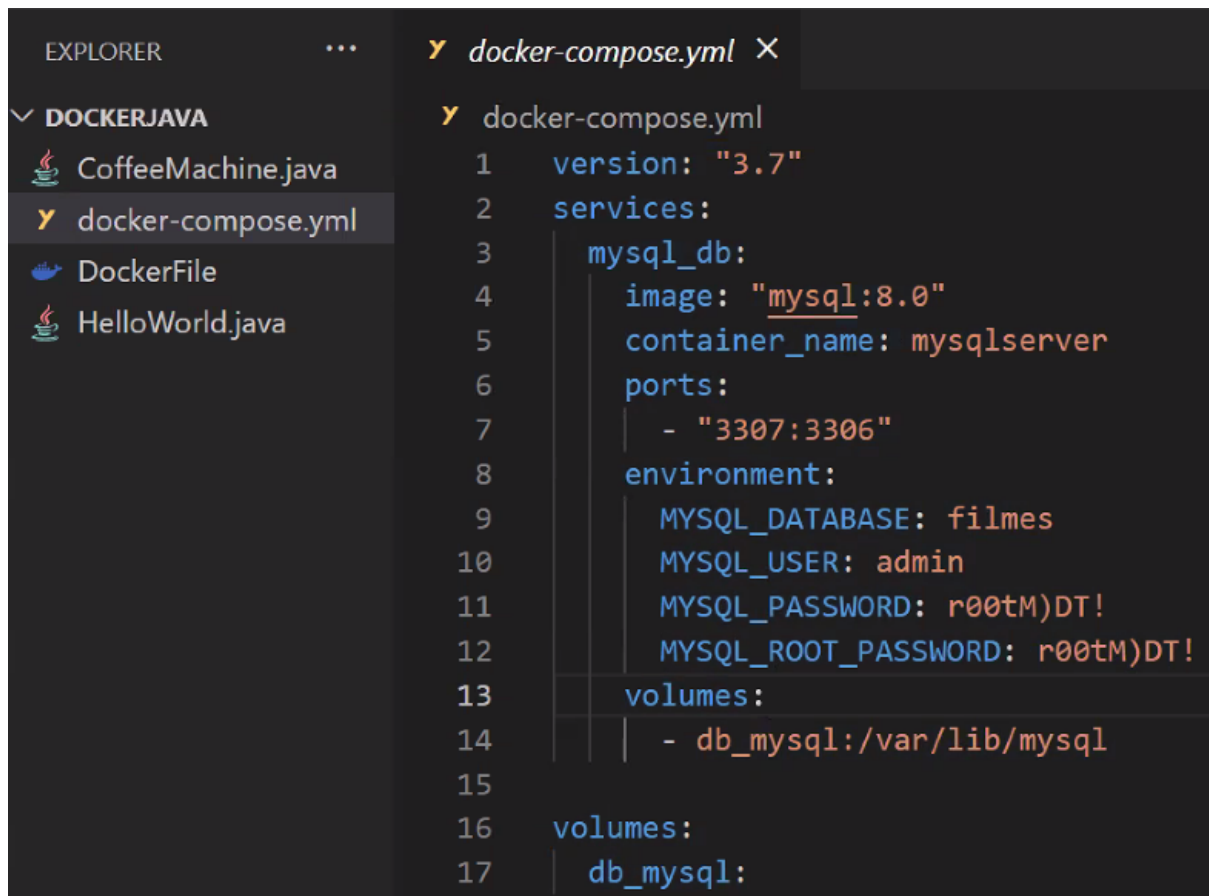


Demais anotações:

- docker run ubuntu

```
EXPLORER    ...    DockerFile x
└─ DOCKER/JAVA
  └─ CoffeeMachine.java
  └─ docker-compose.yml
  └─ DockerFile
  └─ HelloWorld.java

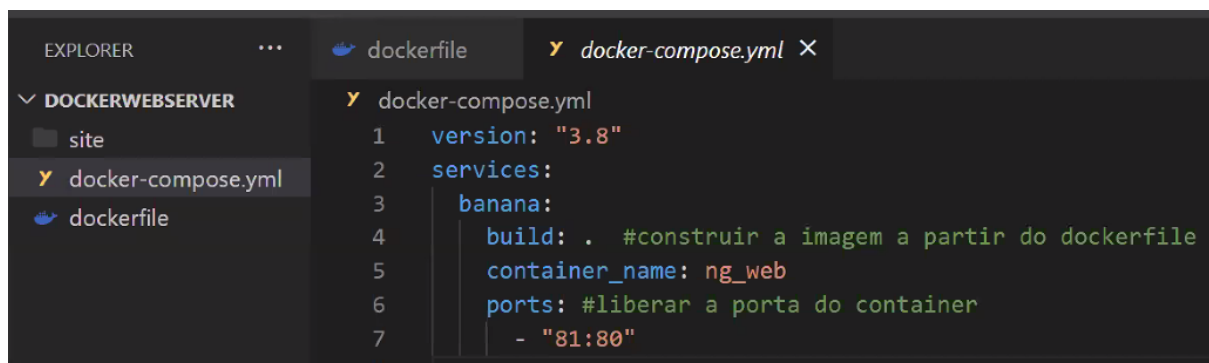
DockerFile > ...
1  #dockerfile serve para gerar uma imagem personalizada
2
3  # FROM ow dockerhub me dá uma imagem com openjdk
4  FROM openjdk:11
5
6  # seta um diretório principal, se o diretório existir ele seta, se não cria e seta
7  WORKDIR /laranja/
8
9  # copia um arquivo para o diretório selecionado no momento de criação/build da imagem
10 COPY HelloWorld.java /laranja/
11
12 #run abre um cmd/terminal(de mentira, não abre nada, mas faça de conta que sim) e gera
13 #um arquivo .class da nossa aplicação java, no momento da criação da imagem
14 RUN ["javac", "HelloWorld.java"]
15
16 #é um comando especial que só executa quando o container é iniciado,
17 #executa nosso arquivo .class no momento que o container é iniciado.
18 CMD ["java", "HelloWorld"]
```



The screenshot shows the VS Code interface with the Explorer on the left and the Editor on the right. The Explorer shows a project named 'DOCKERJAVA' with files 'CoffeeMachine.java', 'docker-compose.yml', 'DockerFile', and 'HelloWorld.java'. The 'docker-compose.yml' file is selected and its content is displayed in the Editor. The file is a Docker Compose configuration for a MySQL database. It specifies version '3.7', a service named 'mysql_db' using the 'mysql:8.0' image, container name 'mysqlserver', and ports '3307:3306'. The environment variables are 'MYSQL_DATABASE: filmes', 'MYSQL_USER: admin', 'MYSQL_PASSWORD: r00tM)DT!', and 'MYSQL_ROOT_PASSWORD: r00tM)DT!'. There is also a volume named 'db_mysql' mapped to '/var/lib/mysql'.

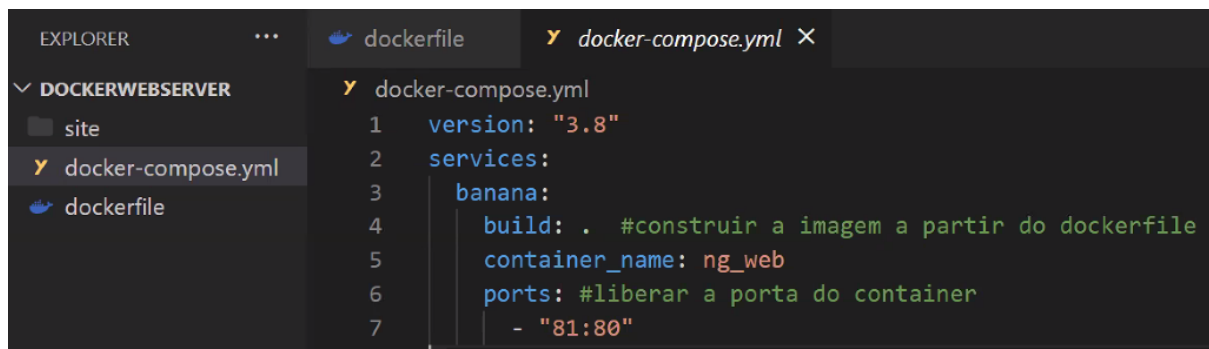
```
1 version: "3.7"
2 services:
3   mysql_db:
4     image: "mysql:8.0"
5     container_name: mysqlserver
6     ports:
7       - "3307:3306"
8     environment:
9       MYSQL_DATABASE: filmes
10      MYSQL_USER: admin
11      MYSQL_PASSWORD: r00tM)DT!
12      MYSQL_ROOT_PASSWORD: r00tM)DT!
13   volumes:
14     - db_mysql:/var/lib/mysql
15
16 volumes:
17   db_mysql:
```

AGORA SIM...



The screenshot shows the VS Code interface with the Explorer on the left and the Editor on the right. The Explorer shows a project named 'DOCKERWEBSERVER' with files 'site', 'docker-compose.yml', and 'dockerfile'. The 'docker-compose.yml' file is selected and its content is displayed in the Editor. The file is a Docker Compose configuration for a web server. It specifies version '3.8', a service named 'banana' using the 'build: .' instruction, container name 'ng_web', and ports '81:80'.

```
1 version: "3.8"
2 services:
3   banana:
4     build: . #construir a imagem a partir do dockerfile
5     container_name: ng_web
6     ports: #liberar a porta do container
7       - "81:80"
```



This screenshot is identical to the one above, showing the VS Code interface with the Explorer on the left and the Editor on the right. The Explorer shows a project named 'DOCKERWEBSERVER' with files 'site', 'docker-compose.yml', and 'dockerfile'. The 'docker-compose.yml' file is selected and its content is displayed in the Editor. The file is a Docker Compose configuration for a web server. It specifies version '3.8', a service named 'banana' using the 'build: .' instruction, container name 'ng_web', and ports '81:80'.

```
1 version: "3.8"
2 services:
3   banana:
4     build: . #construir a imagem a partir do dockerfile
5     container_name: ng_web
6     ports: #liberar a porta do container
7       - "81:80"
```