05

Faça o que eu fiz na aula

Bom pessoal, vamos agora configurar os testes de nossa aplicação, pois um dos problemas que temos no setor de desenvolvimento são os códigos que estão subindo para produção sem antes passar pelos testes, devido aos prazos muito curtos. Nossos desenvolvedores sempre queimam a etapa de testes e isso gera um enorme problema em produção, pois por várias vezes, temos trechos que não testamos ou não dimensionamos corretamente o impacto das melhorias implantadas em produção.

Para realizarmos os testes da aplicação, o ideal é que eles sejam rodados em um ambiente semelhante ao de produção. Como no passo anterior geramos uma imagem semelhante a de produção e a colocamos no docker hub, vamos utilizá-la também no passo de testes.

Para isso vamos adicionar um novo passo com o stage de testes:

```
test-project:
  image: jnlucas/minha-imagem:latest
  stage: test
  services:
  - docker:dind
  dependencies:
  - build-project
  tags:
  - executor-tarefas
  script:
  - python -m unittest setUp
```

Para este passo, notem que estamos utilizando o mesmo runner e a mesma imagem de produção que criamos para o build, assim garantimos um ambiente muito próximo ao ambiente de produção. Como nosso projeto tem apenas um teste, adicionei a chamada dele na tag script. Dessa forma, caso o teste passe, nosso pipeline terá mais um passo com sucesso. Vamos então dar mais um push em nosso projeto e veremos o que acontece em nossa pipeline.

```
git add --all
git commit -m “adicionando job de testes unitários”
git push gitlab master
```

Notem que aparecerá uma nova tarefa no pipeline, e ela será executada com sucesso somente após a tarefa de build ser concluída também com sucesso.