05

Consolidando o seu conhecimento

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com a próxima aula.

1) Crie o *job* para colocar a aplicação em produção. Você pode se guiar pelo script que o instrutor seguiu durante a aula:

```
# Criar o job para colocar a app em producao:
    Nome: todo-list-producao
    Tipo: Freestyle
    # Este build é parametrizado com 2 Builds de Strings:
        Nome: image
        Valor padrão: - Vazio, pois o valor sera recebido do job anterior.

        Nome: DOCKER_HOST
        Valor padrão: tcp://127.0.0.1:2376

    # Ambiente de build > Provide configuration files
        File: .env-prod
        Target: .env

    # Build > Executar shell
        #Execute shell
        #!/bin/sh
        {
            docker run -d -p 80:8000 -v /var/run/mysqld/mysqld.sock:/var/run/mysqld/mysqld.sock -v /

        } || { # catch
            docker rm -f django-todolist-prod
            docker run -d -p 80:8000 -v /var/run/mysqld/mysqld.sock:/var/run/mysqld/mysqld.sock -v /
        }

Ações de pós-build > Slack Notifications: Notify Success e Notify Every Failure
```

2) Faça a integração do *jobs*. Você pode se guiar pelo script que o instrutor seguiu durante a aula:

```
# Post build actions para os 3 jobs

Job: jenkins-todo-list-principal > Ações de pós-build > Trigger parameterized buld on other projects
    Projects to build: todo-list-desenvolvimento
    # Add parameters > Predefined parameters
        image=${image}

Job: todo-list-desenvolvimento

    pipeline {
        environment {
            dockerImage = "${image}"
        }
        agent any
```

```
stages {
    stage('Carregando o ENV de desenvolvimento') {
        steps {
            configFileProvider([configFile(fileId: '2ed9697c-45fc-4713-a131-53bdbeea2ae6', \
                sh 'cat $env > .env'
            }
        }
    }
    stage('Derrubando o container antigo') {
        steps {
            script {
                try {
                    sh 'docker rm -f django-todolist-dev'
                } catch (Exception e) {
                    sh "echo $e"
                }
            }
        }
    }
    stage('Subindo o container novo') {
        steps {
            script {
                try {
                    sh 'docker run -d -p 81:8000 -v /var/run/mysqld/mysqld.sock:/var/run/mys
                } catch (Exception e) {
                    slackSend (color: 'error', message: "[ FALHA ] Não foi possivel subir o
                    sh "echo $e"
                    currentBuild.result = 'ABORTED'
                    error('Erro')
                }
            }
        }
    }
    stage('Notificando o usuario') {
        steps {
            slackSend (color: 'good', message: '[ Sucesso ] O novo build esta disponivel em:
        }
    }
    stage ('Fazer o deploy em producao?') {
        steps {
            script {
                slackSend (color: 'warning', message: "Para aplicar a mudança em produção, a
                timeout(time: 10, unit: 'MINUTES') {
                    input(id: "Deploy Gate", message: "Deploy em produção?", ok: 'Deploy')
                }
            }
        }
    }
    stage (deploy) {
        steps {
            script {
                try {
                    build job: 'todo-list-producao', parameters: [[$class: 'StringParameter\
                } catch (Exception e) {
                    slackSend (color: 'error', message: "[ FALHA ] Não foi possivel subir o
                    sh "echo $e"
                    currentBuild.result = 'ABORTED'
                    error('Erro')
```

```
                    }
                }
            }
        }
    }
}
```