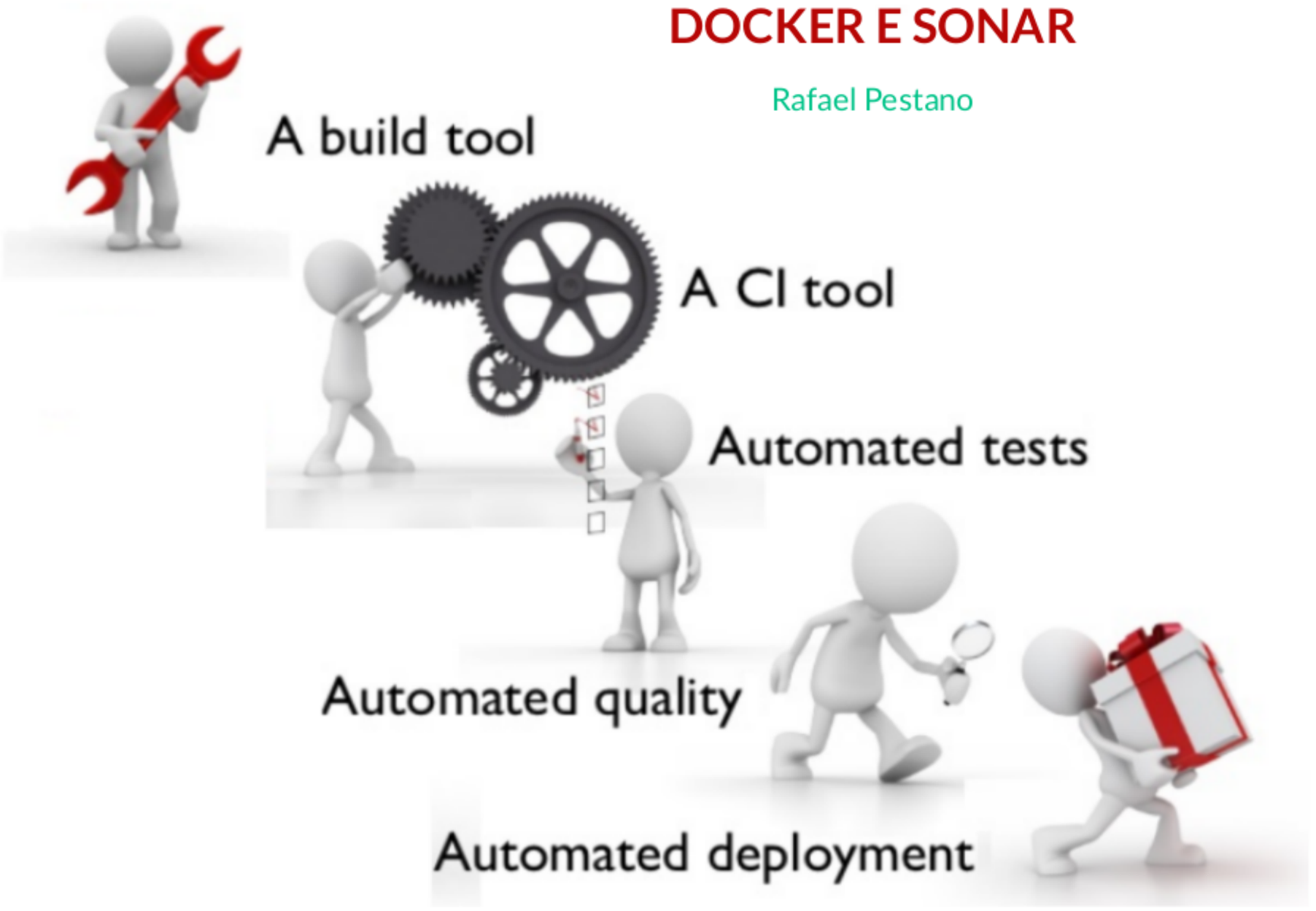
JAVAEE PIPELINE COMO CÓDIGO USANDO JENKINS, DOCKER E SONAR

Rafael Pestano



RAFAEL PESTANO

Desenvolvedor Java na PROCERGS

- @realpestano
- @rmpestano
- Blog

AGENDA

- Continuous Delivery
- Pipeline
- Pipeline
- Pipeline
- Pipeline
- Pipeline
- Pipeline
- Referências

CONTINUOUS DELIVERY

"A software strategy that enables organizations to deliver new features to users as fast and efficiently as possible"

CONTINUOUS DELIVERY

> *"A software strategy that enables organizations to deliver new features to users as fast and efficiently as possible"*

- É estar pronto para entregar em produção a qualquer momento!

POR QUE?

> *"... If it hurts, do it more frequently, and bring the pain forward."*

— Jez Humble

OBJETIVOS

- Reduzir o risco de uma entrega
- Criar um processo bem definido de release
- Tornar o processo de entrega menos doloroso e com menos surpresas

DEPLOY PARA PRODUÇÃO

PRINCÍPIOS

PRINCÍPIOS

- Cada commit gera um release candidate

> *“The longer you delay, the worse(exponentially) the problem becomes“ [Neal Ford - Director at ThoughtWorks]*

PRINCÍPIOS

- Cada commit gera um release candidate

> *“The longer you delay, the worse(exponentially) the problem becomes“ [Neal Ford - Director at ThoughtWorks]*

- Automatize tudo que pode ser automatizado

PRINCÍPIOS

- Cada commit gera um release candidate

> *“The longer you delay, the worse(exponentially) the problem becomes“ [Neal Ford - Director at ThoughtWorks]*

- Automatize tudo que pode ser automatizado
- Testes automatizados são essenciais

PRINCÍPIOS

- Cada commit gera um release candidate

> *“The longer you delay, the worse(exponentially) the problem becomes“ [Neal Ford - Director at ThoughtWorks]*

- Automatize tudo que pode ser automatizado
- Testes automatizados são essenciais
- Feedback rápido e contínuo

PRINCÍPIOS

PRINCÍPIOS

- Melhoria contínua

PRINCÍPIOS

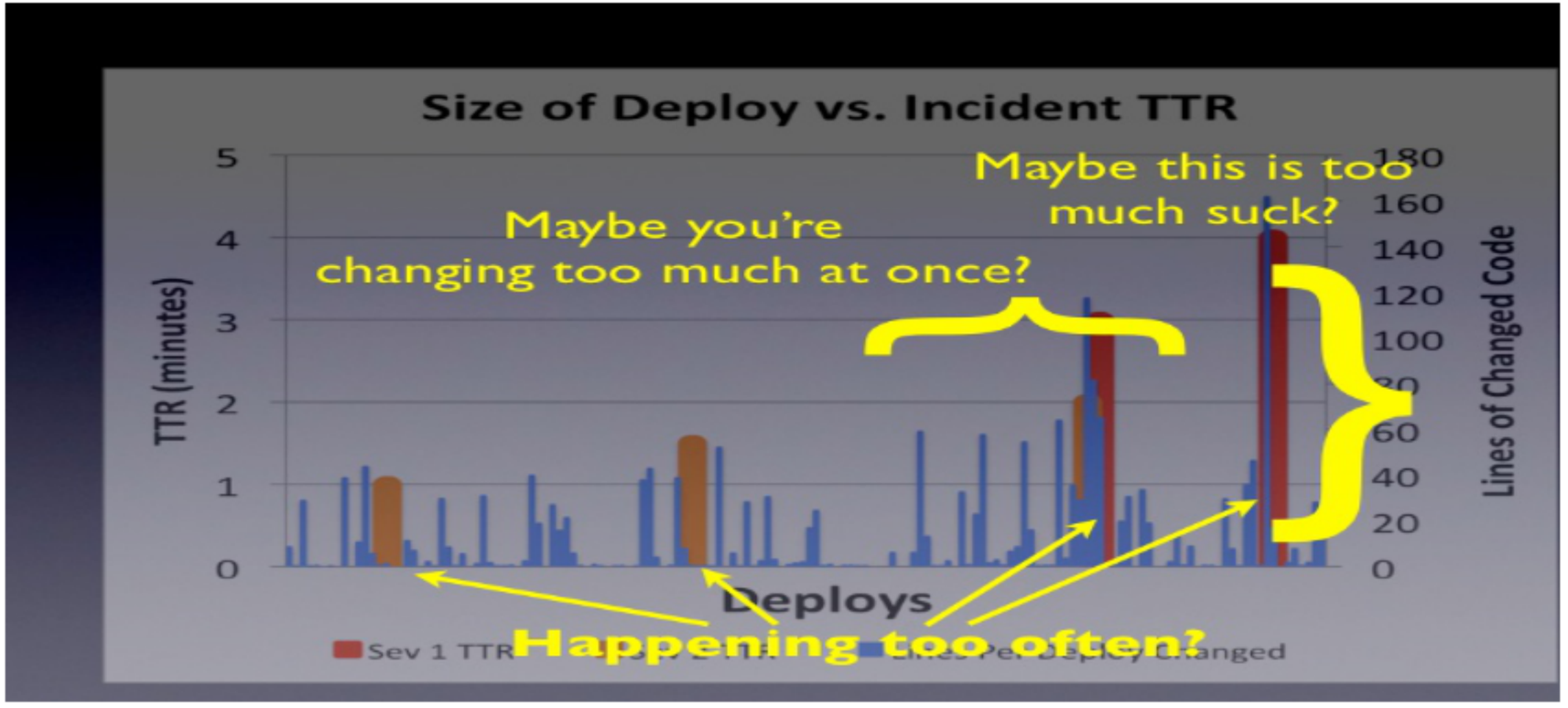
- Melhoria contínua
- Colaboração, todo mundo é responsável pela release

PRINCÍPIOS

- Melhoria contínua
- Colaboração, todo mundo é responsável pela release
- Progresso mensurável
    1. Quantos builds falharam?
    2. Em qual etapa falhou?
    3. Quanto tempo para colocar uma versão em produção?

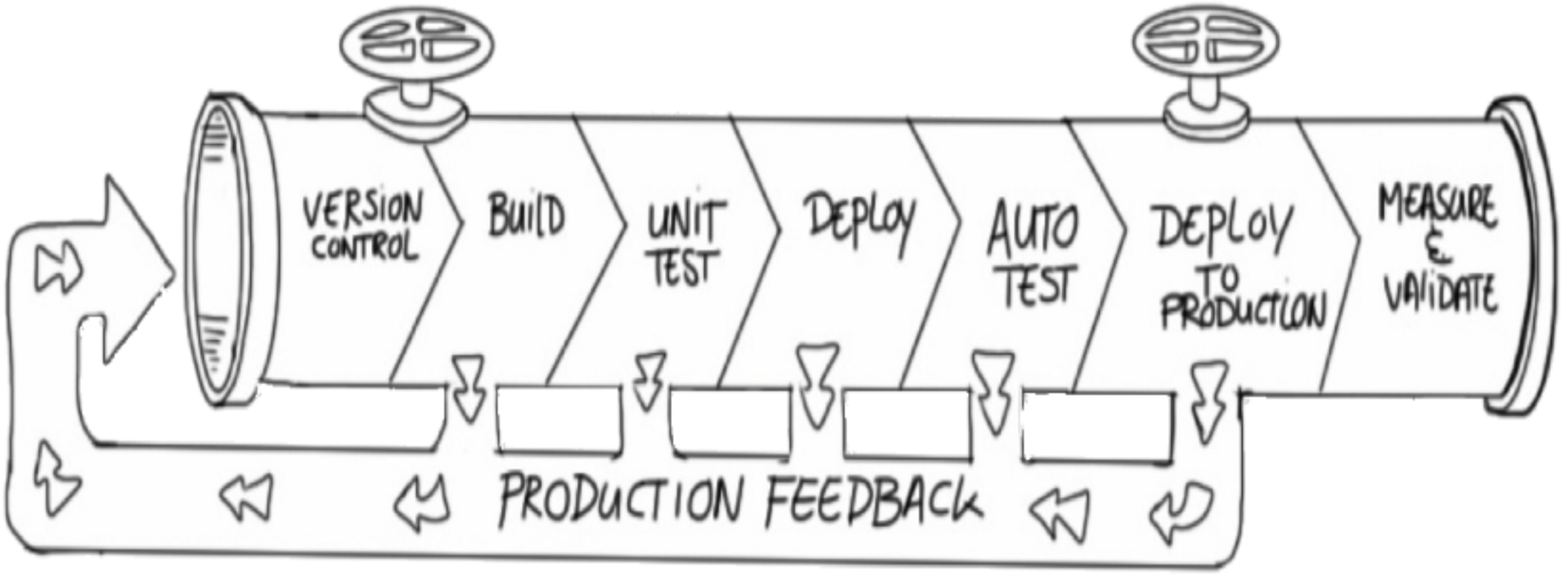
CONSEQUÊNCIAS

- Menor severidade e frequência de falhas relacionadas a um release
- Menor tempo de recuperação à falhas (MTTR)



DEPLOYMENT PIPELINE

"... A pipeline is a set of stages to bring funcionality from developers to end users"



JENKINS 1.X PIPELINE

JENKINS 2.X PIPELINE

Declarative Pipeline Syntax 1.0 is now available

Published on 2017-02-03 by Patrick Wolf

pipeline blueocean

Tweet

I am very excited to announce the addition of Declarative Pipeline syntax 1.0 to Jenkins Pipeline. **We think this new syntax will enable everyone involved in DevOps, regardless of expertise, to participate in the continuous delivery process**.

Whether creating, editing or reviewing a pipeline, having a straightforward structure helps to understand and predict the flow of the pipeline and provides a common foundation across all pipelines.

Pipeline as Code

Pipeline as Code was one of the pillars of the Jenkins 2.0 release and an essential part of implementing continuous delivery (CD). Defining all of the stages of an application's CD pipeline within a `Jenkinsfile` and checking it into the repository with the application code provides all of the benefits inherent in source control management (SCM):

- Retain history of all changes to Pipeline
- Rollback to a previous Pipeline version
- View diffs and merge changes to the Pipeline
- Test new Pipeline steps in branches
- Run the same Pipeline on a different Jenkins server

JENKINS 2.X PIPELINE

JENKINS 2.X PIPELINE

- Versionamento

JENKINS 2.X PIPELINE

- Versionamento
- Reutilização (libraries)

JENKINS 2.X PIPELINE

- Versionamento
- Reutilização (libraries)
- Tudo em um único lugar (Jenkinsfile)

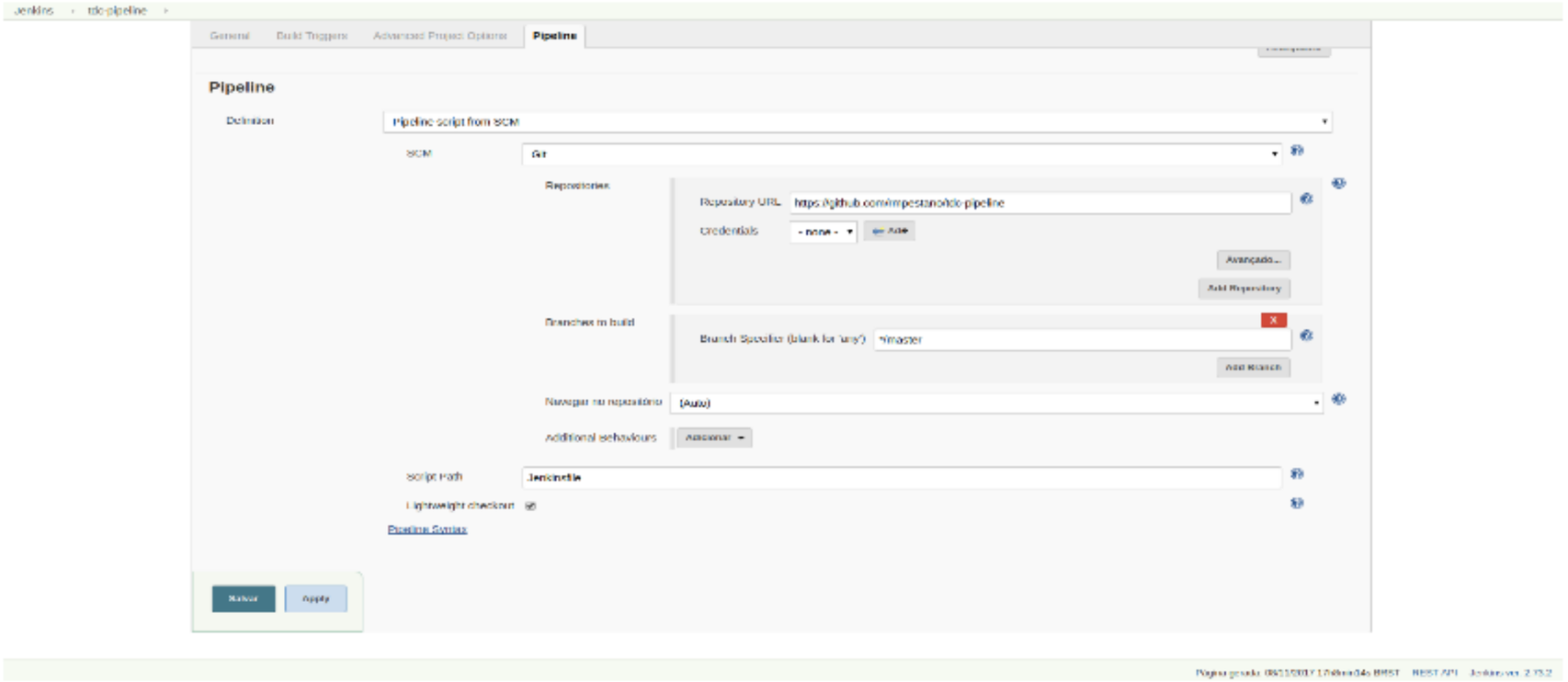
JENKINS 2.X PIPELINE

- Versionamento
- Reutilização (libraries)
- Tudo em um único lugar (Jenkinsfile)
- Sobrevive a restarts

JENKINS 2.X PIPELINE COMO CÓDIGO

```
pipeline {
    agent any

    stages {

        stage('checkout') {

            steps {
                git 'https://github.com/rmpestano/tdc-pipeline.git'
            }
        }

        stage('build') {

            steps {
                sh 'mvn clean package'
            }
```

JENKINS 2.X PIPELINE NO CÓDIGO



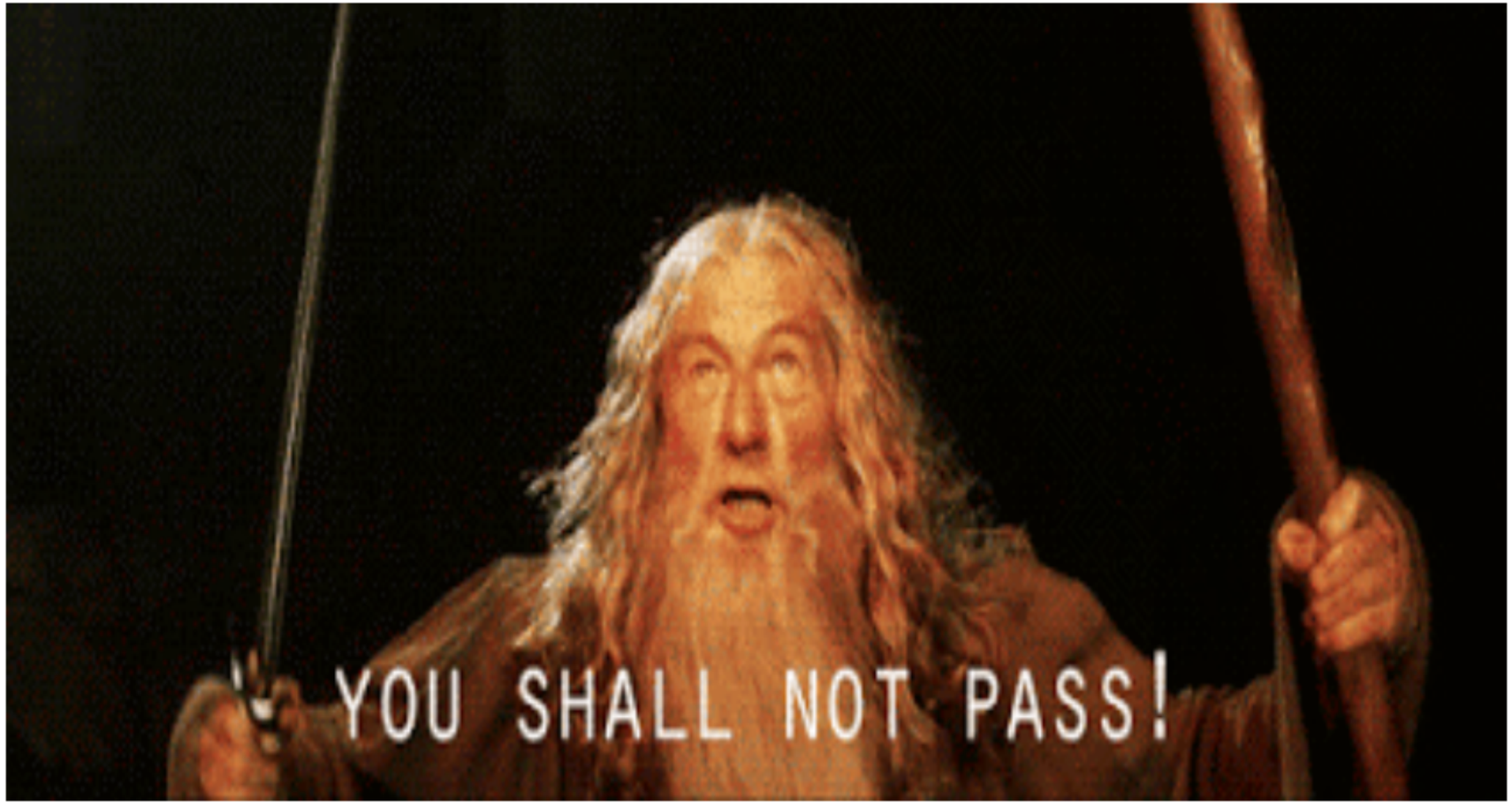
Demo **`v0.1`** (http://github.com/rmpestano/tdc-pipeline)

SONAR

Demo `v0.2`

QUALITY GATE

Demo `v0.3`



POST ACTIONS

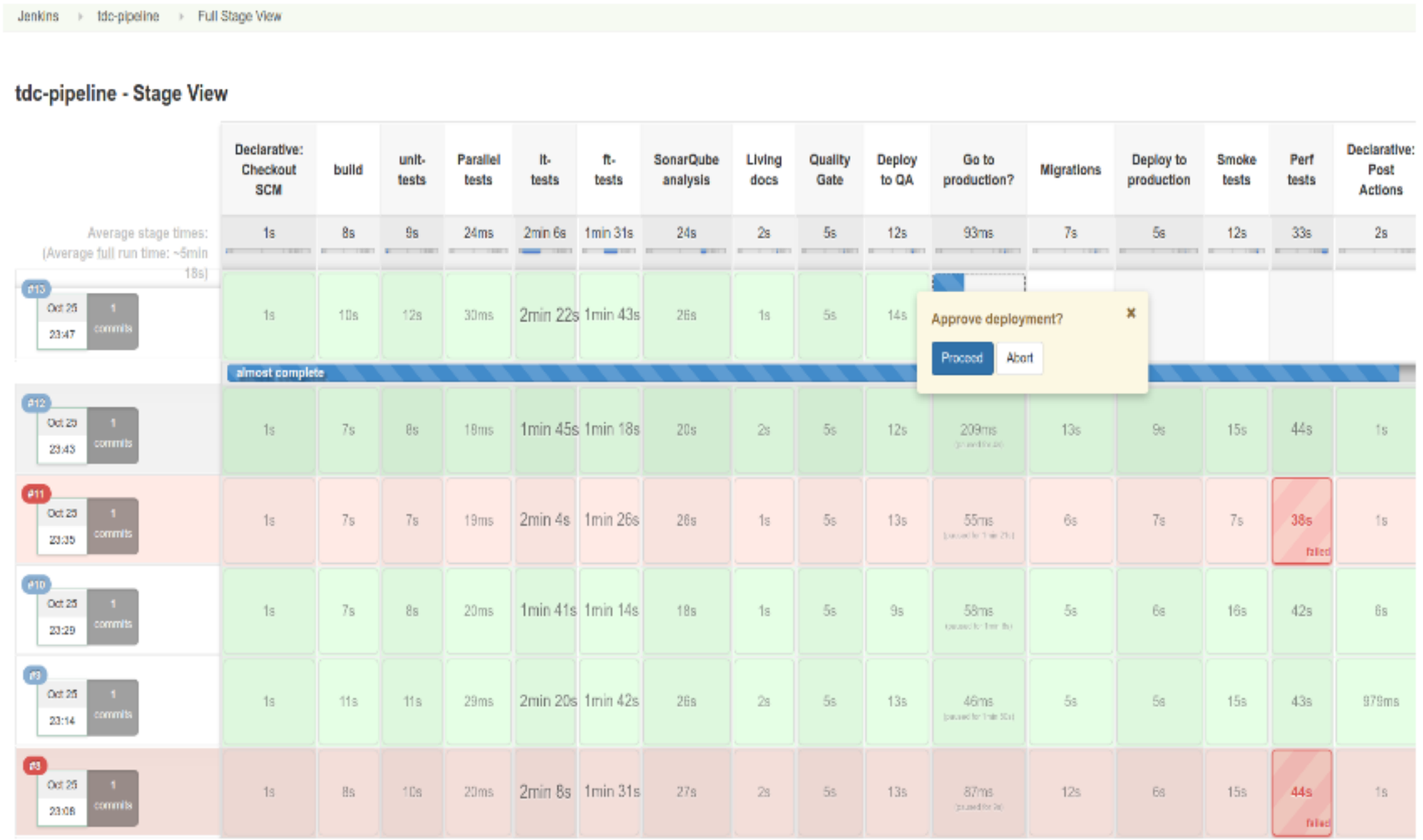
Demo **v0.4**

```
pipeline {
    agent any

    //stages

    post {
      always {
            sendNotification(currentBuild.result)
      }

      success {
           echo 'Build was a success'
      }

      failure {
           echo 'Build failure'
```

PIPELINE LIBRARIES

Permite o reaproveitamento de trechos de um pipeline

TDC PIPELINE FINAL



VIDEO

PERGUNTAS?

REFERÊNCIAS

- https://github.com/rmpestano/tdc-pipeline/
- https://jenkins.io/doc/book/pipeline/syntax/
- https://jenkins.io/blog/2017/02/15/declarative-notifications/
- https://jenkins.io/doc/book/pipeline/shared-libraries/
- https://jenkins.io/blog/2017/02/07/declarative-maven-project/
- https://virtualjug.com/pipeline-as-code-building-continuous-delivery-pipelines-with-jenkins-2/

Slides: https://rmpestano.github.io/talks/slides/javaee-pipeline/

VALEU GALERA!