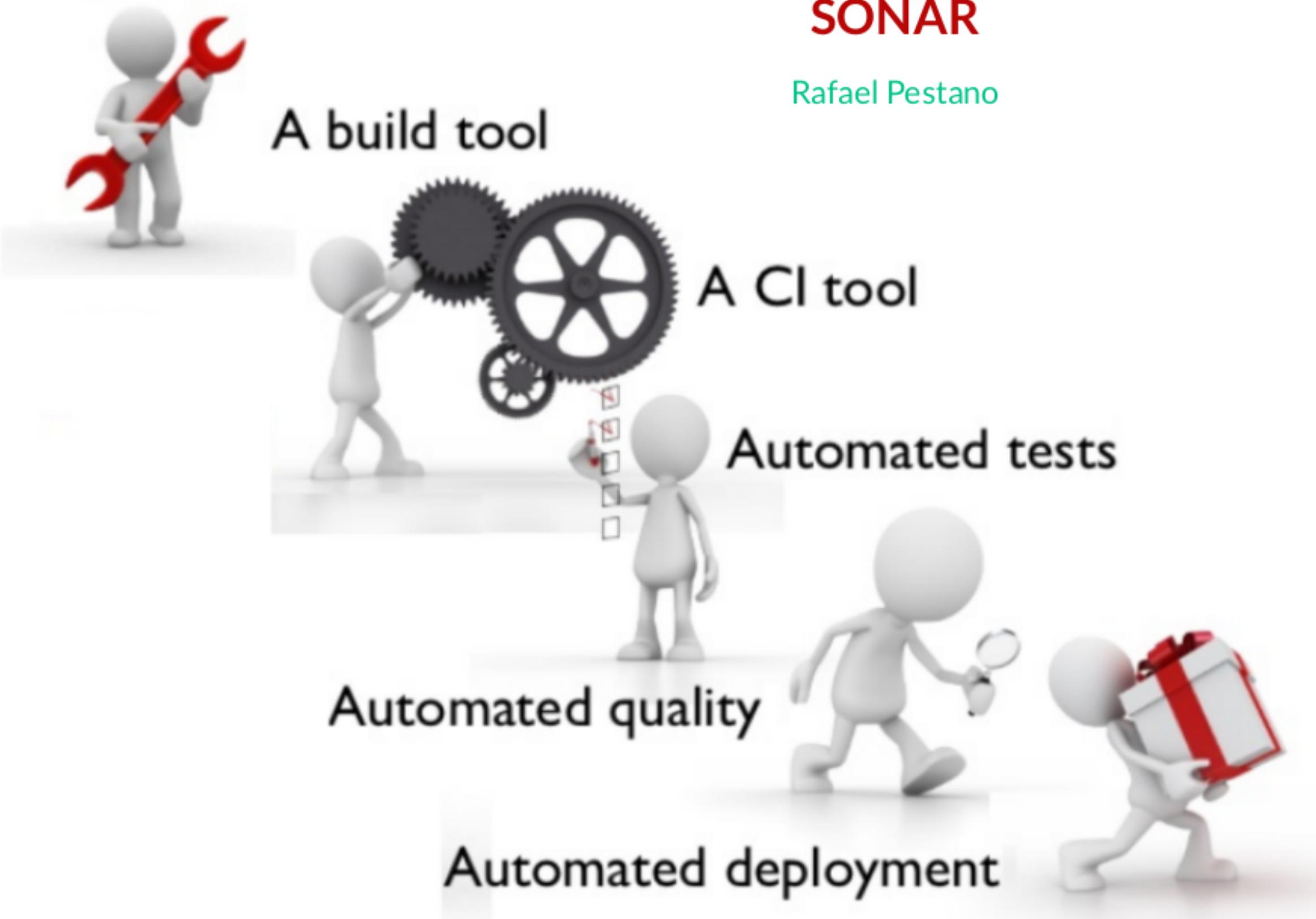


# JAVAEE PIPELINE AS CODE USING JENKINS, DOCKER AND SONAR

Rafael Pestano



# RAFAEL PESTANO

Software Engineer at PROCERGS

- [@realpestano](#) 
- [@rmpestano](#) 
- [Blog](#) 



See this presentation in [HTML here](#).

# 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"*

- Be ready to delivery in production at any moment!

# WHY?

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

— Jez Humble

# GOALS

- Reduce the risk of delivering
- Create a well known delivery process/cycle
- Make release process painless and without surprises
- Be ready to go to production at anytime



# DEPLOY TO PRODUCTION



# PRINCIPLES

# PRINCIPLES

- Each commit/push creates a release candidate

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

# PRINCIPLES

- Each commit/push creates a release candidate

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

- Heavily based on automation

# PRINCIPLES

- Each commit/push creates a release candidate

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

- Heavily based on automation
- Automated tests are primordial

# PRINCIPLES

- Each commit/push creates a release candidate

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

- Heavily based on automation
- Automated tests are primordial
- Continuous and fast feedback (from end user as well from your release process)

# PRINCIPLES

# PRINCIPLES

- Continuous improvement

# PRINCIPLES

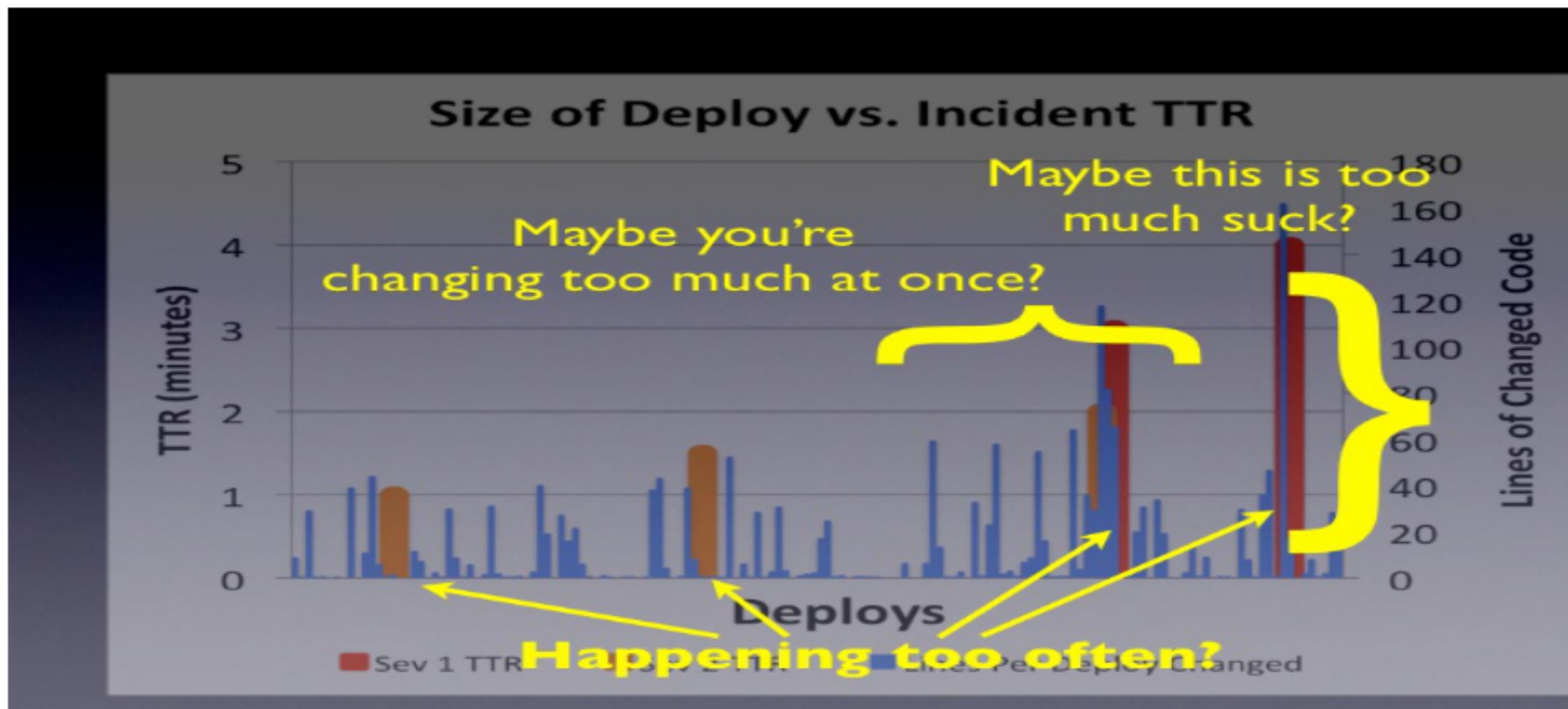
- Continuous improvement
- Collaboration, everyone is responsible for the release process ( DEV, QA, OPs...)

# PRINCIPLES

- Continuous improvement
- Collaboration, everyone is responsible for the release process ( DEV, QA, OPs...)
- Measurable progress
  1. How many builds have failed?
  2. In which stage it failed?
  3. How long it takes to go to production?

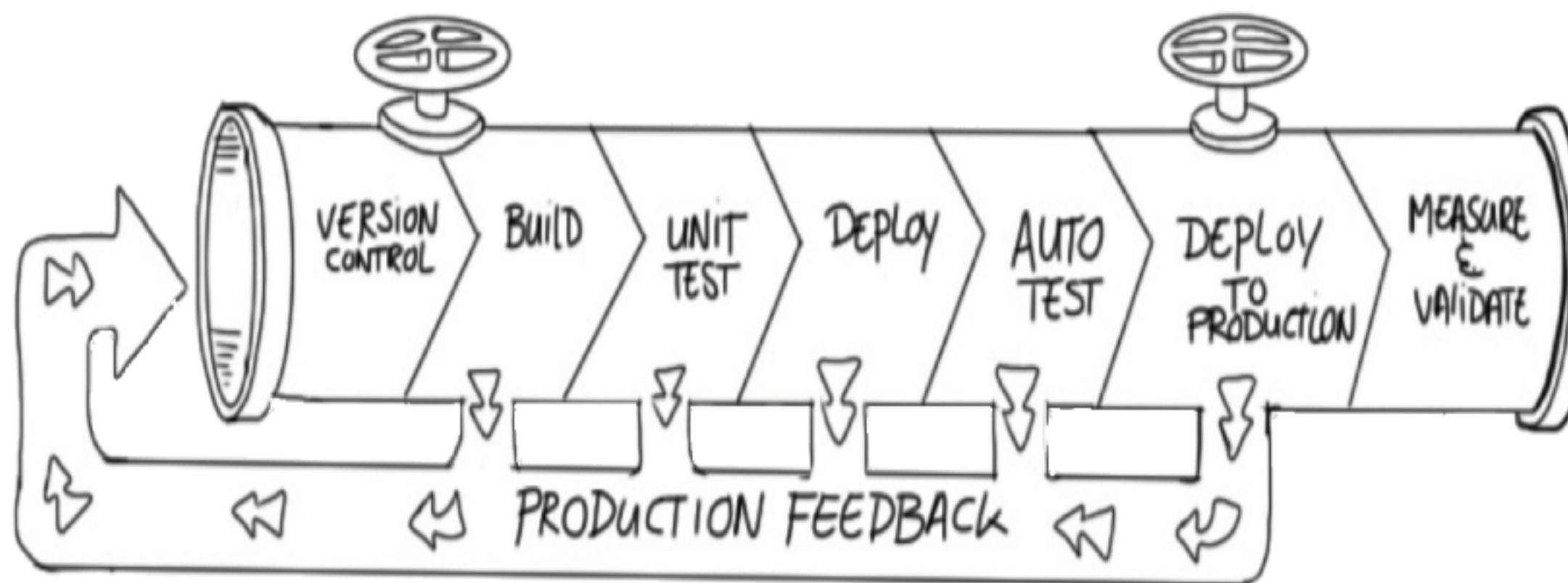
# CONSEQUENCES

- Less severity and frequency of failures related to a release
- Reduced time to recovery from failures (MTTR)



# DEPLOYMENT PIPELINE

*"...A pipeline is a set of stages to bring functionality 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](#)

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

- Described in a very easy and powerful DSL

# JENKINS 2.X PIPELINE

- Described in a very easy and powerful DSL
- Lives on source code (versioning)

# JENKINS 2.X PIPELINE

- Described in a very easy and powerful DSL
- Lives on source code (versioning)
- Reuse with **shared libraries**

# JENKINS 2.X PIPELINE

- Described in a very easy and powerful DSL
- Lives on source code (versioning)
- Reuse with **shared libraries**
- Everything in one place (Jenkinsfile)

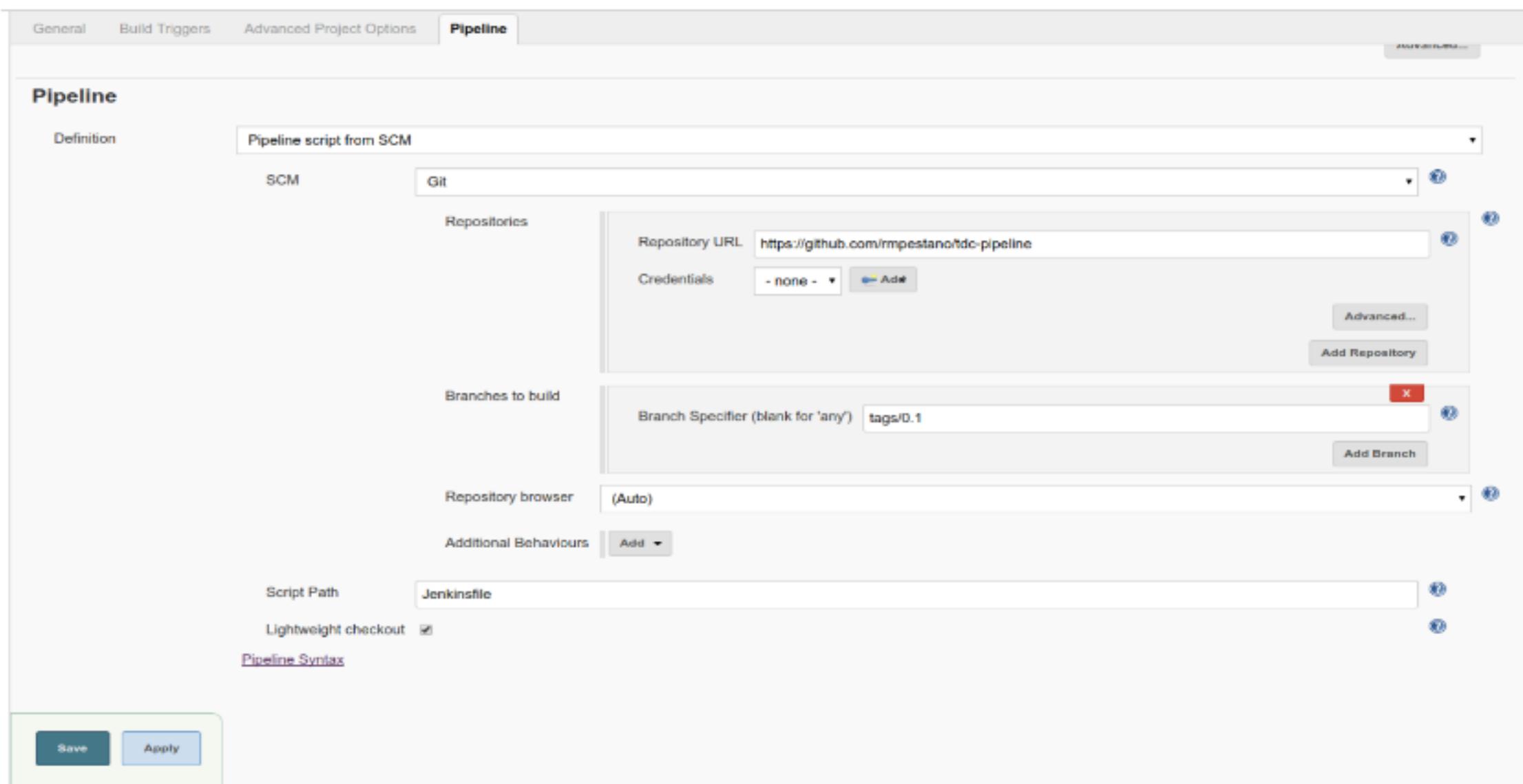
# JENKINS 2.X PIPELINE

- Described in a very easy and powerful DSL
- Lives on source code (versioning)
- Reuse with **shared libraries**
- Everything in one place (Jenkinsfile)
- Recovery from restarts

# JENKINS 2.X PIPELINE AS CODE

```
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 ON CODE



Demo v0.1 (<https://github.com/rmpestano/tdc-pipeline/releases/tag/0.1>)

# JENKINS 2.X PIPELINE ON CODE

```
pipeline {  
    agent any  
  
    stages {  
  
        stage('build') {  
  
            steps {  
                sh 'mvn clean package'  
            }  
        }  
  
        stage('Deploy') {  
            steps {  
                sh 'docker stop tdc-pipeline || true && docker rm tdc-pi  
                sh 'docker build -t tdc-pipeline .'  
            }  
        }  
    }  
}
```

# SONAR

Demo v0.2 (<https://github.com/rmpestano/tdc-pipeline/releases/tag/0.2>)

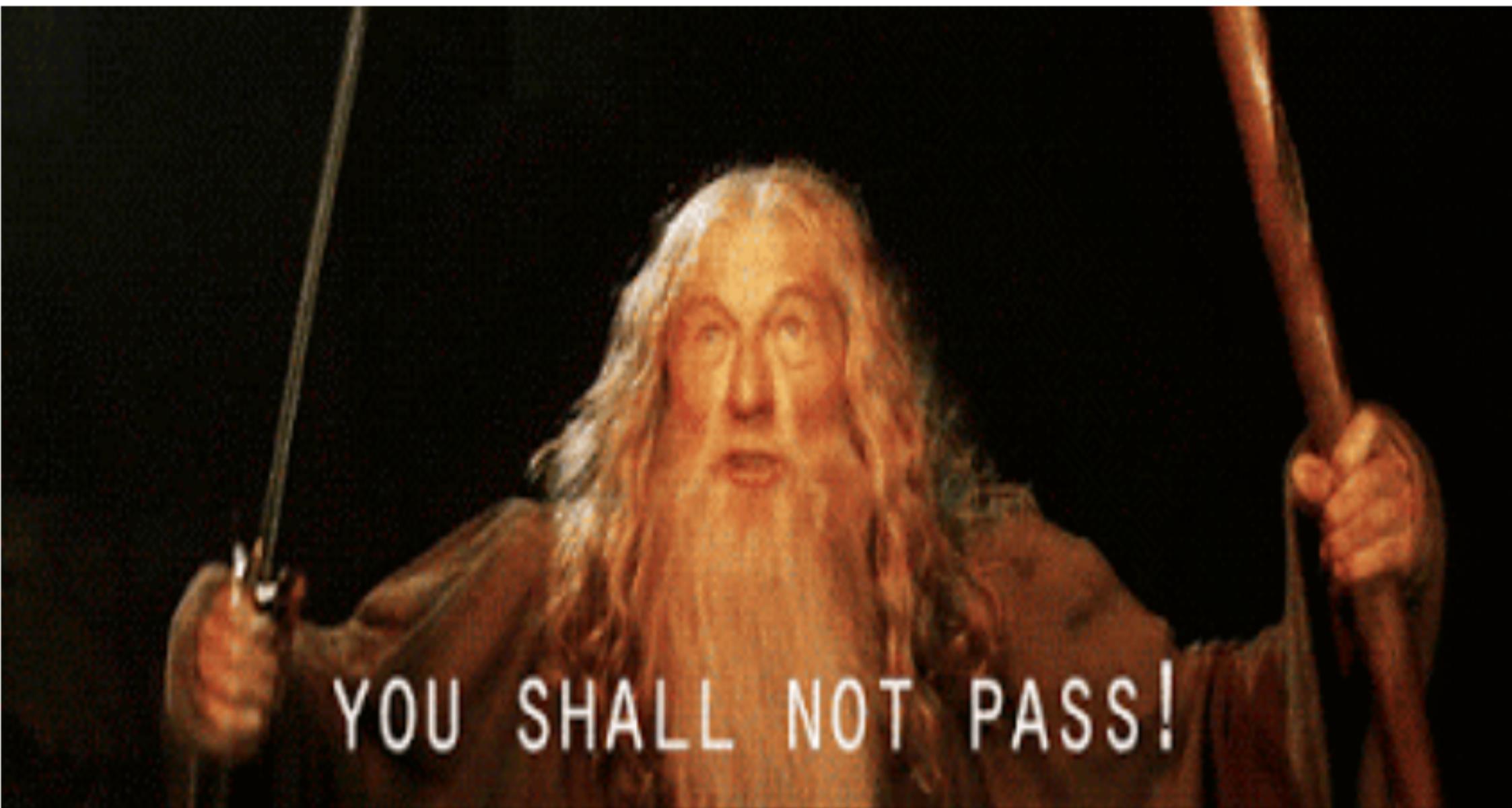


# SONAR

```
pipeline {  
    agent any  
  
    stages {  
  
        stage('build') {  
  
            steps {  
                sh 'mvn clean package -DskipTests'  
            }  
        }  
  
        stage('unit-tests') {  
            steps {  
                sh 'mvn test -Pcoverage'  
            }  
        }  
    }  
}
```

# QUALITY GATE

Demo v0.3 (<https://github.com/rmpestano/tdc-pipeline/releases/tag/0.3>)



# QUALITY GATE

```
pipeline {  
    agent any  
  
    stages {  
  
        stage('build') {  
  
            steps {  
                sh 'mvn clean package -DskipTests'  
            }  
        }  
  
        stage('unit-tests') {  
            steps {  
                sh 'mvn test -Pcoverage'  
            }  
        }  
    }  
}
```

# POST ACTIONS

Demo v0.4 (<https://github.com/rmpestano/tdc-pipeline/releases/tag/0.4>)

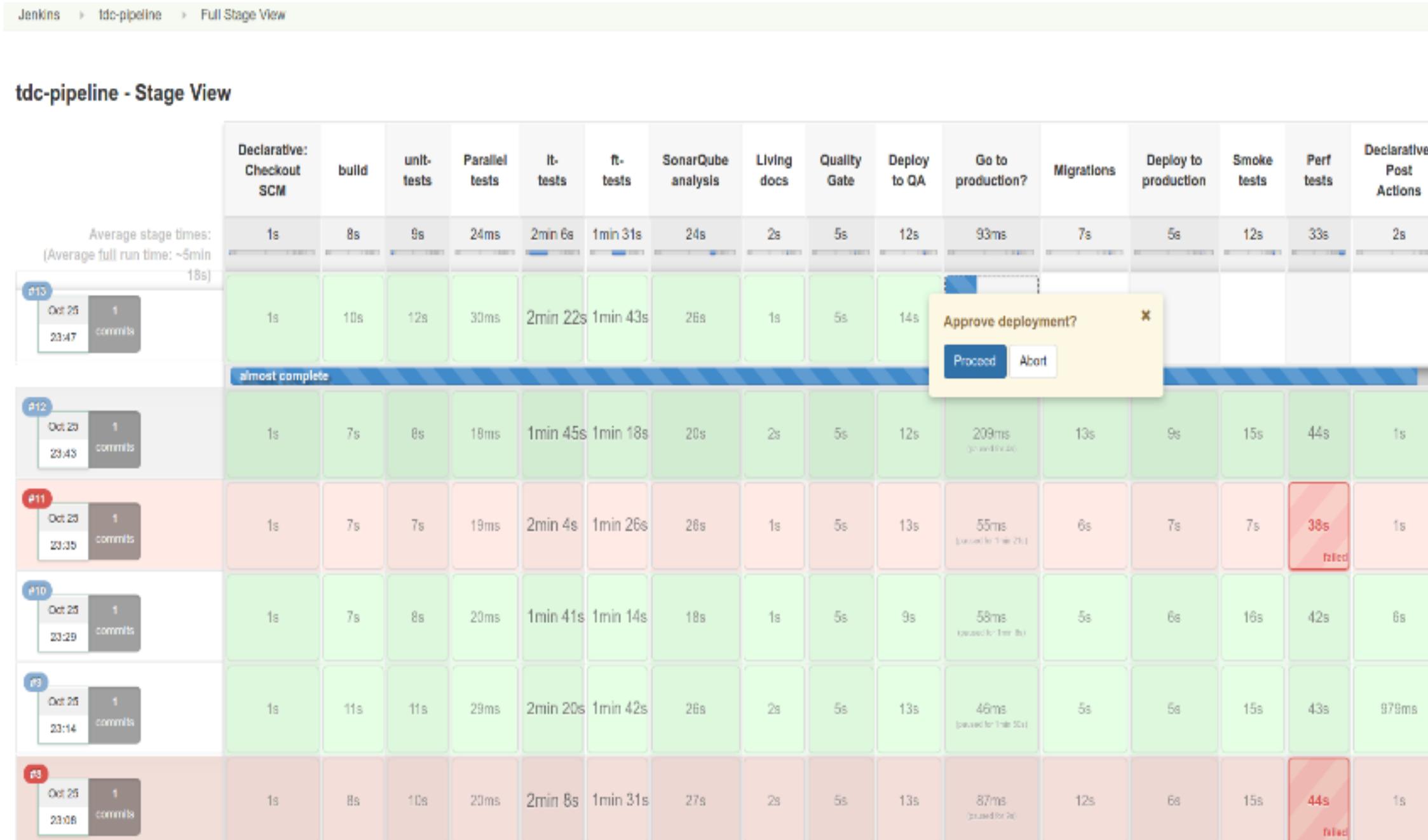
```
pipeline {  
    agent any  
  
    //stages  
  
    post {  
        always {  
            sendNotification(currentBuild.result)  
        }  
  
        success {  
            echo 'Build was a success'  
        }  
  
        failure {  
            echo 'Build failure'  
        }  
    }  
}
```

# PIPELINE SHARED LIBRARIES

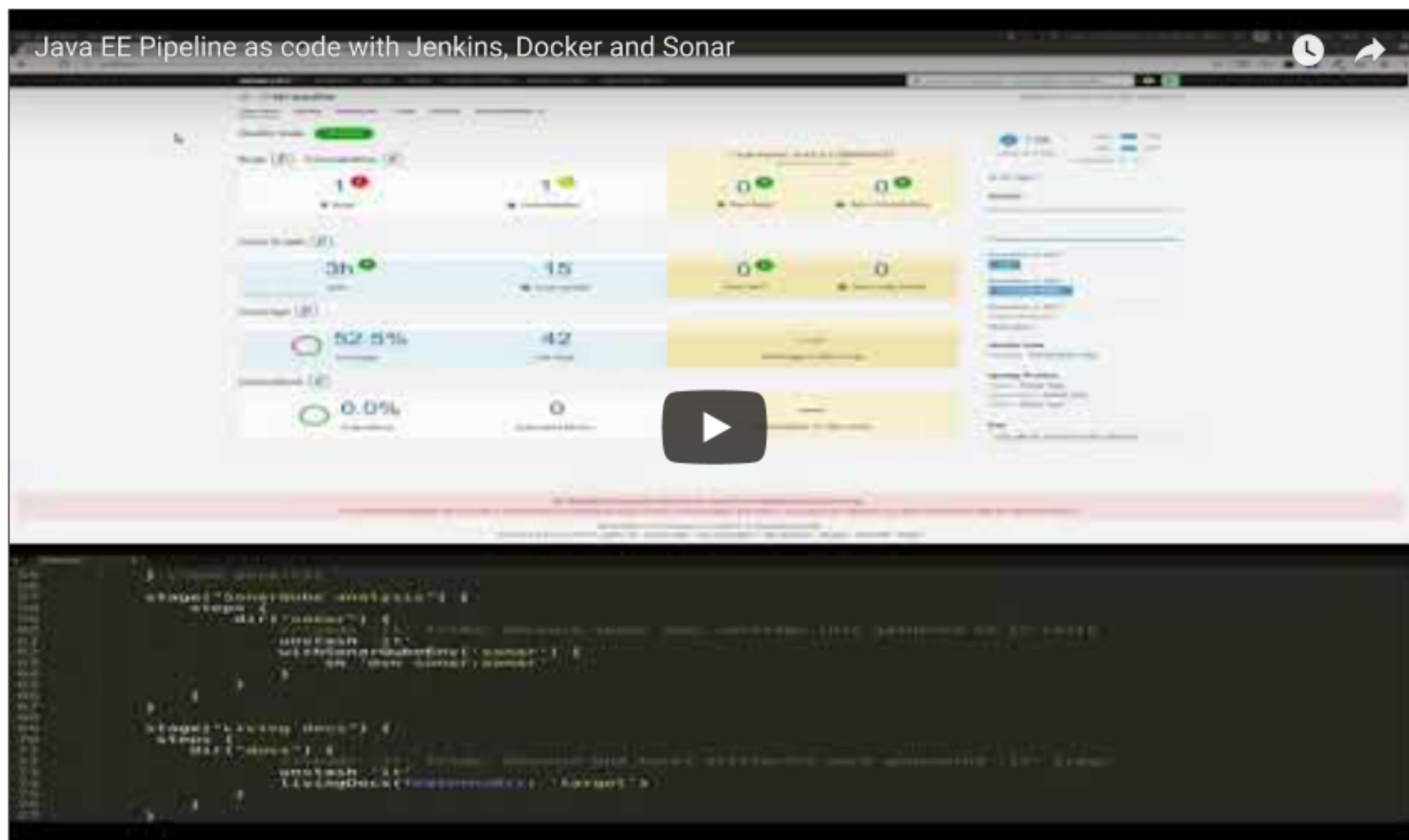
Enables reuse of pipeline sections (even entire stages) among projects

<https://github.com/rmpestano/tdc-pipeline#shared-library>

# TDC PIPELINE FINAL



# VIDEO



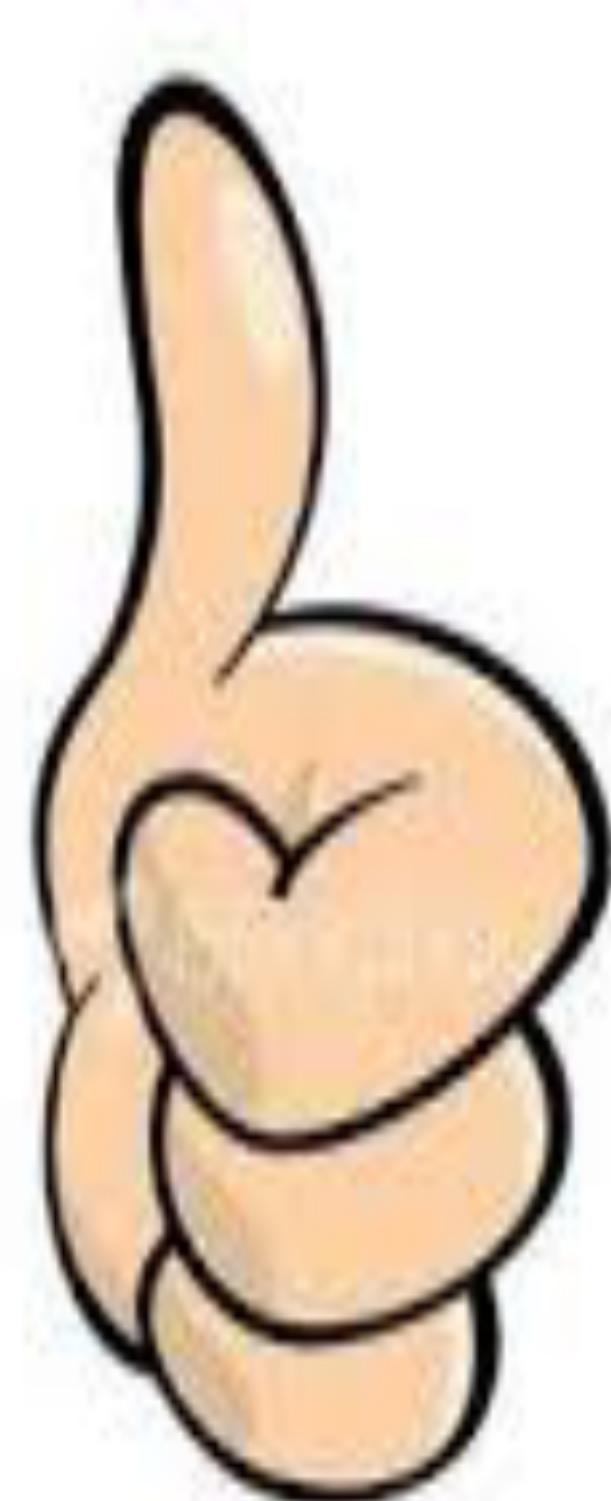
# PERGUNTAS?



# REFERENCES

- <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/index-en.html>



**VALEU  
GALERA!**