



**TRIBUNAL REGIONAL ELEITORAL DO RN**  
**SECRETARIA DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO**

# **MANUAL DO PROCESSO** **IMPLANTAÇÃO DE SISTEMAS INTERNOS**

## APRESENTAÇÃO

Este processo demonstra os passos utilizados para implantação em ambiente de homologação e produção de sistemas administrativos desenvolvidos pela Seção de Desenvolvimento de Sistemas do Tribunal Regional Eleitoral do Rio Grande do Norte.

Todo o processo de implantação após o desenvolvimento de qualquer funcionalidade ou correção de bug é inteiramente automatizado com o uso das ferramentas de automação *Gitlab*, *Jenkins*, *Nexus*, *Nginx*, *Redis* e *Ansible*.

## CONTROLE DE VERSÃO

QUADRO RESUMO	
Versão:	1.0
Elaboração:	Coordenadoria de Sistemas/STIC
Aprovação:	Comitê Executivo de TIC
Referências legais e boas práticas:	Scrum, DevOps

QUADRO DE REVISÕES			
DATA	VERSÃO	ALTERAÇÃO	RESPONSÁVEL
01/08/2019	1.0	Versão inicial.	George Melo de Freitas Barbalho

## ÍNDICE

<b>1. VISÃO GERAL DO PROCESSO</b>	<b>4</b>
<b>2. ATIVIDADES</b>	<b>4</b>
2.1. Checkout	4
2.2. Update Jenkins, Devlocal	4
2.3. Build	5
2.4. Test	5
2.5. Package	5
2.6. Deploy	6
<b>3. EXECUTORES</b>	<b>6</b>
3.1. Git	6
3.2. Jenkins	6

## 1. VISÃO GERAL DO PROCESSO

O processo de **Implantação de Sistemas Internos** é composto por seis atividades e dois papéis distintos, iniciando a partir de um sinal enviado pelo *Gitlab* (iniciado a partir de um *commit* de um desenvolvedor na *branch master* de qualquer repositório) e concluindo, em seu caminho feliz, com disponibilização de uma nova versão de sistema nas máquinas servidoras. A representação do processo em *Business Process Model Notation* (BPMN) é apresentada a seguir.

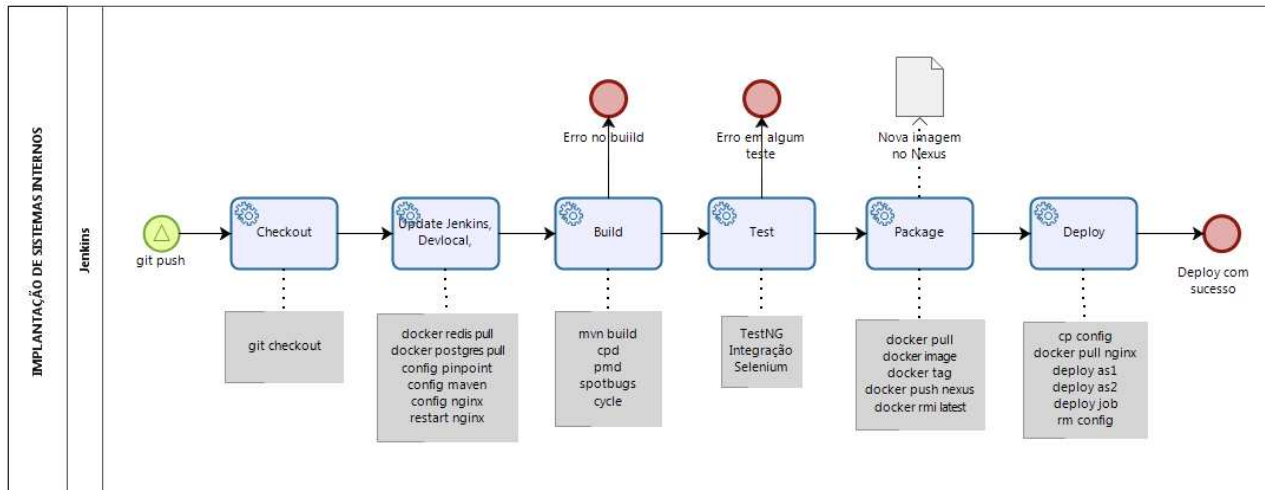


Figura 1. Processo Completo - Implantação de Sistemas Internos

## 2. ATIVIDADES

As atividades que são executadas ao longo da realização do processo são apresentadas a seguir.

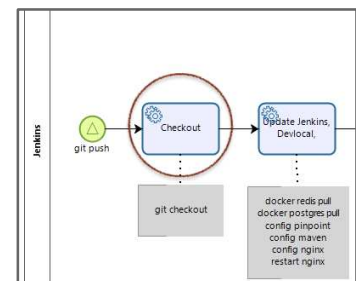
### 2.1. Checkout

#### DESCRIÇÃO:

Esta tarefa é executada após o *Gitlab* notificar o *Jenkins* de que houve alterações de código na *branch master* do repositório. Basicamente nesta atividade o *Jenkins*, via *Ansible*, se conecta ao repositório no *Gitlab* e baixa o código-fonte que está na *branch master*.

#### EXECUTOR:

*Jenkins*



### 2.2. Update Jenkins, Devlocal

#### DESCRIÇÃO:

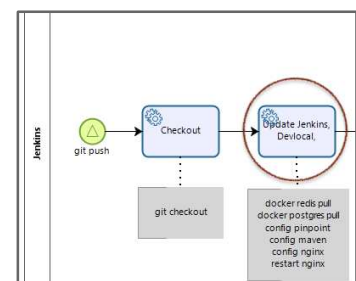
Esta tarefa executa a atualização dos *containers* do *Redis* e do *PostgreSQL* que estão executando na máquina do *Jenkins*. Além destas atualizações, o *script* configura o *pinpoint-agent* e o *Maven* nesta máquina. Após as configurações, o *script* configura e reinicia o *container* que mantém o serviço do *Nginx*.

#### EXECUTOR:

*Jenkins*

#### ARTEFATO:

*Redis* e *Postgres* atualizados.  
*Pinpoint* e *Maven* configurados.  
*Nginx* configurado e reiniciado.

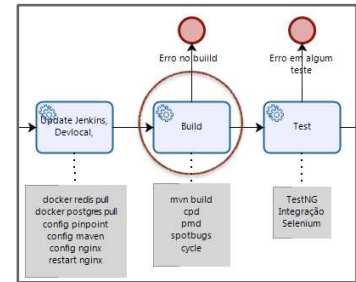


## 2.3. Build

### DESCRIÇÃO:

Nesta etapa o *Jenkins* aciona o *Maven* para executar a *build* do código-fonte que foi enviado para a *branch master*. O *Maven* executa algumas verificações de código-fonte, com os seguintes plugins:

- *Copy/Paste Detector* (CPD): procurar por código duplicado;
- *PMD*: analisador de código-fonte que procura variáveis não utilizadas, blocos de tratamento de exceções vazios, criações de objetos desnecessárias, etc;
- *SpotBugs*: procura por *bugs* em programas *Java*, baseado em conceitos de padrões de *bug*;
- *Cycle*: procura por referências cíclicas entre pacotes em programas *Java*.



### EXECUTOR:

*Jenkins*

### ARTEFATO:

Código-fonte testado para erros de programação.

## 2.4. Test

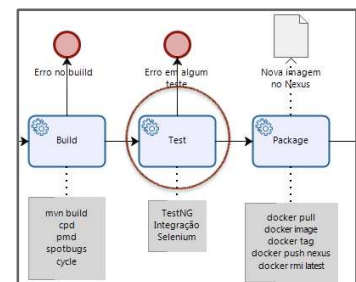
### DESCRIÇÃO:

Nesta tarefa são executados testes unitários, testes de integração e testes de aceitação.

Os testes unitários utilizam a API *TestNG* para testar blocos de código que contém as funcionalidades mais usadas no sistema.

Os testes de integração reproduzem cenários que serão executados pelos usuários quando o sistema estiver em pleno funcionamento. Para execução destes testes, é utilizado um container com uma instância do *PostgreSQL* rodando na máquina do *Jenkins*.

Os testes de aceitação utilizam *Selenium* para testar os casos de uso a nível de interface do usuário.



### EXECUTOR:

*Jenkins*

### ARTEFATO:

Código-fonte testado.

## 2.5. Package

### DESCRIÇÃO:

Nesta tarefa é gerada uma imagem contendo o sistema que será implantada tanto em ambiente de homologação quanto de produção. A imagem contém o sistema operacional *Alpine*, as *libs* utilizadas pelo sistema e a aplicação *SpringBoot*.

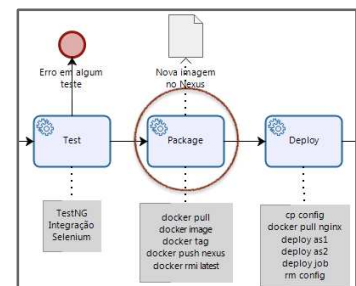
Após a geração da imagem, é criada uma nova *tag latest* e enviada para o *Nexus*.

### EXECUTOR:

*Jenkins*

### ARTEFATO:

Imagem do sistema no *Nexus*.

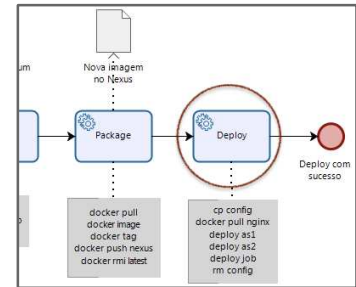


## 2.6. Deploy

### DESCRIÇÃO:

Nesta tarefa a imagem latest do sistema armazenada no *Nexus* é implantada nas máquinas de homologação.

Os arquivos de configuração do sistema são utilizados para que a aplicação *SpringBoot* possa ser iniciada através da execução do respectivo container nas máquinas apropriadas. Após a execução, os arquivos de configuração necessários para o *start* do sistema são removidos.



### EXECUTOR:

*Jenkins*

### ARTEFATO:

*Container* com o sistema executando em ambiente de homologação.

## 3. EXECUTORES

Os executores das rotinas automatizadas são os seguintes:

### 3.1. Git

Responsável por notificar o *Jenkins* de que novo código-fonte foi atualizado na *branch* master do respectivo repositório.

### 3.2. Jenkins

Responsável por toda a execução das tarefas que preparam e entregam o sistema executando nas máquinas em ambiente de homologação e produção.