

MINISTÉRIO DE MINAS E ENERGIA
Secretaria Executiva
Subsecretaria de Tecnologia e Inovação
Coordenação Geral de Inovação e Transformação Digital

Processo de Integração Continua

Versão 1.2

Histórico de Revisões

Data	Versão do documento	Descrição	Autor
31/10/2022	1.0	Atualizado com a definição da nova arquitetura incluindo a parte de QA.	Marcos Martins - Tecnisys
10/04/2024	1.1	Ajustes e atualização do documento	Luiz Alexandre
15/04/2024	1.2	Revisão	Carlos Gomes

Sumário

1. **PRÉ-REQUISITOS 3**
2. **FERRAMENTAS 3**
3. **FLUXO BÁSICO DE ATIVIDADES 3**
4. **AMBIENTES 6**
   Desenvolvimento (Branch) 6
   Qualidade – QA (Branch) 6
   Homologação (Branch) 6
   Produção (Master) 7
5. **VANTAGENS 7**

1. PRÉ-REQUISITOS

1. Toda demanda deverá ser marcada por um identificador único, isso irá definir os “pacotes” de entregas que serão usados para criação das branchs que serão liberadas para produção.
2. Máquina dedicada para funcionar como o servidor de implantação (Jenkins).
3. Máquina dedicada para instalação do repositório de bibliotecas (Nexus).

2. FERRAMENTAS

- GIT – Sistema de controle de versões
- Maven – Ferramenta de automação de compilação
- Jenkins – Ferramenta de integração contínua utilizada para automatizar os processos de build e deploy da aplicação.
- Sonar – Dashboard de qualidade.
- Nexus – Repositório de bibliotecas.

3. FLUXO BÁSICO DE ATIVIDADES

A sugestão é que o ambiente dos projetos seja dividido em 4 nós:

- > Desenvolvimento (branch)

- > QA (branch)

- > Homologação (branch)

- > Produção (master).

Será usado o conceito de *Future Branches* onde cada OS terá sua respectiva branch (criada a partir da Master) para desenvolvimento separada dos desenvolvedores sem a intervenção de outras funcionalidades do restante da equipe. No *Future Branch,* a master é utilizada como a versão mais completa da aplicação, ou seja, **Produção**, contendo todas as implementações já realizadas em branches separadas. A medida que uma nova OS é recebida (tanto corretiva, como evolutiva) é criada uma Branch que representará essa OS. Ao término do desenvolvimento, essa Branch criada será enviada para os respectivos ambientes (desenvolvimento, qualidade, homologação ou

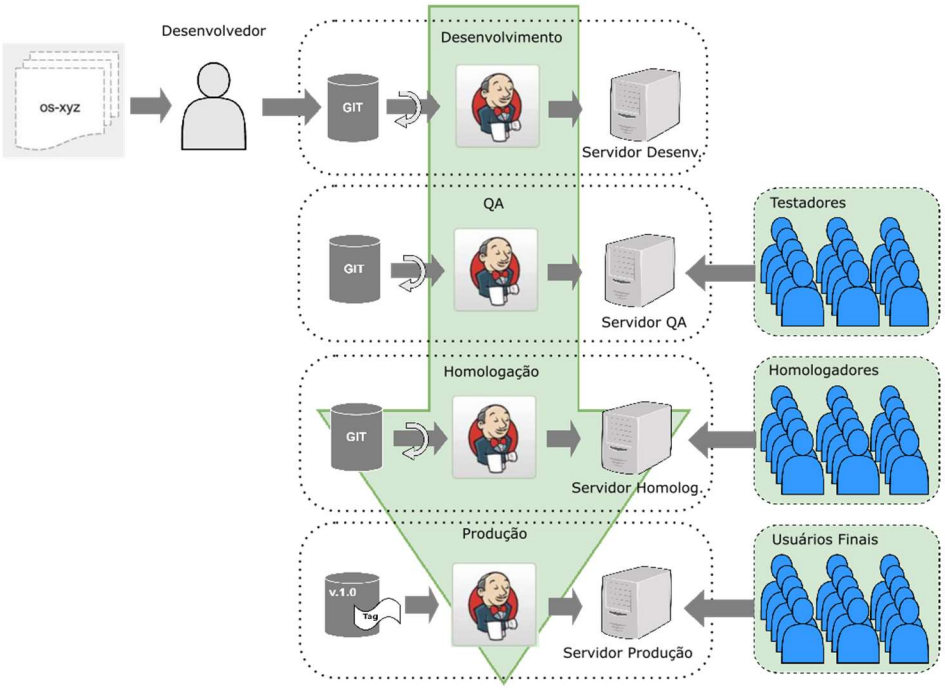
produção), ao qual, deverá ter a respectiva aprovação.

O uso dos branches se dá para facilitar a divisão de tarefas e organização das alterações de modo que solicitações descartadas sejam facilmente ignoradas e não reintegradas ao Master. Conseguimos também escolher quais funcionalidades que deverão estar em um release de produção. Podendo ter várias Branchs em desenvolvimento não impactando uma na outra.

A medida em que os Branchs são aprovados em cada ambiente, deverá ser feito o merge para o próximo ambiente.

Por exemplo:

1. É criada uma nova OS.
2. Cria-se uma branch para essa OS – Nela será desenvolvida toda a funcionalidade.
3. Quando terminado o desenvolvimento, é feito o merge (executado pela Fábrica de software) da branch da OS para o ambiente de desenvolvimento e o processo da integração continua se inicia:
   a. O Jenkins reconhece uma nova versão da branch de desenvolvimento;
   b. É realizada a leitura do código com o Sonar, em caso de falha, um e-mail é retornado para o desenvolvedor realizar as devidas correções. Em caso de sucesso, o processo continua;
   c. O Jenkins verifica as versões disponíveis no Nexus e faz o download das dependências. Ao término, finaliza o empacotamento do WAR e faz upload desse WAR no servidor de desenvolvimento.
4. A equipe de analistas responsáveis pelo levantamento de requisitos, aprova a nova funcionalidade no ambiente de desenvolvimento.
5. Realiza o merge (executado pela equipe de QA) para o ambiente de qualidade – QA:
   a. O processo de integração contínua nesse ambiente é feito de forma síncrona pelo integrador da equipe;
   b. O Jenkins verifica as versões disponíveis no Nexus e faz o download das dependências. Ao término, finaliza o empacotamento do WAR e faz upload desse WAR no servidor de qualidade - QA.
6. Realiza o merge (executado pela equipe de QA após aprovação do Gestor do projeto) para o ambiente de homologação:
   a. O processo de integração contínua, nesse ambiente é feito de forma síncrona pelo integrador da equipe;
   b. O Jenkins verifica as versões disponíveis no Nexus e faz o download das dependências. Ao término, finaliza o empacotamento do WAR e faz upload desse WAR no servidor de homologação.
7. O Requisitante aprova a funcionalidade no ambiente de homologação.
8. É feito o merge (executado pela equipe da Infra após aprovação do Gestor do projeto) na branch Master É criada uma nova release para produção com as funcionalidades aprovadas.



4. AMBIENTES

DESENVOLVIMENTO (BRANCH)

É o ambiente de liberação do código do desenvolvedor para testes iniciais de funcionalidade e de similaridade de ambiente.

O build será feito de forma assíncrona, assim a cada liberação de código do desenvolvedor, o servidor de integração automaticamente verifica se há alterações no repositório de versão (identificada como uma tag do GIT com o sufixo "–dev") e inicia o processo de build consequentemente pelo deploy.

Esse será um ambiente em contínua atualização, por isso os analistas devem evitar fazer testes. Ao término do build o desenvolvedor faz o teste de ambiente, validação do código no Sonar e analisa se está conforme o desenvolvido em sua máquina.

Se aprovado, o desenvolvedor libera o código para a qualidade, para que seja feito os próximos testes antes de entrar em produção.

QUALIDADE - QA (BRANCH)

Será um ambiente de testes. O build será feito de forma sincrona (manual). A cada build será gerada uma release version (identificada como uma tag do GIT com o sufixo "–alpha") vinculada ao identificador único da OS, que será usada para controle de aprovação. Cada build executa os testes automatizados e em

caso de falha, aciona a Fábrica de Software para correção. Será usada também a ferramenta Sonar acompanhado pelo arquiteto, para garantir a qualidade de código, e em caso de desaprovação será retornado para a Fábrica de Software corrigir e colocar no padrão antes mesmo de entrar em homologação. Caso a release seja aprovada, esse mesmo código usado em qualidade - QA é então enviado para homologação.
A diferença entre esses ambientes será o banco de dados, que no caso poderá ser uma réplica de homologação.

HOMOLOGAÇÃO (BRANCH)

Será um ambiente de testes de aceitação e aprovação de demanda. O build será feito de forma síncrona (manual). A cada build será gerada uma release version (identificada como uma tag do GIT com o sufixo "–rc") vinculada ao identificador único da OS, que será usada para controle de aprovação. Em caso de falha, aciona a Fábrica de Software para correção. Caso a release seja aprovada, esse mesmo código usado em homologação é então enviado para produção.
A diferença entre esses ambientes será o banco de dados, que no caso poderá ser uma réplica de produção, mas com os dados embaralhados.

PRODUÇÃO (MASTER)

Esse ambiente terá disponível apenas as funcionalidades que estão em versões estáveis, aprovados no ambiente de homologação anteriormente.
O build será feito de forma sincrona (manual) ou de forma agendada, em intervalos de tempo pré-definidos e em horários de menor uso. A cada build será gerada uma release version (identificada como uma tag do GIT com o sufixo "–stable") vinculada ao identificador único da OS, que será usada para controle de aprovação.
Apenas alguns usuários com permissão especial poderão executar as tarefas de build e deploy no Jenkins.

5. VANTAGENS

- O *build* é executado em um servidor dedicado, levando menos tempo e liberando desenvolvedores dessa responsabilidade.
- Os artefatos são disponibilizados automaticamente nos ambientes.
- Os testes são feitos nos dois principais ambientes, tendo duas barreiras contra erros antes de chegar a produção.
- Os builds poderão ser acompanhados e agendados em horários sem a necessidade de intervenção humana.
- Diminui as chances de conflitos. Apenas as alterações que foram commitadas serão enviadas para o build, evitando que os testes e configurações específicas de cada programador sejam enviadas para o cliente.

- Qualidade. Com as ferramentas integradas e automatizadas, um padrão será seguido diminuindo a heterogeneidade dos sistemas.