Homologacao de Artefatos GIT

Tarefa: TPUP-65 - O projeto Jira não existe ou você não tem permissão para visualizá-lo.

Nesta pagina

Informação passo-a-passo sobre o processo de homologação de artefatos existentes dentro do repositório versionado.

Usuarios DVCS

O procedimento mostrado aqui eh baseado no uso do software GIT, um gerenciador de controle de versão com repositórios distribuídos. Caso você utilize um repositório centralizado (DVCS) como SVN, TFS ou outro, talvez este processo nao seja o mais adequado para você.

Veja tambem

Versionamento de Artefatos para informacoes sobre o controle de numeracao de artefatos versionados.

1. Estrutura de Diretórios

O controle de versões de sistemas e seus respectivos códigos-fonte é realizado pelo software GIT, seguindo a estrutura braches model onde temos branches prefixados para cada etapa do desenvolvimento de um software, conforme:

master/: Contem, o código que esta atualmente em produção, sempre irá receber o código dos branches de releases quando forem homologados, irá originar branchs de hotfix para correção de Bugs de Produção quando necessário. *Somente pessoas do grupo xx-qa tem permissão de escrever neste branch.*

development/: Branch de integração do desenvolvimento, a partir dele branches de features serão gerados e irá receber os códigos dos braches das features quando o desenvolvimento das mesmas forem entregues (via Pull Request). Será utilizado pelo analista para validação do desenvolvimento de cada feature entregue pelos desenvolvedores de forma integrada sendo a ultima etapa de validação antes do inicio da homologação. Artefatos gerados neste branch serão publicados no ambiente de desenvolvimento manualmente ou de forma automática. *Somente pessoas do grupo xx-qa tem permissão de escrever neste branch.*

feature/: Branch criado exclusivamente para o desenvolvimento de uma feature ou melhoria. Geralmente ele é criado a partir do branch de desenvolvimento (developement) e ao término do trabalho o código é retornado para o branch "developement" (via Pull Request) para ser validado. Este branch só será deletado quando a feature equivalente for para homologação em um determinado branch de release (via Pull Request), ação esta que pode ser executada pela interface de aceite do Pull Request do Stash ou via comando de exclusão de branch do GIT. *Somente pessoas do grupo xx-developers tem permissão de escrever neste branch.*

release/: Branch utilizado para homologação de uma determinada versão do software ou projeto. Normalmente este branch nasce a partir do master (produção) e recebe, via merge, código dos branches das features e melhorias a serem homologadas na versão correspondente. Durante a homologação para cada Bug encontrado um novo branch de prefixo Bug deverá ser gerado para o desenvolvedor aplicar a correção, após a correção ele deve solicitar a validação/homologação via Pull Request para o branch de release da versão. Após a versão do branch de release ser homologada o código fonte deve sofre uma TAG e um merge deve ser feito obrigatóriamente para os branchs: development e master e opcionalmente para os branchs de releases abertos se for necessário. *Somente pessoas do grupo xx-qa tem permissão de escrever neste branch.*

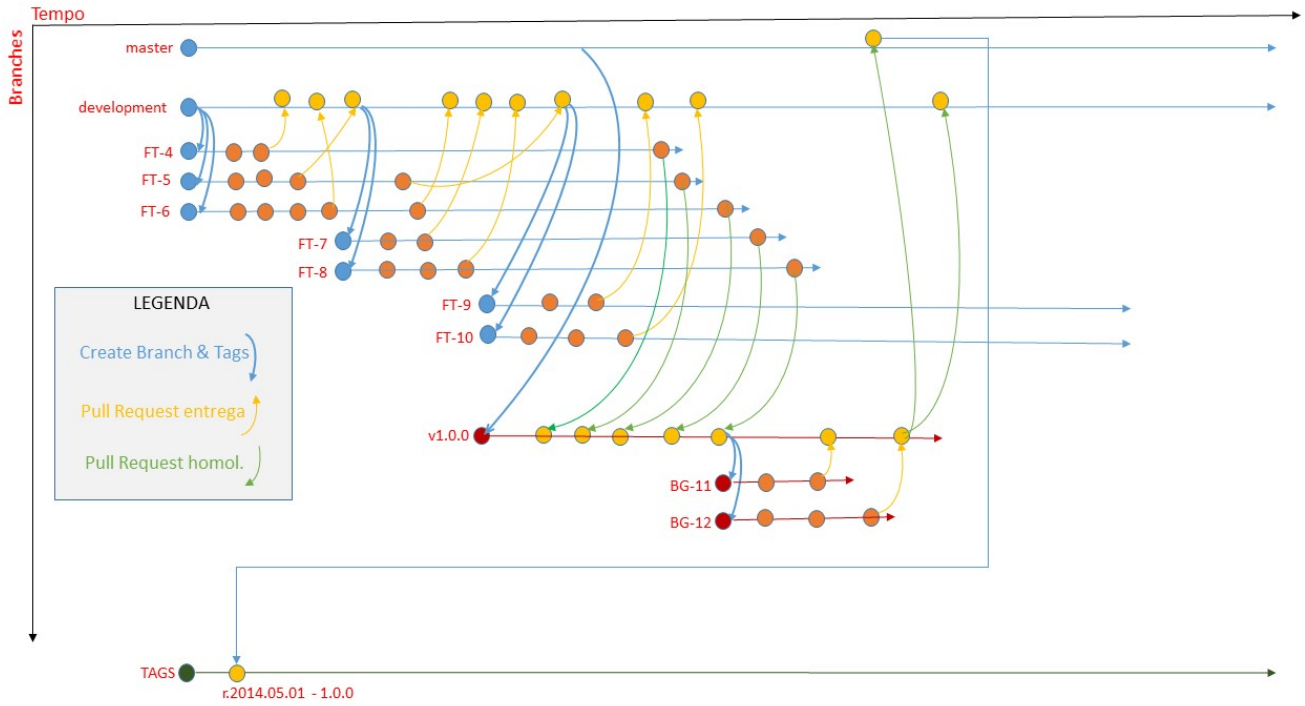
bugfix/: Branch utilizado para corrigir Bugs encontrados durante a homologação de uma versão dentro de branch de release. Este branch sempre será gerado a partir de um branch de release e após a correção ser desenvolvida o seu código deverá voltar para o branch de release (via Pull Request) para validação e assim sucessivamente até a homologação do Bug. *Somente pessoas do grupo xx-developers tem permissão de escrever neste branch.*

hotfix/: Branch utilizado para corrigir um Bug do código da versão que esta em produção (branch master). Após a correção o hotfix deve sofrer merge em um branch de release (versão) já existem ou em um novo branch de release gerado especificamente para homologação do hotfix a partir do branch master (produção). *Somente pessoas do grupo xx-developers tem permissão de escrever neste branch.*

tags/: Sao snapshots do projeto, e uma vez criados nao devem ser alterados. Somente podem criar tags usuarios do grupo da qualidade (QA). Por padrão, as tags seguem o formato de nome r.*aaaa.mm.dd - versao*, onde "r" indica o prefixo de release, "aaaa.mm.dd" representa a data de criacao da tag e "versao" um valor representando a versão logica do projeto frente ao gerenciamento do mesmo; por exemplo "tags/r.20014.05.01 - 1.0.0".
Somente pessoas do grupo xx-qa tem permissão de gerar TAGs

2. Fluxo de Promoção de Código-fonte de Desenvolvimento ate Produção

A figura abaixo mostra o fluxo de desenvolvimento com branches por tarefas para promoção de artefatos em linha de desenvolvimento, passando por homologação de releases ate entrada em produção e, eventual ajuste corretivo pós-produção.



2.1.1. Caso de Uso

Objetivo	Realizar todo o fluxo de criação, alteração e remoção de código-fonte ate entrada em produção, habilitando pontos de checagem(tags) para posterior volta de versão.

Ator principal	QA
Atores envolvidos	Desenvolvimento, QA, Administador
Resumo	Seguir o fluxo de operações: criação dos branches master e development, criação de branches para cada feature, solicitação de validação do desenvolvimento via Pull Request pelo desenvolvedor, criação de branch de release para homologar uma versão pelo analista, merge dos braches das features a serem homologadas no respectivo branch de release pelo analista, durante a homologação criação de branches para correção de Bugs encontrados, após homologação de uma versão, merge no master e desevelopment e geração da TAG respectiva da versão a partir do master.

2.1.2. Premissas

1. O repositório deverá ser criado pela equipe de infra-estrutura seguindo os padrões de segurança e permissão e composto por dois branches: master e development;
2. Todo o desenvolvimento ocorre em branches para cada respectiva tarefa, no inicio até a estabilização do projeto o desenvolvimento poderá ocorrer no branch development pelo arquiteto;
3. Existe integração contínua diária automática para cada banch de prefixo "feature" a qual deve validar no mínimo a compilação;
4. A validação do desenvolvimento de cada feature ou melhoria deve ocorrer no branch de development, sendo assim ao término da feature o desenvolvedor deve solicitar um merge em development através de um Pull Request;
5. Quando um Pull Request é criado os analistas do projeto serão notificados e poderão autorizar ou não o merge em developement;
6. Autorizando o merge em development o Stash fará o merge e movimentará a tarefa no JIRA para o passo "90% - Construida";
7. Se durante a validação de uma feature no ambiente de for encontrado Bugs o analista deverá devolver a tarefa no JIRA sem a criação de Bugs e o desenvolvimento poderá reiniciar na etapa de Planejamento ou Definição dependendo do motivo da rejeição;
8. Exite um plano de build que irá no mínimo compilar e empacotar o código e posteriormente fará deploy no ambiente de desenvolvimento quando houver o aceite de um Pull Request;
9. Quando for necessário iniciar a homologação de uma versão, o analista deverá criar um branch com o prefixo "release" a partir do branch master;
10. Após deve fazer um Pull Request do branch de cada feature ou melhoria para o branch da versão (release) desejado e após autorizar o merge;
11. Irá existir um plano de build para no mínimo compilar e empacotar os fontes do projeto para que fiquei disponível para deploy em uma ambiente de homologação;
12. O analista ou equipe de QA poderão fazer deploy em homologação a qualquer momento escolhendo uma das versões geradas no plano de build;
13. Durante a homologação Bugs poderão serem encontrados e reportados no Jira, após a confirmação da existência do Bug o analista deverá criar um branch, com prefixo bugfix, para correção do mesmo copiando o código do branch de release que o Bug foi diagnosticado;
14. Após receber o tarefa do Bug com o branch devidamente criado o programador poderá inciar a correção do Bug fazendo checkout do branch indicado;
15. Após o término da correção do Bug o programador deverá abrir um Pull Request solicitando a homologação da correção no branch de release que o Bug foi encontrado, ou seja no branch da versão que esta no Fix version da tarefa Bug no Jira;
16. O passos 13,14 e 15 devem ocorrer até que todos os Bugs forem corrigidos e a versão do branch de release em homologação seja considerada homologada para se publicada em produção;
17. Para entregar a versão definitivamente em produção o analista deve abrir dois Pull Request do branch de release: um Pull Request para o branch development e outro para o branch master.
18. Aceitar o merge do Pull Request para os branches developement e master (no aceite do Pull Request em master solicitar a exclusão do branch de release) para efetivar o merge da versão nos respectivos branches;
19. Deve existir um plano build do Bamboo que apontará para master o será responsável pela geração da TAG da versão equivalente finalizando o processo de construção, homologação e entrega de uma versão de software ou projeto;
20. Quando houver um Bug de produção relatado no JIRA e devidamente confirmado pelos analistas, um branch com prefixo "hotfix" deverá ser criado a partir do branch master para a devida correção deste Bug;
21. O analista deverá decidir se o Bug de produção vai ser homologado em uma versão que já esta em homologação (branch de release), ou se vai criar um novo branch com o prefixo "release" para homologar o Hotfix. Em ambos os casos deve seguir os passos 10 a 19.

2.1.3. Fluxo Principal

1. Desenvolvimento diario
    a. Programador realizar checkout do branch relativo a tarefa que irá desenvolver;
    b. Programador realizar alterações e commits no brach da tarefa;
    c. Programador faz Push do código comitado no branch da tarefa para o servidor GIT;
    d. Programador acessa o Stash e abre um Pull Request do branch da tarefa para o branch development, solicitando assim a validação do desenvolvimento;
    e. Analista aceita o Pull Request aprovando a construção e iniciando a validação do desenvolvimento;
    f. Analista rejeita a validação devolvendo a tarefa no JIRA ou aceita a validação deixando a tarefa no status "90% - Construida" pronta para iniciar a homologação.
2. Promoção para homologação (equipe e usuarios acordar momentos adequados, exemplo, a cada 15 dias)
    a. QA criar branch com prefixo "release" e com nome da versão a ser homologada.
    b. Analista cria um Pull Request para cada feature solicitando a entrada em homologação no respectivo branch de release do campo Fix version;
    c. QA analisa o código que irá entrar em homologação e autoriza o merge via Stash;
    d. Stash move workflow da feature no JIRA para o status "95% - Em Homologacao";

    e. Bamboo gerar um artefato com os fontes das features a serem homologas;
    f. QA aciona o Bamboo para fazer deploy do artefato gerado pelo plano do respectivo release a ser homologado;
    g. QA faz os devidos teste e encontra Bug na homologação de features;
    h. QA reporta uma sub-tarefa de Bug na feature testada;
    i. Analista valida a veracidade do Bug reportado, se for um Bug criar um branch com o prefixo "bugfix" para o programado faça as devidas correções;
    j. Programado corrige o Bug e faz um Pull Request para solicitar a homologação da correção no branch de release;
    k. QA faz teste e valida a correção do Bug;
    l. Passos h a k até que todos os Bugs sejam corrigidos;
    m. Usuário protocola o aceite de homologação de cada feature;
3. Reintegração de desenvolvimento
    a. QA faz merge via Pull Request do branch de release para os branch development;
    b. QA resolver conflitos com o branch de development se existir.
    c. QA realizar commits de resolucao de conflitos no branch development se existir.
4. Promocao para producao
    a. QA faz merge via Pull Request do branch de release para os branch master (ao autorizar o merge deve marcar a opção para deletar o branch de release);
    b. Bamboo recebe fontes do release em master e executa plano de build para geração do artefato que vai a produção e da TAG.
    c. QA comunicar Administrador de nova versao de producao disponivel.
    d. Administrador realizar deployment de producao.

3. REFERENCIAS

https://www.atlassian.com/git/workflows#!workflow-gitflow