Informe SVN x `git` / Gitlab

Leslie H. Watter

Sumário

1 Histórico

Atualmente na Celepar existem dois sistemas de controle de versão utilizados para manter o código fonte das aplicações:

1. **CVS**: utilizado primariamente para projetos com codificação de caracteres ISO-8859-1 (Latin1), e, por ser o mais antigo, concentra o maior número de projetos (em 01/10/2018 mais de 950 projetos). Não mais utilizado para novos projetos, apenas sendo mantido para os projetos legados.

2. **SVN**: utilizado para projetos cuja codificação de caracteres é UTF8. É um controle de versão mais novo, introduzido na Celepar em meados de 2006 (em 01/10/2018 mais de 500 projetos).

Ocorre que, à medida em que o tempo passa, o desenvolvimento evolui e ferramentas novas e consequentemente melhores e mais poderosas surgem para facilitar a vida dos desenvolvedores.

Uma dessas ferramentas é o `git`, um controle de versão que vem para substituir o SVN corporativo. A primeira versão estável do `git` foi lançada em 2005 e desde então sua adoção tem crescido muito mundo afora.

Já há algum tempo o mercado adotou o `git` como controle de versão padrão para a grande maioria de projetos, desde open-source até código fechado, em grande parte por sua flexibilidade e quantidade de funcionalidades embutidas.

Em junho de 2018 o GitHub (site que hospeda repositórios `git`) tinha 28 milhões de usuários e hospedava 57 milhões de repositórios, dentre estes 28 milhões públicos, sendo a maior hospedagem de código fonte do mundo.

2 Trabalho Desenvolvido na Celepar

Voltando ao contexto da Celepar, ocorre há algum tempo a iniciativa de migração do controle de versão SVN para `git`, com a preparação dos softwares de deploy (estaleiro e jenkins), instalação de servidores de aplicação (GitLab), e treinamento das equipes da GTI, DIOPE e GIC/COTAP de modo a disponibilizar corporativamente a solução. Isso tudo após ter passado por várias reuniões e apresentações, inclusive para o comitê de internalização de novas tecnologias.

Foram avaliadas possíveis soluções para disponibilização de repositórios `git` de maneira corporativa a serem implementadas internamente à Celepar.

Para uso do `git` na Celepar considerou-se:

1. mantermos o código fonte local (*no Datacenter da Celepar*)

2. facilidade de uso e de administração

3. estabilidade de produto (e boa classificação nos relatórios do Forrest e Gartner).

A partir desses requisitos, foi selecionado o GitLab (`http://gitlab.com`) para gerenciador de repositórios `git`.

Após esse primeiro passo, foram desenvolvidas as seguintes atividades:

2.1 Verificação de Migração do SVN

Foi feita uma simulação de migração de SVN para `git` para todos os projetos que haviam no SVN. Essa simulação serviu para identificar alguns pontos a serem corrigidos antes da migração efetiva dos projetos, permitindo uma transição mais tranquila do ponto de vista técnico.

2.2 Preparação dos Scripts de Migração

Juntamente com a simulação de migração, foram desenvolvidos scripts para facilitar a migração dos projetos do SVN para GIT, aplicando os devidos ajustes. Entre outras coisas, foi necessário identificar usuários que já haviam sido, inclusive, removidos por conta de não fazerem mais parte da empresa e/ou serem terceiros, para manter o histórico completo dos projetos.

2.3 Configuração e Manutenção do GitLab

Foram necessários algumas ações para permitir a utilização do GitLab dentro da Celepar, para adequação ao nosso ambiente, dentre elas:

- **Configuração de suporte ao LDAP.** Na versão comunidade (utilizada na Celepar) somente existe a autenticação de usuários no LDAP. Os usuários devem ser criados no GitLab, para que este controle a autorização do que cada usuário pode fazer dentro dos diferentes perfis no sistema. Como não há a sincronização com grupos do SVN nessa versão, se faz necessário executar a criação e manutenção de usuários manualmente.

- **Estudo da estrutura de usuários, projetos e grupos.** Foi necessário estudar a estrutura do GitLab, o gerenciamento de permissões de grupos, projetos e usuários para só então propor um modo de utilização dessa estrutura que ficasse semelhante à cultura preexistente na Celepar. Essa proposta (menor mudança) visa reduzir o impacto nas ferramentas utilizadas para deploy e também na utilização do GitLab pelos desenvolvedores.

- **Interação com GTI para tratar a infraestrutura.** Antes da liberação do GitLab para inclusão de projetos, foram feitos diversas atualizações (programadas) e testes no GitLab. Dessa maneira, o procedimento de atualização do sistema está sacramentado.

- **Atualização de versões.** As atualizações de sistema ocorrem mensalmente, primeiro em homologação e depois em produção. Todos os meses no dia 22 é lançada uma nova versão do GitLab, com correções e melhorias. Manter-se atualizado nessa velocidade de evolução requereu um treinamento prévio e colaboração entre as diferentes áreas.

2.4 Preparação do Jenkins

Visando automatizar as tarefas de criação de usuários, projetos e grupos dentro do GitLab, além de reduzir o retrabalho por procedimentos interrompidos no meio, foram desenvolvidos projetos no *Jenkins* que automatizam essas tarefas.

Os seguintes projetos foram criados no Jenkins:

- **Cria Projeto** – Cria toda estrutura de grupo e projeto necessária, sendo informado nome do projeto e usuários integrantes.

- **Cria Usuário** – Cria usuário no GitLab e atribui as permissões corretas.

- **Atribui Usuário a Projeto** – Insere usuário no grupo/projeto informado.

- **Remove Usuário de Projeto** – Remove usuário do grupo/projeto informado.

- **Arruma Permissões** – Projeto que corrige permissões de acesso, executado automaticamente.

- **template de deploy de projetos PHP** – template para deploy de projetos PHP.

2.5 Desenvolvimento de Treinamento GIT

Como parte integrante do projeto de implantação do `git` na Celepar, verificou-se a necessidade de treinar os usuários.

Observou-se que vários desenvolvedores sabem utilizar a ferramenta de controle de versão apenas superficialmente, executando somente ações pré-programadas, sem contudo compreender o funcionamento da mesma.

Essa última observação, reforçou ainda mais a necessidade de treinamento para todos os desenvolvedores, de maneira a homogeneizar o conhecimento e permitir uma padronização de processos.

Dessa maneira, foi desenvolvido um treinamento incremental abrangendo os seguintes contextos:

- **Teórico** – é abordada a teoria de controle de versão e diferenças conceituais entre SVN e `git`.

- **Linha de Comando** – o primeiro contato com a utilização do `git` é feito através da linha de comando (tem menos elementos de distração) com foco exclusivamente no controle de versão e as ações executadas por ele/nele, visando fixar e aplicar os conceitos vistos na parte teórica.

- **Eclipse** – continuação do treinamento utilizando a interface do eclipse para executar as tarefas do cotidiano do desenvolvedor, agora aplicando os conceitos aprendidos e já fixados. Nessa fase do treinamento, são mostradas algumas das diferentes maneiras de executar as tarefas do controle de versão no eclipse.

- **GitLab** – parte do treinamento que mostra a interação entre o desenvolvedor, servidor GitLab, projeto e outros desenvolvedores. Utilizada para mostrar a maneira de utilização da ferramenta em conjunto com a equipe de desenvolvimento do projeto.

Exceto a parte teórica, o treinamento é completamente prático */hands on*, com o objetivo de fazer com que o desenvolvedor tenha o conhecimento necessário para trabalhar com o `git` /GitLab e interagir com os outros desenvolvedores usando a ferramenta.

2.6 Elaboração do processo de migração

Um dos fatores principais na negociação com GTIC durante a apresentação da proposta de instalação do `git` como controle de versão foi que somente dois sistemas ficassem em funcionamento.

Dessa forma, ficou combinado que todos os projetos do SVN serão migrados para o ~git~/GitLab no seu devido tempo.

Para que essa migração ocorra, se fez necessário definir um processo de migração dos projetos, que pode ser visto na figura fluxo do processo de migração de projetos.

A migração será executada pela GIC e o resultado verificado pelos desenvolvedores de cada projeto, em um momento a ser combinado previamente por ambas as partes.

2.7 Interação com GTI para atualização do estaleiro

Durante o processo de implantação, verificou-se também a necessidade de atualização da ferramenta de deploy *estaleiro*, para suportar o `git` como ferramenta de controle de versão.

Essa necessidade foi repassada para a GTIC e implementada no estaleiro na versão 3.0, já implantada em produção.

2.8 Treinamento de equipe da GTI

Além da atualização das ferramentas, faz-se necessário também a atualização das pessoas que trabalham com elas. Por esse motivo foi aplicado o treinamento `git` para uma turma da GTIC, de modo a instruir as pessoas que irão administrar o GitLab.

2.9 Treinamento de equipe da DIOPE e Documentação de Procedimentos

A DIOPE, como operacionalizadora da criação/manutenção de usuários/projetos recebeu treinamento de utilização dos projetos no Jenkins que fazem as operações no GitLab, bem como documentação de procedimentos a serem adotados.

3 Definições – `git` e Gitlab

> Git é um sistema de controle de versões distribuído, usado principalmente no desenvolvimento de software.
>
> Cada diretório de trabalho do `git` é um repositório com um histórico completo e habilidade total de acompanhamento das revisões, não dependente de acesso a uma rede ou a um servidor central.
>
> (Adaptado da wikipedia - `https://pt.wikipedia.org/wiki/Git`)

> O GitLab é um gerenciador de repositório de software baseado em `git`, com suporte a Wiki, gerenciamento de tarefas e CI/CD. GitLab é similar ao GitHub, mas o GitLab permite que os desenvolvedores armazenem o código em seus próprios servidores, ao invés de servidores de terceiros. Ele é software livre, distribuído pela Licença MIT.
>
> É usado por mais de 100 000 organizações, incluindo IPqM (Marinha do Brasil) , IBM, NASA, Alibaba, Invincea, O'Reilly Media, CERN, Projeto GNOME e SpaceX.
>
> (Adaptado da wikipedia - `https://pt.wikipedia.org/wiki/GitLab`)

4 Diferenças Estruturais entre SVN e GitLab

Nesta seção trataremos das principais diferenças entre o SVN e GitLab. São elas:

1. centralizado versus distribuído
2. repositórios
3. versões
4. tratamento de branches
5. utilização de tags
6. grupos e projetos
7. permissões dadas a projetos e grupos

4.1 Centralizado versus Distribuído

O SVN é um controle de versões muito parecido com o CVS, mantendo a mesma estrutura centralizada em um único servidor. Essa característica facilita a transição de um para o outro sem que haja maiores implicações, uma vez que o conceito é muito similar.

Por outro lado, o `git` utiliza um conceito novo, sendo um sistema distribuído, onde cada repositório pode fazer as vezes de "*servidor*". Algumas operações novas (*clone, fetch, push, pull, remote*) são adicionadas ao rol de operações no dia-a-dia do desenvolvedor, aumentando um pouco a complexidade do ambiente. Essas operações agregam funcionalidades importantes ao desenvolvimento, permitindo a implementação de processos até então inexistentes, como por exemplo a validação de código antes da entrada no repositório oficial.

Devido à essas mudanças conceituais, faz-se mister a aplicação do treinamento fornecido pela GIC para todos os desenvolvedores, de maneira a homogeineizar o conhecimento e ampliar os horizontes de interação entre os desenvolvedores.

4.2 Repositórios

A estrutura do SVN é compartimentalizada em repositórios, onde cada repositório representa um único projeto e este repositório tem um único lugar no servidor.

No GitLab mantêm-se a mesma idéia onde um repositório representa somente um único projeto. Entretanto, devido à natureza distribuída do `git`, é possível ter várias "cópias"do repositório dentro do servidor GitLab.

Logo pensa-se na questão: Qual é a versão oficial do projeto? Essa questão é tratada via convenção, onde todos os repositórios oficiais estão organizados em grupos cujo nome começa com "gcgit-". Sendo assim o projeto "alfa"estará hospedado no grupo *gcgit-alfa* dentro do projeto *alfa*.

Sendo assim, a estrutura do GitLab para tratar o projeto "alfa"é:

- gcgit-alfa – grupo celepar `git` para o projeto alfa
  - alfa – projeto alfa
    - repositório `git`
    - wiki
    - tíquetes
    - (outras ferramentas associadas ao projeto alfa)

4.3 Versões

Enquanto o SVN utiliza uma numeração sequencial para as versões, o `git` usa o resultado de um *hash* SHA sobre o commit feito. Sendo assim temos

- SVN - números sequenciais
  - 1 (primeiro commit)
  - 2 (segundo commit)
  - 3 (terceiro commit)
- `git` - sequência de hashes SHA
  - 53f9a3e157dbbc901a02ac2c73346d375e24978c (primeiro commit)
  - 1fb9df724833b0d0dffaefc6886ac60247c67a0c (segundo commit)
  - f0ac6e39433c1d7e9339207aa4d01e9bf7a05b8a (terceiro commit)

A finalidade do hash que o `git` utiliza para registrar os commits é, além de manter a identidade do commit, garantir a imutabilidade do conteúdo comitado, pois qualquer modificação nas informações do commit altera o hash produzido.

4.4 Tratamento de branches

Uma das grandes vantagens do `git` frente ao SVN no tratamento de branches é a leveza na criação e manutenção de branches de trabalho separados, além das diferentes abordagens de *merge* melhorado no `git`.

Em termos de espaço ocupado:

- o SVN faz uma cópia da árvore de trabalho para cada branch criado.
- o `git` cria um apontador para um commit de referência, sem ocupar espaço extra (além do marcador do branch)

Estratégias de Merge:

- o SVN faz merge das cópias dos arquivos que foram alteradas no ramo.
- o `git` tem duas políticas de merge:
  - *fast-forward* – quando não há conflitos no merge
  - *3-way merge* – ou merge de 3 vias que identifica a origem dos conflitos e minimiza-os permitindo a resolução mais simples.

A utilização de branches para desenvolvimento de funcionalidades novas é comum no `git`, uma vez que as estratégias de *merge* utilizadas são mais eficientes que as utilizadas pelo SVN.

4.5 Utilização de tags

As tags são tratadas da seguinte maneira:

- SVN: é criada uma cópia do repositório no momento atual e colocada sob a árvore /tags/nome_da_tag no SVN. Ou seja, para cada tag, o espaço equivalente ao projeto naquele instante é ocupado.
- `git`: o `git` tem dois tipos de tags: leve e anotada. Essa abordagem é muito mais econômica em termos de espaço.
  - leve: apenas um marcador para o commit que identifica a tag.
  - anotada: um commit novo com a tag, que pode ser assinado.

No SVN, uma atividade não recomendada e que ocorria em alguns setores é a re-escrita de uma tag. Essa situação ocorria no momento em que o desenvolvedor já havia aberto uma OS para deploy com uma tag, por exemplo *v_1.0* e percebia que havia esquecido de commitar um arquivo. Nesse momento, para evitar re-abrir a OS, o desenvolvedor reescrevia a tag e ligava para a DIOPE pedindo o redeploy da tag.

No `git` + GitLab **não será possível reescrever tags**. Uma vez que a tag foi publicada no servidor, não será permitido a reutilização da mesma tag. Essa política se faz necessária pelo fato de o `git` não atualizar tags já publicadas no repositório remoto.

4.6 Grupos e Projetos

Com relação ao tratamento de grupos e projetos, no SVN, é definido um grupo no LDAP (grupo-celepar-svn-projeto) e criado um repositório para o projeto no servidor SVN. Existe a relação um grupo no SVN para um repositório/projeto.

No GitLab, existe o conceito de grupo de projetos, onde um grupo pode abrigar mais de um projeto. Entretanto, para manter a mesma estrutura que a empresa já utiliza, convencionamos a utilização de apenas um projeto por grupo. Essa convenção facilitou a adaptação da ferramenta de deploy estaleiro.

4.7 Permissões dadas a usuários em projetos e grupos

Atualmente no SVN, somente os membros do projeto conseguem ver o repositório e o código fonte.

No GitLab, para empregados da Celepar, todos os projetos serão visíveis e passíveis de download/fork, entretanto **somente os membros do projeto poderão fazer alterações no repositório** original (commits) e aceitar contribuições. Essa característica facilita a cooperação mútua entre os desenvolvedores da companhia auxiliando no reuso de código.

A restrição aplicada para terceiros/fornecedores é diferente, onde somente os projetos aos quais eles estiverem associados estarão visíveis para esses usuários.

4.7.1 Projetos com senha no Controle de Versão

Projetos que porventura tenham senhas de conexões a bancos de dados, e outras conexões deverão ser analisados para ter essas senhas armazenadas em locais específicos.

Há na estrutura atual do PHP + Laravel a configuração de conexões em arquivos que ficam no SVN. Essas senhas serão migradas para o projeto Jenkins respectivo, de maneira a saírem do controle de versão indo para um controle de credenciais fornecido pelo Jenkins.

Na estrutura de projetos Java, ocorre caso semelhante para as conexões a bancos de dados. Entretanto com a separação de ambientes e redes (desenvolvimento, homologação e produção), a criticidade da informação é menor, haja vista já haver um compartilhamento desse tipo de informação para outros projetos, como é o caso das bases do sentinela.

4.7.2 Casos Especiais

Em alguns casos, pode haver necessidade de deixar o projeto com acesso somente aos envolvidos, por questões de *Non Disclosure Agreements* ou outras questões. Esse tipo de situação deverá ser tratado isoladamente, e como caso especial foge do fluxo padrão.

5 Funcionalidades GitLab

Dentre as diversas funcionalidades encontradas no GitLab destacam-se as seguintes:

Revisão de Código É possível fazer a revisão de código através da interface web e/ou linha de comando antes de aplicá-la ao ramo principal de desenvolvimento.

Rastreamento de Bugs O GitLab possui um sistema de tíquetes interno, separado por projeto. Conforme orientação da equipe do PDS, o sistema de tíquetes do GitLab será utilizado apenas para comunicação interna à equipe de desenvolvimento, **mantendo-se** os controles e sistemas de gestão atuais (PDSMantis, Clarity, etc).

Wiki Dentro de cada projeto, o GitLab fornece um *wiki* para o projeto, onde a equipe de desenvolvimento pode (por opção própria) documentar informações referentes ao projeto. Esse wiki também é mantido como um repositório `git`, o que facilita o controle de atualizações.

Notificações O GitLab permite o envio de notificações em três âmbitos: (1) global, (2) por grupo, (3) por projeto. O GitLab também envia e-mails (configurável) para os usuários de forma a notificá-los sobre modificações de conta, inclusão em projetos novos, etc de maneira a agilizar o processo de comunicação.

Times A organização em grupos e projetos do GitLab permite uma interação facilitada entre os indivíduos dos times de desenvolvimento, permitindo conversas/menções dentro das diferentes ferramentas (tíquetes, wiki, etc), possibilitando a resolução de problemas e manutenção do histórico de resoluções dentro da própria ferramenta.

5.1 Interação com Fornecedores via *merge request*

Um dos problemas enfrentados atualmente na interação com terceiros é o fato de que eles precisam *comitar* o código no controle de versão no projeto "quente"que é utilizado internamente. Quaisquer modificações que eles façam, é integrada primeiro e testada / validada depois.

Utilizando o `git` + GitLab é possível interagir de uma maneira mais profissional/atualizada onde o terceiro consegue trabalhar nas modificações solicidadas de forma que, ao final do trabalho ele consegue submeter as propostas de mudanças para a equipe interna testar antes de integrá-las ao código oficial.

Essa funcionalidade permite aos nossos desenvolvedores avaliar primeiro o código implementado e aceitá-lo/rejeitá-lo antes da modificação entrar no repositório oficial, permitindo uma melhoria no processo atual.

6 Processo de Migração de Projetos SVN para GitLab

6.1 SOC para criação de projeto GIT

Foi criada junto à DIOPE uma SOC para criação de projetos GIT que encontra-se no seguinte caminho:

Tipo Operacional

Categoria do Serviço Administração de Usuários

Serviço Permissão e manutenção de acessos a projetos. Criação de projetos no ambiente de desenvolvimento (CVS - SVN - LDAP - Interdev - Estaleiro - Mantis - Hudson - Sonar - GIT) - DIOPE

- No formulário deve-se selecionar o item:
  - GIT: Criação de projeto em desenvolvimento

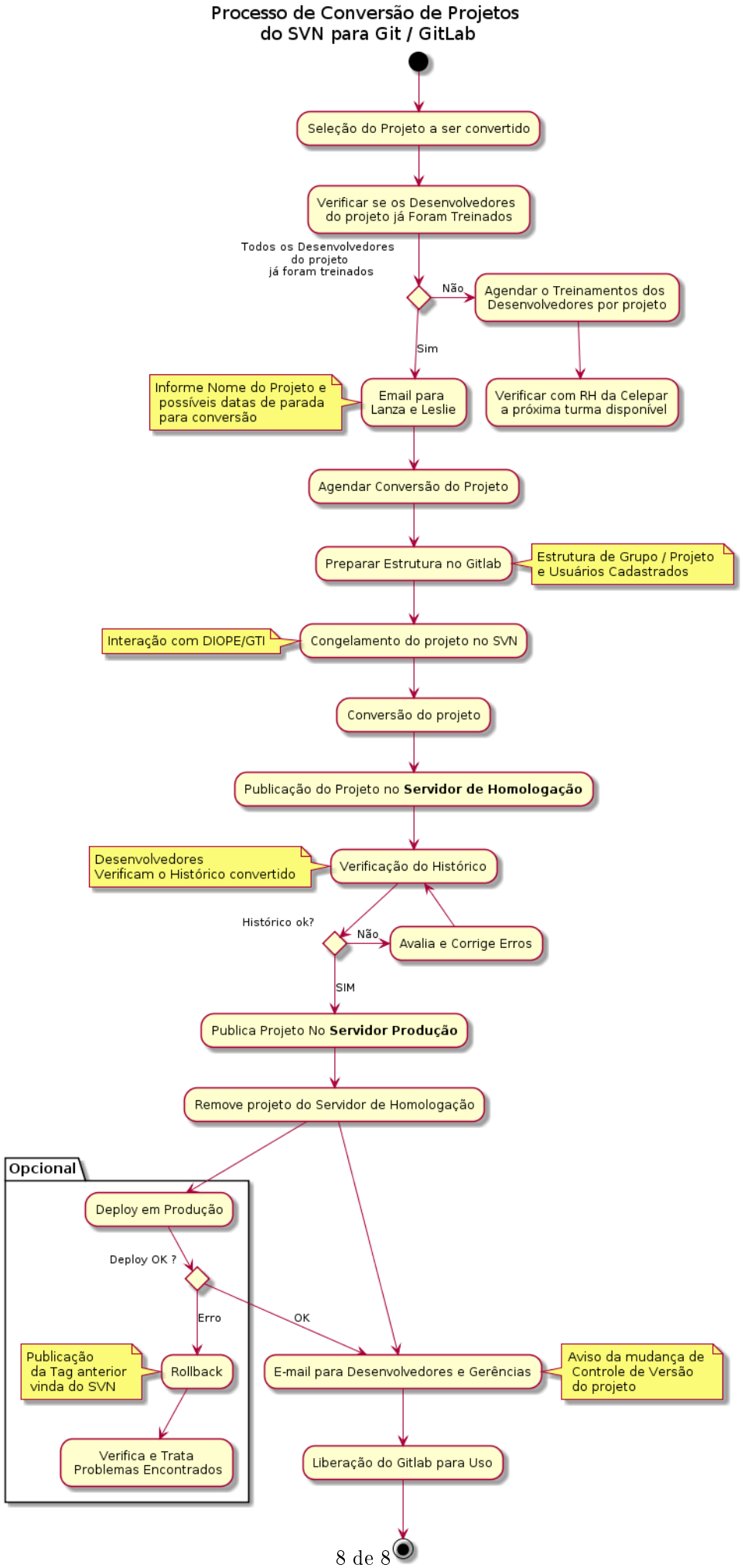

Figura 1: Fluxo do Processo de Migração