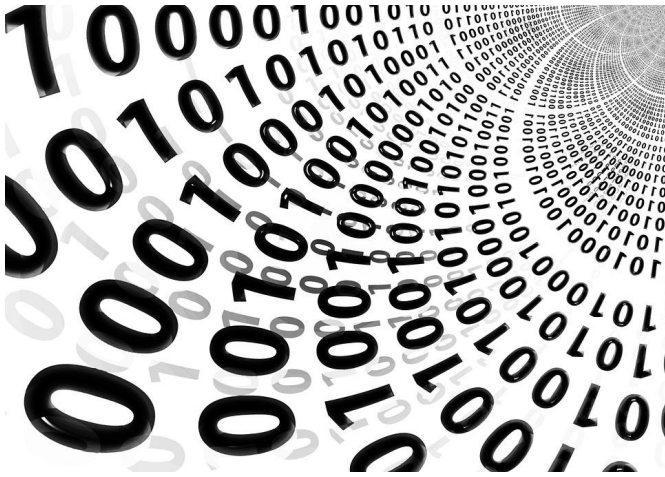
Práticas em Desenvolvimento e Operação - DevOps

Parte 02 – Gerenciamento de Código Fonte

Vanderson Bossi
vanderson.bossi@faculdadeimpacta.com.br

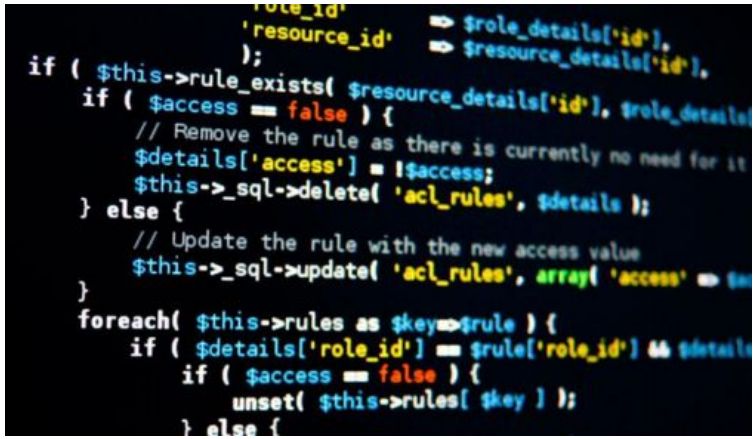
O que é o controle do código-fonte?

O controle de versões ou controle do código-fonte monitora e gerencia alterações efetuadas no código. Há os Sistemas de Source Control Management (SCM – Gerenciamento de controle do código-fonte) que disponibilizam um histórico de execuções de desenvolvimento de código com o intuito de ajudar na resolução de conflitos durante a integração de diferentes atualizações do Sistema que está sendo desenvolvido ou atualizado.



Simplificando o processo de desenvolvimento e tornando-se uma fonte centralizada para todo o código do projeto de software.

Tornando-se possível um desenvolvedor colaborar com a programação do código num projeto em equipe mesmo isolando o seu trabalho até que esteja pronto e solucionar rapidamente problemas ao identificar quem fez alterações e quais foram elas. (AMAZON, 2018)



O que é o Git?

O Git é um sistema de controle de versão de código aberto (VCS) distribuído que permite armazenar código, rastrear histórico de revisão, mesclar alterações de código e reverter para versões de código anteriores quando necessário. A cópia criada no seu repositório é conhecida como ramificação e utilizando essa ramificação é possível trabalhar no código independentemente da versão estável da base de código.

Após a finalização das alterações pode-se armazená-las como um conjunto de diferenças, conhecido como uma confirmação, também, pode-se extrair confirmações de outros colaboradores para o repositório, enviar confirmações para outras pessoas e mesclar confirmações de volta para a versão principal do repositório. (ALJORD; CHACON, 2009)

Toda vez que é feito um commit, o Git compara as versões anteriores com uma operação visualizável chamada diff. Se houve uma mudança de commits anteriores, o Git armazena um novo snapshot no repositório.

Download:

https://git-scm.com/

Repositório:

https://github.com/

Toda vez que é feito um commit, o Git compara as versões anteriores com uma operação visualizável chamada diff. Se houve uma mudança de commits anteriores, o Git armazena um novo snapshot no repositório.

Download:

https://git-scm.com/

Repositório:

https://github.com/

Quais são os benefícios do Git?

O Git tem vários benefícios importantes:

- **Controle histórico de alterações**: Pode-se analisar por um gráfico de como os commits mudaram ao longo do tempo, ver quando e por quem as alterações foram feitas e reverter para um commit anterior, se necessário. Esse histórico facilita a identificação e correção de erros.

Quais são os benefícios do Git?

- **Trabalho em equipe**: Possibilita compartilhar o código com os colegas de equipe para revisão. Os recursos de ramificação e revisão permitem o desenvolvimento simultâneo, ou seja, vários desenvolvedores podem trabalhar no mesmo arquivo e resolver as diferenças mais tarde.

Quais são os benefícios do Git?

- **Melhore a velocidade e a produtividade da equipe**: O Git facilita o acompanhamento de alterações no seu código pela equipe.

- **Disponibilidade e Redundância**: Git é um VCS distribuído, o que significa que não existe um único local central onde tudo é armazenado. Em um sistema distribuído, há vários backups caso necessite, então é possível trabalhar off-line e confirmar as alterações quando estiverem prontos.

Quais são os benefícios do Git?

- **Popularidade do Git:** O Git é suportado por muitos ambientes de desenvolvimento integrados (IDEs) e muitas ferramentas de desenvolvedores populares, incluindo AWS CodeCommit , Jenkins e Travis. Existem muitos recursos gratuitos do Git disponíveis, como a página de código aberto do Git. (ALJORD; CHACON, 2009)

O que é um Branch?

O Git não armazena dados como uma série de mudanças ou deltas, mas sim como uma série de snapshots (registro instantâneo) e quando é feito um commit (atualização do repositório) no Git o sistema guarda um objeto commit que contém um ponteiro para o snapshot do conteúdo que foi colocado na área de seleção.

Para exemplificar iremos supor um diretório contendo três arquivos, após serem colocar na área de seleção e fazer o commit. Sendo que, colocar na área de seleção cria o checksum de cada arquivo e armazenar esta versão do arquivo no repositório Git (o Git se refere a eles como blobs), e acrescenta este checksum à área de seleção (staging con):

$ git add README test.rb LICENSE

$ git commit –m ‘commit inicial do meu projeto’

Quando é criado um commit executando git commit, o Git calcula o checksum de cada subdiretório (neste caso, apenas o diretório raiz do projeto) e armazena os objetos de árvore no repositório Git. O Git em seguida, cria um objeto commit que tem os metadados e um ponteiro para a árvore do projeto raiz, então ele pode recriar este snapshot quando necessário.

Seu repositório Git agora contém cinco objetos: um blob para o conteúdo de cada um dos três arquivos, uma árvore que lista o conteúdo do diretório e especifica quais nomes de arquivos são armazenados em quais blobs, e um commit com o ponteiro para a raiz dessa árvore com todos os metadados do commit.

Caso haja modificação do código fonte e fizer um commit (atualizar o repositório) novamente o próximo commit armazenará um ponteiro para o commit imediatamente anterior.

Em resumo: Um branch no Git é simplesmente um leve ponteiro móvel para um desses commits. O nome do branch padrão no Git é cone. Quando inicialmente fez commits há um branch principal (cone branch) que aponta para o último commit que foi feito sendo que cada vez que é realizado um commit ele avança automaticamente. (ALJORD; CHACON, 2009)

Referências

ALJORD, Patrick; CHACON, Scott. **Pro Git.** Estados Unidos: Apress, 2009. 300 p

AMAZON. **O que é o controle do código-fonte?** Disponível em: <https://aws.amazon.com/pt/devops/source-control/>. Acesso em: 09 abr. 2018

Obrigado!