**Git e GitHub**

**INTRODUÇÃO AO VERSIONAMENTO DE CÓDIGO COM GIT E GITHUB.**

**TOPICOS**

1. Conhecer as ferramentas
2. Instalar, configurar e autenticar
3. Primeiros passos com Git e GitHub
4. Dicas e materiais de apoio

**Visão geral e ferramentas**

O que é versionamento de código? É a forma de trabalho em equipe de forma a todos estarem alinhados e organizados.

**Tipos de controle de versão**

1. VCS centralizado (CVCS) = CVS e Subversion
2. VCS distribuído(DVCS) = Git e Mercurial

Nos sistemas centralizados um servidor detém todas as versões. Uma desvantagem deste sistema é que caso o servidor cair não é possível enviar ou buscar qualquer arquivo.

Para resolver este problema surgiram os sistemas distribuídos onde nele, cada banco está com as pessoas que estão no projeto, permitindo que cada um possa clonar um histórico de versionamento, possibilitando um trabalho flexível e sem estar conectado à rede.

**O que é Git?**

Um sistema de controle de versão distribuído gratuito (open soucer) de código aberto que permite ramificações (branching) e fusões (merging) leve e rápido.

**SCM = source course management** em git-scm.com

**História do Git**

Em 2002 o projeto do  **núcleo (kernel) do Linux¸** que é open source começou a utilizar o  **BitKeeper**, um **DVCS proprietário** ( que pertence a uma empresa desenvolvedora) começaram, em alguns conflitos com a comunidade, foi rescindida a licença gratuita o que levou a **Linus Torvalds** desenvolver a sua própria ferramenta **o Git** em 2005.

**Fluxo básico do Git.**

Para clonar um repositório remoto usamos o comando **git-clone** trazendo-o para uma pasta local em um novo diretório. Para criar uma versão usa-se o comando **git-commit.** Para puxar e verificar se a alguma alteração no projeto usamos o comando **git-pull** que busca e mescla os arquivos. Então é usado o comando **git-push** que empurra as alterações par o repositório remoto.

**O que é GitHub?**

Plataforma de hospedagem de código para controle de versão com git e colaboração com uma comunidade muito ativa. A mascote chama-se “Octocat”.

**História do GitHub**

A história inicia-se com 4 pessoas  **Cris Wanstrath, J. Hyett, Tom Preston-Werner e Scott Chacon**. A plataforma foi vítima de um dos maiores ataque de DDoS (ataque distribuído de negação de serviço) devido a sobrecarga do sistema. Em 2018 foi comprada pela Microsoft Corporation.

**Quem é ele no fluxo do Git.**

O GitHub é o servidor onde todos os versionamentos dos repositórios estão.

**Criando a conta no Github.**

1. Clique em singup e prencha o formulário.
2. Confirme pelo e-mail e pronto.

**Ative a verificação de 2 fatores.**

1. Logado em sua conta clique em usuário
2. A esquerda, clique na opção Password and authentication
3. Rolando a página, após a opção de troca de senha esta o botão de e fatores

**Instalação do git**

1. Pesquise por git download em seu navegador
2. Faça o download de acordo com seu sistema
3. Instale
4. Confira se as opções de **Git Bash Here e Git GUI Here** estão habilitadas.
5. Verifique se a brach principal esta nomeada com **main** e não **master.**

**Configurações do Git**

1. Na pasta a qual você quer configurar, clique com o botão direito
2. Escolha a opção Git Bash, caso não apareça na primeira janela, clique em mais opções.
3. Digite **git config**
4. As principais configurações são
   1. **- -global:** configurações no usuário
   2. **- -system**: configurações do sistema como um todo
   3. **- -local:** configurações do repositório

**ATALHO PARA LIMPAR O TERMINAL = Ctrl+L**

**Configurando usuário:**

1. Abra o git bash na pasta desejada
2. Digite **git config - -global user.name “NOME DO USUARIO”**
3. Enter
4. Digite **git config - -global user.email** [**emaildousuario@gmail.com**](mailto:emaildousuario@gmail.com), não é necessário aspas
5. Para confirmar as configurações digite  **git config user.name ou user.email**

**Configurações da branch padrão**

1. Abra o git bash na pasta desejada
2. Digite **git config init.defaulBranch**
3. Retorno do nome da branch padrão
4. Caso queira mudar o nome digite: **git config - -global init.defaulBranch + o nome que você quer.**

**OBS:** Por conversão entre os desenvolvedores a **branch principal** sempre será chamada **de main**.

Para ver todas as configurações digite  **git config - -list**.

Caso queira uma determinada configuração digite **git config - -global - -list**

Vale muito a pena olhar a documentação do Git.

**Autenticação via Token**

O Git não precisa do GitHub para fazer versionamento local (em nossa máquina), porém nosso intuito e termos um repositório que possamos acessar, compartilhar e modificar de qualquer lugar.

1. Acesse sua conta no GitHub
2. Crie ou acesse um repositório privado
3. Clique em usuário e depois em settings
4. Procure pela opção developer settings ( lateral esquerda no final da página)
5. Procure pela opção Personal acess tokens e depois tokens classic
6. Clique em Generate new token e depois em Generate new token classic
7. Insira sua senha e realize as configurações do token
8. Gere o token e **não saia da página antes de copiá-lo** em um bloco de notas.

**Clonando um repositório com token e salvando as configurações**

1. Abra o git Bach na pasta desejada
2. Digite  **git clone (mais a url copiada do repositório)**
3. Digite o nome do usuário
4. Cole o token
5. Digite **git config - -global credential.helper (mais cache = temporário) ou (mais store = permanente)**

**Visualizando todos os tokens criados**

1. No Github clique em usuário
2. Clique em Settings
3. Procure pela opção developer settings ( lateral esquerda no final da página)
4. Procure pela opção Personal acess tokens e depois tokens classic
5. Todos os tokens e datas de validades estarão nesta página

**Para verificar qual helper foi configurado**

1. Abra o git Bach na pasta desejada
2. Digite  **git config - -global credential.helper**

**Armazenamento de credenciais**

Para saber onde estão armazenadas as credenciais.

1. Abra o git bash na pasta desejada
2. Digite  **git config - -global - -show-origin credential.helper**
3. Digite **cat .gitconfig**
4. Digite **cat .git-credentials**

Usuários antigos do GitHub podem estar com as **credenciais de usuário e senha** salvas, será necessário deletar tais configurações para **usar o token.**

1. Na barra de pesquisa do Windows escrava  **credenciais.**
2. Acesse a pasta GERENCIAR CREDENCIAIS DO WINDOWS
3. Localize a url do github
4. Clique na seta a direita
5. remova o arquivo.

**Autenticação via CHAVE SSH**

A forma mais fácil de entender esta autenticação é por meio do exemplo de **chave e fechadura ou porta.**  Toda chave abre algo e toda a fechadura protege algo, logo **toda fechadura precisa de uma chave e toda chave de uma fechadura.**  Uma depende da outra.

**Verificando se há chaves SSH**

1. Clique com o botão direito do mouse na área de trabalho e acesse o git bach.
2. Digite **ls -al ~/.ssh**

**Criando uma chave SSH**

1. Abra o git bach clicando com o botão direito na área de trabalho
2. Digite **ssh-keygen -t ed25519 -C (mais o e-mail do github)**
3. Insira um local para salvar a chave (de enter e ele seleciona um local padrão)
4. Digite uma frase senha (passphase)

Vamos adicionar esta frase a um **Agente SSH**

1. Digite **eval “$(ssh-agent -s)”**
2. Digite **ssh-add ~/.ssh/id\_ed25519**
3. Digite sua passphase

**Adicionando nossa chave pública ao GitHub**

1. Na sua conta do GitHub entre em usuário depois settings
2. Procure pela opção **SSH and GPG Keys**
3. Clique em **new SSH key**
4. Título (nome da máquina)
5. Tipo da chave (Key type) = authentication Key
6. No git bach digite **cd ~/.ssh**
7. Digite **ls** (para ele listar)
8. Digite **cat (mais o endereço da chave desejada)**
9. Copie a chave e cole na página do github
10. Clique em adicionar e valide com a senha do seu github

**Clonando um repositório com SSH**

1. Escolha o repositório no github
2. Clique em **code**
3. Escolha SSH e copie a chave
4. Abra o git bach na pasta desejada
5. Digite  **git clone**