1. SHA1: Algoritmo de encriptação, que pega o seu arquivo e embaralha ele de uma forma específica.

A encriptação gera conjunto **único** de caracteres identificador de 40 dígitos e serve para fins de identificação.

Se tivéssemos gerado uma chave e entrássemos nesse arquivo mudando apenas 1 vírgula, ele geria uma nova chave de 40 dígitos diferente. Porém, se voltarmos no arquivo e colocarmos aquela vírgula inicial, é gerada novamente a chave de 40 dígitos inicial.

Sigla que significa Secure Hash Algorithm (Alroritmo de Hash Seguro) é um conjunto de funções hash criptográficas projetadas pela NSA (Agência de Segurança Nacional dos EUA).

Para gerar o conjunto identificador de 40 dígitos, deve-se abrir o GIT Bash na pasta em que o arquivo está e rodar o comando: **openssl sha1 nome-do-arquivo.extensão**

1. Objetos fundamentais:

3 tipos básicos de objetos do GIT responsáveis por versionar nosso código: BLOBS, TREES e COMMITS

**BLOBS** (bolha)**:** os arquivos ficam guardados dentro desse objeto que por sua vez contém **metadados** dentro de si.

* Tipo: Blob
* Tamanho
* \0
* Conteúdo do arquivo

Obter o SHA1 do arquivo usando GIT: **git hash-object** **--stdin** (especificação que não é um arquivo, pois a função espera um arquivo

**TREES:** armazenam **BLOBS**. Também contém **metadados** dentro de si. Responsável por montar toda a estrutura de onde está localizado o arquivo. Pode apontar para BLOBS ou para outras TREES (um diretório pode existir dentro de outro diretório)

* Tipo: Tree
* Tamanho
* \0
* BLOB
* SHA1 do BLOB
* Nome do arquivo

As árvores também têm um SHA1 dos seus **metadados**. Ao alterar um arquivo de uma bolha, será alterado o SHA do seu BLOB e consequentemente o SHA da sua árvore

**COMMITS:** objeto que junta todos os outros dando sentido (autor e mensagem) para a alteração feita.

* Tree
* Parente (último commit realizado antes dele)
* Autor
* Mensagem
* Timestamp (carimbo de tempo)

O commit também possui um SHA1 que é o hash de toda informação. Portanto, se você alterar um arquivo dentro de uma BLOB, alterará o SHA1 da BLOB, da TREE e do COMMIT

Dessa forma, é montada uma **linha do tempo** a partir dos Commits, trazendo a segurança para identificar quem fez e o que fez (sistema distribuído seguro)

1. **Chave SSH:** forma de estabelecer uma conexão segura entre duas máquinas gerando uma chave pública e uma privada para autenticação da sua conta do GitHub.

Para gerar essa chave, devemos seguir os seguintes passos:

**1º passo (gerar as chaves e autenticar):**

* No terminal do Git BASH, digite: **ssh-keygen -t ed25519 -C seu-email@gmail.com**;
* Digite o nome do arquivo em que você deseja salvar as chaves: **id\_rsa**;
* Digite a senha: **a seu critério**;
* Chaves geradas.
* Entre na pasta em que estão as chaves: **cd /c/Users/Eduardo/**. Nesse caminho, crie uma pasta com o nome **.ssh**, recorte as chaves para dentro dessa pasta e renomeie as duas como **id\_ed25519**;
* Entre na nova pasta em que estão as chaves: **cd /c/Users/Eduardo/.ssh**;
* Usar o comando **ls** para listar as chaves;
* Use o comando **cat id\_ed25519.pub** para visualizar o conteúdo da chave;
* Copie a chave que aparecer (Ex.: ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGCuVY/gQbKE4rbfJeaEO4dpycR5Ue7by4jUgaO/h8lU eduardo.nalmeida3@gmail.com), entre no GitHub, vá em Settings – SSH and GPG Keys – New SSH key – Digite um título **a seu critério** e cole a chave no campo **Key** – Add SSH key – preencher sua senha de login para autenticação;

**2º passo (Inicializar o ssh-agent):**

* Voltar ao terminal do Git BASH e usar o comando o **ls** para listar os arquivos;
* Usar o comando **pwd** para vermos o caminho em que estamos;
* Digite o comando: **eval $(ssh-agent -s)**;
* Digite o comando: **ssh-add id\_ed25519**;
* Digite a senha que você havia escolhido para as suas chaves;
* Identidade adicionada.

**\*** Quando se tem uma chave SSH configurada na sua máquina, você não pode simplesmente copiar o link do repositório e usar o **git clone** para clonar um repositório localmente. Nesse caso, ao entrarmos no repositório no GitHub e abrirmos o **dropdown <> Code** (botão verde no canto superior direito), devemos selecionar e copiar o caminho da aba **SSH**.

* Entrar na pasta em que queremos que o repositório local fique;
* Abrir um terminal do Git BASH nessa pasta;
* Rodar o comando: **git clone caminho-ssh-copiado-do-github**;
* Ao clonar um repositório dessa forma pela primeira vez, uma mensagem aparecerá. Digite **yes**;
* GitHub adicionado à lista de hosts conhecidos e repositório clonado.

1. **Token de acesso pessoal:** é gerado um token no GitHub que você **guarda** na sua máquina e sempre que você for fazer um commit, o git vai solicitar seu usuário e sua senha. Você preencherá o usuário de acordo com suas informações, porém **na senha, você digitará seu token de acesso pessoal**

* Acessar o GitHub no navegador – Settings – Developer settings – Personal access tokens – Generate new token;
* Preencher o campo **Note** com uma descrição da utilidade daquele token (**a seu critério**);
* É possível configurar uma **data de expiração** para esse token (7 dias – sem expiração);
* Marcar a opção **repo**;
* Descer toda a tela e clicar no botão **Generate token** (botão verde);
* Token gerado;
* Copie o token e salve em algum arquivo seguro da sua máquina em que você não o perderá.

**\*** Quando se tem um token de acesso pessoal, você deve acessar o repositório no GitHub, abrir o **dropdown <> Code** (botão verde no canto superior direito), selecionar e copiar o caminho da aba **HTTPS**

* Entrar na pasta em que queremos que o repositório local fique;
* Abrir um terminal do Git BASH nessa pasta;
* Rodar o comando: **git clone caminho-https-copiado-do-github**;
* Uma janela **Connect to GitHub** aparecerá solicitando seu token, digite-o e clique em **Sign in**;
* Repositório clonado.