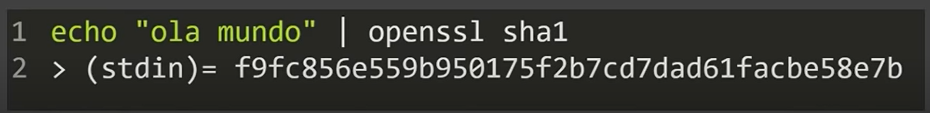
Funcionamento do git

### SHA1

A sigla SHA significa Secure hash algorithm (Algoritmo de Hash seguro), é um conjunto de funções hash criptográficas projetadas pela NSA (Agência de segurança nacional dos EUA).

A encriptação gera um conjunto de Characters identificador de 40 dígitos.

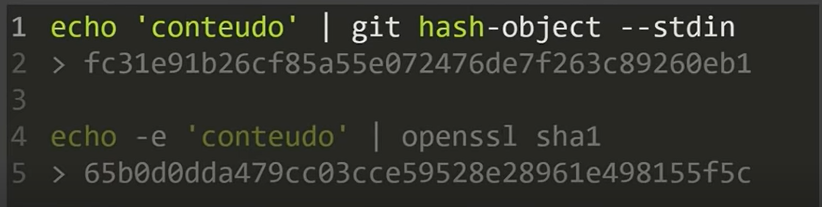
É uma forma curta de representar um arquivo.



(stdin) = Conjunto de 40 Characters.

Objetos internos do Git

### Objetos fundamentais (Blobs)



Nós vamos usar novamente a função **Echo** que vai pegar uma string e vai enviar o output que é o (**conteudo**) nesse caso, e a gente vai passar esse output para uma função do git que é **hash**-object.

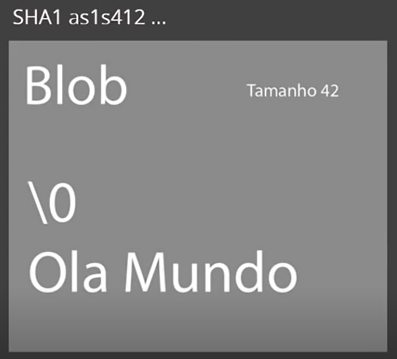
Esse flag que colocamos (--stdin) é apenas porque ela vai receber um arquivo do tipo texto.

Então se a gente passar essa string (**conteudo**) para a função do git então ela vai devolver o SHA1 desse (**conteudo**)

SHA1 : fc31e91b26cf85a55e072476de7f263c89260eb1



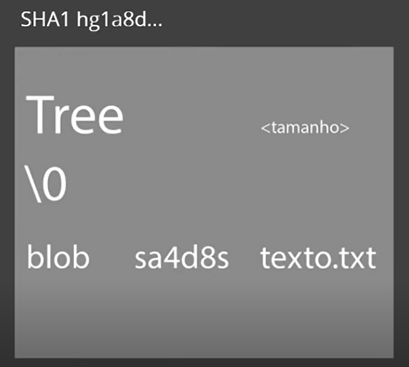
Usamos novamente a função **Echo** e passamos novamente astring (**conteudo**) e jogamos ela para ser encriptada pelo SHA1, só que é gerado um outro conjunto de characters. (e porque é diferente ?)



Isso acontece porque o jeito que o git manipula esses arquivos é através de objetos, esses arquivos ficam armazenados no objeto chamado (Blob) e este objeto contém metadados então o objeto (Blob) vai conter o tipo do objeto que é o (Blob), o tamanho dessa string /arquivo, uma \0 e o conteúdo desse arquivo.



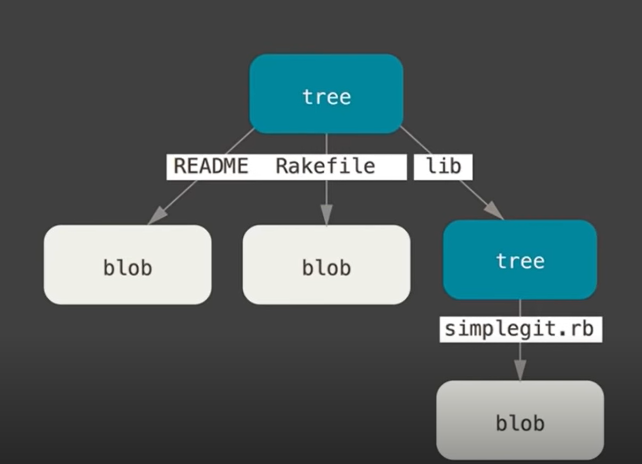
### Objetos fundamentais (Tree)



As “Tree” armazenam “Blobs”, então é uma crescente a “Blob” sendo o bloco básico de composição a “Tree” armazenando e apontando para tipos de “blobs” diferentes, então o “Tree” também contém metadados… ele também possui a “\0” ela aponta para um “blob” que por sua vez tem um “sha1” e a árvore também guarda o nome desse arquivo … então, se repararmos, o “blob” não guarda do arquivo ele só guarda o “sha1” do arquivo, que são os caracteres identificadores dele, e só.

A “Tree” vai ser responsável por montar toda a estrutura e onde estão localizados os arquivos, então as “Tree” podem apontar tanto para “blob” ou outras “Tree”.

Então, faz sentido que o GIT use um tipo de objeto que é recursivo, que são as "Tree''. Então uma “Tree” pode apontar para outra “Tree”, ou um diretório pode apontar para outro diretório.



### 

### 

### 

### 

### Objetos fundamentais (Commit)



E por fim, temos o objeto mais importante de todos, que é o “commit”. O “commit” é o objeto que vai juntar tudo, que vai dar sentido para a alteração que nós vamos estar fazendo.

Então o “commit” aponta para uma “Tree”, o “commit” aponta para um parente, ou seja, ele aponta para o último “commit” realizado antes dele. o “commit” aponta para uma autor, o “commit” aponta para uma mensagem e esse objeto leva a data e hora de quando foi criado.

E os “commits” também possui um “sha1”, possui encriptação, que é a geração desse (**Hash**) de 40 caracteres identificador, do seus metadados.