# Repositório

* Onde o código será armazenado.
* Na maioria das vezes cada projeto tem um repositório.
* Quando criamos um repositório estamos iniciando um projeto.

# criando repositórios

* Para criar um repositório utilizamos o comando: **git init**.
* Após este comando o diretório atual será reconhecido pelo git como um projeto e responderá aos seus demais comandos.

# github

* É um serviço para gerenciar repositórios.
* Podemos enviar nossos projetos para o GitHub e disponibilizá-lo para outros devs.
* O GitHub é gratuito tanto para projetos públicos como privados.

# enviar repositórios para o github

* Precisamos criar o projeto no GitHub, inicializar o mesmo no git em nossa máquina, sincronizar com o GitHub e enviar.

# verificando mudanças do projeto

* AS mudanças do projeto podem ser verificadas pelo comando: **git status**.
* Através desse comando todas as alterações do projeto são mapeadas.
* O que existe no **git status** é tudo o que não existem no GitHub.

# adicionando arquivos ao projeto

* Para adicionar arquivos novos a um projeto utilizamos o comando: **git add**.
* Podemos adicionar um arquivo específico como também diversos arquivos de uma vez.
* Somente adicionando arquivos eles serão monitorados pelo git.
* Arquivos não adicionados, não estarão no controle de versão.

# salvando alterações do projeto

* As alterações do projeto são realizadas pelo comando: **git commit**.
* Podemos commitar arquivos específicos ou vários de uma vez com a flag **–a**.
* É uma boa prática enviar uma mensagem a cada commit, com as alterações que foram feitas.
* A mensagem pode ser adicionada com a flag **–m**.

# enviando código ao repositório remoto

* O envio de arquivos commitados é feito por meio do comando: **git push**.
* Após essa ação o código do servidor será atualizado baseando-se no código local enviado.

# recebendo alterações

* Geralmente é necessário sincronizar o repositório local com as mudanças do remoto.
* Esta ação é feita pelo comando **git pull**.
* Após o comando serão buscadas atualizações, e caso encontradas elas serão unidas ao código atual existente na nossa máquina.

# clonando repositórios

* O ato de baixar um repositório de um servidor remoto é chamado de clonar repositório.
* Para esta ação utilizamos o comando: **git clone**.

# removendo arquivos do repositórios

* Os arquivos podem ser deletados da monitoração do git.
* O comando para deletar é **git rm**.
* Após deletar um arquivo do git ele não terá mais suas atualizações consideradas pelo git.

# histórico de alterações

* Podemos acessar um log de modificações feitas no projeto.
* O comando para este recurso é **git log**.
* Este comando fornece uma informação dos commits realizados no projeto até então.

# renomeando arquivos

* Com o comando **git mv** podemos renomear um arquivos ou mover o mesmo para outra pasta.
* Esse comando faz com que este novo arquivo seja monitorado pelo git, e o arquivo anterior é excluído.

# desfazendo alterações

* O arquivo modificado pode ser retornado ao estado original.
* O comando utilizado é o **git checkout**.
* Após a utilização deste comando o arquivo sai do staged.
* Caso seja feita uma próxima alteração, ele entra em staged novamente.

# ignorando arquivos no projeto

* Para ignorar aquivos devemos inserir um arquivo chamado **.gitignore** na raiz do projeto.
* Neste arquivo podemos inserir todos os arquivos que não devem entrar no versionamento.
* Isso é útil para os arquivos gerados automaticamente ou arquivos que contêm informações sensíveis.

# desfazendo todas as alterações

* Com o comando **git reset** podemos resetar as mudanças feitas.
* Geralmente é utilizado com a flag **- -hard**.
* Todas as alterações commitadas e também as pendentes serão excluídas.