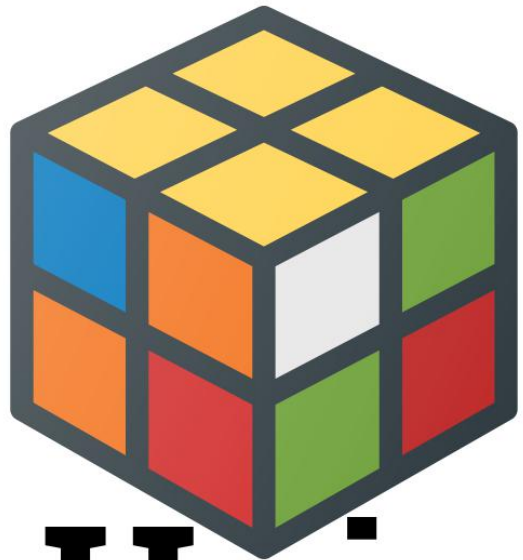
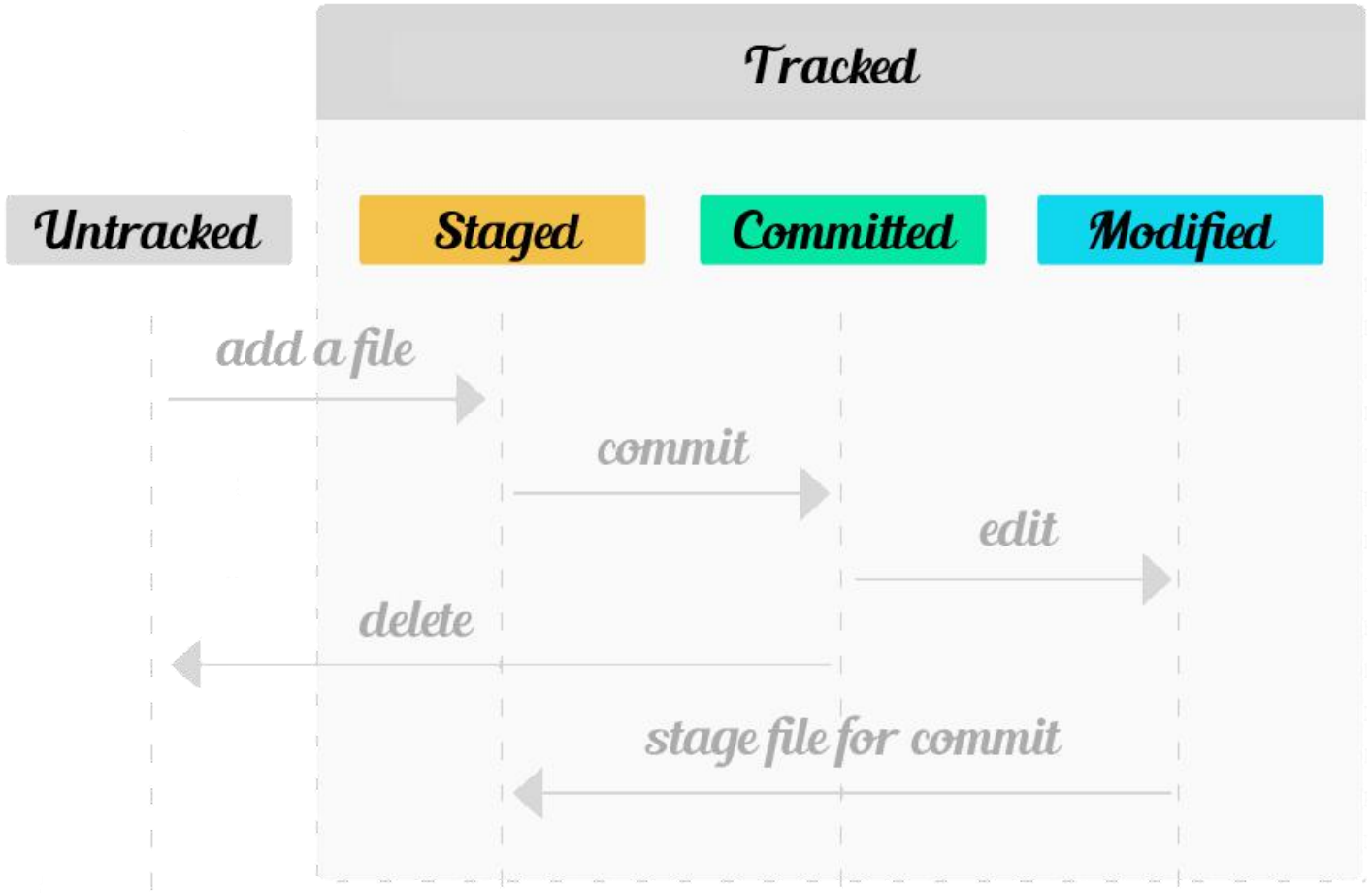


Geek University

Evolua seu lado geek!

www.geekuniversity.com.br

O ciclo de vida dos status de arquivos no Git



O ciclo de vida dos status de arquivos no Git

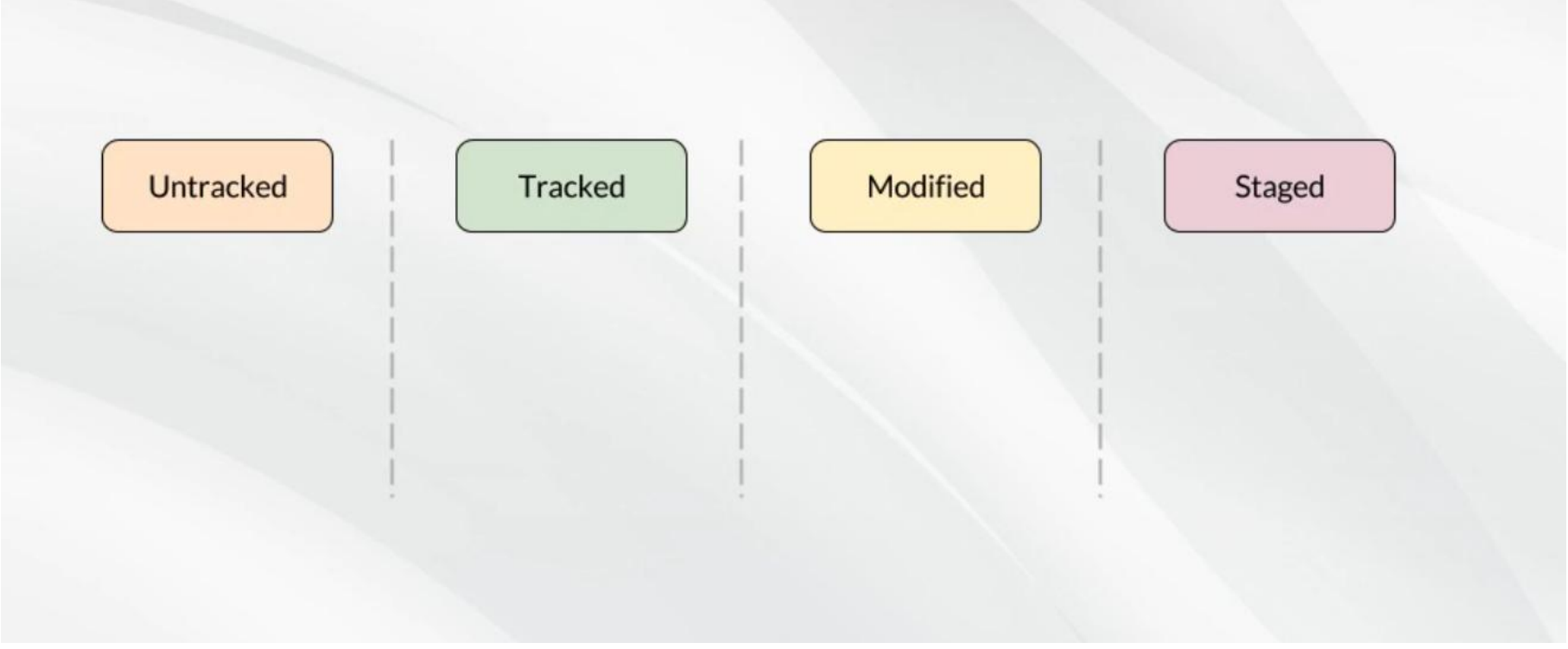
Vimos na aula passada que o git utiliza um poderoso sistema de monitoramento e controle de estágios/fases para os arquivos que fazem parte de um projeto. Este processo é conhecido como Ciclo de vida de status para os arquivos.

Nesta aula tudo ficará mais claro sobre como isto funciona.

O ciclo de vida dos status de arquivos no Git

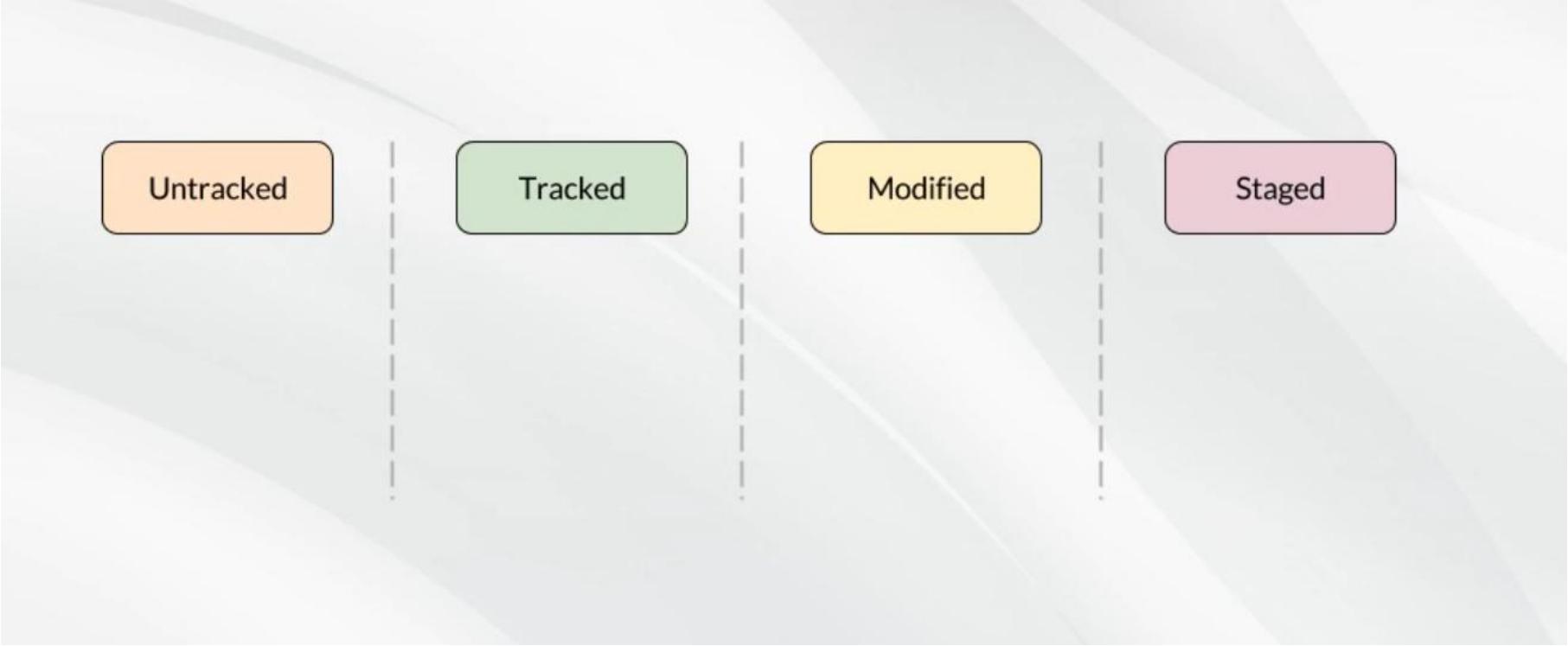
Vimos na aula passada que o git utiliza um poderoso sistema de monitoramento e controle de estágios/fases para os arquivos que fazem parte de um projeto. Este processo é conhecido como Ciclo de vida de status para os arquivos.

Nesta aula tudo ficará mais claro sobre como isto funciona.



O ciclo de vida dos status de arquivos no Git

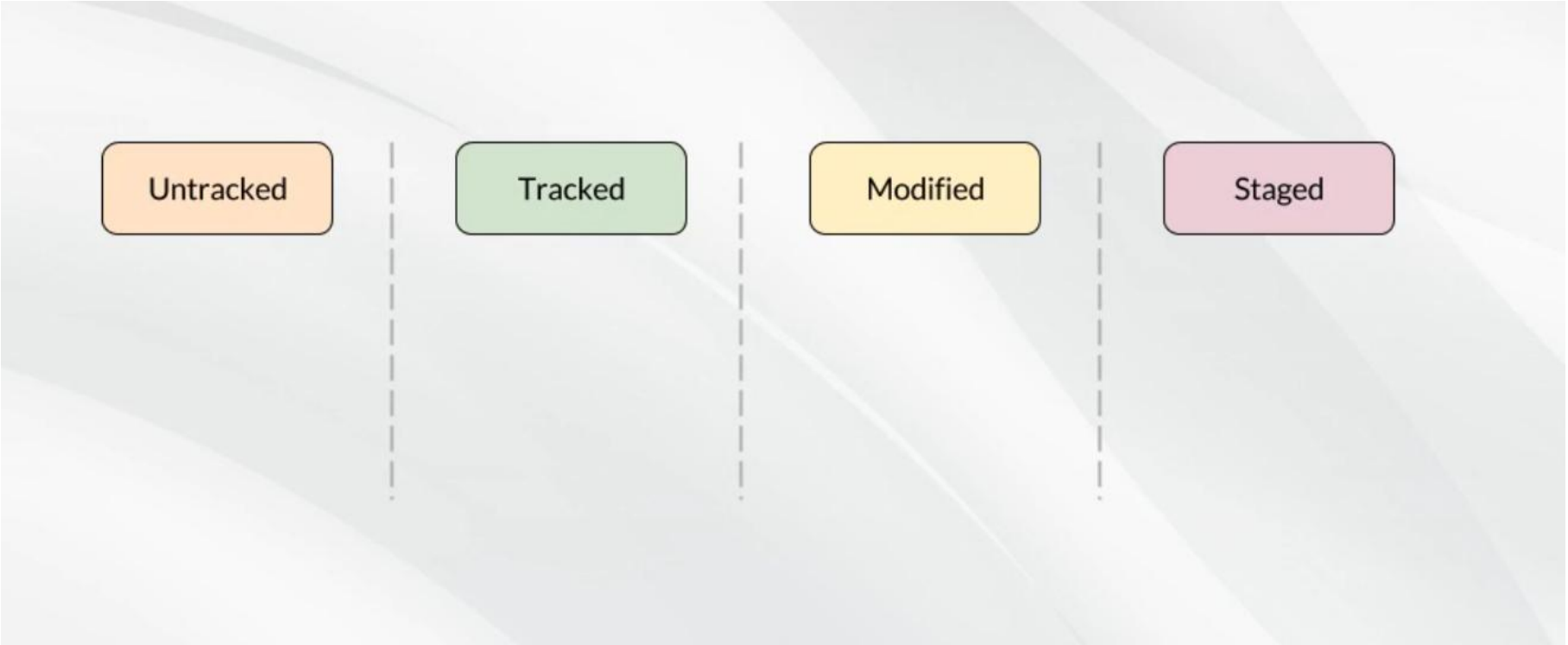
Temos 4 estágios possíveis em que um arquivo pode estar quando faz parte de um projeto:



Um arquivo pode ‘navegar’ por estes estágios por várias vezes durante o processo de desenvolvimento e de acordo com o gerenciamento feito destes arquivos.

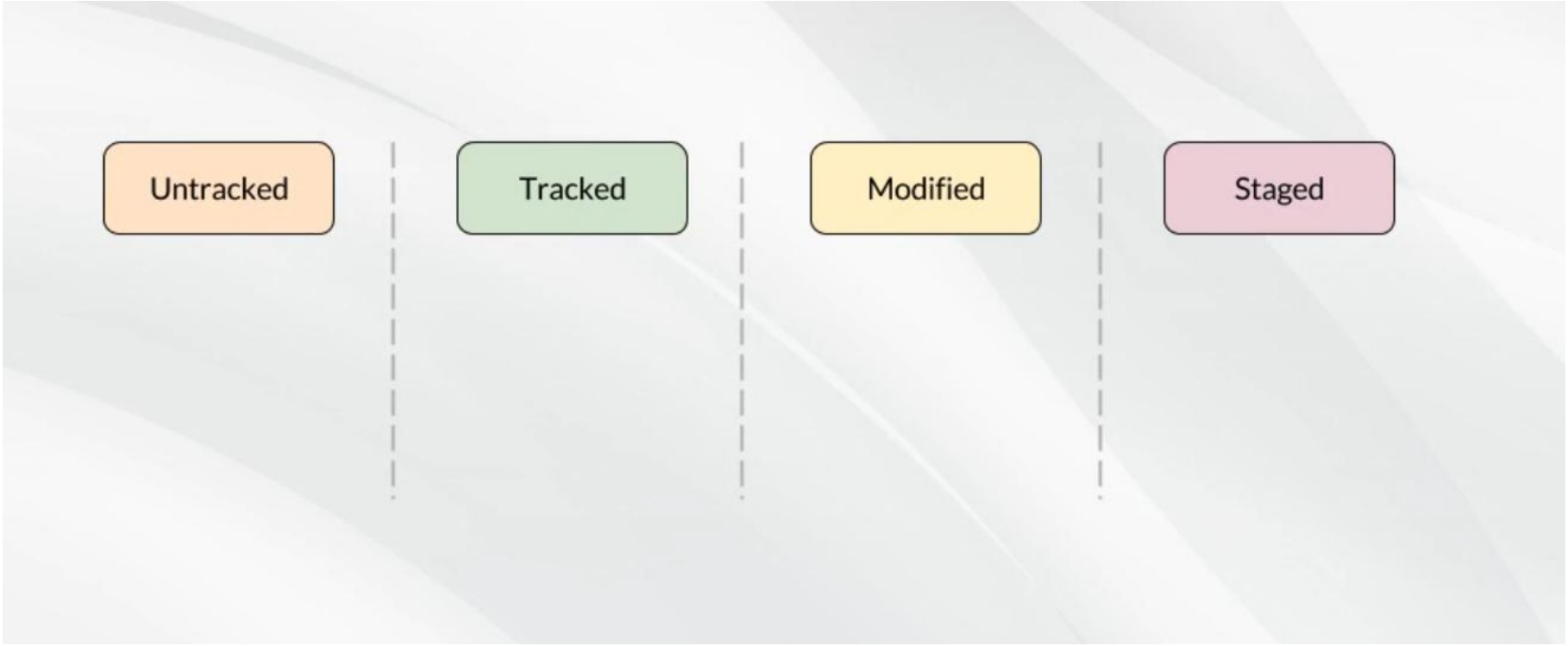
O ciclo de vida dos status de arquivos no Git

Quando um arquivo está em algum estado, nós dizemos que o arquivo está em um estado específico, por exemplo ‘o arquivo teste.py está *untracked’*



O ciclo de vida dos status de arquivos no Git

Entendendo os status



O ciclo de vida dos status de arquivos no Git

Entendendo os status

Quando usamos o git para controlar o versionamento do nosso projeto ou mesmo ao criar um novo arquivo em um projeto já gerenciado pelo git o/os arquivos iniciam seu status como ‘untracked’

Ou seja, se fossemos traduzir seria algo como ‘não ratreado’

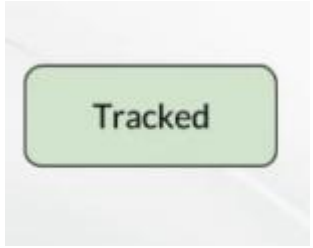
Isso significa que, apesar do arquivo fazer parte do projeto ele ainda não foi adicionado ao monitoramento do git.

Importante: Enquanto não houver o monitoramento do arquivo o git não estará controlando o versionamento deste arquivo. É como se o arquivo apesar de estar no projeto não faz parte do mesmo.

Pode ser que você, desenvolvedor, não queira que determinado arquivo seja monitorado. Mas existem soluções melhores para ignorar arquivos que aprenderemos ainda neste curso.

O ciclo de vida dos status de arquivos no Git

Entendendo os status



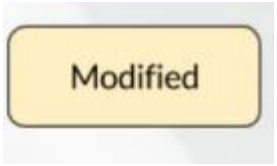
Quando adicionamos arquivos ao monitoramento do git, o status deste arquivo passa a ser ‘tracked’, ou seja, rastreado.

A partir de então o git passa a controlar as mudanças neste arquivo e todo o poder deste controlador de versão distribuído passa a fazer parte do seu projeto.

Os arquivos com status ‘tracked’ são conhecidos como ‘new file’ (novos arquivos)

O ciclo de vida dos status de arquivos no Git

Entendendo os status



Quando modificamos um arquivo que está sendo rastreado, seu status passa a ser ‘modified’, ou seja, ‘modificado’.

Desta forma sabemos facilmente se um arquivo não está sendo rastreado, ou está sendo rastreado ou mesmo se está sendo rastreado mas ocorreu modificações.

OBS: É nesta etapa que podem ocorrer conflitos, pois imaginando que você e outra pessoa baixaram uma cópia do mesmo arquivo e ambos o modificaram, de alguma forma o arquivo deve ser juntado ao final para gerar apenas 1 atualizado.

Esta é a mágica do merge que iremos aprender também neste curso.

O ciclo de vida dos status de arquivos no Git

Entendendo os status

Quando um arquivo está pronto, ou seja, ele está sendo rastreado e já foi motificado e finalizado, então está pronto para ser enviado para o repositório. Então o status passa a ser ‘staged’, que seria algo como ‘preparado’.

Os arquivos no estágio ‘staged’ são enviados para o repositório através de commits.

Iremos falar mais de commits em outra aula.

O ciclo de vida dos status de arquivos no Git

Fluxo de trabalho visual dos estágios de arquivos do git

O ciclo de vida dos status de arquivos no Git

Fluxo de trabalho visual dos estágios de arquivos do git

Imagine que você tenha um projeto, e decide, inteligentemente, gerenciar o versionamento dele com git.

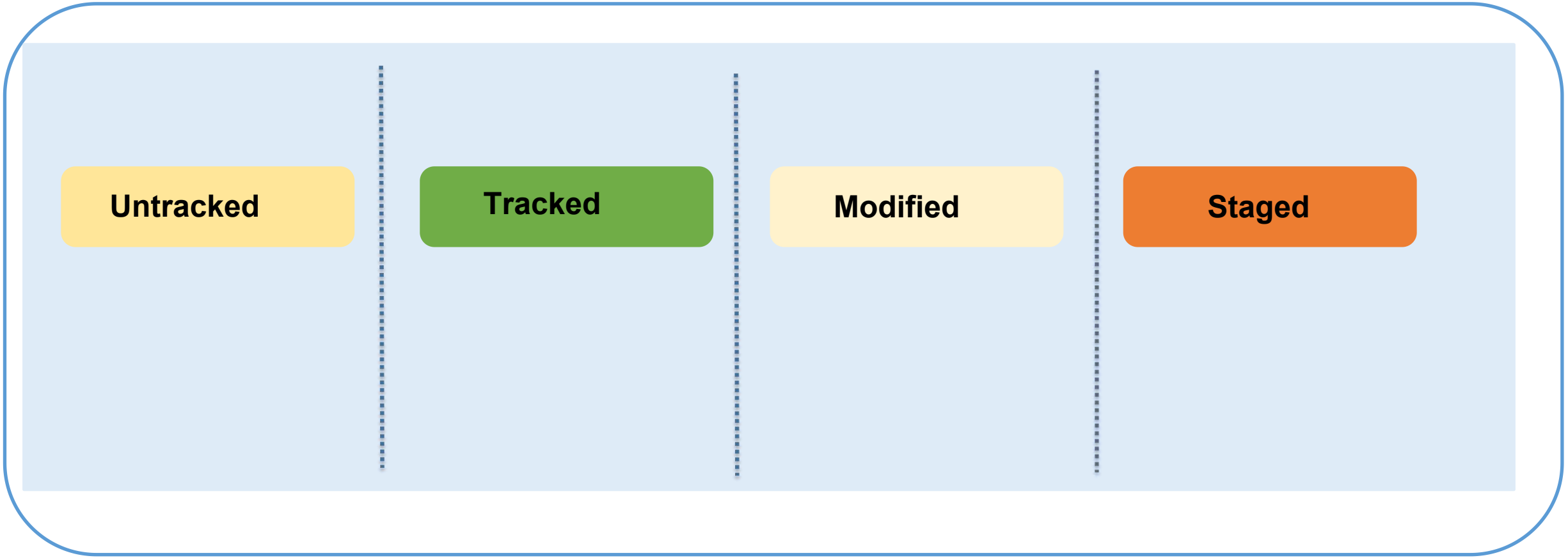
Usamos para isso o comando* na raiz do projeto:

git init

* Não se preocupe com os comandos ainda. Nem execute-os ainda.

O ciclo de vida dos status de arquivos no Git

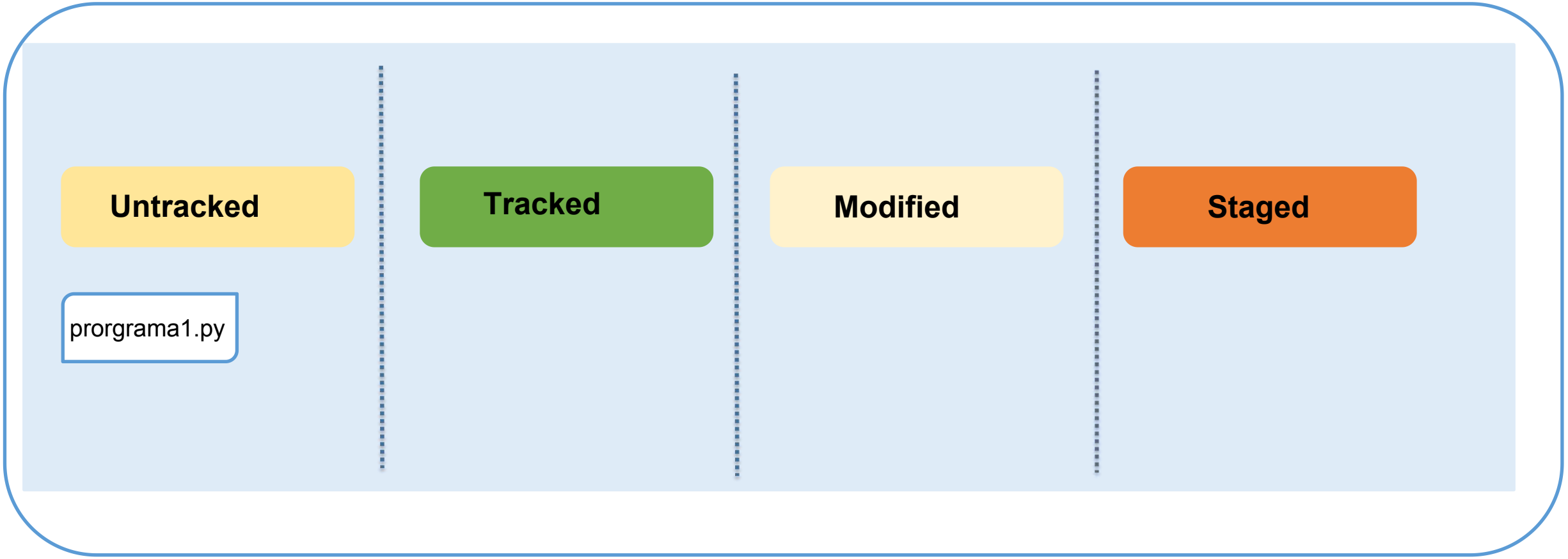
Fluxo de trabalho visual dos estágios de arquivos do git



A partir do comando anterior, o git passará a gerenciar e monitorar de forma inteligente o que fazemos no projeto.

O ciclo de vida dos status de arquivos no Git

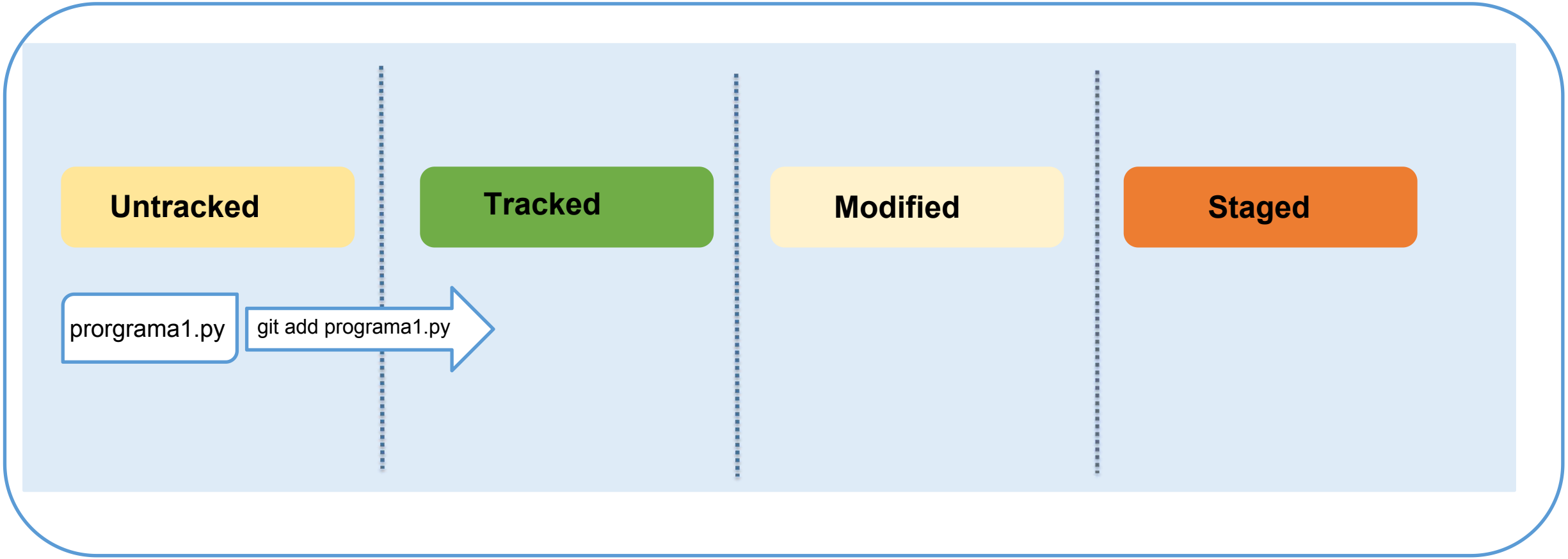
Fluxo de trabalho visual dos estágios de arquivos do git



Imagine que o arquivo programa1.py foi criado no nosso projeto. O primeiro estágio dele é ‘untracked’, ou seja, não rastreado pelo git. Isso significa que apesar do arquivo estar fisicamente no projeto ele não está sendo gerenciado pelo git.

O ciclo de vida dos status de arquivos no Git

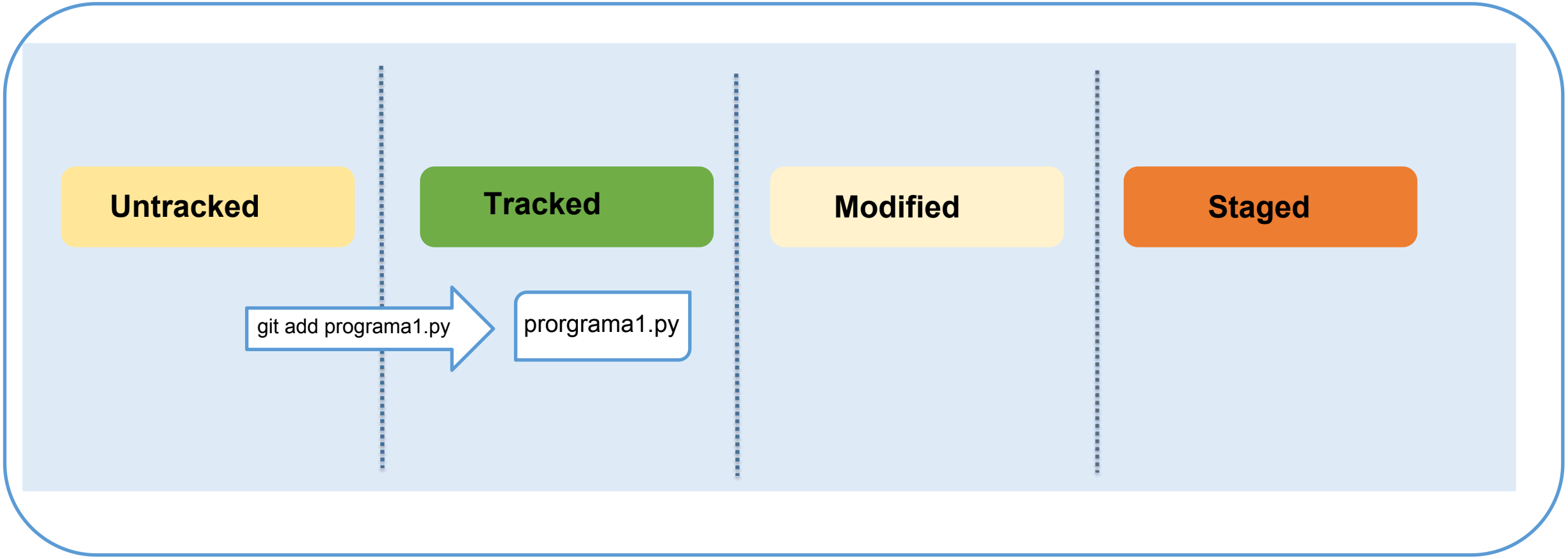
Fluxo de trabalho visual dos estágios de arquivos do git



Para que o arquivo possa ser rastreado usamos o comando ‘git add nome-do-arquivo’ e então seu status muda para ‘tracked’.

O ciclo de vida dos status de arquivos no Git

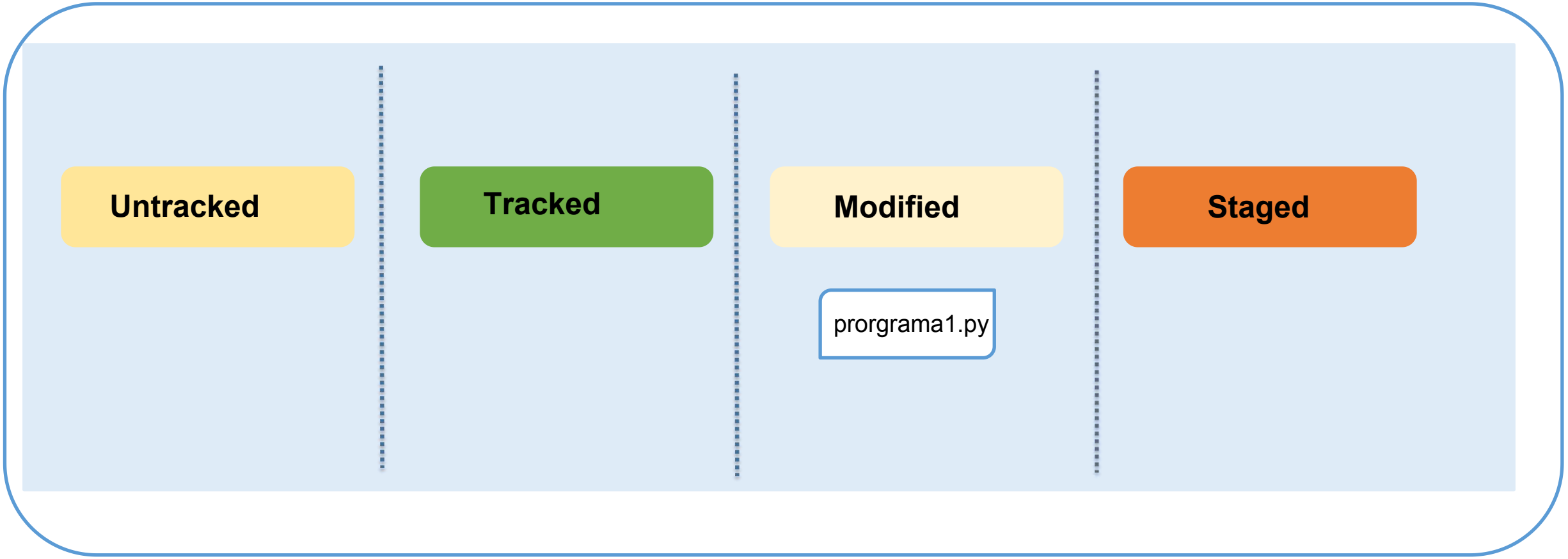
Fluxo de trabalho visual dos estágios de arquivos do git



Para que o arquivo possa ser rastreado usamos o comando ‘git add nome-do-arquivo’ e então seu status muda para ‘tracked’, como um ‘new file’, ou seja, novo arquivo sendo rastreado.

O ciclo de vida dos status de arquivos no Git

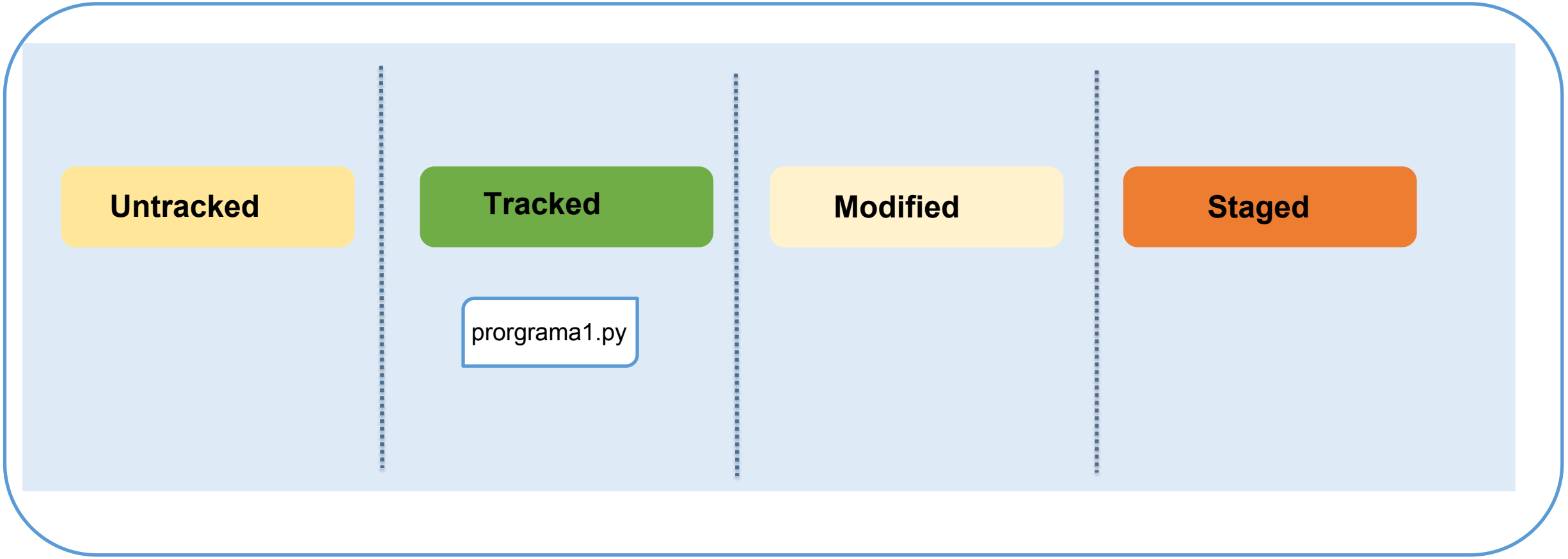
Fluxo de trabalho visual dos estágios de arquivos do git



Assim que alteramos um arquivo que estava ‘Tracked’ (rastreado) ele muda o status para ‘modified’ (modificado).

O ciclo de vida dos status de arquivos no Git

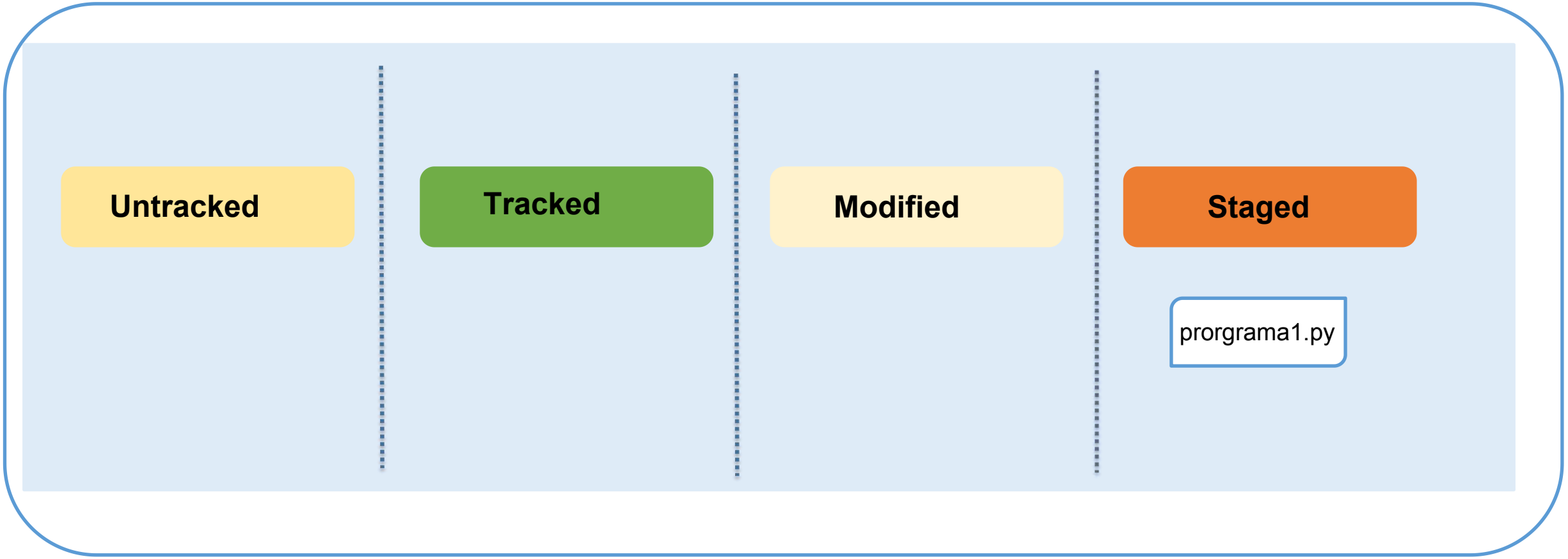
Fluxo de trabalho visual dos estágios de arquivos do git



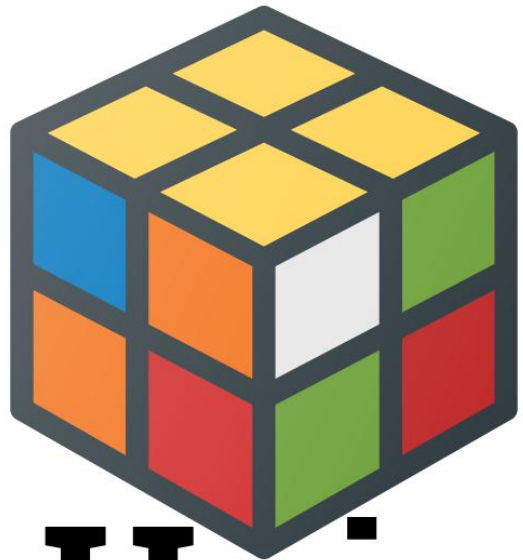
Para que este arquivo esteja pronto para ir para ‘staged’ precisamos adicionário ao status de ‘tracked’ novamente com o comando *git add nome-do-arquivo*, e o arquivo volta a estar rastreado como um ‘new file’.

O ciclo de vida dos status de arquivos no Git

Fluxo de trabalho visual dos estágios de arquivos do git



Quando terminamos alguma atividade (tarefa/etc) e os arquivos estão prontos, podemos enviá-los para 'staged' com o comando *git commit -m "Iformação importante sobre a tarefa realizada"*. Com o commit executado o código é enviado para o repositório local, ou seja, sua própria máquina, já que o git é descentralizado.



Geek University

Evolua seu lado geek!

www.geekuniversity.com.br