Removendo commit do repositório remoto

CRIANDO COMMITS E ENVIANDO PARA O REPOSITÓRIO REMOTO

No repositório de João criar uma nova branch e alternar para a mesma

```
$ git switch -c forgot-password
```

Gerar um novo commit

```
$ echo -e “function forgotPassword(){...}\nfunction sendMail(){...}” >> forgot-password.js
$ git add forgot-password.js
$ git commit -m "add feature for forgot password"
```

Avaliar o conteúdo do arquivo recém-criado

```
$ cat forgot-password.js
```

Nota: *a opção -e permite que o carácter de newline (\n) seja interpretado*

Gerar um segundo commit

```
$ echo “function sendNotification(){...}” >> forgot-password.js
$ git add forgot-password.js
$ git commit -m "enable notification on forgot password feature"
```

Enviar o commit para o repositório remoto

```
$ git push -u origin forgot-password
```

REESCREVENDO O HISTÓRICO PARA REMOVER O ÚLTIMO COMMIT

Verificar o histórico da branch atual

```
$ git hist
```

Remover o último commit que habilitou notificações

```
$ git reset --hard head~
```

Verificar o histórico da branch atual

```
$ git hist
```

ENVIANDO HISTÓRICO REESCRITO

Tentar enviar o histórico após a remoção do commit

```
$ git push
```

Nota: *O Git exibe que há commits no repositório remoto que não existem no repositório de João, solicitando uma atualização do repositório antes do push. Porém, tal commit é exatamente o que foi removido, se houver uma atualização do repositório o commit removido irá retornar. Nesse caso, o que realmente se deseja é a exclusão do commit no repositório remoto, então para isso é necessário forçar o envio do novo histórico*

Enviar o histórico de forma forçada

```
$ git push -f
```

Nota: *reescreve o histórico da branch forgot-password no repositório remoto considerando o histórico do repositório local de João*

Rebase

CRIANDO BRANCH LOCAL PARA A BRANCH REMOTA FORGOT PASSWORD

Atualizar o repositório de Maria

```
$ git fetch
```

Verificar todas as branches

```
$ git branch -a
```

Nota: *não há branch local para "forgot-password"*

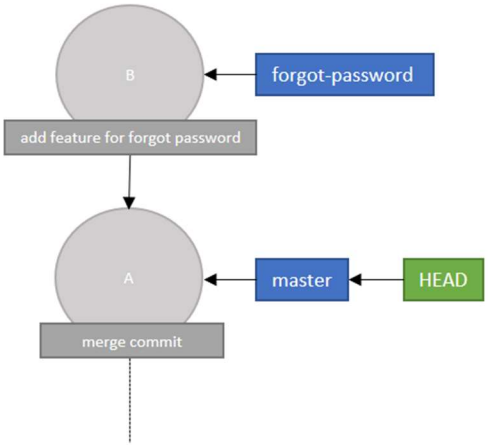
Para esta aula será necessário existir uma branch local desta branch

```
$ git branch forgot-password origin/forgot-password
```

Nota: *cria uma branch local forgot-password que rastreia origin/forgot-password*

Verificar o histórico da branch local forgot-password

```
$ git hist forgot-password
```



CRIANDO NOVA BRANCH NOTIFICATION E GERANDO COMMIT

Criar uma nova branch e alternar para a mesma

```
$ git switch -c notifications
```

Verificar o histórico

```
$ git hist
```

Nota: *como a branch notifications foi criada a partir da branch master, a mesma também não possui o commit "add feature for forgot password"*

Gerar um novo commit

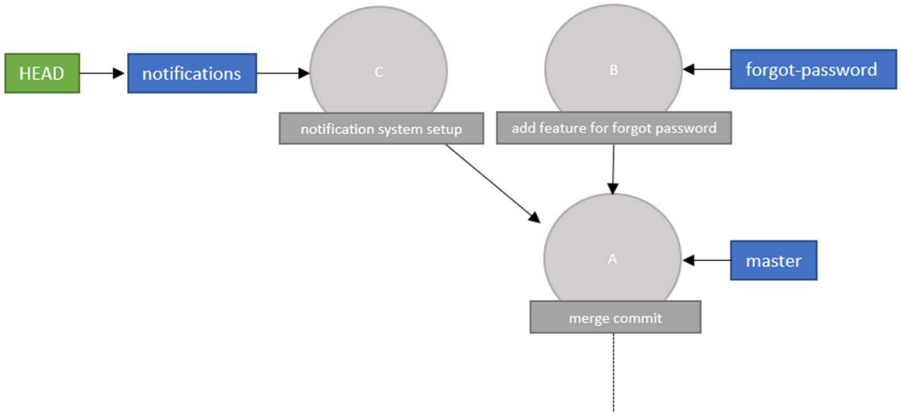
```
$ echo “function setup(){...}” >> notifications.js
$ git add notifications.js
$ git commit -m "notification system setup"
```

Comparar o histórico das branches notifications e forgot-password

```
$ git hist
$ git hist forgot-password
```

Copiar a hash do commit "notification system setup" e salvar no notepad

Nota: será utilizado na próxima aula



REALIZANDO REBASE

Realizar rebase

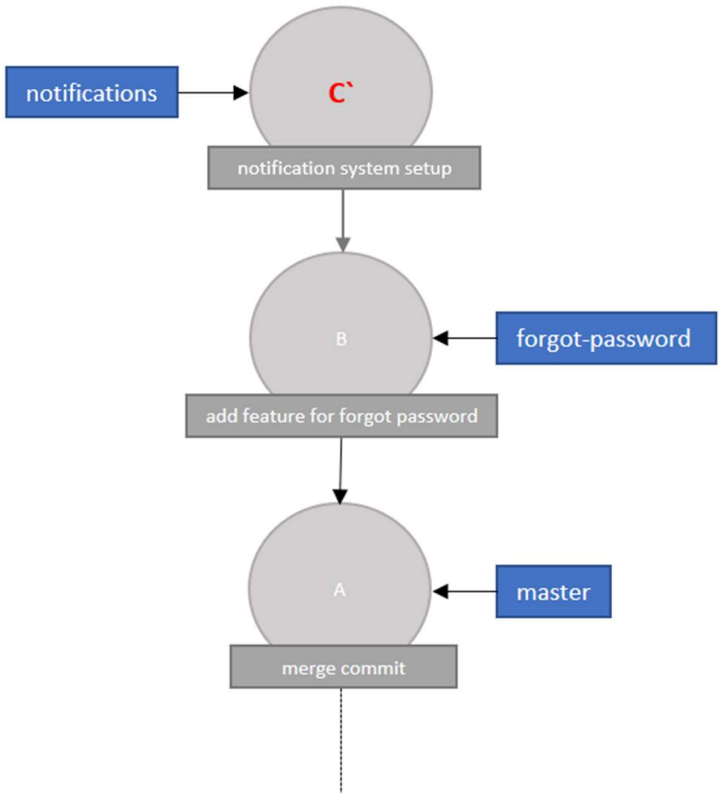
```
$ git rebase forgot-password
```

Nota: o rebase permite que você replique os commits de uma branch em cima de outra branch

Verificar o histórico da branch atual

```
$ git hist
```

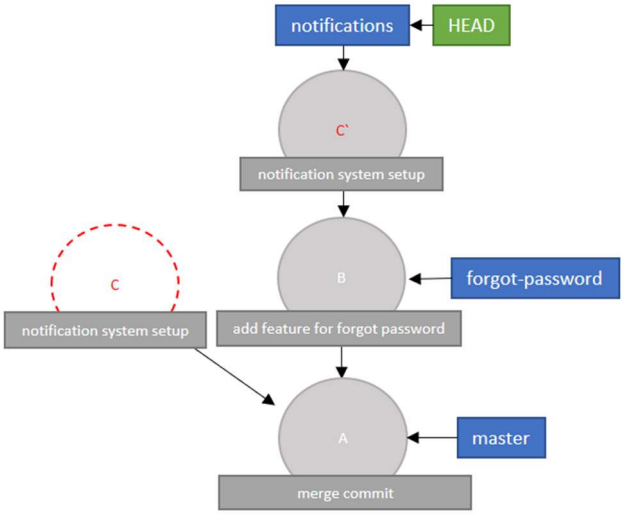
Nota: os commits da branch notifications foram recriados e inserido na topo de outra base (branch forgot-password), mantendo um histórico linear



The Golden Rule of Rebase

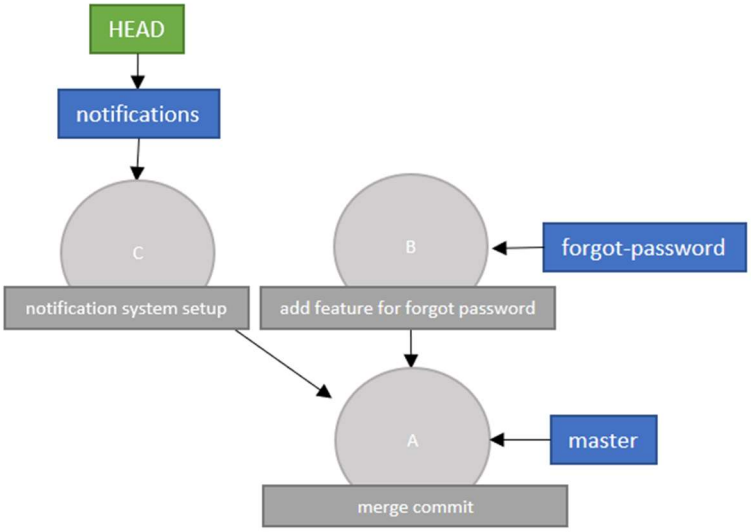
RETORNANDO AO ESTADO INICIAL ANTES DO REBASE

Estado atual do repositório de Maria:



No repositório de Maria, resetar para o commit "notification system setup" original (antes do rebase)

```
$ git reset --hard <hash do commit salvo no notepad na aula anterior>
```



SINCRONIZANDO REPOSITÓRIOS

No repositório de Maria, enviar a branch branch notifications para o repositório remoto

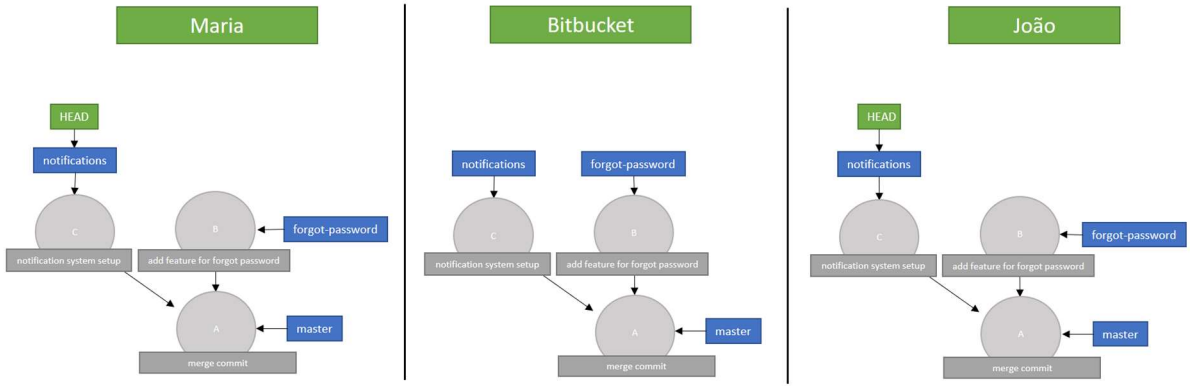
```
$ git push -u origin notifications
```

No repositório de João, baixar o conteúdo remoto (branch notifications)

```
$ git fetch
```

Criar uma branch local notifications que rastreia origin/notifications, já alternando para a mesma

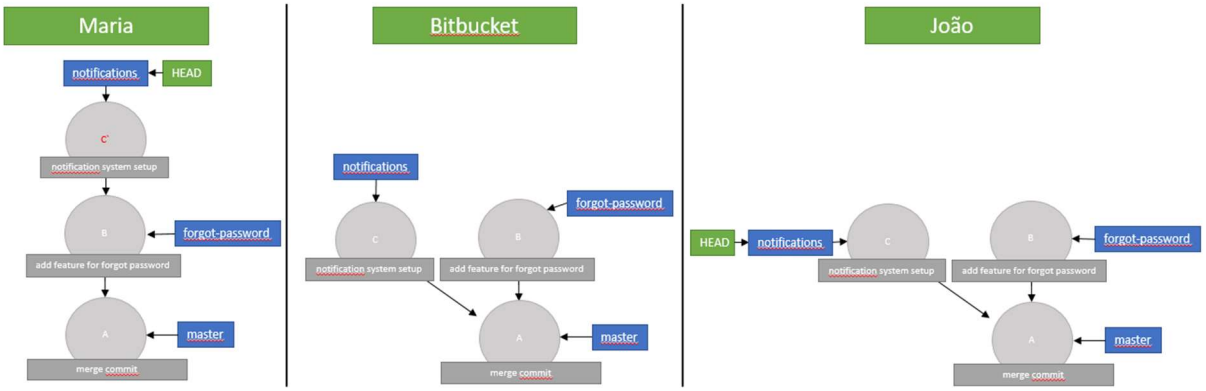
```
$ git switch -c notifications origin/notifications
```



INFRINGINDO A REGRA DE OURO DO REBASE

No repositório Maria, executar o rebase

```
$ git rebase forgot-password
```



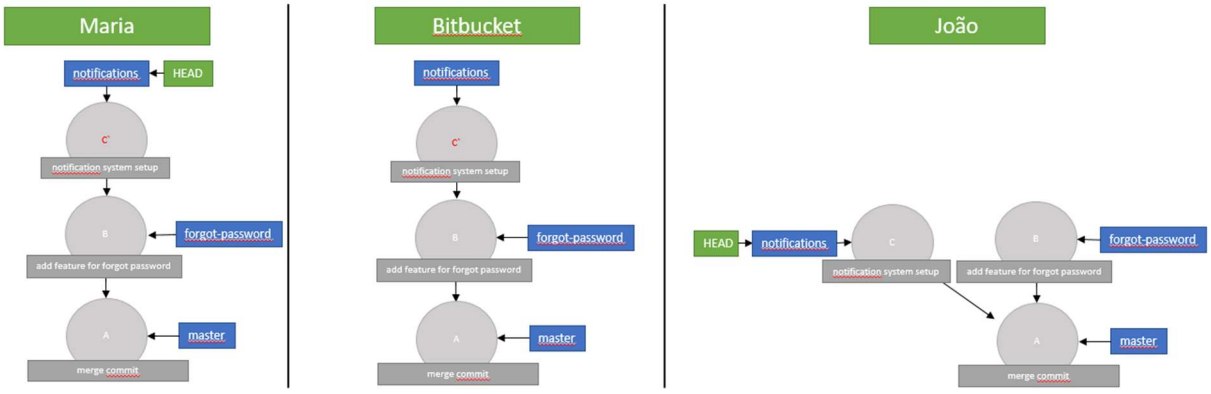
Tentar enviar o histórico da branch notifications ao repositório remoto

```
$ git push
```

Nota: *Houve um erro, pois o commit "notification system setup" com o primeiro identificador existe no repositório remoto e não existe neste repositório local. Neste caso, para alterar o commit existente no Bitbucket pelo novo criado pelo rebase deve-se forçar o envio do novo histórico*

Enviar o histórico de forma forçada

```
$ git push -f
```



ENTENDENDO OS PROBLEMAS GERADOS

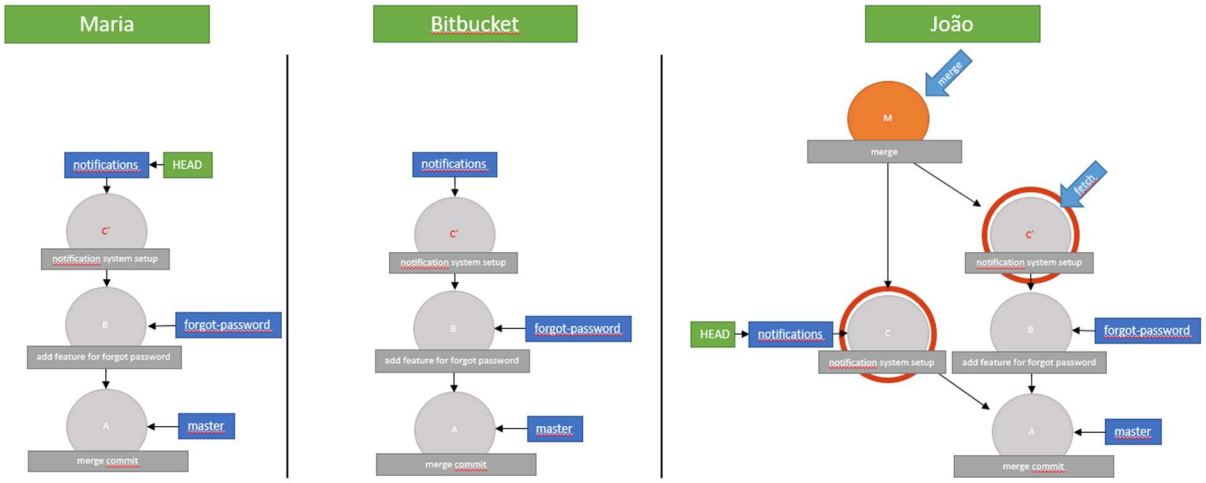
Acessar o repositório de João e atualizar

```
$ git pull
```

Verificar o histórico da branch atual (notifications)

```
$ git hist
```

Nota: *o grande benefício do rebase que é a linearidade do histórico não ocorreu.*
E ainda gerou commits duplicados no historico, que fazem as mesmas alterações, tornando o histórico mais confuso



REMOVENDO A BRANCH NOTIFICATIONS

No repositório de João, remover a branch notifications

```
$ git switch master
$ git branch -D notifications
```

Remover do repositório remoto

```
$ git push origin --delete notifications
```

No repositório de Maria, remover a branch notifications

```
$ git switch master
$ git branch -D notifications
```

Conflitos em rebase

CRIAR UM PULL REQUEST PARA A BRANCH MASTER

No repositório de João, criar uma nova branch e alterar para a mesma

```
$ git switch -c logs
```

Gerar dois novos commit

```
$ echo "function log(){...}" >> users.js
$ git add users.js
$ git commit -m "add log on users"
$ echo "function log(){...}" >> groups.js
$ git add groups.js
$ git commit -m "add log on groups"
```

Publicar a branch logs

```
$ git push -u origin logs
```

No Bitbucket criar um pull request da branch logs para a branch master, ***mas* não aprovar neste primeiro momento**

CRIANDO A BRANCH CACHE E GERANDO COMMITS

No repositório de Maria atualizar a branch master

```
$ git pull
```

Verificar o histórico da branch master

```
$ git hist
```

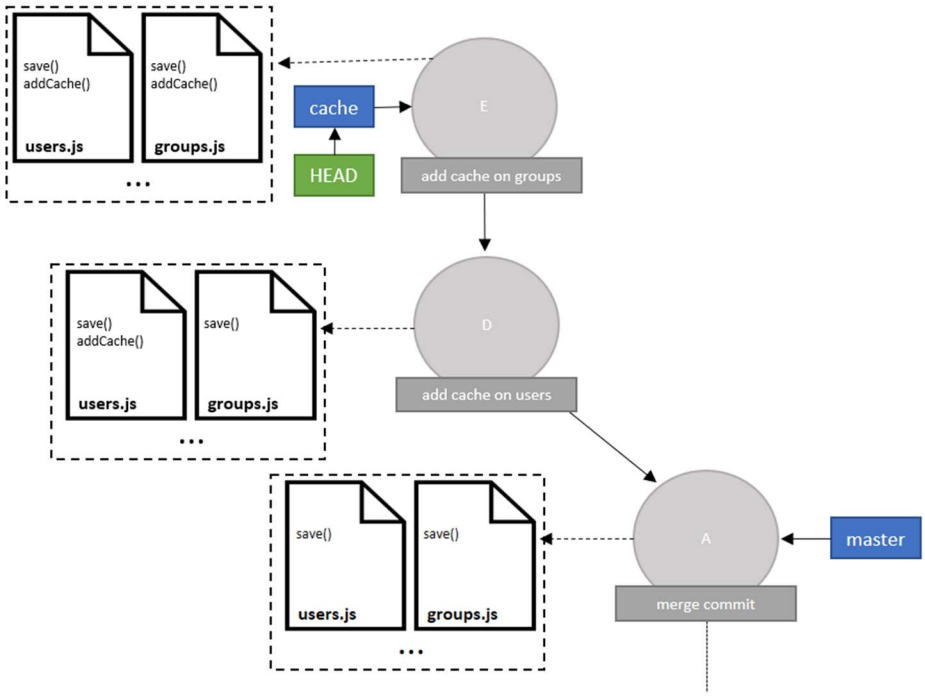
Nota: *como o pull request ainda não foi aprovado, as alterações anteriores ainda não foram incorporadas na branch master*

Criar uma nova branch e alterar para a mesma

```
$ git switch -c cache
```

Gerar dois novos commit

```
$ echo "function addCache(){...}" >> users.js
$ git add users.js
$ git commit -m "add cache on users"
$ echo "function addCache(){...}" >> groups.js
$ git add groups.js
$ git commit -m "add cache on groups"
```



APROVANDO PULL REQUEST ABERTO

No Bitbucket:

- Aprovar o pull request aberto
- Realizar o merge utilizando a estratégia de merge fast forward

ATUALIZANDO O REPOSITÓRIO DE MARIA

No repostório de Maria, alterar para a branch master

```
$ git switch master
```

Atualizar a branch master

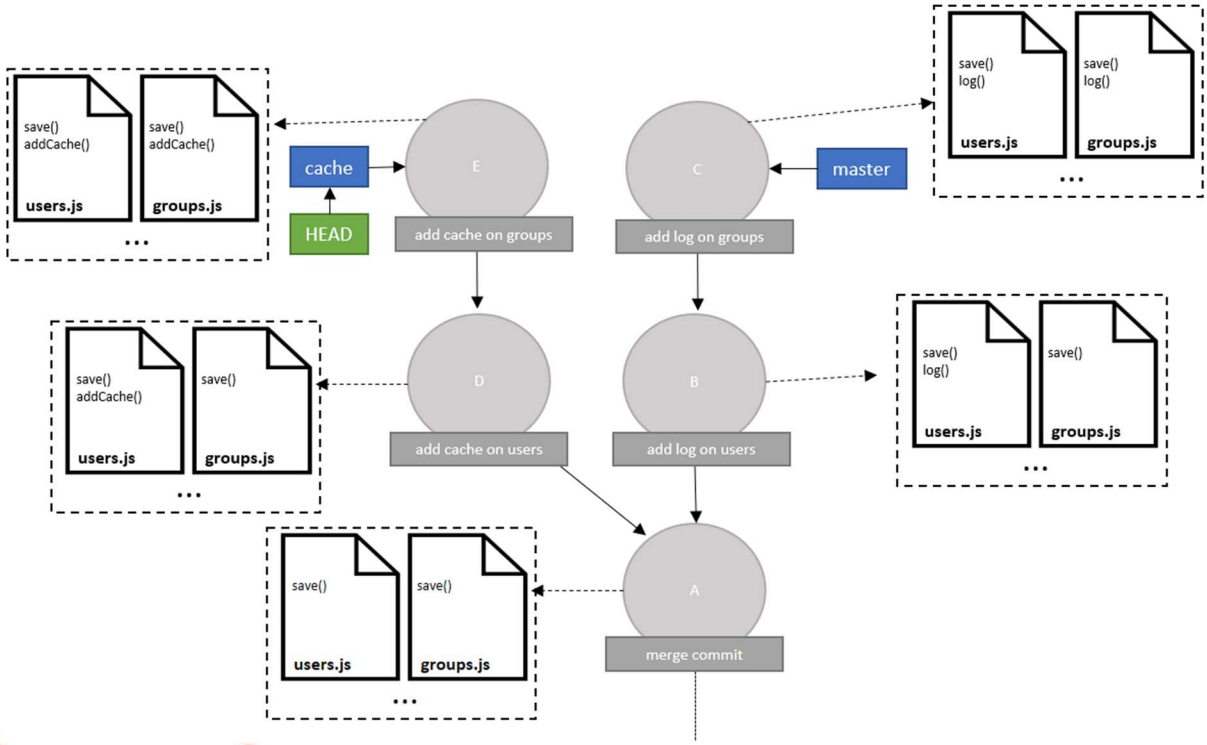
```
$ git pull
```

Retornar para a branch cache

```
$ git switch cache
```

Verificar o histórico das branches master e cache

```
$ git hist
$ git hist master
```



EXECUTANDO REBASE

Realizar rebase

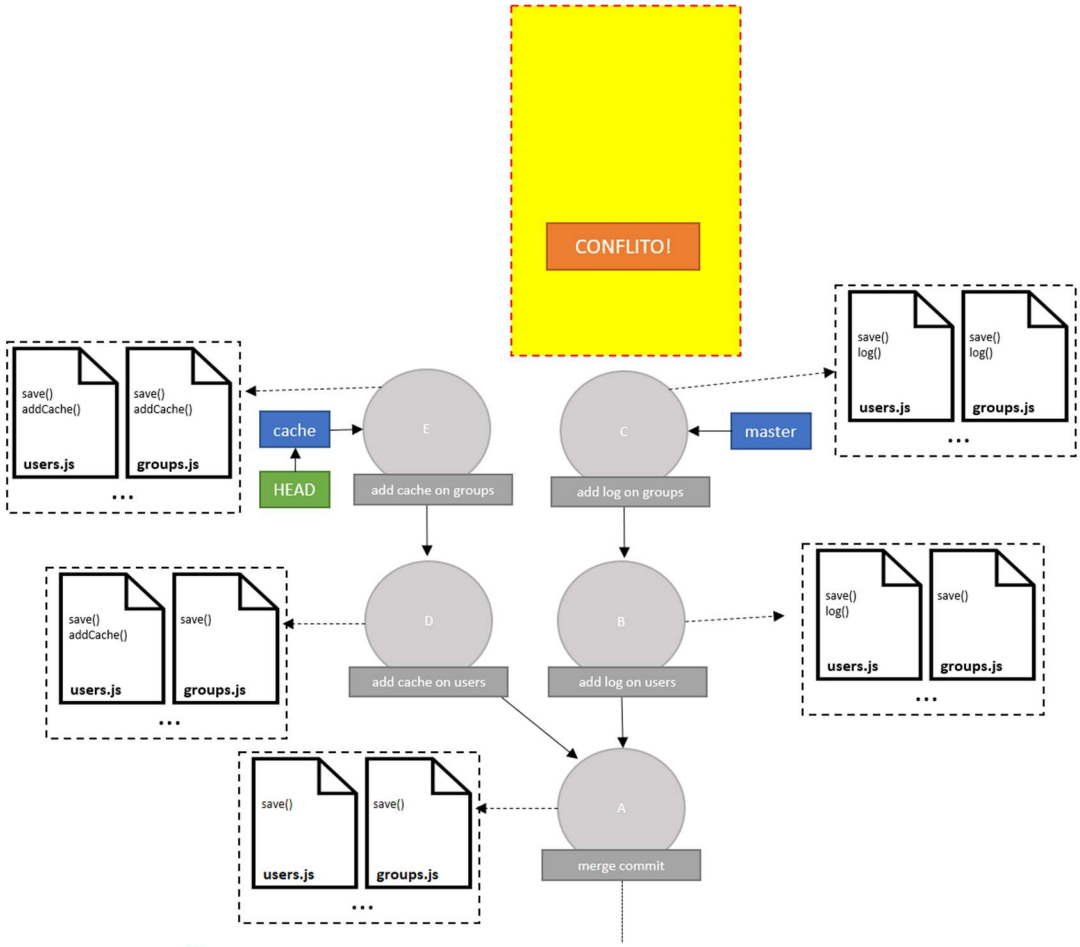
```
$ git rebase master
```

Nota: *tal rebase respeitou a regra de ouro do rebase, uma vez que os commits que serão recriados "add cache on groups" e "add cache on users" só existem no repositório de Maria*
Nota: *considere a seguinte mensagem no terminal: "(master|REBASE 1/2)"*

- *1 é a etapa atual*
- *2 é a quantidade de etapas desse rebase*
    - *Há dois commits "add cache on groups" e "add cache on users" que estão sendo recriados no topo da branch master*
    - *Para cada commit é uma etapa, que pode existir conflito ou não*

Para a primeira etapa o Git informa que houveram conflitos de mesclagem

Processo de rebase (etapa 1)



CORRIGINDO O PRIMEIRO CONFLITO

Abrir o editor de preferência para resolver os conflitos no arquivo users.js

Para resolver o conflito considere utilizar a opção "Acept both changes"

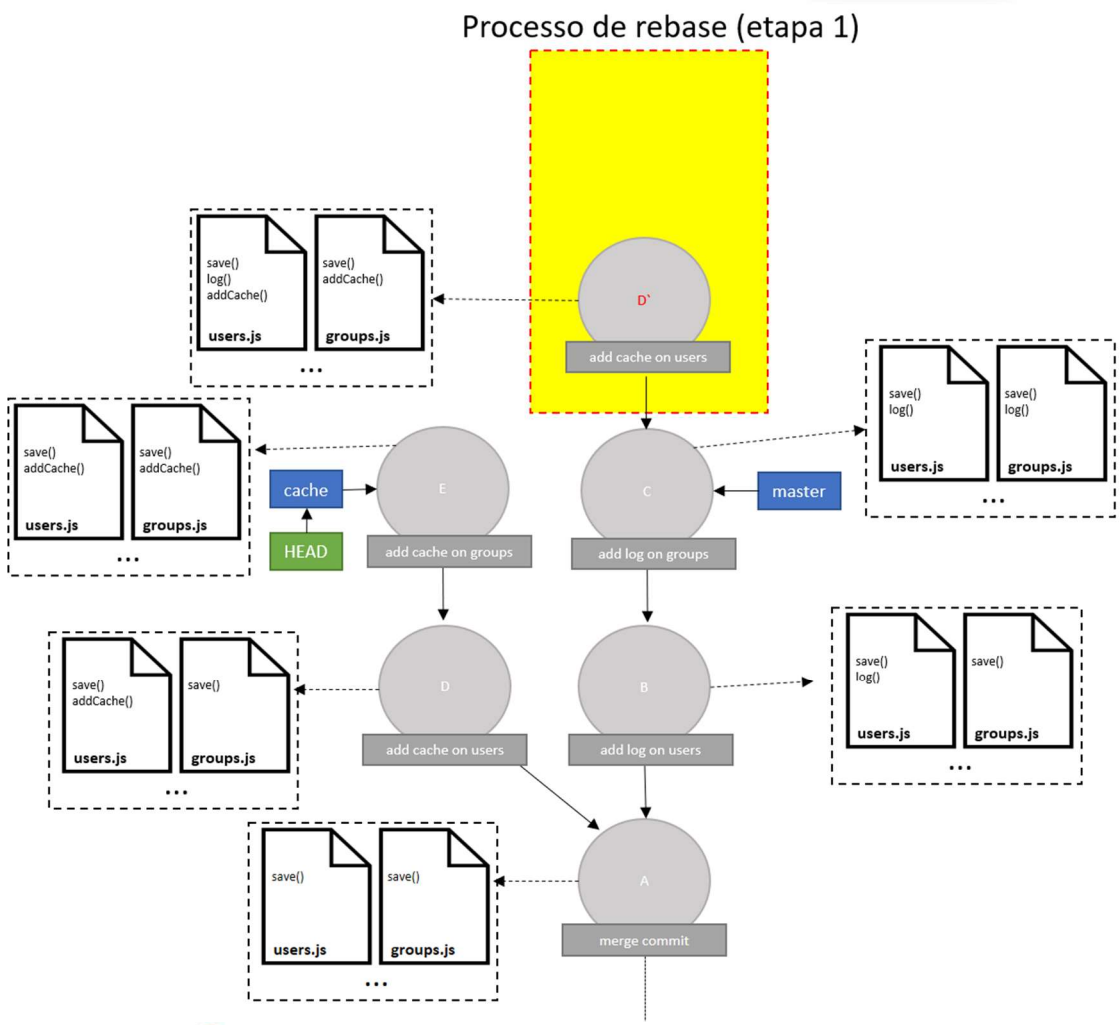
Confirmar a resolução de conflito do arquivo users.js

```
$ git add users.js
```

Concluir a etapa 1 do rebase (commit "add cache on users")

```
$ git rebase --continue
```

Nota: *manter a mensagem original do commit "add cache on users"*



CORRIGINDO O SEGUNDO E ÚLTIMO CONFLITO

Abrir o editor de preferência para resolver os conflitos no arquivo groups.js
Para resolver o conflito considere utilizar a opção "Acept both changes"

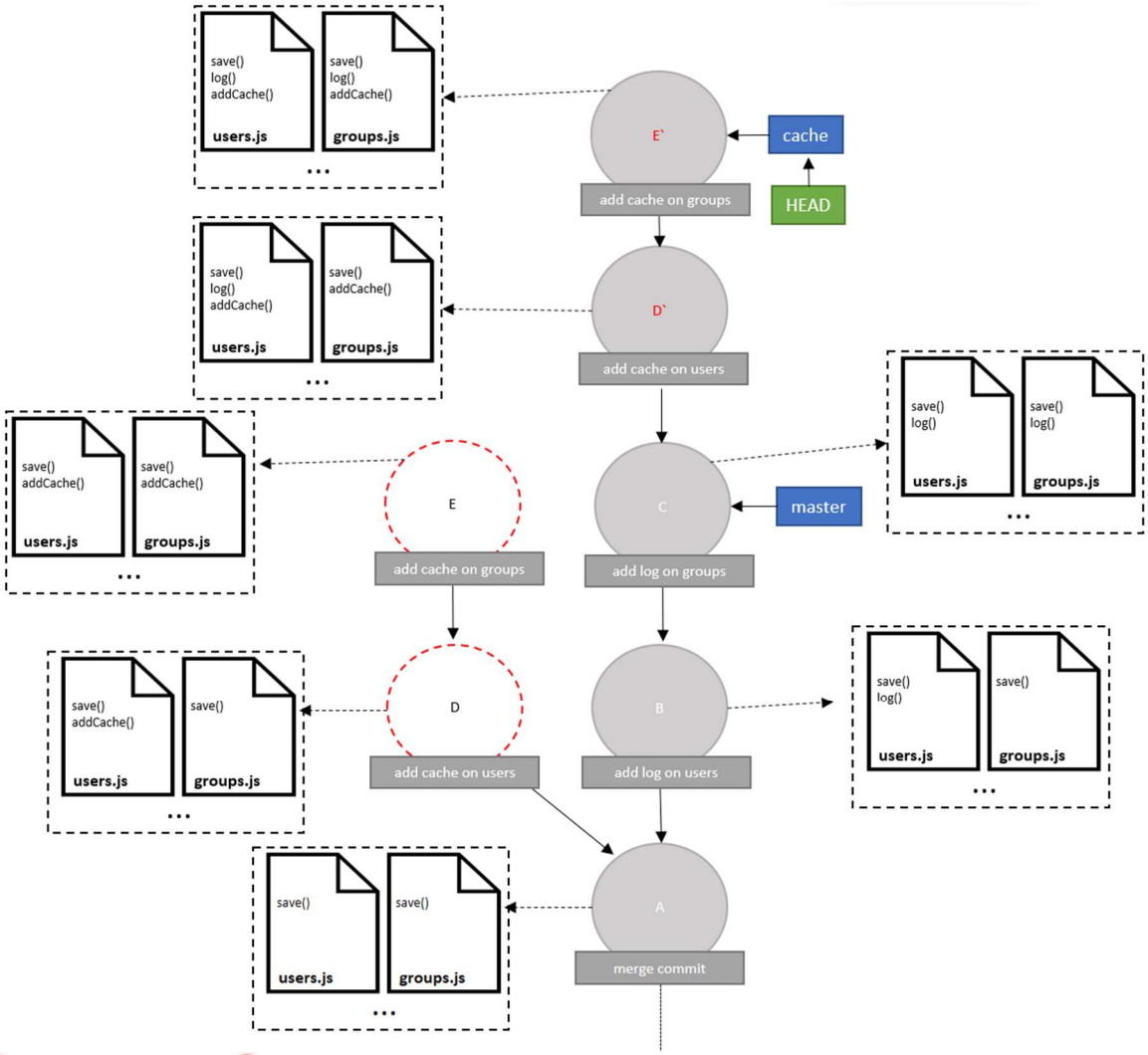
Confirmar a resolução de conflito do arquivo groups.js

```
$ git add groups.js
```

Concluir a etapa 2 do rebase (commit "add cache on groups")

```
$ git rebase --continue
```

Nota: *manter a mensagem original do commit "add cache on groups"*



INTRODUZINDO AS ALTERAÇÕES DA BRANCH CACHE DENTRO DA BRANCH MASTER ATRAVÉS DE PULL REQUEST

Publicar a nova branch cache no repositório remoto

```
$ git push -u origin cache
```

No Bitbucket:

- Criar um pull request da branch cache para a branch master
- Aprovar o pull request aberto
- Realizar o merge utilizando a estratégia de merge fast forward
- Verificar o histórico da branch master

ATUALIZANDO REPOSITÓRIOS LOCAIS

No repositório de João, Atualizar a branch master

```
$ git switch master
$ git pull
```

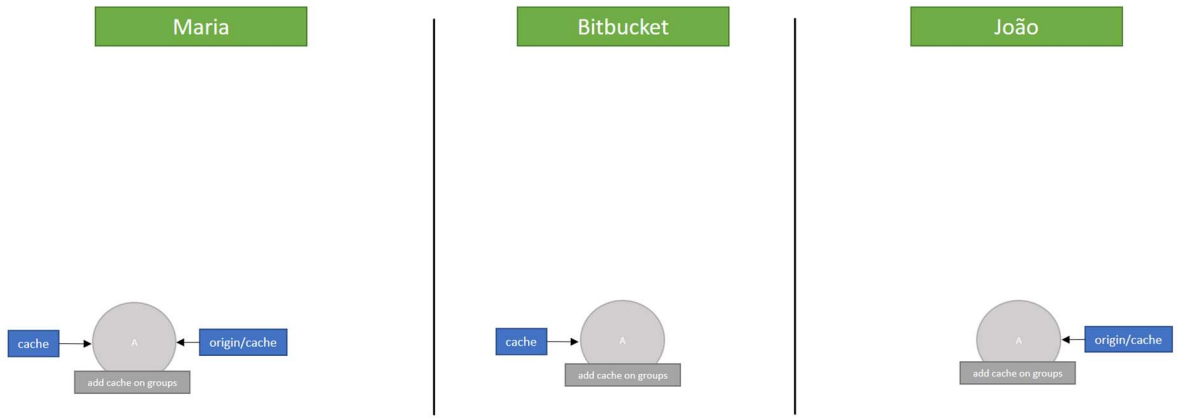
No repositório de Maria, Atualizar a branch master

```
$ git switch master
$ git pull
```

Pull com rebase

GERANDO COMMITS

Estado atual dos repositórios

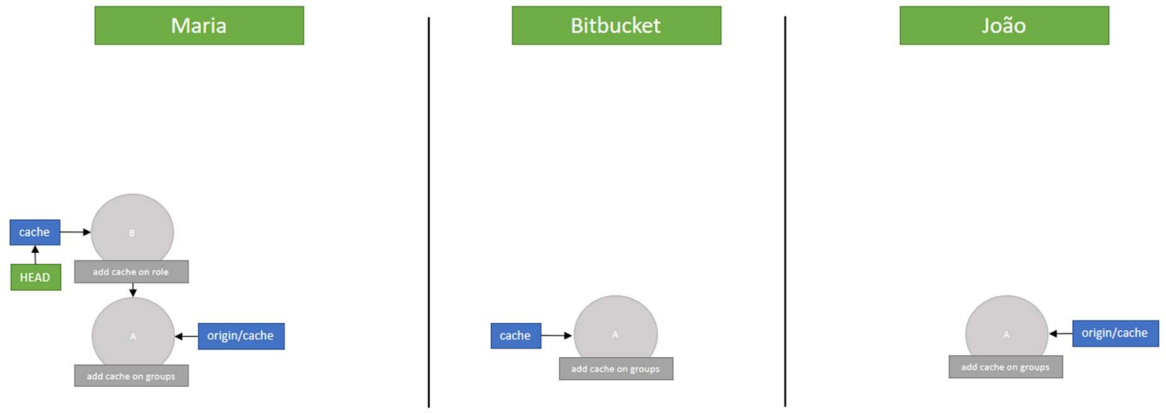


Acessar o repositório de **Maria** e alternar para a branch cache

```
$ git switch cache
```

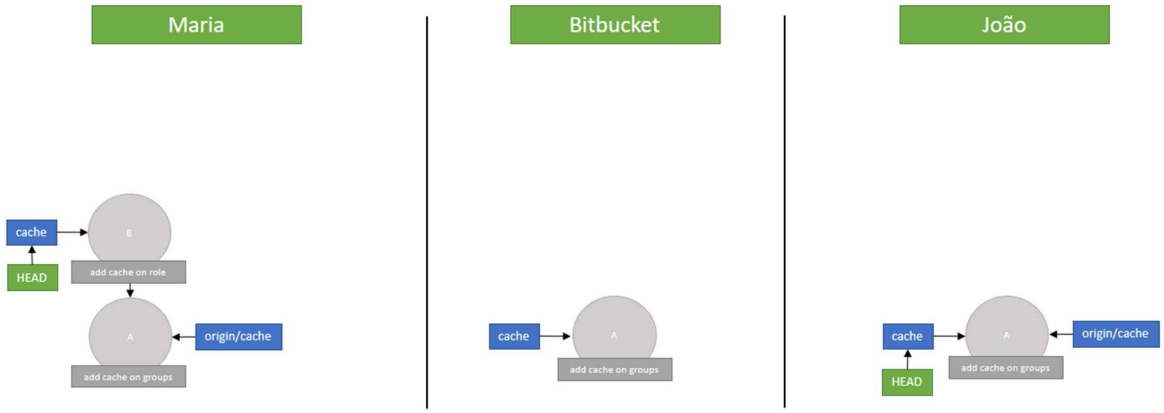
Gerar um novo commit

```
$ echo "function addCache(){...}" >> roles.js
$ git add roles.js
$ git commit -m "add cache on role"
```



Acessar o repositório de **João** e criar uma branch local cache que rastreia origin/cache e alternar para a mesma

```
$ git switch -c cache origin/cache
```



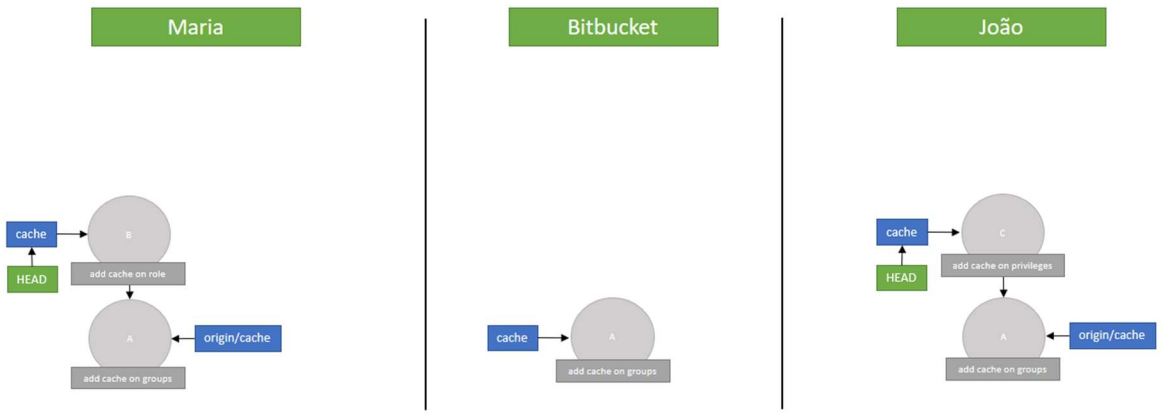
Atualizar o histórico da branch cache

```
$ git pull
```

Nota: *não houve modificações dado que Maria não enviou as alterações dela para o repositório remoto*

Gerar um novo commit

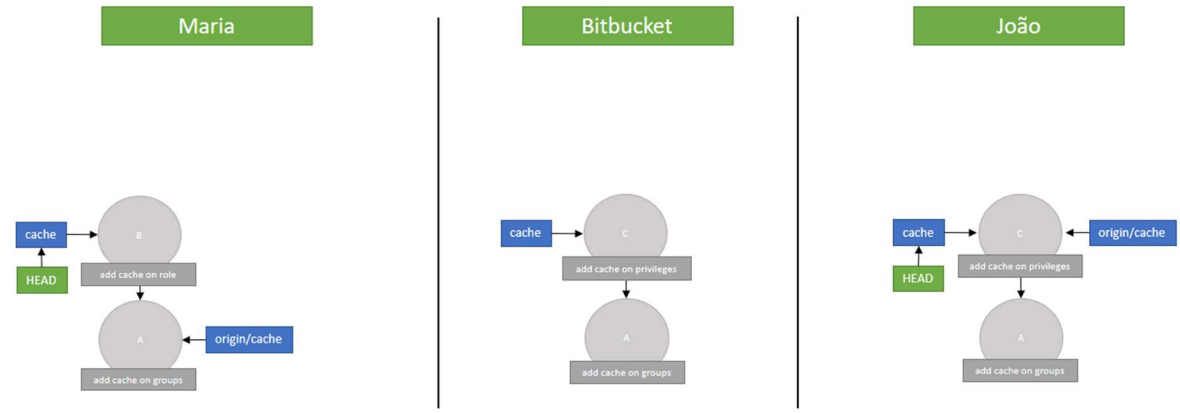
```
$ echo "function addCache(){...}" >> privileges.js
$ git add privileges.js
$ git commit -m "add cache on privileges"
```



ATUALIZANDO COM PULL

No repositório de **joão**, enviar o novo commit para o repositório remoto

```
$ git push
```

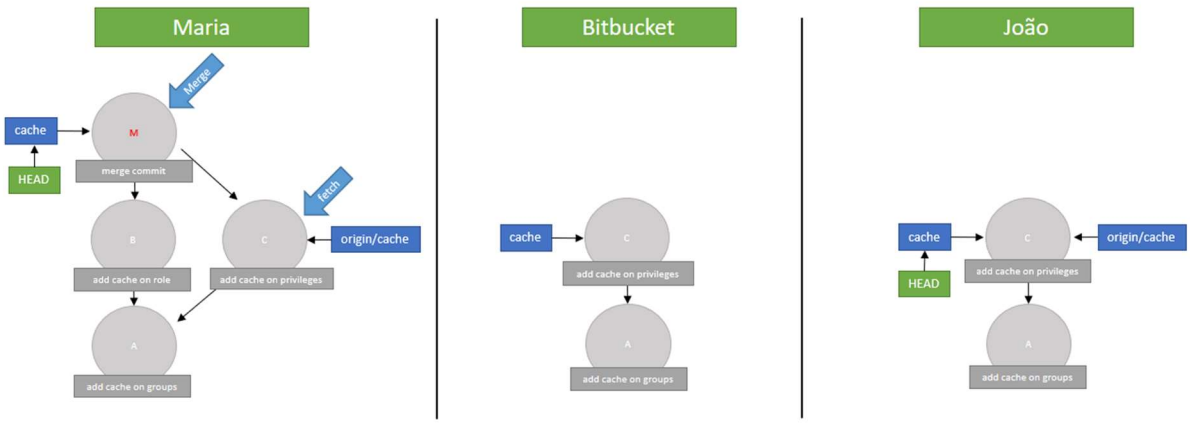


No repositório de **Maria,** atualizar o histórico da branch cache

```
$ git pull
```

Verificar o histórico

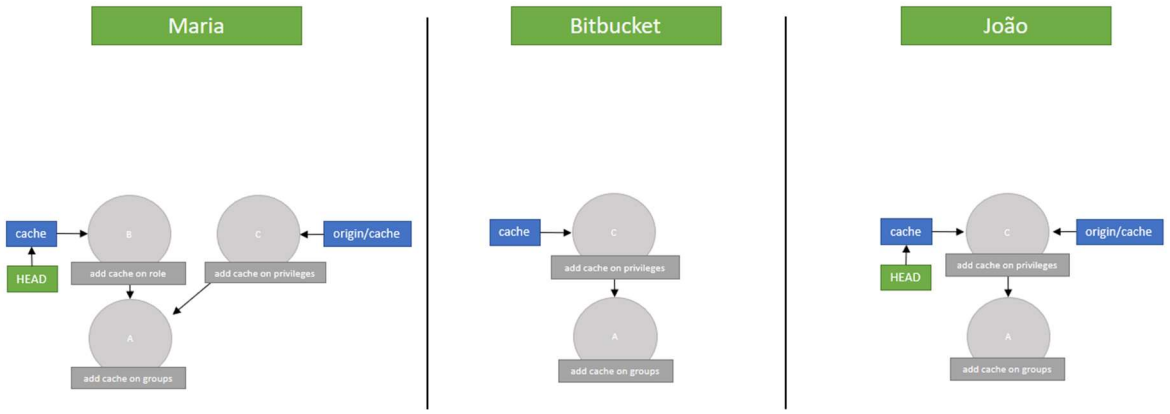
```
$ git hist
```



ATUALIZANDO COM PULL --REBASE

No repositório de **Maria**. desfazer o merge gerado pelo comando git pull acima

```
$ git reset --hard head~
```

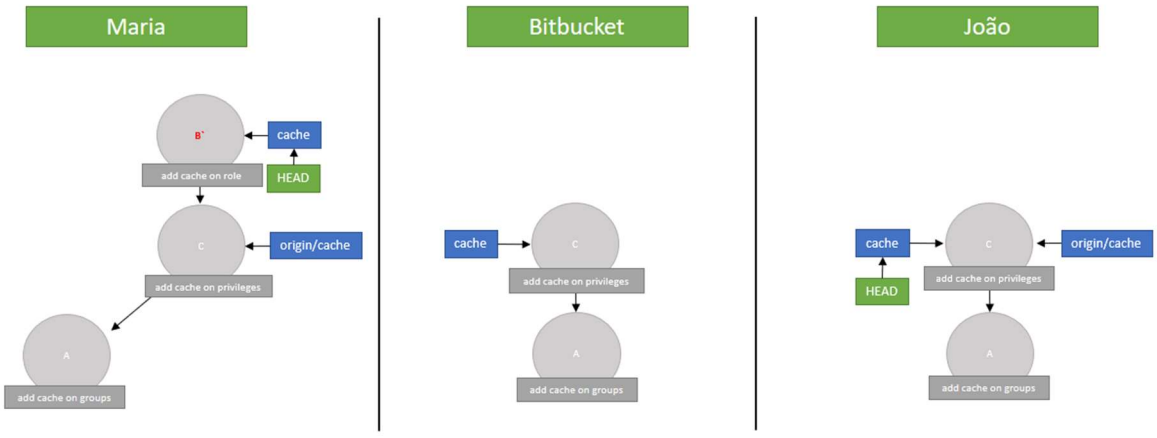


Atualizar com rebase

```
$ git pull --rebase
```

Verificar o histórico

```
$ git hist
```

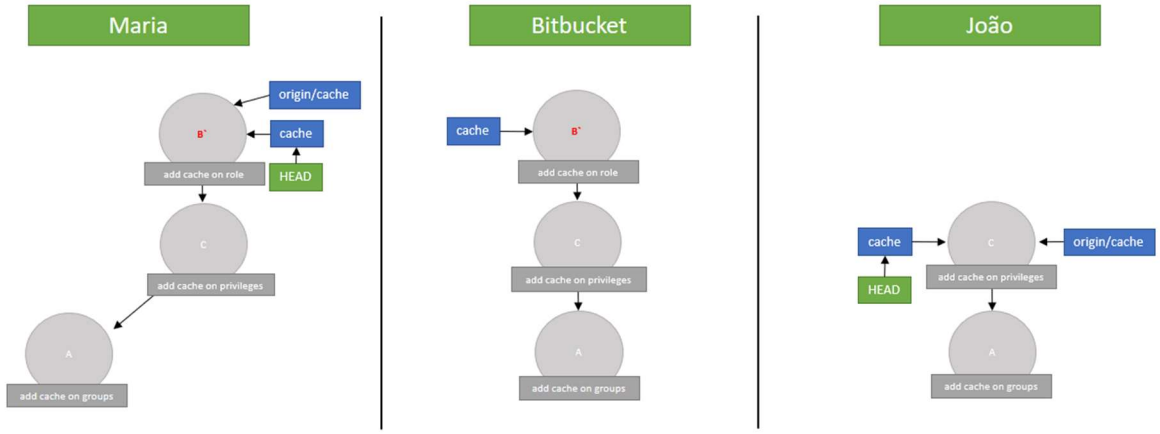


SINCRONIZANDO REPOSITÓRIOS

No repositório de **Maria**, enviar o novo commit para o repositório remoto

```
$ git push
```

Nota: *não foi necessário forçar o envio*

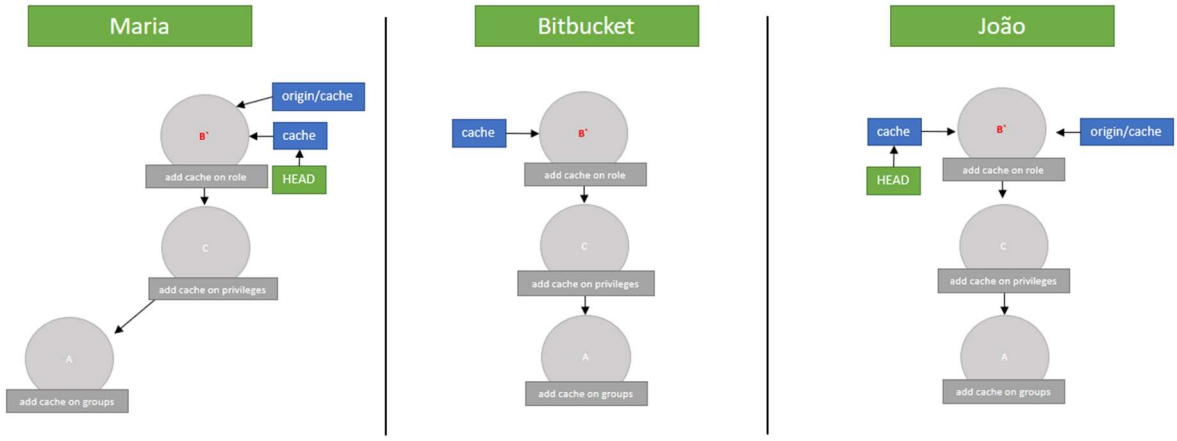


No repositório de **João**, atualizar o histórico da branch cache

```
$ git pull
```

Verificar o histórico

```
$ git hist
```



Emendar commit

ENTENDENDO SOBRE A OPÇÃO --AMEND

No repositório de João gerar um novo commit

```
$ echo "function addCache(){...}" >> login.js
$ git add login.js
$ git commit -m "add cache"
```

Verificar o histórico

```
$ git hist
```

"Atualizar" o último commit ajustando a mensagem

```
$ git commit --amend -m “add cache on login”
```

Verificar o histórico

```
$ git hist
```

Nota: *o segundo commit tem um SHA-1 diferente do primeiro. Ou seja, o segundo commit é um novo com as mesmas alterações do primeiro, mas com uma mensagem diferente*

Enviar para o repositório remoto

```
$ git push
```

Acessar o repositório de Maria e atualizar

```
$ git pull
```

Squash merge

ENTENDENDO SOBRE A OPÇÃO --SQUASH

- Acessar o Bitbucket
- Comparar o histórico da branch cache e branch master
- Abrir um pull request da branch cache para a branch master
- Aprovar o pull request e realizar o merge utilizando a opção squash
- Verificar o histórico da branch master novamente

No repositório de João e Maria:

```
$ git switch master
$ git pull
```

Rebase iterativo

GERANDO COMMITS

No repositório de João criar uma nova branch e alternar para ela

```
$ git switch -c courses
```

Gerar o primeiro commit

```
$ echo "function save(){...}" >> courses.js
$ git add courses.js
$ git commit -m "add save course"
```

Gerar o segundo commit

```
$ echo "function update(){...}" >> courses.js
$ git add courses.js
$ git commit -m "add update course"
```

Gerar o terceiro commit

```
$ echo "function remove(){...}" >> courses.js
$ git add courses.js
$ git commit -m "add remove course"
```

Gerar o quarto commit

```
$ echo "function addCache(){...}" >> courses.js
$ git add courses.js
$ git commit -m "ad cache on course"
```

Nota: *descrição do commit incorreta propositalmente*

Gerar o quinto commit

```
$ echo "bug" >> courses.js
$ git add courses.js
$ git commit -m "add a bug :)"
```

Verificar o histórico

```
$ git hist
```

REORGANIZANDO O HISTÓRICO DE COMMITS COM REBASE ITERATIVO

Realizar rebase iterativo

```
$ git rebase –i head~5
```

Nota: *foi utilizado ~5, pois são os cinco últimos commits dessa aula selecionados para serem reescritos*

No editor, utilizar as seguintes opções

```
pick ... "add save course"
squash ... "add update course"
squash ... "add remove course"
reword ... "ad cache on course"
drop ... "add a bug :)"
```

Nota: *a ordem dos commits na lista acima é do mais antigo para o mais recente*
Nota: *tais opções possuem o seguinte objetivo:*

- *Juntar os três primeiros commits em um único commit com a descrição "add course crud"*
- *Alterar a mensagem do quarto commit para "add cache on course"*
- *Remover o quinto commit*

Verificar o histórico

```
$ git hist
```

Verificar o commit que foi criado através do squash

```
$ git show <hash commit "add course crud">
```

SINCRONIZANDO REPOSITÓRIOS

Enviar ao repositório remoto

```
$ git push -u origin courses
```

No Bitbucket:

- Criar um pull request da branch courses para a branch master
- Aprovar pull request e realizar o merge com a estratégia fast forward
- Verificar o histórico da branch master

No repositório de João e Maria:

```
$ git switch master
$ git pull
```

Está gostando deste curso?

Compartilhe sua experiência nas redes sociais com a tag ***#rsantanatech*** *para que eu possa interagir com a sua postagem.*

Acompanhe nas redes sociais
e fique por dentro de todos os conteúdos.