

WORKSHOP INTRODUÇÃO AO GIT.

aeTua

glua 

andreclerigo committed on Nov 8

OLÁ! Eu sou...

- Aluno de 5º ano
- Mestrado em Engenharia de Computadores e Telemática

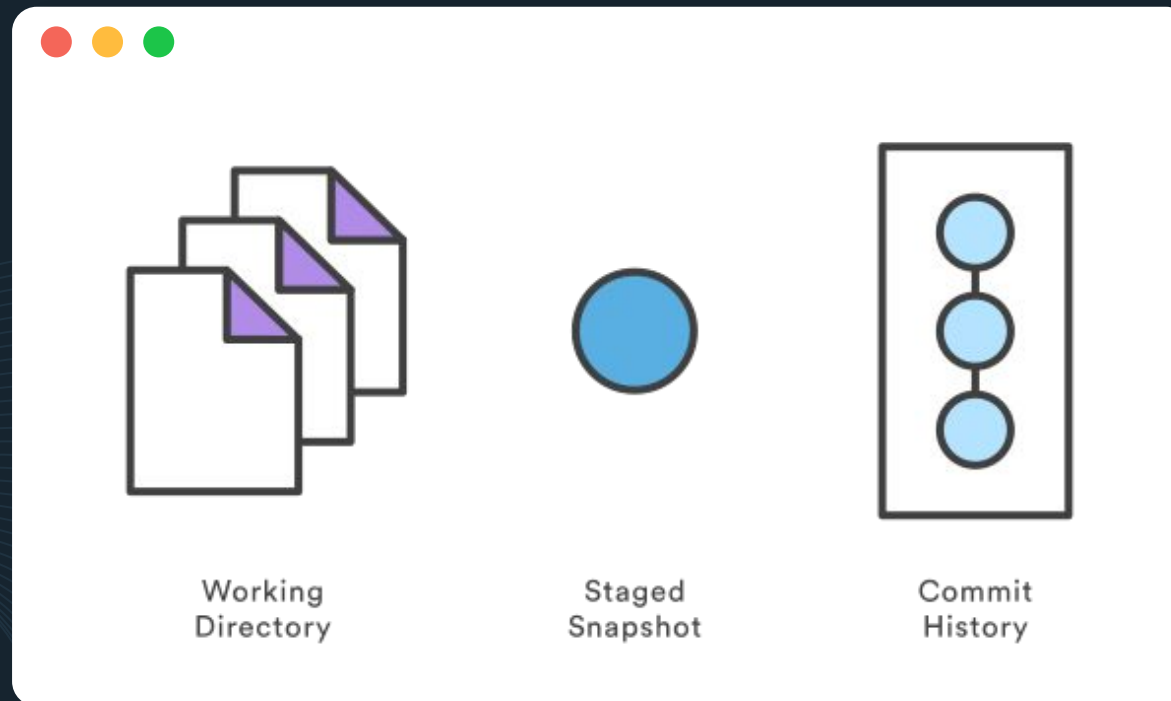


O QUE É O **GIT**?



- Sistema de controlo de versões
- Criado em 2005 por Linus Torvalds
- Trata-se de um Sistema Distribuído de Controlo de Versões
 - **Não necessita de um servidor central, todos programadores têm uma cópia de todas as alterações**
 - **Possibilita desenvolvimento completamente offline**
- Seguro através do uso de SHA1
- É o standard na industria, suportado por todas grandes empresas e pela comunidade OSS (github.com, gitlab.com, bitbucket.com)

COMPONENTES PRINCIPAIS DE UM REPOSITÓRIO **GIT**





O QUE É O GITHUB?

A maior plataforma do mundo para hospedar repositórios git

Permite aos programadores gerirem o seu código:

- Controlo de acesso
- Bug tracking
- Gerir tarefas
- CI/CD

COMO CRIAR UM REPOSITÓRIO?

Podemos criar um repositório directamente a partir da linha de comandos:

```
$ git init
```

Podemos criar o repositório online (ex: github.com) e clonar o repositório para o nosso computador:

```
$ git clone git@github.com:  
andreclerigo/introducao_ao_  
git.git
```



Servidor Github
"origin"



Sistema Local

ADICIONAR UM FICHEIRO

```
$ git add nome_do_ficheiro
```

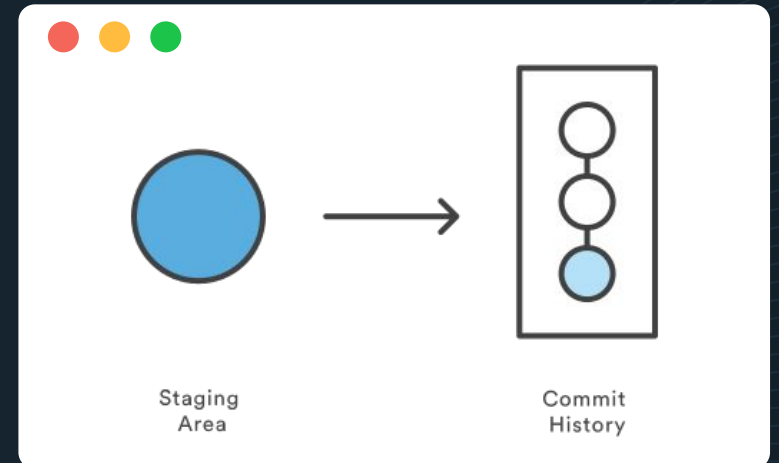
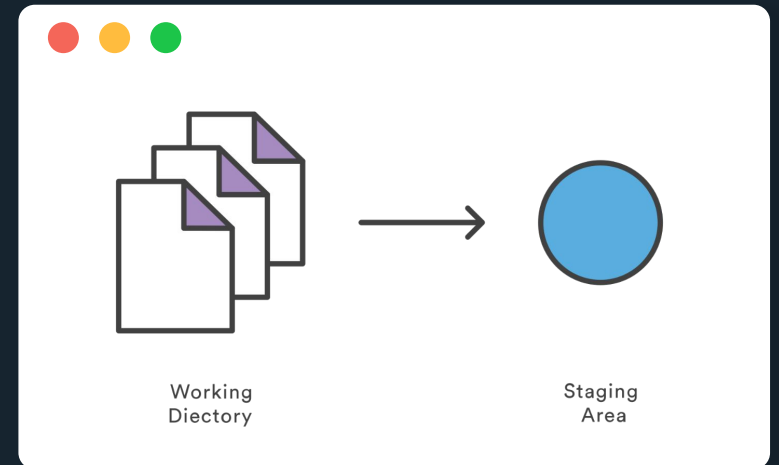
```
$ git commit -m "mensagem"
```

Ficheiro reside agora no repositório local

Podemos igualmente apagar ou mudar o nome de um ficheiro:

```
$ git rm nome_do_ficheiro
```

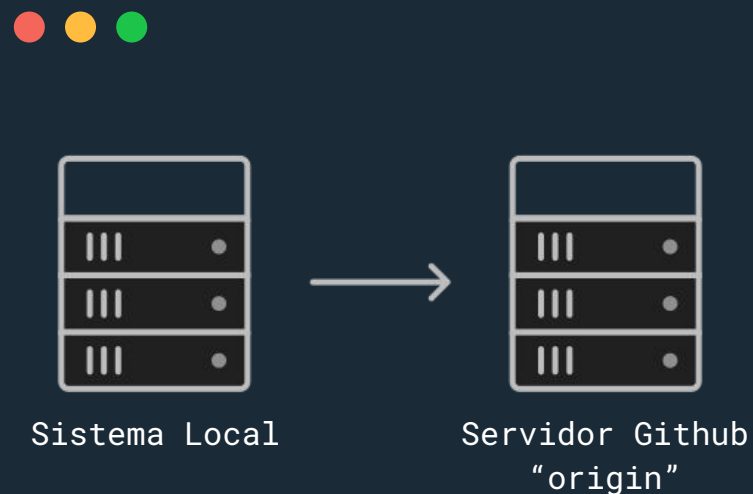
```
$ git mv nome_do_ficheiro
```



SINCRONIZAR COM O SERVIDOR REMOTO

Para adicionar um repositório que já existe localmente:

```
$ git remote add origin git@github.com:  
andreclerigo/introducao_ao_git.git  
$ git branch -M main  
$ git push -u origin main
```



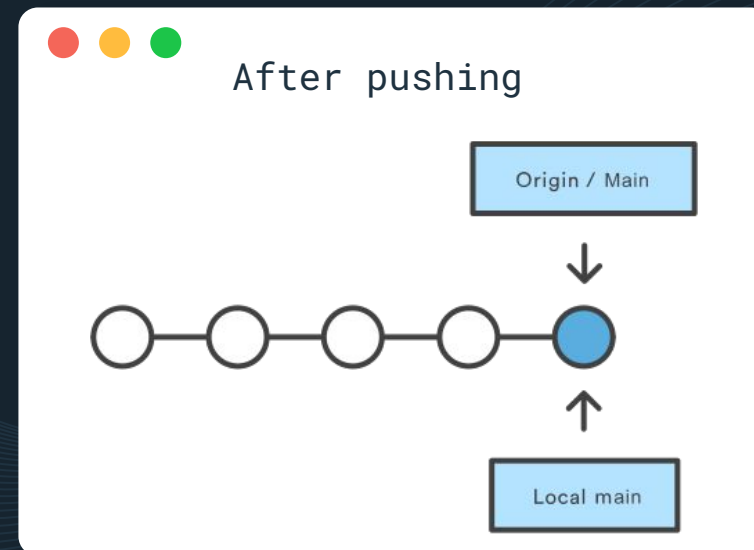
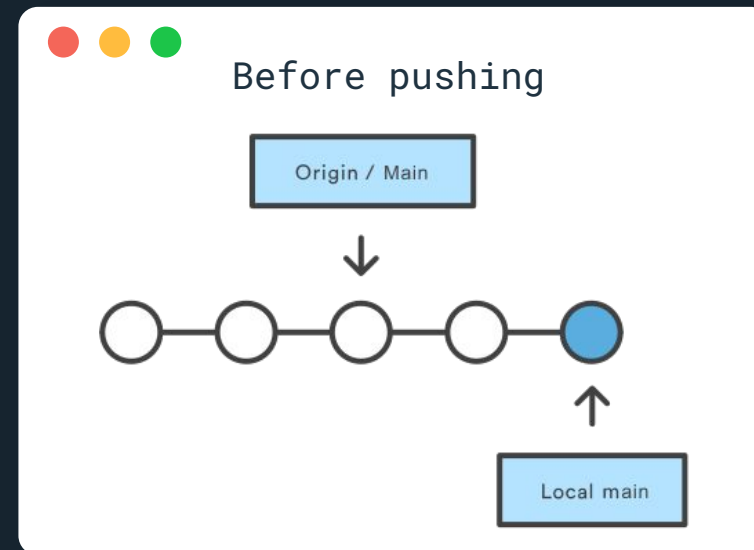
ATUALIZAR O SERVIDOR REMOTO

```
$ git push origin main
```

Envia as nossas alterações feitas na master para o remote origin

Antes de um push num repositório compartilhado é importante sincronizar o repositório local e resolver qualquer conflito:

```
$ git pull
```



VER O ESTADO DA WORKING TREE

```
$ git status
```

Informa que ficheiros precisam ser adicionados (porque foram alterados)

Informa que ficheiros existem no computador mas que não pertencem ao repositório

```
$ git log
```

Informa de todas as alterações feitas até ao momento

Informa de commits anteriores e qual o seu identificador SHA1



VER ALTERAÇÕES DOS COMMIT'S

```
$ git diff commit_id
```

Mostra-nos as diferenças entre commits, working directory e commit, etc..

```
$ git show commit_id
```

Mostra-nos o que foi submetido em commit_id

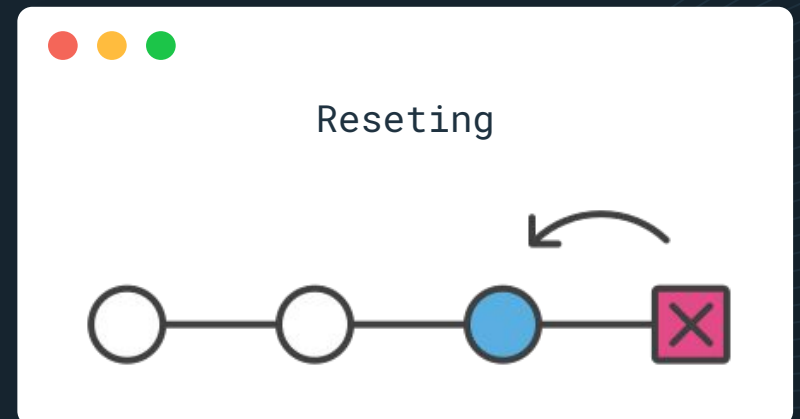
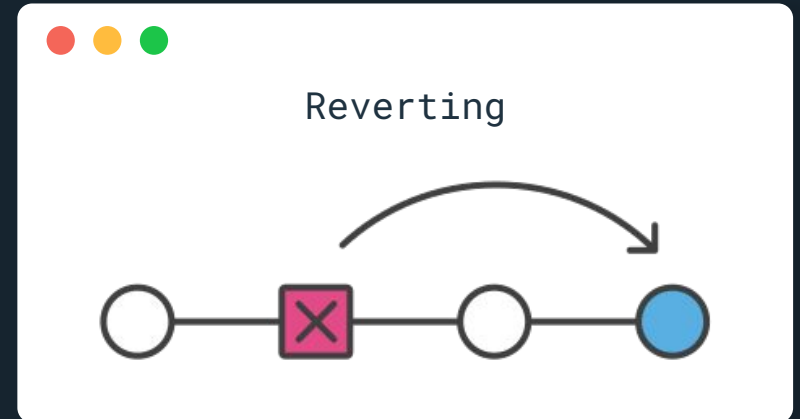
DESFAZER ALTERAÇÕES

```
$ git revert commit_id
```

Cria um novo commit que desfaz as alterações de commit_id e aplica ao branch actual

```
$ git reset --hard commit_id
```

Desfaz todas alterações até ao commit_id, apaga todas alterações (sem --hard, as alterações mantêm-se em disco)



CRIAR UMA BRANCH

Criamos uma branch com:

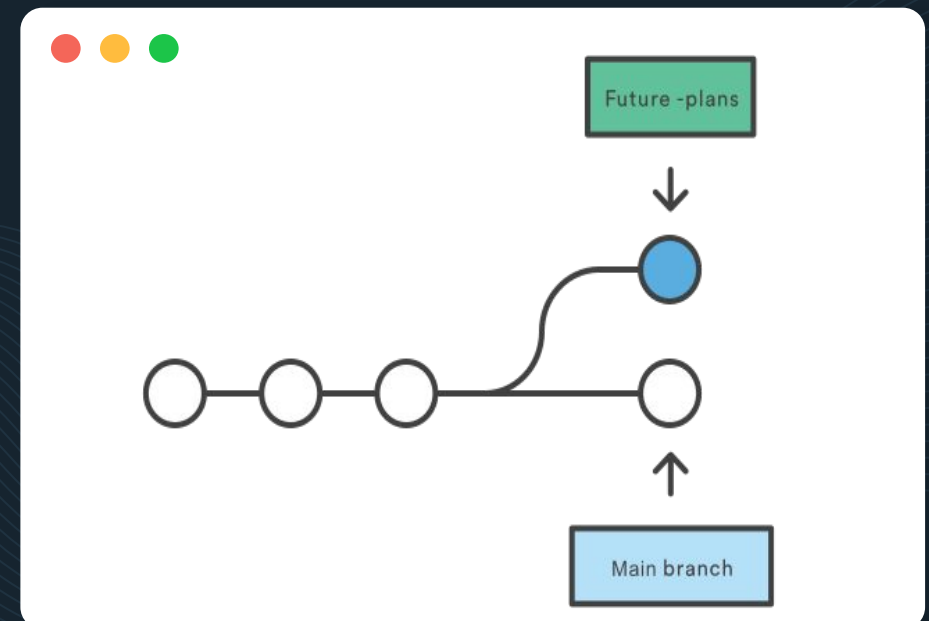
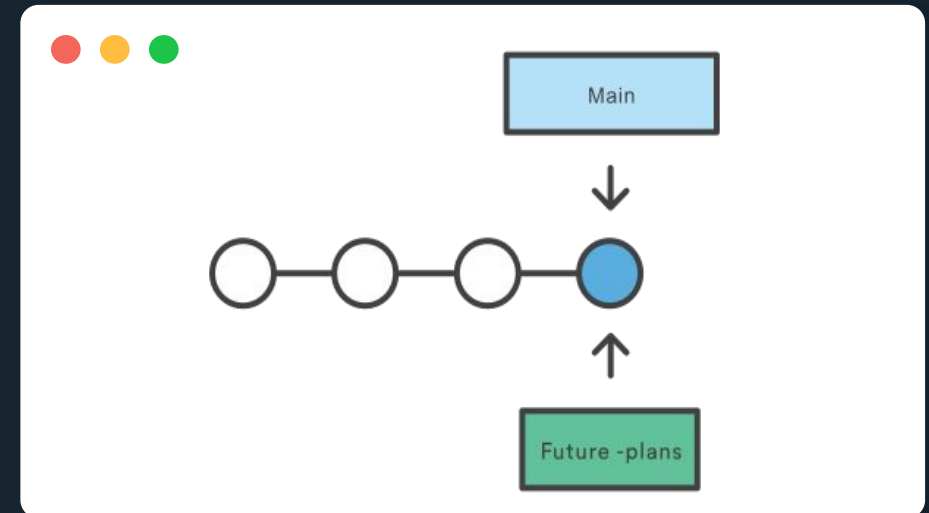
```
$ git branch future-plans
```

Mudamos de branch com:

```
$ git checkout future-plans
```

Para listar os branches e ver a atual:

```
$ git branch
```



MERGING

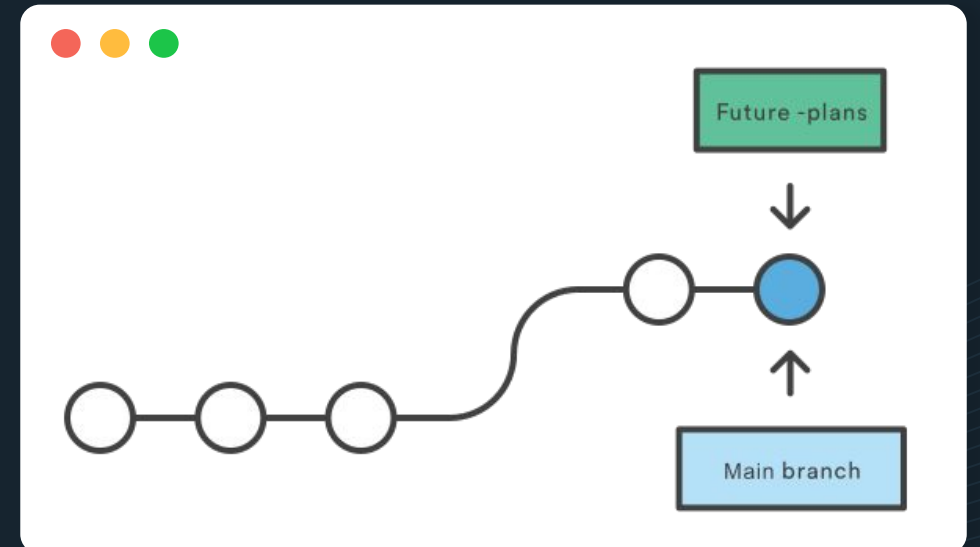
Antes de fazer merge vamos mudar-nos para a branch com que queremos dar merge: `$ git checkout main`

`$ git merge future-plans`

O merge é apenas a actualização do HEAD de main

`$ git branch -d future-plans`

Agora que já não precisamos da branch merge podemos apagá-la



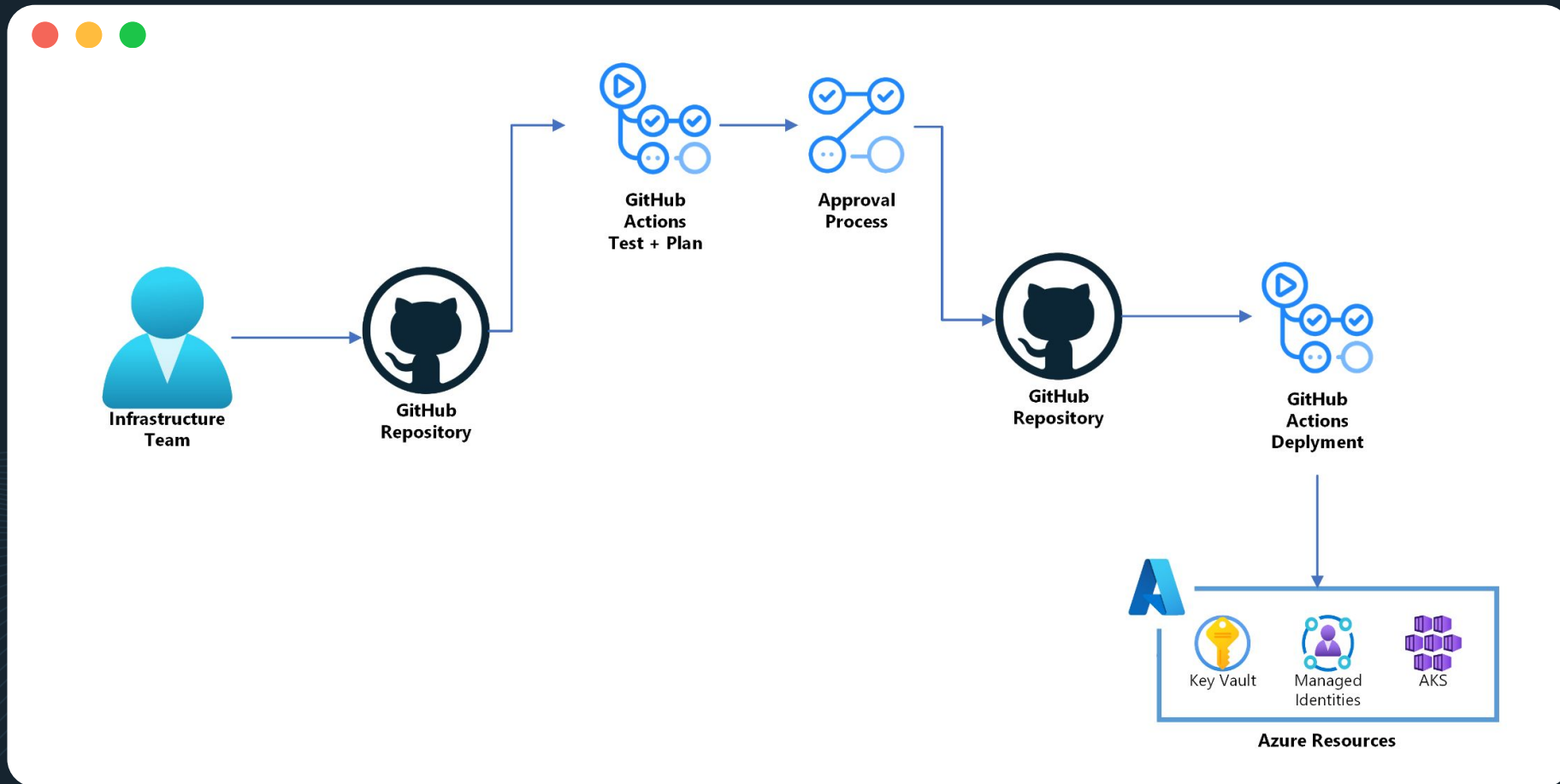
MERGE FALHOU



Por vezes durante um merge podemos ter conflitos (alterações concorrentes no mesmo pedaço de código)

- Vamos fazer `$ git status` para saber que ficheiros precisam de ser resolvidos
 - Procurar por “<<< ==== >>>” e editar os ficheiros
 - `$ git add ficheiro` da versão que queremos manter
 - `$ git commit -m “corrigir conflitos”`
- Em casos muitos específicos: `$ git stash` e `$ git stash pop`
- Se quiserem abortar o merge que deu conflitos: `$ git merge --abort`
- Se quiserem reverter um merge mal feito: `$ git reset --hard`

O QUE SÃO ACTIONS?



DICAS PARA O GIT E GITHUB



- Configurar username e email do Git

<https://docs.github.com/en/get-started/quickstart/set-up-git#setting-up-git>

- Usem chaves SSH em vez de HTTPS quando gerem os repositórios, evitam o uso de senhas

- Gerar as chaves SSH e adicionar à conta Github

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account>

- Evitar editar ficheiros online (maior propensão para criar conflitos)

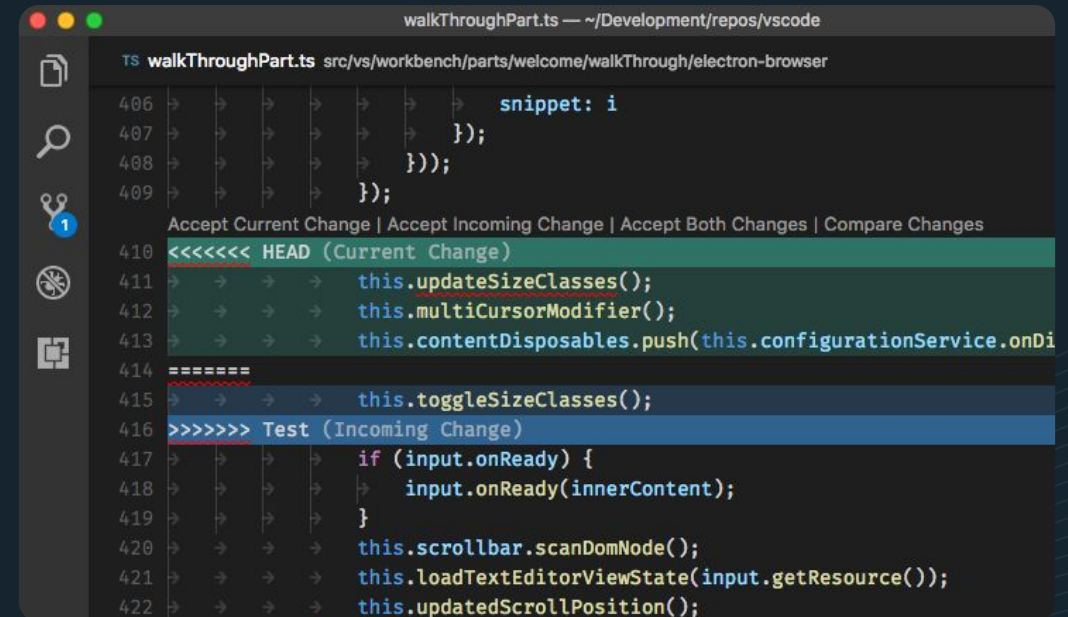
I LOVE VSCODE

Já tem ferramentas built-in para resolver merge conflicts!

Numa sessão Live Share o commit adiciona Co-Author automaticamente

Github Lens:

- Visualizar quem fez as alterações
- História das alterações
- Ver a commit tree



```
TS walkThroughPart.ts src/vs/workbench/parts/welcome/walkThrough/electron-browser


406 → → → → → → → snippet: i
407 → → → → → → → });
408 → → → → → → → }});
409 → → → → → → → });

Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
410 <<<<<< HEAD (Current Change)
411 → → → → → → → this.updateSizeClasses();
412 → → → → → → → this.multiCursorModifier();
413 → → → → → → → this.contentDisposables.push(this.configurationService.onDi
414 =====
415 → → → → → → → this.toggleSizeClasses();
416 >>>>>> Test (Incoming Change)
417 → → → → → → → if (input.onReady) {
418 → → → → → → →   input.onReady(innerContent);
419 → → → → → → → }
420 → → → → → → → this.scrollbar.scanDomNode();
421 → → → → → → → this.loadTextEditorViewState(input.getResource());
422 → → → → → → → this.updatedScrollPosition();
```




MUITO OBRIGADO!

Alguma questão?



Let's take
a break

20 minutos



Algumas das imagens são retiradas de:
<https://www.atlassian.com/git/tutorials>