Taller de git

GIT BÁSICO

Objetivos de hoy

- Consultando la ayuda en git
- Aprender a configurar git global y localmente
- Aprender a trabajar los tres estados de git
- Aprender algunos patrones regex para agregar archivos y hacer un git ignore

Consultar la documentación de git

- La documentación que aparece en línea es igual a la que otorga el comando
- El comando para pedir ayuda es:
  - git help
- Se puede agregar como argumento el comando a correr
- La ayuda se desplaza en more, para salir se puede presionar q

Probemos el comando

- Ejecuten el siguiente comando:
  - git help
- Despues corran el comando:
  - git help status

Configuración de git

- Configuramos git para que se adapte mejor a nuestras necesidades
- Indica quiénes somos así como algunas preferencias en nuestro sistema
- Hay 3 niveles de configuración de git
  - System – Aplica a todos los usuarios del equipo
  - Global – Aplica al usuario actual
  - Local – Aplica al repositorio actual

¿Qué debemos configurar?

- Nuestro nombre
  - user.name
- Nuestro email
  - user.email
- Nuestro editor de texto
  - core.editor

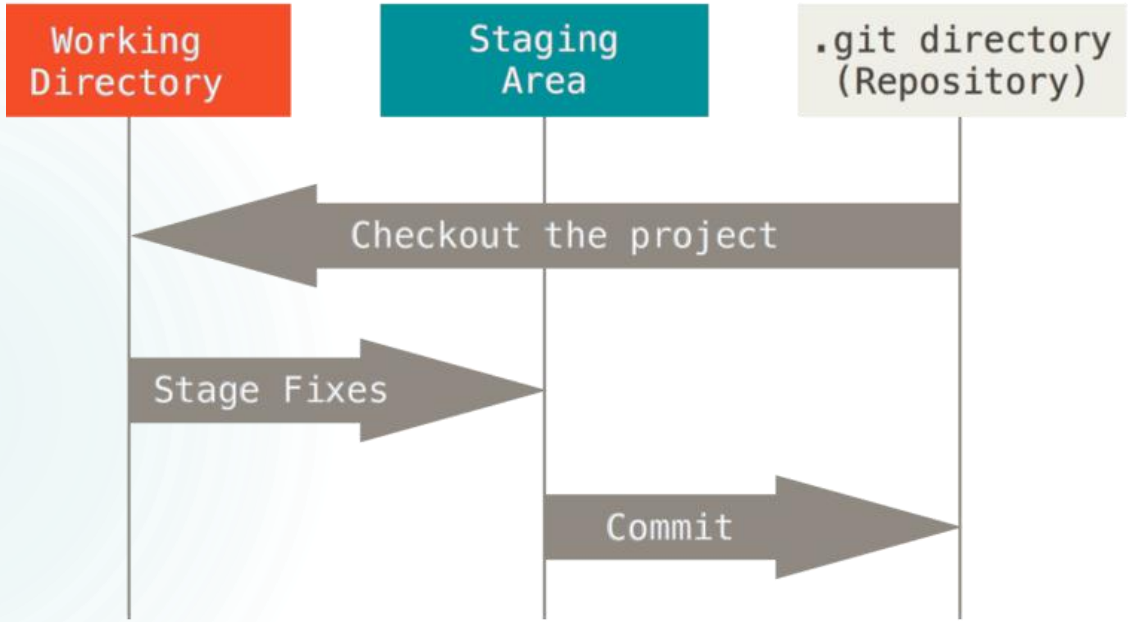
  - ¿Cómo lo hacemos?
    - git config [--global | --system] <opción> <valor>

      - [] = Opcional
      - | = or
      - <valor> = Cadena a remplazar

Opciones útiles

- commit.template
  - Establece un template global que aparece al momento de hacer commit. Ayuda a seguir un estándar
- core.excludesfile
  - Establece un .gitignore general
- help.autocorrect
  - El valor es en décimas de segundo. Si escribimos mal, git tratará de encontrar el comando que queríamos y lo correra tras cierto tiempo

Los tres estados de git



Consultando el estado actual

- Ejecutar el siguiente comando
  - git status

- Los archivos suelen tener una cantidad muy limitada de estados:
  - Untracked -> el archivo no existe en el repositorio
  - Modified
  - Added
  - Deleted
  - Renamed

La opción s

- -s nos muestra el estado resumido mediante dos columnas
- La derecha es el working directory
- La izquierda es el staging area

- ?? Significa que el archivo está untracked

Ejercicio 1

- Correr el comando
  - git status
- Comparar el resultado con
  - git status –s
- Crear un archivo y volver a correr el comando

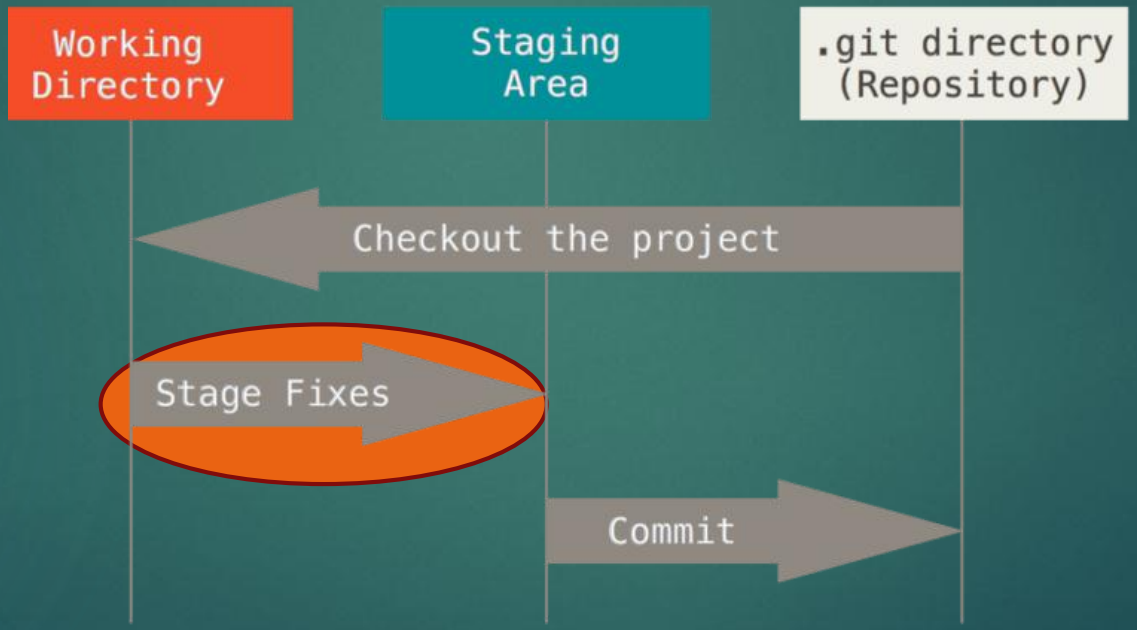
```
 M "Von Newman.xcodeproj/project.pbxproj"
 M "Von Newman.xcodeproj/xcuserdata/rikyfocil.xcuserdatad/xcschemes/Von Newman.xcscheme"
M  "Von Newman/Images.xcassets/AppIcon.appiconset/Contents.json"
M  "Von Newman/Images.xcassets/Button-Normal.imageset/Contents.json"
M  "Von Newman/Images.xcassets/cuadroN.imageset/Contents.json"
M  "Von Newman/Images.xcassets/cuadroS.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_AC --- ALU N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_AC --- ALU S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_AC --- FR N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_AC --- FR S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_ALU -- MAR N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_ALU -- MAR S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_IR -- AC N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_IR -- AC S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_IR -- ALU N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_IR -- ALU S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_IR -- MAR N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_IR -- MAR S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_IR -- PC N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_IR -- PC S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MAR -- RAM N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MAR -- RAM S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR -- ALU N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR -- ALU S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR -- IR N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR -- IR S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR -- MAR N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR -- MAR S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR -- PC N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR -- PC S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR --- AC N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR --- AC S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR --- DISP N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR --- DISP S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR --- RAM N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_MDR --- RAM S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_PC --- ALU N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_PC --- ALU S.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_PC--MAR N.imageset/Contents.json"
M  "Von Newman/Images.xcassets/vns_PC--MAR S.imageset/Contents.json"
 M "Von Newman/Von Newman-Info.plist"
 M "Von NewmanTests/Von NewmanTests-Info.plist"
?? distributioncer copy.p12
```

```
 D "Development cer.p12"
 D distributioncer.p12
?? hola.p12
?? logomark-orange@2x.png
```

```
D  distributioncer.p12
R  "Development cer.p12" -> hola.p12
A  logomark-orange@2x.png
```

Moviendo De working directory a staging area

- El comando para mover de working directory al staging area es
  - git add <archivo>



Opciones

- -n --dry-run
  - Simula el comando
- -A --all
  - Todos los archivos
- .
  - Directorio actual y subdirectorios
- -f --force
  - Agrega archivos ignorados

Ejercicio 2

- En el repositorio que trabajaron la sesión pasada:
  1. ¡Asegúrense que no haya archivos fuera del commit!
  2. Creen un archivo nuevo
  3. Modifiquen un archivo
  4. Eliminen un archivo
     - Los tres archivos deben ser diferentes por favor
  5. Ejecuten git status para ver el resultado
  6. Simulen los cambios a staging area con -n
  7. Agreguen todos los cambios al staging area
  8. Modifiquen de nuevo el archivo que modificaron
  9. Corran git status en su versión larga y corta

Aprendiendo patrones regulares

- Git utiliza algo llamada glob patterns
- Son patrones regulares para manipular strings con una mayor facilidad
- Algunos comandos comunes son:

*	Lo que sea
?	Un solo carácter
[AB]	A o B
[A-F]	Cualquier carácter entre A y F
*	*
[!ab]	Cualquier carácter excepto a y b

- Nota: Los archivos escondidos son excluidos de los patrones regulares a menos que éstos empiecen en .

Ejercicio 3

- Escribir expresiones regulares que hagan lo siguiente:
  1. Agreguen todos los archivos con extensión cpp
  2. Todos los archivos que empiecen con una letra
  3. Todos los archivos que terminen con un número
  4. Todos los archivos que empiecen con un número y terminen con una letra
  5. Todos los archivos que tengan exactamente 3 letras en su nombre y cualquier extensión

*	Lo que sea
?	Un solo caracter
[AB]	A o B
[A-F]	Cualquier caracter entre A y F
*	*
[!ab]	Cualquier caracter exepto a y b

Quitando archivos de staging area

- Asegúrense de no agregar ninguna opción al momento de correr los siguientes comandos ya que podrían resultar en pérdida de trabajo

- En caso de haber cometido un error, podemos sacar un archivo del staging area. Esto es útil en los siguientes casos:
  - Agregamos un archivo que deberíamos ignorar
  - Agregamos un archivo incompleto o con cambios que aún no están listos

- Para deshacer el cambio, podemos correr el siguiente comando
  - git reset HEAD <archivo>

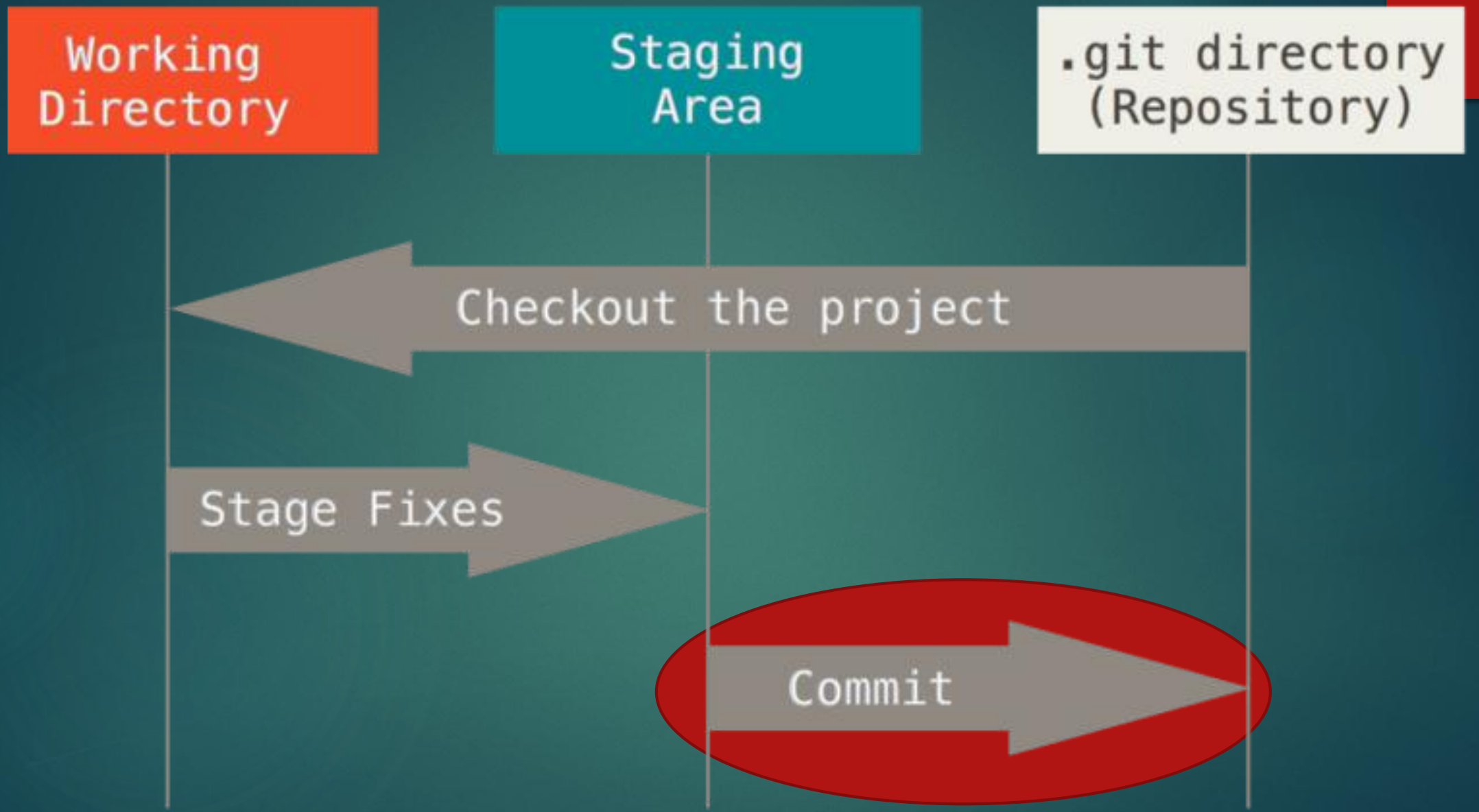
- Al correr el comando el archivo volverá a nuestro working directory.
- HEAD es una palabra especial para git

De staging area al repositorio de git

- Una vez que nuestros cambios estén listos para quedarse es hora de pasarlos del staging area al commit area (o .git area)
- El comando para hacer esto es
  - git commit

- Tras esto, se nos solicitará un mensaje en nuestro prompt elegido

- Opcionalmente podemos especificar el mensaje en la línea de comandos con la opción –m, sin embargo, esto solo es recomendable cuando:
  - El cambio es insignificante y no amerita gran explicación
  - Estamos trabajando en una rama local que integraremos con más commits


Working Directory
Staging Area
.git directory (Repository)
Checkout the project
Stage Fixes
Commit

¿Y si olvidé un archivo?

- El último commit se puede arreglar por los siguientes motivos:
  - Agregar un archivo faltante
  - Cambiar el mensaje de commit
  - Eliminar un archivo del commit *
- Sin embargo, sólo es recomendable hacerlo cuando nuestro commit aún es local. NUNCA arreglen un commit que ya está en la nube.
- Para arreglar un commit
  1. Hagan los cambios que necesitan corregir
  2. Ejecuten el siguiente comando (En este punto se puede cambiar el mensaje)
     - git commit --amend
- * Esto no lo haremos aún porque requiere un poco más de conocimientos sobre como consultar la historia

.gitignore

- En ocasiones no queremos subir algunos archivos al repositorio, por ejemplo entren a la siguiente liga:
  - https://github.com/search?l=PHP&q=wp-config.php&type=Commits&utf8=✓
- Esos archivos que aparecen son archivos de configuración e incuyen contraseñas de las aplicaciones
- Para evitar subir esos archivos creamos un archivo .gitignore, sin embargo, ya hay muchos creados para nosotros:
  - https://github.com/github/gitignore

.gitignore reglas

#	Comentario
\#	#
!	Negación
Foo/	Directorio Foo
Foo	Archivo Foo
**/foo	Todos los archivos foo sin importar donde esten
foo/**	Todos los archivos adentro del directorio foo

Más información en https://git-scm.com/docs/gitignore

Ejercicio

- Buscar en la liga anterior un .gitignore apropiado para su repositorio y agregarlo a su carpeta.
- Agregar algún patrón regular a su .gitignore
- Hacer un commit con el .gitignore agregado

#	Comentario
\#	#
!	Negación
Foo/	Directorio Foo
Foo	Archivo Foo
**/foo	Todos los archivos foo sin importar donde esten
foo/**	Todos los archivos adentro del directorio foo

.gitignore de archivos en el repositorio

- Los siguientes comandos nos pueden ayudar a eliminar archivos de nuestro repositorio:

  - git rm -r --cached .
  - git add -A
  - git commit -m 'Removing ignored files'

- A pesar de que los archivos se eliminan de la rama actual, éstos siguen en el repositorio por lo que aún se podrían consultar