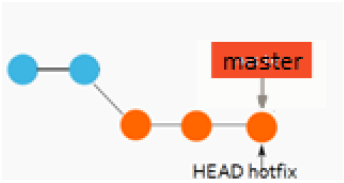
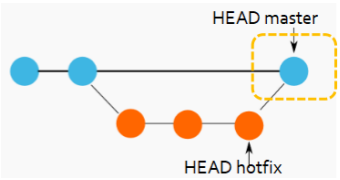
COMANDOS

- **git config:** Accedes a la configuración de git (poner email, nombre…)
- **git init:** Inicializamos git(se crea la carpeta .git)
- **git status :** Nos muestra el estado de git(si hay algún archivo en staged o para commit)
- **git add .:** añade los cambio para ser comprometidos(commit)
- **git commit -m "comentario" :** sompromete los cambios
- **git reset HEAD archivo:** devuelve los cambios que van a ser comprometidos a modificado
- **gitk:** nos lo muestra visualmente
- **git log**: nos muestra el log de lo que hemos ejecutado
- **git ckeckout** -- . : recupera los archivos eliminados o cambiados(en el satged)
- **git hist:** nos muestra el historial (los commit, las ramas) nos da el hash de los commit
- **git checkout (elhash):** nos lleva a la posición de ese commit y sus cambios
- . : el . significa que se aplica a todo
- **master:** esta es la rama principal
- git tag v1.0: le asigna un tag al commit en el que estemos posicionados (podemos usar el git checkout con el tag para movernos entre los commits(git checkout v1.0)
- **git reset HEAD^ --hard:** eliminar un commit por completo
- **git revert HEAD**: en vez de eliminar los commits, los mantiene, pero crea una nueva secuencia que salta esos commits
- **git branch develop**: crea una nueva rama
- **git checkout develop**: nos mueve a esa rama(si ponemos master nos lleva a la principal)
- **git branch –v**: ver el último commit en cada rama
- **git branch –-move <antiguonombre> <nuevonombre>:** cambia el nombre de una rama
- **git branch –merged o unmerged:** vemos que ramas han sido unidas y cuales no

- **git merge develop**: unimos la rama al commit en el que nos encontremos actualmente
- 

- **git merge --no-ff develop:** a diferencia del anterior, en este caso, la rama master no se añade a la nueva rama , sino que es la propia rama master la que al cambiar no sigue el camino de la rama develop sino que sigue por su propia rama, esto no crea un nuevo commit.



- **git merge -s recursive develop**: unimos la rama al commit, en este caso si en la rama master a habido un nuevo commit despues de haber creado la nueva rama, arrastra a todos los commits crados posteriormente en la rama master(utilizar cuanado se se han creado comiits(en la rama master), tras haber creado la rama).
- **git branch –D develop**: eliminamos una rama
- **Git stash**: otra form de ramificar, temporal(mirar en apuntes pag. 26)
- **git rebase develop**: copia un commit y lo pega en otra rama(en la actual)
- **git cherry-pick (hash1) (hash2)**: para mover directamente un commit entre ramas
- **git remote add origin** https://github.com/marcos261099/university.git : conctamos con el repositorio online de github(se pueden tener varios repositorios online a la vez)
- **git remote set-url origin** https://github.org/repo.git: para cambiar a otro repositorio
- **git remote –v :** Verificamos conexión correcta.
- **git push origin master**: subimos los archivos al repositorio
- **git pull origin master**: traemos los cambios de github(inverso a lo anterior)
- **git fetch origin master:** parecido a pull, la diferencia es que: de manera simplificada podemos decir que git pull hace un git fetch

seguido de una git merge, es decir, comprueba cambios de una forma meramente informativa(sin descargar nada) respecto al repositorio online(para ver si alguno de los colaboradores ha cambiado algo).

- **git diff master origin/master**: relacionado con la anterior, sirve para comparar los cambios(lo anterior extrae los metadatos), entre el repositorio local y el online extraeido con git fetch
- **git clone** https://github.com/repo.git: clona un repositorio (se suele usar para colaboradores)

CREAR EL MAVEN

Desde línea de comandos y dentro del directorio donde pondrás tu proyecto ("Prj_Maven") teclea

mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-quickstart

Define value for groupId: : **org.dis**

Define value for artifactId: : **tutorial_maven**

Define value for version: 1.0-SNAPSHOT: : 1.0

Define value for package: org.dis: **org.dis**

Borra el directorio llamado src, ya que no necesitamos código fuente.

Abre con un editor de texto el archivo pom.xml.

Cambia el tipo de packaging de jar a pom, tal y como ves en la siguiente página.

```
<modelVersion>4.0.0</modelVersion>
<groupId>org.dis</groupId>
<artifactId>tutorial_maven</artifactId>
<version>1.0</version>
<packaging>pom</packaging>
<name>tutorial_maven</name>
<url>http://maven.apache.org</url>
....
```

Si todo OK, compilación del proyecto desde el subdir. raíz (*tutorial_maven*):

mvn install

creamos el jar:

mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-quickstart

Define value for groupId: : **org.dis**

Define value for artifactId: : **tutorial_maven_jar**

Define value for version: 1.0-SNAPSHOT: : **1.0**

Define value for package: org.dis: **org.dis**

El archivo tutorial_maven/tutorial_maven_jar/pom.xml

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <artifactId>tutorial_maven</artifactId>
    <groupId>org.dis</groupId>
    <version>1.0</version>
  </parent>
  <groupId>org.dis</groupId>
  <artifactId>tutorial_maven_jar</artifactId>
```

Si todo OK, compilación del proyecto desde la carpeta tutorial_maven_jar:

mvn install

```
<build>
<plugins>
<plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-compiler-plugin</artifactId>
 <configuration>
   <source>1.8</source>
   <target>1.8</target>
 </configuration>
</plugin>
</plugins>
</build>
```

Mvn install

En el tutorial_maven editamos en las dependencias :

```
  <scope>test</scope>
 </dependency>
   <dependency>
       <groupId>log4j</groupId>
       <artifactId>log4j</artifactId>
       <version> 1.2.9 </version>
    </dependency>
 </dependencies>
```