

GIT – REPOSITORIO DISTRIBUIDO

1^{er} Curso – DAW

Entornos de Desarrollo

“Una jornada de mil millas debe comenzar con un primer paso”

Lao-Tse

ÍNDICE DE CONTENIDOS

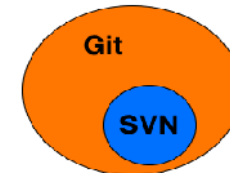
- Introducción a Git
- SVN Vs Git
- SVN Workflow
- Git workflow
- Instalación de Git
- GitHub
- TortoiseGit (Instalación)
- Día a Día con Git
- Webgrafía

Introducción a GIT

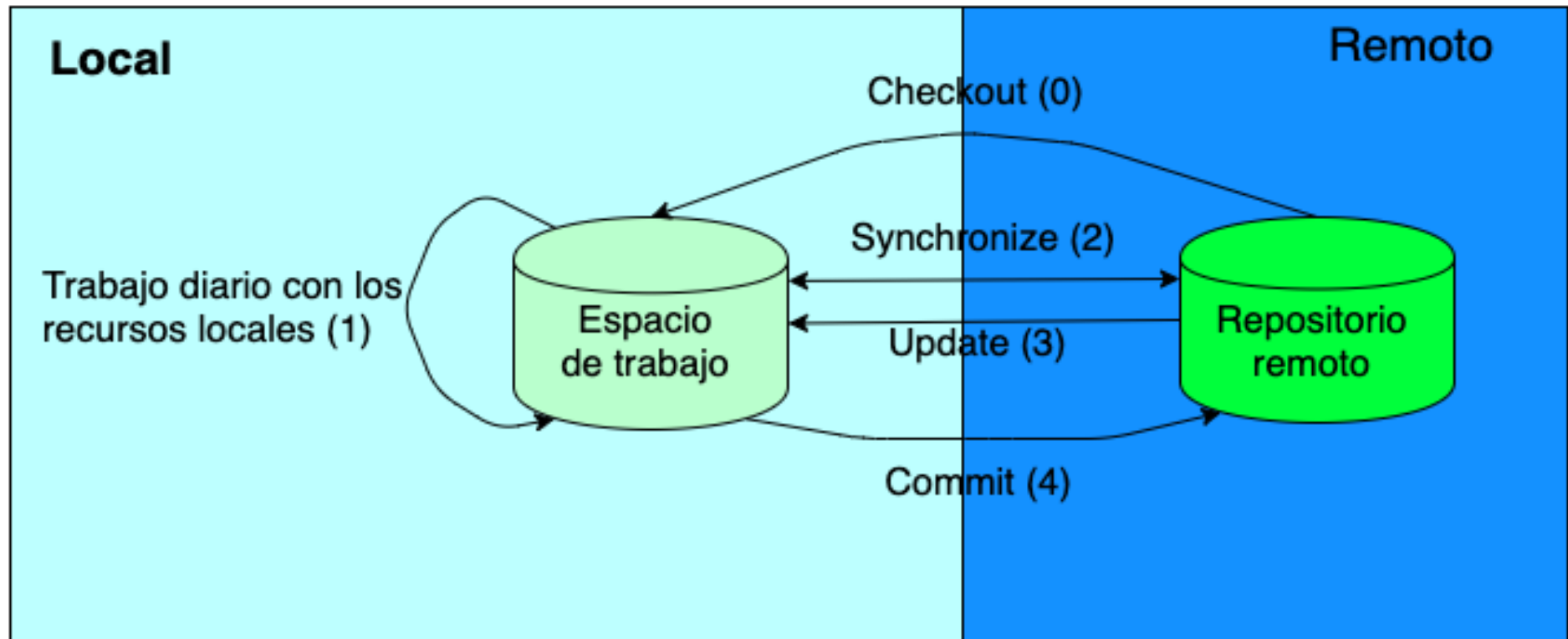
- Git es un sistema de **control de versiones distribuido**.
- Control de versiones:
 - Optimiza y agiliza el **trabajo colaborativo**.
 - **Gestiona los cambios** llevados a cabo en los recursos de un proyecto por cada uno de los miembros del equipo.
- Distribuido:
 - A pesar de que puede existir una copia central repositorio del proyecto (**repositorio remoto**), quizás la característica más importante de Git es que cada uno de los miembros del equipo cuenta con una copia local (**repositorio local**) del repositorio remoto. Es por eso, que se dice que el repositorio se encuentra distribuido entre todos los componentes del equipo de trabajo.

SVN Vs Git

- El propósito de Git es exactamente el mismo que el de SVN: La gestión del versionado del código desarrollado por un equipo de trabajo. Por tanto, las operaciones que podíamos llevar a cabo con SVN, podremos seguir haciéndolas con Git.
- Sin embargo, existen **2 diferencias sustanciales** entre SVN y Git que acarrearán implicaciones:
 - *Diferencia 1:* Con Git cada miembro del equipo cuenta con una copia local del repositorio remoto.
 - Aporta **seguridad** ya que en caso de caída del nodo central (remoto) cada desarrollador puede seguir trabajando con su repositorio local.
 - Aporta **complejidad** en el modo de operación de Git (Aparecen elementos o etapas intermedias).
 - *Diferencia 2:* Aparecen nuevas operaciones para interactuar con las etapas intermedias. Cambia la nomenclatura y significado de las operaciones que se pueden hacer con Git.
- Podemos ver **Git como un superconjunto de SVN:**



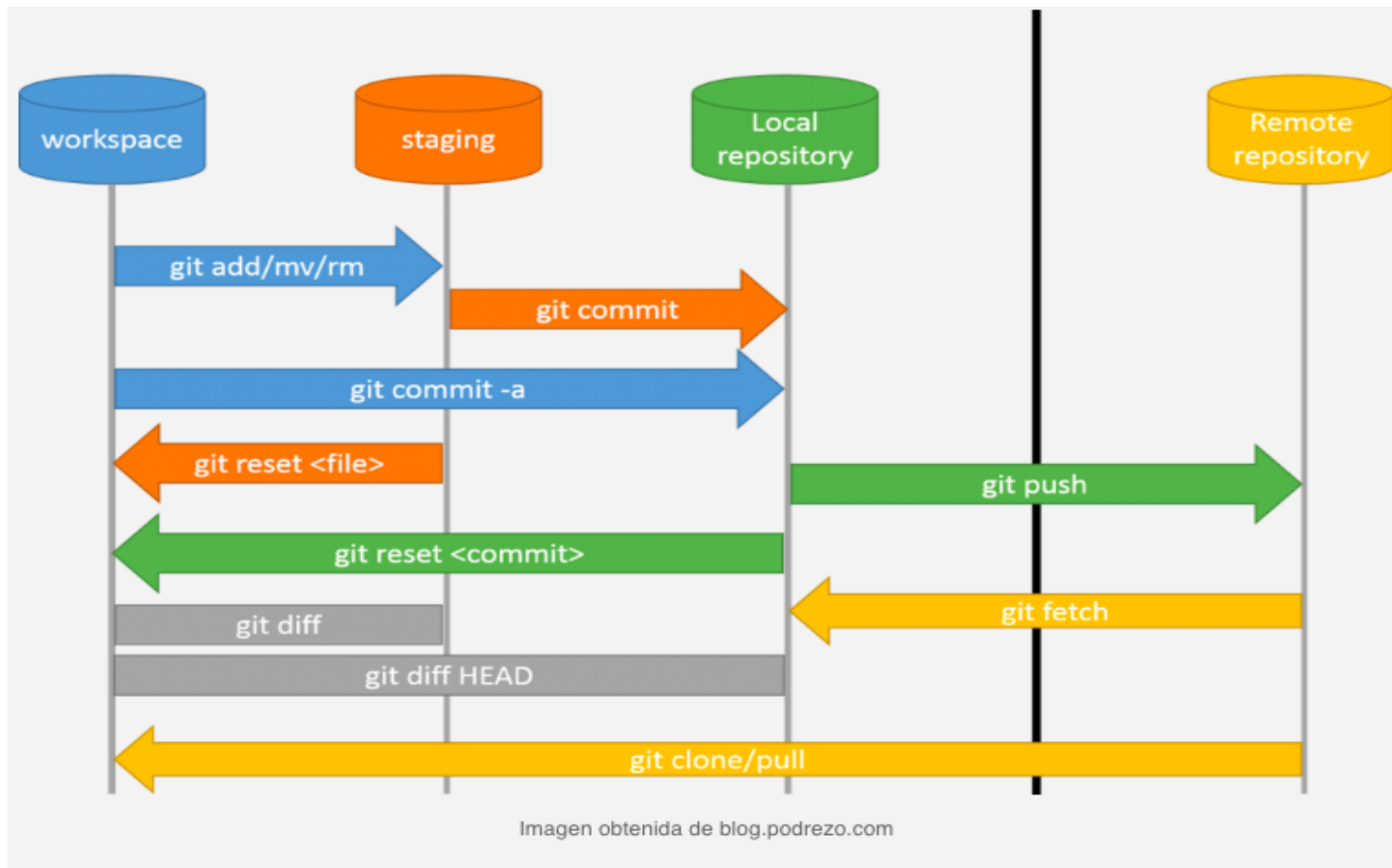
SNV Workflow (I)



SVN Workflow (II)

- **Checkout:** Operación que sólo se hace una vez al comienzo del trabajo con el repositorio remoto. Trae al espacio de trabajo local todos los recursos del repositorio remoto.
- **Synchronize:** Comparación de los cambios realizados en los recursos del espacio de trabajo local, con los posibles cambios realizados por otros desarrolladores y ya actualizados en el repositorio remoto.
- **Update:** Actualizar el espacio de trabajo local con los cambios realizados por otros desarrolladores que han actualizado el repositorio remoto. (Sólo si no hay conflictos!!!! En caso de conflictos, primero se deben resolver.)
- **Commit:** Llevar los cambios que tenemos en el espacio de trabajo local al repositorio remoto.

Git Workflow (I)



Git Workflow (II)

- Comparando los 2 flujos de trabajo observamos que los elementos que coinciden son el **Espacio de trabajo local** y el **Repositorio Remoto**.
- Vemos igualmente que aparecen nuevas etapas o elementos intermedios en el workflow de Git. Las etapas intermedias corresponden a **Staging** y **Repositorio Local**.
- *Staging*: Cuando se hacen cambios o se crean nuevos recursos en el espacio de trabajo local se deben identificar antes de llevar a cabo cualquier otra operación (normalmente *commit*) con ellos. Para ello se usa el comando *add*.
- *Repositorio Local*: Es la copia que cada desarrollador tiene del repositorio remoto. Interactuaremos con él a través de los comandos *commit*, *checkout*, *push* y *fetch*.

Git Workflow (III) - Operaciones

- De entre todas las que aparecen en el diagrama anterior usaremos las siguientes:
- **Clone**: Lleva a cabo una copia del repositorio remoto al repositorio local. Esta operación sólo se debe hacer una vez al comienzo del trabajo con el repositorio remoto.
- **Add**: Añade o identifica los ficheros modificados en el espacio de trabajo local a la zona de staging. Esta operación se debe realizar antes de llevar a cabo el commit.
- **Commit**: Lleva los ficheros de la zona de staging al repositorio local.
- **Checkout**: Operación que actualiza el espacio de trabajo local con el contenido de repositorio local. Normalmente sólo se usa en los cambios de rama.
- **Push**: Actualiza el repositorio remoto con el contenido del repositorio local.
- **Pull**: Actualiza el repositorio local con el contenido del repositorio remoto.

Git Workflow (IV) - Ejercicio

- Trata de localizar en el diagrama cada una de las operaciones explicadas en la diapositiva anterior e identifica que elementos de los que componen el sistema intervienen en cada operación.

- Por ejemplo, para la operación *add*:

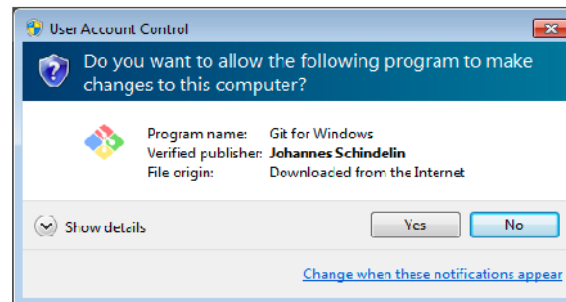
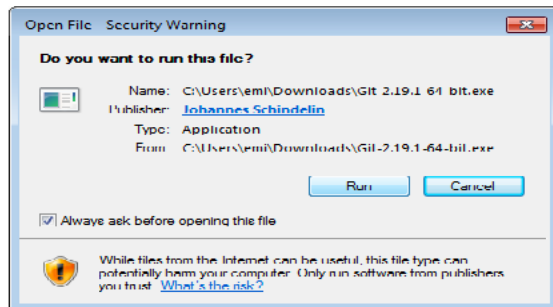
- Localización en el diagrama ->



- Descripción -> *Identifica los recursos modificados en el espacio de trabajo y que serán posteriormente 'commiteados'.*
 - Elementos que intervienen -> *Esta operación se realiza entre el espacio de trabajo local (workspace) y la zona intermedia de staging.*

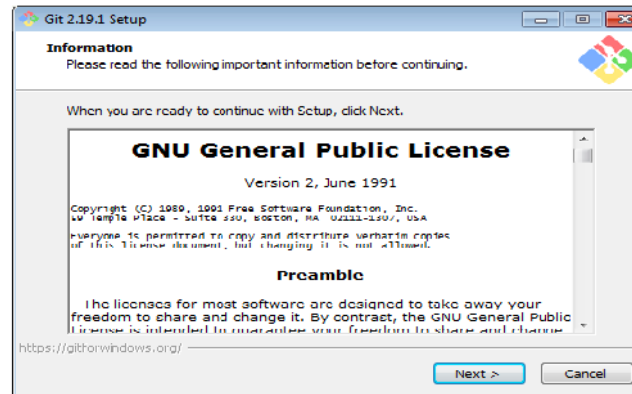
Instalación de Git (I)

- Vamos a explicar paso a paso como instalar Git en un sistema Windows.
- 1.- Descarga Git para Windows a través de éste enlace: [Git for Windows downloads](#).
- 2.- Una vez finalizada la descarga, ejecuta el fichero ejecutable y autoriza la instalación.

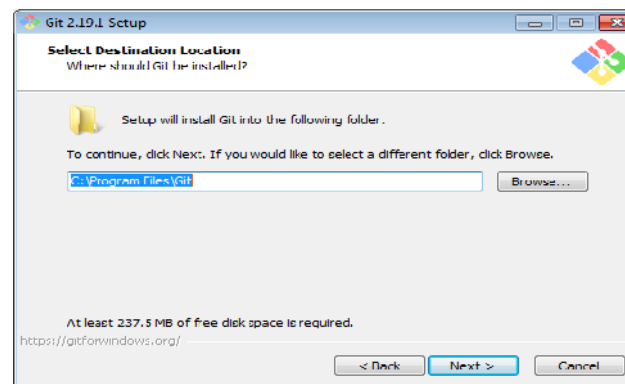


Instalación de Git (II)

- 3.- Acepta la licencia GPL

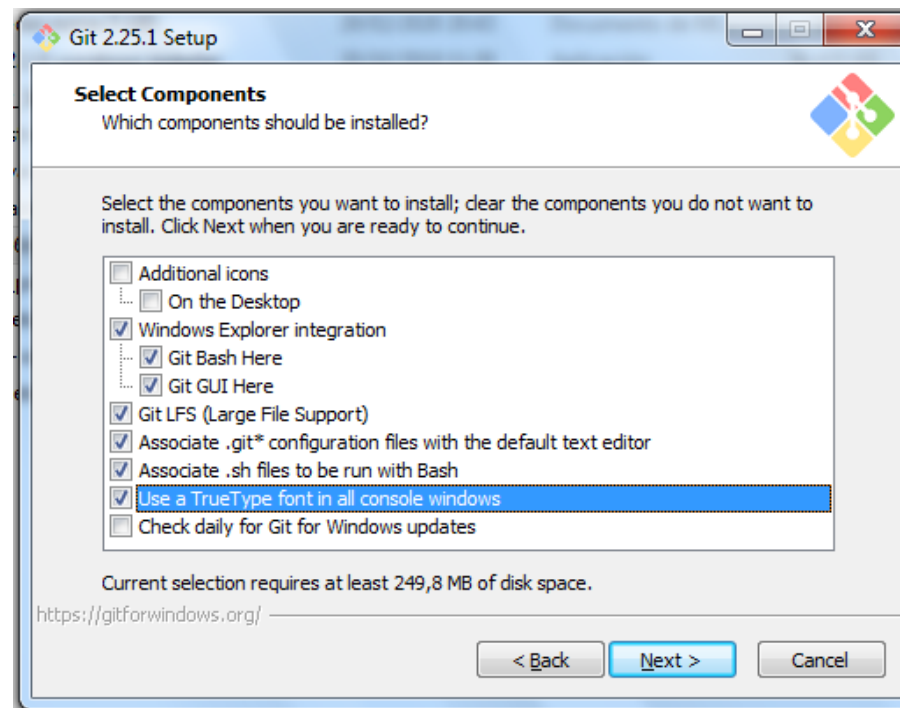


- 4.- Elije la carpeta de instalación. (Se recomienda dejar la de por defecto)



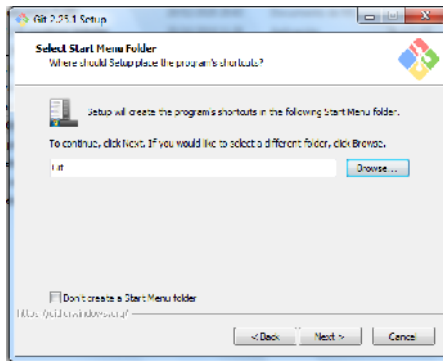
Instalación de Git (III)

- Deja los componentes de instalación por defecto, añadiendo únicamente la opción para usar fuentes TrueType.

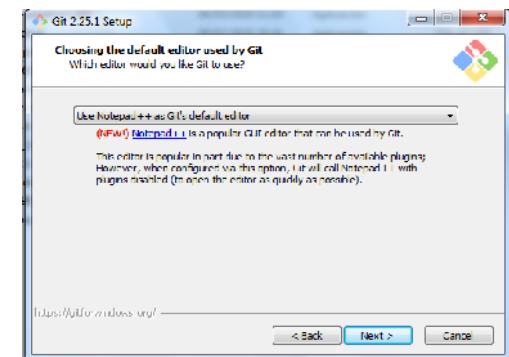


Instalación de Git (IV)

- Se le da nombre a la carpeta que se creará en el menú inicio. Podemos dejar la que se recomienda: *Git*.

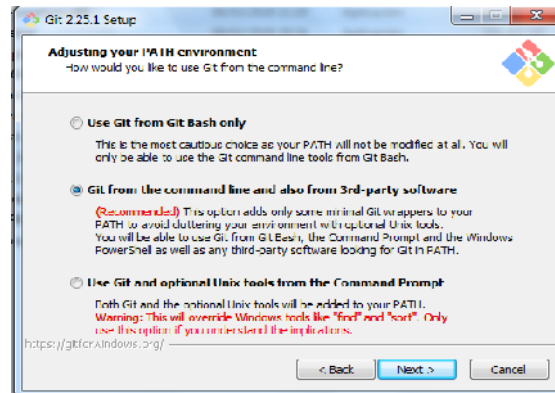


- Elije el editor por defecto usado por Git. Puedes elegir cualquiera de los que aparecen en la lista desplegable. Yo he elegido Notepad++.



Instalación de Git (V)

- En la siguiente ventana deberemos elegir usar Git tanto desde la línea de comandos como desde una aplicación de terceros. En nuestro caso esta aplicación de terceros será TortoiseGit.



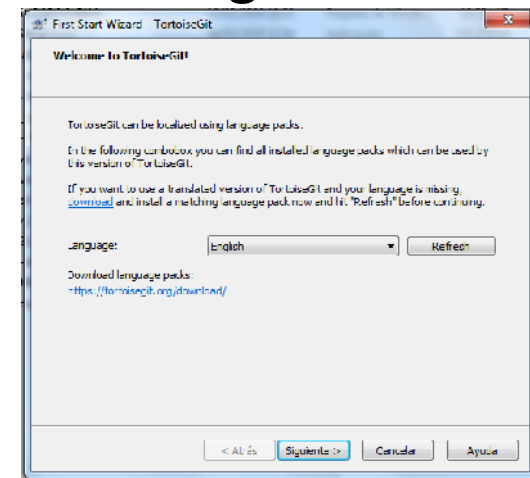
- El proceso de instalación finaliza. Puedes comprobar la instalación abriendo una consola y tecleando el comando **git --versión**.

GitHub

- GitHub será el sitio web que albergará nuestro Repositorio Remoto.
- Como usuarios del repositorio que hemos preparado para el ciclo, lo único que necesitamos es crear una cuenta en GitHub.
- Para ello accederemos por aquí [GitHub](https://github.com/join). (<https://github.com/join>)
- Aportar los datos que se piden. IMPORTANTE: USAD LA CUENTA DE CORREO DEL INSTITUTO.
- Recordar las credenciales (nombre de usuario y contraseña) de vuestra cuenta, tendréis que usarlas cuando hagáis un Pull por primera vez al repositorio remoto.
- En este punto deberéis enviarme un correo con la dirección de correo usada para la creación de la cuenta en GitHub. Os enviaré una invitación que deberéis aceptar accediendo a vuestra cuenta en GitHub.

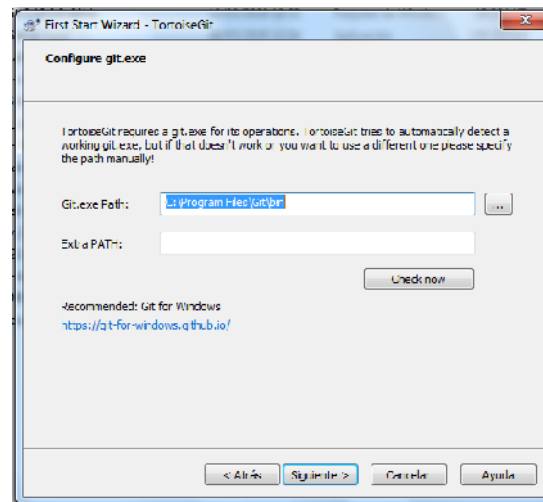
TortoiseGit – Instalación (I)

- TortoiseGit será la aplicación de terceros que usaremos para interactuar con nuestro repositorio Git.
- Para ello deberemos acceder a <https://tortoisegit.org/download/> para descargar una versión apropiada de TortoiseGit.
- Ejecutamos el archivo que se descarga y aparecerá la siguiente pantalla en la que deberemos elegir el idioma:



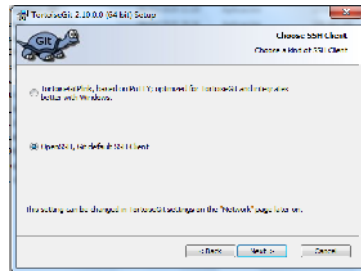
TortoiseGit – Instalación (II)

- En la siguiente ventana simplemente deberemos hacer click en siguiente.
- Posteriormente, se debe elegir la ubicación del ejecutable *Git.exe*. Si se ha instalado previamente Git, deberá aparecer la ubicación ya rellena. En caso contrario, instala en primer lugar Git antes que TortoiseGit.

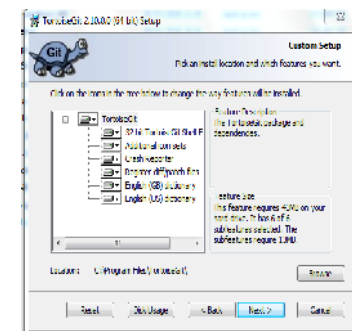


TortoiseGit – Instalación (III)

- En las siguientes ventanas haz click en siguiente hasta que llegues a la ventana en la que debes elegir el cliente SSH. Elige OpenSSH:



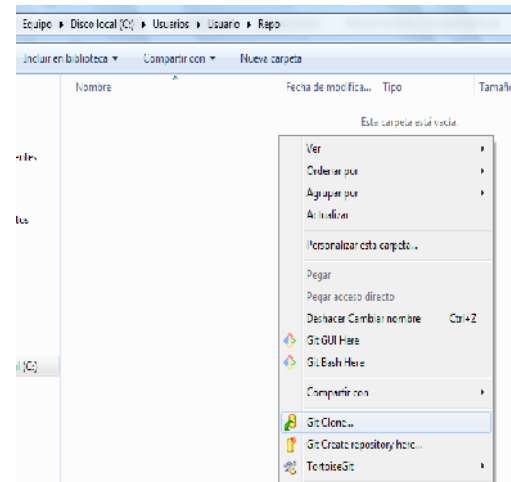
- En la siguiente ventana puedes dejar las opciones que aparecen, finalizando la instalación.



- Puedes comprobar la instalación observando las nuevas opciones del menú contextual cuando se hace click con el botón secundario del ratón sobre una carpeta.

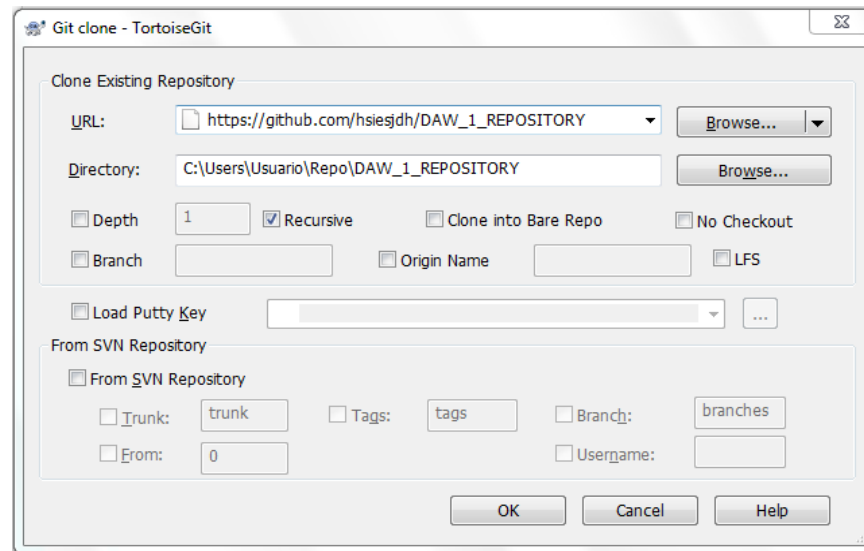
Día a día con Git - Creación y descarga del repositorio remoto (I)

- En vuestra carpeta de usuario, crear una carpeta de nombre **Repo**.
- Meteos en la carpeta y haced click con el botón derecho del ratón en un espacio en blanco, debe aparecer el menú contextual de *TorotoiseGit*.
- Elegir la opción *Git clone ...*



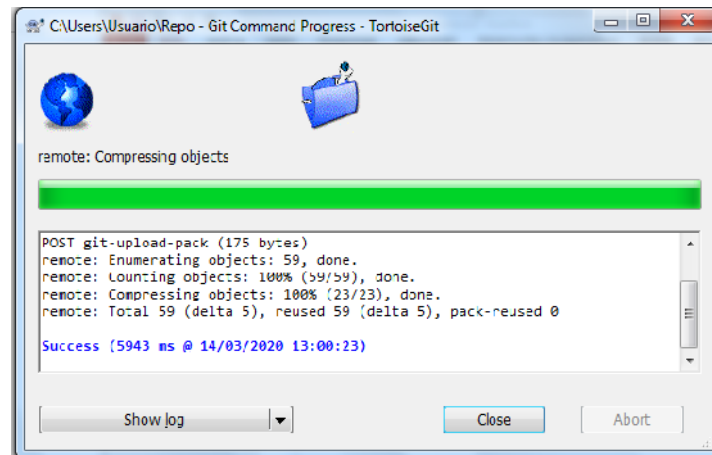
Día a día con Git - Creación y descarga del repositorio remoto (II)

- En la nueva ventana que se muestra debemos poner la URL en la que se encuentra el repositorio remoto. (https://github.com/hsiesjdh/DAW_1_REPOSITORY)
- El resto de opciones no se modifican. Pulsad sobre *OK*.



Día a día con Git - Creación y descarga del repositorio remoto (III)

- TortoiseGit procede con la conexión y descarga desde la URL aportada.







- Se trata de un repositorio remoto público, así que no debería pedir credenciales.
- **Ya tenemos el repositorio en local con la copia de trabajo, ambos completamente actualizados con la última versión.**

Día a día con Git - Estructura del repositorio (I)

- En primer lugar, notar que todos los recursos del repositorio tienen un icono asociado indicando que se encuentran bajo el control de versiones.
- En cada nivel del árbol de directories tenéis disponible un archivo *README.txt*, en el que se explica brevemente que puedes encontrar bajo esa carpeta y si es una ubicación para la entrega de tus prácticas.






Día a día con Git - Estructura del repositorio (II)

Nombre	Fecha de modifica...	Tipo	Tamaño
 DAW_1_REPOSITORY	14/03/2020 13:00	Carpeta de archivos	



Nombre	Fecha de modifica...	Tipo	Tamaño
 Doc	14/03/2020 13:00	Carpeta de archivos	
 Soluciones	14/03/2020 13:00	Carpeta de archivos	
 README	14/03/2020 13:00	Documento de tex...	1 KB

- En la carpeta *Doc* dejaré tanto la teoría como los enunciados de los ejercicios. Se encuentra dividida en los diferentes módulos que tenemos: *BD*, *Prog*, *Ent* y *Marcas*.
- En la carpeta *Soluciones*, por su parte, deberéis subir vuestras soluciones para cada ejercicio propuesto. Tenéis una carpeta individualizada para cada uno de vosotros. El formato de los nombres de carpeta se explica en el archivo *README.txt* en la carpeta destinada para cada módulo.

Día a día con Git - Estructura del repositorio (III)

Nombre	Fecha de modifica...	Tipo	Tamaño
 BD	14/03/2020 13:00	Carpeta de archivos	
 Ent	14/03/2020 13:00	Carpeta de archivos	
 Marcas	14/03/2020 13:00	Carpeta de archivos	
 Prog	14/03/2020 13:00	Carpeta de archivos	
 README	14/03/2020 13:00	Documento de tex...	1 KB

```
1 Soluciones de los ejercicios del módulo bases de datos.
2
3 Encontrarás una carpeta con tu nombre en la que deberás entregar tus trabajos.
4 El nombre identificativo de la carpeta tiene la forma:
5 XXXXXXXZ
6
7 Donde:
8 x -> Inicial del nombre de pila
9 YYYYYY -> Primer apellido completo
10 ZZ -> Las 2 primeras iniciales del segundo apellido.
11
```

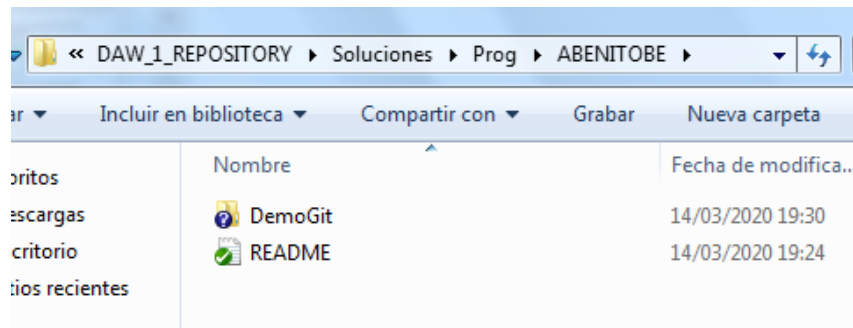
Nombre	Fecha de modifica...	Tipo	Tamaño
 ABENITOB	14/03/2020 13:00	Carpeta de archivos	
 ABERENGU	14/03/2020 13:00	Carpeta de archivos	
 ACORDERO	14/03/2020 13:00	Carpeta de archivos	
 CCOSACO	14/03/2020 13:00	Carpeta de archivos	
 CMENDEZ	14/03/2020 13:00	Carpeta de archivos	
 DBUCUR	14/03/2020 13:00	Carpeta de archivos	
 FPEREZNE	14/03/2020 13:00	Carpeta de archivos	
 GALVAREZ	14/03/2020 13:00	Carpeta de archivos	
 GHERNANDE	14/03/2020 13:00	Carpeta de archivos	
 ICHERKAOU	14/03/2020 13:00	Carpeta de archivos	
 IOCANAMO	14/03/2020 13:00	Carpeta de archivos	
 JCARVAJAL	14/03/2020 13:00	Carpeta de archivos	
 JGUSBE	14/03/2020 13:00	Carpeta de archivos	
 JSALDANAF	14/03/2020 13:00	Carpeta de archivos	
 LMORENOCA	14/03/2020 13:00	Carpeta de archivos	
 LRODRIGUE	14/03/2020 13:00	Carpeta de archivos	
 MMARMOLEJ	14/03/2020 13:00	Carpeta de archivos	
 PHERRANZ	14/03/2020 13:00	Carpeta de archivos	
 RHERRANZ	14/03/2020 13:00	Carpeta de archivos	
 WTENAZACA	14/03/2020 13:00	Carpeta de archivos	
 README	14/03/2020 13:00	Documento de tex...	1 KB

Día a día con Git - Subida de contenido al repositorio (I)

- Según el flujo de trabajo de Git para llegar desde el espacio de trabajo local hasta el repositorio remote debemos seguir una secuencia de pasos.
- Sabemos que para generar las soluciones a los ejercicios propuestos, podremos estar trabajando con *Eclipse*, *MySQL Workbench*, *Visual Studio* o vuestro editor de texto preferido. Una vez que hayáis terminado la resolución del ejercicio, deberéis coger la solución (workspace de Eclipse o carpeta src, como veáis, el archivo o archivos .sql, los html, los css, js, etc) y copiarla en vuestra carpeta de soluciones según el módulo, dentro del espacio de trabajo del repositorio.
- Por ejemplo, si el alumno *John Doe* genera una solución para un ejercicio y usa Eclipse para ello, *John* deberá copiar el proyecto Eclipse (o la carpeta src) en el espacio de trabajo del **repositorio local**, más concretamente en el path: **Repo/DAW_1_REPOSITORY/Soluciones/Prog/JOHNDOE**

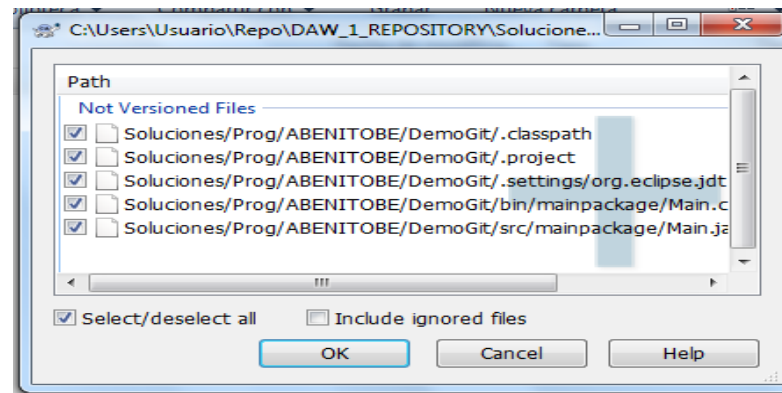
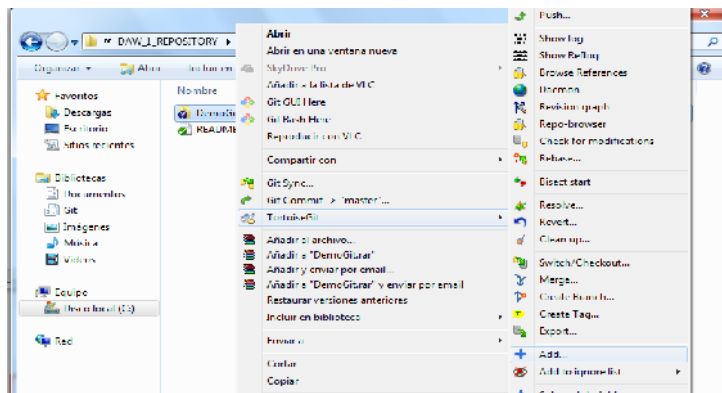
Día a día con Git - Subida de contenido al repositorio (II)

- Ahora ya tenemos la solución del ejercicio en la carpeta adecuada del espacio de trabajo del repositorio.

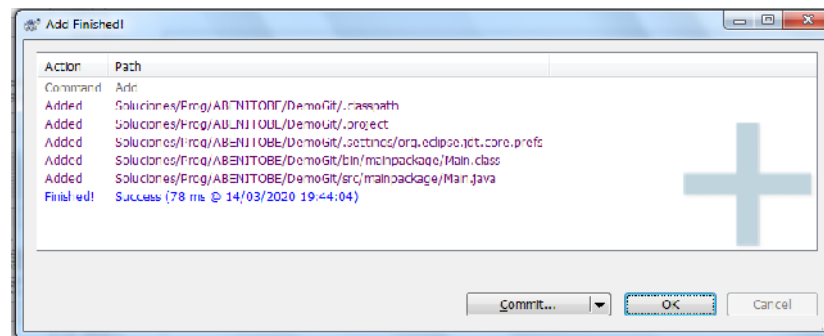


- El siguiente paso es añadir los nuevos recursos no versionados bajo el control de versiones de Git, es decir, John debe añadirlos a la zona de **Staging**. Para ello John Doe deberá hacer click con el botón derecho del ratón sobre la carpeta **DemoGit** (La solución de John). En el menú contextual elegir *TortoiseGit* y de entre todas las opciones elegir *Add...* aparecerá un popup con los recursos que se quieren versionar premarcados.

Día a día con Git - Subida de contenido al repositorio (III)

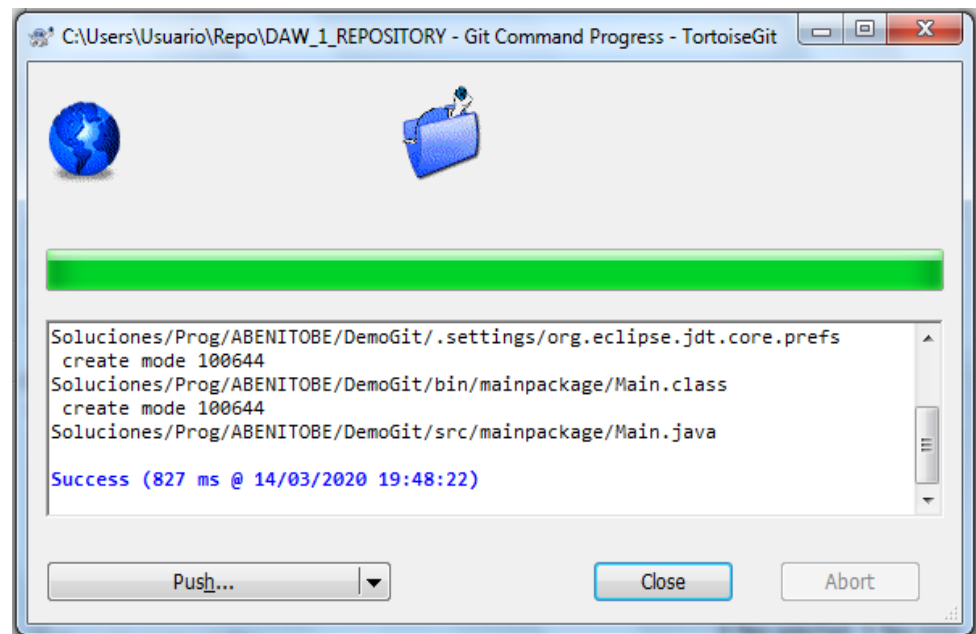
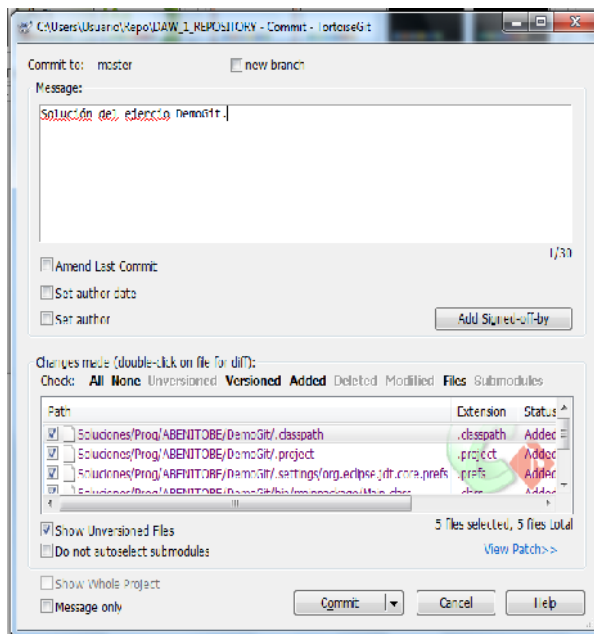


- Hacer click en le botón ok
- Se informa que los recursos han sido añadidos al Sistema de versionado.



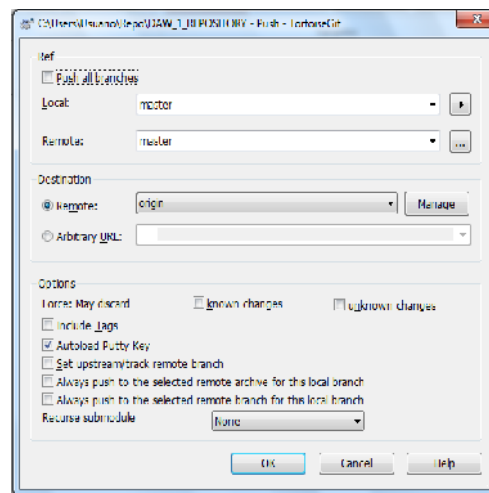
Día a día con Git - Subida de contenido al repositorio (IV)

- Podemos hacer **commit** desde esa misma ventana. Deberemos aportar un comentario explicativo y se vuelve a hacer click sobre **commit**.



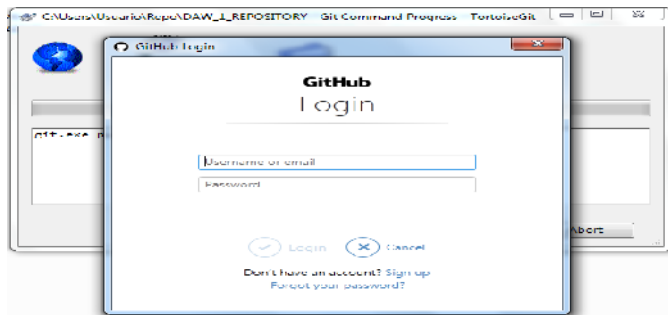
Día a día con Git - Subida de contenido al repositorio (V)

- Si todo ha ido bien, tendremos ahora el nuevo recurso en el **repositorio local**, que es lo que ocurre cuando John hace commit. Ahora deberemos subirlo al repositorio remoto.
- Para ello, en esta misma ventana haremos click sobre el boton **Push**. En la siguiente ventana dejaremos todas las opciones sin modificar y hacemos click sobre **ok**.

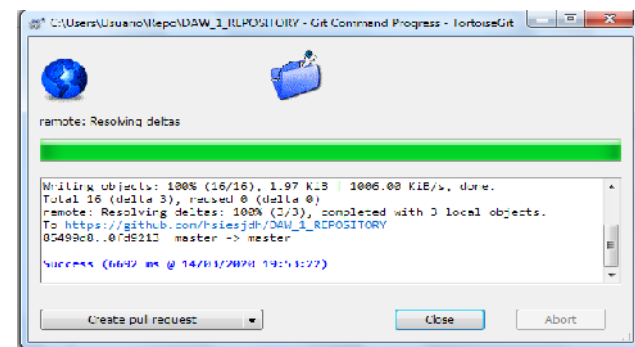


Día a día con Git - Subida de contenido al repositorio (VI)

- Hay que tener en cuenta que la primera vez que se hace un **push** os debería pedir los datos de autenticación, por lo tanto deberéis introducir vuestro nombre de usuario y la password que usastéis en el proceso de creación de vuestro cuenta en GitHub.



- Si todo ha ido bien debería aparecer una ventana como esta:

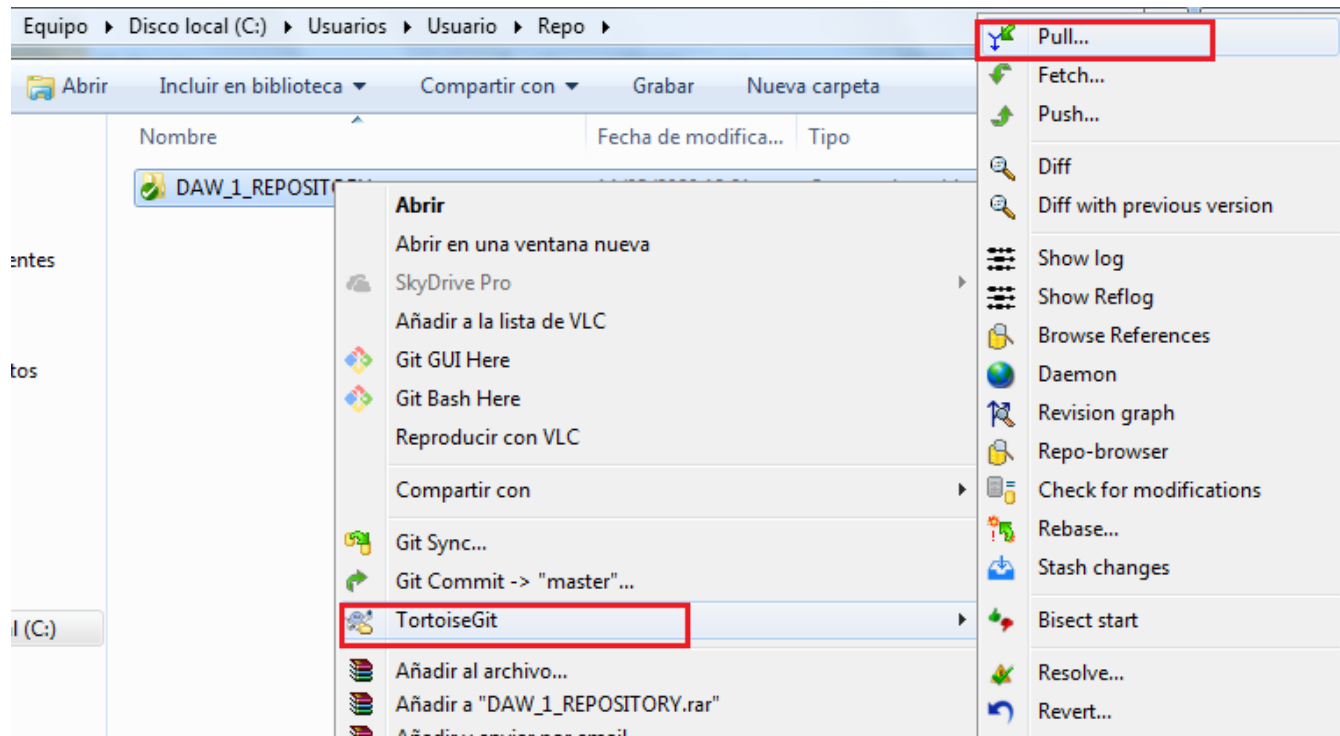


Día a día con Git – Descarga de contenido (I)

- Como ya hemos comentado podréis localizar tanto la teoría como los enunciados de los ejercicios en la carpeta correspondiente a cada módulo bajo la carpeta **Doc** del repositorio. Para ello, deberéis actualizar vuestro **Repositorio Local** y vuestro **Espacio de Trabajo Local**.
- La actualización de ambas áreas (Repo Local y Espacio de Trabajo Local) lo podemos hacer mediante la operación **pull**.
- Cada vez que queramos actualizarnos nos situaremos sobre la carpeta raíz de nuestro Espacio de Trabajo Local y haremos click con el botón derecho del ratón.

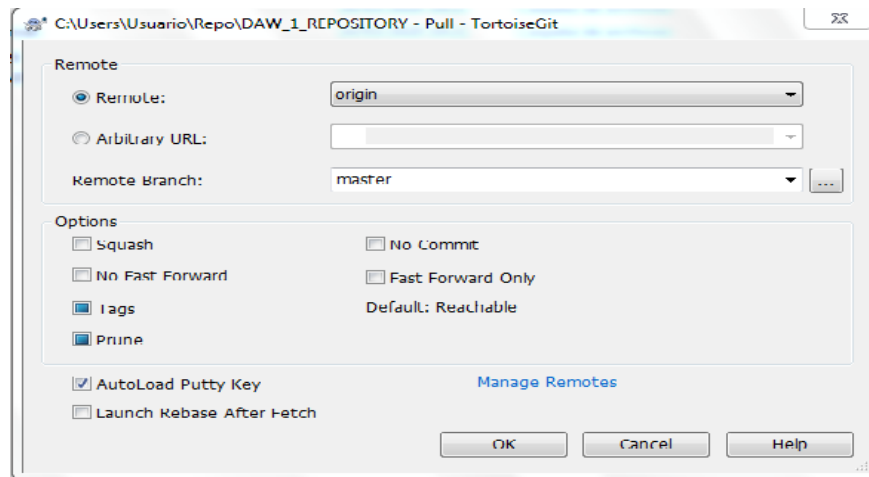
Día a día con Git – Descarga de contenido (II)

- Deberéis elegir TortoiseGit y el el submenú elegir Pull ...

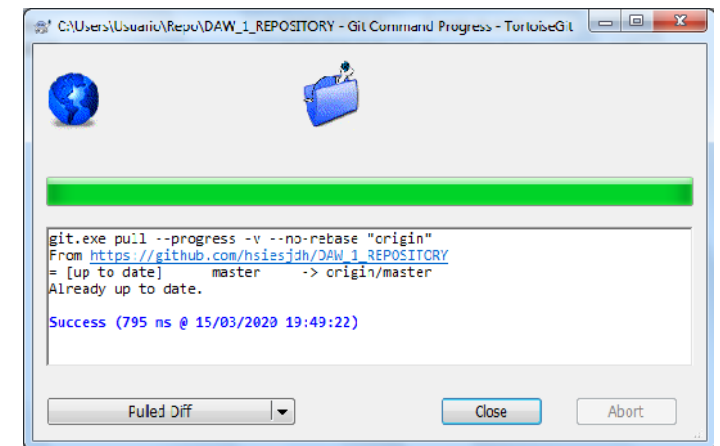


Día a día con Git – Descarga de contenido (III)

- En la siguiente ventana deberemos dejar todas las opciones tal como están y, pulsamos sobre el botón **OK**.



- La siguiente ventana será como la siguiente:



Webgrafía

- <https://git-scm.com/docs>
- https://tortoisesvn.net/docs/release/TortoiseSVN_es/
- <https://guides.github.com/>