|  |
| --- |
| Este ejercicio tiene como objetivo, trabajar con Git y GitHub. Para poder realizar este ejercicio, partimos de la premisa de que todos los alumnos ya tienen creada una cuenta en GitHub.  Simularemos un equipo de desarrolladores formado por **dos** integrantes, **User1** y **User2**, que trabajan en un proyecto común llamado **Marvel**.  Seguir paso a paso las indicaciones que se van mostrando a continuación para conseguir que la colaboración sea satisfactoria. |
| Este ejercicio se puede hacer de dos maneras:  **Individual**: Trabajando en una misma máquina y simulando de forma local la existencia de los dos compañeros.  **Grupal**: Trabajando en dos máquinas diferentes y haciendo que los dos compañeros colaboren de forma real en la distancia.  Dependiendo del formato escogido para resolver el ejercicio, el PASO 1 del ejercicio es diferente. Seleccione el formato que le interese o el que indique el/la profesor/a, y ejecute las acciones correspondientes del PASO 1. Una vez finalizado el PASO 1, puede proseguir con los siguientes pasos independientemente del formato en que este resolviendo el ejercicio. |
| **PASO 1 – RESOLUCIÓN INDIVIDUAL** |
| Si el ejercicio se realiza en el formato "Individual", lo primero que debemos hacer es crear dos directorios diferentes para identificar el espacio de trabajo de cada uno de los dos compañeros en la máquina local:  $ cd # asegurar que se encuentran en vuestro directorio casa  $ mkdir User1 # crear el directorio User1  $ mkdir User2 # crear el directorio User2  Posteriormente, crear un directorio con el nombre Marvel en el espacio de trabajo de User1 y posicionarnos en este nuevo directorio:  $ cd User1 # posicionarnos en el directorio User1  $ mkdir Marvel # crear el directorio Marvel  $ cd Marvel # posicionarnos en el directorio Marvel  A partir de este momento, dependiendo de quién tenga que realizar el siguiente PASO, previamente a realizar las acciones del PASO correspondiente, tendréis que hacer:   |  |  | | --- | --- | | PASO X - realizado por el usuario User1 | $ cd  $ cd User1/Marvel | | PASO X - realizado por el usuario User2 | En el PASO 3 se deberá hacer:  $ cd  $ cd User2  En el PASO 5 se deberá hacer:  $ cd  $ cd User2/Marvel  **Nota**: todavía no tenemos creado el directorio Marvel en el usuario User2, ya lo haremos más adelante. | |

|  |
| --- |
| **PASO 1 – RESOLUCIÓN EN GRUPO** |
| Para resolver este ejercicio en formato "Grupal", deben ser como mínimo dos personas cada una trabajando en una máquina diferente. Asignar a cada uno de los integrantes del grupo uno de los dos roles del ejercicio, el de User1 o el de User2 y seguir, en todo momento, las indicaciones que correspondan al rol que os habéis asignado. Si hace de User1 no ejecute las acciones asignadas a User2 y al revés.  El usuario con rol de User1 debe hacer:  $ cd # asegurarse de que se encuentra en su directorio casa  $ mkdir Marvel # crear el directorio Marvel  $ cd Marvel # posicionarnos en el directorio Marvel  El usuario con rol de User2 debe hacer:  $ cd # asegurarse de que se encuentra en su directorio principal |
| **PASO 2 – Realizado por el usuario User1 (Individual y Grupal)** |
| EL usuario User1 debe crearse un repositorio nuevo vacío en GitHub.  El nombre del repositorio debe ser Marvel y lo hará público.    Al **User1** le aparecerá la siguiente pantalla con la <url> del repositorio que acaba de crear, acto seguido, debe guardar esta dirección en el porta-papeles.    Las siguientes acciones del PASO 2, sólo deben realizarse si se resuelve el ejercicio en Grupo:  Ahora el **User1** añadirá a su compañero **User2** como colaborador del repositorio. Para hacer esto debe ir a la pestaña **Settings** y una vez se le despliegue el menú correspondiente, escoger la opción **Manage access**, en este punto le aparecerá el botón para invitar a colaboradores: **Invite a collaborator.**    Cuando se despliegue este menú, introducir al usuario del alumno que hace de **User2.**    Una vez invitado el colaborador **User2**, éste recibirá un email en la cuenta asociada a su usuario de GitHub con la invitación, solo tiene que ver la invitación y aceptarla, acto seguido ya tendrá acceso al repositorio remoto creado por el **User1**. |

|  |
| --- |
| **Pregunta PASO 2.1:**  Recuerde que el usuario User1 está, actualmente, ubicado en el directorio Marvel. ¿Con qué comando nos aseguramos de que User1 está en el directorio que le interesa? |
| **Respuesta:**  $ pwd |
| **Pregunta PASO 2.2:**  Ahora User1 inicializará un repositorio Git y creará dos ficheros, uno llamado movies y otro llamado personajes.  El contenido de movies debe ser:  Capitana Marvel  Vengadores: Infinity War  Y el contenido de personajes debe ser:  Thor  Iron Man  Indica los comandos necesarios para llevar a cabo estas acciones: |
| **Respuesta:**  echo -e "Capitana Marvel\nVengadores: Infinity War" >> movies && echo -e "Thor\nIron Man" >> personajes |
| **Pregunta PASO 2.3:**  A continuación, User1 hace dos instantáneas en el repositorio local, uno por cada fichero que acaba de crear. La primera instantánea irá acompañada del mensaje "Creación movies" y la segunda instantánea irá acompañada del mensaje "Creación personajes". |
| **Respuesta:**  git add movies && git commit -m "Creacion Movies"  git add personajes && git commit -m "Creacion Personajes" |
| **Pregunta PASO 2.4:**  Acto seguido, User1 verifica que los dos snapshots se han creado correctamente utilizando el comando:  $ git log --oneline #verifique los dos commits  Observar la información que nos muestra este comando y explica el significado de los diferentes punteros que aparecen. |
| **Respuesta:**  10969b2 (HEAD -> master) Creacion Personajes  8aa563e Creacion Movies  The pointers represent the instantaneas and are a unique identifier. The pointer with (head -> master) is the one  that is currently used. Head references the currently used branch, and master is the name of the branch. |

|  |
| --- |
| **Pregunta PASO 2.5:**  User1 enlaza el repositorio local que acaba de crear con el repositorio remoto que ha creado antes en GitHub, verifica que se ha enlazado bien y revisa de nuevo los commits hechos: |
| **Respuesta:**  git remote add origin <https://github.com/LiukPeer/Marvel.git> && git remote -v  git log --oneline |
| **Pregunta PASO 2.6:**  User1 sube su repositorio local (rama master) con los cambios que ha hecho en el repositorio remoto: |
| **Respuesta:**  git push origin master |
| **Pregunta PASO 2.7:**  A continuación, User1 refresca el contenido de su repositorio remoto en GitHub para verificar que se ha actualizado correctamente:  Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente  Una vez verificado, vuelve a revisar los commits recientemente hechos en su repositorio local. ¿Qué diferencia aprecias respecto a la orden de Git ejecutado en la Pregunta PASO 2.4? |
| **Respuesta:**  10969b2 (HEAD -> master, origin/master) Creacion Personajes  8aa563e Creacion Movies  The difference is that now the origin/master label is shown. This means that the online repository is on the same version as the head |
| **Pregunta PASO 2.8:**  User1 será encargado de actualizar las películas del fichero movies y, por ello, se crea una rama llamada bmovies para poder actualizar estos datos sin afectar a la rama máster. Acto seguido se coloca en esta rama para más adelante, actualizar el fichero movies y verifica que, efectivamente, está en la rama que le interesa estar. |
| **Respuesta:**  git branch bmovies && git checkout bmovies  echo “Guardianos de la galaxia” » movies && git add movies && git commit -m “GDG anadido”  git push origin bmovies |

|  |
| --- |
| **PASO 3 – Realizado por el usuario User2 (Individual y Grupal)** |
| **Pregunta PASO 3.1:**  User2 no tiene todavía ningún repositorio creado en su espacio de trabajo, pero quiere colaborar con User1 y poder ampliar los datos de los personajes de las películas.  Así pues, se clona el repositorio remoto que ha creado antes User1 para poder trabajar en el proyecto Marvel: |
| **Respuesta:**  git clone https://github.com/LiukPeer/Marvel.git |
| **Pregunta PASO 3.2:**  Una vez que la clonación ha finalizado con éxito, se ubica en el directorio Marvel donde verifica que tiene los ficheros que había creado el User1 y, también, revisa las instantáneas que hay en el repositorio haciendo git log --oneline |
| **Respuesta:**  cd Marvel  ls -a |
| **Pregunta PASO 3.3:**  Cuando User2 revisa cómo está el árbol de commits que se acaba de bajar del repositorio remoto le aparece algún puntero extra que no le aparecía a User1, indica cuál es y qué significado tiene. |
| **Respuesta:**  Origin/HEAD appeared. It means that the branch we are using in the origin is the main one |
| **Pregunta PASO 3.4:**  Acto seguido, como el trabajo encargado a User2 es el de mejorar los datos de los personajes de las películas, se crea una rama local llamada bpersonajes para trabajar sobre el fichero personajes sin afectar a la rama master. |
| **Respuesta:**  git branch bpersonajes |
| **Pregunta PASO 3.5:**  Una vez a verificado que la rama ha sido correctamente creada, cambia a la rama bpersonajes y verifica que cambió. |
| **Respuesta:**  git checkout bpersonajes  git branch |

|  |
| --- |
| **Pregunta PASO 3.6:**  User2 empieza a actualizar el fichero personajes añadiendo:  Spider-man  Hulk  Thanos  Y, acto seguido, actualiza su repositorio local con una instantánea que incluyan los cambios realizados. El mensaje asociado a esta instantánea es "User2: cambio 1 a personajes". |
| **Respuesta:**  echo -e “Spider-man\nHulk\nThanos” » personajes  git add personajes && git commit -m "User2: cambio 1 a personajes" |
| **Pregunta PASO 3.7:**  Verifica que la instantánea se ha creado correctamente: |
| **Respuesta:**  git status && git log |
| **Pregunta PASO 3.8:**  ¿A dónde apunta el puntero HEAD ahora? |
| **Respuesta:**  It points to the bpersonajes branch |
| **Pregunta PASO 3.9:**  User2 quiere incluir más información y decide añadir al proyecto el nombre de los actores, así pues, vuelve a la rama master del proyecto y crea una rama llamada bactors donde gestionará esta parte. Verifica que la rama se ha creado correctamente. |
| **Respuesta:**  git checkout master  git branch bactors |

|  |
| --- |
| **Pregunta PASO 3.10:**  User2 se cambia a la rama y crea un fichero llamado actors donde incluye los nombres de los siguientes actores:    Robert Downey Jr.  Chris Hemsworth   Tom Holland  Bruce Banner   Scarlett Johansson  Una vez hecha su base de datos de actores en la rama bactors, verifica el estado del repositorio: |
| **Respuesta:** |
| **Pregunta PASO 3.11:**  A continuación, hace un commit para actualizar el repositorio acompañado del mensaje "Crear actors" y vuelve a revisar que la nueva instantánea se haya creado correctamente. |
| **Respuesta:** |
| **Pregunta PASO 3.12:**  Ahora ha llegado el momento de subir los cambios hechos por User2 al repositorio remoto. Para llevar a cabo esta acción, User2 se coloca en la rama master y hace un push de la rama master, ¿cuál es el resultado? |
| **Respuesta:**  The result is that everything is up to date, because the branche modified was bactors not master |
| **Pregunta PASO 3.13:**  Git nos indica que la rama master del repositorio de User2 está igual que la del repositorio remoto,  Para poder ver los cambios hechos por User2 en su repositorio local en el repositorio remoto, ¿qué debe hacer User2? |
| **Respuesta:**  User2 needs first to push the changes of each branch to origin |

|  |
| --- |
| **Pregunta PASO 3.14:**  Ahora User2 refresca el repositorio remoto en GitHub y aparece lo siguiente:  Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto  Descripción generada automáticamente  Una vez User2 ya ve sus aportaciones al repositorio remoto hay que tener en cuenta que:  Para ver el contenido de las ramas añadidas, se debe desplegar la pestaña "master", entonces, haciendo click sobre la rama que nos interese se actualiza la pantalla principal con el contenido de la rama escogida.  **Importante**: **NO** pulsar en ningún momento los botones verdes **<Compare & pull request>** , ya desaparecerán cuando hagamos las fusiones de las ramas en local y subimos los cambios al repositorio remoto.  Como curiosidad, podéis ir a la opción "Insights" y escoger del listado de opciones de la izquierda la opción "Network" y os aparecerá el grafo de commits que hay almacenado en GitHub.    Una vez revisada esta opción, volver a apretar "<> Code " |

|  |
| --- |
| **PASO 4 – Realizado por User1 (Individual y Grupal)** |
| **Pregunta PASO 4.1:**  User1 quiere actualizar su repositorio local con los cambios que ha subido User2 en el repositorio remoto. Para hacerlo primero revisa cuál es la rama de trabajo actual y cómo está el árbol de instantáneas de su proyecto. |
| **Respuesta:** |
| **Pregunta PASO 4.2:**  Acto seguido, define como rama de trabajo la rama master y decide bajarse esta rama del repositorio remoto al repositorio local. ¿Cuál es el resultado? |
| **Respuesta:** |
| **Pregunta PASO 4.3:**  ¿Qué ramas debe bajarse User1 y qué orden de git debe utilizar para hacerlo? Ten en cuenta que no queremos que se fusionen con la rama master, sino que queremos bajarlas abiertas !! |
| **Respuesta:**  I need to first checkout the missing branches and then pull from origin the correct branch  git fetch && git checkout -b bactors origin/bactors |
| **Pregunta PASO 4.4:**  Finalmente, el User1 se baja las ramas correctas y por cada rama bajada verifica el estado de su repositorio. |
| **Respuesta:** |
| **Pregunta PASO 4.5:**  Observando el grafo obtenido en el paso anterior, las ramas bactors y bpersonajes ¿qué peculiaridad tienen? ¿Qué tiene que hacer User1 para poder trabajar con ellas de forma local? |
|  |

|  |
| --- |
| **Pregunta PASO 4.6:**  Ahora User1 actualiza el repositorio remoto con su rama bmovies, pues ha detectado que no está subida y vuelve a revisar su repositorio local utilizando el comando: git log –-oneline –graph --all  ¿Qué diferencia hay en el grafo respecto al obtenido en el apartado anterior? |
| **Respuesta:**    The difference is that this command shows all the branches at the same time |
| **Pregunta PASO 4.7:**  Acto seguido, User1 añade películas nuevas al fichero movies desde la rama bmovies, hace un commit de los cambios hechos con el mensaje asociado "Añadir películas". A continuación revisa que el commit se haya hecho correctamente con el comando git -–oneline –-graph –-all y actualiza el repositorio remoto. Los nuevos títulos que añade son:  Vengadores: Endgame  Pantera negra  Capitán América  Doctor Extraño |
| **Respuesta:** |
| **Pregunta PASO 4.8:**  El User1 revisa de nuevo cómo está su repositorio haciendo git -–oneline –-graph –-all, ¿qué diferencia hay ahora respecto a antes de subir los últimos cambios? |
| **Respuesta:**  There is a more recent instantanea on the bmovies branch, but it’s not uploaded yet to origin |

|  |
| --- |
| **Pregunta PASO 4.9:**  En este punto, User1 actualiza y revisa el repositorio remoto y, en particular, el grafo que ha generado hasta ahora, donde se ve la evolución de la rama máster y la de las tres ramas que se han ido creando en el proyecto Marvel:    El proyecto poco a poco llega a su final. User1 fusiona la rama bmovies con la rama master local y elimina la rama bmovies. Una vez todo verificado, lo sube al repositorio remoto: |
| **Respuesta:**    **After this ‘git push origin master’** |
| **Pregunta PASO 4.10:**  User1 revisa cómo ha quedado el grafo del repositorio de GitHub, ¿qué particularidad observa? |
| **Respuesta:**    The inconsistency is that bmovies is still present online and is noted as outdated |
| **Pregunta PASO 4.11:**  ¿Qué debería hacer User1 para solucionar esta inconsistencia? |
| **Respuesta**:  We need to delete the branch in the repository as well.    git push origin --delete bmovies |
| Para ir cerrando este proyecto, sólo nos queda que User2 fusione sus cambios locales con la rama master, subir las fusiones al repositorio remoto y conseguir que User1 vea en su repositorio local estas actualizaciones del repositorio remoto. |

|  |
| --- |
| **PASO 5 – Realizado por el usuario User2 (Individual y Grupal)** |
| **Pregunta PASO 5.1:**  User2 revisa cuál es su rama de trabajo actual, mira el contenido de los ficheros personajes i actors en la rama actual en la que está. |
| **Respuesta:** |
| **Pregunta PASO 5.2:**  A continuación, cambia a la rama bactors, revisa el contenido del fichero actors. Después cambia a la rama bpersonajes y revisa el contenido del fichero personajes. |
| **Respuesta:** |
| **Pregunta PASO 5.3:**  User2 detecta que no tiene todo el repositorio actualizado, le falta el trabajo de User1. Como sabe que User1 lo ha subido todo al repositorio remoto, decide bajarse los cambios y ponerse al día. Por lo tanto, cambia a la rama master y se baja los cambios del repositorio remoto a local. Acto seguido revisa cómo le ha quedado el grafo de su repositorio local. |
| **Respuesta:**  git checkout master  git pull origin master |
| **Pregunta PASO 5.4:**  Finalmente, User2 fusiona las ramas locales bactors y bpersonajes con master y actualiza el repositorio remoto. ¿Cuál es el resultado? |
| **Respuesta:** |
| **Pregunta PASO 5.5:**  User2 revisa el grafo del repositorio remoto para ver cómo va evolucionando. |
|  |

|  |
| --- |
| **PASO 6 – Realizado por User1 (Individual y Grupal)** |
| **Pregunta PASO 6.1:**  Para dar por terminado el proyecto, User1 se baja a su repositorio local los cambios que ha subido User2. |
|  |
| Finalmente, User1 y User2 tienen sus repositorios locales y el repositorio remoto totalmente iguales y con el proyecto finalizado. |