Trabajo Practico Nª 2 – Python

Sandro Miguel Campisi

Git GithubCampisi1912

Carpeta publica: Programa

Efectuados los reuqerimientos de commit y push

* ¿Que es git?

es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.")

* ¿Que es github?

es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de computadora. El software que opera GitHub fue escrito en Ruby on Rails. Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc.

* ¿Que es gitlab?

es un servicio web de control de versiones y desarrollo de software colaborativo basado en Git. Además de gestor de repositorios, el servicio ofrece también alojamiento de wikis y un sistema de seguimiento de errores, todo ello publicado bajo una Licencia de código abierto.

* Diferencias

Desde hace pocas fechas la versión gratuita de GitHub permite crear repositorios privados y gratuitos. Esto es algo que ya ofrecía GitLab. Sin embargo existen diferencias entre ambos servicios. No son muy grandes, pero sí puede hacer que algunos usuarios se decanten por una u otra, según lo que necesiten. Esta diferencia reside en el número máximo de colaboradores por repositorio. GitHub permite crear repositorios privados de manera ilimitada, pero con un máximo de 3 colaboradores. Por su parte GitLab permite también crear repositorios privados ilimitados y gratuitos, pero en esta ocasión no hay límite de colaboradores.

* ¿Como se usa y cual es su propósito general?

se puede usar en un servidor propio, incluye varias herramientas casi imprescindibles para cualquier proyecto como gestión de peticiones para recoger las historias, wiki como documentación e información y pages para generar pequeños sitios web además de los repositorios de Git o el servidor de integración y entrega continua. Que GitLab incluya por defecto todas estas herramientas hace innecesario recurrir en la mayoría de los casos a varias herramientas especializadas que cubran estas necesidades como JIRA, Jenkins, MediaWiki o un servidor web para los sitios web.

* ¿Quienes pueden usar sus datos?.

Los usuarios en Gitlab son las cuentas que abre la gente. Las cuentas de usuario no tienen ninguna complicación: viene a ser una colección de información personal unida a la información de login. Cada cuenta tiene un espacio de nombres (namespace) que es una agrupación lógica de los proyectos que pertenecen al usuario. De este modo, si el usuario jane tiene un proyecto llamado project, la URL de ese proyecto sería http://server/jane/project.

Para trabajar en un proyecto GitLab lo más simple es tener acceso de escritura (push) sobre el repositorio git. Puedes añadir usuarios al proyecto en la sección “Members” de los ajustes del mismo, y asociar el usuario con un nivel de acceso (los niveles los hemos visto en Grupos). Cualquier nivel de acceso tipo “Developer” o superior, permite al usuario enviar commits y ramas sin ninguna limitación.""")

* ¿Que proyectos colocarias en github y cuales en gitlab?¿Porque?

Con GitLab es que podemos montarnos una réplica de un producto on premise, es decir, en nuestra instalación.

Si formamos parte de una estructura empresarial que es celosa de albergar su código fuente en un tercero, por problemas o necesidades varias,

podemos montarnos en nuestra instalación este mismo servicio tal cual y es gratuito..

con GitHub : Los repositorios públicos quedan igual: gratis y con colaboradores ilimitados.

Bajo el nombre GitHub Enterprise se unirán el antiguo Enterprise Cloud y Enterprise Server, y al precio de uno solo. pero solo te permite repositorios privados bajo suscripción.

* ¿Cuales son sus principales comandos?¿ como se usan?

git help

Muestra una lista con los comandos más utilizados en GIT.

git init

Podemos ejecutar ese comando para crear localmente un repositorio con GIT y así utilizar todo el funcionamiento que GIT ofrece. Basta con estar ubicados dentro de la carpeta donde tenemos nuestro proyecto y ejecutar el comando. Cuando agreguemos archivos y un commit, se va a crear el branch master por defecto.

git add + path

Agrega al repositorio los archivos que indiquemos.

git add -A

Agrega al repositorio TODOS los archivos y carpetas que estén en nuestro proyecto, los cuales GIT no está siguiendo.

git commit -m "mensaje" + archivos

Hace commit a los archivos que indiquemos, de esta manera quedan guardados nuestras modificaciones.

git commit -am "mensaje"

Hace commit de los archivos que han sido modificados y GIT los está siguiendo.

git checkout -b NombreDeBranch

Crea un nuevo branch y automaticamente GIT se cambia al branch creado, clonando el branch desde donde ejecutamos el comando.

git branch

Nos muestra una lista de los branches que existen en nuestro repositorio.

git checkout NombreDeBranch

Sirve para moverse entre branches, en este caso vamos al branch que indicamos en el comando.

git merge NombreDeBranch

Hace un merge entre dos branches, en este caso la dirección del merge sería entre el branch que indiquemos en el comando, y el branch donde estémos ubicados.

git status

Nos indica el estado del repositorio, por ejemplo cuales están modificados, cuales no están siendo seguidos por GIT, entre otras características.

git clone URL/name.git NombreProyecto

Clona un proyecto de git en la carpeta NombreProyecto.

git push origin NombreDeBranch

Luego de que hicimos un git commit, si estamos trabajando remotamente, este comando va a subir los archivos al repositorio remoto, específicamente al branch que indiquemos.

git pull origin NombreDeBranch

Hace una actualización en nuestro branch local, desde un branch remoto que indicamos en el comando.

Aporte en grupo:

* ¿Que utilidad encontras git?

es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.

* ¿Que ventajas y desventajas tiene usar git como sistema de seguridad?.
* **Seguridad** : Pero… ¿y si tengo un sistema de control de versiones instalado en local y tengo una caída de la tensión eléctrica que provoca el apagado de la infraestructura? ¿Los datos se habrán corrompido en el proceso de escritura? La respuesta es que resultaría muy complejo, ya que en su inmensa mayoría, los controladores de versiones cuentan con sistemas de cifrado y otros tipos de medidas de seguridad que se aplicarán para que nuestros datos permanezcan lo más íntegros posible. Por poner un ejemplo, Git hace uso de **sistemas de árbol SHA1** , lo que asegurará que hasta que no se realice la comprobación del cifrado o firma, los cambios no se escribirán en el servidor.
* ¿Que ventajas y desventajas tiene usar git como sistema de trabajo en grupo?

**Ventajas**

* **Compartición selectiva** : El desarrollo de la aplicación serán únicamente nuestro, pudiendo decidir qué parte de nuestro proyecto compartimos y con quién, restringiendo a que sólo pueda verlo, que tenga la posibilidad de añadir notas, comentarios o que pueda añadir cambios. No todo tiene porque ser público (aunque en la compartición encontramos la riqueza y agilidad que fundamenta el software libre ;).
* **Velocidad** : Muchas empresas deciden implementar GIT como servicio local en su infraestructura física, por lo tanto el control de versiones se realizaría dentro de la propia red con la consiguiente ganancia en velocidad de acceso y escritura, así como eliminando el requisito de contar con una conexión a internet obligatoria. No obstante (como ya se verá en el [curso de Git, GitHub y GitLab](https://openwebinars.net/cursos/git-github/) , contar con el respaldo de un hosting para estos menesteres siempre es un plus de seguridad pues ganaremos la despreocupación para con respecto de la seguridad y accesibilidad de nuestro trabajo.
* **Ramificación** : Ya hablamos de la ramificación [en una entrada anterior](https://openwebinars.net/blog/git-que-es-como-funciona/) , y como vimos ofrece un amplio abanico de posibilidades a la hora de realizar cambios en la estructura principal, pudiendo crear diferentes ramas sobre las que aplicar nuestras modificaciones en entornos aislados de la línea principal de desarrollo.
* **Convergencia** : Si en la creación de una rama del proyecto encontramos que uno de los cambios incluidos se integra tal y como deseamos, sin presentar conflictos con las diferentes partes de nuestra aplicación, podremos incluir o hacer converger dicha ramificación con el desarrollo principal de forma sencilla y segura, contando así con una nueva versión o revisión de nuestro proyecto lista para ser distribuida, compartida, liberada…
* **Sandbox** : Esta sería una ventaja de una ventaja, ya que las ramificaciones nos preparan un entorno aislado de pruebas sobre el desarrollo de la línea central de nuestra app. Los cambios realizados en una de las ramas del proyecto no tendrán consecuencias para los usuarios que actualmente usen o accedan a la versión sin modificar o principal. Muy útil si lo que vamos a hacer es incluir servicios que antes no existían en nuestra aplicación y esto pudiese poner en peligro la estabilidad de otros componentes.
* **Flujo de trabajo adaptable** : En el sector de los controladores de versiones encontraremos diferentes formas para gestionar el flujo de desarrollo de la aplicación, destacando entre éstos los modelos centralizados y los modelos de libre configuración. Con esto encontramos que trabajemos como trabajemos encontraremos un control de versiones que se adapte a nosotros o nuestra empresa, haciendo uso desde un simple sistema jerárquico hasta un ligeramente más complejo sistema centralizado. En este apartado Git gana puntos sobre la competencia, admitiendo multitud de configuraciones que nos permitirán dentro de su estructura organizar el trabajo tal y como deseemos nosotros, nuestro equipo de desarrollo, etc…

* **Coste** : Obviamente podremos encontrar software de control de versiones que nos ofrezcan las mismas o similares características que nos otorga Git o BitBucket entre otros, pero lo que será complejo es que alguna de estas alternativas sea gratuita. Los sistemas de hosting pueden llegar a presentar algún coste, pero entrará en nuestro juicio contemplar si realizar una pequeña inversión en infraestructura online (asegurando ya de paso un poco más el acceso y seguridad de nuestro proyecto) o adquirir una costosa solución similar a los ejemplos ya citados. Poniéndonos en la piel de un empresario, creo que cuanto más consigamos ahorrar a la empresa mejor para la empresa (y para nosotros demostrando preocupación por los recursos económicos de la misma ;).

**Desventaja**

* **Aprendizaje** : Es algo por lo que tendremos que pasar cada vez que queramos incluir alguna nueva tecnología en un flujo de trabajo ya establecido. Deberemos formar al equipo de desarrollo o a aquellas personas a las que tendremos que dar acceso a nuestro trabajo en el servicio de control de versiones; para evitar así posibles errores en la realización de cambios (salvables por su puesto al estar ahí el registro de cambios y versiones.