Unidad 1 / Escenario 2

Lectura fundamental

Manejo de repositorios

Contenido

Palabras clave: ambientes de desarrollo, pruebas, versionado y producción.

1. Control de versiones con GIT

El control de versiones es un método que permite la administración de cambios a través del tiempo de archivos o carpetas, a pesar de que generalmente se hace referencia a administración de código fuente, y es muy utilizado en el desarrollo de software, puede ser utilizado para la administración de texto, escritos, notas y versionamiento de documentos.

En el mejor de los casos, puede ser de gran utilidad el hecho de que cada departamento en una empresa considere el manejo de control de versiones para administrar sus documentos; por ejemplo: si eres el contador de la empresa y has hecho cambios en el balance de la semana, pero tal vez cambiaste algunos valores de los que te arrepientes, con el manejo de versiones, puedes remitirte a versiones anteriores de tu documento y salvar lo que creías perdido.

> ¿Sabía qué...?
>
> Si quiere revertir un documento a lo que era hace unos días, git te da herramientas de uso sencillo para hacerlo.

Una forma arcaica en las que las personas hacían uso de versiones anteriores de cada archivo, era añadiendo copias y copias del mismo archivo en un disco, o un espacio específico en memoria; en el mejor de los casos agregaban la fecha de creación a cada archivo para llevar un "control de versiones". Frente a este desorden que se ocasionaba, en caso de que el disco tuviera una falla, los archivos dejarían de servir, a partir de esto surgieron los sistemas de control de versiones distribuido (Distributed Version Control Systems o DVCSs en inglés).

En un DVCS (como Git, Mercurial, Bazaar o Darcs), los clientes no tienen únicamente acceso a la última versión creada en el repositorio (el lugar donde ha sido alojado cada uno de los archivos de los que se ha hecho un versionamiento), los clientes descargan todo el repositorio desde el servidor, a cada uno de sus computadores; así, en caso de que el servidor presente una falla, de igual forma existirá una copia total del repositorio en cada uno de los computadores que han colaborado con el proyecto, y de esta forma se podría realizar una copia en el servidor para restaurarlo. (Véase figura 1).

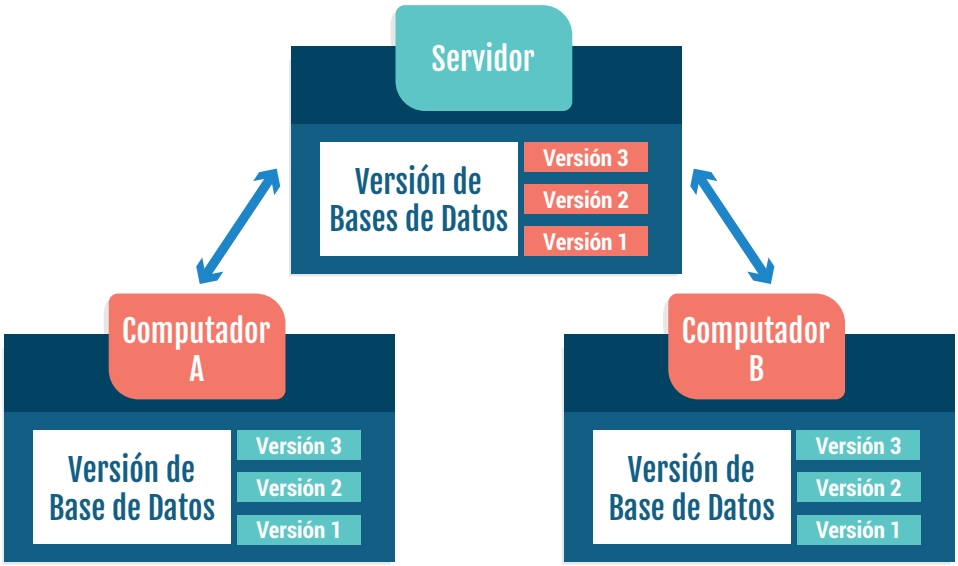

Figura 1. Estructura de repositorios

Fuente: Elaboración propia (2017).

El control de versiones ha tenido una gran acogida en el desarrollo de software, gracias a su increíble sistema de ramificación (branching) para el desarrollo no lineal, es decir el desarrollo ejecutado por varios miembros de un grupo, en el que constantemente dan aportes para generar un producto sólido.

1.1. ¿Qué es git?

Git es tal vez la más importante herramienta para el control de versiones, esto quiere decir, que ayuda a administrar los cambios que se realicen en cada uno de los archivos, alojando los documentos en un repositorio, el cual posee toda la información necesaria para que cada persona observe el flujo de cambios correspondiente en cada archivo.

Git fue lanzado en el 2005 para almacenar y modelar la información de forma muy diferente a los otros sistemas de control de versiones, a pesar de que su interfaz sea bastante similar, la forma de manejo es diferente, ya que grosso modo, el modelamiento de las versiones en Git se hacen como "instantáneas" del repositorio en el tiempo; por otro lado, los demás sistemas de control de versiones, cada versión es definida como un cambio de cada archivo con respecto a una versión base, esta es solo una de las diferencias entre Git y cualquier otro sistema de control de versiones (Subversion, Perforce, Bazaar, etc), para profundizar más en este tema, se le sugiere la lectura complementaria.

1.2. Servicios de administración de repositorios

Existen varias tecnologías que implementan Git como herramienta que facilita la administración de los repositorios a tu cargo, acá los más importantes:

1.2.1. Coding

Esta compañía lanzada en el 2014, tiene total participación en China, con más de 500.000 proyectos en su portafolio, tuvo una historia de crecimiento bastante apresurada gracias a un financiamiento de 15 millones de dólares en el mismo año de lanzamiento, aunque aún no se ha expandido mundialmente, ya tiene puesto los ojos en el mercado internacional.

Dentro del plan gratuito, esta herramienta no tiene grandes características diferenciables, aunque el plan pago es más económico que los demás, tiene un límite de 20 usuarios para el plan de desarrollo y 50 en plan avanzado como se puede ver en la tabla 2.

1.2.2. Gitlab

Esta compañía creada en el 2011 fue diseñada para administrar, hacer pruebas y desplegar aplicaciones en conjunto con tu grupo de desarrollo, fue hasta el 2014 cuando se incorporó; tiene más de 100.000 organizaciones en su portafolio, entre ellas grandes compañías como Uber, EA, Siemens, entre otros.

Gitlab es el único servicio de administración de repositorios con código abierto, lo que la hace algo especial. Para más información puede visitar la página oficial.

Cómo mejorar...

Gitlab estuvo a punto de quedar en quiebra por no tener un correcto funcionamiento de los respaldos de los datos de producción. Siempre, asegúrese, de que los respaldos tengan un correcto funcionamiento.

Dentro de las ventajas que tiene esta herramienta frente a las otras, dentro del plan gratuito, es que se puede tener cuantos repositorios gratuitos se quiera, y no existe límite de colaboradores, lo que es de gran ayuda cuando se está iniciando una compañía y se desea minimizar costos, como lo podemos observar en la tabla 1.

1.2.3. Bitbucket

Aunque fue lanzada en el 2008, fue hasta el 2011 cuando se implementó Git en Bitbucket, antes solo soportaba Mercurial; este control de versiones ha sido reducido debido al éxito de Git.

Dentro de las ventajas de los planes que ofrece Bitbucket es la integración con otras tecnologías para administración de proyectos, entre ellos Jira lo que permite hacer seguimiento al desarrollo de cada actividad, medición de tiempos y estadísticas del desarrollo del proyecto; Atlasian es una compañía de software que ha integrado Bitbucket junto con otras soluciones para tener un conjunto de herramientas que permitan el buen desarrollo de aplicaciones. Para más información puede visitar la página oficial.

1.2.4. Github

Esta plataforma lanzada en el 2008 es la más grande de todas, con más de 62 millones de proyectos alojados a nivel global, de las cuales se destacan IBM, Airbnb, Paypal, entre otras; es tal vez la más representativa que trabaja con Git, y una buena herramienta para empezar a trabajar Git.

Github es característica por su amplio apoyo a productos de código abierto, en ella se puede encontrar un sinnúmero de proyectos públicos, en los que probablemente se puede encontrar grandes soluciones a problemas de software; por lo que vale la pena visitar la página principal.

1.2.5. Planes gratuitos

En la siguiente tabla se puede observar de manera detallada las diferentes características de cada plataforma dentro del plan gratuito.

Tabla 1. Características planes de pago

Planes gratuitos	Repositorios públicos	Repositorios privados	Colaboradores	Espacio de almacenamiento	*Hosting*	Soporte
GitHub Public	Ilimitados	0	Ilimitado	N / A	Cloud	Email / Foro
Bitbucket Small teams	Ilimitados	Ilimitados (1Gb / Proyecto)	5	N / A	Cloud	Email / Foro
GitLab Cloud Hosted	Ilimitados	Ilimitados (10Gb / Proyecto)	Ilimitado	Ilimitado	Cloud	Foro
Gitlab Community Edition	Ilimitados	Ilimitado	Ilimitado	N / A	Hosting Propio	Foro
Coding Free Plan	Ilimitados	Ilimitado	10	1GB	Cloud	Email / Foro

Fuente: Traducido al español de flow.ci, (2017).

Todas las herramientas tienen características muy similares, de acuerdo a las necesidades se debe observar cuál es la herramienta que se adecua a dichas necesidades específicas.

1.2.6. Planes pagos

En la siguiente tabla (tabla 2) se puede observar de manera detallada las diferentes características de cada plataforma dentro del plan pago.

Tabla 2. Características planes de pago

Planes pagos	Repositorios publicos	Repositorios privados	Colaboradores	Espacio de almacenamiento	Hosting	Soporte
GitHub Personal	1	Ilimitados	7	Mensual	Cloud	Email / Foro
Github Organización	5	Ilimitados	25	Mensual	Cloud	Email / Foro
	200 + 5		25 + 1800			
Bitbucket Growing Team (min 10 – Ilimitado)	10	Ilimitados	10	Mensual	Cloud	Email / Foro
	Ilimitados		200			

Coding Developer Plan	20	Ilimitado (5GB)	7.3	MES	Cloud	Email / Foro
			36.67	6 Meses		
			66	AÑO		
Coding Advanced Plan	50	Ilimitado (10GB)	30	MES	Cloud	Email / Foro / Chat / Telefono
			149	6 Meses		
			268	AÑO		

Fuente: Traducido al español de flow.ci, (2017).

En caso de que las características de las herramientas no pagas se queden cortas, los planes pagos ofrecen características especiales, generalmente útiles para empresas medianas o grandes, si quiere mayor información, visita la página oficial de cada plan.

2. Manejo de ambientes

Los ambientes en informática hacen referencia a espacios en donde se ejecutará una versión de la aplicación y base de datos para una función específica, dentro de las características compartidas que encontramos en todos los ambientes son:

- Cada ambiente tendrá una base de datos independiente, de este modo, los datos de los usuarios en producción no serán involucrados en el ambiente de pruebas y desarrollo.
- Los datos manejados en cada ambiente deben estar actualizados.
- Cada ambiente tiene un servidor destinado para la ejecución de peticiones.

Dentro de las prácticas óptimas de desarrollo se recomiendan por lo menos tres tipos de ambientes:

2.1. Desarrollo

El ambiente de desarrollo, como su nombre indica, hace referencia a un servidor generalmente local del cual los desarrolladores hacen uso, permitiendo que durante el desarrollo de la aplicación no influyan en la modificación de datos visualizados por el cliente final o el usuario de dicha aplicación. Este puede residir en el computador personal del desarrollador, o incluso en un computador local, del cual los desarrolladores en conjunto pueden tener acceso. (Ministerio de Salud de Chile, 2014).

Dentro de las características individuales de este entorno se encuentran:

- Este entorno es utilizado generalmente para incluir nuevas características y cambios en la aplicación, al igual que el versionamiento de aplicaciones.
- Tiene manipulación directa por los desarrolladores involucrados, generalmente no es necesario de un integrador para la implementación del mismo.
- Tiene un volumen de manejo de datos menor, proporcional al flujo de datos que requiere el desarrollador.
- Este ambiente debe ser utilizado únicamente para desarrollo y pruebas del desarrollador.
- La seguridad del entorno no debe ser tan alta, debido a que los usuarios en su mayoría son usuarios de prueba y los datos ficticios, de igual forma el acceso al mismo es local por lo que no permite infiltraciones de terceros.

2.2. Pruebas

Este entorno está desarrollado para la ejecución de pruebas unitarias, por parte automática o de forma manual por el equipo QA (Aseguramiento de la calidad), para conseguir los objetivos que plantean las metodologías ágiles, el equipo de desarrollo y QA deben ir de la mano, integrados en el día a día. (BBVA, 2016).

Dentro de las características individuales de este entorno se encuentran:

- Debe residir en un servidor compartido por todos los integrantes del equipo de desarrollo.
- Debe tener la mayor similitud al entorno intermedio y de producción, por lo que no debe poseer fallas; en caso de tenerlas debe ser informado inmediatamente al líder del proyecto para acción oportuna.
- El único fin de este entorno es para el desarrollo de pruebas.

2.3. Producción

Este entorno es necesario que se despliegue con gran responsabilidad, debido a que es aquel en el que el usuario final va a tener acceso y el éxito de la aplicación parte a partir de este; las fases anteriores son requeridas para que la calidad de este servidor sea óptima.

En algunas ocasiones se observa un entorno intermedio de producción, en el cual el cliente y algunos usuarios de prueba tienen acceso (personas del común que realizan pruebas de usuario), generalmente personas sin conocimientos técnicos los cuales cumplen con el perfil de usuario. La ejecución de este entorno, se realiza como un despliegue de prueba minimizando los errores en el lanzamiento definitivo de la aplicación, teniendo datos, que en su mayoría ayudan a la formación de la idea.

Dentro de las características individuales de este entorno se encuentran:

- La seguridad del entorno debe ser mayor, por el manejo de datos sensibles, usuarios reales, con datos deseables por terceros.
- El tamaño reservado para este entorno debe ser considerable, es importante percibir la posibilidad de gran éxito y estar prevenido de una carga alta de información.
- Los fallos deben ser cero, cualquier falla perceptible por el usuario puede llevar al fracaso del producto.
- Solo el administrador se debe encargar de instalar, actualizar y desplegar el ambiente de pruebas, debe ser una persona altamente calificada.
- Deben estar establecidos estrictos controles de cambios.
- No debe ser utilizado para realización de operaciones de desarrollo y de pruebas.

El manejo de ambientes virtuales y un correcto manejo de repositorios es una de las características más importantes para el desarrollo de software, del buen uso y respeto de las herramientas y prácticas de desarrollo, depende el triunfo de una aplicación de software.

En síntesis...

Github es una excelente herramienta para la administración de repositorios en proyectos de desarrollo, dado que permite hacer seguimiento de todos los cambios dentro de los ambientes de software.

Referencias

BBVA. (2016) BBVA API market. Recuperado de: https://bbvaopen4u.com/es/actualidad/que-es-qa-y-por-que-no-debe-faltar-en-tu-proyecto

Ministerio de Salud de Chile. (2014). Procedimiento separación ambientes. Recuperado de: http://web.minsal.cl/sites/default/files/files/2014%20Procedimiento%20separación%20ambientes%20DPP.pdf

Referencias de tablas

Flow.ci. (2016). Medium. Recuperado de: https://medium.com/flow-ci/github-vs-bitbucket-vs-gitlab-vs-coding-7cf2b43888a1

INFORMACIÓN TÉCNICA

FACULTAD DE
INGENIERÍA, DISEÑO
E INNOVACIÓN

Módulo: Énfasis Profesional I (Integración continua)

Unidad 1: Principios de Integración Continua

Escenario 2: Manejo de Repositorios

Autor: Eduardo Enrique Oliveros Acosta

Asesor Pedagógico: Amparo Sastoque Romero

Diseñador Gráfico: Paola Andrea Melo

Asistente: Eveling Peñaranda