Evidencia de conocimiento: identifica herramientas de versionamiento.

GA7-220501096-01-AA1-EV03

Introducción

El control de versiones es un aspecto fundamental en el desarrollo de software moderno. Permite a los equipos de desarrollo rastrear y gestionar los cambios en el código a lo largo del tiempo. Las herramientas de control de versiones facilitan la colaboración entre desarrolladores, mejoran la calidad del software y reducen el riesgo de conflictos en el código. Este documento explora algunas de las herramientas más populares de control de versiones utilizadas en la industria: Git, Subversion (SVN), Mercurial, y herramientas de alojamiento de repositorios como GitHub, GitLab, y Bitbucket.

Objetivos

Explorar las principales herramientas de control de versiones disponibles en la actualidad.

Entender las ventajas y desventajas de cada herramienta.

Evaluar cómo cada herramienta puede integrarse en el flujo de trabajo de desarrollo de software.

Proporcionar una guía para la selección de la herramienta de control de versiones más adecuada para un proyecto o equipo específico.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Herramienta** | **Descripción** | **Ventajas** | **Desventajas** | **Casos de Uso** |
| **Git** | Sistema de control de versiones distribuido. Permite trabajo sin conexión y eficiente gestión de ramas. | Alta velocidad y eficiencia. Amplia comunidad. | Curva de aprendizaje pronunciada. Gestión de grandes archivos no ideal. | Proyectos donde la colaboración distribuida y la ramificación son críticas. |
| **Subversion** | Sistema de control de versiones centralizado. Rastrea cambios a través de un servidor central. | Rastreo detallado. Fácil de usar para principiantes. | Requiere conexión constante al servidor central. Menos eficiente en gestión de ramas. | Proyectos donde la simplicidad y el control centralizado son prioritarios. |
| **Mercurial** | Similar a Git pero diseñado para ser fácil de aprender y usar. | Interfaz simple. Rápido y eficiente. | Menos adoptado que Git. Menos integraciones disponibles. | Equipos que prefieren una curva de aprendizaje más suave y proyectos de tamaño mediano. |
| **GitHub** | Plataforma de desarrollo colaborativo que utiliza Git. Ofrece herramientas robustas para la gestión de proyectos y la colaboración en equipo. | Amplia comunidad. Integración con CI/CD. | Algunas funciones avanzadas requieren suscripción de pago. Enfoque principal en Git. | Proyectos de código abierto y privados que buscan herramientas avanzadas para colaboración y gestión de código. |
| **GitLab** | Plataforma de DevOps que ofrece alojamiento de repositorios Git y herramientas integradas para CI/CD, gestión de proyectos y seguridad. | Gestión completa del ciclo de vida del software. | Menos usuarios que GitHub. Curva de aprendizaje para nuevos usuarios. | Equipos que necesitan una solución integral para el ciclo de vida del desarrollo de software. |
| **Bitbucket** | Plataforma que soporta Git y Mercurial. Popular en el ecosistema Atlassian con integraciones con Jira y Confluence. | Soporte para Git y Mercurial. Integración con Atlassian. | Menos flexibilidad en comparación con GitHub y GitLab. | Equipos que utilizan otros productos de Atlassian y necesitan soporte para varios tipos de repositorios. |
| **SVN** | Herramienta de control de versiones centralizado con amplio soporte para versionamiento de directorios completos y seguimiento de cambios en binarios. | Ideal para proyectos que requieren un control estricto sobre versiones. | Menos eficiente en la gestión de ramas y colaboración distribuida. | Proyectos donde la simplicidad y la estabilidad en el versionamiento son críticas. |
| **DVC** | Herramienta específica para el control de versiones de datos y modelos de aprendizaje automático. Permite rastrear cambios en datos de manera eficiente. | Gestión eficiente de datos y modelos. Integración con almacenamiento en la nube. | Enfoque limitado a proyectos de datos y machine learning. | Proyectos de ciencia de datos y aprendizaje automático que requieren control riguroso sobre datos y modelos. |
| **Perforce** | Herramienta de control de versiones centralizado conocida por su robustez y capacidad para manejar grandes volúmenes de archivos. | Escalabilidad y rendimiento para grandes proyectos. | Licenciamiento costoso y menos popular que Git y SVN. | Proyectos con grandes equipos y requisitos de control estricto sobre versiones y acceso. |

**Referencias**

(*▷ Qué es un script, para qué sirve y cómo crear uno | InboundCycle*, 2024; *CSS | MDN*, 2024; *Diagrama entidad relación: ¿Qué es para qué sirve? Con ejemplos | Miro*, s. f.; *Diferencias entre DDL, DML y DCL - TodoPostgreSQL*, s. f.; *HTML*, 2023; *Las primeras microtiendas del mercado comienzan a abrir | Hoy*, s. f.; *¿Qué es la Factura Electrónica?*, s. f.; *Qué es una base de datos | Oracle Colombia*, s. f.)

*▷ Qué es un script, para qué sirve y cómo crear uno | InboundCycle*. (2024, enero 8). https://www.inboundcycle.com/diccionario-marketing-online/script

*CSS | MDN*. (2024, junio 6). https://developer.mozilla.org/es/docs/Web/CSS

*Diagrama entidad relación: ¿Qué es para qué sirve? Con ejemplos | Miro*. (s. f.). Recuperado 8 de junio de 2024, de https://miro.com/es/diagrama/que-es-diagrama-entidad-relacion/

*Diferencias entre DDL, DML y DCL - TodoPostgreSQL*. (s. f.). Recuperado 8 de junio de 2024, de https://www.todopostgresql.com/diferencias-entre-ddl-dml-y-dcl/

*HTML: Lenguaje de etiquetas de hipertexto | MDN*. (2023, julio 24). https://developer.mozilla.org/es/docs/Web/HTML

*Las primeras microtiendas del mercado comienzan a abrir | Hoy*. (s. f.). Recuperado 8 de junio de 2024, de https://www.hoy.es/20090819/don-benito/primeras-microtiendas-mercado-comienzan-20090819.html

*¿Qué es la Factura Electrónica?* (s. f.). Recuperado 8 de junio de 2024, de https://micrositios.dian.gov.co/sistema-de-facturacion-electronica/factura-electronica/

*Qué es una base de datos | Oracle Colombia*. (s. f.). Recuperado 8 de junio de 2024, de https://www.oracle.com/co/database/what-is-database/