Системы контроля версий. Знакомство с VCS.

весна 2023

Что такое Version Control System?

Контроль версий, также известный как управление исходным кодом, — это практика отслеживания изменений программного кода и управления ими.

Система управления версиями или система контроля версий — программное обеспечение для работы с изменяющейся информацией.

Система контроля версий записывает изменения в специальную базу данных в течение времени и позволяет вернуться позже к определённой версии.

Командная работа над проектом

Система контроля версий нам нужна, потому что мы делаем большие системы, работаем в командах.

Система контроля версий помогает нам **управлять кодом и совместной разработкой**.

Помогает управлять изменениями в исходном коде с течением времени.

Разработчики программного обеспечения, работающие в командах, постоянно пишут новый исходный код и изменяют существующий. Код проекта, приложения или программного компонента обычно организован в виде структуры папок или «дерева файлов».

Один разработчик в команде может работать над новой возможностью, а другой в это же время изменять код для исправления несвязанной ошибки, т. е. каждый разработчик может вносить свои изменения в несколько частей дерева файлов.

Контроль версий помогает командам решать проблемы путем отслеживания каждого изменения, внесенного каждым участником и предотвращать возникновение конфликтов при параллельной работе.

Изменения, внесенные в одну часть программного обеспечения, могут быть не совместимы с изменениями, внесенными другим разработчиком, работавшим параллельно. Такая проблема должна быть обнаружена и решена согласно регламенту, не создавая препятствий для работы остальной части команды.

Disclaimer

Если не выработаны или не соблюдаются правила совместной работы над проектом, система контроля версий снимает с себя ответственность.

Разрабатывать большой проект
без системы контроля версий
очень неуютно.

Предоставляют следующие основные возможности:

- Позволяют восстанавливать ранние версии файлов.
- Дают возможность узнать, кто и когда добавил или изменил конкретный набор строк в файле.
- Создавать разные, параллельно развивающиеся версии проекта (“ветви”) и их впоследствии объединять в одну.
- Ведут журнал изменений, в который пользователи могут записывать пояснения о том, что и почему они изменили в данной версии.

Термины git

SCM = Source Code Management

Distributed VCS

Repository - Data base of linked states

Working copy

Stage = prepare state

Commit = State != Changes

History = linked states

Repository - Data base of linked states

Репозиторием в Git’е называется база данных, связанных между собой состояний вашего проекта, которые были на тот или иной момент времени.

Репозиторий может храниться локально на компьютере и где-то на сервере/серверах.

Working copy

Рабочая копия - текущий срез вашего проекта на вашей файловой системе, то есть когда вы видите непосредственно файлы, а не объекты, хранящиеся в базе данных git’а.

Stage

Stage - состояние между рабочей копией и ещё не положенное в базу данных

Stage = prepare state

Commit

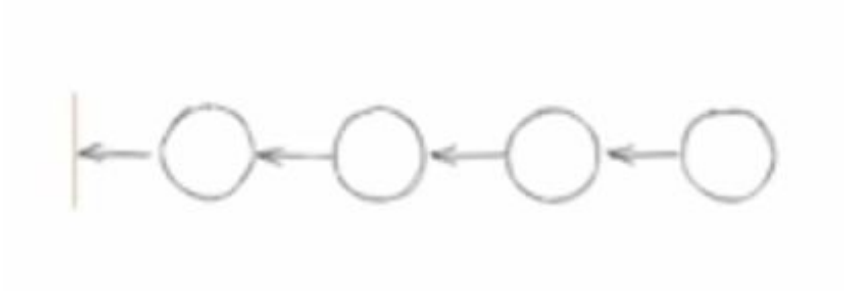
В разных системах контроля версий под commit’ом понимают разные штуки.

В git - это фиксация конкретного состояния. Не изменение, а именно состояние. Мы внесли некоторые изменения в наш проект и хотим зафиксировать текущее состояние проекта на текущий момент времени.

Commit = State != Changes

History = linked states

История проекта представляет собой нецикличный направленный граф состояний.



У первого commit’а истории нет предыдущего состояния, но каждый последующий commit будет ссылаться на предыдущее состояние. На этой картинке 4 состояния, каждое указывает на предыдущее состояние.

Посмотрим на git в консоли

> mkdir git-test

> cd git-test

> ls -la

> git init .

> ls -la

Посмотрим на git в консоли (продолжение)

> vim hi.txt

> git status

> git add hi.txt

> git status

> git commit -m "первый коммит"

Посмотрим на git в консоли (продолжение)

> git log

> vim hi.txt

…

> git log

“id коммита” - это идентификатор (hash), а не “заклинание”

или HEAD~№

Посмотрим на git в консоли (продолжение)

> git show “id коммита”

git show вычисляет изменения в runtime

> git cat-file -p “id коммита”

в поле parent мы видим commit, фиксирующий предыдущее состояние

Посмотрим на git в консоли (продолжение)

> git log --oneline --all --graph

> git diff “id коммита”

> git revert “id коммита”

Посмотрим на git в консоли (продолжение)

> git reset передвигает указатель

--soft сохраняет рабочую копию и stage

--mixed (по умолчанию) оставит состояние файловой системы, которое было, stage почистит, файлы которые были добавлены в него, окажутся в состоянии untracked

--hard чистит изменения в рабочей копии и в stage

Посмотрим на git в консоли (продолжение)

С помощью git reset мы можем потерять состояния!

Они сохранятся в базе и с помощью ряда манипуляций мы можем их добыть оттуда, но лучше не попадать в такие ситуации.

Хорошо, что у нас есть ветки и удалённые репозитории (см. в следующей лекции).