Приложение B. Поиск ломающего коммита методом половинного деления.

Если у вас возникла поломка, которой раньше не было, существует несколько вариантов её исправления. Например, постепенно откатывая HEAD, но есть и более прогрессивный метод.

Итак, речь пойдет о команде **git bisect**. Git bisect основан на методе половинного деления.

Мы указываем коммит при котором воспроизводится ошибка (обычно это текущее состояние) и коммит при котором все работает. Далее происходит переход (checkout) к коммиту между этими состояниями. Наша задача проверить работоспособность и известить git bisect о результате (все хорошо или все по-прежнему плохо). И так до тех пор пока не закончится поиск. В конце концов мы получаем коммит, после которого обнаружилась ошибка. Остается только посмотреть все изменения и исправить код в нужных местах.

Пример:

Допустим, вы только что развернули некоторую версию вашего кода в production environment окружении и теперь получаете отчеты о некоторой ошибке, которая не возникала в вашем разработческом окружении, и вы не можете представить, почему код ведет себя так. Вы возвращаетесь к вашему коду и выясняете, что можете воспроизвести проблему, но всё еще не понимаете, что работает неверно. Вы можете воспользоваться бинарным поиском, чтобы выяснить это. Во-первых, выполните команду **git bisect start** для запуска процесса поиска, а затем используйте **git bisect bad**, чтобы сообщить Git, что текущий коммит сломан. Затем, используя **git bisect good [good_commit]**, вы должны указать, когда было последнее известное рабочее состояние:

$ git bisect start
$ git bisect bad
$ git bisect good v1.0
Bisecting: 6 revisions left to test after this
[ecb6e1bc347ccecc5f9350d878ce677feb13d3b2] error handling on repo

Git выяснил, что произошло около 12 коммитов между коммитом, который вы отметили как последний хороший коммит (v1.0), и текущим плохим коммитом, и выгрузил вам один из середины. В этот момент вы можете запустить ваши тесты, чтобы проверить присутствует ли проблема в этом коммите. Если это так, значит она была внесена до выгруженного промежуточного коммита, если нет, значит проблема была внесена после этого коммита. Пусть в данном коммите проблема не проявляется, вы сообщаете об этом Git с помощью git bisect good и продолжаете ваше путешествие:

$ git bisect good

Bisecting: 3 revisions left to test after this
[b047b02ea83310a70fd603dc8cd7a6cd13d15c04] secure this thing

Теперь вы оказались на другом коммите, расположенном посредине между только что протестированным и плохим коммитами. Вы снова выполняете ваши тесты, обнаруживаете, что текущий коммит сломан, и сообщаете об этом Git с помощью команды git bisect bad:

$ git bisect bad
Bisecting: 1 revisions left to test after this
[f71ce38690acf49c1f3c9bea38e09d82a5ce6014] drop exceptions table

Этот коммит хороший и теперь Git имеет всю необходимую информацию для определения того, где была внесена ошибка. Он сообщает вам SHA-1 первого плохого коммита и отображает некоторую информацию о коммите и файлах, которые были изменены в этом коммите, так, чтобы вы смогли разобраться что же случилось, что могло привнести эту ошибку:

$ git bisect good
b047b02ea83310a70fd603dc8cd7a6cd13d15c04 is first bad commit
commit b047b02ea83310a70fd603dc8cd7a6cd13d15c04
Author: PJ Hyett <pjhyett@example.com>
Date: Tue Jan 27 14:48:32 2009 -0800

secure this thing

:040000 040000 40ee3e7821b895e52c1695092db9bdc4c61d1730
f24d3c6ebcfc639b1a3814550e62d60b8e68a8e4 M config

Когда вы закончили бинарный поиск, нужно выполнить git bisect reset для того, чтобы вернуть HEAD туда, где он был до начала поиска, иначе вы останетесь в, довольно, причудливом состоянии:

$ git bisect reset

Это мощный инструмент, который помогает вам за считанные минуты проверить сотни коммитов на возможность внесения ошибки. В действительности, если у вас есть скрипт, который будет возвращать 0 если проект находится в рабочем состоянии и любое другое число в обратном случае, то вы можете полностью автоматизировать git bisect. Сперва, вы снова сообщаете границы бинарного поиска, указывая известные плохие и хорошие коммиты. Вы можете сделать это, передавая их команде bisect start – первым аргументом известный плохой коммит, а вторым известный хороший коммит:

$ git bisect start HEAD v1.0
$ git bisect run test-error.sh

Это приведет к автоматическом выполнению test-error.sh на каждый выгруженный коммит до тех пор, пока Git не найдет первый сломанный коммит. Вы также можете использовать что-то еще, что у вас есть для запуска автоматизированных тестов.

Плюсы решения:

- Экономие приличного количество времени при поиске ошибки;

- Простота работы.

Использованные источники:

- https://git-scm.com/book/ru/v2/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D1%8B-Git-%D0%9E%D0%B1%D0%BD%D0%B0%D1%80%D1%83%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5-%D0%BE%D1%88%D0%B8%D0%B1%D0%BE%D0%BA-%D1%81-%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E-Git