Self-Learning Points of Gerrit

Contents

[1. What is Gerrit 7](#_Toc439856552)

[1) Characteristics 7](#_Toc439856553)

[2) Advantage 7](#_Toc439856554)

[3) Disadvantage 7](#_Toc439856555)

[2. Key Points 7](#_Toc439856556)

[1) Push to Gerrit is same as push to Git. Only target branch name is different: **refs/for/** 7](#_Toc439856557)

[2) Difference between new change and new patch set 8](#_Toc439856558)

[=> 8](#_Toc439856559)

[=>A common scenario: 8](#_Toc439856560)

[3) How to revert changes in git 9](#_Toc439856561)

[4) Git has 3 level config 9](#_Toc439856562)

[5) Merge 9](#_Toc439856563)

[6) Rebase 11](#_Toc439856564)

[7) Pull and Push 12](#_Toc439856565)

[3. FAQ 12](#_Toc439856566)

[1) What’s SHA1? 12](#_Toc439856567)

[2) How to allow multiple people push code to a same repository? 13](#_Toc439856568)

[4. Frequently Used Commands 13](#_Toc439856569)

[*1)* *git log* 13](#_Toc439856570)

[*git log --follow pom.xml* 14](#_Toc439856571)

[*Shows commits that changed pom.xml, including those that occurred before the file was given its present name.* 14](#_Toc439856572)

[*git log --name-only pom.xml* 14](#_Toc439856573)

[*Shows the commits for file pom.xml, listing only name* 14](#_Toc439856574)

[*git log --name-status pom.xml* 14](#_Toc439856575)

[*Shows the commits for file pom.xml, listing name and status* 14](#_Toc439856576)

[*git log --stat pom.xml* 14](#_Toc439856577)

[*Generates a diffsta.* 14](#_Toc439856578)

[*显示简要的增改行数统计,每次提交文件的变更统计.* 14](#_Toc439856579)

[*git log -p pom.xml* 14](#_Toc439856580)

[*Similar to--stat, with more info.* 14](#_Toc439856581)

[*git log -1 pom.xml* 14](#_Toc439856582)

[*Limits the number of commits to show to 1* 14](#_Toc439856583)

[*git shortlog pom.xml* 14](#_Toc439856584)

[*Shows commit with only name and short description.* 14](#_Toc439856585)

[*git log --pretty=oneline pom.xml* 14](#_Toc439856586)

[*Set the output to oneline format* 14](#_Toc439856587)

[*git log --since=200.days -- pom.xml* 14](#_Toc439856588)

[*Shows the changes during the last 200 days to the file pom.xml. “-- is necessary to avoid confusion with a branch named pom.xml.* 14](#_Toc439856589)

[*--since, --after 仅显示指定时间之后的提交。* 14](#_Toc439856590)

[*git log --until=200.days pom.xml* 14](#_Toc439856591)

[*--until, --before 仅显示指定时间之前的提交。* 14](#_Toc439856592)

[*git log rel-5.10 extensions/* 14](#_Toc439856593)

[*Shows all commits since version rel-5.10 that changed any file in extensions/ subdirectories* 14](#_Toc439856594)

[*git log --graph pom.xml* 14](#_Toc439856595)

[*--graph 显示 ASCII 图形表示的分支合并历史。* 14](#_Toc439856596)

[*git log --branches --not --remotes=origin* 14](#_Toc439856597)

[*Shows all commits that are in any of local branches but not in any of remote-tracking branches for*origin*(what you have that origin doesn’t).* 14](#_Toc439856598)

[*git log master --not --remotes=\*/master* 14](#_Toc439856599)

[*Shows all commits that are in local master but not in any remote repository master branches.* 14](#_Toc439856600)

[*2)* *git cat-file* 14](#_Toc439856601)

[*git ls-tree* 14](#_Toc439856602)

[*git cat-file -t 24aa5cbbe229* 16](#_Toc439856603)

[**Output:** 16](#_Toc439856604)

[*commit* 16](#_Toc439856605)

[*git cat-file commit 24aa5cbbe229* 16](#_Toc439856606)

[**Output:** 16](#_Toc439856607)

[*git ls-tree 24aa5cbbe229* 16](#_Toc439856608)

[**Output:** 16](#_Toc439856609)

[*git cat-file -p master^{tree}* 16](#_Toc439856610)

[**Output:** 16](#_Toc439856611)

[100644 blob 05000bc34e374d540116fa83662108a6e96a1da0 .gitignore 16](#_Toc439856612)

[100644 blob 5ebfd75fb43b68f670c5148e7cfa15834325eadb .gitreview 16](#_Toc439856613)

[040000 tree e88c475df6e75cbaae601b866a7770948d233c3d container 16](#_Toc439856614)

[100644 blob b97b4bc1a3129e4f56d2435f42ce9d106ecea1b9 cvom.html\_context\_Visual Intelligence.xml 16](#_Toc439856615)

[040000 tree a2ac5ce35313d2f9902151d1eb5634843b987d27 documentation 16](#_Toc439856616)

[040000 tree 3733cb7468f0aff8a21f91caee9b266e0d07bf5a extension 16](#_Toc439856617)

[040000 tree 3cd5614b2a8d7a89651f34d76f43c5f99d71a4ce fortify 16](#_Toc439856618)

[040000 tree d47fd2b8eb9eaeeb82f89d8a037c0eeaeae71d03 info 16](#_Toc439856619)

[100644 blob 55c0e6df5087017a5783d4ed13035e2a1925c13e pom.xml 16](#_Toc439856620)

[040000 tree 60df7c4ecef2192d5aa9086a6040b33cff324337 testPnR 16](#_Toc439856621)

[040000 tree 93d31d1d7f2aaf4cac21735a858ee45c567a7cbe viz 16](#_Toc439856622)

[040000 tree 2ea8a93bd3e2758031d0f61c37a5d1d056a5fe2d vizPacker 16](#_Toc439856623)

[*git cat-file blob 05000bc34e374d5* 17](#_Toc439856624)

[**Output:** 17](#_Toc439856625)

[*git config --get-all user.name* 17](#_Toc439856626)

[Get value of a given key. 17](#_Toc439856627)

[*git config --global gc.auto 0* 17](#_Toc439856628)

[Disable this behavior permanently without further considerations 17](#_Toc439856629)

[*git gc* 17](#_Toc439856630)

[Clean up unnecessary files and optimize the local repository 17](#_Toc439856631)

[*git log --until=200.days pom.xml* 17](#_Toc439856632)

[*--until, --before 仅显示指定时间之前的提交。* 17](#_Toc439856633)

[*git log rel-5.10 extensions/* 17](#_Toc439856634)

[*Shows all commits since version rel-5.10 that changed any file in extensions/ subdirectories* 17](#_Toc439856635)

[*git log --graph pom.xml* 17](#_Toc439856636)

[*--graph 显示 ASCII 图形表示的分支合并历史。* 17](#_Toc439856637)

[*git log --branches --not --remotes=origin* 17](#_Toc439856638)

[*Shows all commits that are in any of local branches but not in any of remote-tracking branches for*origin*(what you have that origin doesn’t).* 17](#_Toc439856639)

[*git log master --not --remotes=\*/master* 17](#_Toc439856640)

[*Shows all commits that are in local master but not in any remote repository master branches.* 17](#_Toc439856641)

[5. Training Materials 17](#_Toc439856642)

[1. Gerrit Concepts 18](#_Toc439856643)

[1) Push 18](#_Toc439856644)

[2) Push New Patch Set 19](#_Toc439856645)

[Table 1 Difference of using Soft, Mixed and Hard in git reset B 9](#_Toc426823106)

[Table 2 Examples of git log 13](#_Toc426823107)

[Table 3 Examples of git cat-file and git ls-tree 16](#_Toc426823108)

[Figure 1 Index in git repository: git add 7](#_Toc426823119)

[Figure 2 Folder objects in git repository: git commit 8](#_Toc426823120)

[Figure 3 Git has 3 level config 9](#_Toc426823121)

[Figure 4 Before Fast Forward merge 9](#_Toc426823122)

[Figure 5 After Fast Forward merge 9](#_Toc426823123)

[Figure 6 Before Cherry-Pick merge 10](#_Toc426823124)

[Figure 7 After Cherry-Pick merge 10](#_Toc426823125)

[Figure 8 Git before rebase 11](#_Toc426823126)

[Figure 9 Git after rebase: fast forward merge is possible 11](#_Toc426823127)

Gerrit

# What is Gerrit

Gerrit is a git server with 2 extra functions: access management and code review.

## Characteristics

TODO

## Advantage

TODO

## Disadvantage

TODO

# Key Points

## Push to Gerrit is same as push to Git. Only target branch name is different: **refs/for/**

|  |  |  |
| --- | --- | --- |
| Push to Git | Push to Gerrit | Notes |
| git push origin HEAD:master  Git:   * Could has 2 commits in 1 push | git push origin HEAD:refs/for/master  Gerrit:   * Create a new branch for the commit you push * Create a new open Gerrit change in Gerrit DB for **each** push: 2 changes |  |

Table Difference between Git and Gerrit Push

Difference between Change and Patch Set in Gerrit

|  |  |
| --- | --- |
| Term | Description |
| Change | The unit of review.  Results in a single commit when merged to the git repository.  Change numbers are unique and never change. |
| Patch Set | A revision of a Change.  Each time a Change is modified, it will receive a new Patch Set.  Patch Set numbering starts from 1.   * Technically, a Patch Set is a unique git commit. |

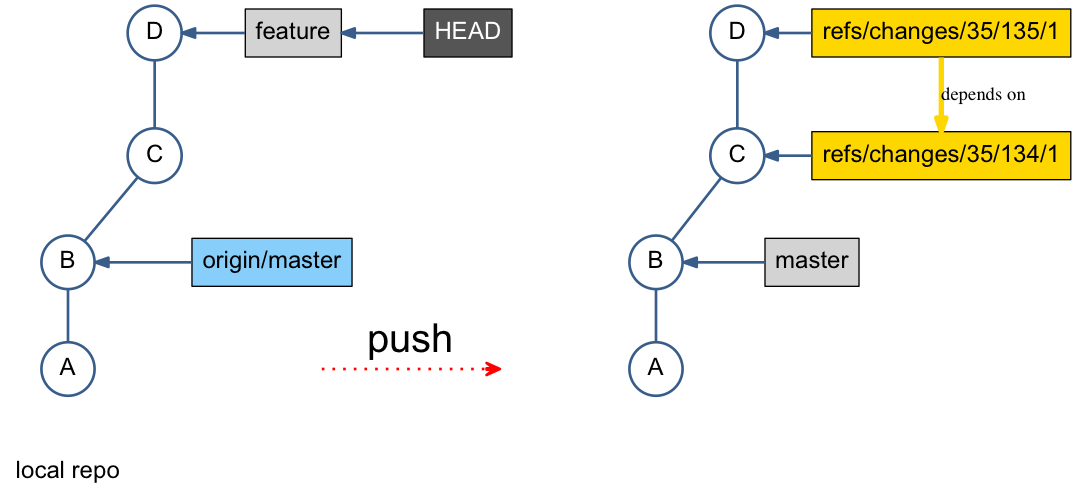


Figure Difference between Git and Gerrit Push

## Difference between new change and new patch set

|  |  |
| --- | --- |
| Change | Patch Set |
| Contains:   * Change-Id * Meta-data (project, owner, etc…) * One or more patch set(s) * Comments * Votes | * New version of an existing change * Only the latest patch set is relevant * No dependencies between patch sets |

Table Difference between Change and Patch Set

## =>

1. git commit –amend pushs a new patch set.
2. git fetch downloads an open change locally for test (using commands created by Gerrit.)

## => A common scenario:

Author of Patch Set 1 is not available and somebody else needs to continue and provide Patch Set 2

1. git fetch
2. Create a new branch based on the fetched path set 1
3. Fix the issue
4. git commit –amend
5. push

## How to revert changes in git

* When change is only in working directory, not staged yet.

TODO

* When change is staged, not committed yet.

TODO

* When change is committed.

TODO

* Difference of using Soft, Mixed and Hard in git reset B

|  |  |  |  |
| --- | --- | --- | --- |
| Reset | Working Directory | Stage (Index) | Remote Repo(Branch) |
| soft | No | No | Yes |
| mixed | No | Yes | Yes |
| hard | Yes | Yes | Yes |

Table Difference of using Soft, Mixed and Hard in git reset B

## Git has 3 level config

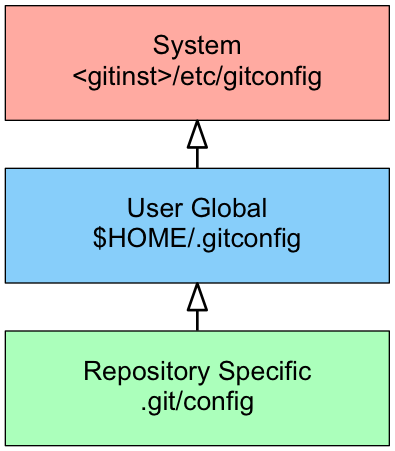


Figure Git has 3 level config

## Merge

* Merge commit is a commit with more than one parent.
* git merge feature1 by default is Fast Forward merge, which just moves the pointer, no new merge commit is created.

<= this is why merge in git is fast.

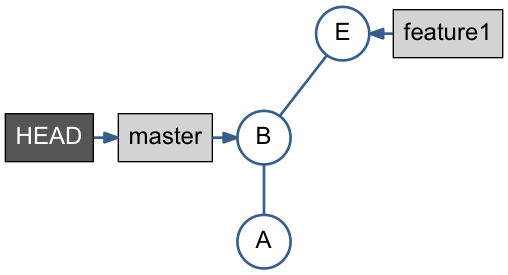


Figure Before Fast Forward merge

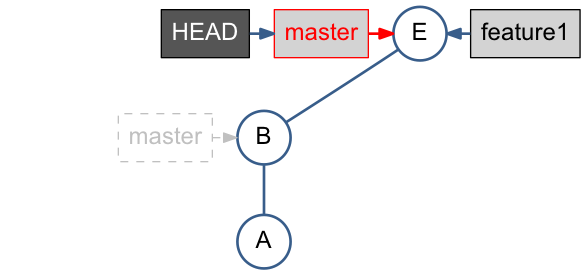


Figure After Fast Forward merge

* git cherry-pick feature1 applies only changes done by F, means the diff-2, has no parent relation to F.

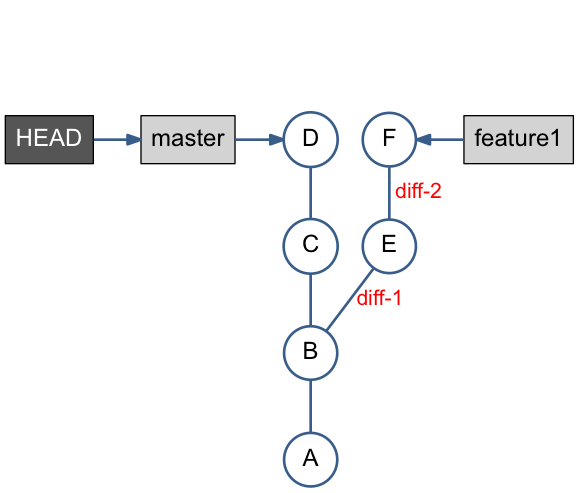
**

Figure Before Cherry-Pick merge

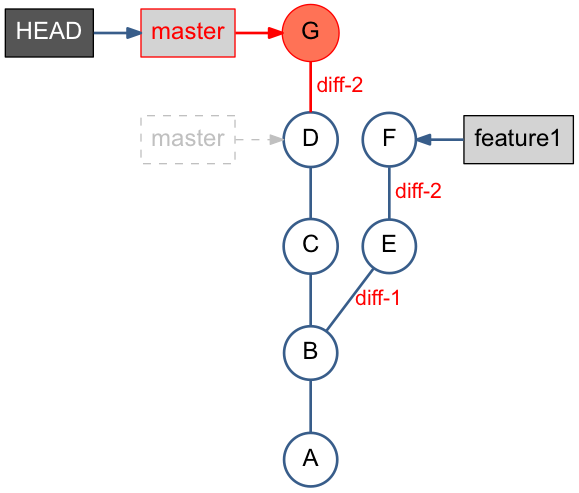
**

Figure After Cherry-Pick merge

## Rebase

* **重新定义某个分支的参考基准.** 就好比移花接木那樣（稼接），把某個樹枝接到別的樹枝。
* git rebase master redo the work done in teature1 branch on top of the master*.*
* **Alternative to Merge** – Keeping history linear
* Fast Forward merge is possible after rebase.
* git rebase –onto *<new* base *commit> <current base commit>指定要從哪裡開始接枝*

[**https://blog.yorkxin.org/posts/2011/07/29/git-rebase/**](https://blog.yorkxin.org/posts/2011/07/29/git-rebase/)

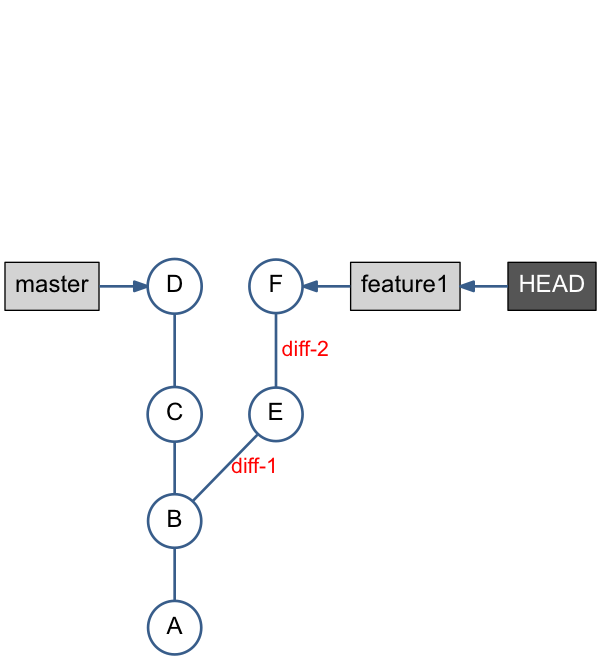
****

Figure Git before rebase

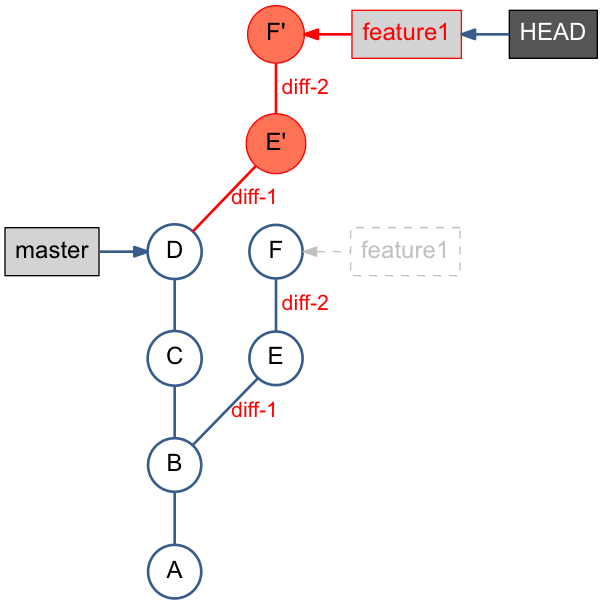
****

Figure Git after rebase: fast forward merge is possible

## Pull and Push

Pull is equal to 3 possibilities:

* git pull = git fetch + git merge

or

* git pull = git fetch + git rebase

or

* git pull = git fetch

Push could mean to deal with 3 scenarios:

* push (when remote ‘origin’ repo is **NOT** changed)

or

* fetch, merge, push (when remote ‘origin’ repo is changed)

or

* fetch, rebase, push (when remote ‘origin’ repo is changed)

# FAQ

### What’s SHA1?

SHA1 is a globally unique commit ID

SHA1 is a function of the commit object content.

SHA1 is a 40-digit hexadecimal number, seen in git log output, git history view etc. ex. e168254… 23400e6…

### How to allow multiple people push code to a same repository?

Using core.sharedRepository to control access.

Ref: <http://criticallog.thornet.net/2010/01/07/sharing-your-git-repository/>

Steps:

* Create a group git, put multiple people into it.

useradd git

usermod -g git steven

useradd -g git nicholas

chmod 770 /home/git

chmod g+w objects/

* *Update .git/config, adding sharedRepository*

sharedRepository=1

加这个参数的目的是git在objects目录下创建的目录的属性由

drwxr-xr-x 变成 drwxrwsr-x

* Done.

# Frequently Used Commands

TODO