rev-list-description

List commits that are reachable by following the `parent` links from the given commit(s), but exclude commits that are reachable from the one(s) given with a ’^’ in front of them. The output is given in reverse chronological order by default.

You can think of this as a set operation. Commits reachable from any of the commits given on the command line form a set, and then commits reachable from any of the ones given with ’^’ in front are subtracted from that set. The remaining commits are what comes out in the command’s output. Various other options and paths parameters can be used to further limit the result.

Thus, the following command:

ifdef::git-rev-list[] ---------- $ git rev-list foo bar ^baz ---------- endif::git-rev-list[] ifdef::git-log[] ---------- $ git log foo bar ^baz ---------- endif::git-log[]

means "list all the commits which are reachable from ’foo’ or ’bar’, but not from ’baz’".

A special notation “<commit1>..<commit2>” can be used as a short-hand for “^<commit1> <commit2>”. For example, either of the following may be used interchangeably:

ifdef::git-rev-list[] ---------- $ git rev-list origin..HEAD $ git rev-list HEAD ^origin ---------- endif::git-rev-list[] ifdef::git-log[] ---------- $ git log origin..HEAD $ git log HEAD ^origin ---------- endif::git-log[]

Another special notation is “<commit1>...<commit2>” which is useful for merges. The resulting set of commits is the symmetric difference between the two operands. The following two commands are equivalent:

ifdef::git-rev-list[] ---------- $ git rev-list A B --not $(git merge-base --all A B) $ git rev-list A...B ---------- endif::git-rev-list[] ifdef::git-log[] ---------- $ git log A B --not $(git merge-base --all A B) $ git log A...B ---------- endif::git-log[]

□□□ #1
□ □□□ □□□ 2026-10-31 20:15:47 UTC
□ □□□ □□□ 2026-10-31 20:15:48 UTC