git-rebase-update(1) Manual Page

NAME

git-rebase-update - Updates all branches to have the latest changes from their upstreams.

SYNOPSIS

> *git rebase-update* [-v | --verbose] [-n | --no-fetch] [-k | --keep-going]

DESCRIPTION

Brings all branches up-to-date with their tracking branches. This involves several phases:

Preparation

If you currently have a branch checked out, any changes on that branch are *frozen* (See git-freeze(1)for more detail). Additionally, the current branch is recorded for the *Restoration* phase later (see *CONFIGURATION VARIABLES* for details on `depot-tools.rebase-update.starting-branch`).

Fetching

All branches are examined to find their upstream references. The correct set of git remotes is determined, and fetched accordingly. Note that if any branches have a tag as their upstream, we are forced to pull all remotes.

Pass `--no-fetch` to skip this phase.

Rebasing

All branches are rebased in topological order from roots (upstreams) to leaves. Each branch is rebased from its marked merge-base (see *CONFIGURATION VARIABLES*) to the branch tip on top of its parent branch. If the parent branch is *frozen*(see git-freeze(1)), the branch will be rebased onto the last non-freeze commit on the parent branch.

Things get interesting when there are merge conflicts on rebase. The **most**

common cause for conflicts is when your branch has been committed to the upstream in squashed form, ala git-squash-branch(1), which is whatgit-cl(1) and the *Commit Queue* will do. Because of that, `git rebase-update` will attempt to squash your conflicted branch to see if the squashed version applies cleanly to its upstream.

If it does not apply cleanly, then your original (non-squashed) branch will be left in mid-rebase and `git rebase-update` will exit. You can deal with this like any other conflicted rebase. When you're done, just `git rebase-update` again to pick up where you left off.

If you'd like to rebase all rebaseable branches in one pass and manually process the unrebaseable ones later, use -k or --keep-going. Cleanup will not happen until all branches apply cleanly.

Cleanup

Once all the branches have been rebased, any empty branches (i.e. branches with no commits on them) are removed. If a branch is removed in this fashion, any branches which depend on it are reparented to the parent of the removed branch (see git-reparent-branch(1)).

Restoration

`git rebase-update` checks out the branch that you started on, and *thaws* it, if necessary (see git-thaw(1)). If the branch you started on got cleaned up, `git rebase-update` will checkout the *root* ref (defaults to *origin/master*, as configured by `depot-tools.upstream`, see git-new-branch(1)).

OPTIONS

-k

--keep-going

Keep processing past failed rebases.

-n

--no-fetch

Skip the `git fetch` phase of rebase-update.

-v

--verbose

More text than your terminal can handle.

--current

Only rebase the current branch.

CONFIGURATION VARIABLES

depot-tools.rebase-update.starting-branch

When `git rebase-update` first runs, it will record the current branch here so that when it completes successfully, it will return back to the same branch you started on, even if `git rebase-update`is interrupted due to rebase conflicts. When `git rebase-update` completes successfully, this configuration variable is removed.

branch.<name>.dormant

If `true`, will cause rebase-update to skip all processing on the branch. Useful for old/high-conflict branches which you want to keep for posterity, but don't want to deal with when running `git rebase-update`

branch.<name>.base

Holds the *base* reference for this branch. By default this is equivalent to `git merge-base <name> <name>@{upstream}`.

However, it can diverge
if `<name>@{upstream}` is manually rebased.
In this case, it correctly preserves the value it had before, where `git merge-base` would now report the wrong value.

All of the tools in the depot_tools(1) suite collude to keep this value as up-to-date as possible, including git-reparent-branch(1), and git-new-branch(1). git-map(1) also shows the location of these marker values in **white**.

git-mark-merge-base(1) allows easy manual interaction for this value, in the unlikely event that it gets out of sync.

SUGGESTED ALIASES

Some common short-hand aliases. Feel free to add these to your *~/.gitconfig* file.

```
[alias]
  git reup = rebase-update
```

SEE ALSO

git-new-branch(1), git-reparent-branch(1),git-rename-branch(1), git-upstream-diff(1), git-freeze(1), git-mark-merge-base(1)