# Gerrit/Advanced usage

The basic instructions to set up and operate with Git and Gerrit are described at the [Tutorial](http://www.mediawiki.org/wiki/Gerrit/Tutorial) **Contents**

* [1 Warning](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Warning)
* [2 Installation](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Installation)
  + [2.1 Setup SSH shortcut (optional)](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Setup_SSH_shortcut_.28optional.29)
* [3 Submitting patches](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Submitting_patches)
  + [3.1 Howto - Merging your amend back into your branch](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Howto_-_Merging_your_amend_back_into_your_branch)
  + [3.2 Submitting a change to a branch for review ("backporting")](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Submitting_a_change_to_a_branch_for_review_.28.22backporting.22.29)
  + [3.3 Acting on remote branches](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Acting_on_remote_branches)
    - [3.3.1 Pushing having used automatic setup](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Pushing_having_used_automatic_setup)
    - [3.3.2 Pushing having used manual (Windows) setup](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Pushing_having_used_manual_.28Windows.29_setup)
  + [3.4 Committing to non master](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Committing_to_non_master)
  + [3.5 Partial revert of previous commit](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Partial_revert_of_previous_commit)
  + [3.6 Troubleshooting](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Troubleshooting)
    - [3.6.1 SSH and "permission denied (publickey)"](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#SSH_and_.22permission_denied_.28publickey.29.22)
    - [3.6.2 "[remote rejected] master -> master" and "failed to push some refs"](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#.22.5Bremote_rejected.5D_master_-.3E_master.22_and_.22failed_to_push_some_refs.22)
    - [3.6.3 "[remote rejected] HEAD -> refs/publish/master/??? (Cannot merge change to another branch)](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#.22.5Bremote_rejected.5D_HEAD_-.3E_refs.2Fpublish.2Fmaster.2F.3F.3F.3F_.28Cannot_merge_change_to_another_branch.29)
    - [3.6.4 Email doesn't match](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Email_doesn.27t_match)
    - [3.6.5 Build failed due to merge conflict](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Build_failed_due_to_merge_conflict)
    - [3.6.6 Working tree is dirty](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Working_tree_is_dirty)
    - [3.6.7 missing Change-Id in commit message](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#missing_Change-Id_in_commit_message)
    - [3.6.8 Don't push](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Don.27t_push)
    - [3.6.9 git review complains about multiple commits](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#git_review_complains_about_multiple_commits)
    - [3.6.10 git review doesn't like merge commits](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#git_review_doesn.27t_like_merge_commits)
  + [3.7 Unlink bogus dependencies (rebase changes)](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Unlink_bogus_dependencies_.28rebase_changes.29)
  + [3.8 Create a dependency](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Create_a_dependency)
  + [3.9 Splitting a commit in smaller one](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Splitting_a_commit_in_smaller_one)
* [4 Working on an existing change set](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Working_on_an_existing_change_set)
* [5 Reviewing code](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Reviewing_code)
  + [5.1 Viewing and commenting on code](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Viewing_and_commenting_on_code)
  + [5.2 How to comment on, review, and merge code in Eclipse](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#How_to_comment_on.2C_review.2C_and_merge_code_in_Eclipse)
  + [5.3 How to review and merge code via command line](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#How_to_review_and_merge_code_via_command_line)
  + [5.4 Troubleshooting](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Troubleshooting_2)
    - [5.4.1 Your change could not be merged due to a path conflict](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Your_change_could_not_be_merged_due_to_a_path_conflict)
    - [5.4.2 Your change requires a recursive merge to resolve](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Your_change_requires_a_recursive_merge_to_resolve)
    - [5.4.3 Your change isn't merged](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Your_change_isn.27t_merged)
    - [5.4.4 You need to override Jenkins](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#You_need_to_override_Jenkins)
* [6 How to create a repository ("Gerrit project")](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#How_to_create_a_repository_.28.22Gerrit_project.22.29)
* [7 Other tips](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Other_tips)
  + [7.1 Code Review links](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Code_Review_links)
  + [7.2 Gerrit review scores](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#Gerrit_review_scores)
* [8 See also](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage#See_also)

Debian/Ubuntu: sudo apt-get install git

### Configure Git

git config -l

git config --global user.email "example@wikimedia.org"

git config --global user.name "example"

### Generate a new SSH key

To generate a new SSH key, enter the code below. We want the default settings so when asked to enter a file in which to save the key, just press enter.

$ ssh-keygen -t rsa -C "your\_email@youremail.com"

### Add ssh key to your gerrit account

Log into the [web interface for gerrit](https://gerrit.wikimedia.org). Click on your username in the top right corner, then choose "Settings". On the left you will see SSH PUBLIC KEYS. Paste your SSH Public Key into the corresponding field. Alternately, once logged in, use these direct links to add your public key to [gerrit](https://gerrit.wikimedia.org/r/#/settings/ssh-keys) and [wikitech](https://wikitech.wikimedia.org/wiki/Special:NovaKey).

### Add ssh key to use with git

Start Git Bash.

* Get ssh-agent running using

$ eval `ssh-agent`

Be sure to use the accent " ` " located under the tilde " ~ ", not the single quote " ' ".

* Add your private key to the agent (use path to your private key file)

$ ssh-add .ssh/id\_rsa

if the key is in the default location, use $ ssh-add ~/.ssh/id\_rsa

Key should be in OpenSSH format (.ppk keys can be exported to this format in PuTTyGen).

* Run ssh. (Do be paranoid and check for the matching [SSH fingerprint for gerrit.wikimedia.org:29418](https://wikitech.wikimedia.org/wiki/Help:SSH_Fingerprints/gerrit.wikimedia.org:29418) when logging in for the first time.)

$ ssh <USERNAME>@gerrit.wikimedia.org -p 29418

**Warning**

This page mostly documents how to do things "the hard way" in gerrit. The git review tool continues to improve, and now contains built-in mechanisms to push to a branch, upload a set of dependent patches, etc. You should probably review man git review before deciding that you need to proceed further here.

**Installation**

**Setup SSH shortcut (optional)**

It's easier to access the repository if you don't have to specify the full *myname*@gerrit.wikimedia.org:29418 every time. You can edit your ~/.ssh/config file and add

Host gerrit

Hostname gerrit.wikimedia.org

Port 29418

User *yourusername*

Then you can use "gerrit" instead.

*git review -s adds a gerrit remote to git which should make this step unnecessary.* [*Cscott*](http://www.mediawiki.org/wiki/User:Cscott) *(*[*talk*](http://www.mediawiki.org/wiki/User_talk:Cscott)*)*

**Submitting patches**

**Howto - Merging your amend back into your branch**

*This section is optional. It's offered as a convenience way to offer your a solution to a common problem. At this stage, your new changeset is already in Gerrit.*

After you have amended your change, you may want to merge it back into your local branch.

You can do this by going to the Gerrit change in question.

Here is an example:

<https://gerrit.wikimedia.org/r/#/c/7669/4>

Go to the *Download* section and copy cherry pick.

We will select patch set 4.

Switch back to your branch. You will be in your review branch where you just made your change.

Note: use the branch relevant to your change number.

git checkout mingle-fr-2012-59

Paste in the cherry pick and merge any conflicts.

git fetch ssh://<USERNAME>@gerrit.wikimedia.org:29418/mediawiki/extensions/DonationInterface refs/changes/69/7669/4 && git cherry-pick FETCH\_HEAD

Perform a git add on the modified files.

git add payflowpro\_gateway/payflowpro.adapter.php

Do not forget to check your status and run a diff.

git diff

You should see there are no differences:

diff --cc payflowpro\_gateway/payflowpro.adapter.php

index d7e510a,738c9df..0000000

--- a/payflowpro\_gateway/payflowpro.adapter.php

+++ b/payflowpro\_gateway/payflowpro.adapter.php

Then commit the changes:

git commit -m 'Merging patch set 4.'

[mingle-fr-2012-59 4e82e5a] Merging patch set 4.

1 files changed, 3 insertions(+), 3 deletions(-)

**Submitting a change to a branch for review ("backporting")**

[Backporting fixes](http://www.mediawiki.org/wiki/Backporting_fixes) discusses backporting changes to MediaWiki core (coordinate with the [MediaWiki Release Manager](http://www.mediawiki.org/wiki/MediaWiki_Release_Manager), bugzilla flags, etc.)

In this example, we'll backport [Gerrit #Ib27792](https://gerrit.wikimedia.org/r/#/q/Ib27792b661666ac5f704eeaaadcdd9668dd81b56,n,z) from master to REL1\_20. The basic idea is to use git cherry-pick to apply the changes from the commit to master to a different branch.

Before you start, look up the git commit hash of the commit that was merged into master. This can be found on the Gerrit change page. Scroll down to the last Patch Set, and the git commit hash is between "Patch Set *NN*" and "(gitweb)" (not to be confused with the *Gerrit Change id* which starts with a capital 'I'). Make sure that this commit was indeed merged into the master branch. If it wasn't then wait until it has been reviewed and merged in master — the commit may still be amended and we don't want to merge an old version).

$ git fetch origin

# The git commit hash of the change in master.

$ git show d4f2c0e8f76a7634fce1631669f4ce037965d8b5

$ git checkout origin/REL1\_20

$ git reset --hard origin/REL1\_20 # Ensure latest version, undo any local dependencies

$ git cherry-pick d4f2c0e8f76a7634fce1631669f4ce037965d8b5

# Do not change the commit message. In particular leave the

# "Change-Id" intact at the bottom of the message, since this is

# what Gerrit uses to relate the master change and the branch merge.

# If the merge causes conflicts, you should fix them manually,

# use git add <files> and run git commit. Move the "Conflicts" section

# of the commit message before "Change-Id", so "Change-Id" remains at

# the bottom of the message, otherwise the push will be rejected.

# Verify history looks as expected

$ git log --graph --decorate --oneline -n5

# View the original change in Gerrit and look up the topic-name,

# then use it below in place of "topic-name", e.g. "refs/for/REL1\_20/bug/36151"

# or "refs/for/wmf/1.21wmf1/my-topic-name"

$ git push origin HEAD:refs/for/REL1\_20/bug/36151

remote:

remote: New Changes:

remote: https://gerrit.wikimedia.org/r/25756

remote:

\* [new branch] HEAD -> refs/for/REL1\_20/bug/36151

*Is there any need to use complicated git push instead of git review here?* -- [S Page (WMF)](http://www.mediawiki.org/wiki/User:S_Page_%28WMF%29) ([talk](http://www.mediawiki.org/wiki/User_talk:S_Page_%28WMF%29)) 02:28, 17 May 2013 (UTC)

The only reason would be so that you can give it a new topic. So alternatively, after the git log check, look up the topic name first (or think of a new topic) and do git checkout -b random-new-topic-name then followed by regular git review -R (instead of git push); as of git-review version 1.23, it will reuse the original topic.

git review -R remote-branch-name works too if you want to push to a remote branch from a review branch.

As a result:

* <https://gerrit.wikimedia.org/r/25756> is created for review.
* [Gerrit #Ib27792](https://gerrit.wikimedia.org/r/#/q/Ib27792b661666ac5f704eeaaadcdd9668dd81b56,n,z) shows both the original and the merge changes

**Acting on remote branches**

By default, your local clone will only have a local master branch set up to track the remote master branch. Tracking means that whenever you fetch objects from the remote repository, git status or git branch will be able to tell you how up-to-date is your local branch, which is very useful. So, whenever you want to regularly act on a remote branch (lets says REMOTE\_BRANCH, you want to setup a one locally (REMOTE\_BRANCH too to easily remember about it) that track it (with -t).

git branch -vv will give the full details:

$ git clone ...

$ git checkout -b REL1\_19 -t gerrit/REL1\_19

$ git branch -vv

REL1\_19 3b2bfd3 [gerrit/REL1\_19: ahead 1] .gitreview for REL1\_19 branch

\* master 13169c8 [gerrit/master: behind 1] \* (bug 34212) ApiBlock/ApiUnblock a[...]

$

**Pushing having used automatic setup**

From [bug 35456](https://bugzilla.wikimedia.org/show_bug.cgi?id=35456) : git-review accepts, as an optional argument, the branch name to interact with. When that argument is not specified, it falls back to look for the defaultbranch parameter in a .gitreview file at the root of the repository.

Every branch should have a .gitreview having a correct defaultbranch value. For mediawiki/core.git, else people will have to use something like: git-review BRANCH\_NAME.

|  |  |
| --- | --- |
|  | By default git-review uses a branch named 'master' which might not be what you want. Check the .gitreview file to be sure. |

**Pushing having used manual (Windows) setup**

To change where you push to for review having performed a manual setup, run git config alias.push-for-review "push gerrit HEAD:refs/for/BRANCH\_NAME" to override the local alias, then use git push-for-review as per usual.

**Committing to non master**

To make a change to the 1.17 branch, create a branch and tag, and push both:

git checkout -b REL1\_17 origin/REL1\_17

<make code changes>

git add <files-changed>

git commit

git push gerrit REL1\_17

git tag 1.17.3

git push --tags

**Partial revert of previous commit**

git show <commit> -- <path> | git apply -R

<commit> Can be found in gerrit patch view in small letters next to text *Patch Set N*. Then push for review normally.

**Troubleshooting**

**SSH and "permission denied (publickey)"**

If you get the error

Permission denied (publickey).

fatal: The remote end hung up unexpectedly

Then you're not logged in to your ssh key right now. Solution: do ssh-add ~/.ssh/id\_rsa to make it prompt you for the passphrase for your key and add it to the active keychain. Then you can check what's in your keychain with ssh-add -l. Then try pushing for review again.

The fingerprint of the Gerrit server is

dc:e9:68:7b:99:1b:27:d0:f9:fd:ce:6a:2e:bf:92:e1

so you can say yes when it asks you to add that fingerprint to the known hosts file.

Keep in mind that gerrit listens on port 29418 and if for some reason you forgot to specify port number, you might be hitting "normal" SSH daemon listening on port 22 (this one has RSA key fingerprint b5:e9:dc:b2:dd:6e:70:f7:18:8a:dc:a3:5d:ab:99:4d).

To check whether SSH connectivity and public key authentication work you can use:

ssh -p 29418 username@gerrit.wikimedia.org

\*\*\*\* Welcome to Gerrit Code Review \*\*\*\*

Hi username, you have successfully connected over SSH.

Unfortunately, interactive shells are disabled.

To clone a hosted Git repository, use:

git clone ssh://username@gerrit.wikimedia.org:29418/REPOSITORY\_NAME.git

Connection to gerrit.wikimedia.org closed.

**"[remote rejected] master -> master" and "failed to push some refs"**

If you forget to add HEAD:refs/for/master as an alias, you will receive something along the lines of:

git push

Counting objects: 5, done.

Delta compression using up to 8 threads.

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 709 bytes, done.

Total 3 (delta 1), reused 0 (delta 0)

remote: Resolving deltas: 0% (0/1)

To ssh://username@gerrit.wikimedia.org:29418/test/mediawiki/extensions/examples

! [remote rejected] master -> master (prohibited by Gerrit)

error: failed to push some refs to 'ssh://username@gerrit.wikimedia.org:29418/test/mediawiki/extensions/examples'

This means you tried to commit to branch "master" instead of submitting your changes for review.

**"[remote rejected] HEAD -> refs/publish/master/??? (Cannot merge change to another branch)**

When attempting to cherry-pick a change or merge an entire branch and then submit for review; gerrit will be very confused if you do not explicitly tell it what branch you're modifying. You might get errors like:

! [remote rejected] HEAD -> refs/publish/master/mwalker\_test (change 59546 closed)

or

! [remote rejected] HEAD -> refs/publish/master/mwalker\_test (no new changes)

In this case you must ensure that your branch's .gitreview file has the correct defaultbranch option (e.g. it should be pointing to the branch you're attempting to modify) or you must manually push the refspec using git push gerrit HEAD:refs/for/<branch>

**Email doesn't match**

remote: ERROR: committer email address (email)

remote: ERROR: does not match your user account.

There are two possible problems that could cause this error. If the email address that git pops back \_is\_ the email address you intend to use with Gerrit, then you should add that email address in Gerrit, and make sure you click the confirmation link in the email Gerrit sends you. Then try pushing again.

If, however, git sends back some nonsense email (like one you don't use anymore, or a local mail address like root@localhost), you should do the following:

$ git config --global user.email yournewemail@example.org

$ git commit --amend --reset-author

To do it locally for just that repo, instead do:

$ git config user.email yournewemail@example.org

$ git commit --amend --reset-author

Then, try pushing again.

**Build failed due to merge conflict**

*See also:* [*Gerrit/resolve conflict*](http://www.mediawiki.org/wiki/Gerrit/resolve_conflict)

After `git review` jenkins-bot emails ***This change was unable to be automatically merged with the current state of the repository. Please rebase your change and upload a new patchset.*** This might mean that server master branch now has merge conflicts with your patch.

* Make sure your master branch hard-reset to the gerrit's master (to avoid [git review complaining about multiple commits](http://www.mediawiki.org/wiki/Gerrit/Tutorial#git_review_complains_about_multiple_commits))

$ git checkout master

$ git fetch --all

$ git reset --hard origin/master

* Checkout for review, rebase, recommit

$ git review -d <patchNumber>

$ git rebase master

# Fix merge conflicts and add them with 'git add'

$ git rebase --continue

$ git review -R

If the error occurs even though you have rebased your patch as above, try making an arbitrary change to your commit message, and then running git review again (related to [bug 53895](https://bugzilla.wikimedia.org/show_bug.cgi?id=53895)).

**Working tree is dirty**

Note: if upon doing 'git review' you receive a message "Working tree is dirty" try doing 'git add' for the file(s) changed (or created), then 'git commit', and then 'git review'. This was seen on OSX with an older git client.

**missing Change-Id in commit message**

Note: if upon doing 'git review' you receive a message about 'missing Change-Id', then your /.git/hooks/commit-msg is probably incorrect. it should look something like:

CHANGE\_ID\_AFTER="Bug|Issue"

MSG="$1"

# Check for, and add if missing, a unique Change-Id

#

add\_ChangeId() {

clean\_message=`sed -e '

/^diff --git a\/.\*/{

s///

q

}

/^Signed-off-by:/d

/^#/d

' "$MSG" | git stripspace`

if test -z "$clean\_message"

then

return

fi

You will also get a missing Change-ID message when trying to merge (git cherry-pick) some change from git that does not have Change-ID. It seems that the hook isn't called by cherry-pick, but it is fortunately called by git commit -c *some-commit-id*.

In the example below we will be moving trivial change [bd7a268e4be2da23ba0b9943c3b0ba1ac88294dc](https://gerrit.wikimedia.org/r/gitweb?p=mediawiki/core.git;a=commit;h=bd7a268e4be2da23ba0b9943c3b0ba1ac88294dc) from git master in the mediawiki/core project (it came via SVN trunk) to REL1\_19.

[Sample session](http://tools.wikimedia.pl/%7Esaper/fail/cherry-pick-to-branch-message-fail) to exhibit this problem:

Script started on Sun Mar 25 01:51:46 2012

$ git checkout REL1\_19

Switched to branch 'REL1\_19'

$ git log --pretty=oneline -n 1

9074142447ab0bd06f4e6a7ca7c9d8c70d33d109 \* (bug 35449) Removed double call to OutputPage::setRobotPolicy() in

SpecialWatchlist::execute() (the other one is hidden in SpecialPage::setHeaders()) \* Special:Watchlist no

longer sets links to feed when the user is anonymous

$ git show bd7a268e4be2da23ba0b9943c3b0ba1ac88294dc

commit bd7a268e4be2da23ba0b9943c3b0ba1ac88294dc

Author: Marcin Cieślak <saper@users.mediawiki.org>

Date: Wed Mar 14 00:36:11 2012 +0000

Cosmetic improvements to PostreSQL updater output

\* Don't WARN on sequences already existing

\* Align dots nicely to the rest

diff --git a/includes/installer/PostgresUpdater.php b/includes/installer/PostgresUpdater.php

index d1fc6f7..d4412cb 100644

--- a/includes/installer/PostgresUpdater.php

+++ b/includes/installer/PostgresUpdater.php

@@ -394,7 +394,7 @@ END;

protected function renameSequence( $old, $new ) {

if ( $this->db->sequenceExists( $new ) ) {

- $this->output( "WARNING sequence $new already exists\n" );

+ $this->output( "...sequence $new already exists.\n" );

return;

*(...full trivial diff to one file omitted...)*

$ git cherry-pick -x bd7a268e4be2da23ba0b9943c3b0ba1ac88294dc

[REL1\_19 a354acd] Cosmetic improvements to PostreSQL updater output

Author: Marcin Cieślak <saper@users.mediawiki.org>

1 files changed, 18 insertions(+), 18 deletions(-)

$ git log HEAD ^FETCH\_HEAD

commit a354acd879c3dd840e7be1e3c6d6fc78d696631d

Author: Marcin Cieślak <saper@users.mediawiki.org>

Date: Wed Mar 14 00:36:11 2012 +0000

Cosmetic improvements to PostreSQL updater output

\* Don't WARN on sequences already existing

\* Align dots nicely to the rest

(cherry picked from commit bd7a268e4be2da23ba0b9943c3b0ba1ac88294dc)

$ git push gerrit HEAD:refs/for/REL1\_19/PostgreSQL

Counting objects: 9, done.

Delta compression using up to 8 threads.

Compressing objects: 20% (1/5)

Compressing objects: 40% (2/5)

Compressing objects: 60% (3/5)

Compressing objects: 80% (4/5)

Compressing objects: 100% (5/5)

Compressing objects: 100% (5/5), done.

Writing objects: 20% (1/5)

Writing objects: 40% (2/5)

Writing objects: 60% (3/5)

Writing objects: 80% (4/5)

Writing objects: 100% (5/5)

Writing objects: 100% (5/5), 750 bytes, done.

Total 5 (delta 4), reused 0 (delta 0)

remote: Resolving deltas: 0% (0/4)

remote: Resolving deltas: 0% (0/4)

remote: ERROR: missing Change-Id in commit message

remote: Suggestion for commit message:

remote: Cosmetic improvements to PostreSQL updater output

remote:

remote: \* Don't WARN on sequences already existing

remote: \* Align dots nicely to the rest

remote: (cherry picked from commit bd7a268e4be2da23ba0b9943c3b0ba1ac88294dc)

remote:

remote: Change-Id: Ia354acd879c3dd840e7be1e3c6d6fc78d696631d

To ssh://saper@review:29418/mediawiki/core.git

! [remote rejected] HEAD -> refs/for/REL1\_19/PostgreSQL (missing Change-Id in commit message)

error: failed to push some refs to 'ssh://saper@review:29418/mediawiki/core.git'

$

Script done on Sun Mar 25 01:54:08 2012

To fix this, use -n *(don't commit)* option to git cherry-pick:

git cherry-pick -n bd7a268e4be2da23ba0b9943c3b0ba1ac88294dc

git commit -c bd7a268e4be2da23ba0b9943c3b0ba1ac88294dc

Unfortunately, if you want to add the original commit ID to the message (as done by git cherry-pick -x) you have to add it yourself.

The change above has been submitted as [3c88c61f1b7e36d5d374a42bb0f50783ab5391a4](https://gerrit.wikimedia.org/r/gitweb?p=mediawiki%2Fcore.git;a=commit;h=3c88c61f1b7e36d5d374a42bb0f50783ab5391a4) for REL1\_19 review.

**Don't push**

If you attempt to do 'git push' after doing 'git commit' you may receive a response 'Everything up-to-date'. **You have not pushed to the branch.** You have to do 'git review' to move your changes to gerrit, and only from gerrit will the branch be updated. This seems to be a side effect of checking out master as a branch as of February 2012.

In some projects (e.g. test/) it is possible to do 'git push' instead of 'git review' and have the push succeed. It is probably better not to do that, as it confuses those who find your changes later and don't know where they came from.

**git review complains about multiple commits**

If git review asks you if you really want to submit multiple commits, and lists a bunch of unrelated commits from different branches, try this:

git fetch --all

git remote update

* Or better yet, use git review -u to "Skip cached local copies and force updates from network resources". --[Cscott](http://www.mediawiki.org/wiki/User:Cscott) ([talk](http://www.mediawiki.org/wiki/User_talk:Cscott))
* If the error persists, [git rebase -i origin/master](https://stackoverflow.com/a/6934882/1333493) usually fixes everything just by following instructions on screen.

**git review doesn't like merge commits**

If you merged a development branch, and now want to submit a merge commit to Gerrit, git review may not let you. It may ask you for submitting lots of changes from one of the merged branches, or otherwise mangle the commit. To avoid this, push the commit directly to Gerrit, bypassing git review:

git push gerrit HEAD:refs/for/master

For more information, see the [Gerrit documentation](http://gerrit-documentation.googlecode.com/svn/Documentation/2.3/user-upload.html#push_create).

**Unlink bogus dependencies (rebase changes)**

Example for <https://gerrit.wikimedia.org/r/#change,5154>

git fetch --all # To make sure we have latest changes

git review -d Ie6e3c9be

git rebase -i gerrit/master # Delete the commits you want to get rid of

git commit --amend # Add a note

git review -f # -f deletes the branch after submit

Maybe also have look at [Gerrit/resolve conflict](http://www.mediawiki.org/wiki/Gerrit/resolve_conflict)

**Create a dependency**

If you are about to create a patch that depends on another (unmerged) patch, or if you already submitted a patch but need to fix the dependency (i.e. currently it is based on master and would break if merged without the dependency, or maybe you squashed your change on top of the dependency). Then this is the section you're looking for. If you want to fix the patch to have the right dependency rather than to create a new patch with a dependency, then make sure your working copy is clean (no uncommitted changes).

git fetch --all # Make sure we have latest info from the repository

git review -d 1234 # Gerrit change number of the change you want as dependency ("parent")

Now we need to make sure the patch has the correct git-parent. Depending on whether you are creating a new patch or fixing an existing patch, there is two different ways to do this. If you are starting fresh:

git checkout -b bug/1234 # Creates a new branch, with the current branch (the dependency) as parent

# Edit files: make your changes

git add someFile.php some/other/file.js

git commit # Commit your patch

git log -n5 --decorate --pretty=oneline # Verify that the last 5 entries of the log now start with:

# \* (HEAD, bug/1234) your change

# \* (review/john/700) the dependency

# \* (gerrit/master)

git push gerrit HEAD:refs/for/master # Use this instead of `git review`

*git review works fine as the last line here. no need to use git push.* [Cscott](http://www.mediawiki.org/wiki/User:Cscott) ([talk](http://www.mediawiki.org/wiki/User_talk:Cscott))

If you need to amend your patch to have the correct dependency:

git branch # Take note of the review/\* branch that was created for this, it has an "\*" in front of it

git checkout bug/1234 # Check out the local topic branch of your change

git rebase review/john/7000 # The branch name of the gerrit change we checked out earlier

# Resolve conflicts if needed,

# - use "git status" to see the files that need resolution

# - after fixing it in your editor, "git add filename" for each of the fixed files

git rebase --continue

git log -n5 --decorate --pretty=oneline # Verify that the last 5 entries of the log now start with:

# \* (HEAD, bug/1234) your change

# \* (review/john/700) the dependency

# \* (gerrit/master)

git push gerrit HEAD:refs/for/master # Use this instead of `git review`

*Again, just use git review for the last line.* [Cscott](http://www.mediawiki.org/wiki/User:Cscott) ([talk](http://www.mediawiki.org/wiki/User_talk:Cscott))

The git push directly will make sure that the dependency is kept and the other revisions that are already in gerrit will not be re-submitted.

*git review is smart enough not to touch the already-present revisions.* [Cscott](http://www.mediawiki.org/wiki/User:Cscott) ([talk](http://www.mediawiki.org/wiki/User_talk:Cscott))

Create a [git alias](http://www.mediawiki.org/wiki/Git/aliases#log-nice) to get shortcuts like "git log-nice" for commands such as git log --decorate --pretty=oneline.

If you wish to set a topic, push instead with the following syntax:

git push gerrit HEAD:refs/for/master/myawesometopic # Append your topic name

*The -t option to git review is an easier way to do this.* [Cscott](http://www.mediawiki.org/wiki/User:Cscott) ([talk](http://www.mediawiki.org/wiki/User_talk:Cscott))

**Splitting a commit in smaller one**

Explained in detail at [Gerrit/split a submitted change](http://www.mediawiki.org/wiki/Gerrit/split_a_submitted_change).

**Working on an existing change set**

Sometimes you want to work on a change set started by some else and then upload your changes as a new patch set.

# Note in the gerrit URL the number reference to the change set,

# e.g., https://gerrit.wikimedia.org/r/#/c/70112/, thus 70112

# In your local copy of the master branch, pull down the change set

# and switch to that branch with the following command.

git review -d 70112

# Make any necessary changes and commit them as an amendment,

# adding appropriate comments to the commit message.

git commit --all --amend

# Push the patch set up to gerrit as usual.

git review -R

# Other developers can then update their local copy of the change set

# with the following command.

git review -d 70112

**Reviewing code**

**Viewing and commenting on code**

The basic functionality is explained in [the Git and Gerrit tutorial](http://www.mediawiki.org/wiki/Gerrit/Tutorial#Viewing_and_commenting_on_code).

Some extra bits:

* **Old Version History** dropdown menu. This menu will allow you to change what changes you're reviewing. This is helpful if you reviewed a past changeset, and want to make sure your changes were taken into account. Rather than reading through the entire changeset diff'd against the base commit, you can read only the differences between the current changeset and the changeset you reviewed. There's a bonus, too: You can see your comments on the left hand side of the If there was a rebase commit, there will be garbage in the diffs, but you can read things one changeset at a time and it will still be faster.
* **Diff All Unified** button:
* Opens the diff(s) in a new tab. You can double-click on a line and comment on that line, then save a draft comment! Then, click "Up to change" to go back to the changeset.
* For commits that contain whitespace changes (i.e. indent a block that was changed), it is best to set the diff-preferences appropriately to make it easier to review. When viewing a diff, on top there is a link "Preferences". Then there is two important settings to focus on. "*Ignore Whitespace*" and "*Intraline Difference*". The last one (*Intraline Difference*) is especially useful if a block of code was indented, as this setting will show the added tabs themselves allowing other changes to be recognizable without having to compare every word in your mind (see screenshot).

**How to comment on, review, and merge code in** [**Eclipse**](http://en.wikipedia.org/wiki/Eclipse_%28software%29)

As an alternative to Gerrit's web interface, you can also review code from [Eclipse](http://www.eclipse.org/downloads/) using the [Mylyn](http://www.eclipse.org/mylyn/) task-management framework. To get started, download and install Eclipse, and then install Mylyn from the [Install New Software](http://help.eclipse.org/juno/index.jsp?topic=/org.eclipse.platform.doc.user/tasks/tasks-124.htm) menu (as of Oct 5th, 2013 you need the [snapshots](http://download.eclipse.org/reviews/nightly) update site to use the Wikimedia Gerrit installation). When you next launch Eclipse, you'll be prompted to add a task for Mylyn. From there, you'll need to install the connector for Gerrit, specify <https://gerrit.wikimedia.org/r/> as the server URL, and add your username and password.

Diff / comment interface in Eclipse

**How to review and merge code via command line**

Using [dippy-bird](https://gerrit.wikimedia.org/r/gitweb?p=mediawiki/tools/dippybird.git) you can easily do command line review and merging. The query parameter is the change you want to deal with.

php dippy-bird.php --username=USERNAME --server=gerrit.wikimedia.org --port=29418 --action=submit --query=12345

You can therefore use that to approve a range of commits:

#!/bin/bash

for i in {51541..51545}

do

php dippy-bird.php --username=USERNAME --server=gerrit.wikimedia.org --port=29418 --action=submit --query=$i

done

**Troubleshooting**

**Your change could not be merged due to a path conflict**

Answered at [Gerrit/resolve conflict](http://www.mediawiki.org/wiki/Gerrit/resolve_conflict).

**Your change requires a recursive merge to resolve**

If you get the error "Your change requires a recursive merge to resolve", you need to rebase the change set against master.

1. Make sure your master branch is up to date: git pull origin master
2. Create and switch to a new branch in which to checkout the change set with the conflict: git checkout -b BRANCHNAME
3. Checkout the conflicting change set into this branch. You can copy/paste the correct command from the 'Download' section in the Gerrit review. It will look something like this: git fetch <ssh://awjrichards@gerrit.wikimedia.org:29418/mediawiki/extensions/MobileFrontend> refs/changes/14/3414/3 && git checkout FETCH\_HEAD
4. Rebase against master: git rebase master
5. Push the change to Gerrit for review: git review
6. Re-review the change set in Gerrit, and then submit the changes to be merged to master.

**Your change isn't merged**

If someone enters +2 Code Reviewed, it should trigger a series of automated builds and tests. [Continuous integration/Workflow](http://www.mediawiki.org/wiki/Continuous_integration/Workflow) describes the flow.

In Gerrit, the "user" jenkins-bot should add a comment

Starting gate-and-submit jobs. <https://integration.wikimedia.org/zuul/>

Follow the link to see the progress of the tests for your change that [Zuul](http://www.mediawiki.org/wiki/Zuul) has submitted to [Jenkins](http://www.mediawiki.org/wiki/Jenkins).

If you don't see the "Starting gate-and-submit jobs" comment in gerrit, look at <https://integration.wikimedia.org/zuul/> It shows everything that Zuul has submitted to Jenkins, and the "Queue length" number on top is the number of events not yet processed. If your job doesn't appear below this on the page below but the queue length is non-zero, it is likely in the queue; Zuul and Jenkins are probably busy and you just have to wait. But if the queue length is 0 and your change doesn't appear, it means it isn't going to happen and you need to resubmit it.

If the required tests all pass (note that some are "non-voting"), then jenkins-bot will add a comment "Build succeeded" followed by "Change has been successfully merged into the git repository."

**You need to override Jenkins**

First of all, you should more or less never do this. (So far the reason was backporting a change to an old release that had broken tests). If Jenkins has a -2 you generally can't merge the change. What you need to do:

* Removing the jenkins-bot "Verified" review (click the [x])
* Review the latest patch set with "Verified +2" in addition to "Code-Review +2" (this is normally reserved for Jenkins, but since we're overriding it we're replacing its score with our own)
* Click "Publish and Submit"

**How to create a repository ("Gerrit project")**

See ["Request a new Git repository"](http://www.mediawiki.org/wiki/Git/New_repositories). There's a form to fill out. It should get processed very quickly (within a couple of days).

**Other tips**

**Code Review links**

Links to old SVN [Code Review](http://www.mediawiki.org/wiki/Special:Code/MediaWiki/) revisions are stored in commit notes. They may be fetched for display in the git log using the following command:

git fetch gerrit refs/notes/commits:refs/notes/commits

Note this must be done separately for each git repository.

**Gerrit review scores**

As above, they are stored in commit notes and may be fetched using:

git fetch gerrit refs/notes/review:refs/notes/review

To display them in git log (similar syntaxes work for related tools):

git log --notes=review

<http://gitref.org/basic/>

## git init initializes a directory as a Git repository

To create a repository from an existing directory of files, you can simply run git init in that directory. For example, let's say we have a directory with a few files in it, like this:

**$ cd konnichiwa**

**$ ls**

README hello.rb

This is a project where we are writing examples of the "Hello World" program in every language. So far, we just have Ruby, but hey, it's a start. To start version controlling this with Git, we can simply run git init.

**$ git init**

Initialized empty Git repository in /opt/konnichiwa/.git/

Now you can see that there is a .git subdirectory in your project. This is your Git repository where all the data of your project snapshots are stored.

**$ ls -a**

. .. .git README hello.rb

Congratulations, you now have a skeleton Git repository and can start snapshotting your project.

**In a nutshell**, you use git init to make an existing directory of content into a new Git repository. You can do this in any directory at any time, completely locally.

## [docs](http://git-scm.com/docs/git-clone)   [book](http://git-scm.com/book/en/Git-Basics-Getting-a-Git-Repository#Cloning-an-Existing-Repository) git clone copy a git repository so you can add to it

If you need to collaborate with someone on a project, or if you want to get a copy of a project so you can look at or use the code, you will clone it. You simply run the git clone [url] command with the URL of the project you want to copy.

**$ git clone git://github.com/schacon/simplegit.git**

Initialized empty Git repository in /private/tmp/simplegit/.git/

remote: Counting objects: 100, done.

remote: Compressing objects: 100% (86/86), done.

remote: Total 100 (delta 35), reused 0 (delta 0)

Receiving objects: 100% (100/100), 9.51 KiB, done.

Resolving deltas: 100% (35/35), done.

**$ cd simplegit/**

**$ ls**

README Rakefile lib

This will copy the entire history of that project so you have it locally and it will give you a working directory of the main branch of that project so you can look at the code or start editing it. If you change into the new directory, you can see the .git subdirectory - that is where all the project data is.

**$ ls -a**

. .. .git README Rakefile lib

**$ cd .git**

**$ ls**

HEAD description info packed-refs

branches hooks logs refs

config index objects