qf

quilt git flow script

Author: Daniel Vetter <daniel.vetter@ffwll.ch>

Date: 2014-05-15

Manual section: 1

Manual group: maintainer tools

SYNOPSIS

qf *command* [*arg* ...]

DESCRIPTION

qf is a workflow script to manage a quilt patch pile on top of a git baseline and track any changes in git itself. The quilt patches and any other metadata is tracked in git refs outside of the normal tag/branch namespace. The current baseline (as a git commit sha1) of the quilt patch pile is also tracked. An integral part is support to export the quilt patch to an git branch in the baseline repository and push/fetch changes to the remote linked to this branch.

Assuming there's already a branch "test" set up to track a remote a quickstart overview (output omitted):

Getting started

$ qf setup test

$ qf checkout test

This automatically either creates a new, empty patch pile or checks out the state of an existing remote.

Note that this only initializes the quilt side of things and presumes that the test branch is set up already, including remote tracking connection. To create a completely new branch hence first run

$ git checkout -b test –track origin/test

Committing changes and publishing the resulting git tree

$ qf git gui # commit any changes in the patch directory

$ qf export && qf push

Rebasing

$ qf baseline # print old baseline commit

$ qf rebase upstream

$ qf baseline # check new baseline commit

Following changes in upstream

$ qf fetch

$ qf git pull && qf checkout test

Doing only a git pull on the quilt branch leads to an inconsitent state if the baseline changed.

Bisecting patch history

Git history in the patches directory records every rebase or any other changes to the patch pile, including the baseline. Which means it can be used for bisecting not just the individual patches, but the history of the entire patch pile:

$ qf git bisect *old-sha1 new-sha1*

to start the bisect process, and then for each bisect step:

$ qf checkout HEAD

Run tests to figure out whether it's part of the new (bad) or old (good) history, then tell git using

$ qf git bisect new|old

COMMANDS

list-aliases

List all aliases for the subcommand names.

See $qf_alias_<alias> under ENVIRONMENT below on how to define aliases.

list-commands

List all subcommand names, including aliases.

setup [*branch-name*]

Sets up a git repository for this quilt worflow script by creating and initializing (if this isn't done yet) the git repository in <git-root>/patches.

If a branch name is given it must exist in the baseline repository. If so a quilt branch of the same name with $QUILT_PREFIX is created and initialized (but without any patches). The remote configuration is copied over from the branch in the baseline repository. The script will fall over if the branch in the baseline repository doesn't have a remote properly set up.

If the quilt branch already exists in the remote the quilt then it is initialized with the latest state (which is updated with git fetch beforehand). The remote quilt branch must have the same name as the branch in the local baseline repository, the script doesn't support disparate tracking names for the quilt branch.

Before the newly created branch can be used it needs to be checked out first.

checkout|co *commit-ish|quilt-branch*

Checks out the given branch/commit-ish (same rules as for git checkout apply) in the patches directory. If you want to check out an earlier revision you might need to run qf fetch first to make sure that the base repo has all the baseline refs. When checking out a branch it's not necessary to add the $QUILT_PREFIX (quilt/ by default) to the branch name, the script will complete the name automatically.

At the end all quilt patches will be pushed. Presuming the quilt flow state hasn't been hand-edited or an incompletely pushed quilt branch has been committed this will always succeed.

rebase *commit-ish*

Pops off all quilt patches, rebases the baseline to the given commit and then reapplys all the quilt patches up to the same patch as before. Then updates the BASELINE variable in patches/config.

clean-patches

Removes all unused patch files from the patches/ directory. This is the same as calling "qf list-unused-patches –purge".

refresh

Refreshes all the quilt patches up to the currently applied patch. Then it commits a wash-up commit with all the refreshed patches. The command aborts if there are uncommited changes in the patches repository.

export

Flattens the current quilt branch and exports it into the respective git branch in the base tree.

The reflog'ed QUILT_EXPORT is always updated with the result of the export. This is useful to export when in the detached HEAD state in the quilt repository hence there's no branch to export to.

export-visualize|ev

Visualizes the latest export result using gitk, down to the current baseline.

push

Exports the tree and then pushes both the quilt branch, the exported branch and any new baselines to the respective remotes.

fetch

Fetches both the main and patches barnch remotes and pulls all the baseline refs into the main repo.

pull

First runs qf fetch, then updates the patches branch, and then checks out the latest working copy. If fails if the patches branch can't be fast forwarded.

stage

Resets the git index and then (re-)applies all currently applied quilt patches to it. Useful to use git tools like git diff to compare changes against the quilt patch state.

conflict-files

List the files that are probably in a unresolved stage of conflict. Wiggle push will leave .rej and .porig files behind. This is useful for manual conflict solving during the quilt rebase.

continue

Clean up after wiggle and continue with quilt push -a to continue applying and rebasing all the following patches.

wiggle-clean

Clean up all .rej and .porig files that wiggle probably left behind.

wiggle-push|wp

Force-push the next patch and then wiggle in any conflicts. Does not refresh the patch automatically, so that the conflict resolution can be cross-checked.

resolved

Little helper when a patch conflict was resolved. First refreshes the topmost patch, then fires up $EDITOR to edit the headers.

apply *patch-name*

Adds a patch to the quilt series and tries to push it.

patch-amend|pa

Open the top most patch in the editor directly to e.g. amend the commit message.

list-unused-patches [*–purge*]

Lists unused patches and if –purge is specified deletes them. Since the quilt patch pile is managed with git itself nothing should get lost.

baseline

Prints out the current baseline sha1.

git|g [*args*]

Run git with the given arguments in the quilt patches directory.

gitk|k [*args*]

Run gitk with the given argumenst in the quilt patches directory.

help

This help text here

usage

Short form usage help listing all subcommands.

all other subcommands - IMPORTANT

Any other subcommands are executed directly in the quilt patches directory as git commands. When using quilt flow in scripts it is import to use the explicit forwarding to avoid clashes with furture extensions.

ALIASES

Extending qf functionalities

It is possible to create your own qf helper and aliases by adding them to $HOME/.qfrc:

```
qf_my_fancy_list_aliases()
{
        echo "Hello world!"
        qf_list_aliases
}

qf_alias_list_aliases=my-fancy-list-aliases
```

ENVIRONMENT

QUILT_PREFIX

Quilt branch prefix. This is a prefix for the git branch that contains the patch files and quilt series file.

CONTRIBUTING, BUG REPORTS AND DISCUSSION

Submit patches, bug reports, and questions for any of the maintainer tools and documentation to the dim-tools@lists.freedesktop.org mailing list.

Please make sure your patches pass the build and self tests by running:

```
$ make check
```

Push the patches once you have an ack from maintainers (Jani/Daniel).