Egg (Draft) Manual

Egg is an attempt to provide a seamless integration of Git within Emacs. Egg is currently tested with Git 1.7.12 and Emacs 23. It may work with other versions of Emacs and Git but it's very difficult for the Egg's developer to investigate and fix bugs that only appear in versions or platforms that he doesn't use. Patches to fix bugs in other emacsen or volunteers to maintain compatibility however are welcome.

Rant

EGG wasn't designed to replace the GIT excellent official porcelain layer, especially when used with the `bash_completion` package. In the EGG's developer's opinion: it's pointless to write a `git-mode` in emacs to let user type *`M-x git-reset`* when he/she can do exactly the same thing in BASH. An emacs interface to GIT should provide the conveniences that would make it easier to accomplish the same task than the GIT's porcelain. EGG, thus, was designed to perform the same goal as GIT ALIASES. Make it more convenient to accomplish common git operations.

One example:

The in EGG'S status buffer, when the user *unstages* a file, EGG will invoke different git operations depend on the origin/status of the file: it was an existing file, it was new file or it was a file with merge conflicts which were just resolved in the index.

Again, EGG was design to help you accomplish many git common operations in very convenient EMACS way.

As with GIT ALIASES, EGG won't save you from learning GIT. Instead it expects its user to be familiar with GIT. This manual tries to document all EGG functionalities and explain the design of some of the commands.

Acknowledgements

In the beginning, Egg was a fork of Marius Vollmer's excellent Magit. While Egg was later completely rewritten, it still keeps the genius *staging/unstaging* metaphor in the *status buffer*.

ByPlayer gracefully picked up the maintainership when Bogolisk went hiatus for several... years!!! He's the current maintainer of Egg and his repo is at: '`https://github.com/byplayer/egg`'

Overview

Using Egg is simple:

- download '`egg.el`' and put in your Emacs's `loadpath`.
- add `(require 'egg)` in your '`.emacs`'
- very important: disable Vc's git backend. Do *`M-x customize-option RET vc-handled-backends`*. Then DEL the Git option.
- open a file in a git repo from within emacs. Et voila!

Using Egg

Egg has many components but the 3 most important ones are:

- The Status Buffer This is where one deals with the Index and Work-Tree:
  - staging/unstaging files/hunks
  - resolving merge conflicts (from merge/rebase/apply/cherry-pick/revert/stash)
  - launch ediff to view delta or conflicts.
  - committing/amending
  - saving/applying stashed work-in-progresses

  See [Status], page 8.
- The Log Buffer This is where one manipulates the repo's DAG:
  - starting merge/rebase/cherry-pick/revert/reset operation. If the operations started from the Log Buffer created a conflict, Egg will open the Status Buffer so you can resolve it.
  - creating branch/tag.
  - marking commits for the *next* INTERACTIVE-REBASE.
  - fetching refs from remote (egg refuses to support pull).
  - pushing refs to remote.
  - viewing commits, use ediff with changes introduced by a commit.

  See [Log], page 14.
- The Minor Mode This mode let you issue git commands (on the buffer) while editing. Some are:
  - staging/unstaging/cancel the current file's modifications.
  - diffing the current file against the Index or another revision
  - search (pickaxing) history for a string/regexp/line.
  - view other revisions of the current file
  - blame revisions for lines in the file, See [Blame], page 19.
  - launch ediff to compare file vs index or other revisions.

  See [File], page 6.

Context-sensitive Key-bindings Menu

Egg heavily uses context-sensitive bindings. The command bound to a key depend where the cursor was. Example: on a commit line in the Log Buffer, *o* will checkout a branch if the cursor was on top of a branch name. On the other hand, it will detatch HEAD and checkout the commit if the cursor was on a branch name.

C-mouse-2 is bound it to the context menu when using menu. There's also a text-based electric context menu (after loading the '`egg-key`') , bound to *kp-enter* by default. In the text-based menu, each line show the key and the description of a command. To run a command you can type the command's key or move the cursor to the line and select the line with *kp-enter*. For example, on a commit line in the Log Buffer, the 3 sequences below will perfom the same action: create a new branch pointing at the commit.

- *B*.
- *kp-enter* to show the menu, then *B*.
- *kp-enter* to show the menu, then move the line marked with *B* then type *kp-enter*.

Each variant of the command that would be invoked with *C-u* would be displayed on a separate line under the main line of the command. For example, *C-u /* will be displayed on a line right after */*. The 3 sequences below ill also perform the same thing: create a new branch even if the name is already in used.

- *C-u B*.
- *kp-enter* to show the menu then *C-u B*.
- *kp-enter* to show the menu, move to the line marked with *C-u B* then select the line with *kp-enter*.

Sections and Subsections

In Egg special buffers, information are usually grouped in hierarchical sections. A section can contains one or more subsections which in turn can contains one or more sub-sub-sections, etc. Sections can hidden/collapsed, turning the whole buffer into a hierachical folding structure. The commands mapped when the cursor is on top of a section are:

h	Toggle the hidden state of the current section. [egg-section-cmd-toggle-hide-show], page 22
H	Toggle the hidden state of the subsections of the current section. [egg-section-cmd-toggle-hide-show-children], page 24
n	Move to the next section. [egg-buffer-cmd-navigate-next], page 22
C-u n	Move to the next section of the same type.
p	Move to the previous section. [egg-buffer-cmd-navigate-prev], page 22
C-u p	Move to the previous section of the same type.
kp-enter	Show a text-based context menu (if '`egg-key`' package was loaded).

Specific types of sections (diff, hunk, commit, etc.) bind extra commands in addition to the commands listed above.

The Egg Minor Mode

Using Egg when Editing Files

When the `egg` library is loaded, `egg-minor-mode` will be activated when visiting a file in a git repository. The mode-line will initially show `Egg`. However, after the first time the repo's status was read by Egg, it will show `Git:branch` where `branch` is the current branch-name. The following commands are mapped in the minor mode.

`C-x v a` Toggle blame mode for the current-file, See [Blame], page 19. [egg-file-toggle-blame-mode], page 19

`C-u C-x v a`
: Do not ask for confirmaton before saving the buffer.

`C-x v c` Open the commit buffer for composing a message. [egg-commit-log-edit], page 25

`C-u C-x v c`
: The message will be use to amend the last commit.

`C-u C-u C-x v c`
: Just amend the last commit with the old message.

`C-x v e` Compare, using ediff, the current file's contents in work-dir with vs a rev. [egg-file-ediff], page 22

`C-x v f` Open a file tracked by git. [egg-find-tracked-file], page 27

`C-x v h` Show the commits in the current branch's DAG that modified the current file. [egg-file-log], page 25

`C-x v i` Add the current's file contents into the index. [egg-file-stage-current-file], page 22

`C-x v o` Checkout HEAD's version of the current file. [egg-file-checkout-other-version], page 22

`C-u C-x v o`
: Ask for confirmation if the current file contains unstaged changes.

`C-x v u` Checkout INDEX's version of the current file. [egg-file-cancel-modifications], page 22

`C-u C-x v u`
: Then ask for confirmation if the current file contains unstaged changes.

`C-x v v` Guess and perform the next logical action. [egg-next-action], page 22

`C-u C-x v v`
: Ask for confirmation before executing the next-action.

`C-x v /` Search the current file's history for changes introducing or removing a string [egg-search-file-changes], page 24

`C-u C-x v /`
: Search for a regexp instead of a string.

C-u C-x v C-u /
Prompt the user for advanced search mode.

C-x v =	Diff the current file in another window. [egg-file-diff], page 22
C-x v ~	Show other version of the current file in another window. [egg-file-version-other-window], page 22

The following commands launch Egg's special buffers to perform various git related tasks.

C-x v b	Start a new branch from HEAD. [egg-start-new-branch], page 24
C-x v ?	Search the current branch's history for changes introducing/removing a term. [egg-search-changes], page 27
C-x v l	Show the commit DAG of a ref, See [Log], page 14. [egg-log], page 25
C-x v L	Show commit DAG of a ref and its reflogs, See [Log], page 14. [egg-reflog], page 25
C-x v s	Show the status of the current repo, See [Status], page 8. [egg-status], page 22

Status Buffer, manipulating the Index and the Worktree.

This special buffer is one of Egg's two important special buffers (the other is the Log Buffer). It's launched by the [egg-status], page 22. This buffer is designed to manipulate the Index and the WorkTree. As its name indicates, it show the current status of the repository. It's also the place where user goes, to resolve conflicts during merge or rebase. Since its purpose is to manipulate the Index and the WorkTree, the Status Buffer also shows the list of *stashed* WIPs.

Out of the box, the status buffer displays a *lot* of informations. However, you can customize what and how it displays informations. For *what* to show, See [egg-status-buffer-sections], page 21. For *how* to show, See [egg-buffer-hide-sub-blocks-on-start], page 21, and See [egg-buffer-hide-sub-blocks-on-start], page 21.

By default, the status buffer display 5 sections. On the top is the Repo section where it shows the current branch (or current commit if HEAD was detached), the sha1, the repository and optionally the Help section (the Help section can be initially hidden see [egg-buffer-hide-help-on-start], page 21, or simply omitted see [egg-show-key-help-in-buffers], page 21). Following the repo section is the Unstaged section. This section shows a sequence of diffs, describing difference between the index and the work-tree. Next comes the Staged section, this section also contains a sequence of diffs but they describe the difference between HEAD and the index. Another one is the Untracked section where the untracked files in the repository are listed. Finally, the last section is Stash, where stashed WIPs are displayed. To navigate among the sections as well as their subsection, See [Sections], page 5.

The basic *non*-context-sensitive bindings in the Status Buffer (and the Log Buffer) are:

q	Leave (and burry) the special buffer [egg-quit-buffer], page 24
G	Re-initialize the current special buffer. [egg-buffer-cmd-refresh], page 24
g	Refresh the current egg special buffer. [egg-buffer-cmd-refresh], page 24
n	Move to the next section. [egg-buffer-cmd-navigate-next], page 22
C-u n	Move to the next section of the same type.
p	Move to the previous section. [egg-buffer-cmd-navigate-prev], page 22
C-u p	Move to the previous section of the same type.

The Status Buffer also bind the following *non* context-sensitive commands:

c	Open the commit buffer for composing a message. [egg-commit-log-edit], page 25
C-u c	The message will be use to amend the last commit.
C-u C-u c	Just amend the last commit with the old message.
d	Prompt a revision to compare against worktree. [egg-diff-ref], page 24
l	Show the commit DAG of a ref. [egg-log], page 25
o	Prompt a revision to checkout. [egg-status-buffer-checkout-ref], page 24

w	Stash current work-in-progress in the WorkTree and the Index. [egg-status-buffer-stash-wip], page 24
L	Show commit DAG of the current branch (or a ref) and its reflogs. [egg-reflog], page 25
S	Stage all tracked files in the repository. [egg-stage-all-files], page 23
U	Unstage all files in the index. [egg-unstage-all-files], page 24
C-c C-h	Hide all sections in current special egg buffer. [egg-buffer-hide-all], page 22
C-u C-c C-h	Show all sections in current special egg buffer.
X	When in the status buffer, throw away local modifications in the work-tree.
C-u X	When in the status buffer, throw all (staged and unstaged) modifications. [egg-status-buffer-undo-wdir], page 24

When the repo is in mid-rebase (due to conflicts or editings), there are extra commands bound in the Status Buffer:

x	Abort the current rebase session. [egg-buffer-rebase-abort], page 25
u	Skip the current commit and continue the current rebase session. [egg-buffer-selective-rebase-skip], page 25
RET	Continue the current rebase session. [egg-buffer-selective-rebase-continue], page 25

Due to the context-sensitve nature of Egg, position the cursor on the *repo* section of the Status Buffer before issueing the above commands. It's to avoid the context-sensitve bindings of the sections from taking over. For example, if the cursor was in the repo sectionk ***RET*** would be non-context-sensitively bound to [egg-buffer-selective-rebase-continue], page 25. However, if the cursor was on top of a hunk section, ***RET*** would be context-sensitively bound to [egg-hunk-section-cmd-visit-file-other-window], page 23.

To see the context binding of the current cursor's location, type ***kp-enter***, See [Context Menu], page 5.

Unstaged or Unmerged Changes

Normally, the Unstaged Section shows the difference between the Index and the WorkTree. In this case, the diff, if present, would be a regular diff sequence. If the repo is in mid-merge (due to conflict), the diff, if present, would be a combined-diff sequence. Sometimes, there would be empty combined-diff deltas. Those are the files where conflicts were resolved but the files were not yet added to the Index.

Diff

Context bindings for all types of the diff header: all Section commands ([egg-section-map], page 5), plus

RET	Visit file the current file in other window. [egg-diff-section-cmd-visit-file-other-window], page 23

f	Visit file the current file. [egg-diff-section-cmd-visit-file], page 24
=	Ediff src and dest versions of the current file based on the diff at POS. [egg-diff-section-cmd-ediff], page 23

Unstaged Diff

Context bindings for an *unstaged* diff header: all Diff Section commands ([egg-diff-section-map], page 9), plus

u	Checkout the contents of the current file from the Index. [egg-diff-section-cmd-undo], page 22
=	Compare the current file and its staged contents using ediff. [egg-unstaged-section-cmd-ediff], page 23
s	Update the Index with the file. [egg-diff-section-cmd-stage], page 23
DEL	Revert the file and its slot in the index to its contents in HEAD. [egg-diff-section-cmd-revert-to-head], page 24

Unmerged Diff

Context bindings for an *unmerged* diff header: all Unstaged Diff Section commands ([egg-unstaged-diff-section-map], page 10), plus

=	Run ediff3 to resolve merge conflicts in the current file. [egg-unmerged-section-cmd-ediff3], page 23

Hunk

Context bindings for all types of the hunk: all Section commands ([egg-section-map], page 5), plus

RET	Visit the current file in other-window and goto the current line of the hunk. [egg-hunk-section-cmd-visit-file-other-window], page 23
=	Ediff src and dest versions of the current file based on the diff under the cursor. [egg-diff-section-cmd-ediff], page 23
f	Visit the current file and goto the current line of the hunk. [egg-hunk-section-cmd-visit-file], page 24

Unstaged Hunk

Context bindings for an unstaged hunk: all Hunk commands ([egg-hunk-section-map], page 10), plus

u	Remove the file's modification described by the hunk under the cursor. [egg-hunk-section-cmd-undo], page 23
=	Compare the current file and its staged copy using ediff. [egg-unstaged-section-cmd-ediff], page 23
s	Add the hunk under the cursor to the index. [egg-hunk-section-cmd-stage], page 23

Unmerged Hunk

Context bindings for an unmerged hunk: all Unstaged Hunk commands ([egg-unstaged-hunk-section-map], page 10), plus

`=`	Run ediff3 to resolve merge conflicts in the current file. [egg-unmerged-section-cmd-ediff3], page 23

Unmerged Conflict

This is conflict section inside an unmerged hunk. It's marked by git with the strings: `<<<<<<<`, `=======` and `>>>>>>>`. The context bindings for this section included all commands Unstaged Hunk commands ([egg-unstaged-hunk-section-map], page 10), plus

`m`	interactively resolve the conflict under the cursor. [egg-unmerged-conflict-take-side], page 23
`M`	choose one side (*ours* or *theirs*) of the conflict to resolved the whole file. (`git checkout --ours` or`git checkout --theirs`. [egg-unmerged-conflict-checkout-side], page 22

Staged or Merged Changes

The Staged Section shows the difference between HEAD and the Index. It also show unmerged entries: WorkTree's files that have conflicts or those with resolved conflicts but not added into the Index. The delta in the Staged Section are always of the regular diff.

Staged Diff

Context bindings for an *staged* diff header: all Diff Section commands ([egg-diff-section-map], page 9), plus

`=`	Compare the WorkTree's file, its contents in the Index and in HEAD, using 3-way ediff. [egg-staged-section-cmd-ediff3], page 23
`s`	For the file under the cursor, revert its contents in the Index to HEAD's. [egg-diff-section-cmd-unstage], page 23
`DEL`	Revert the file and its slot in the index to its contents in HEAD. [egg-diff-section-cmd-revert-to-head], page 24

The Untracked Section

This simple section lists the untracked and *unignored* files in the repository. Its purpose is to catch files that one forgot to add into the Index. Context bindings for the untracked section: all Section commands ([egg-section-map], page 5), plus

`DEL`	Add an ignore pattern based on the string at point. [egg-ignore-pattern-from-string-at-point], page 23
`s`	add untracked file(s) to the repository [egg-status-buffer-stage-untracked-file], page 23

The Stash Section

This section displays the stashed WIPs. The context bindings for the stash section: all Section command ([egg-section-map], page 5), plus:

SPC	Load the show the details of the stashed WIP under the cursor. [egg-sb-buffer-show-stash], page 27
RET	Apply the stashed WIP under the cursor to WorkTree and Index. [egg-sb-buffer-apply-stash], page 27
`a`	Apply the stashed WIP under the cursor to WorkTree and Index. [egg-sb-buffer-apply-stash], page 27
DEL	Drop the stashed WIP under the cursor. [egg-sb-buffer-drop-stash], page 27
o	Pop and apply the stash under the cursor to WorkTree and Index.. [egg-sb-buffer-pop-stash], page 27

TBD: binding for the stash's contents.

Commit Buffer

This special buffer is for composing a commit log message. It has a heading section decribing information about the next commit:

- the branch to commit into
- the repository
- the commiter
- the gpg key if the commit is to be signed.

Following the heading section is the text area. The user should compose the log message in here. This section use text-mode's keymap with the following extra bindings:

`C-c C-k`	Cancel composing the message. [egg-log-msg-cancel], page 25
`C-c C-s`	Toggle the gpg-signed status of the message. [egg-log-msg-buffer-toggle-signed], page 27
`C-c C-c`	Done editing, proceed with commiting. [egg-log-msg-done], page 25

After the the text-area is the 3 sections: staged, unstaged and untracked. Here, they are called respectively: Changes to Commit, Deferred Changes and Untracked Files. The context bindings are identical to same sections in the Status Buffer, See [Status], page 8.

Log Buffer

[egg-log], page 25, [egg-file-log], page 25 and [egg-reflog], page 25 are three of the different way of launching the Log Buffer. This buffer shows a list of commits, usually forming a DAG. In the regular cycle: edit-add-commit, commits are added to the current branch in a linear fashion. However, with rebase, amending and reset, the branch's head can move in a non-linear manner and previous commits where it used to point at might no longer reachable from the latest commit. That's why git's Reflog is so important, it shows the previous incarnations of a ref. Those previous incarnations are commits that might not be reachable by walking history from the latest commit. Realizing that importance, Egg's Log Buffer was designed to show both the combined DAG and branch (or a ref) and its reflogs.

When launch without any prefix, `egg-log` will show the combined DAG of the current branch (or HEAD) and its reflogs. With *`C-u`* prefix, the buffer will show the combined DAG of all the refs in the repository (without the reflogs). With *`C-u C-u`* prefix, the command will prompt for a ref then show the combined DAG of that ref and its reflogs.

The buffer is composed of the repo section including the help subsection and the DAG section where almost every line describes a commit. The refs that point at the commit are listed on the commit's line. The details of a commit, i.e. the diffs, can be loaded with SPC ([egg-log-buffer-insert-commit], page 26). As with the Status Buffer, the hide/show and the navigation commands work the same way here See [Status], page 8. The Status Buffer was designed primarily to deal with the index, the Log Buffer, on the other hand, was intended for actions that operate on the DAG. Most commands in the Log Buffer are context sensitive. Their behaviour depends on the current location of the cursor. In the DAG section of buffer, almost every line described a commit (Egg calls it a commit line.) Thus, most commands in the buffer act upon the commit described on the line under the cursor. However, if the cursor was on a ref name (a branch, a tag, a remote or a reflog), many commands will operate upon the ref instead of the commit.

There also minor variants of the Log Buffer, examples:

- the File's history where the buffer only displays the DAG's commits modifying the file.
- the Query Buffer, this buffer display the results of a *pickaxe* search. The commits are only those introducing or removing a term where *term* can be a string, a regexp or a line.

Moving HEAD

`o`	Checkout. This key will checkout... something. If the cursor was on a branch-name, then Egg will checkout that branch. HEAD will be symbolic. On the other hand, the cursor was on simple commit line or on a non-branch ref such as a tag or a remote-tracking branch, Egg will detach HEAD and checkout the commit.
`a`	Anchor HEAD (Egg's weird name for `git reset`. If HEAD was attached (symbolic), then Egg will move the current branch to the new commit. If HEAD was detached, HEAD will be moved to the new commit (but still detached).
`u`	push the current branch to the commit under the cursor.

Creating REFs

`t`	Tag the commit under the cursor.
`T`	Create an annotated tag pointing to the commit under the cursor.
`B`	Create a new branch pointing to the commit under the cursor.
`b`	Start (checkout) a new branch poiting to the commit under the cursor.

Merging Changes

`m`	Merge the chain of commits starting at the one under the cursor into HEAD.
`r`	rebase the current branch on to the commit under the cursor.
`c`	pick the *one and only one* commit under the cursor and put it on HEAD.
`M-x egg-log-buffer-do-revert-rev`	undo the changes introduced by *the* commit under the cursor.
`r`	rebase the current branch onto the commit under the cursor.
`R`	rebase the chain of *marked commits* of the current branch onto the commit under the cursor.

Marking Commits for interactive rebase

Before starting an interactive rebase, at least one commits should be marked.

`+`	mark the commit under the cursor to be picked in the *upcoming* interactive rebase.
`~`	mark the commit under the cursor to be edited in the *upcoming* interactive rebase.
`.`	mark the commit under the cursor to be squashed in the *upcoming* interactive rebase.

Using `C-u` prefix with the mark commands, one can reorder the commits in the rebase. In this case, pressing `g` to refresh the Log Buffer will display the reordered to-be-rebased sequence of commits at the top of the buffer, right after the Help section.

Pushing and Fetching

`U`	Uploading (pushing) to a remote site. If the cursor was on a branch and the branch has an defined upstream, Egg will push the branch to the remote site of the upstream branch. If the branch has no defined usptream, Egg will prompt the user for a remote site to push it to. If the cursor was on a remote *site* (i.e. the *site* part in the site/branch name), Egg will prompt for a ref (branch or tag) to push to that remote site.
`D`	Download (fetching) from a remote site. If the cursor was on a remote tracking *branch* (i.e. the branch part of a site/branch name). Egg will attempt to fetch the remote branch from that site. If the cursor was on remote *site* (i.e. the *site* part of the site/branch name), Egg will prompt for a ref to fetch (or all) from that remote site.

Summary

All *log-style* buffers have the following bindings: All buffer bindings describe in [egg-buffer-mode-map], page 8, plus

n	Move cursor to the next ref. [egg-log-buffer-next-ref], page 27
p	Move cursor to the previous ref. [egg-log-buffer-prev-ref], page 27
s	Show the status of the current repo. [egg-status], page 22

The log buffer have the following bindings: All buffer bindings describe in [egg-log-buffer-base-map], page 16, plus

L	Show reflogs for the ref under the cursor [egg-log-buffer-reflog-ref], page 26
/	Search the current branch's history for changes introducing/removing a term. [egg-search-changes], page 27

A commit line, whether in the main Log Buffer or in one of the log-style buffers, has the following context bindings: all section commands as described in [egg-section-map], page 5, plus

SPC	Load and show the details of commit under the cursor. [egg-log-buffer-insert-commit], page 26
B	Create a new branch pointing to commit under the cursor, without checking it out. [egg-log-buffer-create-new-branch], page 26
C-u B	Force the branch creation by deleting the old one with the same name.
b	Create a new branch pointing to commit under the cursor, and make it the new HEAD. [egg-log-buffer-start-new-branch], page 26
C-u b	Force the creation by deleting the old branch with the same name.
o	Checkout ref or commit under the cursor. [egg-log-buffer-checkout-commit], page 25
C-u o	Force the checkout even if the index was different
t	Tag commit under the cursor. [egg-log-buffer-tag-commit], page 26
C-u t	Force the creation of the tag
T	Start composing the message to create an annotated tag on commit under the cursor. [egg-log-buffer-atag-commit], page 26
C-u T	The annotated tag will be gpg-signed.
a	Move the current branch or the detached HEAD to commit under the cursor. [egg-log-buffer-anchor-head], page 25
C-u a	HEAD will be moved, index will be reset and the work tree updated.
C-u C-u a	Prompt for the git reset mode to perform.
m	Merge to HEAD the path starting from commit under the cursor. [egg-log-buffer-merge], page 26

C-u m	Do not auto commit the merge result.
C-u C-u m	Prompt the user for the type of merge to perform.
r	Rebase HEAD using commit under the cursor as upstream. [egg-log-buffer-rebase], page 26
c	Pick commit under the cursor and put on HEAD. [egg-log-buffer-pick-1cherry], page 26
C-u c	Will not auto-commit but let the user re-compose the message.
i	Show information about the ref or the mark under the cursor. [egg-log-show-ref], page 27

In the main Log Buffer, a commit line has the additional context bindings:

R	Start an interactive session to rebase the marked commits onto commit under the cursor. [egg-log-buffer-rebase-interactive], page 26
+	Mark commit under the cursor to be picked in the upcoming interactive rebase. [egg-log-buffer-mark-pick], page 28
C-u +	Prompt for reordering.
.	Mark commit under the cursor to be squashed in the upcoming interactive rebase. [egg-log-buffer-mark-squash], page 28
C-u .	Prompt for reordering.
~	Mark commit under the cursor to be edited in the upcoming interactive rebase. [egg-log-buffer-mark-edit], page 28
C-u ~	Prompt for reordering.
DEL	Unmark commit under the cursor. [egg-log-buffer-unmark], page 28
C-u DEL	Unmark all.
***	Mark commit under the cursor as the BASE commit. [egg-log-buffer-mark], page 27
=	Compare HEAD against commit under the cursor. [egg-log-buffer-diff-revs], page 26
C-u =	Prompt for a string and restrict to diffs introducing/removing it.
C-u C-u =	Prompt for a regexp and restrict to diffs introducing/removing it.
C-u C-u C-u =	Prompt for a pickaxe mode.
u	Push commit under the cursor onto HEAD. [egg-log-buffer-push-to-local], page 26
C-u u	Instead of HEAD, prompt for another ref as destination.
C-u C-u u	Will force the push evel if it would be non-ff.

A ref has all the context bindings of the commit line ([egg-log-commit-map], page 17), plus

L	Show reflogs for the ref under the cursor [egg-log-buffer-reflog-ref], page 26
x	Remove the ref under the cursor. [egg-log-buffer-rm-ref], page 26
U	Upload the ref under the cursor to a remote repository. [egg-log-buffer-push-to-remote], page 27
d	Push HEAD to the ref under the cursor. [egg-log-buffer-push-head-to-local], page 26

A local branch has all context bindings of a local ref ([egg-log-local-ref-map], page 18), plus

C-c C-=	Compare the branch under the cursor against its upstream [egg-log-buffer-diff-upstream], page 28
C-u C-c C-=	Prompt for a string and restrict to diffs introducing/removing it.
C-u C-u C-c C-=	Prompt for a regexp and restrict to diffs introducing/removing it.
C-u C-u C-u C-c C-=	Prompt for a pickaxe mode.

A remote tracking branch has all the context bindings of the commit line ([egg-log-commit-map], page 17), plus

L	Show reflogs for the remote branch under the cursor [egg-log-buffer-reflog-ref], page 26
x	Remove the remote branch under the cursor. Egg will ask if it should attempt to delete the branch on the remote site as well. [egg-log-buffer-rm-ref], page 26
D	Download and update the remote tracking branch under the cursor. [egg-log-buffer-fetch-remote-ref], page 27

A remote site (it's the *site* part in a site/branch ref name) has all the context bindings of the commit line ([egg-log-commit-map], page 17), plus

D	Prompt for the refs to fetch the remote site under the cursor. [egg-log-buffer-fetch], page 28
U	Prompt for the ref to push to the remote site under the cursor. [egg-log-buffer-push], page 28

Blame Mode

Pointing Finger

`C-x v a` ([egg-file-toggle-blame-mode], page 19), will bring the current buffer in-or-out of annotation-mode, or more accurately, *blame mode*. In blame-mode, editing is disabled (the buffer is in read-only mode). Block of lines are prefix with a header describing the *last* commit that modified those lines. The command `egg-blame-locate-commit` pick up that commit, open the Log Buffer and locate that commit in the DAG.

The context bindings in blame mode are:

	[egg-blame-locate-commit], page 28
RET	Jump to a commit in the branch history from an annotated blame section. [egg-blame-locate-commit], page 28
q	Toggle blame mode for the current-file. [egg-file-toggle-blame-mode], page 19
n	Move to the next annotation. [egg-buffer-cmd-navigate-next], page 22
p	Move to the previous annotation.

Resolving Conflicts

Conflicts resolution is done in the STATUS BUFFER. First put the cursor on top of the conflict area, ie. the text between the `<<<<<<<` and `>>>>>>>`. Then there 3 ways of performing a conflict resolution

THE GIT WAY

Press RET, and git will open the file and locate the conflict. Edit the conflict and remove the `<<<<<<<`, `=======` and `>>>>>>>` markers.

THE SIMPLE (EGG) WAY

Press *`m`* on the conflict. Egg will ask the user to choose either, *ours*'s changes or *theirs*'s changes. The first choice depends on the location of the cursor. If the *`m`* was press on *ours*'s side (i.e. between `<<<<<<<` and `=======`, the *ours* will be the first choice. Otherwise, *theirs*'s changes will be the first choice.

THE HEAVY (EDIFF) WAY

Press `=` on the conflict. Egg will launch `ediff3` to help resolving the conflict. If it was a merge operation, the buffers (from left-to-right or top-to-bottom, depending on the value of `ediff-split-window-function`) are:

- theirs (the branch being merged in)
- ours (the current branch)
- WorkTree's file (with conflicts)

If it was a rebase operation, the buffers are:

- ours (the upstream or *onto* branch)
- theirs (the branch being rebased)
- WorkTree's file (with conflicts)

After resolving the conflict, add the file into the Index by pressing *`s`* on the diff header in the STATUS BUFFER's Unmerged Section.

Customisations

`egg-buffer-hide-sub-blocks-on-start` [User Option]
: For each type of egg special buffers, should its sub-blocks be initially hidden.

`egg-buffer-hide-section-type-on-start` [User Option]
: For each type of egg special buffers that show sequence of diffs, control the types of sub-block to be initially hidden.

`egg-buffer-hide-help-on-start` [User Option]
: For each type of egg special buffers has a help section, should the help section be initially hidden.

`egg-status-buffer-sections` [User Option]
: Select the sections that will be shown in the status buffer.

`egg-show-key-help-in-buffers` [User Option]
: For each type of egg special buffers, should the Help section be displayed.

Commands

`egg-file-stage-current-file` [Command]
: Add the contents of the current file into the index.

`egg-file-diff` [Command]
: Compare the file's current contents vs its contents in the index. With prefix, prompt for a revision to compare (instead of using the index) with the file's current contents.

`egg-file-ediff` [Command]
: Compare the file's current contents vs its contents in the index. With prefix, prompt for a revision to compare (instead of using the index) with the file's current contents.

`egg-file-version-other-window` [Command]
: Show the contents of the current file from the index. With prefix, prompt for a git revision and show the file's contents from that revision.

`egg-file-checkout-other-version` [Command]
: Replace the file's current contents with its contents from a revision.

`egg-file-cancel-modifications` [Command]
: Revert the file to its contents in the index.

`egg-next-action` [Command]
: Perform the *next* action, whatever that is!

`egg-status` [Command]
: Show Egg Status buffer in another window but do not select it. With prefix, select the status buffer, See [Status], page 8.

`egg-buffer-hide-all` [Command]
: Hide all sections in buffer. With prefix, show all sections.

`egg-buffer-cmd-navigate-next` [Command]
: Move cursor to the next section

`egg-buffer-cmd-navigate-prev` [Command]
: Move cursor to the previous section

`egg-section-cmd-toggle-hide-show` [Command]
: Toggle the visibility of the section under the cursor.

`egg-diff-section-cmd-undo` [Command]
: For the file of diff header under the cursor, remove its differences vs the source revision. Usually, this command revert the file to its staged state in the index. However, in a diff special egg buffer, it can change the file's contents to the one of the source revision.

`egg-unmerged-conflict-checkout-side` [Command]
: Checkout one side of the conflict under the cursor.

`egg-unmerged-conflict-take-side` [Command]
: Interactive resolve conflict under the cursor.

`egg-hunk-section-cmd-undo` [Command]
: Remove the file's modification described by the hunk under the cursor.

`egg-stage-all-files` [Command]
: Stage all tracked files in the repository.

`egg-hunk-section-cmd-stage` [Command]
: Add the hunk under the cursor to the index.

`egg-diff-section-cmd-stage` [Command]
: Update the index with the file in the diff header under the cursor. If the file was deleted in the workdir then remove it from the index.

`egg-staged-section-cmd-ediff3` [Command]
: Compare the staged copy of FILE and the version in HEAD using ediff.

`egg-unmerged-section-cmd-ediff3` [Command]
: Run ediff3 to resolve merge conflicts in file of the combined diff section under the cursor.

`egg-unstaged-section-cmd-ediff` [Command]
: For the file in the diff section under the cursor. Compare it with its staged copy using ediff.

`egg-diff-section-cmd-ediff` [Command]
: Run ediff on src and dest versions of the file in the diff section under the cursor.

`egg-hunk-section-cmd-visit-file-other-window` [Command]
: Visit the file in the diff section under the cursor, in the other window and goto the current line of the hunk.

`egg-diff-section-cmd-visit-file-other-window` [Command]
: Visit the file in the diff section under the cursor, in the other window.

`egg-ignore-pattern-from-string-at-point` [Command]
: Add an ignore pattern based on the string at point.

`egg-find-file-at-point` [Command]
: alias for `find-file-at-point`

`egg-status-buffer-stage-untracked-file` [Command]
: Add untracked file(s) to the repository. Acts on a single file or on a region which contains the names of untracked files. With prefix, only create the index entries without adding the contents.

`egg-diff-section-cmd-unstage` [Command]
: For the file in the diff header under the cursor, revert its stage in the index to original. If the file was a newly created file, it will be removed from the index. If the file was added after a merge resolution, it will reverted back to conflicted state. Otherwise, its stage will be reset to HEAD.

`egg-hunk-section-cmd-unstage` [Command]
: Remove the hunk under the cursor from the index.

`egg-diff-section-cmd-revert-to-head` [Command]
: Revert the file in the diff section under the cursor and its slot in the index to its state in HEAD.

`egg-diff-section-cmd-visit-file` [Command]
: Visit FILE.

`egg-hunk-section-cmd-visit-file` [Command]
: Visit FILE and goto the current line of the hunk.

`egg-section-cmd-toggle-hide-show-children` [Command]
: Toggle the visibility of the subsections of the section under the cursor.

`egg-start-new-branch` [Command]
: start a new branch from HEAD, keeping local modifications.

`egg-search-file-changes` [Command]
: search in the DAG for commits that introduced or removed the *search term*.

`egg-grep` [Command]
: run grep on files tracked by git.

`egg-status-buffer-stash-wip` [Command]
: Prompt for a description and stash current work-in-progress in workdir and the index. With prefix, the command will also stash untracked/unignored files.

`egg-diff-ref` [Command]
: Prompt a revision compare against worktree.

`egg-quit-buffer` [Command]
: Leave (and bury) an egg special buffer

`egg-status-buffer-checkout-ref` [Command]
: Prompt a revision to checkout.

`egg-buffer-cmd-refresh` [Command]
: Refresh the current egg special buffer.

`egg-log-buffer-style-command` [Command]
: Re-run the command that create the buffer.

`egg-status-buffer-undo-wdir` [Command]
: When in the status buffer, throw away local modifications in the work-tree. With prefix, reset the work-tree to its state in HEAD. Otherwise, reset the work-tree to its staged state in the index.

`egg-unstage-all-files` [Command]
: Unstage all files in the index.

`egg-commit-log-edit` [Command]

Open the commit buffer for composing a message. With *C-u* prefix, re-compose the message of the last commit. With *C-u C-u* prefix, avoid all composition, amend the last commit by reusing its commit message.

`egg-log-msg-done` [Command]

Take the appropriate action with the composed message.

`egg-log-msg-newer-text` [Command]

Cycle forward through comment history.

`egg-log-msg-older-text` [Command]

Cycle backward through comment history.

`egg-log-msg-cancel` [Command]

Cancel the current message editing.

`egg-buffer-selective-rebase-continue` [Command]

Continue the current rebase session. The mode, sync or async, will depend on the nature of the current rebase session.

`egg-buffer-selective-rebase-skip` [Command]

Skip the current commit and continue the current rebase session. The mode, sync or async, will depend on the nature of the current rebase session.

`egg-buffer-rebase-abort` [Command]

Abort the current rebase.

`egg-log` [Command]

Show the commit DAG of REF-NAME.

`egg-reflog` [Command]

Show commit DAG of BRANCH and its reflogs. This is just an alternative way to launch 'egg-log'

`egg-file-log` [Command]

Show the commits in the current branch's DAG that modified the current file. With prefix,do not restrict the commits to the current branch's DAG.

`egg-log-buffer-anchor-head` [Command]

Move the current branch or the detached HEAD to the commit under the cursor. The index will be reset and files will in worktree updated. If a file that is different between the original commit and the new commit, the git command will abort. This is basically git reset –keep. With C-u prefix, HEAD will be moved, index will be reset and the work tree updated by throwing away all local modifications (this is basically git reset –hard). With C-u C-u prefix, the command will prompt for the git reset mode to perform.

`egg-log-buffer-checkout-commit` [Command]

Checkout the commit at POS. With prefix, force the checkout even if the index was different from the new commit.

`egg-log-buffer-start-new-branch` [Command]
: Create a new branch, and make it a new HEAD

`egg-log-buffer-create-new-branch` [Command]
: Create a new branch, without checking it out.

`egg-log-buffer-push-to-local` [Command]
: Push a ref or a commit under the cursor onto HEAD. With C-u, instead of HEAD, prompt for another ref as destination. With C-u C-u, will force the push even if it would be non-ff. When the destination of the push is HEAD, the underlying git command would be a pull (by default –ff-only).

`egg-log-buffer-insert-commit` [Command]
: Load and show the details of the commit under the cursor.

`egg-log-buffer-tag-commit` [Command]
: Tag the commit under the cursor. With prefix, force the creation of the tag even if it replace an existing one with the same name.

`egg-log-buffer-atag-commit` [Command]
: Start composing the message for an annotated tag on the commit at the cursor. With prefix, the tag will be gpg-signed.

`egg-log-buffer-merge` [Command]
: Merge to HEAD the path starting from the commit under the cursor. With C-u prefix, do not auto commit the merge result. With C-u C-u prefix, prompt the user for the type of merge to perform.

`egg-log-buffer-pick-1cherry` [Command]
: Pick one cherry under the cursor and put on HEAD. With prefix, will not auto-commit but let the user re-compose the message.

`egg-log-buffer-rebase` [Command]
: Rebase HEAD using the commit under the cursor as upstream. If there was a commit marked as BASE, then rebase HEAD onto the commit under the cursor using the BASE commit as upstream.

`egg-log-buffer-rebase-interactive` [Command]
: Start an interactive rebase session using the marked commits. The commit under the cursor is the where the chain of marked commits will rebased onto.

`egg-log-buffer-diff-revs` [Command]
: Compare HEAD against the rev under the cursor.

`egg-log-buffer-rm-ref` [Command]
: Remove the ref under the cursor.

`egg-log-buffer-reflog-ref` [Command]
: Show reflogs for the ref under the cursor.

`egg-log-buffer-push-head-to-local` [Command]
: WTF.

`egg-log-buffer-fetch-remote-ref` [Command]
: Download and update the remote tracking branch at under the cursor.

`egg-search-changes` [Command]
: Search history from HEAD for changes introducing/removing a term.If a BASE ref was marked, then restrict the search to BASE..HEAD.

`egg-search-changes-all` [Command]
: Search entire history for changes introducing/removing a term.

`egg-log-locate-commit` [Command]
: Relocate the commit under the cursor back to the full history in the log buffer.

`egg-log-buffer-push-to-remote` [Command]
: Upload the ref under the cursor to a remote repository. If the ref track a remote tracking branch, then the repo to upload to is the repo of the remote tracking branch. Otherwise, prompt for a remote repo.

`egg-search-file-changes-all` [Command]
: Search file's full history for changes introducing/removing a term.

egg-find-tracked-file

`egg-find-tracked-file` [Command]
: Open a file tracked by git.

`egg-sb-buffer-show-stash` [Command]
: Load the details of the stash under the cursor.

`egg-sb-buffer-apply-stash` [Command]
: Apply the stash under the cursor.

`egg-sb-buffer-drop-stash` [Command]
: Drop the stash under the cursor.

`egg-sb-buffer-pop-stash` [Command]
: Pop and apply the stash under the cursor.

`egg-log-msg-buffer-toggle-signed` [Command]
: Toggle the to-be-gpg-signed state of the message being composed.

`egg-log-buffer-next-ref` [Command]
: Move cursor to the next ref.

`egg-log-buffer-prev-ref` [Command]
: Move cursor to the previous ref.

`egg-log-show-ref` [Command]
: Show information about the ref or the mark under the cursor.

`egg-log-buffer-mark` [Command]
: Mark commit under the cursor as the BASE commit.

`egg-log-buffer-unmark` [Command]
: Unmark commit under the cursor. With C-u prefix, unmark all.

`egg-log-buffer-mark-edit` [Command]
: Mark commit under the cursor to be edited in the upcoming interactive rebase. With C-u prefix, prompt for reordering.

`egg-log-buffer-mark-squash` [Command]
: Mark commit under the cursor to be squashed in the upcoming interactive rebase. With C-u prefix, prompt for reordering.

`egg-log-buffer-mark-pick` [Command]
: Mark commit under the cursor to be picked in the upcoming interactive rebase. With C-u prefix, prompt for reordering.

`egg-log-buffer-diff-upstream` [Command]
: Compare commit under the cursor against its upstream.

`egg-log-buffer-fetch` [Command]
: Fetch some refs from remote at POS.

`egg-log-buffer-push` [Command]
: Push some refs to the remote at POS

`egg-blame-locate-commit` [Command]
: Jump to a commit in the branch history from an annotated blame section. With C-u prefix, the history of all refs is used.