↖ Topic Index

VCS — Mercurial

Operation	Keystroke	Function	Note
Mercurial in Emacs • **Mercurial Home** Last updated on:	Emacs **VC interface** provides built-in support for many Version Control Systems including Mercurial. It's sufficient for most operations. The following external packages provide more support: 📦 **hgignore-mode** font lock **.hgignore** files. PEL activates it when the **pel-use-hgignore** user-option is turned on (set to **t**). 📦 **Monky** provides a Magit-like interface to hg commands PEL activates it when the **pel-use-monky** user-option is turned on (set to t). • See also: **Mercurial hg(1) manual** • Related to VCS: Σ VCS-Git ⚧Magit , Σ VCS-Subversion 2025-10-29		
Open this PDF file. See also: Σ **Help/Info**	`<f11> v <f1>` `<f12> <f1>`	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Prompt to open one of the VCS PDF files like the Σ **VCS-Mercurial** local PDF. If the prefix argument (like **`C-u`** or **`M--`**) is used, then it opens the remote GitHub hosted raw PDF instead. If the **pel-flip-help-pdf-arg** user-option is set it's the other way around.
Σ **Customize PEL VCS control**	`<f11> v <f2>` `<f12> <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL Version Control System support. • If OTHER-WINDOW is non-nil (use **`C-u`**), display in other window.
Σ **Customize Emacs VCS control**	`<f11> v <f3>`	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs Version Control System support: vc, vc-hg, vc-git, magit, monky.
	`<f12> <f3>`		Customize Emacs Version Control System support: vc, vc-hg, vc-git.
Emacs Built-in VCS package: VC • **Forcing selection of a VCS backend for VC commands** • Select VCS backend with **directory local variable** • See also: Σ **File/Directory Variables**	• The VC library supports several **VCS tools** identified by the **vc-handled-backends user-option**, which, by default, identifies: **RCS** and **SCCS**, **CVS** and **Subversion**, **Bazaar**, **Git**, **Mercurial** and **Monotone**, in that order. • When visiting a directory or a file and using a VC command to manage the repository, VC automatically detects the (D)VCS type for the repository, set the variable ***vc-dir-backend*** and adjusts its backend accordingly. If the directory has a **.hg** directory or is enclosed in a directory tree where the root has a **.hg** directory, then VC uses the Mercurial backend and the commands behaves as described in this table. If a directory tree is managed by more than one VCS tool, the tool selected by the VC commands may not be the one you want. • For example if a directory tree is managed both by Git (there is a .git directory in the parent directories) and by Mercurial (and there is a .hg directory in the parent directories) then VC will select Git because its code looks for Git before it looks for Mercurial. You can **force the selection** of a specific VCS backend by using one of the following methods: 1. Select a different VCS when issuing the vc-dir command by typing the **`C-u`** prefix before the command keystroke. 2. Select a different VCS back-end one file at a time, using the **pel-vcs-switch-backend** command. It searches for the VCS repository directories in the directory parents, then prompt for the one to use and switches to it using the vc-switch-backend command. The vc-switch-backend command can be invoked directory but it does not seem to work when trying to to switch backend in a multi-backend directory environment. 3. Avoid using VC. Instead use a VCS-specific tool. Like **Magit** for Git and **Monky** for Mercurial. 4. Force VCS backend for the directory using a directory local variable: Write the following in a .dir-locals.el file if the file is empty: `((vc-dir-mode . ((vc-dir-backend . Hg))))` If the file already contains a list, insert the following entry in the list: `(vc-dir-mode . ((vc-dir-backend . Hg)))` You might also have to add the safe values of *vc-dir-backend* to the *safe-local-variable*-values, by adding the following statement inside the **custom-set-variables** arguments in your Emacs custom file.: `'(safe-local-variable-values (quote ((vc-dir-backend . Git) (vc-dir-backend . Hg))))`		
Detect available VCS backend and switch to one selected	`<f11> v s`	(**pel-vcs-switch-backend**)	Switch VCS back-end for the current file. • If no VCS back-end or only one VCS back-end is available for the file, the command issues an error, otherwise it prompts for the VCS back-end to use and switch the VCS back-end for this file. The switch is temporary: it is restricted to the current Emacs • session. • Use this command when the file is managed by more than one VCS back-end. • This uses the function 'vc-switch-backend' to perform the switch.
Switch VCS backend	`C-x v b`	(**vc-switch-backend** FILE BACKEND)	Make BACKEND the current version control system for FILE. • FILE must already be registered in BACKEND. The change is not permanent, only for the current session. This function only changes VC's perspective on FILE, it does not register or unregister it. • By default, this command cycles through the registered backends. • To get a prompt, use a prefix argument. ⚠️This command does not seem to be working if a directory is both managed by Mercurial and Git. Use **pel-vcs-switch-backend** instead, it does work.
Open the VCS Status Window	With VC, you manage the repository through a buffer using the **vc-dir** command opens a *vc-dir* buffer that uses the vc-dir-mode major mode. • The buffer shows the status of files unregistered, modified and not yet committed. • The buffer supports a set of commands, mostly single character commands to quickly perform operations on the repository. ☝ These keys are shown in class with light blue background beside globally bound commands that can be issued from any buffer. • For Mercurial this runs the hg status command and displays the list of files that require attention.		
Open a VC Status buffer **List status of files in *vc dir* buffer** **hg status**	• `C-x v d` • `<f11> v v`	(**vc-dir** DIR &optional BACKEND)	Executes: **hg status.** Open a *vc dir* buffer that shows the hg status of files. ☝Type **C-u** prefix to request a different VC backend: the command prompts for the backend allowing you to select a different backend than the automatically selected one.
		Show the VC status for "interesting" files in and below DIR. • This allows you to mark files and perform VC operations on them. • The list omits files which are up to date, with no changes in your copy or the repository, if there is nothing in particular to say about them. • Preparing the list of file status takes time; when the buffer first appears, it has only the first few lines of summary information. The file lines appear later.	
Control .hgignore	Identify files that must not be managed by Mercurial with the **.hgignore file**.		
Ignore a file (update the .hgignore file)	`C-x v G`	(**vc-ignore** FILE &optional DIRECTORY REMOVE)	Update the depot's **.hgignore** file to ignore a specific file or files. • Ignore FILE under the VCS of DIRECTORY. • Normally, FILE is a wildcard specification that matches the files to be ignored. When REMOVE is non-nil, remove FILE from the list of ignored files. • DIRECTORY defaults to 'default-directory' and is used to determine the responsible VC backend. • When called interactively, prompt for a FILE to ignore, unless a prefix argument is given, in which case prompt for a file FILE to remove from the list of ignored files.
	`G`	(**vc-dir-ignore**)	Emacs standard binding: Ignore the file on the **current** line; update the .hgignore file.
Ignore all marked files	`!`	(**pel-vc-ignore-marked**)	Ignore all marked files: update the .hgignore file.
Describe VC commands	`?`	(**describe-mode** &optional BUFFER)	Shows a list of the the key bindings available in the *vc-dir* buffer.
Hiding vc-dir lines	You can hide files that are in specific state (like unregistered) by using the following command.		
Hide specific states	`H`	(**pel-vc-dir-hide** STATES)	Hide lines corresponding to specified STATES. Prompt for state with tab completion. • Press **g** to refresh list and display all states again.
Log Hg commands	Log the VC backend command into the *pel-vc-log* buffer. The events show the event as lisp lists.		
Toggle logging VC backend commands	`<f11> v l`	(**pel-vcs-toggle-vc-log**)	Start/stop logging VC commands in the *pel-vc-log* buffer. • When starting, does not create the buffer. It is created on the first VC event.

Operation	Keystroke	Function	Note
VC commands for Mercurial • hg manual	Mercurial is well supported by Emacs VC mode as long as Mercurial is installed and the Emacs process has access to the Mercurial hg command. No special setup is required, VC will automatically detect that the file is inside a Mercurial repository as described above. When editing a file that is inside a Mercurial repository, you can use the following VC commands. Since the VC command key bindings is the same for all VCS backends, some of the commands do not apply to Mercurial and are not listed below. ☝ The commands are listed in alphabetical order of the hg command names.		
• **hg commands for** • hg add • hg commit	In Mercurial files must be identified for inclusion in the repository with the **hg add** command. • Changes in a file can only be committed with the **hg commit** once they have been added. • Inside the *vc-dir* buffer, to add file first select them (with the **m** key) and then the **v** key to add them. • The command will be rejected if files currently managed are selected. Unselect them with the **u** or **U** key first then try again. • Once a file is managed by Mercurial, you can select its name and commit it by using the **v** key again: it issues the **hg commit** command then.		
Add the file to the repo	`C-x v i`	(**vc-register** &optional VC-FILESET COMMENT)	Register into a version control system: perform a **hg add** for the file.
Add/Commit current file	`C-x v v`	(**vc-next-action** VERBOSE)	Executes, for the current file (or all marked files in the *vc-dir* buffer): • **hg add** of the files if the files are not part of the repository yet. • **hg commit**, of all marked files, otherwise. • Refuses to perform the action when state of *vc-dir* selected items differ.
Add/Commit selected file(s)	`v`		
• **hg annotate command**	List changes in files, showing the revision id responsible for each line. • This command is useful for discovering when a change was made and by whom.		
Annotate version of each line of filem **hg annotate**	`C-x v g`	(vc-annotate FILE REV &optional DISPLAY-MODE BUF MOVE-POINT-TO VC-BK)	Annotate the lines of the current file: open a ***Annotate file (rev #)**** buffer that shows the annotation of each line of the current file, each changes has its own background colour. • More commands are available in the *Annotate* buffer.
• **hg cat command**	Print the specified files as they were at the given revision.		
Show the content of a specific revision of a file **hg cat**	`C-x v ~`	(**vc-revision-other-window** REV)	Visit revision REV of the current file in another window. • If the current file is named 'F', the revision is named 'F.~REV~'. • If 'F.~REV~' already exists, use it instead of checking it out again.
• **hg diff command**	Show differences between revisions for the specified files.		
Diff versions of all files in directory: **hg diff**	`C-x v D`	(**vc-root-diff** HISTORIC &optional NOT-URGENT)	Executes **hg diff** command to list differences between all files in the local directory and in repo. • Display diffs between VC-controlled whole tree revisions. • Normally, this compares the tree corresponding to the current fileset with the working revision. • With a prefix argument HISTORIC, prompt for two revision designators specifying which revisions to compare. • The optional argument NOT-URGENT non-nil means it is ok to say no to saving the buffer.
Diff versions of current file: **hg diff**	`C-x v =` `=`	(**vc-diff** &optional HISTORIC NOT-URGENT)	Executes: **hg diff** to list differences between current file version in repo and local file's content. • Display diffs between file revisions: • Normally this compares the currently selected fileset with their working revisions. • With a prefix argument HISTORIC (use `C-u`), it reads two revision designators specifying which revisions to compare. • In Elisp code: the optional argument NOT-URGENT non-nil means it is ok to say no to saving the buffer.
Diff currently selected file(s) **hg diff**	`d`	(**log-view-diff** BEG END)	Show the file(s) difference(s) for the log entry inside a *vc-diff* buffer. • Get the diff between two revisions. • If the region is inactive or the mark is on the revision at point, get the diff between the revision at point and its previous revision. Otherwise, get the diff between the revisions where the region starts and ends. • Unlike 'log-view-diff-changeset', this function only shows the part of the changeset which affected the currently considered file(s).
	`D`	(**log-view-diff-changeset** BEG END)	
• **hg incoming command**	Show new changesets found in the specified path/URL or the default pull location. • These are the changesets that would have been pulled by hg pull at the time you issued this command.		
List files incoming from parent (or specified) repo **hg incoming**	`C-x v I` `I`	(**vc-log-incoming** &optional REMOTE-LOCATION)	Executes: **hg incoming** to list files incoming from parent (or specified) repo. • Show a log of changes that will be received with a pull operation from REMOTE-LOCATION. • When called interactively with a prefix argument, prompt for REMOTE-LOCATION.
• **hg log command**	Print the revision history of the specified files or the entire project. • Open a ***vc-change-log*** window buffer, using the vc-hg-log-view-mode major mode which provides other commands. See section below.		
Print repo log **hg log**	`C-x v L`	(**vc-print-root-log** &optional LIMIT)	Executes: **hg log** to list the repo log in a ***vc-change-log*** buffer. • List the change log for the current VC controlled tree in a window. • If LIMIT is non-nil, it should be a number specifying the maximum number of revisions to show; the default is 'vc-log-show-limit'. • When called interactively with a prefix argument, prompt for LIMIT.
Open the change log for the file **hg log**	`C-x v l` `l`	(**vc-print-log** &optional WORKING-REVISION LIMIT)	Executes: **hg log.** Open the change log for the file in the ***vc-change-log*** buffer. • List the change log of the current fileset in a window.
• **hg merge command** • Merge Tools • hgrc merge	Merge another revision into working directory.		
Merge **hg merge**	`C-x v m`	(**vc-merge**)	Executes: **hg merge**. Perform a version control merge operation. • You must be visiting a version controlled file, or in a 'vc-dir' buffer. • This runs a "merge" operation to incorporate changes from another branch onto the current branch, prompting for an argument list.
• **hg outgoing command**	Show changesets not found in the destination.		
Show change sets not found in the destination **hg outgoing**	`C-x v O` `O`	(**vc-log-outgoing** &optional REMOTE-LOCATION)	Executes: **hg outgoing** to list deltas to push to parent repo. • Show a log of changes that will be sent with a push operation to REMOTE- LOCATION. • When called interactively with a prefix argument, prompt for REMOTE-LOCATION.
• **hg pull command**	Pull changes from a remote repository to a local one.		
Pull files from parent repo: **hg pull**	`C-x v +`	(**vc-update** &optional ARG)	Executes: **hg pull** to pull files from parent repository. Update the current fileset or branch. • You must be visiting a version controlled file, or in a 'vc-dir' buffer. • On a distributed version control system, this runs a "pull" operation to update the current branch, prompting for an argument list if required. • Optional prefix ARG forces a prompt for the VCS command to run, allowing the addition of command line options.
• **hg push command**	Push changesets from the local repository to the specified destination.		
Push changes to the specified destination **hg push**	`C-x v P` `P`	(**vc-push** &optional ARG)	Executes: **hg push** to push committed change sets to the parent repo: Push the current branch. • You must be visiting a version controlled file, or in a 'vc-dir' buffer. • On a distributed version control system, this runs a "push" operation on the current branch, prompting for the precise command if required. Optional prefix ARG non-nil forces a prompt for the VCS command to run.
• **hg remove command**	Schedule the indicated files for removal from the current branch.		
Delete a file **hg remove**	`C-x v x`	(**vc-delete-file** FILE)	Executes: **hg remove** Delete file and mark it as such in the version control system. Prompts for the file name, using the directory of the file visited in current buffer.
• **hg rename command**	Rename files; equivalent of copy + remove: Mark dest as copies of sources; mark sources for deletion. • If dest is a directory, copies are put in that directory. If dest is a file, there can only be one source.		
Rename a file **hg rename**	`M-x vc-rename-file`	(**vc-rename-file** OLD NEW)	Rename file OLD to NEW in both work area and repository. • Interactively: read OLD and NEW, defaulting OLD to the current buffer's file name if it's under version control.
• **hg revert command**	Restore files to their checkout state (the last committed version by default).		

Operation	Keystroke	Function	Note
Revert file(s) **hg revert**	`C-x v u`	(**vc-revert**)	Revert working copies of the selected fileset to their repository contents. • This asks for confirmation if the buffer contents are not identical to the working revision (except for keyword expansion).
• **hg tag command**	Name a particular revision.		
Create a tag **hg tag**	`C-x v s` `B c`	(**vc-create-tag** DIR NAME BRANCHP)	Executes: **hg tag**. Descending recursively from DIR, make a tag called NAME. • For each registered file, the working revision becomes part of the named configuration. If the prefix argument BRANCHP is given, the tag is made as a new branch and the files are checked out in that new branch.
Retrieve a tagged version	`C-x v r` `B s`	(**vc-retrieve-tag** DIR NAME)	For each file in or below DIR, retrieve their tagged version NAME. • NAME can name a branch: this command will switch to the named branch in the directory DIR. • Interactively, prompt for DIR only for VCS that works at file level; otherwise use the repository root of the current buffer. • If NAME is empty, it refers to the latest revisions of the current branch. • If locking is used for the files in DIR, then there must not be any locked files at or below DIR (but if NAME is empty, locked files are allowed and simply skipped). • Tab completion on tag is available at the prompt. ⚠ Tags show in tab completion do not include the tags created with hg tag. Instead they only show the changeset short ID number. However, it accepts names of tags previously created with hg tag. • You can list the tags with the command **hg tags** on the command line.
Show the change log	`B l`	(**vc-print-branch-log** BRANCH)	Show the change log for BRANCH in a *vc-change-log* window. • Prompt for the branch name. Tab completion shows the list of changeset short ID numbers. It does not show them but you can also use the names of tags previously created with hg tag. • You can list the tags with the command **hg tags** on the command line.
***vc-dir* buffer**	The **vc-dir** command opens a *vc-dir* buffer to show the status of files unregistered, modified and not yet committed. The buffer supports a set of commands, mostly single character commands to quickly perform operations on the repository. • Some are shown in the mode-specific key bindings show above. • Some commands are not tied to hg commands. They are shown below.		
Show help for the mode	• `?` • `h`	(**describe-mode** &optional BUFFER)	Opens the *Help* buffer with information about the mode, listing the various key bindings and commands available
Refresh buffer	`g`	(**revert-buffer** &optional IGNORE-AUTO NOCONFIRM PRESERVE-MODES)	Restore buffer to the default list view.
Mark file	`m`	(**vc-dir-mark**)	Mark the current file or all files in the region. • If the region is active, mark all the files in the region. Otherwise mark the file on the current line and move to the next line. • Operations from the buffer apply to all marked files, like adding or committing files, or other operations. 💡Useful to commit several files in the same changeset.
Unmark file	`u`	(**vc-dir-unmark**)	Unmark the current file or all files in the region. • If the region is active, unmark all the files in the region. Otherwise unmark the file on the current line and move to the next line.
Unmark all files ☝ The cursor must be located on the line of a marked or updated item.	• `U` • `M-<DEL>`	(**vc-dir-unmark-all-files** ARG)	Unmark all files with the same state as the current one. With a prefix argument unmark all files. • If the current entry is a directory, unmark all the child files. • The commands operate on files that are on the same state. • This command is intended to make it easy to deselect all files that share the same state.
Pull from parent repo **hg pull**	`+`	(**vc-update** &optional ARG)	Execute: **hg pull** Update the current fileset or branch. • Optional prefix ARG forces a prompt for the Mercurial command to run. ☝For Mercurial, to pull and update, type `C-u +` to get the hg pull prompt and then add `-u` to the command to also perform an **hg update**
Hide items	`x`	(**vc-dir-hide-up-to-date** &optional STATE)	Hide items that are in STATE from display. • Hitting **x** alone hides both 'up-to-date' and 'ignored' items. • To hide a specific file status, move point to a file with that status and then hit `C-u x` while on that line. • To view the files again, hit **g**. ⚠ Unfortunately the current implementation never shows the up-to-date and ignored items again at least with the implementation as of early 2020. There is a bug on Emacs for this and it was closed without action in October 2019. The reason was they did not see a good reason for fixing it… Well.. being able to open the file from the vc-dir buffer would be one reason…
Close the window	`q`	(**quit-window** &optional KILL WINDOW)	Quit WINDOW and bury its buffer.
Commands available in the *vc-change-log* buffer	The `C-x v L` and `C-x v l` commands list the repo change log into the ***vc-change-log*** buffer. The following commands are available in the buffer. ⚠Note that some other commands that are available (like **e** to modify the change comment) do not work for the Mercurial back-end and navigation commands. Use the **describe-mode** command to get more information.		
Show help for the mode	• `?` • `h`	(**describe-mode** &optional BUFFER)	Opens the *Help* buffer with information about the mode, listing the various key bindings and commands available
Toggle log line entry view	`RET`	(**log-view-toggle-entry-display**)	Toggle the view of the log line between a single line to a multi-line with more details.
Show diff for log entry	`=`	(**vc-diff** &optional HISTORIC NOT-URGENT)	Get and show the diff between the two revisions. • If the region is inactive or the mark is on the revision at point, get the diff between the revision at point and its previous revision. Otherwise, get the diff between the revisions where the region starts and ends.
Visit the content of the revision at point.	`f`	(**log-view-find-revision** POS)	Visit the version at POS. If called interactively, visit the version at point.
Mark log entry for future diff operation.	`m`	(**log-view-toggle-mark-entry**)	Mark the current log entry line to use as the version to participate in a diff operation. • Toggle the marked state for the log entry at point. • Individual log entries can be marked and unmarked. The marked entries are denoted by changing their background color. • 'log-view-get-marked' returns the list of tags for the marked log entries.
Close the window	`q`	(**quit-window** &optional KILL WINDOW)	Quit WINDOW and bury its buffer.
***Annotate* buffers** **hg annotate**	The C-x v g command opens an ***annotate*** buffer where the file revision annotation is shown. The following single key commands are available in this buffer. Use the **describe-mode** command to get more information.		
Show help for the mode	• `?` • `h`	(**describe-mode** &optional BUFFER)	Opens the *Help* buffer with information about the mode, listing the various key bindings and commands available
Show the diff of the revision of the current annotated line	• `d` • `=`	(**vc-annotate-show-diff-revision-at-line**)	Visit the diff of the revision at line from its previous revision. • Open the changes diff in the ***vc-diff*** buffer.
Show the change set corresponding to the annotated line.	`D`	(**vc-annotate-show-changeset-diff-revision-at-line**)	Visit the diff of the revision at line from its previous revision for all files in the changeset. • Open the changes diff in the ***vc-diff*** buffer.
Open the file for the revision that corresponds to the revision of the current annotated line	`f`	(**vc-annotate-find-revision-at-line**)	Visit the file at the revision identified in the current line. • The file at specific revision is opened in a buffer that has the same file name with a suffix that identifies the revision number.
Open the log of the revision of the current annotated line	`l`	(**vc-annotate-show-log-revision-at-line**)	Visit the log of the revision at line; only show the log entry for the revision. • If a *vc-change-log* buffer exists and already shows a log for the file in question, search for the log entry required and move point.

Operation	Keystroke	Function	Note
Update the annotated view to the revision as it was at the version of the current annotated line	**j**	(**vc-annotate-revision-at-line**)	Visit the annotation of the revision identified in the current line.
Update the annotated view to the next revision	**n**	(**vc-annotate-next-revision** PREFIX)	Visit the annotation of the revision after this one. • With a numeric prefix argument, annotate the revision that many revisions after.
Update the annotated view to the previous revision	**p**	(**vc-annotate-prev-revision** PREFIX)	Visit the annotation of the revision previous to this one. • With a numeric prefix argument, annotate the revision that many revisions previous.
Update the annotated view to the current working version	**w**	(**vc-annotate-working-revision**)	Visit the annotation of the working revision of this file.
Close the window	**q**	(**quit-window** &optional KILL WINDOW)	Quit WINDOW and bury its buffer.
Using Monky • **Monky User Manual**	📦 Requires the **Monky** external package. PEL activates it when the ***pel-use-monky*** user option is turned on (set to **t**). Monky provides another way to control Mercurial from within Emacs. • It is similar to the highly praised Magit but for supports Mercurial instead of Git. ☝ One advantage of using Monky over the standard VC implementation with Mercurial support is that you can use Monky to create or manage a Mercurial repo for a directory tree that is also managed as a Git repo. The Monky commands are shown below. The first section show the global commands. The next section show commands available in multiple Monky buffers, followed by the commands used in the Monky repo buffer that apply generally, followed by the commands that apply to specific items.		
Show the status of a Mercurial repository using Monky	**<f11> v m s**	(**monky-status** &optional DIRECTORY)	Show the status of Hg repository. Open a *monky: repo* buffer in monky-mode.
Show the current file blame • **hg annotate** • **Monky blame**	**<f11> v m b**	(**monky-blame-current-file**)	Show the current file in a *monky-blame* buffer with each line showing the mercurial changeset of the last change. • Hit RET on the line to open a view of the corresponding Mercurial changeset inside a *monky-commit* buffer.
Monky Modes commands	The following commands are available in the *monky: repo* buffer and other Money buffers., the buffer that shows the status of the Mercurial repository. These commands are available in mostly all Monky buffers.		
Show help for the mode	• **?** • **h**	(**describe-mode** &optional BUFFER)	Opens the *Help* buffer with information about the mode, listing the various key bindings and commands available
Move to beginning of buffer	**<**	(**beginning-of-buffe**r &optional ARG)	Move point to the beginning of buffer.
Move to end of buffer	**>**	(**end-of-buffer** &optional ARG)	Move point to the end of buffer.
Move point to next section	**n**	(**monky-goto-next-section**)	Go to the next monky section.
Move point to previous section	**p**	(**monky-goto-previous-section**)	Goto the previous monky section.
Contract/expand current section	**TAB**	(**monky-toggle-section**)	Toggle hidden status of current section
Collapse all sections	**M-1**	(**monky-section-show-level-1-all**)	Collapse all the sections in the monky status buffer.
Show files that changed	**M-2**	(**monky-section-show-level-2-all**)	Show all the files changes, but not their contents.
Expand file content	**M-3**	(**monky-section-show-level-3-all**)	Expand all file contents and line numbers, but not the actual changes.
Expand all sections	**M-4**	(**monky-section-show-level-4-all**)	Expand all sections.
Visit current item	**RET**	(**monky-visit-item** &optional OTHER-WINDOW)	Visit current item: open the item in the current window buffer. • With a prefix argument, visit in other window.
Show current item / Scroll window up	**SPC**	(**monky-show-item-or-scroll-up**)	Show current item (if possible) in a separate window. Or scroll window up.
Scroll down	**S-SPC**	(**scroll-down-command** &optional ARG)	Scroll text of selected window down ARG lines; or near full screen if no ARG. • If 'scroll-error-top-bottom' is non-nil and 'scroll-down' cannot scroll window further, move cursor to the top line. • When point is already on that position, then signal an error. • A near full screen is 'next-screen-context-lines' less than a full screen. • Negative ARG means scroll upward. • If ARG is the atom '-', scroll upward by nearly full screen.
Show current item / Scroll window down	**DEL**	(**monky-show-item-or-scroll-down**)	Show current item (if possible) in a separate window. Or scroll window down.
Refresh buffer	**g**	(**monky-refresh**)	Refresh current buffer to match repository state. • Also revert every unmodified buffer visiting files in the corresponding directory.
Quit window	**q**	(**monky-quit-window** &optional KILL-BUFFER)	Bury the buffer and delete its window. • With a prefix argument, kill the buffer instead.
Monky repo Commands	The following commands are available in the *monky: repo* buffer; the buffer that shows the status of the Mercurial repository. These commands apply to the entire buffer, as opposed to commands that apply to specific entries, listed below.		
Display output of last hg command	**$**	(**monky-display-process**)	Display output from most recent hg command.
Perform arbitrary hg command	**:**	(**monky-hg-command** COMMAND)	Perform arbitrary Hg COMMAND. • Do not type the 'hg' command just what follows. • The result is shown inside a *monky-process* buffer window. • Dismiss that buffer with **q**.
Enter Monky-queue minor mode to manage Hg Queues See also: **Money Queue**	**Q**	(**monky-queue**)	Enter the monky-queue-mode, the minor mode for hg mq, an extension to manage a stack of patches. ⚠ The **hg mq extension** must be installed separately.
Show branches **hg branch** **Monky branch**	**b**	(**monky-branches**)	Open a *monky-branches* buffer window to show the permanent branches. • Use **C** to checkout a branch.
Open the ediff diff/merge editor on the current item See ⧖ **Diff & Merge**	**e**	(**monky-ediff-item**)	Open the ediff merge editor on the item.
Show entire log	**l a**	(**monky-log-all**)	Open a *monky-log* buffer window that shows the entire log of the repository. • Simple key commands can be issued in that buffer to view the content of each log entry.
Show entire log of current branch	**l l**	(**monky-log-current-branch**)	Open a *monky-log* buffer window that shows the entire log of the current branch of the repository. • Simple key commands can be issued in that buffer to view the content of each log entry.
Show log entry for specific revision	**l r**	(**monky-log-revset** REVSET)	Open a *monky-log* buffer window that shows the log entry for the specified revision. • Prompts for the revision. This is the Mercurial short changeset ID.
Roll back last transaction **hg rollback** ⚠	**L**	(**monky-rollback**)	Roll back last transaction. Does not prompt!!! ⚠ This is a DANGEROUS and DEPRECATED command. Use commit amend to correct mistakes in the last commit instead.
Add specific file	**s**	(**monky-stage-item**)	Add the item at point to the staging area.
Add all files	**S**	(**monky-stage-all**)	Add all items in Changes to the staging area.

Operation	Keystroke	Function	Note
Un-select file	`u`	(**monky-unstage-item**)	Remove the item at point from the staging area.
Un-select all files	`U`	(**monky-unstage-all**)	Remove all items from the staging area
Merge	`M`	(**monky-merge** NODE)	Merge
Push current branch to parent **hg push**	`P`	(**monky-push**)	Pushes current branch to the default path.
Discard all uncommitted changes **hg update --clean**	`X`	(**monky-reset-tip**)	Discard all uncommitted changes. Prompts before proceeding. ⚠️ There is no way to go back. Make sure you do not need the uncommitted changes.
Create bookmarks **hg bookmarks**	`y`	(**monky-bookmark-create** BOOKMARK-NAME)	Create a bookmark at the current location
Monky Status Commands	The following commands apply to the item at point in one of the various Monky buffer windows.		
Commit amend **hg commit --amend**	`a`	(**monky-commit-amend**)	Amends previous commit. Brings up a buffer to allow editing of commit message. ☝ The command line is more powerful; it allows changing more than the commit message.
addremove: Add all new files, delete missing files	`A`	(**monky-addremove-all**)	
Show branches **hg branch**	`b`	(**monky-branches**)	Open a *monky-branches* buffer window to show the permanent branches.
Reverse effect of previous changeset **hg backout**	`B`	(**monky-backout** REVISION)	Runs hg backout to reverse effect of earlier changeset.
Edit commit message	`c`	(**monky-log-edit**)	Bring up a buffer to allow editing of commit messages. • Complete with `C-c C-c` or cancel with `C-c C-k`
Check out item • **hg update**	`C`	(**monky-checkout** NODE)	Checkout the revision represented by specified item.
		(**monky-checkout-item**)	Checkout the revision represented by current item.
Hg pull (fetch)	`f`	(**monky-pull**)	Run hg pull. • The monky-pull-args variable contains extra arguments to pass to hg.
Delete untracked file/revert tracked file	`k`	(**monky-discard-item**)	Delete the file if not tracked, otherwise revert it.
Mark item at point resolved	`m`	(**monky-resolve-item**)	Mark the item at point as resolved.
Mark item at point unresolved	`x`	(**monky-unresolve-item**)	Mark the item at point as unresolved.

VCS - Mercurial — Reference

Topic/URL	Comment
Mercurial itself	
Mercurial @ Wikipedia	Describes Mercurial
Mercurial home page	Official Mercurial home. Documentation is available on this website.
Mercurial in Emacs	
Mercurial @ EmacsWiki	Lists the various Emacs packages that support Mercurial within Emacs
Using Mercurial in Emacs @ superuser	Discussion about the alternatives as they were in 2014.
Mercurial margin with Emacs	How to setup .hgrc to support command line Hg merge using Emacs.
- Emacs VC Mode	
GNU Emacs Manual - 28.1: Version Control	Describes the VC mode, the integrated VCS mode, that supports several systems, including Mercurial. Note an important variable: *vc-handled-backends*. It contains a list of supported (D)VCS backends. To completely disable VC, set it to nil.
Emacs VC Mode and Mercurial	A quick introduction to VC mode for Mercurial, from the Mercurial wiki. Lists enough Emacs commands to get going. • The page reports that VC support for hg pull and hg push was broken in Emacs 23.1-24.1.1, but now it works fine in Emacs 26.
vc.el	Emacs file: drives a version control system within Emacs.
vcdir.el	Directory status display under VC
- Mercurial.el	
Mercurial Mode @ EmacsWiki	The mercurial.el implements a mercurial mode, which is another package that supports Mercurial.
How to use Emacs to work with Mercurial @ alexott.net	A blog describing how to use mercurial.el package.
mercurial.el @ GitHub	
- Monky	
Monky - an Emacs mode for Hg	Monky is a Mercurial support interface that is similar to Magit, the very popular Emacs support package for Git.
Monky User Manual	
- aHG	
aHg: An Emacs front-end for the Mercurial SCM	Description of aHG from the author.
aHG @ MELPA	
- hgignore-mode	A mode to edit Mercurial .hgignore files.