↖ Topic Index

VCS — Mercurial

Operation	Keystroke	Function	Note
Emacs Version Control	Emacs supports version control through its VC interface. That works well with Mercurial, one of the supported (D)VCS. Emacs VC interface support multiple backends. • The list of backends it supports is controlled by the customizable variable *vc-handled-backends* which by default is set to (RCS CVS SVN SCCS SRC Bzr Git Hg Mtn). You can remove some of them if you do not want to support them. • 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. ☝While working with one backend, you can also temporary switch to another backend using the **`C-x v b`** command issued in a file or a directory under VC control. This can be of great use in some specific situations. Unfortunately this does not seem to work. 👽 To be able to manage the files of a directory tree with more than one (D)VCS through the VC interface, work with Emacs variables seem to be necessary. It's possible to make Emacs deal with various backends, one at a time and force selection of one backed by setting the *vc-dir-backend* variable to the backend you want to use at the moment. One way is to set the value of the *vc-dir-backend* variable for the *vc-dir-mode* via the directory local variables by adding the following inside the .dir-locals.el of the directory (or above): `(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. For example if you wanted to be able to control a set of files with both Git and Mercurial to compare them and test their capabilities, you could add the following statement in the **custom-set-variables** arguments: '(safe-local-variable-values (quote ((vc-dir-backend . Git) (vc-dir-backend . Hg)))) Why would someone want to do something like that? Well, you can use this technique for example if you are using Git for publishing your files to a shared Git repo and use Mercurial to store more or different files located in the same directory tree inside a private Mercurial repo you use for backup or keep versions of private files you do not want to publish to the Git repo yet still want them inside the same directory tree. Of course that means extra maintenance burden. ☝Another way to deal with directories that are managed with multiple DVCS (such as Git and Mercurial) is to use the VC interface for one and use *another* tool for the other DVCS. For Git you can use the excellent Magit library, and for Mercurial you can use the Monky library a Magit clone for Mercurial. See info on Monky toward the end of this table. **hgignore-mode support**. 📦 With the **hgignore-mode** external package Emacs will font lock .hgignore files. PEL activates it when the **pel-use-hgignore-mode** user-option is turned on (set to **t**).		
Open this PDF file. See also: **∑ Help/Info**	**`<f11> v <f1>`** **`<f12> <f1>`**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the **∑ VCS-Mercurial** PDF using method specified by the **pel-open-pdf-method** user-option or the alternate one if a command prefix (like **`C-u`**) was used.
∑ 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.
∑ Customize Emacs VCS control	**`<f11> v <f3>`** **`<f12> <f3>`**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs Version Control System support: vc, vc-hg, vc-git, magit, monky. Customize Emacs Version Control System support: vc, vc-hg, vc-git.
VC commands while editing files	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.		
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.

Pull files from parent repo:
 	
hg pull
	`C-x v +`	(**vc-update** &optional ARG)	Executes: **hg pull** command 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.

Diff versions of current file:
 	
hg diff
	`C-x v =`	(**vc-diff** &optional HISTORIC NOT-URGENT)	Executes: **hg diff** command 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, it reads 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 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 the repository. • 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.

Operation	Keystroke	Function	Note
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.
List files incoming from parent (or specified) repo **hg incoming**	`C-x v 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.
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.
Show change sets not found in the destination **hg outgoing**	`C-x v 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.
Push changes to the specified destination **hg push**	`C-x v 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.
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. 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.
Annotate version of each line of file **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.
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.
Open the change log for the file	`C-x v l`	(**vc-print-log** &optional WORKING-REVISION LIMIT)	Executes: **hg log.** Open the change log for the file in the ***vc-change-log**** buffer.
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.
Retrieve a tagged version	`C-x v r`	(**vc-retrieve-tag** DIR NAME)	For each file in or below DIR, retrieve their tagged version NAME. • NAME can name a branch, in which case 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.
Create a tag **hg tag**	`C-x v s`	(**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.
Revert **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).
Commit: hg ci	`C-x v v`	(**vc-next-action** VERBOSE)	Executes: **hg commit**, when the file has non-committed changes.
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.
Rename a file **hg rename**	`M-x vc-rename-file`	(**vc-rename-file** OLD NEW)	Executes: **hg rename** Rename file OLD to NEW in both work area and repository. If called interactively, read OLD and NEW, defaulting OLD to the current buffer's file name if it's under version control.
Open a specific revision of the file in another window	`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.
***vc-dir* buffer**	The C-x v d 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. The following shows some of the available commands. Use the **describe-mode** command to get more information and list the other commands (type **?** in the buffer's window).		
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. 💡 This is really 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.

Operation	Keystroke	Function	Note
Unmark all files	• **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. ⚠️ This only seems to works when point is on the line of an marked or updated item.
Add or commit marked files	**v**	(**vc-next-action** VERBOSE)	Executes:, for all marked files: • **hg add** of the files if the files are not part of the repository yet. • **hg commit**, of all marked files, otherwise.
Diff marked files against current repository revision	**=**	(**vc-diff** &optional HISTORIC NOT-URGENT)	Executes: **hg diff** command to list differences between current file version in repo and local file's content for all marked files.
Push current branch	**P**	(**vc-push** &optional ARG)	Executes: **hg push**.
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**
Ignore the current file	**G**	(**vc-dir-ignore**)	Ignore the current file; update the .hgignore file.
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…
Show items	**g**	(**revert-buffer** &optional IGNORE-AUTO NOCONFIRM PRESERVE-MODES)	Restore buffer to the default list view.
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
Close the window	**q**	(**quit-window** &optional KILL WINDOW)	Quit WINDOW and bury its buffer.
***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. There are some other commands that are available (like **e** to modify the change comment) that do not work for the Mercurial back-end and navigation commands. Use the **describe-mode** command to get more information.		
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	• **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).
Show diff for log entry	**D**	(**log-view-diff-changeset** BEG END)	Does the same as the command described above at least for Mercurial).
	f		
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.
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
Close the window	**q**	(**quit-window** &optional KILL WINDOW)	Quit WINDOW and bury its buffer.
***Annotate* buffers**	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 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.
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.

Operation	Keystroke	Function	Note
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
Close the window	**q**	(**quit-window** &optional KILL WINDOW)	Quit WINDOW and bury its buffer.
Using Monky	📦 The Monky library provides another way to control Mercurial from within Emacs. It is similar to the highly praised Magit but for supports Mercurial instead of Git. • It must first be installed. PEL activates it with the ***pel-use-monky*** user option. ☝ 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. ⚠🚧 Monky operation requires further analysis and code review to properly document. Some commands are listed below but its not fully documented . More work required.		
Show the status of a Mercurial repository using Monky	**<f11> v m**	(**monky-status** &optional DIRECTORY)	Show the status of Hg repository.
Monky Status Commands			
	a	(**monky-commit-amend**)	Amends previous commit. Brings up a buffer to allow editing of commit message.
	A	(**monky-addremove-all**)	
	b	(**monky-branches**)	
	B	(**monky-backout** REVISION)	Runs hg backout.
	c	(**monky-log-edit**)	Bring up a buffer to allow editing of commit messages.
	C	(**monky-checkout** NODE)	
	e	(**monky-ediff-item**)	Open the ediff merge editor on the item.
Hg pull (fetch)	**f**	(**monky-pull**)	Run hg pull. • The monky-pull-args variable contains extra arguments to pass to hg.
	g	(**monky-refresh**)	Refresh current buffer to match repository state. • Also revert every unmodified buffer visiting files in the corresponding directory.
	• **h** • **?**	(**describe-mode** &optional BUFFER)	Display documentation of current major mode and minor modes. • A brief summary of the minor modes comes first, followed by the major mode description. This is followed by detailed descriptions of the minor modes, each on a separate page.
	k	(**monky-discard-item**)	Delete the file if not tracked, otherwise revert it.
	l a	(**monky-log-all**)	
	l l	(**monky-log-current-branch**)	
	l r	(**monky-log-revset** REVSET)	
	L	(**monky-rollback**)	
	m	(**monky-resolve-item**)	
	M	(**monky-merge** NODE)	
	n	(**monky-goto-next-section**)	Go to the next monky section.
	p	(**monky-goto-previous-section**)	Goto the previous monky section.
	P	(**monky-push**)	Pushes current branch to the default path.
	q	(**monky-quit-window** &optional KILL-BUFFER)	Bury the buffer and delete its window. • With a prefix argument, kill the buffer instead.
	Q	(**monky-queue**)	
	s	(**monky-stage-item**)	Add the item at point to the staging area.
	S	(**monky-stage-all**)	Add all items in Changes to the staging area.
	u	(**monky-unstage-item**)	Remove the item at point from the staging area.
	U	(**monky-unstage-all**)	Remove all items from the staging area
	x	(**monky-unresolve-item**)	Mark the item at point as unresolved.
	X	(**monky-reset-tip**)	
	y	(**monky-bookmark-create** BOOKMARK-NAME)	Create a bookmark at the current location
	$	(**monky-display-process**)	Display output from most recent hg command.
	SPC	(**monky-show-item-or-scroll-up**)	
	:	(**monky-hg-command** COMMAND)	Perform arbitrary Hg COMMAND.
	RET	(**monky-visit-item** &optional OTHER-WINDOW)	Visit current item. • With a prefix argument, visit in other window.
	M-1	(**monky-section-show-level-1-all**)	Collapse all the sections in the monky status buffer.
	M-2	(**monky-section-show-level-2-all**)	Show all the files changes, but not their contents.
	M-3	(**monky-section-show-level-3-all**)	Expand all file contents and line numbers, but not the actual changes.
	M-4	(**monky-section-show-level-4-all**)	Expand all sections.

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.
- 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.