**depot\_tools\_tutorial(7) Manual Page**

**NAME**

depot\_tools\_tutorial - A tutorial introduction to the Chromium depot\_tools git extensions.

**DESCRIPTION**

The Chromium [depot\_tools(7)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools.html) suite contains many git workflow-enhancing tools which are designed to work together to enable anyone to wrangle the Chromium codebase expertly. This tutorial explains how to do development on Chromium using these tools. This will cover:

* [Setting up](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools_tutorial.html#_setting_up)
* [Getting the code](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools_tutorial.html#_getting_the_code)
* [TL;DR](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools_tutorial.html#_tl_dr)
* [Creating / Uploading a CL](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools_tutorial.html#_creating_uploading_a_cl)
* [Updating the code](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools_tutorial.html#_updating_the_code)
* [Managing multiple CLs](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools_tutorial.html#_managing_multiple_cls)
* [Managing dependent CLs](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools_tutorial.html#_managing_dependent_cls)
* [Example Walkthrough](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools_tutorial.html#_example_walkthrough)

Please refer to the manpages (or --help output) for details about any of the commands mentioned in this tutorial.

|  |  |
| --- | --- |
| **Note** | If your platform does not support manpages (or you prefer something a bit more expressive(富有表现) than plain text) you can find all documentation in *html* form in the [DEPOT\_TOOLS]/man/html folder. |

**PREREQUISITES**

This tutorial assumes basic familiarity with git terminology（专门名词） and concepts. If you need to brush up（复习） on these, the following are very good resources:

* [Think like (a) Git](http://think-like-a-git.net/) - A lighthearted overview of git. If you’re sorta-familiar with git, but not *comfortable* with it, then give this a look.
* [Git Immersion（沉浸; 洗礼; ） Tutorial](http://gitimmersion.com/) - An in-depth git tutorial.
* [pcottle’s Visual Git Branching](http://pcottle.github.io/learnGitBranching) - An excellent interactive/graphical demo on how git handles commits, branches, and shows the operations git performs on them.
* [Pro Git book](http://git-scm.com/book) - “The” book for learning git from basics to advanced concepts. A bit dry, but very through.

If you’ve tried these out and are still having some trouble getting started, there are *many* other resources online which should help. If you’re *really* ***really*** stuck（动不了的; 被卡住的; 被…缠住的; 被…难住的，不知所措; ）, then chat up（闲谈; ） one of the Chromium infrastructure（基础设施; 基础建设; ） team members for some pointers.

Litmus Test

If you know what git add, git status, git commit do and you know *essentially* what git rebase does, then you should know enough to follow along.

**SETTING UP**

**GET DEPOT TOOLS**

**LINUX / MAC**

Clone the *depot\_tools* repository:

**$ git clone https://chromium.googlesource.com/chromium/tools/depot\_tools**

Add *depot\_tools* to the *end* of your PATH (you will probably want to put this in your ~/.bashrc or ~/.zshrc). Assuming you cloned *depot\_tools* to /path/to/depot\_tools:

**$ export PATH=$PATH:/path/to/depot\_tools**

**WINDOWS**

Download the *depot\_tools* [bundle](https://src.chromium.org/svn/trunk/tools/depot_tools.zip) and extract it somewhere.

|  |  |
| --- | --- |
| **Warning** | **DO NOT** use drag-n-drop or copy-n-paste extract from Explorer, this will not extract the hidden “.git” folder which is necessary for *depot\_tools* to autoupdate itself. You can use “Extract all…” from the context menu though. |

Add *depot\_tools* to the *end* of your PATH. Assuming you unzipped the bundle to C:\workspace\depot\_tools:

With Administrator access:

**Control Panel → System and Security → System → Advanced system settings**

Modify the PATH system variable to include C:\workspace\depot\_tools.

Without Administrator access:

**Control Panel → User Accounts → User Accounts → Change my environment variables**

Add a PATH user variable: %PATH%;C:\workspace\depot\_tools.

From a cmd.exe shell, run the command gclient (without arguments). On first run, gclient will install all the Windows-specific(windows专用) bits needed to work with the code, including msysgit and python.

|  |  |
| --- | --- |
| **Note** | * If you run gclient from a non-cmd shell (e.g., cygwin, PowerShell), it may appear to run properly, but msysgit, python, and other tools may not get installed correctly. * If you see strange errors with the file system on the first run of gclient, you may want to [disable Windows Indexing](http://tortoisesvn.tigris.org/faq.html#cantmove2). * If you are running Windows XP and see errors like “The system cannot execute the specified program”, try installing the [“Microsoft Visual C++ 2008 Redistributable Package”](http://code.google.com/p/chromium/issues/detail?id=75886). |

**BOOTSTRAPPING CONFIGURATION**

If you have never used git before, you’ll need to set some global git configurations; substitute（代替，替换，代用;） your name and email address in the following commands:

**$ git config --global user.name “John Doe”**

**$ git config --global user.email “jdoe@email.com”**

**$ git config --global core.autocrlf false**

**$ git config --global core.filemode false**

**$** # and for fun!

**$ git config --global color.ui true true**

**TL;DR**

**$** # get the code

**$** # In an empty directory:

**$ fetch {chromium,blink,...}**

**$** # Update third\_party repos and run pre-compile hooks

**$ gclient sync**

**$** # Make a new change and upload it for review

**$ git new-branch <branch\_name>**

**$** # repeat: [edit, git add, git commit]

**$ git cl upload**

**$** # After change is reviewed, commit with the CQ

**$ git cl set\_commit**

**$** # Note that the committed hash which lands will /not/ match the

**$** # commit hashes of your local branch.

**GETTING THE CODE**

Pick an empty directory and run one of the following:

**$ fetch chromium** # Basic checkout for desktop Chromium

**$ fetch blink** # Chromium code with Blink checked out to tip-of-tree

**$ fetch android** # Chromium checkout for Android platform

**$ fetch ios** # Chromium checkout for iOS platform

When the fetch tool completes you should have the following in your working directory:

**.gclient** # A configuration file for you source checkout

**src/** # Top-level Chromium source checkout.

If you are on linux and fetching the code for the first time, then you’ll need to run:

**$ cd src && ./build/install-build-deps.sh**

And finally:

**$ gclient sync**

This will pull all dependencies of the Chromium src checkout. You will need to run this any time you update the main src checkout, including when you switch branches.

**CREATING / UPLOADING A CL**

|  |  |
| --- | --- |
| **Note** | The remainder of the tutorial assumes that your current working directory is the src/ folder mentioned in [Getting the code](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools_tutorial.html#_getting_the_code). |

Each CL corresponds exactly with a single branch in git. Any time you want to begin a new CL:

**$ git new-branch <branch\_name>**

This will create and checkout a new branch named branch\_name which will track the default upstream branch (origin/master). See [git-new-branch(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-new-branch.html) for more features.

Commit as many changes as you like to this branch. When you want to upload it for review, run:

**$ git cl upload**

This will take the diff of your branch against its upstream branch (in that case origin/master), and will post it to the [Chromium code review site](https://codereview.chromium.org/).

**UPDATING THE CODE**

Inevitably, you’ll want to pull in changes from the main Chromium repo. This is pretty easy with *depot\_tools*:

**$ git rebase-update**

This command will update all of your CLs to contain the latest code from their upstreams. It will also automatically clean up CLs which have been committed and a couple other nice things. See [git-rebase-update(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-rebase-update.html) for the full scoop.

One thing to look out for are *merge conflicts*. These happen for exactly the same as they do with SVN, but the experience is a little more controllable with git. git rebase-update will try to rebase all your branches for you, but if it encounters a merge conflict in one, it will halt and leave you in a rebase conflict state (see [git-rebase(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-rebase.html)). Resolving git rebase merge conflicts is beyond the scope of this tutorial, but there are many good sources online (see the [Prerequisites](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools_tutorial.html#_prerequisites) for some).

Sometimes you’re pretty certain that you’ve committed a certain branch, but git rebase-update isn’t able to tell that for sure. This is usually because your branch doesn’t rebase cleanly. You could just delete the branch with git branch -D <branch>, but you’d like to double check the diff of your branch against its upstream before deleting it. If this is the case you can abort the rebase started by git rebase-update, and then run [git-squash-branch(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-squash-branch.html) to flatten your branch into a single commit. When you run git rebase-update again, you’ll get a (hopefully) much smaller / saner diff. If it turns out you were wrong about your branch being fully committed, you can use [git-reflog(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-reflog.html) to reset your branch back to where it was before. If the diff looks inconsequential, you can use git rebase --skip to ignore it, and then git rebase-update will clean it up for you.

Once you’re done resolving all of the merge conflict, just run git rebase-update, and it will pick up where it left off. Once the command has finished updating all of your branches, it will return you back to the branch you started on.

|  |  |
| --- | --- |
| **Note** | Running git rebase-update will update all your branches, but it will not automatically run gclient sync to update your dependencies. |

**MANAGING MULTIPLE CLS**

Sometimes you want to work on more than one CL at once (say, you have a CL posted for review and want to work on something else). For each CL that you want to work on, just use git new-branch <branch\_name>.

Once you start to have more than one CL at a time, it can be easy to lose your bearings. Fortunately, *depot\_tools* has two tools to help you out:

**$ git map**

\*​ **7dcfe47**  (​**frozen\_changes**​) 2014-03-12 ~ FREEZE.unindexed

\* **4b0c180** 2014-03-12 ~ modfile

\* **59a7cca** 2014-03-12 ~ a deleted file

\* **6bec695** (​origin/master​) 2014-03-11 ~ Add neat feature **<(frozen\_changes)**

\* **d15a38a** 2014-03-11 ~ Epic README update

\* **d559894** (​**master**​) 2014-03-11 ~ Important upstream change

| \* **9c311fd** (​**cool\_feature**​) 2014-03-11 ~ Respond to CL comments

| | \* **2a1eeb2** (​**subfeature**​) 2014-03-11 ~ integrate with CoolService

| | \* **d777af6** 2014-03-11 ~ slick commenting action

| |/

| \* **265803a** 2014-03-11 ~ another improvement **<(subfeature)**

| \* **6d831ac** (​**spleen\_tag**​) 2014-03-11 ~ Refactor spleen

| \* **82e74ab** 2014-03-11 ~ Add widget

|/

\* **d08c5b3** (​**bogus\_noparent**​) 2014-03-11 ~ Wonderful beginnings **<(cool\_feature)**

Note that this example repo is in dire need of a [git-rebase-update(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-rebase-update.html)!

**$ git map-branches**

origin/master

cool\_feature

subfeature

frozen\_changes \*

master

[git-map(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-map.html)

This tool shows you the history of all of your branches in a pseudo-graphical format. In particular, it will show you which commits all of your branches are on, which commit you currently have checked out, and more. Check out the doc for the full details.

[git-map-branches(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-map-branches.html)

This tool just shows you which branches you have in your repo, and their upstream relationship to each other (as well as which branch you have checked out at the moment).

Additionally, sometimes you need to switch between branches, but you’ve got work in progress. You could use [git-stash(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-stash.html), but that can be tricky to manage because you need to remember which branches you stashed what changes on. Helpfully *depot\_tools* includes two tools which can greatly assist in case:

[git-freeze(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-freeze.html) allows you to put the current branch in 'suspended animation' by committing your changes to a specially-named commit on the top of your current branch. When you come back to your branch later, you can just run [git-thaw(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-thaw.html) to get your work-in-progress changes back to what they were.

Another useful tool is [git-rename-branch(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-rename-branch.html). Unlike git branch -m <old>< new>, this tool will correctly preserve the upstream relationships of your branch compared to its downstreams.

Finally, take a look at [git-upstream-diff(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-upstream-diff.html). This will show you the combined diff for all the commits on your branch against the upstream tracking branch. This is *exactly* what git cl upload will push up to code review. Additionally, consider trying the --wordwise argument to get a colorized per-word diff (instead of a per-line diff).

**MANAGING DEPENDENT CLS**

Now that you know how to manage *independent* CLs, we’ll see how to manage *dependent* CLs. Dependent CLs are useful when your second (or third or fourth or …) CL depends on the changes in one of your other CLs (such as: CL 2 won’t compile without CL 1, but you want to submit them as two separate reviews).

Like all of the other CLs we’ve created, we use [git-new-branch(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-new-branch.html), but this time with an extra argument. First, git checkout the branch you want to base the new one on (i.e. CL 1), and then run:

**$ git new-branch --upstream\_current <branch\_name>**

This will make a new branch which tracks the *current* branch as its upstream (as opposed to *origin/master*). All changes you commit to this branch will be in addition to the previous branch, but when you git cl upload, you will only upload the diff for the dependent (child) branch. You may have as many branches nested in this fashion as you like.

[git-map(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-map.html) and [git-map-branches(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-map-branches.html) are particularly helpful when you have dependent branches. In addition, there are two helper commands which let you traverse your working copy up and down this tree of branches: [git-nav-upstream(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-nav-upstream.html) and [git-nav-downstream(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-nav-downstream.html).

Sometimes when dealing with dependent CLs, it turns out that you accidentally based a branch on the wrong upstream, but since then you’ve committed changes to it, or even based *another* branch off of that one. Or you discover that you have two independent CLs that would actually be much better off as dependent CLs. In instances like these, you can check out the offending branch and use [git-reparent-branch(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-reparent-branch.html) to move it to track a different parent. Note that this can also be used to move a branch from tracking origin/master to lkgr or vice versa.

**EXAMPLE WALKTHROUGH**

This section will demo what a typical workflow looks like when writing, updating, and committing multiple CLs.

**$ fetch chromium**

... truncated output ...

**$ cd src**

(only on linux)

**$ ./build/install-build-deps.sh**

... truncated output ...

Pull in all dependencies for HEAD

**$ gclient sync**

... truncated output ...

Let's fix something!

**$ git new-branch fix\_typo**

**$ echo -e '/Banana\ns/Banana/Kuun\nwq' | ed build/whitespace\_file.txt**

1503

1501

It was a Domo-Banana.

It was a Domo-Kuun.

**$ git commit -am 'Fix terrible typo.'**

[fix\_typo 615ffa7] Fix terrible typo.

1 file changed, 1 insertion(+), 1 deletion(-)

**$ git map**

**\* 615ffa7**  (**fix\_typo**) 2014-04-10 ~ Fix terrible typo.

\* **beec6f4**  (origin/master, origin/HEAD) 2014-04-10 ~ Make ReflectorImpl use mailboxes **<(fix\_typo)**

\* **41290e0**  2014-04-10 ~ don't use glibc-specific execinfo.h on uclibc builds

\* **a76fde7**  2014-04-10 ~ [fsp] Add requestUnmount() method together with the request manager.

\* **9de7a71**  2014-04-10 ~ linux\_aura: Use system configuration for middle clicking the titlebar.

\* **073b0c2**  2014-04-10 ~ ContentView->ContentViewCore in ContentViewRenderView

\* **2250f53**  2014-04-10 ~ ozone: evdev: Filter devices by path

\* **33a7a74**  2014-04-10 ~ Always output seccomp error messages to stderr

**$ git status**

# On branch fix\_typo

# Your branch is ahead of 'origin/master' by 1 commit.

# (use "git push" to publish your local commits)

#

nothing to commit, working directory clean

**$ git cl upload -r domo@chromium.org --send-mail**

... truncated output ...

While we wait for feedback, let's do something else.

**$ git new-branch chap2**

**$ git map-branches**

**origin/master**

**chap2 \***

fix\_typo

**$ cat >> build/whitespace\_file.txt <<EOF**

"You recall what happened on Mulholland drive?" The ceiling fan rotated slowly

overhead, barely disturbing the thick cigarette smoke. No doubt was left about

when the fan was last cleaned.

EOF

**$ git status**

# On branch chap2

# Your branch is up-to-date with 'origin/master'.

#

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: build/whitespace\_file.txt

#

no changes added to commit (use "git add" and/or "git commit -a")

Someone on the code review pointed out that our typo-fix has a typo :( We're still working on 'chap2' but we really want to land 'fix\_typo', so let's switch over and fix it.

**$ git freeze**

**$ git checkout fix\_typo**

Switched to branch 'fix\_typo'

Your branch is ahead of 'origin/master' by 1 commit.

(use "git push" to publish your local commits)

**$ echo -e '/Kuun\ns/Kuun/Kun\nwq' | ed build/whitespace\_file.txt**

1501

1500

It was a Domo-Kuun.

It was a Domo-Kun.

**$ git upstream-diff --wordwise**

**diff --git a/build/whitespace\_file.txt b/build/whitespace\_file.txt**

**index 3eba355..57cdcee 100644**

**--- a/build/whitespace\_file.txt**

**+++ b/build/whitespace\_file.txt**

@@ -17,7 +17,7 @@ swept up the streets (for it is in London that our scene lies), rattling along

the housetops, and fiercely agitating the scanty flame of the lamps that

struggled against the elements. A hooded figure emerged.

It was a Domo-BananaKun.

"What took you so long?", inquired his wife.

**$ git commit -am 'Fix typo for good!'**

[fix\_typo 2c0ad9c] Fix typo for good!

1 file changed, 1 insertion(+), 1 deletion(-)

**$ git cl upload**

... truncated output ...

Since we got lgtm, let the CQ land it.

**$ git cl set\_commit**

**$ git map**

\* **0e2e52e**  (**chap2**) 2014-04-10 ~ FREEZE.unindexed

| **\* 2c0ad9c**  (**fix\_typo**) 2014-04-10 ~ Fix typo for good!

| \* **615ffa7**  2014-04-10 ~ Fix terrible typo.

|/

\* **beec6f4**  (origin/master, origin/HEAD) 2014-04-10 ~ Make ReflectorImpl use mailboxes **<(chap2, fix\_typo)**

\* **41290e0**  2014-04-10 ~ don't use glibc-specific execinfo.h on uclibc builds

\* **a76fde7**  2014-04-10 ~ [fsp] Add requestUnmount() method together with the request manager.

\* **9de7a71**  2014-04-10 ~ linux\_aura: Use system configuration for middle clicking the titlebar.

\* **073b0c2**  2014-04-10 ~ ContentView->ContentViewCore in ContentViewRenderView

\* **2250f53**  2014-04-10 ~ ozone: evdev: Filter devices by path

\* **33a7a74**  2014-04-10 ~ Always output seccomp error messages to stderr

Switch back to where we were using the nav\* commands (for fun... git checkout would work here too)

**$ git map-branches**

origin/master

chap2

**fix\_typo \***

**$ git nav-upstream**

Note: checking out 'origin/master'.

You are in 'detached HEAD' state. You can look around, make experimental

changes and commit them, and you can discard any commits you make in this

state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may

do so (now or later) by using -b with the checkout command again. Example:

git checkout -b new\_branch\_name

HEAD is now at beec6f4... Make ReflectorImpl use mailboxes

**$ git nav-downstream**

Please select a downstream branch

0. chap2

1. fix\_typo

Selection (0-1)[0]: 0

Previous HEAD position was beec6f4... Make ReflectorImpl use mailboxes

Switched to branch 'chap2'

Your branch is ahead of 'origin/master' by 1 commit.

(use "git push" to publish your local commits)

**$ git map-branches**

origin/master

**chap2 \***

fix\_typo

Now we can pick up on chapter2 where we left off.

**$ git thaw**

**$ git diff**

**diff --git a/build/whitespace\_file.txt b/build/whitespace\_file.txt**

**index 3eba355..9d08d9d 100644**

**--- a/build/whitespace\_file.txt**

**+++ b/build/whitespace\_file.txt**

@@ -34,3 +34,7 @@ with his fork, watching the runny jelly spread and pool across his plate,

like the blood of a dying fawn. \"It reminds me of that time --\" he started, as

his wife cut in quickly: \"-- please. I can't bear to hear it.\". A flury of

images coming from the past flowed through his mind.

+

+"You recall what happened on Mulholland drive?" The ceiling fan rotated slowly

+overhead, barely disturbing the thick cigarette smoke. No doubt was left about

+when the fan was last cleaned.

**$ cat >> build/whitespace\_file.txt <<EOF**

There was an poignant pause.

EOF

**$ git diff**

**diff --git a/build/whitespace\_file.txt b/build/whitespace\_file.txt**

**index 3eba355..e3a55de 100644**

**--- a/build/whitespace\_file.txt**

**+++ b/build/whitespace\_file.txt**

@@ -34,3 +34,9 @@ with his fork, watching the runny jelly spread and pool across his plate,

like the blood of a dying fawn. \"It reminds me of that time --\" he started, as

his wife cut in quickly: \"-- please. I can't bear to hear it.\". A flury of

images coming from the past flowed through his mind.

+

+"You recall what happened on Mulholland drive?" The ceiling fan rotated slowly

+overhead, barely disturbing the thick cigarette smoke. No doubt was left about

+when the fan was last cleaned.

+

+There was an poignant pause.

**$ git commit -am 'Finish chapter 2'**

[chap2 ceef712] Finish chapter 2

1 file changed, 6 insertions(+)

**$ git map**

**\* ceef712**  (**chap2**) 2014-04-10 ~ Finish chapter 2

| \* **2c0ad9c**  (**fix\_typo**) 2014-04-10 ~ Fix typo for good!

| \* **615ffa7**  2014-04-10 ~ Fix terrible typo.

|/

\* **beec6f4**  (origin/master, origin/HEAD) 2014-04-10 ~ Make ReflectorImpl use mailboxes **<(chap2, fix\_typo)**

\* **41290e0**  2014-04-10 ~ don't use glibc-specific execinfo.h on uclibc builds

\* **a76fde7**  2014-04-10 ~ [fsp] Add requestUnmount() method together with the request manager.

\* **9de7a71**  2014-04-10 ~ linux\_aura: Use system configuration for middle clicking the titlebar.

\* **073b0c2**  2014-04-10 ~ ContentView->ContentViewCore in ContentViewRenderView

\* **2250f53**  2014-04-10 ~ ozone: evdev: Filter devices by path

\* **33a7a74**  2014-04-10 ~ Always output seccomp error messages to stderr

**$ git cl upload -r domo@chromium.org --send-mail**

... truncated output ...

We poke a committer until they lgtm :)

**$ git cl set\_commit**

While that runs through the CQ, let's get started on chapter 3. Since we know that chapter 3 depends on chapter 2, we'll track the current chapter2 branch.

**$ git new-branch --upstream\_current chap3**

**$ cat >> build/whitespace\_file.txt <<EOF**

CHAPTER 3:

Mr. Usagi felt that something wasn't right. Shortly after the Domo-Kun left he

began feeling sick.

EOF

**$ git commit -am 'beginning of chapter 3'**

[chap3 7d4238a] beginning of chapter 3

1 file changed, 4 insertions(+)

**$ git map**

**\* 7d4238a**  (**chap3**) 2014-04-10 ~ beginning of chapter 3

\* **ceef712**  (**chap2**) 2014-04-10 ~ Finish chapter 2 **<(chap3)**

| \* **2c0ad9c**  (**fix\_typo**) 2014-04-10 ~ Fix typo for good!

| \* **615ffa7**  2014-04-10 ~ Fix terrible typo.

|/

\* **beec6f4**  (origin/master, origin/HEAD) 2014-04-10 ~ Make ReflectorImpl use mailboxes **<(chap2, fix\_typo)**

\* **41290e0**  2014-04-10 ~ don't use glibc-specific execinfo.h on uclibc builds

\* **a76fde7**  2014-04-10 ~ [fsp] Add requestUnmount() method together with the request manager.

\* **9de7a71**  2014-04-10 ~ linux\_aura: Use system configuration for middle clicking the titlebar.

\* **073b0c2**  2014-04-10 ~ ContentView->ContentViewCore in ContentViewRenderView

\* **2250f53**  2014-04-10 ~ ozone: evdev: Filter devices by path

\* **33a7a74**  2014-04-10 ~ Always output seccomp error messages to stderr

We haven't updated the code in a while, so let's do that now.

**$ git rebase-update**

Fetching origin

From https://upstream

beec6f4..59cdb73 master -> origin/master

Rebasing: chap2

Rebasing: fix\_typo

Failed! Attempting to squash fix\_typo ... Success!

Rebasing: chap3

Deleted branch fix\_typo (was 5d26fec).

Reparented chap3 to track origin/master (was tracking chap2)

Deleted branch chap2 (was 5d26fec).

Well look at that. The CQ landed our typo and chapter2 branches already and git rebase-update cleaned them up for us.

**$ gclient sync**

... truncated output ...

**$ git map**

**\* 93fe917**  (**chap3**) 2014-04-10 ~ beginning of chapter 3

\* **5d26fec**  (origin/master, origin/HEAD) 2014-04-10 ~ Finish chapter 2 **<(chap3)**

\* **df7fefb**  2014-04-10 ~ Revert 255617, due to it not tracking use of the link doctor page properly.

\* **4b39cda**  2014-04-10 ~ Fix terrible typo.

\* **248c5b6**  2014-04-10 ~ Temporarily CHECK(trial) in ChromeRenderProcessObserver::OnSetFieldTrialGroup.

\* **8171df8**  2014-04-10 ~ Remove AMD family check for the flapper crypto accelerator.

\* **d6a30d2**  2014-04-10 ~ Change the Pica load benchmark to listen for the polymer-ready event

\* **beec6f4**  2014-04-10 ~ Make ReflectorImpl use mailboxes

\* **41290e0**  2014-04-10 ~ don't use glibc-specific execinfo.h on uclibc builds

\* **a76fde7**  2014-04-10 ~ [fsp] Add requestUnmount() method together with the request manager.

\* **9de7a71**  2014-04-10 ~ linux\_aura: Use system configuration for middle clicking the titlebar.

\* **073b0c2**  2014-04-10 ~ ContentView->ContentViewCore in ContentViewRenderView

\* **2250f53**  2014-04-10 ~ ozone: evdev: Filter devices by path

\* **33a7a74**  2014-04-10 ~ Always output seccomp error messages to stderr

Someone on IRC mentions that they actually landed a chapter 3 already! We should pull their changes before continuing. Brace for a code conflict!

**$ git rebase-update**

Fetching origin

From https://upstream

5d26fec..59cdb73 master -> origin/master

Rebasing: chap2

... lots of output, it's a conflict alright :(...

**$ git diff**

**diff --cc build/whitespace\_file.txt**

**index 1293282,f903ea2..0000000**

**--- a/build/whitespace\_file.txt**

**+++ b/build/whitespace\_file.txt**

@@@ -42,4 -42,5 +42,9 @@@ when the fan was last cleaned

There was an poignant pause.

CHAPTER 3:

++<<<<<<< HEAD

+Hilariousness! This chapter is awesome!

++=======

+ Mr. Usagi felt that something wasn't right. Shortly after the Domo-Kun left he

+ began feeling sick.

++>>>>>>> beginning of chapter 3

Oh, well, that's not too bad. In fact... that's a terrible chapter 3!

**$ $EDITOR build/whitespace\_file.txt**

... /me deletes bad chapter 3 ...

**$ git add build/whitespace\_file.txt**

**$ git diff --cached**

**diff --git a/build/whitespace\_file.txt b/build/whitespace\_file.txt**

**index 1293282..f903ea2 100644**

**--- a/build/whitespace\_file.txt**

**+++ b/build/whitespace\_file.txt**

@@ -42,4 +42,5 @@ when the fan was last cleaned.

There was an poignant pause.

CHAPTER 3:

-Hilariousness! This chapter is awesome!

+Mr. Usagi felt that something wasn't right. Shortly after the Domo-Kun left he

+began feeling sick.

Much better

**$ git rebase --continue**

Applying: beginning of chapter 3

**$ git rebase-update**

Fetching origin

chap3 up-to-date

**$ gclient sync**

... truncated output ...

**$ git map**

**\* 1cb4f5b**  (**chap3**) 2014-04-10 ~ beginning of chapter 3

\* **59cdb73**  (origin/master, origin/HEAD) 2014-04-10 ~ Refactor data interchange format. **<(chap3)**

\* **34676a3**  2014-04-10 ~ Ensure FS is exited for all not-in-same-page navigations.

\* **7d4784e**  2014-04-10 ~ Add best chapter2 ever!

\* **5d26fec**  2014-04-10 ~ Finish chapter 2

\* **df7fefb**  2014-04-10 ~ Revert 255617, due to it not tracking use of the link doctor page properly.

\* **4b39cda**  2014-04-10 ~ Fix terrible typo.

\* **248c5b6**  2014-04-10 ~ Temporarily CHECK(trial) in ChromeRenderProcessObserver::OnSetFieldTrialGroup.

\* **8171df8**  2014-04-10 ~ Remove AMD family check for the flapper crypto accelerator.

\* **d6a30d2**  2014-04-10 ~ Change the Pica load benchmark to listen for the polymer-ready event

\* **beec6f4**  2014-04-10 ~ Make ReflectorImpl use mailboxes

\* **41290e0**  2014-04-10 ~ don't use glibc-specific execinfo.h on uclibc builds

\* **a76fde7**  2014-04-10 ~ [fsp] Add requestUnmount() method together with the request manager.

\* **9de7a71**  2014-04-10 ~ linux\_aura: Use system configuration for middle clicking the titlebar.

\* **073b0c2**  2014-04-10 ~ ContentView->ContentViewCore in ContentViewRenderView

\* **2250f53**  2014-04-10 ~ ozone: evdev: Filter devices by path

\* **33a7a74**  2014-04-10 ~ Always output seccomp error messages to stderr

**$ git cl upload**

... truncated output ...

So there you have the basic flow. Note that you don’t *have* to do chromium development using these tools. Any git workflow is compatible, as long as git cl upload is able to upload good patches.

**CONCLUSION**

Hopefully that gives you a good starting overview on Chromium development using *depot\_tools*. If you have questions which weren’t answered by this tutorial or the man pages for the tools (see the index of all tools here: [depot\_tools(7)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools.html)), please feel free to ask.

**GLOSSARY**

CL

A *change-list*. This is a diff which you would like to commit to the codebase.

DEPS

A file in the chromium checkout which gclient sync uses to determine what dependencies to pull in. This file also contains *hooks*.

LKGR

Last Known Good Revision. This is a [git-tag(1)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/git-tag.html) which tracks the last version of origin/master which has passed the full set of testing on the [main Chromium waterfall](http://build.chromium.org/).

**CHROMIUM DEPOT\_TOOLS**

Part of the chromium [depot\_tools(7)](http://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools.html) suite. These tools are meant to assist with the development of chromium and related projects. Download the tools from [here](https://chromium.googlesource.com/chromium/tools/depot_tools.git).

Last updated 2014-05-09 17:43:43 PDT