# Mozilla Conduit

*Release 0*

**Jun 17, 2021**

# Contents

Conduit is the set of systems involved in submitting code to mozilla-central, the Mercurial repository that contains all the code required to build Firefox. These systems handle everything from posting a patch for review up to landing it in mozilla-central. The applications are a mix of third-party tools, extensions to these tools, and custom applications. Code for these systems is in various places; please see the documentation for individual applications.

# CHAPTER 1

# Table of Contents

## 1.1 Mozilla Phabricator User Guide

### 1.1.1 User Guide

As Mozilla's Phabricator instance has only small modifications from stock Phabricator, much of Phabricator's user documentation is fully applicable. Several sections are of particular interest.

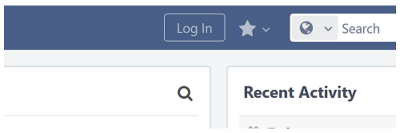
**Differential** is Phabricator's code-review tool. Useful articles include the Differential User Guide, the FAQ, and the Inline Comments guide. As usual, there are other articles available for specific subjects.

Another useful application is **Herald**, which can perform actions, such as sending notifications, based on object changes (such as a Differential revision being created or updated). There is a short user guide available.

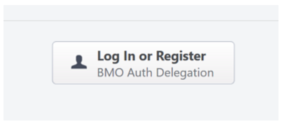
**MozPhab** is a custom command-line tool, moz-phab, which communicates to Phabricator's API, providing several conveniences, including support for submitting series of commits.

### 1.1.2 Creating an Account

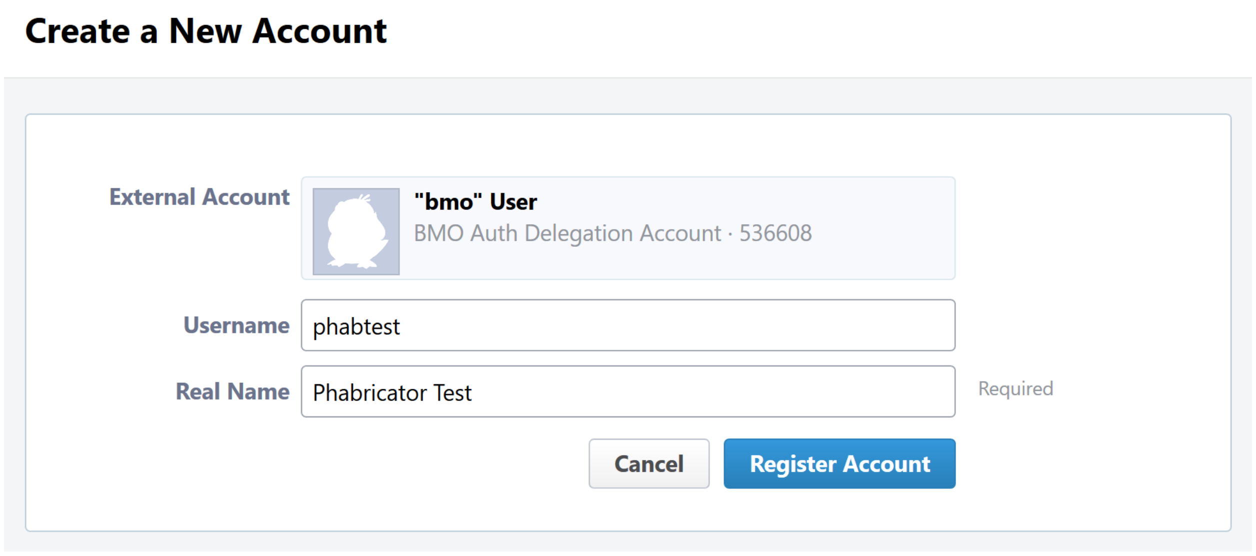
The first step toward submitting a patch via Phabricator is to create an account. Visit our Phabricator instance at https://phabricator.services.mozilla.com/ and click the "Log In" button at the top of any page.



You'll be taken to another page with a single button, which will in turn take you to bugzilla.mozilla.org (BMO) to log in or register a new account.



After you've done so, you'll be redirected back to Phabricator, where you will be prompted to create a new Phabricator account. On this form, the "Real Name" field is taken from your BMO account's real name. If your BMO real name also contains the `:<matrixnick>` convention, that is, a username starting with a colon, it will be extracted and placed into the Phabricator account's username field. Common surrounding punctuation, e.g. parentheses (`()`) and brackets (`[]`) will be stripped out and discarded. If you've used some other way to separate or emphasize your username, you'll have to remove the extraneous characters from the Real Name field manually before clicking "Register Account". The following screenshot shows the account-creation form, with default values, for a BMO user with the real name "Phabricator Test [:phabtest]".



Create a New Account

External Account: "bmo" User — BMO Auth Delegation Account · 536608

Username: phabtest

Real Name: Phabricator Test — Required

Note that the username field is mandatory, so if you didn't have one automatically filled in, you'll have to pick one.

**Important:** The username field is unique. You should pick a clearly identifiable username, particularly if you will be doing code reviews, such as your nick on chat.mozilla.org. If your nick is not available but you think it should be because, for example, you are at least somewhat known in the Mozilla community, please file a bug or let us know in #conduit on chat.mozilla.org.

You now have a Phabricator account set up and can both submit and review patches (along with using the other Phabricator applications).

### 1.1.3 Setting up MozPhab

The preferred and officially supported ways to submit patches are via our custom command-line tool, moz-phab. `moz-phab` currently requires Arcanist and will install it automatically.

Installing the tool depends on your operating system:

- *Windows 10 MozPhab Installation Guide*
- *Linux MozPhab Installation Guide*
- *macOS MozPhab Installation Guide*

The next step is to authenticate MozPhab with our Phabricator installation. From within your project's repository, run the following command:

```
$ moz-phab install-certificate
```

This will prompt you to visit a page on our Phabricator instance, which will generate an API key for you to paste into your terminal. The key is stored in the file `.arcrc` in your home directory.

## 1.1.4 Submitting Patches

### Using moz-phab

moz-phab is a custom command-line tool that improves on Arcanist's limited support for commit series, as well as providing other conveniences, including the parsing of bug IDs and reviewers from commit messages. We recommend using it if you regularly construct stacks of dependent changesets, or even if you regularly review them.

Installation and usage instructions are in the repository's README.md. Note that moz-phab is in active development, with new features and improvements landing regularly. See the current bug list for details.

## 1.1.5 Reviewing Patches

Performing a review involves two steps, both of which are technically optional but will usually be used together:

1. Leaving comments on the diff and/or on the revision generally.
2. Choosing an action to indicate the next step for the author.

Leaving comments is fairly straightforward. For inline diff comments, click on the line number where you want to leave a comment, and enter some text. The text editor is quite rich; you can use many styling and formatting tools. Below the diff is another text-entry box, which can be used for general comments ("Looks good to me", "Here are some suggestions for your overall design", etc.).

At this point you can click the "Submit" button at the bottom; however, this will leave the review open. You might want to do this if you have some preliminary comments and plan to give a more detailed review later. Usually you will want to use the "Add Action…" dropdown to signal a clear intent to the revision author and to communicate what they should do next. These actions include:

- **Accept Revision**: The diff is good as it is and can be landed, or at most requires small changes that do not need re-review.
- **Request Changes**: The diff needs some changes before it can be landed. Specific change requests should be left as comments, as described above.
- **Resign as Reviewer**: This indicates that you are not able to or do not wish to review this change. You will be removed from the reviewers list and hence will not get notifications of updates to the revision. You should explain in a comment why you are resigning (e.g. going on vacation soon, not your area of expertise, etc.) and ideally a substitute reviewer or other action for the author to take, if there are no longer sufficient reviewers on the revision.

## 1.1.6 Other Revision Actions

In addition to the review-related actions mentioned in the *Reviewing Patches* section, there are other common tasks that are accomplished through the actions dropdown. The following are available to revision authors:

- **Request Review**: Asks the reviewer(s) to take another look at the revision. If it is not already, the revision status will be changed to "Needs Review". If a reviewer has previously accepted the revision, their review status will be changed to "Accepted Prior Diff" (the icon for this status is similar to the "Accepted" checkmark, but it is grey instead of green).

- **Plan Changes**: Removes revisions from reviewers' queues, meaning that they will no longer be visible under "Ready to Review" on their "Active Revisions" dashboards, until a new diff is uploaded. The revision will appear under "Ready to Update" on the author's "Active Revisions" dashboard.
- **Abandon Revision**: Indicates that a revision is no longer relevant and should be disregarded.
- **Reopen Revision**: Reopens a revision that has been closed (either manually or automatically) after a revision landed.
- **Reclaim Revision**: Reopens a revision that has been abandoned.

There is another action available specifically to nonauthors:

- **Commandeer Revision**: Allows you to take over a revision by becoming its author. Note that the original author will no longer be able to post updated diffs to the revision. Note: Lando doesn't care who owns the revision on Phabricator, but, it does care about the commit author. When updating someone else's commit, you can use `hg commit --amend --user "Other Person <example@mozilla.com>"` or `git commit --amend --author="Other Person <example@mozilla.com>"` to set the commit author information to the right person.

After selecting an action, you must always hit the "Submit" button below. You may optionally add a comment to indicate your reasoning behind the action or other relevant notes.

### 1.1.7 Landing Patches

For Mercurial repositories, in particular mozilla-central, we highly recommend using *Lando*. See *Getting in Touch* to have repositories added to Phabricator and Lando.

If you cannot use Lando, we highly recommend manually landing to mozilla-inbound without the use of `arc patch` nor `arc land`, both of which add metadata to the commit message which may not be desirable, such as the list of revision subscribers.

If you do not have the commit applied locally and you are landing someone else's patch, you can run `moz-phab patch D<revision id> --nobranch` to apply the commit(s) locally (`--nobranch` ensures the commits are applied to the current branch/head). You can then push the commits as usual.

You could also run `moz-phab patch --apply-to here --nocommit --skip-dependencies D<revision id>` instead. This will apply the diff locally but not commit it, nor will it apply any parents. You can then commit it manually, using the revision title as the first line of the commit message and the Summary field as the body.

### 1.1.8 Our Installation

Mozilla's Phabricator instance is a stock installation, with a small patch applied, and some custom extensions. The patch and extensions are intentionally small in scope and are limited to supporting integration points with bugzilla.mozilla.org ("BMO").

See *Conduit Repositories* for the location of our source code.

### 1.1.9 Applications

Phabricator is actually a suite of many applications, from a code-review tool to wikis to a blogging platform. At Mozilla, we already have existing applications that solve many of these problems. To prevent the re-emergence of the all-too-common problem of having to choose between several tools that are all functionally similar, we have disabled the use of some of these applications.

The default left-side menu in Phabricator lists the most important applications for Mozilla's use case. In addition to Differential and Herald, described above, we support or are trialing several other applications and utilities:

- **Dashboards** allow users to set up custom pages to display useful information, for example assigned reviews. It seems somewhat limited, though, so we'll evaluate how useful it really is.
- **Pholio** is an application for reviewing mock-ups and designs. Mozilla doesn't have a central application for this, so we'd like your input on whether Pholio is useful.
- **Badges**, **macros**, and **tokens**: These are mostly bits of whimsy that might enhance user experience by providing some levity. If they're fun, or at least harmless, we'll leave them; if they become annoying or distracting, we may remove them.

Note that Phabricator also has a post-commit review system called **Audit**. This application is mandatory, that is, it cannot be disabled in a Phabricator installation. However, at the moment Mozilla has no defined engineering processes for post-commit review of Firefox and related code, so we do not recommend its use, at least until such time as a process is deemed necessary and implemented. Audit may, of course, be useful to projects hosted on the Mozilla Phabricator instance outside of Firefox.

### 1.1.10 BMO Integration

Since issue tracking and code review are tightly related, and since BMO is currently the authority for identity and authorization around both issue tracking and code review, including security and other confidential bugs and fixes, our Phabricator instance is integrated with BMO. This integration is intentionally lightweight in order to limit customization of Phabricator, which has both maintenance and opportunity costs. It consists of identity, authorization, links between bugs and revisions, and basic review-status mirroring.

#### Identity

As described in the *Creating an Account* section, the main way to log into Phabricator is via BMO's auth delegation. A user logging into Phabricator is taken to BMO to log in as usual and will be redirected back to Phabricator if the login succeeds. If this is the first time the user has logged into Phabricator, they will be prompted to create an account. New users will also be prompted to enter a separate username, unlike BMO.

#### Authorization

If a bug has one or more security groups applied to it, that is, it has restricted visibility, any Differential revisions associated with it are similarly restricted in visibility. This will initially only apply to Firefox security groups, that is, groups with names matching `*core-security*`. Any revision associated with a bug restricted via other groups, e.g. infra, is visible only to the author and admins. We can add proper support for such groups on request.

#### Links from Differential to BMO

A bug number must be entered when a patch is submitted to Phabricator. This is stored in the revision metadata and provided in the UI as a link to the associated bug on BMO.

#### Links from BMO to Differential

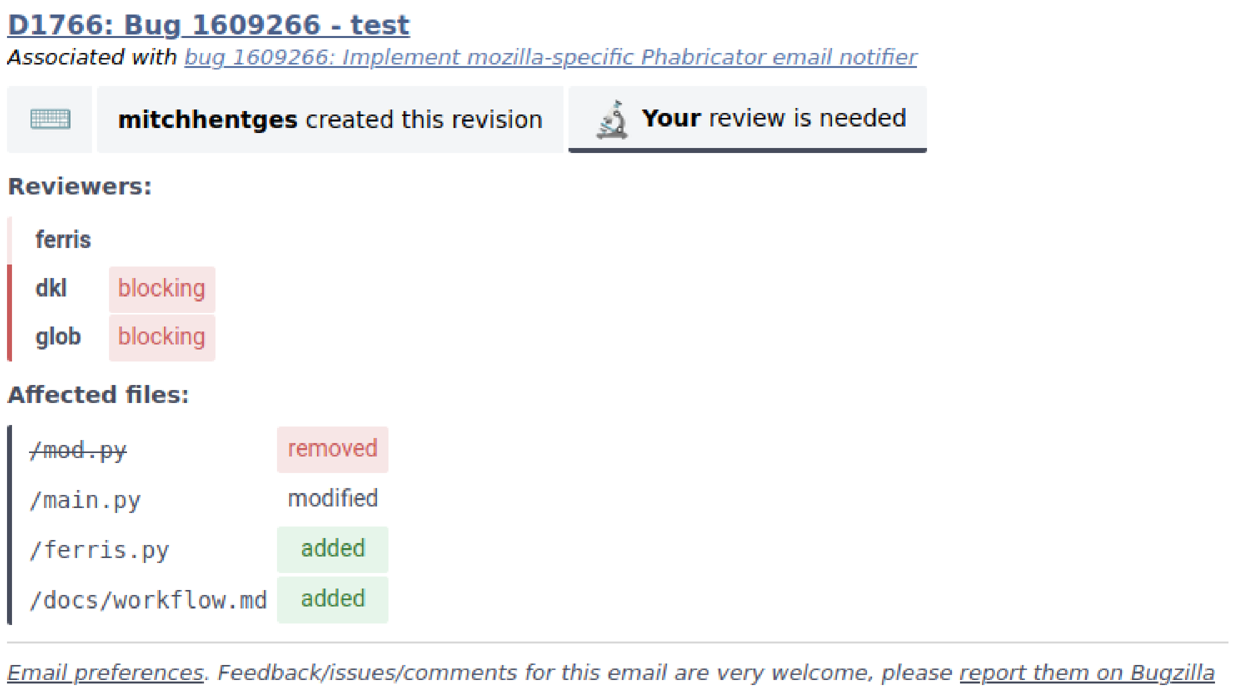
Upon the creation of a new revision in Differential, a stub attachment, containing only the URL of the revision, is added to the associated bug. Based on the attachment type, BMO automatically redirects to Differential if the attachment link is clicked.

#### Review flags

Review flags are not set on Differential stub attachments. The difference in models between the two systems make any such mapping both difficult and potentially misleading, the requisite information is not exposed via Phabricator's Conduit API, and Phacility have informed us that Differential's models may be changing.

We will, however, display some revision metadata in associated bugs; see bug 1489706.

### 1.1.11 Mozilla Phabricator Emails



D1766: Bug 1609266 - test

Associated with bug 1609266: Implement mozilla-specific Phabricator email notifier

**mitchhentges** created this revision

**Your** review is needed

**Reviewers:**

| | |
|---|---|
| ferris | |
| dkl | blocking |
| glob | blocking |

**Affected files:**

| | |
|---|---|
| ~~/mod.py~~ | removed |
| /main.py | modified |
| /ferris.py | added |
| /docs/workflow.md | added |

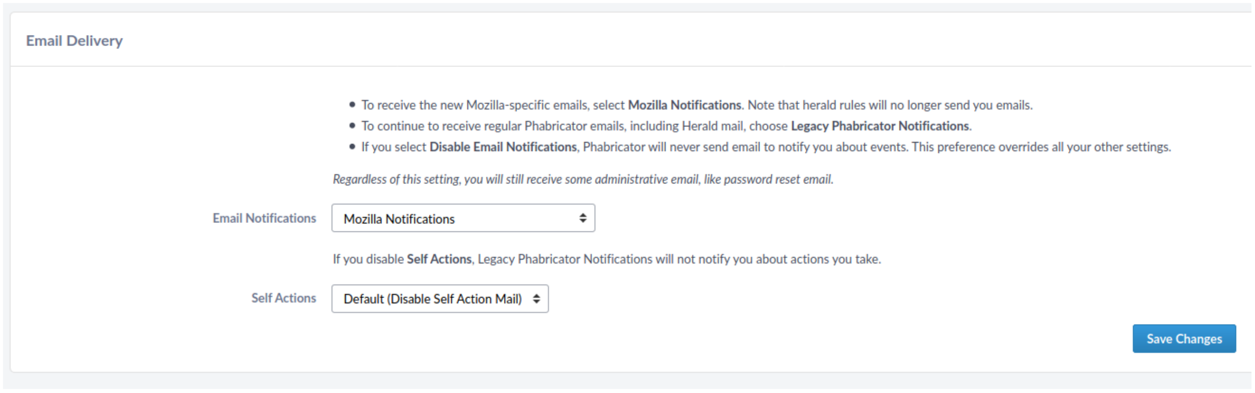
*Email preferences. Feedback/issues/comments for this email are very welcome, please report them on Bugzilla*

Mozilla has an internal email implementation that integrates with Phabricator and has the following benefits:

- There's less noise per email, and fewer emails are sent per event.
- Relevant contextual information is now more clearly available.
- Each email is visually structured to be helpful and fast to read.

These emails are used by default. However, if you'd prefer to opt-out and use the standard Phabricator Emails, you can adjust your settings:

1. Open Phabricator.
2. Click on your avatar in the top right, and click "Settings".
3. On the left panel, click on "Email Delivery".
4. Change the value of the "Email Notifications" dropdown box and click "Save Changes".


Email Delivery

- To receive the new Mozilla-specific emails, select **Mozilla Notifications**. Note that herald rules will no longer send you emails.
- To continue to receive regular Phabricator emails, including Herald mail, choose **Legacy Phabricator Notifications**.
- If you select **Disable Email Notifications**, Phabricator will never send email to notify you about events. This preference overrides all your other settings.

*Regardless of this setting, you will still receive some administrative email, like password reset email.*

Email Notifications: Mozilla Notifications

If you disable **Self Actions**, Legacy Phabricator Notifications will not notify you about actions you take.

Self Actions: Default (Disable Self Action Mail)

**Note:** There are features from the original Phabricator emails that aren't (yet) replicated to the new implementation, including:

- Herald rules to notify via email aren't used.
- X- headers are missing, most notably X-Phabricator-Stamps.

## 1.1.12 Getting in Touch

If you have questions about our Phabricator installation, you can find the team in #conduit on chat.mozilla.org and mozilla.slack.com. Feel free to join if you'd like to help us out!

Issues can be filed in Bugzilla under the Conduit product. These are the main components:

- Administration: For requests to add new repositories and similar tasks.
- Documentation: For issues with these and other project docs.
- Phabricator: For issues with Phabricator, including our extensions (authentication, BMO integration, etc.) and with the upstream Phabricator product. For bugs in our extensions, we may move them to bugzilla.mozilla.org :: Extensions: PhabBugz depending on where the problem exists in our code. Also note that, as discussed in *BMO Integration*, we are strictly limiting customizations to our instance. We may, however, work with upstream in fixing important issues.
- Lando: For issues with Lando, the UI/API for requesting and monitoring commit landings.
- Transplant: For issues with Transplant, the backend service which takes landing requests from Lando and pushes them to the relevant repository.
- General: Feel free to file issues here if you aren't sure where they should go. We'll triage them as needed.

## 1.1.13 Frequently Asked Questions

See the FAQ on the wiki for answers to common questions and issues. The FAQ is on a wiki to make it easier to maintain; please feel free to update it if you come across other frequently asked questions!

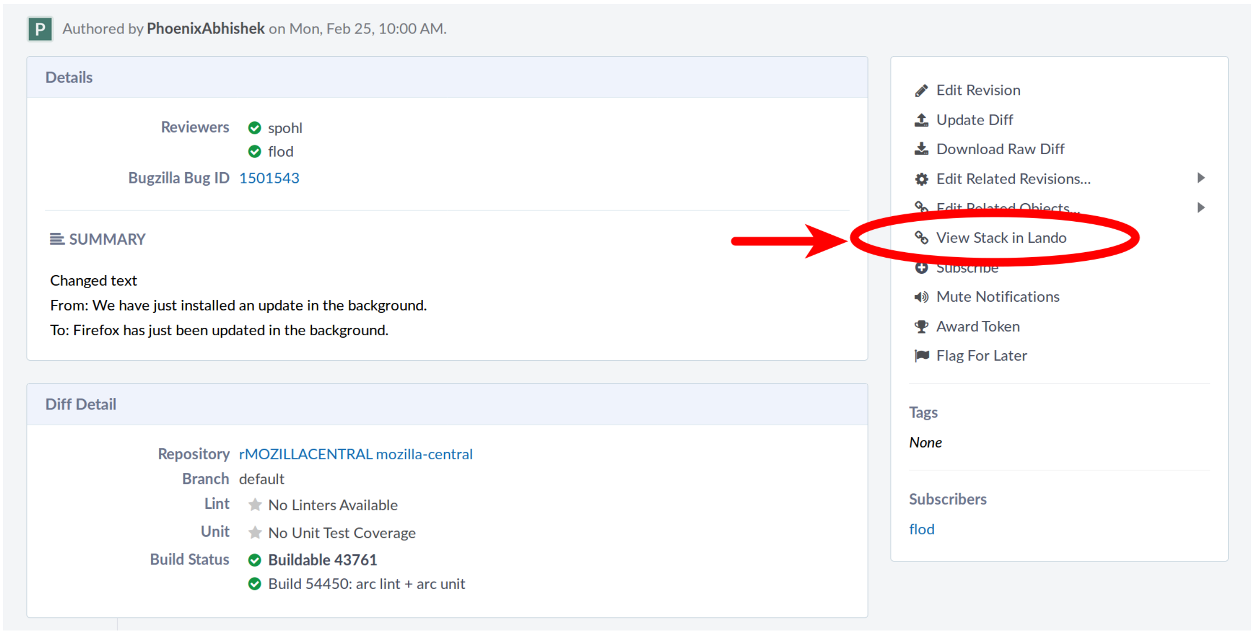
## 1.2 Lando User Guide

### 1.2.1 About Lando

Lando is Mozilla's new automatic code-landing service. It is loosely integrated with our Phabricator instance. Its purpose is to easily commit Phabricator revisions to their destination repository.

### 1.2.2 Viewing a Revision

All revisions in Phabricator have a "View Stack in Lando" link. This link will let you land the current revision and all dependent ancestor revisions (see *Landing a Stack of Revisions* for more details). If the revision has been accepted (approved), the link will be active; otherwise, it will be greyed out. A link looks like the following:


Authored by **PhoenixAbhishek** on Mon, Feb 25, 10:00 AM.

Details

| | |
|---|---|
| Reviewers | spohl<br>flod |
| Bugzilla Bug ID | 1501543 |

SUMMARY

Changed text
From: We have just installed an update in the background.
To: Firefox has just been updated in the background.

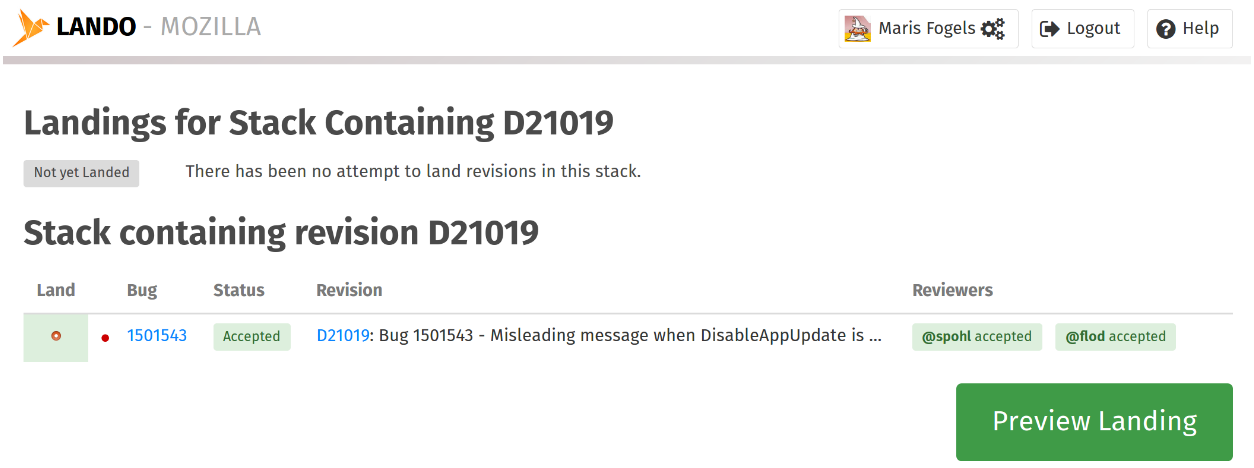
Diff Detail

| | |
|---|---|
| Repository | rMOZILLACENTRAL mozilla-central |
| Branch | default |
| Lint | No Linters Available |
| Unit | No Unit Test Coverage |
| Build Status | **Buildable 43761**<br>Build 54450: arc lint + arc unit |

Tags
*None*

Subscribers
flod

Clicking this link will take you to the Lando page for that revision:



Landings for Stack Containing D21019

Not yet Landed — There has been no attempt to land revisions in this stack.

Stack containing revision D21019

| Land | Bug | Status | Revision | Reviewers |
|---|---|---|---|---|
| ○ | ● 1501543 | Accepted | D21019: Bug 1501543 - Misleading message when DisableAppUpdate is ... | @spohl accepted @flod accepted |

You can also go directly to a revision's Lando page by specifying the revision ID in the Lando URL: `https://lando.services.mozilla.com/D<rev number>/`.

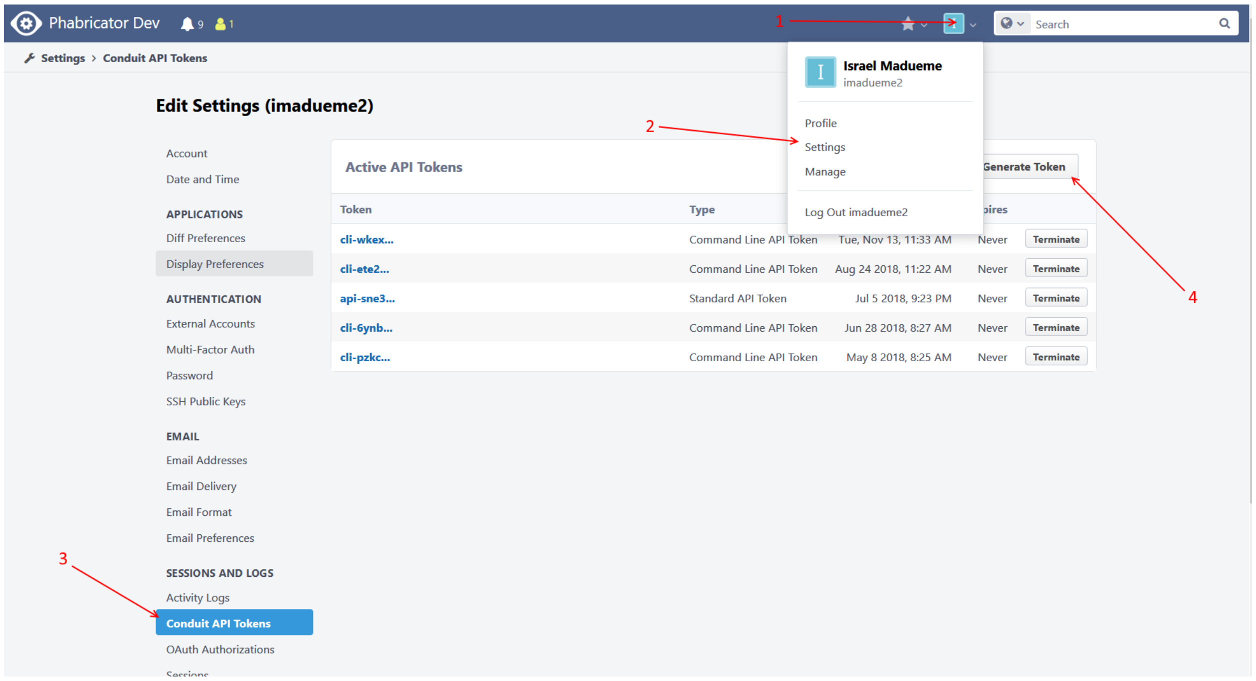
A relevant set of metadata about the revision is presented, including the revision ID, the author, the status of reviews, and the commit message. There is also a timeline of previous landing attempts, if any.

### 1.2.3 Viewing a Confidential Revision

Lando can view and land secure Phabricator revisions that you have access to.

If you try to view a secure revision on Lando without proper setup, you will see an error page telling you that the revision could not be found, or it is locked down. To view it, you'll need to provide Lando with a Phabricator API token for your account.

First, generate a Phabricator API token. Go to Settings -> Conduit API Tokens -> Generate Token. Keep this token secret and use it only for Lando. You cannot reuse a "cli-" token; please generate a new API token (these start with "api-").

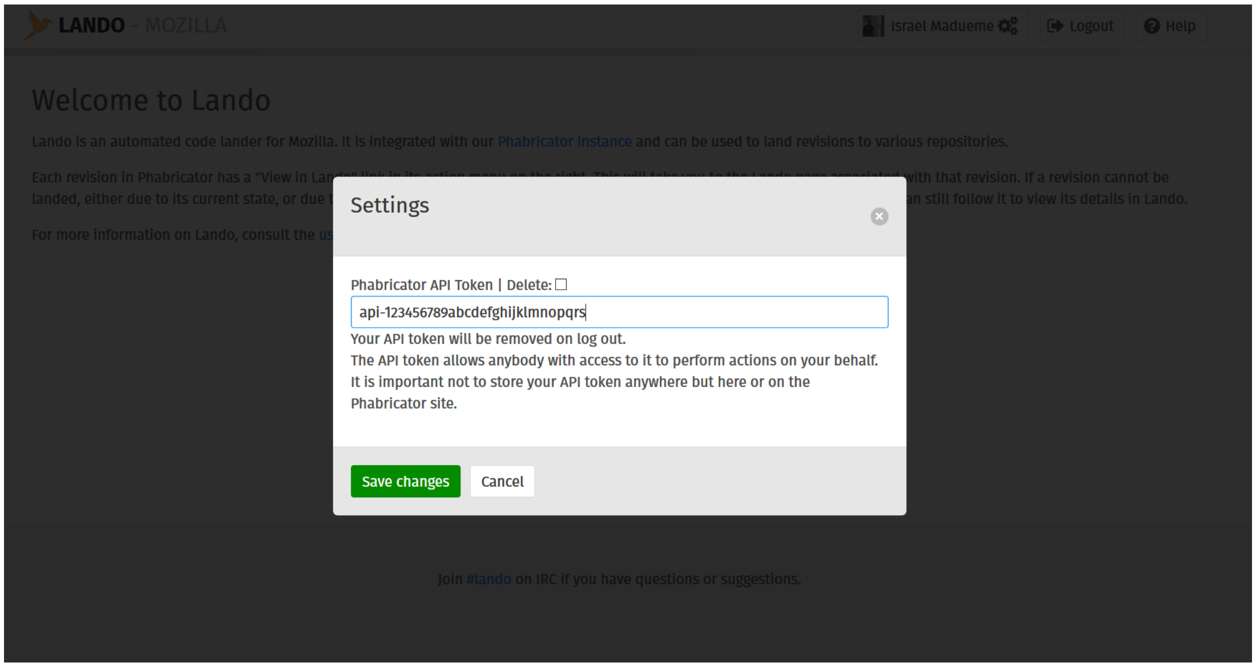


Edit Settings (imadueme2)

Active API Tokens

| Token | Type | | pires | |
|---|---|---|---|---|
| cli-wkex... | Command Line API Token | Tue, Nov 13, 11:33 AM | Never | Terminate |
| cli-ete2... | Command Line API Token | Aug 24 2018, 11:22 AM | Never | Terminate |
| api-sne3... | Standard API Token | Jul 5 2018, 9:23 PM | Never | Terminate |
| cli-6ynb... | Command Line API Token | Jun 28 2018, 8:27 AM | Never | Terminate |
| cli-pzkc... | Command Line API Token | May 8 2018, 8:25 AM | Never | Terminate |

Next go to Lando and click the settings icon next to your name, near the logout button. Once clicked, a settings modal will be displayed and you can enter the Phabricator API token you generated here.



Welcome to Lando

Lando is an automated code lander for Mozilla. It is integrated with our Phabricator instance and can be used to land revisions to various repositories.

Each revision in Phabricator has a "View in Lan … with that revision. If a revision cannot be landed, either due to its current state, or due t … an still follow it to view its details in Lando.

For more information on Lando, consult the us

Settings

Phabricator API Token | Delete: ☐

api-123456789abcdefghijklmnopqrs

Your API token will be removed on log out.
The API token allows anybody with access to it to perform actions on your behalf.
It is important not to store your API token anywhere but here or on the Phabricator site.

Join #lando on IRC if you have questions or suggestions.

Click save and then reload the secure revision page on Lando again.

To delete an API token, check the “Delete” checkbox on the settings modal and click save. Or you can simply log out.

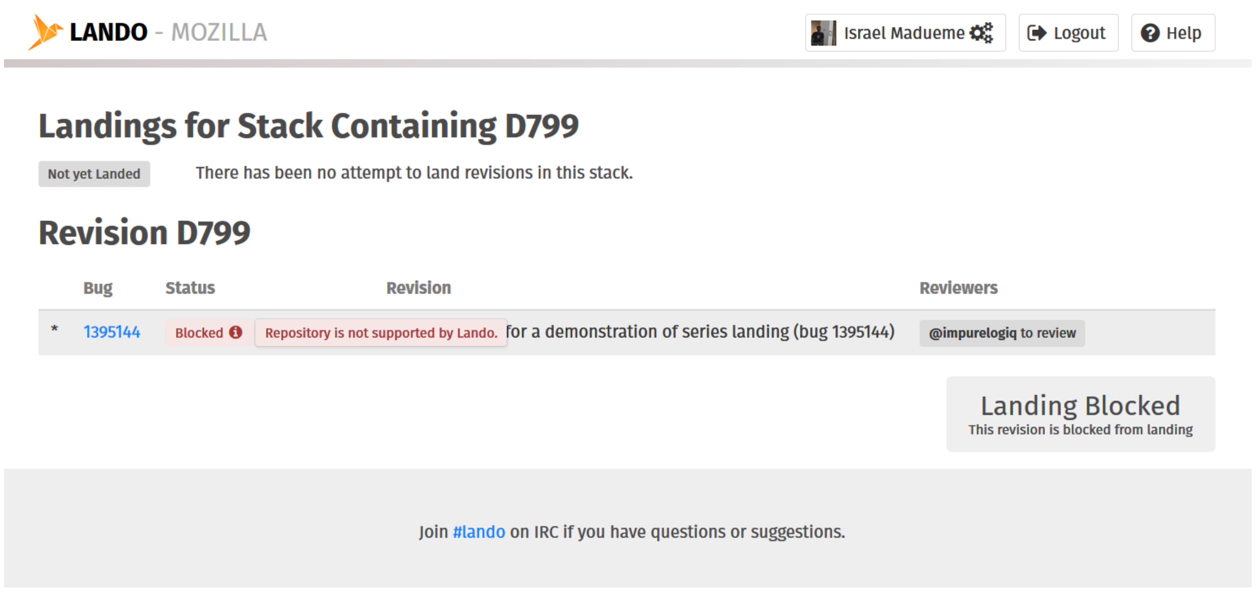
### 1.2.4 Landing a Revision

You must be logged in to initiate a landing. Logins are handled by Auth0 and follow the same flow as many other Mozilla systems.

In addition to logging in, the following conditions must be true:

- The revision must be associated with a repository in Phabricator.
- A destination repository must be configured in Lando. Ask in the #lando channel on irc.mozilla.org for help if you get this error.
- You must have the required SCM permissions to land to the destination repo (e.g. `scm_level_3` for `mozilla-central`). See the FAQ for help with this error.
- Your permissions for the repo must be active (i.e. not expired).
- A landing for this revision must not already be in progress.
- You are not attempting to land an old diff of a revision that was updated.

When you preview the landing, if any of the above are not true, you will be shown an appropriate error message at the bottom of the page, and the “Land” button will be disabled.



Landings for Stack Containing D799

Not yet Landed There has been no attempt to land revisions in this stack.

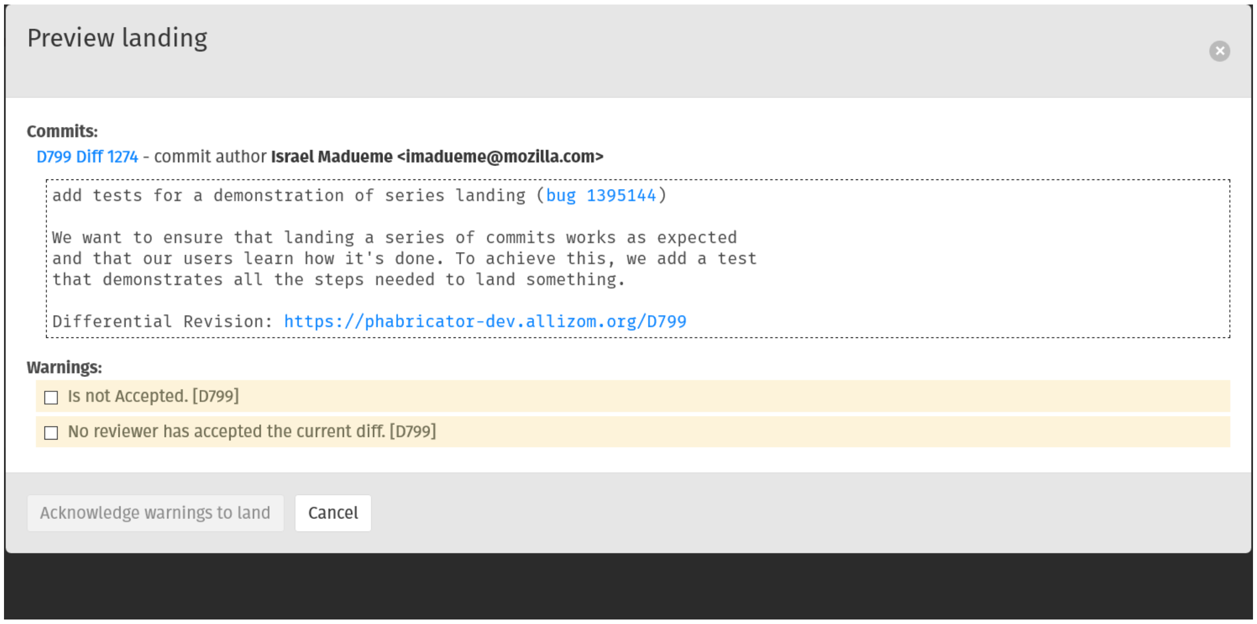
Revision D799

| | Bug | Status | Revision | Reviewers |
|---|---|---|---|---|
| * | 1395144 | Blocked Repository is not supported by Lando. | for a demonstration of series landing (bug 1395144) | @impurelogiq to review |

Landing Blocked
This revision is blocked from landing

Join #lando on IRC if you have questions or suggestions.

If there are no landing blockers, you can click the "Preview Landing" button to see a preview before landing. Here you can verify the final commit message as well as acknowledge any warnings.



Preview landing

Commits:
D799 Diff 1274 - commit author **Israel Madueme <imadueme@mozilla.com>**

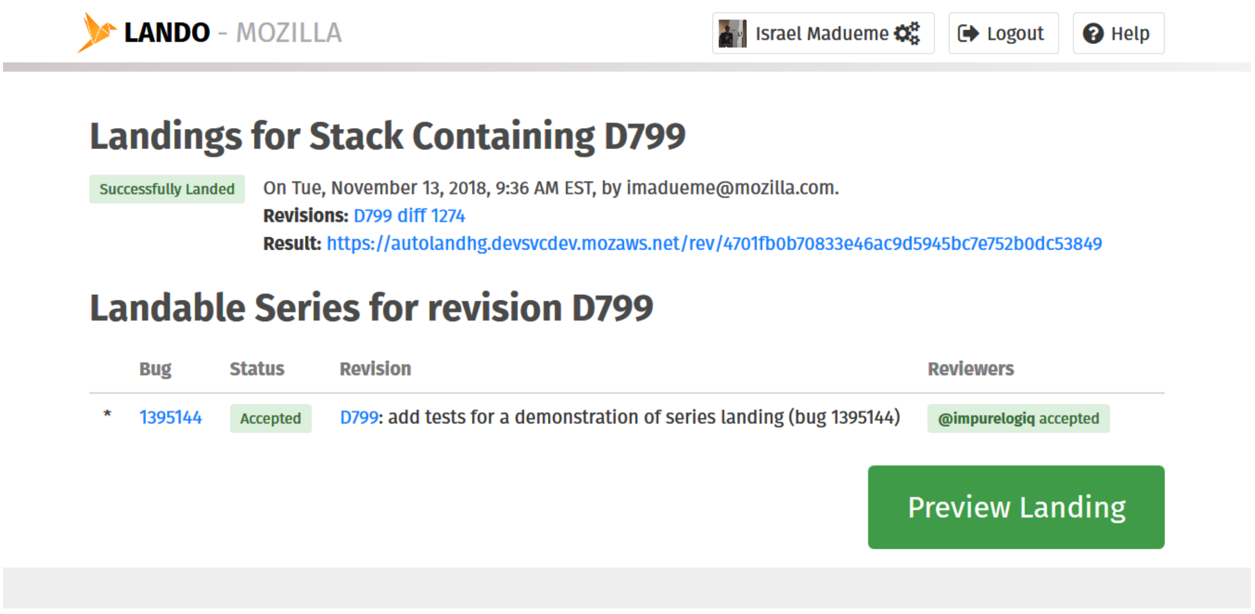
```
add tests for a demonstration of series landing (bug 1395144)

We want to ensure that landing a series of commits works as expected
and that our users learn how it's done. To achieve this, we add a test
that demonstrates all the steps needed to land something.

Differential Revision: https://phabricator-dev.allizom.org/D799
```

Warnings:
- ☐ Is not Accepted. [D799]
- ☐ No reviewer has accepted the current diff. [D799]

Warnings are things which Lando suspects could be a problem but will allow you to determine if they really are, e.g., the revision has already been landed previously, or the revision has not been accepted by a blocking reviewer. You can acknowledge these warnings by checking the checkboxes next to them if you believe the landing should proceed regardless.

Once "Land" is clicked, a request will be queued. Generally, this will execute quickly, but if there are a lot of pending landings, or if the trees are closed, it may take longer. The landing request will stay in the queue until it is executed.

Once the landing is executed, the timeline will be updated with the results:



Landings for Stack Containing D799

Successfully Landed — On Tue, November 13, 2018, 9:36 AM EST, by imadueme@mozilla.com.
Revisions: D799 diff 1274
Result: https://autolandhg.devsvcdev.mozaws.net/rev/4701fb0b70833e46ac9d5945bc7e752b0dc53849

Landable Series for revision D799

| | Bug | Status | Revision | Reviewers |
|---|---|---|---|---|
| * | 1395144 | Accepted | D799: add tests for a demonstration of series landing (bug 1395144) | @impurelogiq accepted |

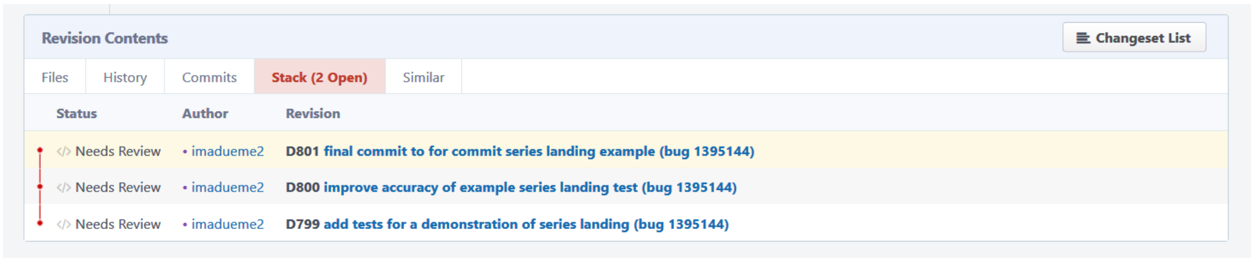
**Note:** Lando pages do not currently automatically refresh; you will have to reload them manually to see updates. There are a couple bugs open to fix this, such as *bug 1460364 <https://bugzilla.mozilla.org/show_bug.cgi?id=1460364>*.

If the landing failed, an error message will be displayed. This error may represent a problem with the revision, e.g. a merge conflict. In this case, the revision will have to be updated and resubmitted. If it appears to be an error with Lando itself (or related services), please let us know in #lando on IRC or file a bug.

### 1.2.5 Landing a Stack of Revisions

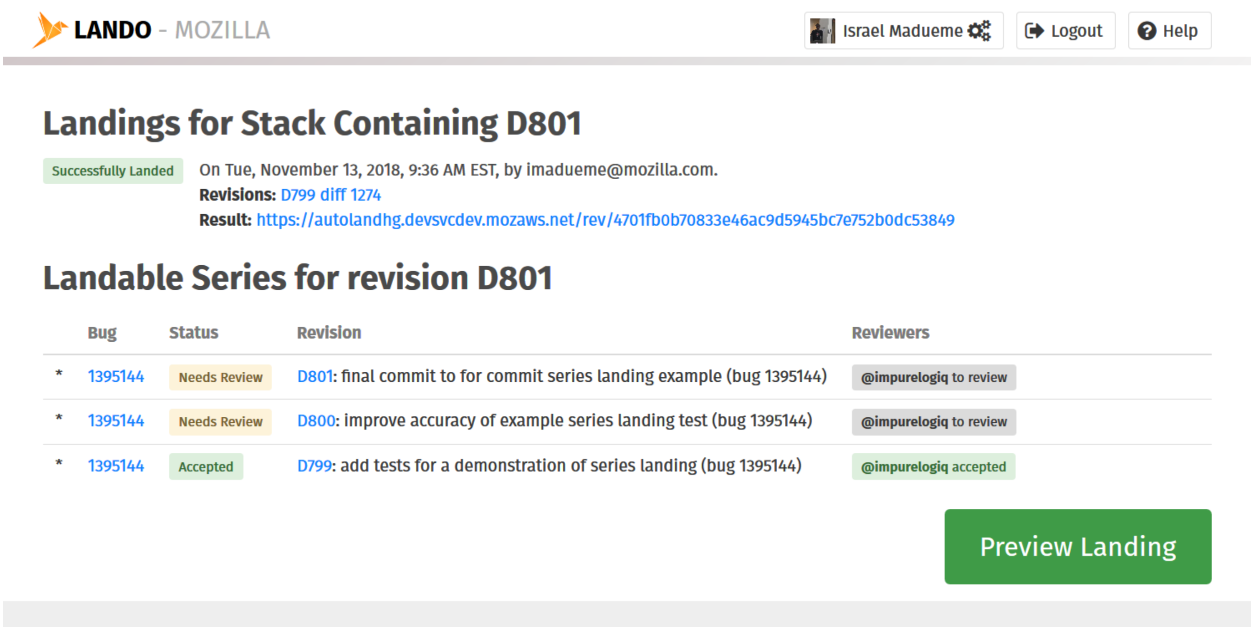
Lando can also land a stack of revisions at once.

Ensure that the dependency chain is properly set in Phabricator:



| Status | Author | Revision |
|---|---|---|
| Needs Review | imadueme2 | D801 final commit to for commit series landing example (bug 1395144) |
| Needs Review | imadueme2 | D800 improve accuracy of example series landing test (bug 1395144) |
| Needs Review | imadueme2 | D799 add tests for a demonstration of series landing (bug 1395144) |

Load the child-most commit of the stack you want to land. For example, in this case we would load D799 to only land D799, or load D800 to land D799 and D800, or load D801 to land all 3 revisions. Once loaded in Phabricator, click the “View Stack in Lando” link in the sidebar.



LANDO - MOZILLA

Israel Madueme | Logout | Help

Landings for Stack Containing D801

Successfully Landed — On Tue, November 13, 2018, 9:36 AM EST, by imadueme@mozilla.com.
**Revisions:** D799 diff 1274
**Result:** https://autolandhg.devsvcdev.mozaws.net/rev/4701fb0b70833e46ac9d5945bc7e752b0dc53849

Landable Series for revision D801

| | Bug | Status | Revision | Reviewers |
|---|---|---|---|---|
| * | 1395144 | Needs Review | D801: final commit to for commit series landing example (bug 1395144) | @impurelogiq to review |
| * | 1395144 | Needs Review | D800: improve accuracy of example series landing test (bug 1395144) | @impurelogiq to review |
| * | 1395144 | Accepted | D799: add tests for a demonstration of series landing (bug 1395144) | @impurelogiq accepted |

The resulting page will show you information for each revision in the stack. The timeline can show partial landings of a subset of the stack, as demonstrated in the screenshot above (D799 was landed, but D800 and D801 have not yet been landed). The same principles for landing a single revision apply for landing a stack; you will get a warning or confirmation when previewing the landing and a success or failure result after landing.

**Note:** If there are any confidential revisions in the stack that you want to view, you must have permission to view all of them. Make sure to set a Phabricator API token on Lando if this is the case.

### 1.2.6 Frequently Asked Questions

See the FAQ on the wiki for answers to common questions and issues. The FAQ is on a wiki to make it easier to maintain; please feel free to update it if you come across other frequently asked questions!

## 1.3 Phabricator Workflow Walk-through

**IMPORTANT:** Make sure you have *set up Phabricator* and *set up MozPhab* before proceeding!

While some developers use bookmarks/etc to track changes, for this this guide we will use just repository heads: no bookmarks or labels. This essentially means "just start coding off tip and commit". The `hg wip` alias provides a view of the repository that allows for keeping track of the work.

(If you want to dig deeper into the "to label or not to label" discussion, see this document)

### 1.3.1 Landing a commit to mozilla-central

For this example we will craft one change for review. When we get feedback that changes are required we will amend our commit in-place.

We'll use:

- One commit

- One review request per commit
- `hg amend` to add fix-ups to our commit
- `moz-phab` to request code reviews

### Fixing the code

Let's start with a clean checkout.

```
$ hg wip
@    460880 tip Merge inbound to mozilla-central.  a=merge
|\
~ ~

$ vim dom/audiochannel/AudioChannelAgent.cpp
# hack hack

$ hg status
M dom/audiochannel/AudioChannelAgent.cpp
```

When we write the commit message we should follow the Firefox source tree's common commit message format. We will include a Bug ID and list of reviewers. See the committing rules for more information about proper commit message formatting.

```
$ hg commit
Bug 1445923 - WebAudio: Remove b2g dead code r?sylvestre


Removed dead code from ...
```

- You are allowed to skip the bug number and reviewer names if you don't have them yet, but the Lando automated landing system, used later in this walk-through, will insist that you add them before the code can be landed.

### Requesting a Review

Before we request a review we should check for changes upstream.

```
$ hg pull --rebase
```

The `moz-phab` command will take care of creating a code review for us. It will automatically link the review to the BMO bug as well as notifying the reviewers.

```
$ moz-phab
Submitting 1 commit:
(New) 460880:e376ef5bf453 Bug 1445923 - WebAudio: Remove b2g dead code r?sylvestre
Submit to Phabricator (YES/No/Always)?

...

Completed
(D55555) 460880:e376ef5bf453 Bug 1445923 - WebAudio: Remove b2g dead code r?sylvestre
-> https://phabricator.services.mozilla.com/D55555
```

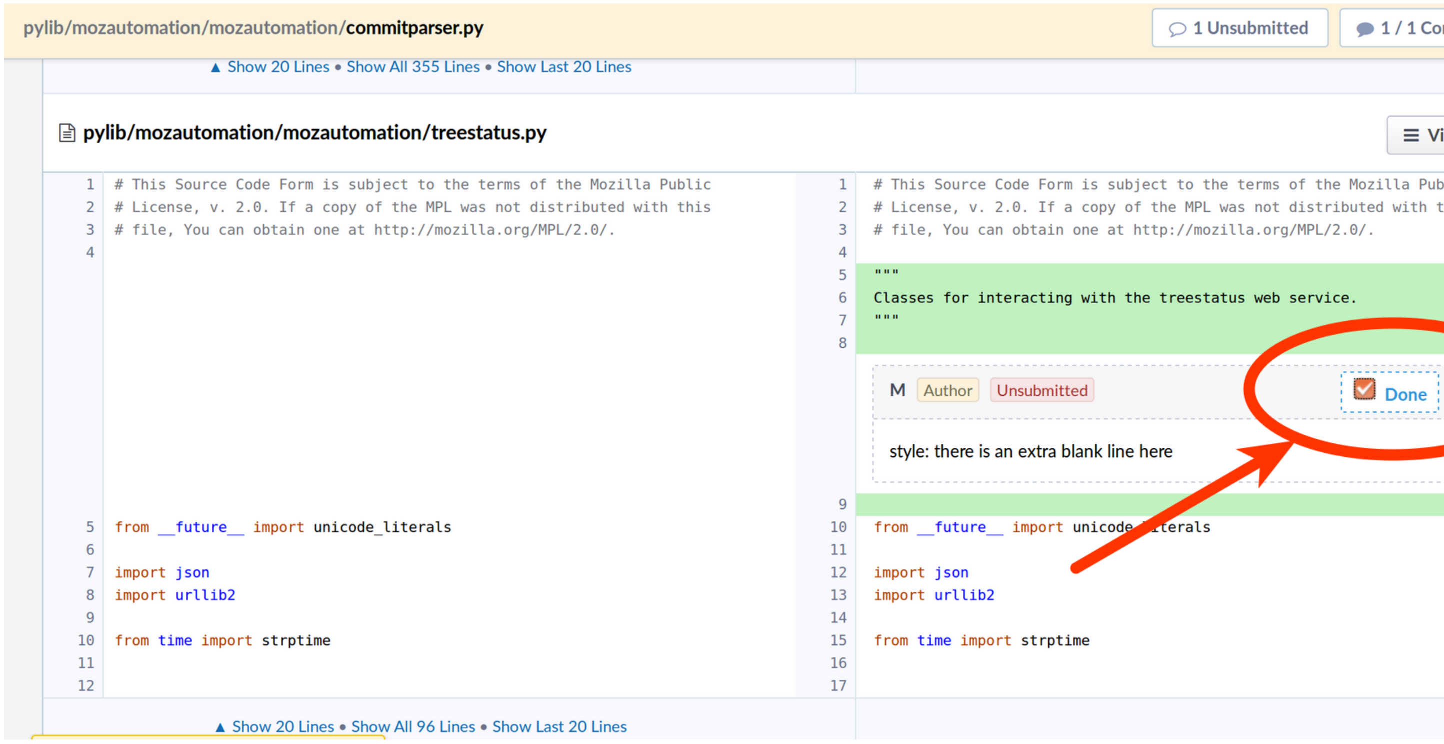
### Addressing feedback

When it's time to address feedback we use `hg amend`.

- `hg commit --amend` also works, and allows you to update the commit description while amending the commit.

```
$ hg wip
@  460881 tip Bug 1445923 - WebAudio: Remove b2g dead code r?sylvestre
o    460880 Merge inbound to mozilla-central.  a=merge
|\
~ ~


$ hg checkout 460881
$ vim dom/audiochannel/AudioChannelAgent.cpp
# fixup fixup

$ hg amend
```

Check off the Done item in the Phabricator UI.


pylib/mozautomation/mozautomation/commitparser.py

pylib/mozautomation/mozautomation/treestatus.py

```
# This Source Code Form is subject to the terms of the Mozilla Public
# License, v. 2.0. If a copy of the MPL was not distributed with this
# file, You can obtain one at http://mozilla.org/MPL/2.0/.

from __future__ import unicode_literals

import json
import urllib2

from time import strptime
```

```
# This Source Code Form is subject to the terms of the Mozilla Pub
# License, v. 2.0. If a copy of the MPL was not distributed with t
# file, You can obtain one at http://mozilla.org/MPL/2.0/.

"""
Classes for interacting with the treestatus web service.
"""

from __future__ import unicode_literals

import json
import urllib2

from time import strptime
```

style: there is an extra blank line here

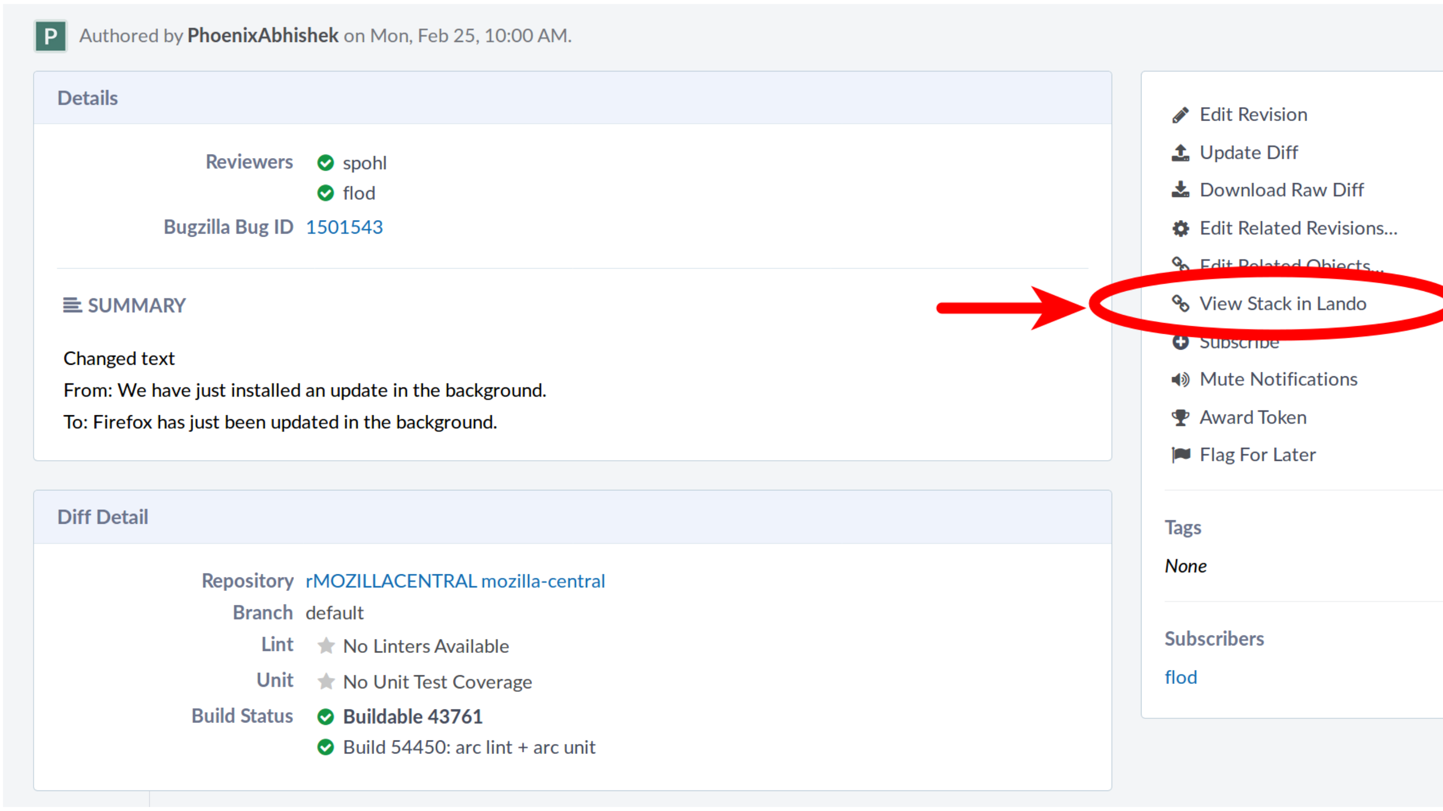
Now run `moz-phab` a second time. Phabricator will automatically submit your Done items in the UI and notify your reviewers that you have made updates.

```
$ moz-phab
```

### Landing the changes

Everything looks good: the reviewers have approved our changes. Let's land our changes.

On your revision page in Phabricator click the "View Stack in Lando" link in the right-hand menu:



Authored by PhoenixAbhishek on Mon, Feb 25, 10:00 AM.

Details

Reviewers spohl, flod

Bugzilla Bug ID 1501543

SUMMARY

Changed text
From: We have just installed an update in the background.
To: Firefox has just been updated in the background.

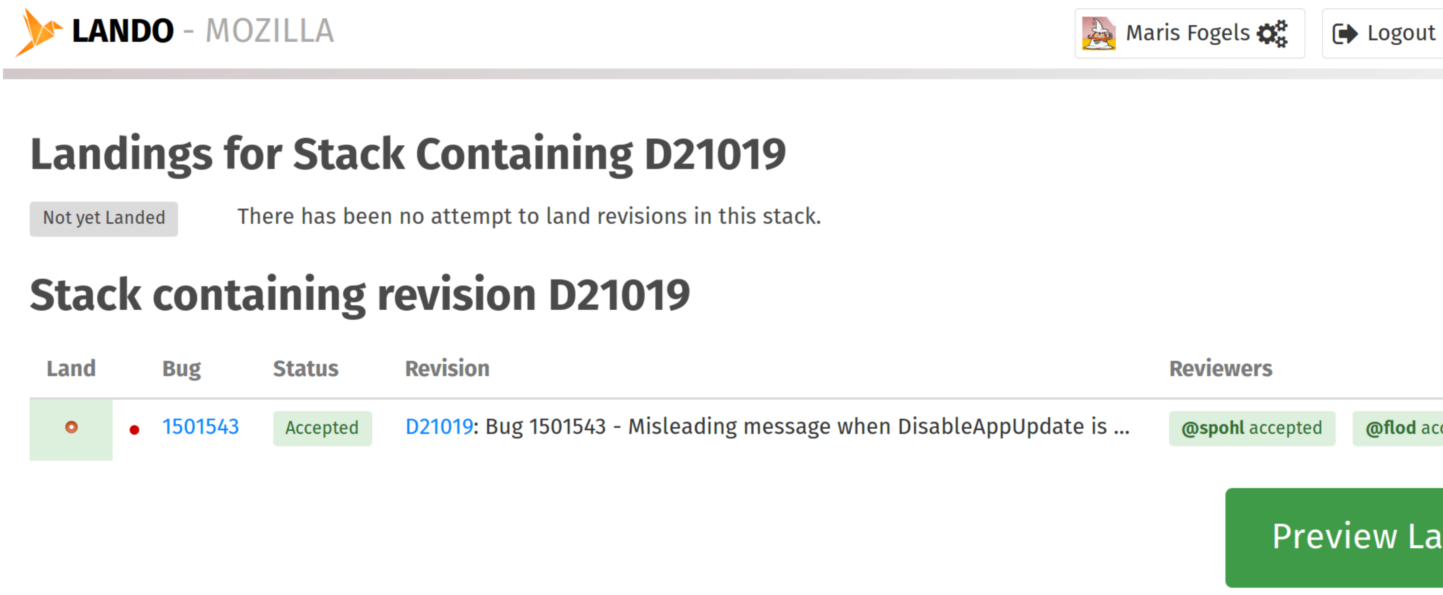
Diff Detail

Repository rMOZILLACENTRAL mozilla-central
Branch default
Lint No Linters Available
Unit No Unit Test Coverage
Build Status Buildable 43761
Build 54450: arc lint + arc unit

Tags None

Subscribers flod

You will be taken to the Lando revision overview page. Press "Preview Landing" and give the change one last review: double-check the commit message, etc., before hitting the "Land" button.



Landings for Stack Containing D21019

Not yet Landed There has been no attempt to land revisions in this stack.

Stack containing revision D21019

| Land | Bug | Status | Revision | Reviewers |
|---|---|---|---|---|
| ● | 1501543 | Accepted | D21019: Bug 1501543 - Misleading message when DisableAppUpdate is ... | @spohl accepted @flod ac |

Hit the "Land" button and Lando will automatically land your changes.

### 1.3.2 Where to go from here

`moz-phab` has other features including the ability to update and land entire stacks of related commits. Check out the *Submitting Patches* section for more information.

## 1.4 Windows 10 MozPhab Installation Guide

Here are step-by-step instructions to getting MozPhab working on Windows 10 in a MozillaBuild environment. They should work for Git Bash and PowerShell as well. For the sake of this documentation we will use the terminal provided by running the `C:\mozilla-build\start-shell.bat`.

1. Install the Microsoft Visual C++ 2017 Redistributable (x64) if you don't already have it. You can see if you have it through Settings -> Apps -> App & features. It is available at https://support.microsoft.com/en-ca/help/2977003/the-latest-supported-visual-c-downloads (vc_redist.x64.exe).
2. If you don't have Git already installed (note that it is not currently packaged with MozillaBuild), you'll need to grab it from https://git-scm.com/download/win and install it.
3. Run `pip3 install MozPhab`.
4. Add `moz-phab` and `git` to the `$PATH` variable. If you use MSYS (including MozillaBuild) exclusively, you can add this to `~/.bash_profile`:

   ```
   export PATH=$PATH:/c/Program\ Files/Git/bin:/c/mozilla-build/Python3/Scripts
   ```

5. Ensure running `moz-phab` works:

   ```
   $ moz-phab -h
   ```

## 1.5 Linux MozPhab Installation Guide

MozPhab can be installed from PyPI.

This requires Git and Python 3.6 or higher with `pip3`.

### 1.5.1 Ensure the requirements are installed

Verify that pip3 and git are installed and working when run from the command line:

```
$ git --version
git version 2.20.1
$ pip3 --version
pip 18.1 from /usr/lib/python3/dist-packages/pip (python 3.7)
```

The versions you have do not need match the above.

If either are missing use your distro's package manager to install. For example if you use Ubuntu:

```
$ sudo apt-get install git python3-pip
```

### 1.5.2 Install MozPhab

1. Call `pip3 install --user MozPhab`

   This will install `moz-phab` into your home directory, under `~/.local/bin`.

2. If `moz-phab` has not been found, add your `~/.local/bin` directory to the `PATH` variable. Running this command in terminal will change the `PATH` for current session. Add it to your profile file (`~/.bashrc` or equivalent) to keep the `$PATH` changed

```
$ export PATH=~/.local/bin:$PATH
```

3. Ensure running `moz-phab` works:

```
$ moz-phab -h
```

## 1.6 macOS MozPhab Installation Guide

MozPhab can be installed from PyPI

This requires Git and Python 3.6 or higher with *pip*.

### 1.6.1 Install Python3 using Homebrew

1. Follow the instructions on https://brew.sh/ if Homebrew is not installed.
2. Install Python 3 and Pip with one command:

   ```
   $ brew install python
   ```

### 1.6.2 Install MozPhab

1. Call `pip3 install MozPhab`
2. Ensure running `moz-phab` works:

```
$ moz-phab -h
```

## 1.7 Arcanist User Guide

### 1.7.1 Arcanist

Mozilla's Engineering Workflow team has created a custom command-line tool, *moz-phab*, which has better support for submitting, updating, and applying series of commits. It also has conveniences for parsing bug IDs and reviewers from commit-message summaries or specifying them as command-line options.

This document is an older guide for Arcanist, the official upstream command-line interface to Phabricator. It may still be useful for those who do not frequently submit patches for review.

This guide is a quick overview of how to submit, update, and apply patches with Arcanist. Note that we have written our own guide for installing Arcanist on Windows 10. The official Phabricator documentation also has an Arcanist Quick Start guide, a larger Arcanist User Guide, and a specific guide to "arc diff".

There are a few ways to use Arcanist and Differential. We'll cover two common use cases: fix-up commits, which is somewhat similar to GitHub's process, and amended commits, which is similar to MozReview's model.

### 1.7.2 Submitting and Updating Patches

#### The Initial Patch

Submitting the initial patch is the same in both processes. First, commit a change. Here's an example:

```
$ echo "Test" > PHABTEST
$ hg add PHABTEST && hg commit -m "Add test file."
```

Then create a revision in Differential:

```
$ arc diff
```

You'll be taken to an editor to add extra details. Here is an example of input to `arc diff` from a real revision (https://phabricator.services.mozilla.com/D1298):

```
heartbeat: check all backing services in heartbeat (Bug 1442911).

Summary:

Before this change /__heartbeat__ was only checking for connectivity
to Phabricator. We now also check the database, transplant, redis
(cache), s3, and auth0. The /__heartbeat__ endpoint now returns a more
useful response, indicating which services are unhealthy when a 500
response is sent.

Test Plan:

invoke test passes, new tests added. Ran lando api locally with
services in both healthy and unhealthy states and observed the response
from /__heartbeat__

Reviewers: glob, imadueme

Subscribers:

Bug #: 1442911
```

Your commit message will be used to create the revision title and summary. The other fields are optional. If they are given in a similar format in the commit message, the fields will be prepopulated here as well. This includes `Bug`; omitting a bug ID will result in the revision not being associated with a bug, and thus it will automatically be public. If set, the field must contain a valid BMO bug number. Note that mozilla-central commit policy currently requires a bug number in the commit message under most circumstances.

Unfortunately, a limitation of Phabricator currently prevents us from seeding this field with a bug ID from the commit message (at least from the first line, where bug IDs are usually mentioned in mozilla-central changesets). Note that we have worked around these restrictions in *moz-phab*.

You may want to add a reviewer, which should be a Phabricator username (e.g. `mcote`). You can also add one or more subscribers, who will be notified of updates to the revision. Again, we cannot parse these out of the commit summary with Arcanist, but *moz-phab* supports this.

Note that the commits to be included in this revision are present in the comment at the bottom of the text. You can use this to double-check that you are sending the correct commits to Phabricator.

After you exit the editor, the revision should be created. Here's example output from a different revision on our development instance:

```
Created a new Differential revision:
        Revision URI: https://mozphab.dev.mozaws.net/D29

Included changes:
  A       PHABTEST
```

If you visit the revision at the provided URL, you will see that it is labelled "Needs Review", which is the default state of a newly created revision. It will also be marked "Public", unless the bug ID you entered is a confidential bug to which you have access. For convenience, an attachment is created on the bug containing just the URL to the new revision, with the description being the revision's title. Finally, you will also see a few actions on the revision, which are automatically performed by our BMO-integration code. For more on Phabricator-BMO integration, see *BMO Integration* in the *Phabricator User Guide*.

### Fix-Up Commits

After your patch has been reviewed, you may have to update your patch and get another round of reviews. As mentioned, there are two ways to do this in Differential.

The "fix-up commit" model involves creating a new commit containing the updates. This is similar to GitHub's standard process. You will end up with a series of commits that should be "squashed" into a single commit before landing, since the fix-up commits are not useful history once a change has landed.

Here's an example that adds another line to our test file from above:

```
$ echo "Update" >> PHABTEST
$ hg commit -m "Update patch."
```

Submitting the change to Differential is the same command:

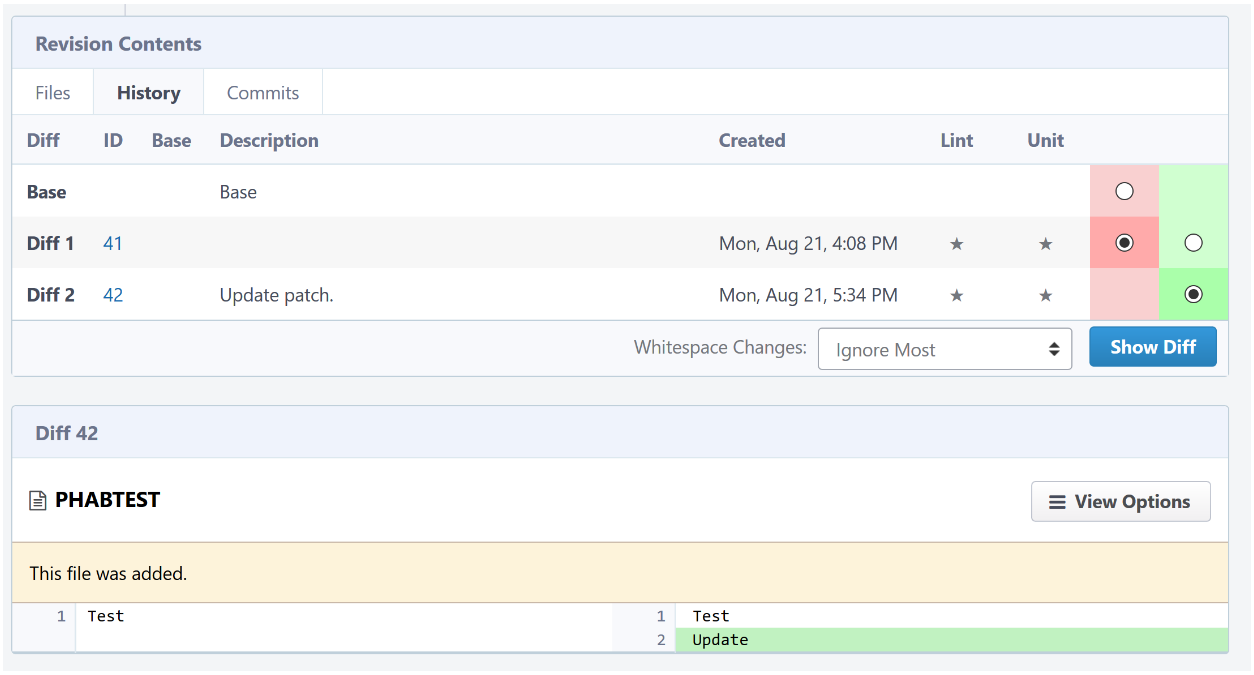
```
$ arc diff
```

Your editor will again be opened, but this time the format is much simpler. You just need to provide a change summary, which again is automatically seeded from your commit message. Arcanist should also have determined which revision to update. If for some reason it was not able to, you can use the `--update` option to specify a revision ID.

After the update has been submitted, you will see output similar to this:

```
Updated an existing Differential revision:
        Revision URI: https://mozphab.dev.mozaws.net/D29

Included changes:
  A       PHABTEST
```

Going to the revision's URL will show the change in the activity log. There will also be new entries in the "History" and "Commits" tabs in the "Revision Contents" table. You can use the History tab to switch between various diff views: the current patch, the patch at a particular point in history, and the changes between different commits, i.e., an interdiff. Here are the changes between the first and second commit ("Diff 1" and "Diff 2" in Phabricator language):



Revision Contents

Files | History | Commits

| Diff | ID | Base | Description | Created | Lint | Unit | | |
|---|---|---|---|---|---|---|---|---|
| Base | | | Base | | | | ○ | |
| Diff 1 | 41 | | | Mon, Aug 21, 4:08 PM | ★ | ★ | ● | ○ |
| Diff 2 | 42 | | Update patch. | Mon, Aug 21, 5:34 PM | ★ | ★ | | ● |

Whitespace Changes: Ignore Most

Diff 42

PHABTEST

This file was added.

```
1 Test                        1 Test
                              2 Update
```

## Amended Commits

The other method for updating patches is to amend the commits in place. This is similar to MozReview's standard process.

Starting from the end of the above section, *The Initial Patch*, rather than creating a new commit, we amend the existing commit, like so:

```
$ echo "Update" >> PHABTEST
$ hg commit --amend
```

After running `arc diff`, an editor is again opened for a change summary, although this time there is no new commit message to use, so we must enter one manually. Once the update is processed, the revision looks very similar to the revision with fix-up commits, except the "Commits" tab of the "Revision Contents" table has only a single entry. The "History" tab, however, is identical to the fix-up commits scenario, with "Diff 1" and "Diff 2" entries, and the same ability to see the different patches and differences between them.

## Series of Commits

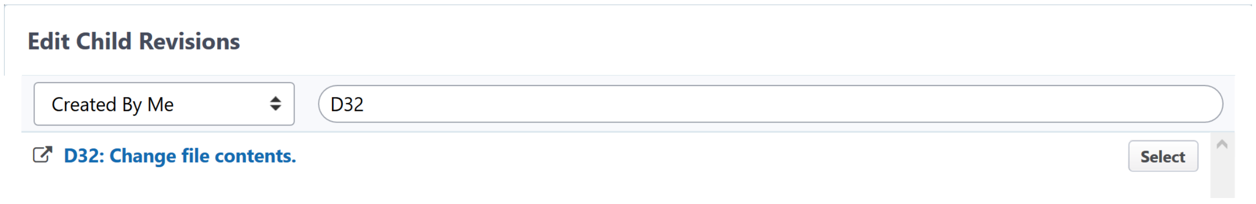
It is possible to chain a series of revisions together in Differential, although it is currently a manual process. This feature can be used to represent a stack of commits to split up a complicated patch, which is a good practice to make testing and reviewing easier.

To use this pattern, you will need to specify the exact commit you want to send to Differential, since the default is to send all your draft commits to a single revision, i.e., the *Fix-Up Commits* method, which is not what we want here. To send only the currently checked-out Mercurial commit, run the following:

```
$ arc diff .^
```

To set the parent-child relationship, you can use the UI or put a directive into the child's commit message. To use the UI, go to your first commit, choose "Edit Related Revisions..." from the right-hand menu, then "Edit Child

Revisions". Your child revision may be suggested, or you can enter an ID into the search box, including the `D` to denote a differential revision, e.g. `D32`:

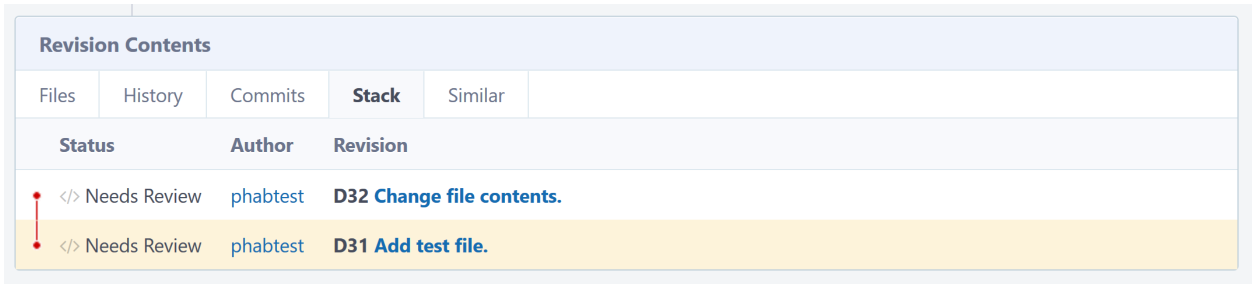


Edit Child Revisions

Created By Me | D32

D32: Change file contents. Select

Select the appropriate revision and click "Save Child Revisions". The "Revision Contents" table will now have a new tab, "Stack", which shows the current stack of revisions:



Revision Contents

| Status | Author | Revision |
|---|---|---|
| Needs Review | phabtest | D32 Change file contents. |
| Needs Review | phabtest | D31 Add test file. |

You can also add `Depends on D<revision ID>` to the child's commit message, replacing `<revision ID>` with the ID of the parent revision. (This needs to be its own paragraph, separated by a blank line.) The relationship will be created when `arc diff` is run.

Unfortunately there is not currently a way to see a combined diff of all the stacked commits together without applying the commits locally. Also, when you update any commits, you'll need to run `arc diff .^` for each child commit as well. This was the primary purpose of writing *moz-phab*.

See also this blog post on working with commit series in Phabricator.

We will be working on a solution to automate the submission and updating of commit series.

### 1.7.3 Applying Patches

You can pull down the commits from any revision you have access with this command:

```
$ arc patch <revision id>
```

It is helpful to understand that `arc patch`, by default, will not attempt to patch the revision on top of your current working set. Instead, it applies the changes on top of the same parent commit the author used and creates a new commit and a new branch (git) or bookmark (hg). If it cannot find the same parent commit in your local repo then it will warn you and give you the option to apply it on top of the current working set. If you wish to test a revision on top of your current working set use `arc patch --nobranch`.

If you have a stack of revisions (see above section *Series of Commits*), the commits from all previous revisions will be applied as well. Note that if you are pulling down a stack of revisions but have a different commit currently checked out than was used as the parent of the first commit, you will get warnings like this:

```
This diff is against commit a237e16c2f716f55a22d53279f3914a231ae4051, but
the commit is nowhere in the working copy. Try to apply it against the
current working copy state? (.) [Y/n]
```

This is because the first commit now has a different parent and hence a different SHA. You can avoid this problem by updating to the parent of the first commit before running `arc patch`.

## 1.8 Migrating from MozReview to Phabricator

### 1.8.1 Key differences between MozReview and Phabricator

**MozReview**

- Most of the interfacing with the review tool is done by the `hg` command.
- Commits are updated in-place with `hg amend`, histedit, etc.
- When landing, the commit messages appear the same as you authored them, with the list of reviewers rewritten to reflect approvals.
- Patches in a series all land at the same time.

**Phabricator**

- Most of the interfacing with the review tool is done by the `arc` command.
- You choose how commits are updated. Phabricator supports both MozReview-style amended commits or GitHub-style fixup commits.
- When landing, the commit message is copied from your Phabricator review summary.
- Patches in a series can land at different times.

### 1.8.2 Common Questions

#### What will happen to my active code reviews?

MozReview will stop accepting new code reviews 2 weeks before it is taken offline. During that period you will have time to finish your in-flight reviews.

Authors must manually migrate their reviews to Phabricator if they still have unfinished reviews at the end of the 2 week shutdown notice period. Join us in **#phabricator** on IRC or Slack if you need assistance with this.

#### What will happen to the patches MozReview contains today?

All of the patches in MozReview will be preserved. All of the links to MozReview patches in Bugzilla will continue to work.

The links in Bugzilla will change during the migration. Instead of linking to MozReview reviews, Bugzilla will link to raw patches in unified-diff format. A snapshot of the review repository is available for advanced users. See *How do I apply patches from old code reviews to my source tree after MozReview has shut down?*.

### I like editing a series of commits in-place before landing them. Can I still do that?

Yes. Phabricator supports the Mercurial workflow we have today, where review feedback is amended to the commit you originally submitted for reviews. See the *Mozilla Phabricator User Guide* for details.

The amended-commit workflow is easy to use with Git, too.

### I like the GitHub workflow of fixup commits that are squashed during landing. Can I do that?

Yes. Phabricator has support for the GitHub-style squash-on-merge workflow. See our *Mozilla Phabricator User Guide* for details.

### How do I run Try builds?

Try builds need to be run from the command line. Adding the ability to trigger Try builds from the Lando UI is a priority for us.

### Can I chain related reviews together in Phabricator like I did in MozReview?

Yes. Related reviews in Phabricator, called “stacks”, are also more flexible than reviews in MozReview.

In MozReview, all reviews in a series of commits must be approved before any one review can land. All of the commits in the series land together.

In Phabricator, code reviews can be stacked however the author wishes, with a change depending on one or more other reviews. You can land the reviews that are lowest in the stack while still taking feedback on the reviews higher up. This is great for, say, landing a refactoring or bugfix before landing a feature that builds on top of it.

Our *Mozilla Phabricator User Guide* has instructions for using stacks.

### Can I use ‘hg push’ to create and update reviews?

Not at this time. We have created prototypes to see how this could work. This may be developed in the future.

### How do I apply patches from old code reviews to my source tree after MozReview has shut down?

1. Visit the bug associated with the review you want to reconstruct.
2. Click on the “MozReview Requests” section.
3. Click on the link for the review you want to get patches for. You will be taken to the MozReview patch archive for that revision.
4. Copy the link to the diff you want to download.
5. Import the diff with `hg import --no-commit --exact https://mozreview-archive.s3-website.us-east-2.amazonaws.com/12345/r12345-diff[SOME-VERSION].patch`

There is also a snapshot of the review repo available for more advanced users.

## 1.9 MozPhab Data Collection

Mozphab is collecting user data to make data-driven decisions.

Data collection is automatically enabled for Mozilla employees, with the ability to opt out. Non-employees can opt in or opt out at any time.

You are able to manually opt out of telemetry by changing the *telemetry.enabled* setting in the MozPhab's configuration file *<HOME_DIR>/.moz-phab-config*.

```
[telemetry]
enabled = True
```

For information about the data collected see: https://github.com/mozilla-conduit/review/blob/master/TELEMETRY.md.

## 1.10 Contributing to Conduit

### 1.10.1 General Guidelines

We appreciate any contributions to the applications that make up the Conduit system. There are some common tools, processes, and guidelines that most of the Conduit apps follow. Each application also has specific setup instructions; see the individual *Conduit Repositories*.

We've created the Suite project to simplify the integration of all containers within the developer's machine.

#### Bugs

We use Bugzilla for bug tracking, even for those repos hosted on GitHub. Although Bugzilla is a heavier-weight tool than GitHub issues, it provides us with better tracking and coordination, along with support for security bugs and other confidential information. Most bugs belong in the Conduit product, with the exception of the PhabBugz BMO extension.

#### Code Review

We use our Phabricator instance for code review, even for those repos hosted on GitHub. It is tied to our Bugzilla instance for both identity and issue management. See the *Mozilla Phabricator User Guide* for more information.

**Note:** Please test your code before creating a revision in Phabricator.

#### Technologies

We have a wiki page detailing the tech stack we have chosen. Although it is, and will always be, a work in progress, it lists both our chosen technologies and reasons behind them. New Conduit applications are expected to follow these guidelines unless there is good reason to deviate, which should be discussed with the Conduit development team.

### 1.10.2 Conduit Repositories

The source code for most Conduit applications lives on GitHub.

- lando-api has the main logic of the custom automatic-landing system that has been integrated with our Phabricator instance.
- lando-ui is a separate web application that is the main graphical user interface to lando-api.
- PhabBugz, the BMO extension that maps BMO security policy to Phabriator and other such integrations, is a directory within the monolithic BMO repo. As with the rest of BMO, it is written in Perl.
- phabricator-extensions contains our Conduit extensions to Phabricator, which mainly relate to integration with BMO. As with Phabricator itself, these are written in PHP.
- mozphab contains the deployment scripts and configuration for our Phabricator installation. The files are a mixture of Python, PHP, shell scripts, and config files.
- bmo-extensions is a docker-based development environment for our Bugzilla-Phabricator integration pieces.
- conduit-docs contains the source for the docs you are reading now.
- autoland-transplant is a tool that automatically lands patches from one Mercurial tree to another.
- suite allows you to connect and run all above services in a local development environment.

# CHAPTER 2

# Indices and tables

- search