Release Procedure

Become a contributor

You need to be added as a contributor to edit the wiki. But don't worry! Just email any Mailing List or grab us on IRC and let us know your user name.

Introduction

Any Apache CouchDB committer is free to make a source release, but they are usually made by the release team.

If you'd like to help out with making a release, lets us know on the couchdb-dev mailing list.

Step 1 of any release process is: every time you go through the release process, update this wiki page to reflect any changes you make! Do this every release, so nothing gets out of date.

Admin Resources

You may want to grab a copy of the admin resources:

```
git clone http://git-wip-us.apache.org/repos/asf/couchdb-admin.git
```

Any path in this document that references the `couchdb-admin` directory should be modified to point to this Git clone. Some of these scripts take actions on your behalf, such accessing `people.apache.org` or checking files into Subversion. Because of this, you must to read and understand the source code for every script that you use. It would be prudent to examine the recent Git history before you start the release.

You are responsible for the actions these scripts take. Many have not yet been updated for the 2.x series of releases. At the moment, the release team is not using the admin repo scripts at all.

Getting CouchDB Ready to Release

Preparing the Code

Before starting the release process for CouchDB, the following sanity checks should be made:

- Ensure you are running the *lowest* support version of Erlang locally. If you run with too new a version of Erlang, you may end up building a release asset that is unbuildable on older Erlang VMs (because the shipped `rebar` binary will be incompatible).
- Checkout the code locally and run a clean build and test:

```
# If you don't have CouchDB downloaded yet:
$ git clone https://github.com/apache/couchdb && cd couchdb
#
# or, if you already have CouchDB cloned:
$ cd couchdb && git pull && git checkout master && git reset --hard && git clean -ffdx
#
# Then:
$ ./configure && make check
```

- Verify that any sub-repositories (fauxton, docs, etc.) have had their dependency versions updated in the top-level `rebar.config.script` file.
    - You may need to consult with the committers of changes to each sub-repo if these changes are ready to be incorporated into the next release build.
    - If necessary, bump the dependency specification, commit the change, and rebuild and re-run the tests as above.
- Identify issues that should be resolved before cutting a release:
    - Check the last few Travis CI and Jenkins runs (master or release branch only) for any failed builds.
        - Any test failures should be logged as issues in GitHub issues.
        - Issues with the CI environment should not be considered release blocking, but should be resolved in a timely fashion.
    - Check for any open issues that appear critical or high priority, especially those tagged with the security label in GitHub.
    - Create a milestone in GitHub Issues, "x.x.x", representing the proposed version number for the release.
    - Mark all of the issues found above with the newly created milestone "x.x.x".
- Ensure that the right **version number** is referenced in the code. That version is stored in two files**: `version.mk`** and **`rel/reltool.config`**.
- Work with the development team to resolve these issues before proceeding. If changes are necessary, you are encouraged to issue pull requests and get the changes merged.
- If possible, check that the automated packages are valid. Check the last full CI run in jenkins or build them yourself on a Linux machine with Docker installed:

```
$ git clone https://github.com/apache/couchdb-pkg
$ git clone https://github.com/apache/couchdb && cd couchdb && ./configure --without-proper && make dist
$ cd ../couchdb-pkg
$ ./make-releases.sh ../couchdb/apache-couchdb-*.tar.gz
```

Preparing the Docs

Meanwhile, make the following updates to the documentation:

- Update the `README.rst` file with important information.
- Update the `README-DEV.rst` file with important information.
- In the `couchdb-documentation` repository, update the `src/whatsnew/X.X.rst` file with important information.

    - Compile a list of bugs fixed and improvements that have been fixed in this release.

        - See `src/whatsnew/2.2.rst` for an example.
        - Be sure to use the special RST syntax to link to JIRA or GH issues as appropriate.
        - This can be done with a GitHub search of the form: https://github.com/apache/couchdb/compare/2.1.0...master
        - From the command line, the following approach works well:

          ```
          git show-branch --sha1-name master X.X.X | more
          ```

            - use this to find common ancestor (very last thing in the output)
            - use this to make sure everything on X.X.X has been pulled fwd to master
            - use this to eliminate anything that "looks the same" on X.X.X branch and master
                - cannot use sha-1s because they will be different on each branch
                - research if there is a way to auto-eliminate cherry picked stuff?

          For each remaining commit on master:

            1. use GH web UI to find the commit's related PRs or issues and write up something nice
            2. sort by GH issue/PR number
            3. any issues that merge without a useful GH/PR number should go either at the very top or bottom of the list, depending on importance of issue
            4. do not explicitly list out things like Jenkins/Travis/rebar.config changes unless they are significant in some fashion

          Now do the same thing as above, but for documentation and fauxton. Pull out only the big-ticket items.

    - If the release has already been forked onto a maintenance branch, be sure to look at the `master` branch for any changes that won't make this release. These may be candidates for Known Issues in the documentation.
    - Add a breaking changes section to this file if necessary.
    - Add any deprecated endpoints to this file if necessary.
    - Ensure any deprecated endpoints are marked as such in the HTTP API documentation section as well.

- Ensure that any new CVEs identified are summarized in the documentation. CVEs are described in files under `couchdb-documentation/src/cve/*.rst`.

Preparing the Community

Generate a release proposal from the discussion release template. Edit as appropriate. Send it to the dev@couchdb.apache.org mailing list.

Preparing to Sign a Release or Release Candidate

You will need a GPG key pair to sign the release. If you do not have one already, see the Useful Resources section for more help from Apache resources. It is generally better if your key is also signed by other people, especially by members of the CouchDB PMC.

Your public key **must** be added to: http://www.apache.org/dist/couchdb/KEYS

You **should** also upload your key to a public key server for good measure.

Make sure your key is registered to gpg:

```
gpg --list-keys
```

You may need to set the GPG_TTY first:

```
export GPG_TTY=`tty`
```

If your signing key is not the default, add the `--default-key=<hash>` option to the gpg signing command(s) below. You can change the default in your gpg.conf file with the `default-key` directive.

Cutting a Release Candidate

If the proposal meets no objection, you can continue:

Preparing the Candidate

- Confirm that no one else is building the RC at the same time as you are. 🙂
- Update your local git clone to the latest version of the code.
- Check the `rebar.config.script` file and look at all of the dependency version numbers. If any are raw git hashes (not x.x.x-style version numbers), tag those sub-repositories first. Then, update the `rebar.config.script` with these version numbers and commit the change. No dependencies should be referenced by a raw git hash.
- Run one final `make check` as in the first section of this process, **being sure to run the `git clean` and `git reset` commands**.
- If this is the first RC for a new major or minor release (3.0.0, 2.3.0, etc. but not 2.3.1) then create a branch:

```
$ git checkout -b #.#.x
git push origin
```

- Tag this version with `x.x.x-RC#` where # represents which release candidate this is (1, 2, 3...):

```
$ git tag -a x.x.x-RC# -m "Version x.x.x-RC#" --sign
$ git push origin x.x.x-RC#
```

- Build the release candidate tarball (substitute the correct number for the # below):

```
```

```
$ cd couchdb
$ git reset --hard && git clean -ffdx
$ ./configure --without-proper
$ make dist
```

Checking the Candidate

This wiki has separate instructions on testing a release candidate.

Assume that you cannot trust:

- The source code the archive was built from.
- The host operating system the archive was built on.

An attacker may have compromised either. Accordingly, you should subject the release candidate to a number of your own tests.

Some ideas:

- Verify the contents of the generated files.
    - Spot check a few of the files by hand.
    - Prepare your own reference candidate and compare the files.
- Audit the types of file contained within the archive.
- Run a virus scanner on the archive.

This part of the process is left to your discretion.

Publishing the Candidate

Once you are satisfied with the release candidate, you must sign it, publish it, and call the vote.

- Sign the release and generate the appropriate checksum files:

```
$ gpg -b --armor apache-couchdb-x.x.x-RC#.tar.gz
$ sha256sum apache-couchdb-x.x.x-RC#.tar.gz > apache-couchdb-x.x.x-RC#.tar.gz.sha256
# If on Windows, fix the line endings in the checksum files!
```

- Add the files to the official Apache distribution site:

```
$ svn co https://dist.apache.org/repos/dist/dev/couchdb/source
$ mkdir -p source/x.x.x/rc.#
$ cp apache-couchdb-x.x.x-RC#.tar.gz* source/x.x.x/rc.#
#
# if this is the first RC for a new release:
$ svn add source/x.x.x
# otherwise:
$ svn add source/x.x.x/rc.#
$ svn propset svn:mime-type text/plain source/x.x.x/rc.#/*.gz.asc source/x.x.x/rc.#/*.gz.sha*
#
# Add a meaningful comment, like "Adding CouchDB x.x.x-RC1 to dev tree"
$ svn ci -m "Adding CouchDB x.x.x-RC# to dev tree"
```

- Download and customise the release_2_vote.txt announcement template from the couchdb-admin repo. Then, send it to the dev@couchdb.apache.org mailing list.
- Publicise the RC vote through Twitter, the weekly newsletter, IRC/Slack, etc.

Remember!

Publishing the candidate is not the same as voting on it. You should now follow the test procedure and vote on the candidate yourself.

Preparing the macOS or Windows Binary Packages

Note: As of 3.1.0, we're not doing Apache binary packages for macOS or Windows. Neighbourhoodie have taken over this work.

Everyone is welcome to prepare binary packages for the release. Binary packages are **never** required for release candidates. All binaries must be prepared from the release artifacts. After voting on the release, you should prepare your binaries from the release artifacts you just voted on. If you can't build these yourself, ask for help on IRC or the mailing list.

- Create the directory in Subversion:

```
svn mkdir --parents \
https://dist.apache.org/repos/dist/dev/couchdb/binary/{mac|win}/x.x.x/rc.# \
-m 'Add x.x.x-rc.# {mac|win} dir'
```

- Check out this directory:

```
svn co https://dist.apache.org/repos/dist/dev/couchdb/binary/PLATFORM/VERSION/rc.CANDIDATE
```

- Add your files to this directory, then check in your files:

```
svn ci -m 'Add x.x.x-rc.# {mac|win} files'
```

Go to https://dist.apache.org/repos/dist/dev/couchdb/binary/ and check that your changes are visible. If you are not a committer, please submit your binaries to the list and request a review. If a committer feels comfortable with your submission, they will move your binaries into Subversion. It is the committer's individual responsibility to review your submission and determine that your binaries are suitable for distribution. Once the binaries are available in Subversion, the committer who checked them in should post a note to the list. The community should be invited to download and test the binaries before distribution.

Preparing the Debian/Ubuntu/RHEL/CentOS package repos for testing

If you have shell access to `couchdb-vm2.apache.org`, you can capture the latest auto-built Debian/Ubuntu/RHEL/CentOS package repositories from Jenkins and move it aside to a x.x.x-RC# directory for testing. The repos all live under the `/var/www/html` path.

> **Remember!**
>
> *Binaries are never voted on. They are provided by individuals for the convenience of the community.*

Wrapping Up the Vote

You need to wait a minimum of 72 hours. A successful vote needs a majority approval and at least three +1 binding votes. A binding vote is a vote cast by a member of the PMC. Releases may not be vetoed, but any -1 votes should be taken seriously. The release manager has the power to abort a vote at any point and for any reason. Please try to make it a good one though! If the vote fails, go back to the *Preparing the Candidate* section, and increment the candidate number.

When the vote passes, download and customize the release_3_result.txt announcement template from the couchdb-admin repo. Then, send it to the dev@couchdb.apache.org mailing list.

Making the Release

If the release was approved by formal vote, you can proceed to make the release happen.

Publish the code

Tag the candidate with the x.x.x version number:

```
$ cd couchdb && git checkout x.x.x-RC#
$ git tag -a x.x.x -m "Version x.x.x" --sign
$ git push origin x.x.x
```

The release tarball can now be renamed:

```
$ cp apache-couchdb-x.x.x-RC##.tar.gz apache-couchdb-x.x.x.tar.gz
$ cp apache-couchdb-x.x.x-RC##.tar.gz.asc apache-couchdb-x.x.x.tar.gz.asc
$ cp apache-couchdb-x.x.x-RC##.tar.gz.sha256 apache-couchdb-x.x.x.tar.gz.sha256
```

Push the files to the Apache distribution Subversion tree:

```
$ svn mkdir --parents \
    https://dist.apache.org/repos/dist/release/couchdb/source/x.x.x \
    -m 'Add CouchDB x.x.x dir'
$ svn co https://dist.apache.org/repos/dist/release/couchdb/source/x.x.x
$ cp apache-couchdb-x.x.x.tar.gz* x.x.x/
$ cd x.x.x && svn add . && svn ci -m 'Add CouchDB x.x.x release files'
```

Archive older releases

While you're at it, clean up the dist/release/couchdb directory:

- Archive older bugfix releases on the most recent minor feature line
  - *i.e.* If there is a 1.0.1 and you just released 1.0.2, archive 1.0.1
- Archive any past minor versions
  - *i.e.* If there is a 1.1 and you just released 1.2, archive 1.1
- If this is a new major release (*x.0*), keep the last version of the previous major release only and archive the rest.
- Archive any release that is over a year old

The process is simple. Run these commands, once for each version and platform combination:

```
svn rm \
https://dist.apache.org/repos/dist/release/couchdb/source/VERSION \
-m 'Remove Apache CouchDB $VERSION'
svn rm \
https://dist.apache.org/repos/dist/release/couchdb/binary/PLATFORM/VERSION \
-m 'Remove Apache CouchDB $VERSION $PLATFORM binaries'
```

- Replace VERSION with the version you are archiving.
- Replace PLATFORM with the platform of the binaries you are archiving.

Do not worry about deleting these files. They will live on in the archives.

Publishing the Convenience Binaries

macOS and Windows Binaries

Note: As of 3.1.0, we're not doing Apache binary packages for macOS or Windows. Neighbourhoodie have taken over this work.

The macOS and Windows binaries should be uploaded to the official https://dist.apache.org/repos/dist/release/couchdb/binary tree, using instructions similar to the above.

However, because it provides better visibility into our downloads (and is linked directly from our website), these should also be uploaded to Bintray. You will need a Bintray account to do this, and have access to https://bintray.com/apache/couchdb/ granted via INFRA before you can upload.

- Visit the appropriate repo: https://bintray.com/apache/couchdb/couchdb-win or https://bintray.com/apache/couchdb/couchdb-mac .
- Click on the **New Version** link. The version number should be x.x.x, and the description should be "Version x.x.x".
- Click on the breadcrumb of the version number to return to the view of the version itself, such as https://bintray.com/apache/couchdb/couchdb-win/2.1.0 .
- Click on the **Upload Files** link.
- Specify the appropriate target path: **`/win/X.X.X`** or **`/mac/X.X.X`** as appropriate.
- Click to add and upload the installer file. (Bintray automatically generates sha256 checksums.)

- Wait for the upload to finish, then click **Save Changes**.
- In the banner that appears, click **Publish**. You're done.

Debian/Ubuntu and RHEL/CentOS Binaries

- First, build the binaries. On a Linux machine with Docker:

```
$ docker pull --all-tags couchdbdev
$ git clone https://github.com/apache/couchdb-pkg && cd couchdb-pkg
$ ./build.sh couch-all https://dist.apache.org/repos/dist/release/couchdb/source/#.#.#/apache-couchdb-#.
#.#.tar.gz
```

- Double-check that the contents of the `pkgs` directory include the expected results:
  - `couch/centos-6/x86_64` should contain the couchdb RPM as well as the SpiderMonkey 1.8.5 RPMs.
  - `couch/centos-7/x86_64` should contain the couchdb RPM.
  - `couch/debian-jessie|debian-stretch|ubuntu-trusty|ubuntu-xenial|ubuntu-bionic` should each contain the appropriate couchdb `deb`.
- Now, upload the binaries to Bintray and publish them:

```
$ BINTRAY_USER="your-userid-here" BINTRAY_API_KEY="your-key-here" ./build.sh couch-upload-all
```

If this fails, or you want to do it the old labour intensive way, the old web-based approach is below:

  - Visit the appropriate repo: https://bintray.com/apache/couchdb-deb/CouchDB or https://bintray.com/apache/couchdb-rpm/CouchDB
  - Click on the **New Version** link. The version number should be x.x.x, and the description should be "Version x.x.x".
  - Click on the breadcrumb of the version number to return to the view of the version itself, such as https://bintray.com/apache/couchdb-rpm/CouchDB/2.1.0.
  - Click on the **Upload Files** link.
  - For RPMs, specify the appropriate target path: `/el6/x86_64` or `/el7/x86_64` as appropriate.
  - For debs, specify the appropriate parameters: distribution name (**jessie, trusty, xenial**), component name **main**, architecture name **amd 64**. All platforms can be done in one go.
  - Click to add and upload the package. (Bintray automatically generates sha256 checksums.)
  - Wait for the upload to finish, then click **Save Changes** for RPM. For Debian, repeat the last 2 steps for all the packages, then click **Save Changes**.
  - Visit the appropriate repo: https://bintray.com/apache/couchdb-deb/CouchDB or https://bintray.com/apache/couchdb-rpm/CouchDB
  - Click on the newly created version under the **Versions** heading, then click on **Files** to see the uploaded files. Confirm everything is as expected.
  - In the banner that appears, click **Publish**.
  - ***Note: this may not be necessary...*** Sign the Debian repository. You will need your Bintray API Key for this, which you can find under your account settings.

```
$ curl -X POST -u $BINTRAY-USER:$BINTRAY-API-KEY https://api.bintray.com/gpg/apache/couchdb-deb
/CouchDB/versions/X.X.
```

Docker images

- Update the contents of https://github.com/apache/couchdb-docker so that the new version is being built. This is usually just a global search and replace on the Dockerfile as well as a directory rename or copy, plus updating .travis.yml. If this is a security release, just rename the old version directory.
- A PR will then do a test build in Travis and ensure everything is functional. Get that merged into `master`.
- Use the new repo to build the image(s) the same way as in .travis.yml, tag the image(s), and upload to `apache/couchdb:X.X.X` using `docker push`. It's also recommended to push the image as **apache/couchdb:#.#, apache/couchdb:#** and **apache/couchdb:latest**. Do not do the last 2 if this is just a maintenance release on an older supported branch.

  Note that the dev image cannot be uploaded per Apache policy!

- Make a pull request against https://github.com/docker-library/official-images to reference the new commit you made on `master`. Make any changes as requested by the Docker team.

Scheduling the Release Announcement

Choose a date for the release. Tuesdays mornings PST are better, because they get more press coverage. Your date should take into consideration the 24 hour delay while you wait for the mirrors to update themselves after you have uploaded the release files.

Use the contents of `couchdb-admin/email/schedule_release.txt` file to schedule the release, and send the email to the dev@couchdb.apache.org mailing list.

While waiting for the actual announcement to occur, the following preparations can take place:

- Prepare a blog post and review its text on the `marketing@couchdb.apache.org` mailing list.
- Prepare the text of Twitter/other social media updates.
- Prepare any personal blog posts you might like to make at the same time (discussing new features, the release process, etc.)
- Suggest other contributors do the same.

Announcing the Release

You must now wait until all the release files have been synced to the public mirrors. The mirrors can take up to 24 hours to update, so it's usually best to wait a day. You should be sure that both the source download and the binary downloads are all working.

Then:

- Wait for the scheduled release time before announcing the release. If there is no scheduled release time, you can simply wait until the mirrors are up-to-date.
- Update http://couchdb.apache.org/ to point to the new files (checkout the repository from https://git-wip-us.apache.org/repos/asf/couchdb-www.git , branch `asf-site`.)
- Update the wiki with the new release information.
- Wait for all changes to be synced to the website.
- Publish the blog post.
- Prepare the email release announcement, using the `couchdb-admin/email/announce_release.txt` template. You will want to cut and paste the release notes from the website into this email and prepare appropriately.
- Then, send to the following lists:
    - announce@apache.org
    - announce@couchdb.apache.org
    - user@couchdb.apache.org
    - dev@couchdb.apache.org
    - marketing@couchdb.apache.org

Promoting the Release

Twitter

Log in to the CouchDB Twitter account, and send a tweet using this template:

```
Apache CouchDB VERSION has been released. This is a RELEASE release. Download: http://couchdb.apache.org/
Release notes: BLOG_URL
```

Hacker News

Post a new item to Hacker News.

Use the following title template: `Apache CouchDB VERSION Released`

Submit the URL of the blog post you just published.

Reddit

Post several new items to Reddit.

Use the following title template: `Apache CouchDB VERSION Released`

Submit the URL of the blog post you just published.

Submit a new item for each of the following subreddits:

- /r/programming
- /r/Database
- /r/nosql
- /r/CouchDB

Lobste.rs

If this is an important release, you might want to post it to https://lobste.rs/. Follow the same procedure as Hacker News.

Facebook

Post a link to the CouchDB page. Try to provide a useful summary of the release.

Be sure to like the post from your own account.

Mailing List

Open the `couchdb-admin/email/request_promotion.txt` template and edit as necessary. Then, send to the following lists:

- user@couchdb.apache.org
- dev@couchdb.apache.org
- marketing@couchdb.apache.org

Useful Resources

- CouchDB Bylaws - our official policy on proposing, voting on, and making a CouchDB release.
- Apache Release Policy - the official ASF stance to which we must match
- Apache Release Publishing - the technical steps necessary to publish a release
- Apache Release Distribution Policy - where the releases go, how to link to them and how to announce them.
- Apache's How to OpenPGP - for setting up OpenPGP
- Apache Signing Releases - how to use OpenPGP to sign a release
- Verifying ASF Releases - how to check a signature once it's been made
- Useful resources to check release integrity and mirror status: http://mirror-vm.apache.org/~henkp/
- Apache Committee Report Helper - for PMC members to note that an official release has been made for the next Board Report