# Release Process for Upstream Ceph

Ensure that you are building for the distros that Ceph supports for the given release. For example in Jewel it would mean building for: CentOS 7, Jessie, Xenial, Trusty.

**ARCHS variable matrix**
Luminous (12.X.X): xenial centos7 trusty jessie stretch
Mimic (13.X.X): bionic xenial centos7
Nautilus (14.X.X): bionic xenial centos7 centos8
Octopus (15.X.X): focal bionic centos7 centos8 buster
Pacific (16.X.X): focal bionic centos8 buster

## New major releases:

For new MAJOR (alphabetical) releases, RPM repos need a 'ceph-release' rpm created. The chacra repos are configured to include this RPM but it must be built separately. You must make sure that chacra is properly configured to include this RPM though for the release you are doing. See this PR for an example:

https://github.com/ceph/chacra/pull/219

If chacra is not configured correctly please make that change and deploy it before starting the build of ceph or ceph-release.

There is a job in Jenkins specific for this that takes the name of the release as the parameter. For the kraken release, the irc command would look like:

```
!ci build ceph-release-rpm RELEASE=kraken
```

Or in the Jenkins Web UI, enter the release/branch name and click **Build**.

Once it is complete, chacra has a trigger that will rebuild the branch's repository to include this one binary. It's easier if you do this step before the ceph build so that the ceph-release rpm will be included in the repos from the start.

## Freezing the branch (Yuri usually does this):

1. Go to https://github.com/ceph/ceph/settings/branches
2. Click **Edit** next to the BRANCH you're releasing
3. Set **Required approving reviews** to **6**
4. Save Changes

## Starting the build:

1. Visit https://jenkins.ceph.com/job/ceph/build?delay=0sec
2. Set BRANCH to the release (e.g., nautilus, octopus, etc.)
3. Check TAG (**unless** you're rebuilding an existing version)
4. Set VERSION to the version (e.g., 12.2.X [**do NOT add the 'v'**])
5. Set DISTROS using the matrix at the top of this document
6. ARCHS should be 'x86_64 arm64'
7. Click **Build** and wait

## Release Notes:

While packages are building, now's a good time to work on the Release Notes. That is documented here: HOWTO write the release notes - Stable releases

## Signing the build:

The signing key lives on a NitroKey and should already be unlocked unless the signer box was rebooted or gpg-agent was restarted. The PIN to unlock the NitroKey is in /root/secrets on magna001.ceph.redhat.com so a Red Hatter will need to unlock the key before packages can be signed without entering the passphrase.

1. To get the sha1, open any of the ceph-build sub-jobs
    a. e.g., https://jenkins.ceph.com/job/ceph-build/ARCH=x86_64,AVAILABLE_ARCH=x86_64,AVAILABLE_DIST=centos8,DIST=centos8,MACHINE_SIZE=gigantic/
    b. Open the most recent job's **Environmental Variables** and copy the **sha1** variable
2. `ssh ubuntu@signer.front.sepia.ceph.com`
3. `sync-pull ceph octopus <sha1>`

**To sign DEB binaries:**

1. `merfi gpg /opt/repos/ceph/nautilus/debian`
   OR
   `merfi gpg /opt/repos/ceph/octopus-15.X.X/debian`

   The output should look like it actually did something and it didn't error. It should look beautiful

**To sign RPM binaries:**

1. `~/bin/sign-rpms octopus`

## Publishing the build:

```
sync-push octopus
```

## Pushing Tags:

If you haven't already (on your workstation),

1. `git clone github.com/ceph/ceph-releases.git`
2. `git remote add gh git@github.com:ceph/ceph.git`

Otherwise/then,

1. `git checkout [nautilus|octopus|etc.]`
2. `git pull`

TEMPORARILY relax the BRANCH's GitHub settings so that you can push the release commit

- Visit https://github.com/ceph/ceph/settings/branches
- Click **Edit** next to the branch you're releasing
- Uncheck **Include administrators**
- Click **Save changes**

Push the changes:

```
git push gh <release>
```

So for jewel it would look like:

```
git push gh jewel
```

That will push the commit but not the tag. For the tag:

~~`git push gh --tags`~~
```
git push gh vX.X.X
```

If you try to push and get rejected it may be because you do not have admin privileges to the branch. You will also might need to 'unprotect' the branch in the github.com UI.` Make sure the 'Restrict who can push to this branch' is unchecked for the branch you want to push to.

If commits get merged into the ceph/ceph.git release branch before you're able to push the release commit from ceph-releases.git, just `git fetch gh; git merge gh/$branch` the latest commits. You'll have an ugly extra "Merge commit asdf1234" commit as the new tip but that's okay.

## Unfreeze the branch:

You need to undo the **Freezing the branch** bit now so only 1 reviewer is required.

1. Visit https://github.com/ceph/ceph/settings/branches
2. Click **Edit** next to the BRANCH you're releasing
3. Set **Required approving reviews** to **1**
4. Check **Include administrators**
5. Click **Save changes**

## Tarballs:

After tags are pushed and repos are synced, one last item needs to be done: tarballs need to be placed on download.ceph.com. This process could be automated (TODO!), for the 10.2.4 Jewel release, it meant:

```
$ ssh signer@download.ceph.com (from the signer node)
$ get-tarballs.sh RELEASE SHA1 VERSION
(e.g., get-tarballs.sh mimic 5533ecdc0fda920179d7ad84e0aa65a127b20d77
13.2.1)
```

## Containers:

Start

- https://2.jenkins.ceph.com/job/ceph-container-build-ceph-base-push-imgs/
- https://2.jenkins.ceph.com/job/ceph-container-build-ceph-base-push-imgs-arm64/

## Publish Release Notes:

Send your Release Notes e-mail and publish the ceph.io blog post.

# Brainstorming new ideas

- ~~Use Nitrokey Pros for storing the secret keys. Advantages:~~
    - ~~Secret key material cannot leave device. No one can copy the keys anywhere dangerous any more.~~
    - ~~A long-running gpg-agent daemon becomes safer, and this would allow more automated signing (passphrase is only needed once at gpg-agent daemon startup).~~
- Extend Merfi to sign more content:
    - Git tags
    - Tarballs
    - RPMs (and use sha256, as documented here: https://access.redhat.com/solutions/2973701 )
    - Yum repodata

    - (Docker containers in the future?)
        - Atomic Host Signing demo: https://youtu.be/0yoQu-YylwA
        - Atomic Host Managing trust demo: https://youtu.be/93-71phWiOg
- Sample workflow:
    - Ansible playbook to log into signer node and create the new signed tag:
        - “merfi git-tag” creates the signed Git tag
        - Ansible pushes the tag to https://github.com/ceph/ceph-releases
    - Jenkins sees the new tag in ceph-releases.git and triggers new “release” builds.
    - Jenkins completes builds and uploads them to shaman/chacra.
    - Merfi signs all newly-built content.
        - We could have a polling job on the signer node, or we could do this with a second Ansible playbook
    - rsync signed content to download.ceph.com
    - celebrate