

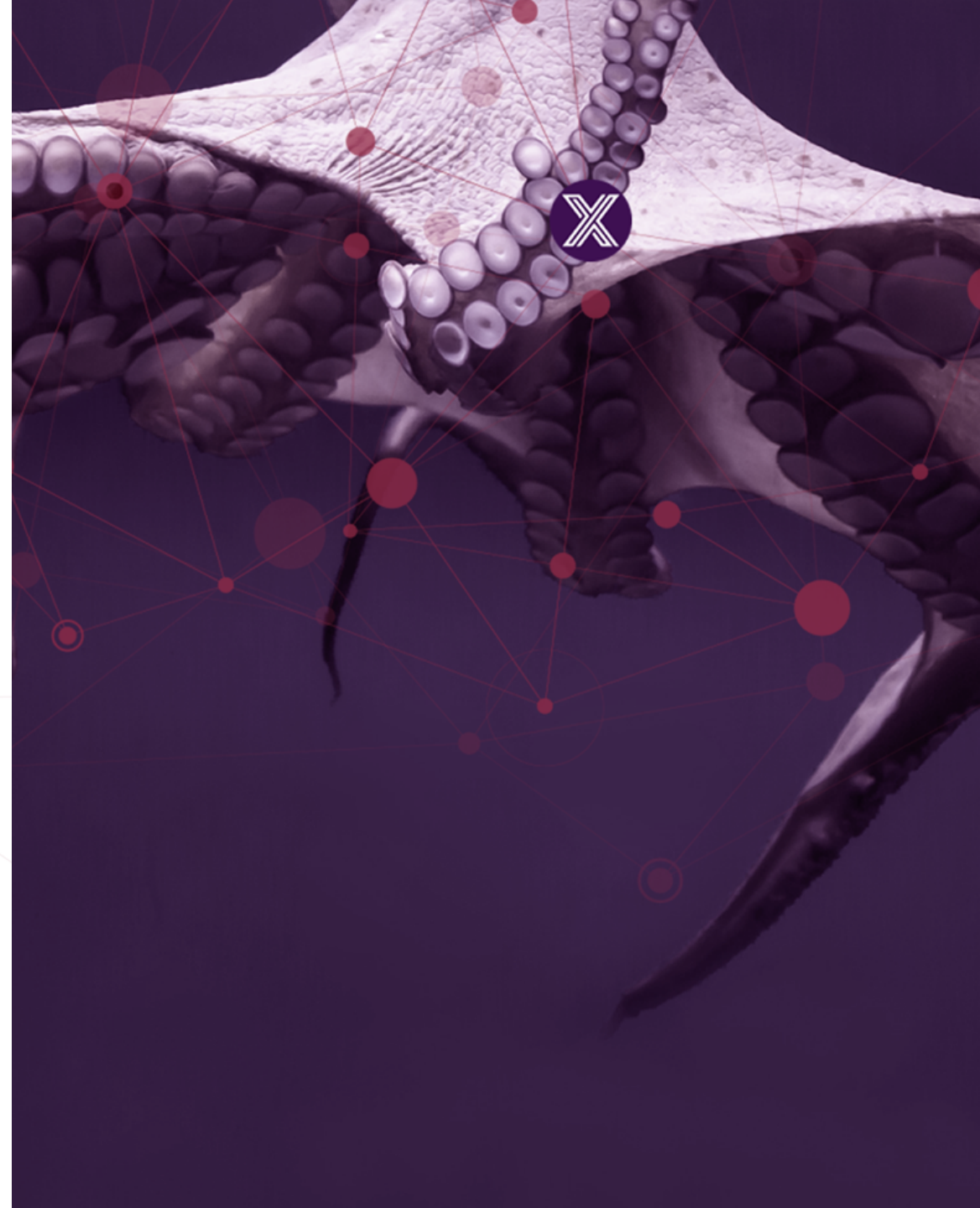
EDGE X FOUNDRY™

Docker Official Images

OFFICIAL IMAGE



Nov 19, 2020



❖ Docker Official Images

- Docker Official Images are a set of carefully chosen and thoughtfully organized Docker repositories hosted on Hub. Mainly designed to:
 - Guide new docker users
 - Ensures **security updates** are applied on time.
 - **Exemplifies** Dockerfile best practices and to serve as a reference.
- A **sponsored dedicated team** from Docker, which takes complete responsibility for reviewing and publishing all content.
- An **upstream software authors** team (preferable to have) maintains their corresponding Official Images, in collaboration with dedicated team, security experts and the broader Docker community.

❖ What does the word Official mean and How to create an Official Image ???

- The images in this program are not only created/supported but also maintained directly by the relevant upstream projects, in collaboration with a docker sponsored dedicated team.
 - To put that another way, a particular 2.5.6 software release is considered supported if a severe bug being found would cause a 2.5.7 release (and once 2.5.7 is released, 2.5.6 is no longer considered supported, but the Docker Hub tag is typically left available for pulling -- it will simply never get rebuilt after that point given that it is unsupported).
- From a high level, an Official Image starts out as a proposal in the form of a set of GitHub pull requests.
- Each proposal goes through a back-and-forth review process by the Official Images team, with help from community contributors.
- For a smooth review process every user/upstream software authors team should ensure that the Dockerfiles adhere to all the points mentioned in the Docker best practices, as well as the official review guidelines.


➤ [Review Guidelines](#)

❖ What does the word Official mean and How to create an Official Image ???

- No official images can be derived from, or depend on, non-official images.
- Issue section of GH page or a freenode IRC [#docker-library](#) for general feedback and support questions.
- If anybody wants to provide feedback, contribute code, suggest process changes or even propose a new Official Image, they can place a PR openly on relevant GitHub repositories.
- Bashbrew ([bashbrew](#)): tool for cloning, building, tagging, and pushing the Docker official images.

➤ [#bashbrew](#)

❖ An Example Docker Official Images



golang ☆

Docker Official Images

Go (golang) is a general purpose, higher-level, imperative programming language.

↓ 1B+

Container Windows Linux ARM 64 mips64le ARM 386 PowerPC 64 LE x86-64 IBM Z

Programming Languages Official Image

Description Reviews Tags

Quick reference


- **Maintained by:** the Docker Community
- **Where to get help:** the Docker Community Forums, the Docker Community Slack, or Stack Overflow

Linux - IBM Z (latest)

Copy and paste to pull this image

```
docker pull golang
```

View Available Tags



kong ☆

Docker Official Images

The Cloud-Native API Gateway for APIs and Microservices

↓ 100M+

Container Linux ARM 64 x86-64 Application Frameworks Official Image

Description Reviews Tags

Quick reference

- **Maintained by:** the Kong Docker Maintainers
- **Where to get help:** the Docker Community Forums, the Docker Community Slack, or Stack Overflow

Linux - x86-64 (latest)

Copy and paste to pull this image

```
docker pull kong
```

View Available Tags

❖ Build Process

► An image's source changed in Git, now what?

Let's walk through the full lifecycle of a change to an image to help explain the process better. We'll use the `golang` image as an example to help illustrate each step.

1. a change gets committed to the relevant image source Git repository (either via direct commit, PR, or [some automated process](#) -- somehow some change is committed to the Git repository for the image source)
 - for `golang`, that would be a commit to the [github.com/docker-library/golang](#) repository, such as `a9171b` (the update to Go 1.12.6)
2. a PR to the relevant `library/xxx` manifest file is created against <https://github.com/docker-library/official-images> (which is the source-of-truth for the official images program as a whole)
 - for `a9171b`, that PR would be [docker-library/official-images#6070](#) (changing `library/golang` within that repository to point to the updated commits from the [github.com/docker-library/golang](#) repository and updating `Tags:` references if applicable)
3. that PR and a full diff of the actual `Dockerfile` and related [build context](#) files are then reviewed by the [official images maintainers](#)
 - for [docker-library/official-images#6070](#), that can be seen in the "Diff:" comment
4. a basic build test is produced, typically on `amd64` (to ensure that it will likely build properly on the real build servers if accepted, and to run [a small series of official images tests](#) against the built image)
 - for [docker-library/official-images#6070](#), that can be seen in [the build test comment](#)
5. once merged, [the official images build infrastructure](#) will pick up the changes and build and push to the relevant per-architecture repositories (`amd64/xxx`, `arm64v8/xxx`, etc)
 - for `golang`, those per-architecture build jobs can be seen in [the "golang" view in the build infrastructure](#)

5. once merged, [the official images build infrastructure](#) will pick up the changes and build and push to the relevant per-architecture repositories (`amd64/xxx`, `arm64v8/xxx`, etc)

- for `golang`, those per-architecture build jobs can be seen in [the "golang" view in the build infrastructure](#)

6. after those jobs push updated artifacts to the architecture-specific repositories (`amd64/xxx`, `arm64v8/xxx`, etc), [a separate job](#) collects those updates into "index" objects (also known as "manifest lists") under `library/xxx` (which is the "default" namespace within Docker)

- for `golang`, that would be the [put-shared/light](#) job, because it is not a "heavy hitter"

For images [maintained by the docker-library team](#), we typically include a couple useful scripts in the repository itself, like `./update.sh` and `./generate-stackbrew-library.sh`, which help with automating simple version bumps via `Dockerfile` templating, and generating the contents of the `library/xxx` manifest file, respectively. [We also have infrastructure](#) which performs those version bumps along with a build and test and commits them directly to the relevant image repository (which is exactly how [the illustrative golang a9171b commit](#) referenced above was created).

► How are images built? (especially multiarch)

Images are built via a [semi-complex Jenkins infrastructure](#), and the sources for much of that can be found in [the github.com/docker-library/oi-janky-groovy](#) repository.

The actual infrastructure is a combination of machines provided by our generous donors:

- `amd64`, `i386`, Jenkins nodes: [Docker, Inc.](#)
- `arm32v7`, `arm64v8`: [WorksOnArm](#)
- `mips64le`: [Loongson](#)
- `ppc64le`, `s390x`: [IBM](#)

For a more complete view of the full image change/publishing process, see ["An image's source changed in Git, now"](#)

► [#build-process](#)

► [#doi-jenkins-infra](#)

❖ PR Process for Official Image request

docker-library / official-images

<> Code Issues 9 Pull requests 23 Actions Security Insights

Add new caddy image #7574

Merged yosifkit merged 8 commits into docker-library:master from hairyhenderson:add-caddy-image on Apr 14

Conversation 25 Commits 8 Checks 0 Files changed 1

hairyhenderson commented on Mar 5 • edited

Hi there! This proposes a new official image for Caddy, "a powerful, enterprise-ready, open source web server with automatic HTTPS written in Go."

This is for Caddy v2+, which is in RC (currently at v2.0.0-rc.3). It complements the (quite popular) Caddy v1 image <https://hub.docker.com/r/abiosoft/caddy>. I've been working with @mholt (Caddy's author) to create this one.

I've got a bashbrew-based build currently working for the caddy/caddy image, which I'm hoping to replace with an official caddy image.

For now I'm only building on amd64, but hope to add new architectures soon. I'm somewhat uncertain how to test multi-platforms myself (I'd use buildx, but I gather bashbrew doesn't support that?).

Docs PR is here: [docker-library/docs#1667](https://github.com/docker-library/docs/pull/1667)

Checklist for Review

NOTE: This checklist is intended for the use of the Official Images maintainers both to track the status of your PR and to help inform

Example PR which is merged

Request to add archivebox to official images #8457

Open pirate wants to merge 2 commits into docker-library:master from pirate:patch-1

Conversation 1 Commits 2 Checks 3 Files changed 1



pirate commented on Jul 30 • edited

Hi! This is my first attempt at submitting an official image for ArchiveBox (sorry if I got anything wrong), we've had many of our users request one since they're thrown off by my personal username being nikisweeting, and aren't sure what image they can trust out of the several options available.

The project has about ~7k stars with 50+ contributors and 100K+ Docker hub pulls, I don't know if that's big enough to qualify or if there are additional steps needed. So far I've added the commit hashes for our latest release, 0.4.13 to the file in this PR. Here is the image in its current form if you'd like to vet it for quality:

- <https://hub.docker.com/layers/nikisweeting/archivebox>
- <https://github.com/pirate/ArchiveBox/blob/master/Dockerfile>
- <https://github.com/pirate/ArchiveBox/blob/master/docker-compose.yml>

I took great pains to make the docker CLI experience absolutely seamless and interchangeable with running directly on the host, including detecting the ownership of mounted volumes and changing the container user to match the host machine so as to "automagically" create files with the correct permissions. So far this seems to have gone well, and many of our users are moving

Example PR which is pending for long

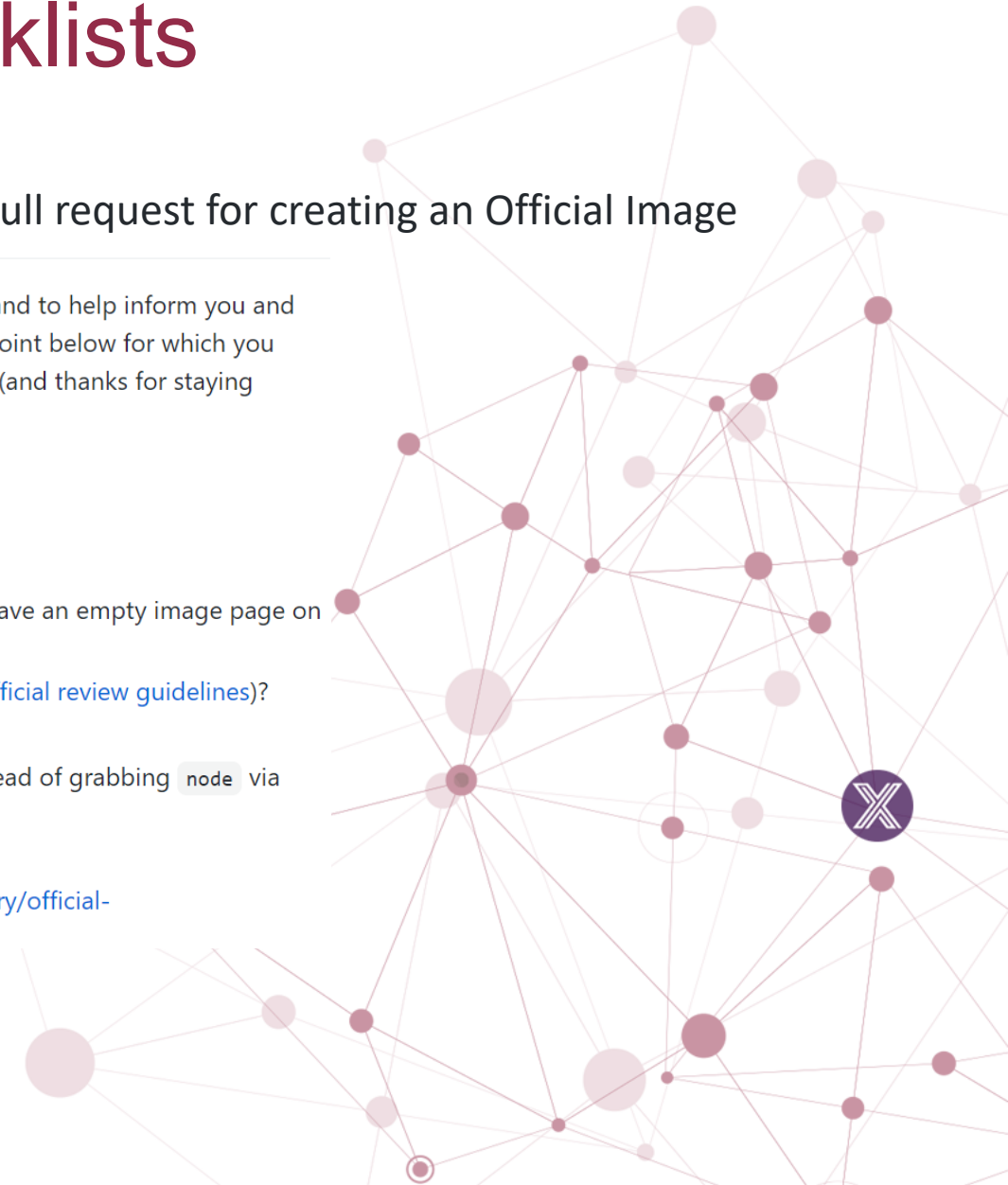
❖ Review guidelines and Checklists

A checklist which may be used by the maintainers during review of Pull request for creating an Official Image

NOTE: This checklist is intended for the use of the Official Images maintainers both to track the status of your PR and to help inform you and others of where we're at. As such, please leave the "checking" of items to the repository maintainers. If there is a point below for which you would like to provide additional information or note completion, please do so by commenting on the PR. Thanks! (and thanks for staying patient with us ❤️)

- associated with or contacted upstream?
- does it fit into one of the common categories? ("service", "language stack", "base distribution")
- is it reasonably popular, or does it solve a particular use case well?
- does a [documentation](#) PR exist? (should be reviewed and merged at roughly the same time so that we don't have an empty image page on the Hub for very long)
- official-images maintainer dockerization review for best practices and cache gotchas/improvements (ala [the official review guidelines](#))?
- 2+ official-images maintainer dockerization review?
- existing official images have been considered as a base? (ie, if `foobar` needs Node.js, has `FROM node:...` instead of grabbing `node` via other means been considered?)
- if `FROM scratch`, tarballs only exist in a single commit within the associated history?
- passes current tests? any simple new tests that might be appropriate to add? (<https://github.com/docker-library/official-images/tree/master/test>)

➤ [#review-guidelines](#) [#checklists](#)



- Have an idea, feature request or issue with any image?

<https://github.com/docker-library/official-images/pulls>

- Whom to contact for an image?

For Individual Image: https://github.com/docker-library/docs/tree/master/<img_name>

- Where to post your feedback?

Pull requests or Freenode IRC channel #docker-library

- Want to create an Official Image?

An Official Image starts out as a proposal in the form of a set of GitHub pull requests.

- Supported Architectures

Some images have been ported for other architectures than amd64, and many of these are officially supported.

- Review Guidelines

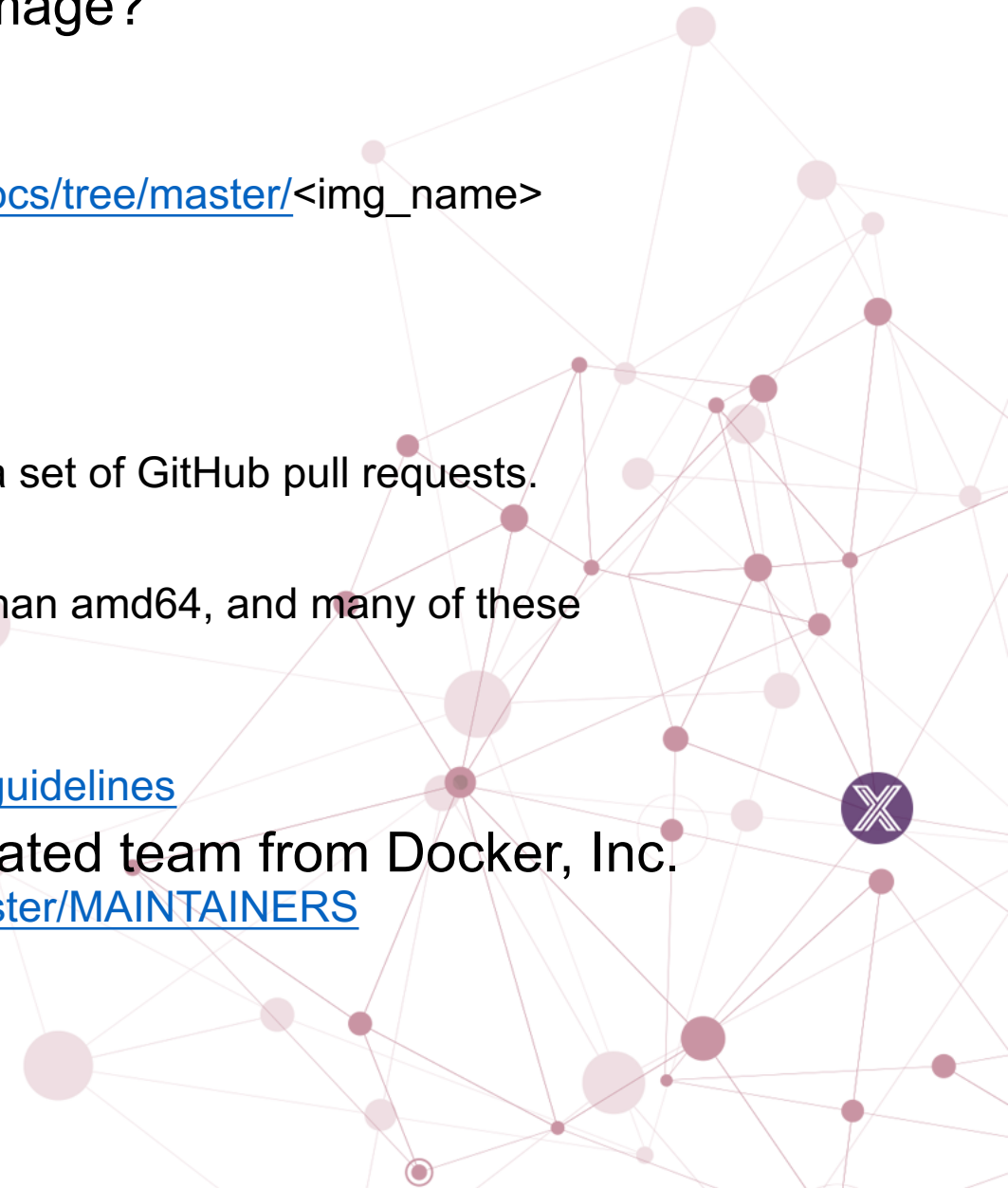
<https://github.com/docker-library/official-images#review-guidelines>

- Maintainers from The Official Images team: dedicated team from Docker, Inc.

<https://github.com/docker-library/official-images/blob/master/MAINTAINERS>

- List of Official Images from the HUB

<https://hub.docker.com/u/library>



❖ More References:

- https://docs.docker.com/docker-hub/official_images/
- <https://hub.docker.com/u/library>
- <https://github.com/docker-library/official-images>
- <https://github.com/docker-library/docs>
- <https://github.com/docker-library/faq>
- <https://github.com/docker-library/bashbrew>
- https://github.com/docker-library/<image_name>