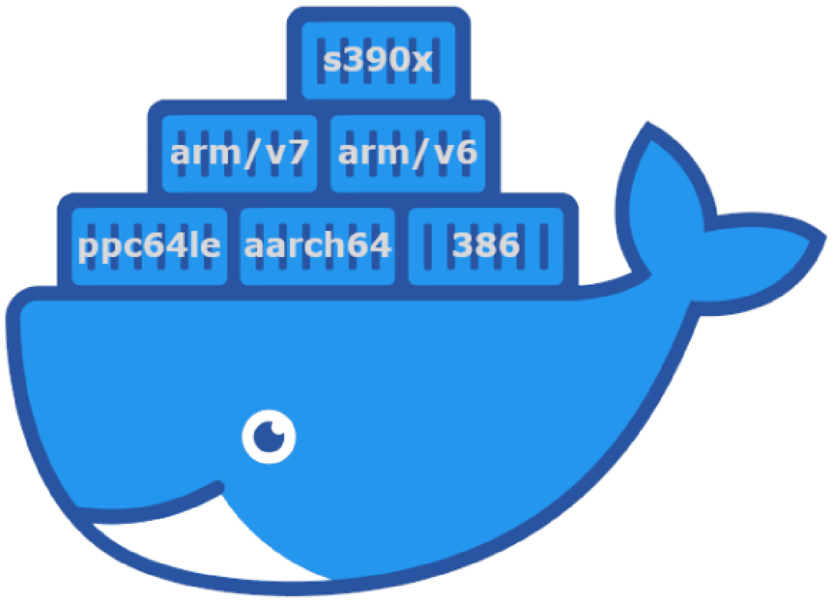
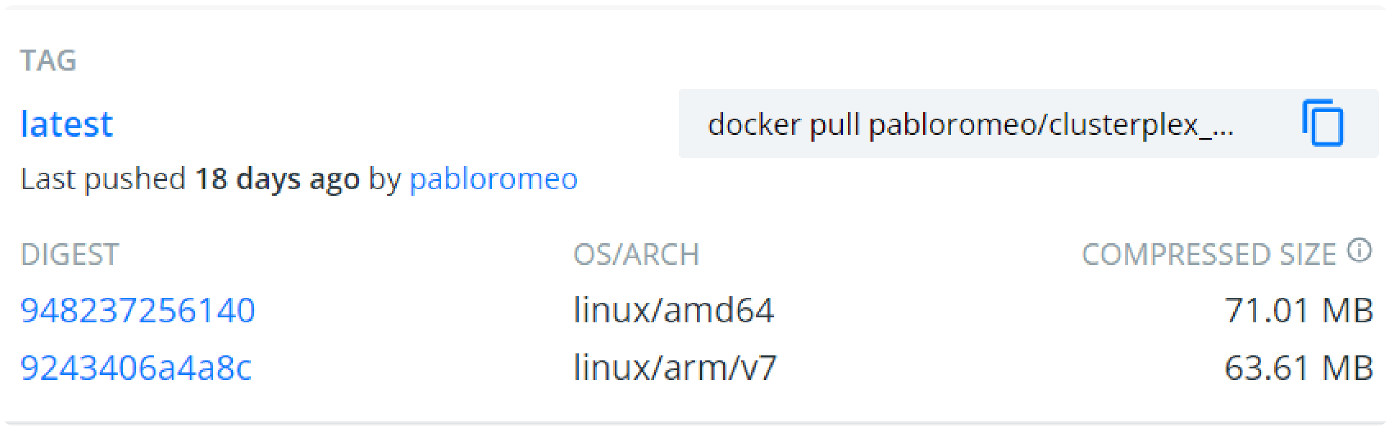


Pablo Romeo for Cloud(x);
Posted on Jan 12, 2022 • Updated on Feb 3, 2022

11

Multi-arch docker images the easy way, with Github Actions

#docker #github #githubactions #cicd


TAG

latest

docker pull pabloromeo/clusterplex_...

Last pushed **18 days ago** by pabloromeo

DIGEST	OS/ARCH	COMPRESSED SIZE
948237256140	linux/amd64	71.01 MB
9243406a4a8c	linux/arm/v7	63.61 MB

You may have come across Docker images like the one above ☝. In this example, you can see that the `latest` tag supports two different architectures. Meaning there are two different images for the same tag, and docker will pull the appropriate one depending on the target architecture. But...how are they built?

The old way

Back in the day, the only option was to manually built each separate image using the `ARCH` build-arg, push it, and then use the `docker manifest` command to group the images together under the same manifest.
This manual process is described in depth in this post by the Docker team. But now there's even an easier way.

Buildx to the rescue

Thankfully Buildx automates this process and simplifies it quite a bit. A single command can do all that for you.
But who wants to be building images locally anyway? 😁
Plus, automating it with Github Actions is a great excuse to also implement a proper tagging and versioning strategy as well as exploring multiple Container Registries.
Let's get to it! 💪

Github Actions build

The overall process is as follows:

1. Checkout the code
2. Setup QEMU
3. Setup Buildx
4. Log into the target container registry
5. Docker Metadata tag voodoo magic *(optional)*
6. Build and push

Let's assume we have a Github repo with a `main` and a `dev` branch, PRs, as well as versioned releases.

We'll have a github actions YAML similar to the following:

```
name: Build

# Controls when the workflow will run
on:
  workflow_dispatch
  push:
    branches:
      - 'main'
      - 'dev'
    tags:
      - 'v*.*.*'
  pull_request:
    branches:
      - 'main'
      - 'dev'

# permissions are needed if pushing to ghcr.io
permissions:
  packages: write

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
```

Steps 1,2 and 3

These are quite straightforward:

```
      # Get the repository's code
      - name: Checkout
        uses: actions/checkout@v2
      # https://github.com/docker/setup-qemu-action
      - name: Set up QEMU
        uses: docker/setup-qemu-action@v1
      # https://github.com/docker/setup-buildx-action
      - name: Set up Docker Buildx
        id: buildx
        uses: docker/setup-buildx-action@v1
```

Step 4

For Github's Container Registry you can use the following:

```
      - name: Login to GHCR
        if: github.event_name != 'pull_request'
        uses: docker/login-action@v1
        with:
          registry: ghcr.io
          username: ${{ github.repository_owner }}
          password: ${{ secrets.GITHUB_TOKEN }}
```

And for Dockerhub:

```
      - name: Login to Docker Hub
        if: github.event_name != 'pull_request'
        uses: docker/login-action@v1
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
```

Create a dockerhub access token and set the corresponding github secrets.

Step 5

We could skip this step and just use the `latest` docker tag, but in real world scenarios you'll likely want a more elaborate image tagging strategy. Such as sha, branch, Semantic Versioning, etc.

So we'll use the extremely useful `docker/metadata-action@v3` for preparing those tags for us.
Check out https://github.com/docker/metadata-action for more example tagging strategies. The documentation is great!

```
      - name: Docker meta
        id: <meta_id> # you'll use this in the next step
        uses: docker/metadata-action@v3
        with:
          # list of Docker images to use as base name for tags
          images: |
            <specify the image name>
          # Docker tags based on the following events/attributes
          tags: |
            type=schedule
            type=ref,event=branch
            type=ref,event=pr
            type=semver,pattern={{version}}
            type=semver,pattern={{major}}.{{minor}}
            type=semver,pattern={{major}}
            type=sha
```

- This prepares image tags for branch, PR, major, major.minor, version, and sha, which you should adapt according to your own release process.
- Replace `meta_id` with something more appropriate. It will be used in the next step.
- Specify the image name under the `images:` section. For ex: `ghcr.io/pabloromeo/foo` for GHCR or `pabloromeo/foo` for dockerhub.

Step 6

And the final step is to actually build for each target platform and push to the registry.

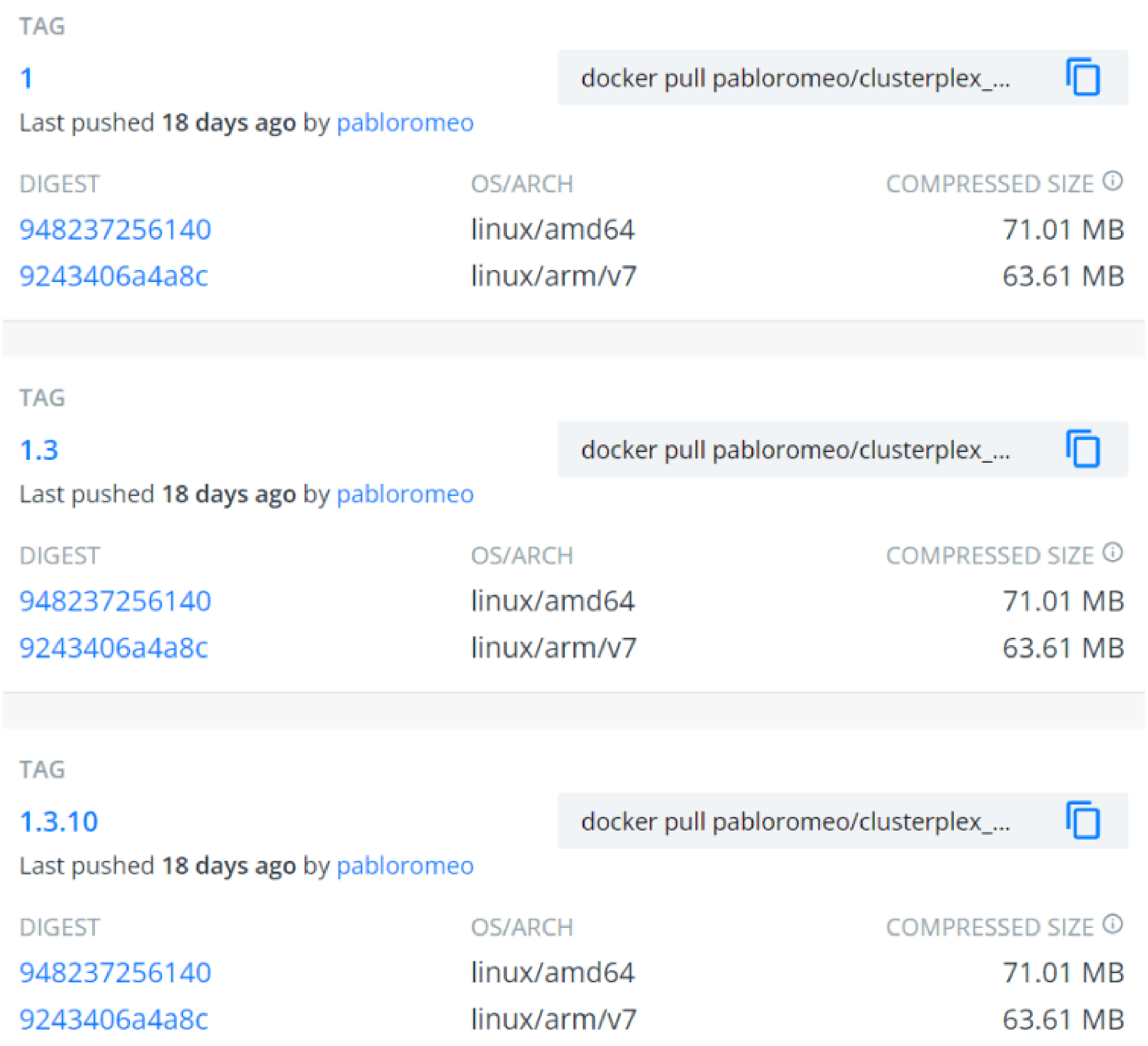
```
      - name: Build and push
        uses: docker/build-push-action@v2
        with:
          context: .
          platforms: linux/amd64,linux/arm/v7
          push: ${{ github.event_name != 'pull_request' }}
          tags: ${{ steps.<meta_id>.outputs.tags }}
          labels: ${{ steps.<meta_id>.outputs.labels }}
```

Change the following:

- The target architecture platforms to build, using `platforms:` (in this example AMD64 and ARMv7)
- Replace `<meta_id>` with the proper id used previously.

Final comments

This turned out to be a lengthy post, but the actual build definition is quite simple when put together, and you'll not only get multi-arch docker images, but nice semver versioning and tagging as well. 😄


TAG
1
Last pushed **18 days ago** by pabloromeo

docker pull pabloromeo/clusterplex_...

DIGEST	OS/ARCH	COMPRESSED SIZE
948237256140	linux/amd64	71.01 MB
9243406a4a8c	linux/arm/v7	63.61 MB

TAG
1.3
Last pushed **18 days ago** by pabloromeo

docker pull pabloromeo/clusterplex_...

DIGEST	OS/ARCH	COMPRESSED SIZE
948237256140	linux/amd64	71.01 MB
9243406a4a8c	linux/arm/v7	63.61 MB

TAG
1.3.10
Last pushed **18 days ago** by pabloromeo

docker pull pabloromeo/clusterplex_...

DIGEST	OS/ARCH	COMPRESSED SIZE
948237256140	linux/amd64	71.01 MB
9243406a4a8c	linux/arm/v7	63.61 MB

To see full working examples of this we can take a look at the builds of two pet projects of mine: Docker-Dogecoin and ClusterPlex.

In Docker-Dogecoin's Dockerfile there's an example of how to write conditional logic as part of the docker build process depending on the target architecture to install different binaries for each.

In Clusterplex's build it actually pushes to BOTH ghcr.io and Dockerhub simultaneously, as well as updating Dockerhub's Description page to the content of the README.md in the repo.

Top comments (1)

Andrzej Rehmann • Dec 27 '22

What a gem, thanks mate. This worked perfectly out of the box. Great examples and perfectly explained.

Code of Conduct • Report abuse

DEV Community

50 CLI Tools You Can't Live Without

>> Check out this classic DEV post <<

Cloud(x);

More from Cloud(x);

Running graphic apps in Docker: AWS WorkSpaces
#tutorial #docker #aws #security

Running Azure SQL Edge on Docker on macOS M1
#azuresqledge #docker #m1 #arm

Connecting through OpenVPN with deprecated ciphers, using Docker
#tutorial #docker #devops #security