Build multi-arch extensions

- Build and push for multiple architectures
- Adding multi-arch binaries
- Can I develop extensions that run Windows containers?

It is highly recommended that, at a minimum, your extension is supported for the following architectures:

- `linux/amd64`
- `linux/arm64`

Docker Desktop retrieves the extension image according to the user's system architecture. If the extension does not provide an image that matches the user's system architecture, Docker Desktop is not able to install the extension. As a result, users can't run the extension in Docker Desktop.

Build and push for multiple architectures

If you created an extension from the `docker extension init` command, the `Makefile` at the root of the directory includes a target with name `push-extension`.

You can run `make push-extension` to build your extension against both `linux/amd64` and `linux/arm64` platforms, and push them to Docker Hub.

For example:

```
$ make push-extension
```

Alternatively, if you started from an empty directory, use the command below to build your extension for multiple architectures:

```
$ docker buildx build --push --platform=linux/amd64,linux/arm64 --tag=username/my-
extension:0.0.1 .
```

You can then check the image manifest to see if the image is available for both architectures using the `docker buildx imagetools` command:

```
$ docker buildx imagetools inspect username/my-extension:0.0.1
Name:      docker.io/username/my-extension:0.0.1
MediaType: application/vnd.docker.distribution.manifest.list.v2+json
Digest:    sha256:f3b552e65508d9203b46db507bb121f1b644e53a22f851185d8e53d873417c48

Manifests:
  Name:      docker.io/username/my-extension:0.0.1@sha256:71d7ecf3cd12d9a99e73ef448bf63ae12751fe3a436a007cb0969f0dc4184c8c
  MediaType: application/vnd.docker.distribution.manifest.v2+json
  Platform:  linux/amd64

  Name:      docker.io/username/my-extension:0.0.1@sha256:5ba4ceea65579fdd1181dfa103cc437d8e19d87239683cf5040e633211387ccf
  MediaType: application/vnd.docker.distribution.manifest.v2+json
  Platform:  linux/arm64
```

> If you're having trouble pushing the image, make sure you're signed in to Docker Hub. Otherwise, run `docker login` to authenticate.

For more information, see Multi-platform images page.

Adding multi-arch binaries

If your extension includes some binaries that deploy to the host, it’s important that they also have the right architecture when building the extension against multiple architectures.

Currently, Docker does not provide a way to explicitly specify multiple binaries for every architecture in the `metadata.json` file. However, you can add architecture-specific binaries depending on the `TARGETARCH` in the extension’s `Dockerfile`.

The following example shows an extension that uses a binary as part of its operations. The extension needs to run both in Docker Desktop for Mac and Windows.

In the `Dockerfile`, download the binary depending on the target architecture:

```
#syntax=docker/dockerfile:1.3-labs

FROM alpine AS dl
WORKDIR /tmp
RUN apk add --no-cache curl tar
ARG TARGETARCH
RUN <<EOT ash
    mkdir -p /out/darwin
    curl -fSsLo /out/darwin/kubectl "https://dl.k8s.io/release/$(curl -Ls
https://dl.k8s.io/release/stable.txt)/bin/darwin/${TARGETARCH}/kubectl"
    chmod a+x /out/darwin/kubectl
EOT
RUN <<EOT ash
    if [ "amd64" = "$TARGETARCH" ]; then
        mkdir -p /out/windows
        curl -fSsLo /out/windows/kubectl.exe "https://dl.k8s.io/release/$(curl -Ls
https://dl.k8s.io/release/stable.txt)/bin/windows/amd64/kubectl.exe"
    fi
EOT

FROM alpine
LABEL org.opencontainers.image.title="example-extension" \
    org.opencontainers.image.description="My Example Extension" \
    org.opencontainers.image.vendor="Docker Inc." \
    com.docker.desktop.extension.api.version=">= 0.3.3"

COPY --from=dl /out /
```

In the `metadata.json` file, specify the path for every binary on every platform:

```
{
  "icon": "docker.svg",
  "ui": {
    "dashboard-tab": {
      "title": "Example Extension",
      "src": "index.html",
      "root": "ui"
    }
  },
  "host": {
```

```
    "binaries": [
      {
        "darwin": [
          {
            "path": "/darwin/kubectl"
          }
        ],
        "windows": [
          {
            "path": "/windows/kubectl.exe"
          }
        ]
      }
    ]
  }
}
```

As a result, when `TARGETARCH` equals:

- `arm64`, the `kubectl` binary fetched corresponds to the `arm64` architecture, and is copied to `/darwin/kubectl` in the final stage.
- `amd64`, two `kubectl` binaries are fetched. One for Darwin and another for Windows. They are copied to `/darwin/kubectl` and `/windows/kubectl.exe` respectively, in the final stage.

> The binary destination path for Darwin is `darwin/kubectl` in both cases. The only change is the architecture-specific binary that is downloaded.

When the extension is installed, the extension framework copies the binaries from the extension image at `/darwin/kubectl` for Darwin, or `/windows/kubectl.exe` for Windows, to a specific location in the user's host filesystem.

Can I develop extensions that run Windows containers?

Although Docker Extensions is supported on Docker Desktop for Windows, Mac, and Linux, the extension framework only supports Linux containers. Therefore, you must target `linux` as the OS when you build your extension image.

□□□ #2
□ □□□ □□□ 2026-10-31 19:33:56 UTC

□ □□□ □□□ 2026-10-31 19:33:58 UTC