Multi-platform image with GitHub Actions

- Build and load multi-platform images
- Distribute build across multiple runners
  - With Bake

You can build multi-platform images using the `platforms` option, as shown in the following example:

> [!NOTE]
>
> - For a list of available platforms, see the Docker Setup Buildx action.
> - If you want support for more platforms, you can use QEMU with the Docker Setup QEMU action.

```
name: ci

on:
  push:

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      - name: Login to Docker Hub
        uses: docker/login-action@{{% param "login_action_version" %}}
        with:
          username: ${{ vars.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
```

```
      - name: Set up QEMU
        uses: docker/setup-qemu-action@{{% param "setup_qemu_action_version" %}}

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@{{% param "setup_buildx_action_version" %}}

      - name: Build and push
        uses: docker/build-push-action@{{% param "build_push_action_version" %}}
        with:
          platforms: linux/amd64,linux/arm64
          push: true
          tags: user/app:latest
```

Build and load multi-platform images

The default Docker setup for GitHub Actions runners supports building and pushing multi-platform images to registries. However, it does not support loading multi-platform images to the local image store of the runner after building them. To load a multi-platform image locally, you need to enable the containerd image store option for the Docker Engine.

There is no way to configure the default Docker setup in the GitHub Actions runners directly, but you can use `docker/setup-docker-action` to customize the Docker Engine and CLI settings for a job.

The following example workflow enables the containerd image store, builds a multi-platform image, and loads the results into the GitHub runner's local image store.

```
name: ci

on:
  push:

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

```
      - name: Set up Docker
        uses: docker/setup-docker-action@{{% param "setup_docker_action_version" %}}
        with:
          daemon-config: |
            {
              "debug": true,
              "features": {
                "containerd-snapshotter": true
              }
            }

      - name: Login to Docker Hub
        uses: docker/login-action@{{% param "login_action_version" %}}
        with:
          username: ${{ vars.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}

      - name: Set up QEMU
        uses: docker/setup-qemu-action@{{% param "setup_qemu_action_version" %}}

      - name: Build and push
        uses: docker/build-push-action@{{% param "build_push_action_version" %}}
        with:
          platforms: linux/amd64,linux/arm64
          load: true
          tags: user/app:latest
```

Distribute build across multiple runners

Building multiple platforms on the same runner can significantly extend build times, particularly when dealing with complex Dockerfiles or a high number of target platforms. If you want to split platform builds across multiple runners without maintaining a custom matrix and merge job, use the Docker GitHub Builder. The reusable workflows compute the per-platform matrix, run each platform on its own runner, and create the final manifest for you.

The following workflow uses the `build.yml` reusable workflow to distribute a multi-platform Dockerfile build:

```
name: ci

on:
  push:

permissions:
  contents: read

jobs:
  build:
    uses: docker/github-builder/.github/workflows/build.yml@{{% param "github_builder_version"
%}}
    permissions:
      contents: read
      id-token: write
    with:
      output: image
      push: true
      platforms: linux/amd64,linux/arm64
      meta-images: user/app
      meta-tags: |
        type=ref,event=branch
        type=ref,event=pr
        type=semver,pattern={{version}}
        type=semver,pattern={{major}}.{{minor}}
    secrets:
      registry-auths: |
        - registry: docker.io
          username: ${{ vars.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
```

With `distribute: true`, which is the default, the workflow splits the build into one platform per runner and assembles the final multi-platform image in its finalize phase. The default `runner` mapping sends Linux Arm platforms to `ubuntu-24.04-arm` and uses `ubuntu-24.04` for other platforms. To customize the mapping, see runner selection. If you need to control the Docker build inputs directly, see Build with Docker GitHub Builder build.yml.

With Bake

You can use the `bake.yml` reusable workflow for the same pattern when your build is defined in a Bake file. The workflow reads the target platforms from the Bake definition, distributes the per-platform builds, and publishes the final manifest without a separate prepare or merge job.

```
variable "DEFAULT_TAG" {
  default = "app:local"
}

// Special target: https://github.com/docker/metadata-action#bake-definition
target "docker-metadata-action" {
  tags = ["${DEFAULT_TAG}"]
}

// Default target if none specified
group "default" {
  targets = ["image-local"]
}

target "image" {
  inherits = ["docker-metadata-action"]
}

target "image-local" {
  inherits = ["image"]
  output = ["type=docker"]
}

target "image-all" {
  inherits = ["image"]
  platforms = [
    "linux/amd64",
    "linux/arm/v6",
    "linux/arm/v7",
    "linux/arm64"
  ]
}
```

```yaml
name: ci

on:
  push:

permissions:
  contents: read

jobs:
  bake:
    uses: docker/github-builder/.github/workflows/bake.yml@{{% param "github_builder_version"
%}}
    permissions:
      contents: read
      id-token: write
    with:
      output: image
      push: true
      target: image-all
      meta-images: user/app
      meta-tags: |
        type=ref,event=branch
        type=sha
    secrets:
      registry-auths: |
        - registry: docker.io
          username: ${{ vars.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
```

□□□ #2
□ □□□ □□□ 2026-10-31 19:05:33 UTC
□ □□□ □□□ 2026-10-31 19:05:34 UTC