dockercon 23

Streamlining CI/CD with Modern Container Builds

Kevin Alvarez

Software Engineer | Docker

GitHub Actions
Introduction and key features

Docker Actions
Current state of our official actions

Basic workflow
A simple workflow to get started

Leverage cache
Cache management for fast builds

Multi-platform build
Create container images for different architectures

Automating tags/labels
Strategies to manage and automate tags and labels based on CI events

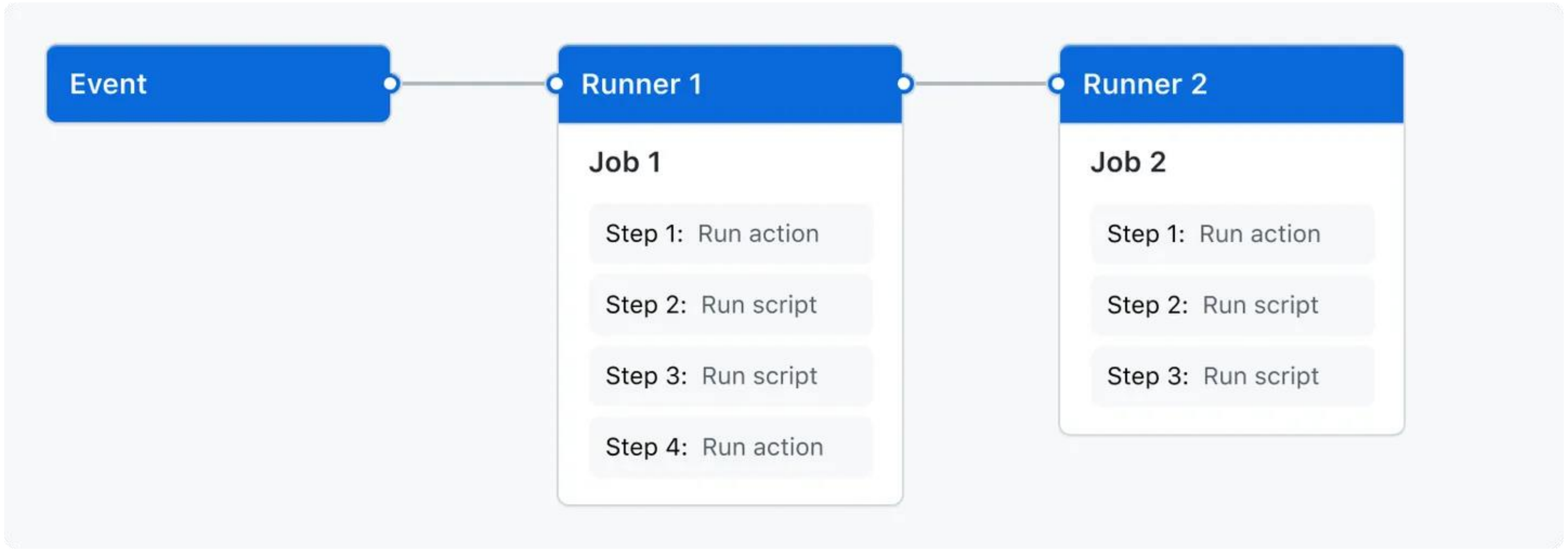
Bake
Unleash the power of Bake to simplify your development workflows

Secrets
Securely integrate build-time secrets within your workflow

Container-based workflow
Explore the vast potential of using containers in CI

GitHub Actions

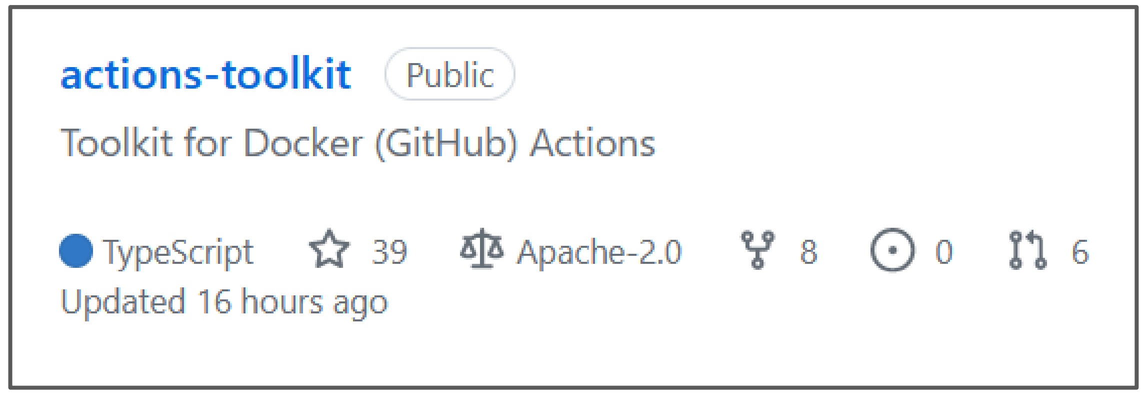
Key features



- **Workflow**: A set of automated steps defined in YAML files
- **Events**: Workflows are triggered by events (e.g., push, pull request, workflow_dispatch)
- **Runners**: Agents that execute workflows on various platforms
- **Jobs**: Units of work within a workflow
- **Actions**: Individual tasks within a workflow

More info: https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions/

Docker Actions

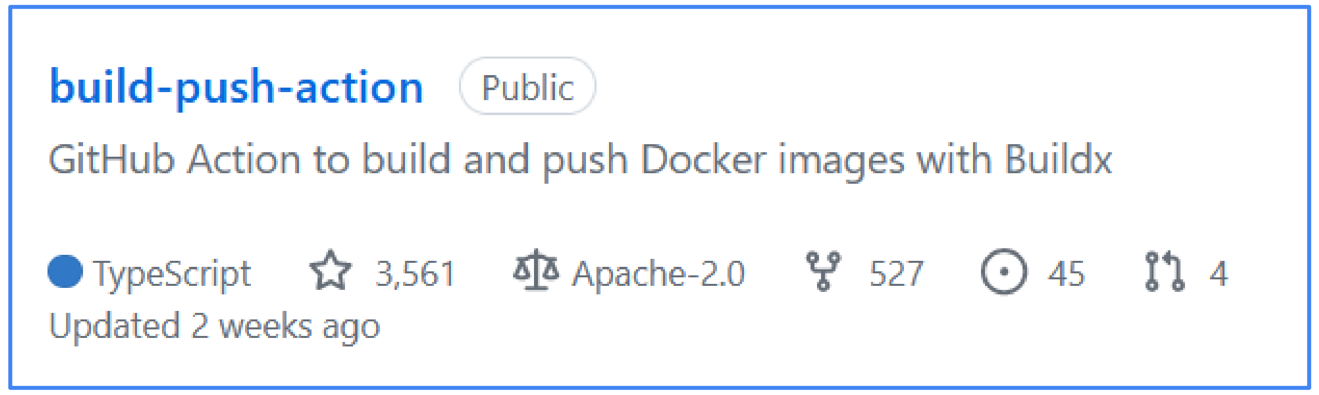


actions-toolkit Public

Toolkit for Docker (GitHub) Actions

TypeScript 39 Apache-2.0 8 0 6

Updated 16 hours ago

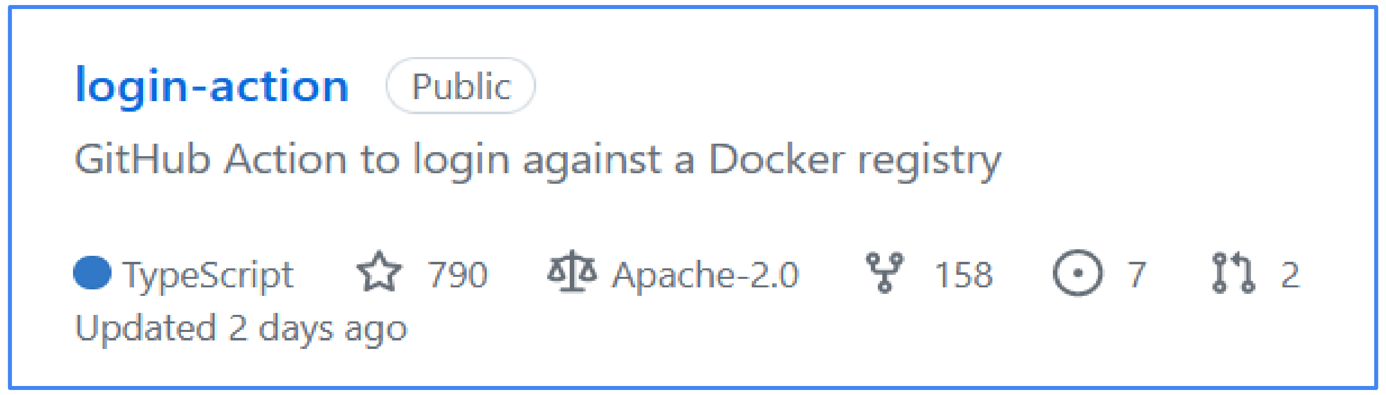


build-push-action Public

GitHub Action to build and push Docker images with Buildx

TypeScript 3,561 Apache-2.0 527 45 4

Updated 2 weeks ago

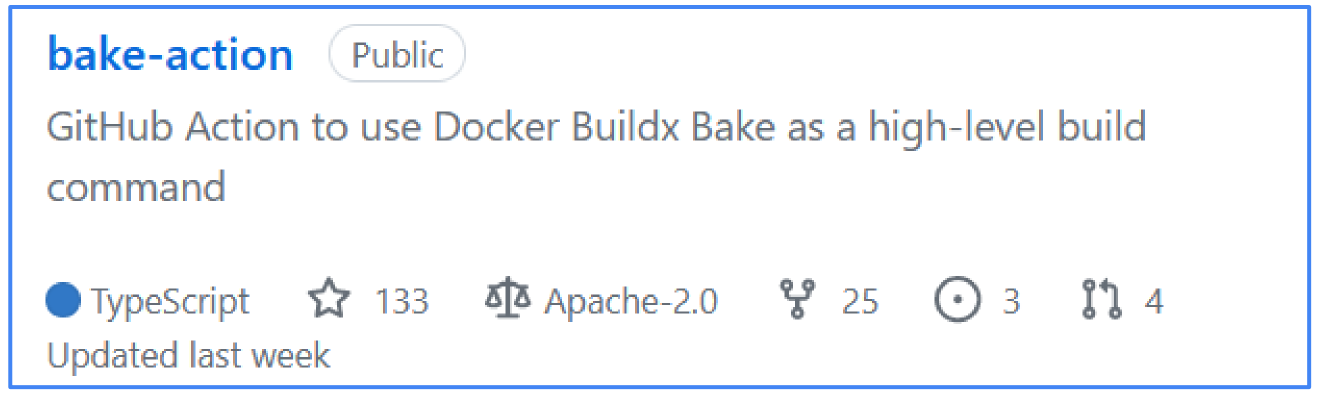


login-action Public

GitHub Action to login against a Docker registry

TypeScript 790 Apache-2.0 158 7 2

Updated 2 days ago

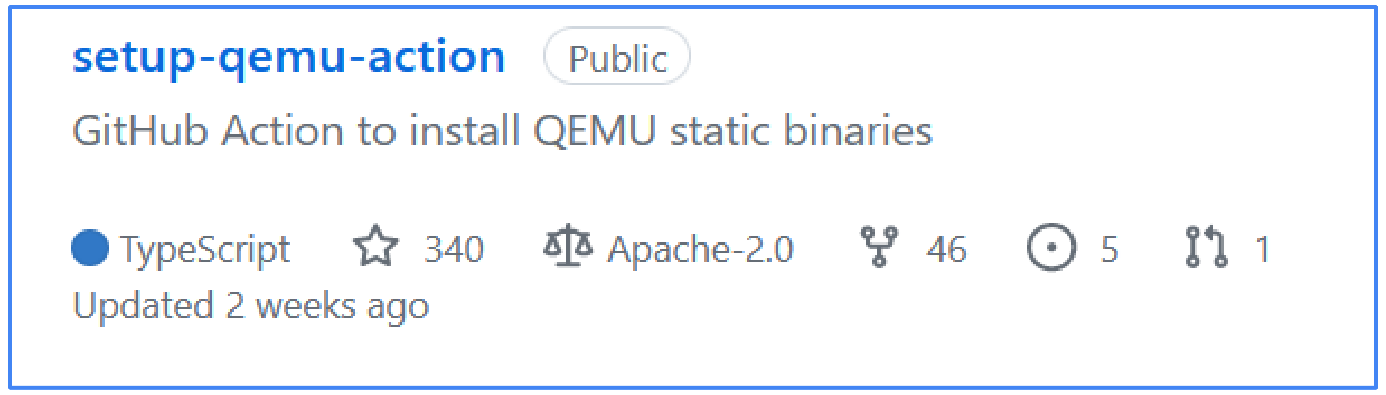


bake-action Public

GitHub Action to use Docker Buildx Bake as a high-level build command

TypeScript 133 Apache-2.0 25 3 4

Updated last week

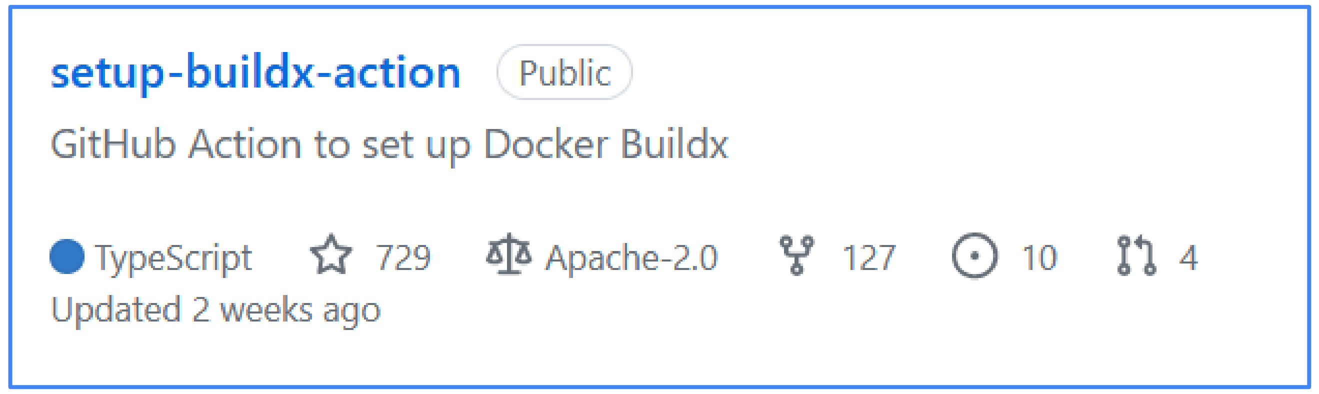


setup-qemu-action Public

GitHub Action to install QEMU static binaries

TypeScript 340 Apache-2.0 46 5 1

Updated 2 weeks ago

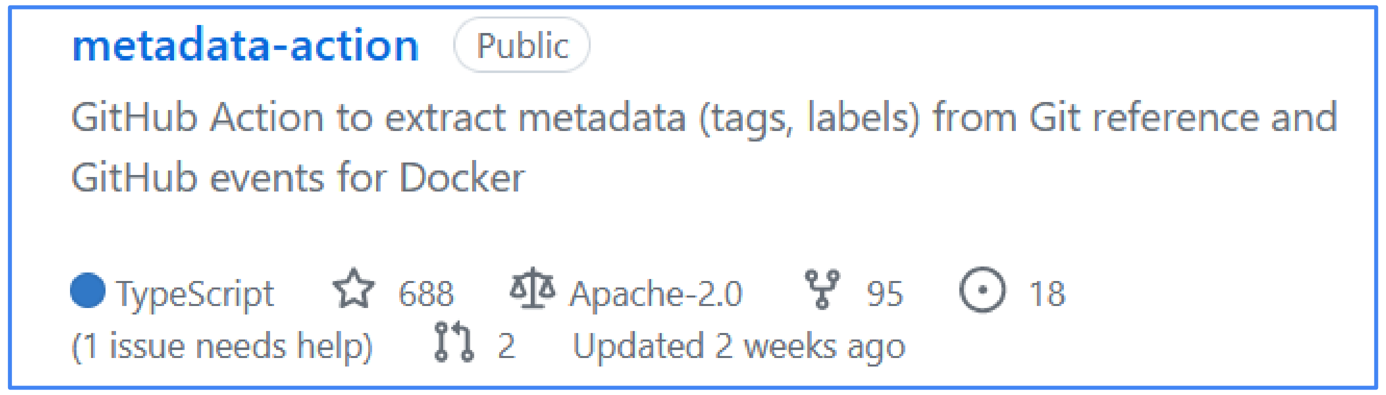


setup-buildx-action Public

GitHub Action to set up Docker Buildx

TypeScript 729 Apache-2.0 127 10 4

Updated 2 weeks ago



metadata-action Public

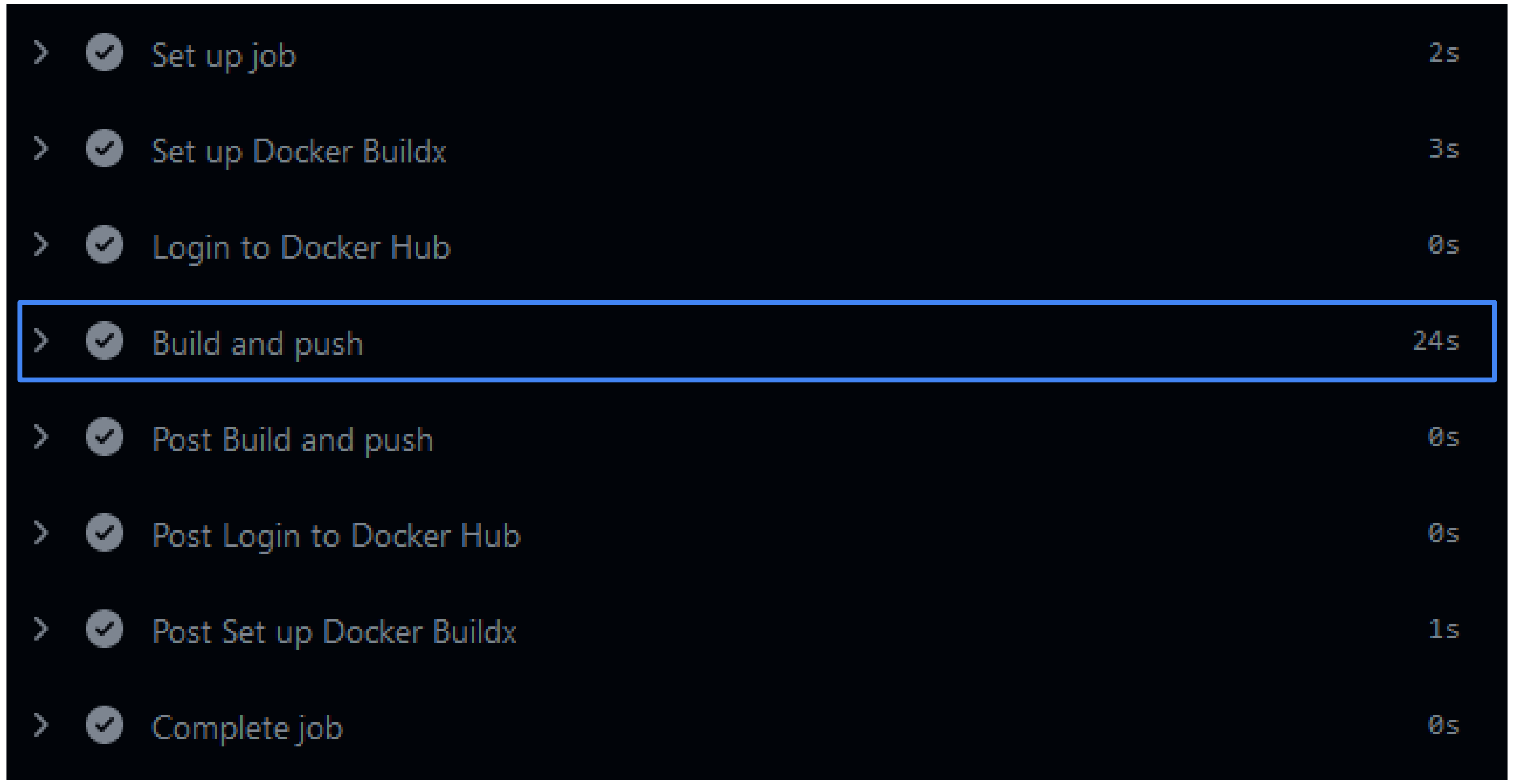
GitHub Action to extract metadata (tags, labels) from Git reference and GitHub events for Docker

TypeScript 688 Apache-2.0 95 18 (1 issue needs help) 2 Updated 2 weeks ago

Basic workflow

```yaml
name: ci

on:
  workflow_dispatch:

env:
  IMAGE_NAME: "crazymax/dockercon2023"

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      -
        name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      -
        name: Build and push
        uses: docker/build-push-action@v5
        with:
          push: true
          tags: ${{ env.IMAGE_NAME }}:latest
```

```dockerfile
# syntax=docker/dockerfile:1

FROM golang:1.20-alpine AS build
WORKDIR /src
RUN --mount=type=bind,target=. \
  go build -ldflags "-s -w" -o /usr/bin/app .

FROM alpine
COPY --from=build /usr/bin/app /usr/bin/app
CMD ["app"]
```



```
> ✓ Set up job                  2s
> ✓ Set up Docker Buildx        3s
> ✓ Login to Docker Hub         0s
> ✓ Build and push             24s
> ✓ Post Build and push         0s
> ✓ Post Login to Docker Hub    0s
> ✓ Post Set up Docker Buildx   1s
> ✓ Complete job                0s
```

```yaml
name: ci

on:
  workflow_dispatch:

env:
  IMAGE_NAME: "crazymax/dockercon2023"

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      -
        name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      -
        name: Build and push
        uses: docker/build-push-action@v5
        with:
          push: true
          tags: ${{ env.IMAGE_NAME }}:latest
```

```dockerfile
# syntax=docker/dockerfile:1

FROM golang:1.20-alpine AS build
WORKDIR /src
RUN --mount=type=bind,target=. \
  go build -ldflags "-s -w" -o /usr/bin/app .

FROM alpine
COPY --from=build /usr/bin/app /usr/bin/app
CMD ["app"]
```

```
#10 [build 1/3] FROM docker.io/library/golang:1.20-
alpine@sha256:c63dbdb3cca37abbee4c50f61e34b1d043c2669d03f34485f9ee6fe5feed4e48
#10 sha256:c4d48a809fc2256f8aa0aeee47998488d64409855adba00a7cb3007ab9f3286e 284.69kB /
284.69kB 0.1s done
#10 ...
#10 DONE 4.0s

#11 [build 2/3] WORKDIR /src
#11 DONE 1.3s

#12 [build 3/3] RUN --mount=type=bind,target=.   go build -ldflags "-s -w" -o /usr/bin/app .
#12 DONE 12.7s

#13 [stage-1 2/2] COPY --from=build /usr/bin/app /usr/bin/app
#13 DONE 0.0s

...
```

Leverage cache

BuildKit cache backends

inline
writes cache metadata into the image configuration

registry
exports build cache to a cache manifest in the registry

local
exports cache to a local directory on the client

s3
exports cache to an AWS S3 bucket

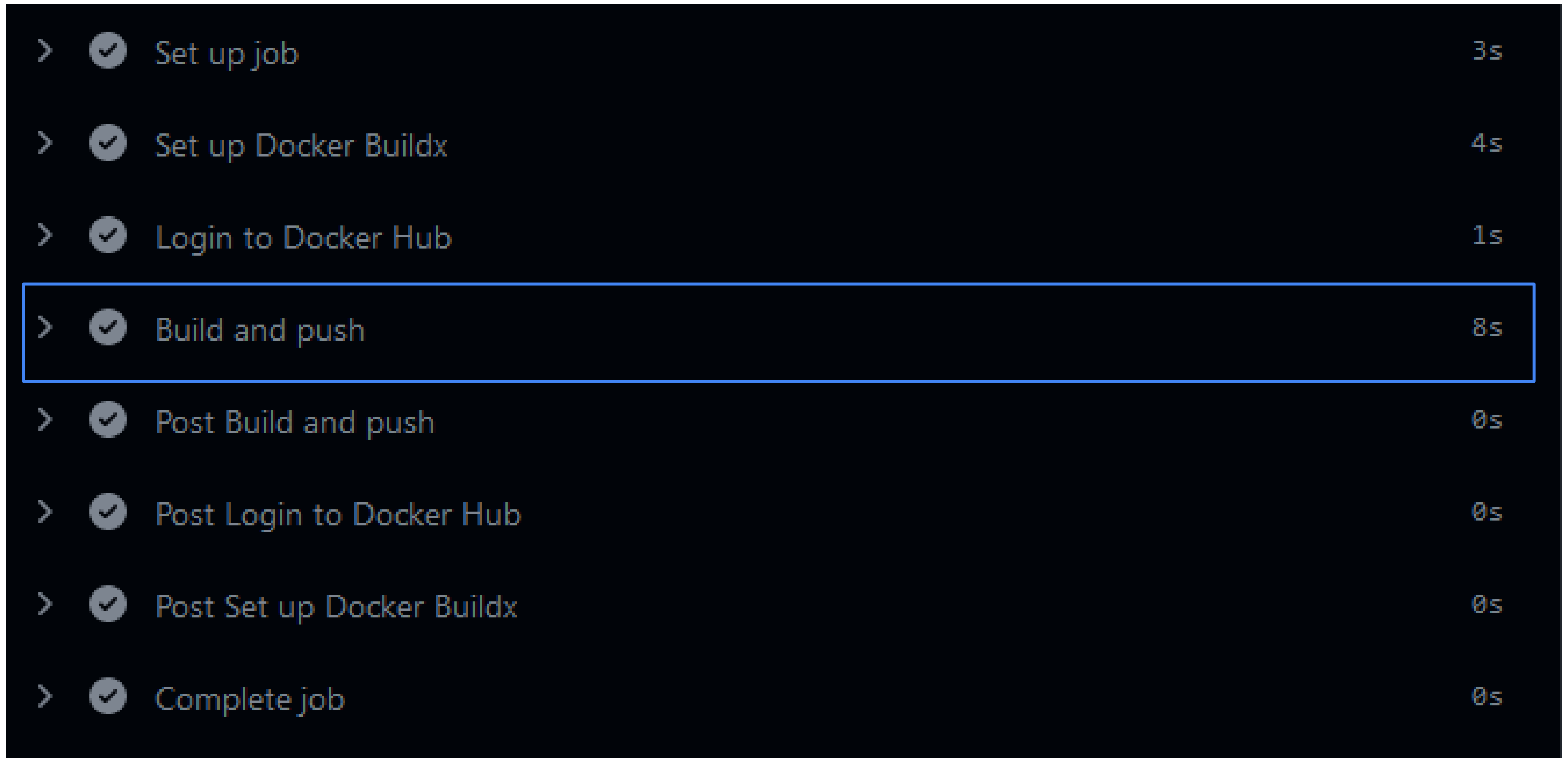
azblob
exports cache to Azure Blob storage

gha
exports cache through the GitHub Actions Cache service API

More info: https://docs.docker.com/build/cache/backends/

Set up GitHub Cache backend

```yaml
name: ci

on:
  workflow_dispatch:

env:
  IMAGE_NAME: "crazymax/dockercon2023"

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      -
        name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3
      -
        name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      -
        name: Build and push
        uses: docker/build-push-action@v5
        with:
          push: true
          tags: ${{ env.IMAGE_NAME }}:latest
          cache-from: type=gha
          cache-to: type=gha
```



```
Set up job                3s
Set up Docker Buildx      4s
Login to Docker Hub       1s
Build and push            8s
Post Build and push       0s
Post Login to Docker Hub  0s
Post Set up Docker Buildx 0s
Complete job              0s
```

Set up GitHub Cache backend

```yaml
name: ci

on:
  workflow_dispatch:

env:
  IMAGE_NAME: "crazymax/dockercon2023"

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      -
        name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3
      -
        name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      -
        name: Build and push
        uses: docker/build-push-action@v5
        with:
          push: true
          tags: ${{ env.IMAGE_NAME }}:latest
          cache-from: type=gha
          cache-to: type=gha
```

```
#10 [build 1/3] FROM docker.io/library/golang:1.20-
alpine@sha256:c63dbdb3cca37abbee4c50f61e34b1d043c2669d03f34485f9ee6fe5feed4e48
#10 resolve docker.io/library/golang:1.20-
alpine@sha256:c63dbdb3cca37abbee4c50f61e34b1d043c2669d03f34485f9ee6fe5feed4e48 done
#10 DONE 0.0s

#11 importing cache manifest from gha:2138736114910800190
#11 DONE 0.6s

#12 [build 2/3] WORKDIR /src
#12 CACHED

#13 [build 3/3] RUN --mount=type=bind,target=.   go build -ldflags "-s -w" -o /usr/bin/app .
#13 CACHED

#14 [stage-1 2/2] COPY --from=build /usr/bin/app /usr/bin/app
#14 CACHED

...
```

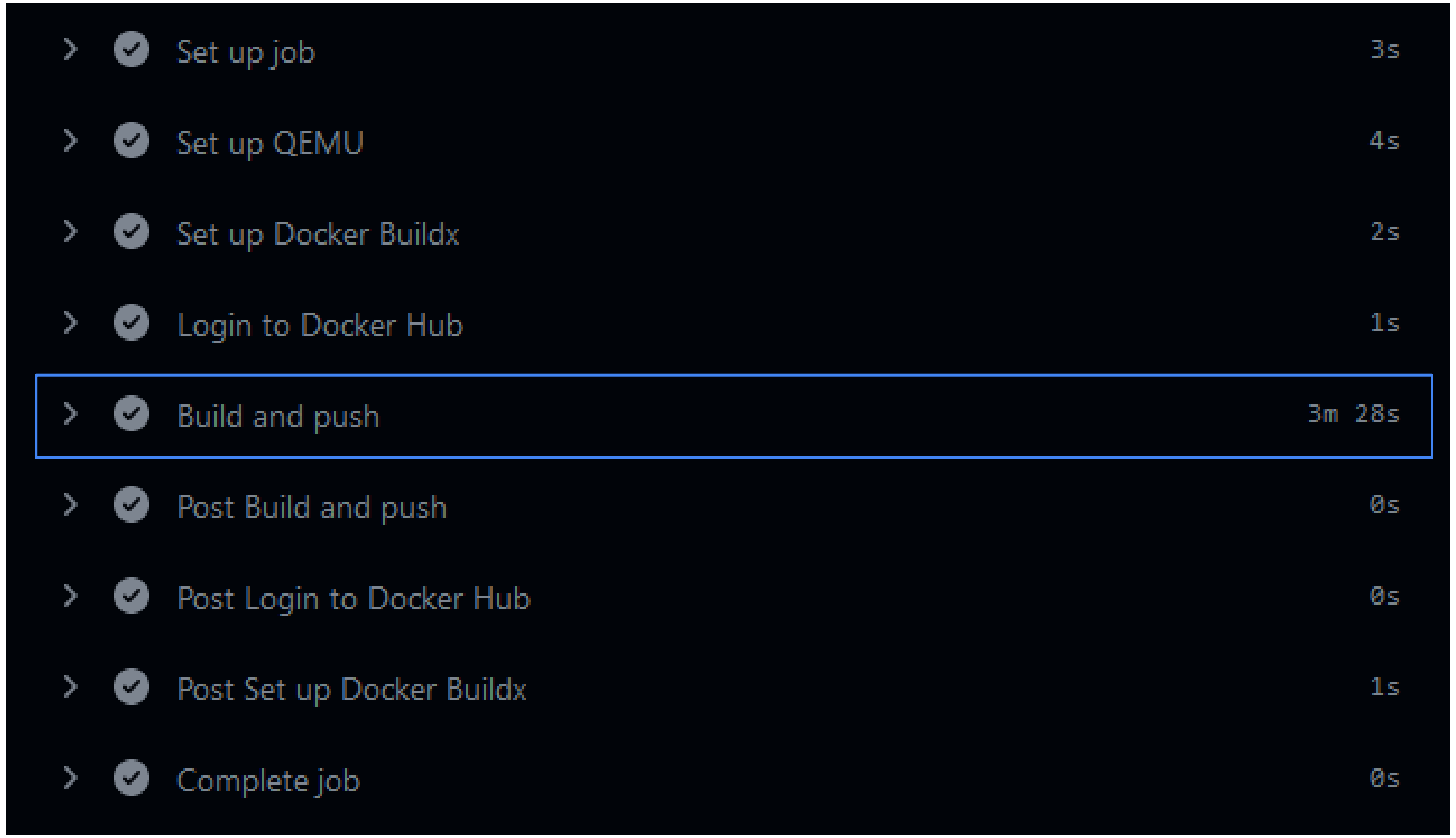
Advanced cache usage

- Cache has a **size limit of 10GB** shared across different caches in your repo.
- You can define a **scope** key for your cache to define where it belongs to.
- You can export all layers and intermediate steps with the **mode=max** option.
- Cache mounts are currently **not exported**. You can use a workaround provided by reproducible-containers/buildkit-cache-dance action.

Multi-platform build

Build multi-platform image

```yaml
...

env:
  IMAGE_NAME: "crazymax/dockercon2023"

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      -
        name: Set up Docker QEMU
        uses: docker/setup-buildx-qemu@v3
      -
        name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3
      -
        name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      -
        name: Build and push
        uses: docker/build-push-action@v5
        with:
          push: true
          tags: ${{ env.IMAGE_NAME }}:latest
          cache-from: type=gha
          cache-to: type=gha
          platforms: linux/amd64,linux/arm64
```



```
Set up job                 3s
Set up QEMU                4s
Set up Docker Buildx       2s
Login to Docker Hub        1s
Build and push         3m 28s
Post Build and push        0s
Post Login to Docker Hub   0s
Post Set up Docker Buildx  1s
Complete job               0s
```

Build multi-platform image

```yaml
...

env:
  IMAGE_NAME: "crazymax/dockercon2023"

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      -
        name: Set up Docker QEMU
        uses: docker/setup-buildx-qemu@v3
      -
        name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3
      -
        name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      -
        name: Build and push
        uses: docker/build-push-action@v5
        with:
          push: true
          tags: ${{ env.IMAGE_NAME }}:latest
          cache-from: type=gha
          cache-to: type=gha
          platforms: linux/amd64,linux/arm64
```

```
#17 [linux/amd64 build 2/3] WORKDIR /src
#17 DONE 1.3s

#16 [linux/arm64 build 2/3] WORKDIR /src
#16 DONE 2.7s

#18 [linux/amd64 build 3/3] RUN --mount=type=bind,target=.   go build -ldflags "-s -w" -o /usr/bin/app .
#18 ...

#19 [linux/arm64 build 3/3] RUN --mount=type=bind,target=.   go build -ldflags "-s -w" -o /usr/bin/app .
#19 ...

#18 [linux/amd64 build 3/3] RUN --mount=type=bind,target=.   go build -ldflags "-s -w" -o /usr/bin/app .
#18 DONE 27.5s

#14 [linux/amd64 stage-1 1/2] FROM
docker.io/library/alpine:latest@sha256:7144f7bab3d4c2648d7e59409f15ec52a18006a128c733fcff20d3a4a54ba44a
#14 DONE 0.0s

#20 [linux/amd64 stage-1 2/2] COPY --from=build /usr/bin/app /usr/bin/app
#20 DONE 0.0s

#19 [linux/arm64 build 3/3] RUN --mount=type=bind,target=.   go build -ldflags "-s -w" -o /usr/bin/app .
#19 DONE 187.1s

#21 [linux/arm64 stage-1 2/2] COPY --from=build /usr/bin/app /usr/bin/app
#21 DONE 0.0s
```

Dockerfile and cross compilation

```
# syntax=docker/dockerfile:1

FROM golang:1.20-alpine AS build
WORKDIR /src
RUN --mount=type=bind,target=. \
  go build -ldflags "-s -w" -o /usr/bin/app .

FROM alpine
COPY --from=build /usr/bin/app /usr/bin/app
CMD ["app"]
```

Automatic platform ARGs in the global scope

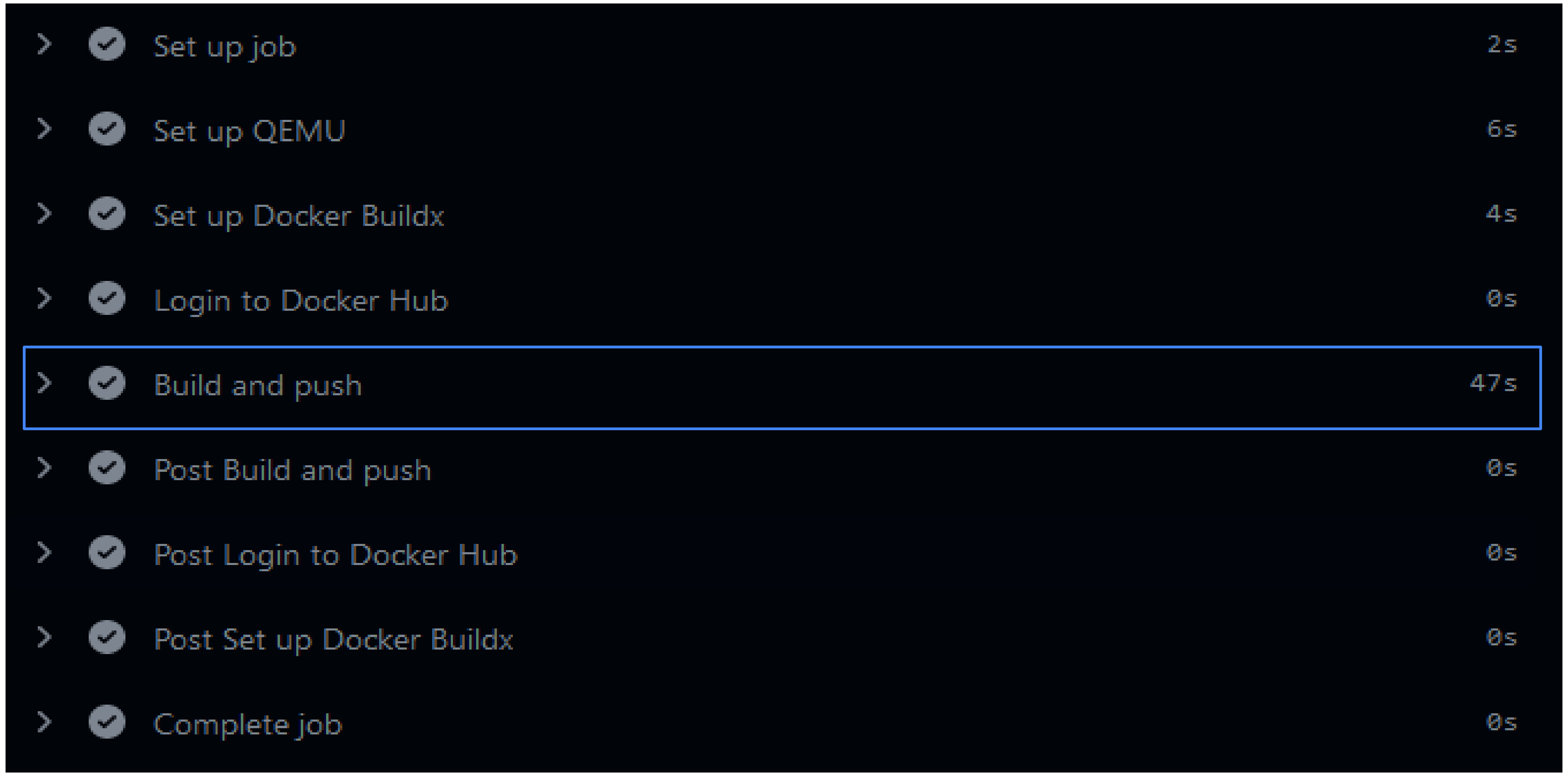
- **TARGETPLATFORM** - platform of the build result. (e.g, linux/amd64, linux/arm/v7).
- **TARGETOS** - OS component of TARGETPLATFORM
- **TARGETARCH** - architecture component of TARGETPLATFORM
- **TARGETVARIANT** - variant component of TARGETPLATFORM
- **BUILDPLATFORM** - platform of the node performing the build.
- **BUILDOS** - OS component of BUILDPLATFORM
- **BUILDARCH** - architecture component of BUILDPLATFORM
- **BUILDVARIANT** - variant component of BUILDPLATFORM

```
# syntax=docker/dockerfile:1

FROM --platform=$BUILDPLATFORM golang:1.20-alpine AS build
WORKDIR /src
ARG TARGETOS TARGETARCH
RUN --mount=type=bind,target=. \
  GOOS=$TARGETOS GOARCH=$TARGETARCH go build -ldflags "-s -w" -o /usr/bin/app .

FROM alpine
COPY --from=build /usr/bin/app /usr/bin/app
CMD ["app"]
```

More info: https://medium.com/@tonistiigi/faster-multi-platform-builds-dockerfile-cross-compilation-guide-part-1-ec087c719eaf

Build multi-platform image (with cross-comp!)

```yaml
...

env:
  IMAGE_NAME: "crazymax/dockercon2023"

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      -
        name: Set up Docker QEMU
        uses: docker/setup-buildx-qemu@v3
      -
        name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3
      -
        name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      -
        name: Build and push
        uses: docker/build-push-action@v5
        with:
          push: true
          tags: ${{ env.IMAGE_NAME }}:latest
          cache-from: type=gha
          cache-to: type=gha
          platforms: linux/amd64,linux/arm64
```



```
Set up job                 2s
Set up QEMU                6s
Set up Docker Buildx       4s
Login to Docker Hub        0s
Build and push            47s
Post Build and push        0s
Post Login to Docker Hub   0s
Post Set up Docker Buildx  0s
Complete job               0s
```

Build multi-platform image (with cross-comp!)

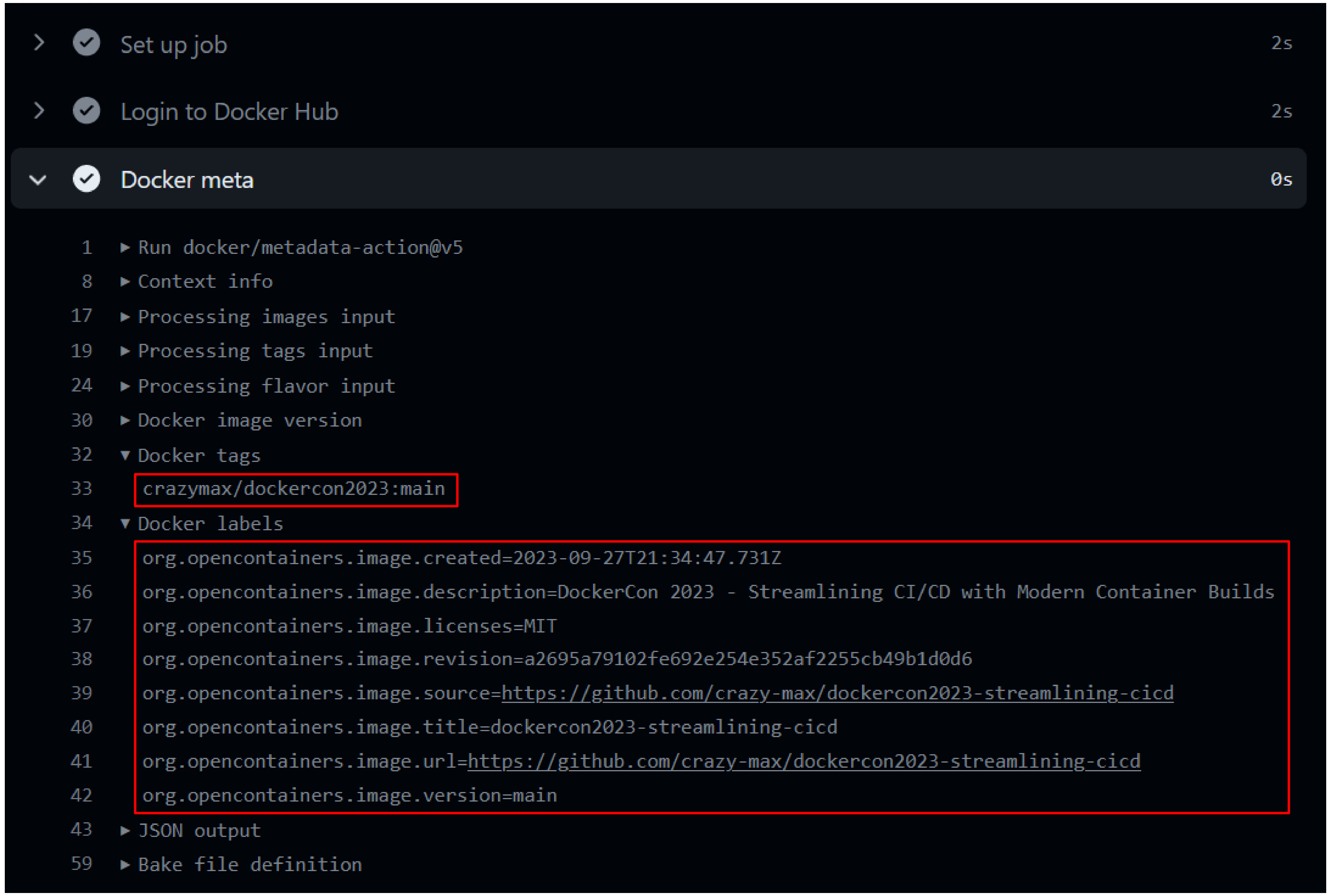
```yaml
...

env:
  IMAGE_NAME: "crazymax/dockercon2023"

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      -
        name: Set up Docker QEMU
        uses: docker/setup-buildx-qemu@v3
      -
        name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3
      -
        name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      -
        name: Build and push
        uses: docker/build-push-action@v5
        with:
          push: true
          tags: ${{ env.IMAGE_NAME }}:latest
          cache-from: type=gha
          cache-to: type=gha
          platforms: linux/amd64,linux/arm64
```

```
#14 [linux/amd64 build 2/3] WORKDIR /src
#14 DONE 1.2s

#15 [linux/amd64 build 3/3] RUN --mount=type=bind,target=.   GOOS=linux GOARCH=amd64 go build -ldflags
"-s -w" -o /usr/bin/app .
#15 ...

#16 [linux/amd64->arm64 build 3/3] RUN --mount=type=bind,target=.   GOOS=linux GOARCH=arm64 go build -
ldflags "-s -w" -o /usr/bin/app .
#16 DONE 31.8s

#15 [linux/amd64 build 3/3] RUN --mount=type=bind,target=.   GOOS=linux GOARCH=amd64 go build -ldflags
"-s -w" -o /usr/bin/app .
#15 DONE 31.9s

#17 [linux/arm64 stage-1 2/2] COPY --from=build /usr/bin/app /usr/bin/app
#17 DONE 0.0s

#18 [linux/amd64 stage-1 2/2] COPY --from=build /usr/bin/app /usr/bin/app
#18 DONE 0.0s
```

Automating tags/labels

```yaml
on:
  push:
    branches:
      - "main"
    tags:
      - "v*"
  pull_request:

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      -
        name: Login to Docker Hub
        if: github.event_name != 'pull_request'
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      -
        name: Docker meta
        id: meta
        uses: docker/metadata-action@v5
        with:
          images: crazymax/dockercon2023
      -
        name: Build and push
        uses: docker/build-push-action@v5
        with:
          push: ${{ github.event_name != 'pull_request' }}
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}
```

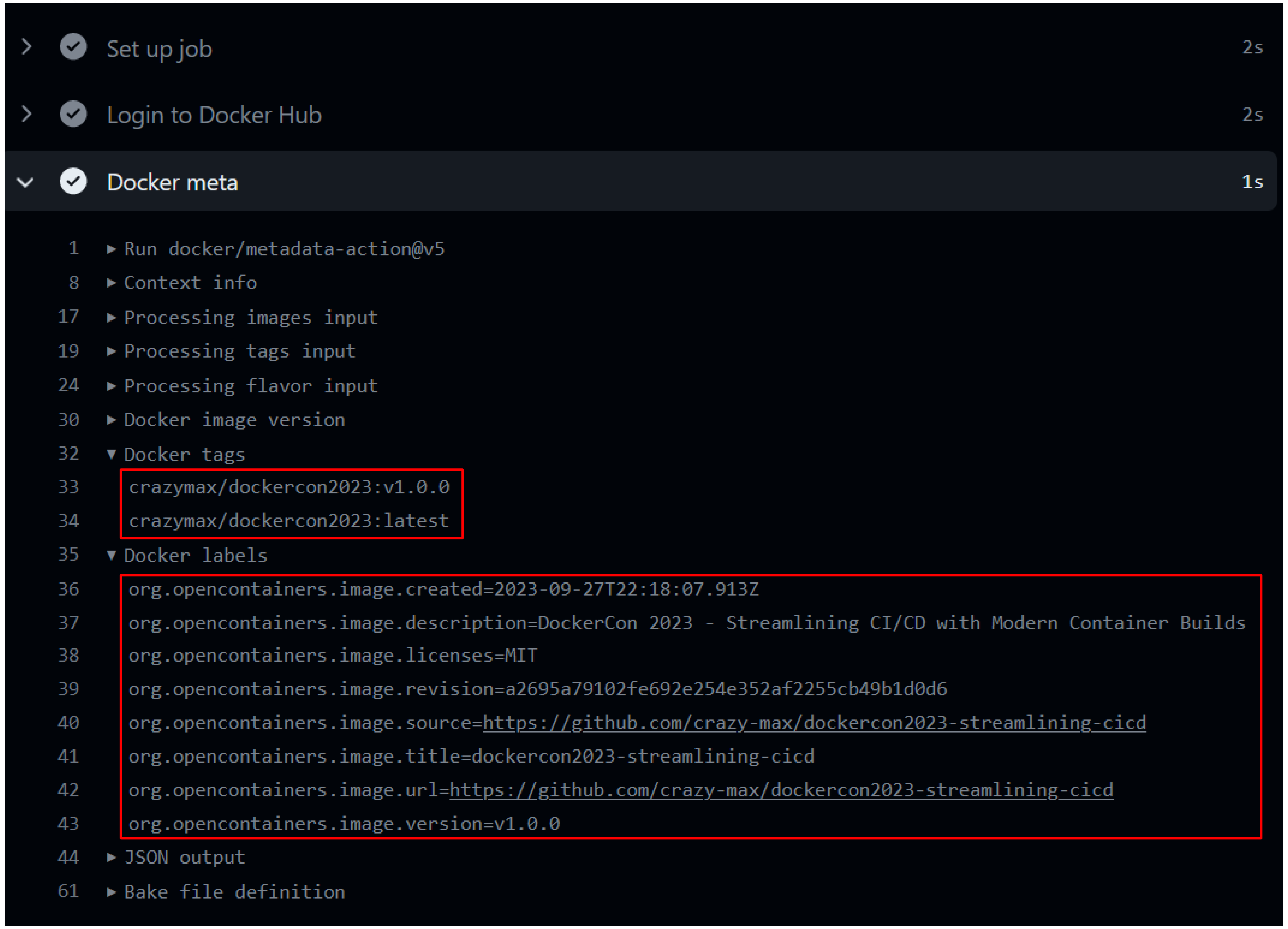
Push to main



```
▶ Run docker/metadata-action@v5
▶ Context info
▶ Processing images input
▶ Processing tags input
▶ Processing flavor input
▶ Docker image version
▼ Docker tags
  crazymax/dockercon2023:main
▼ Docker labels
  org.opencontainers.image.created=2023-09-27T21:34:47.731Z
  org.opencontainers.image.description=DockerCon 2023 - Streamlining CI/CD with Modern Container Builds
  org.opencontainers.image.licenses=MIT
  org.opencontainers.image.revision=a2695a79102fe692e254e352af2255cb49b1d0d6
  org.opencontainers.image.source=https://github.com/crazy-max/dockercon2023-streamlining-cicd
  org.opencontainers.image.title=dockercon2023-streamlining-cicd
  org.opencontainers.image.url=https://github.com/crazy-max/dockercon2023-streamlining-cicd
  org.opencontainers.image.version=main
▶ JSON output
▶ Bake file definition
```

```yaml
on:
  push:
    branches:
      - "main"
    tags:
      - "v*"
  pull_request:

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      -
        name: Login to Docker Hub
        if: github.event_name != 'pull_request'
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      -
        name: Docker meta
        id: meta
        uses: docker/metadata-action@v5
        with:
          images: crazymax/dockercon2023
      -
        name: Build and push
        uses: docker/build-push-action@v5
        with:
          push: ${{ github.event_name != 'pull_request' }}
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}
```

Push v1.0.0 tag



```
Set up job                2s
Login to Docker Hub       2s
Docker meta               1s

▶ Run docker/metadata-action@v5
▶ Context info
▶ Processing images input
▶ Processing tags input
▶ Processing flavor input
▶ Docker image version
▼ Docker tags
  crazymax/dockercon2023:v1.0.0
  crazymax/dockercon2023:latest
▼ Docker labels
  org.opencontainers.image.created=2023-09-27T22:18:07.913Z
  org.opencontainers.image.description=DockerCon 2023 - Streamlining CI/CD with Modern Container Builds
  org.opencontainers.image.licenses=MIT
  org.opencontainers.image.revision=a2695a79102fe692e254e352af2255cb49b1d0d6
  org.opencontainers.image.source=https://github.com/crazy-max/dockercon2023-streamlining-cicd
  org.opencontainers.image.title=dockercon2023-streamlining-cicd
  org.opencontainers.image.url=https://github.com/crazy-max/dockercon2023-streamlining-cicd
  org.opencontainers.image.version=v1.0.0
▶ JSON output
▶ Bake file definition
```

Customizing tags

```yaml
-
  name: Docker meta
  id: meta
  uses: docker/metadata-action@v5
  with:
    images: crazymax/dockercon2023
```

Default value for « tags » input

```yaml
tags: |
  type=schedule
  type=ref,event=branch
  type=ref,event=tag
  type=ref,event=pr
```

```yaml
-
  name: Docker meta
  id: meta
  uses: docker/metadata-action@v5
  with:
    images: crazymax/dockercon2023
    tags: |
      type=ref,event=branch
      type=ref,event=pr
      type=semver,pattern={{version}}
      type=semver,pattern={{major}}.{{minor}}
      type=sha
```

Push v1.0.0 tag

- **crazymax/dockercon2023:1.0.0**
- **crazymax/dockercon2023:1.0**
- **crazymax/dockercon2023:latest**
- **crazymax/dockercon2023:sha-ad132fa**

Open pull request #186

- **crazymax/dockercon2023:pr-186**
- **crazymax/dockercon2023:sha-b84ffa3**

More info: https://github.com/docker/metadata-action#tags-input

Bake

Build command

```
docker buildx build \
    --push \
    --cache-from "type=registry,ref=foo/myapp" \
    --cache-to "type=inline" \
    --platform "linux/amd64,linux/arm64" \
    --label "org.opencontainers.image.title=myapp" \
    --label "org.opencontainers.image.source=https://github.com/foo/myapp" \
    --label "org.opencontainers.image.version=1.0.0" \
    --label "org.opencontainers.image.licenses=Apache-2.0" \
    --tag "foo/myapp:v1.0.0" \
    --tag "foo/myapp:latest" \
    --file "./main.Dockerfile" \
    .
```

In your workflow

```yaml
-
  name: Build and push
  uses: docker/build-push-action@v5
  with:
    context: .
    push: true
    file: ./main.Dockerfile
    cache-from: type=registry,ref=foo/myapp
    cache-to: type=inline
    platforms: linux/amd64,linux/arm64
    tags: |
      foo/myapp:v1.0.0
      foo/myapp:latest
    labels: |
      org.opencontainers.image.title=myapp
      org.opencontainers.image.source=https://github.com/foo/myapp
      org.opencontainers.image.version=1.0.0
      org.opencontainers.image.licenses=Apache-2.0
```

With Bake

```
docker buildx bake \
    --push
    ~~--cache-from "type=registry,ref=foo/myapp" \~~
    ~~--cache-to "type=inline" \~~
    ~~--platform "linux/amd64,linux/arm64" \~~
    ~~--label "org.opencontainers.image.title=myapp" \~~
    ~~--label "org.opencontainers.image.source=https://github.com/foo/myapp" \~~
    ~~--label "org.opencontainers.image.version=1.0.0" \~~
    ~~--label "org.opencontainers.image.licenses=Apache-2.0" \~~
    ~~--tag "foo/myapp:v1.0.0" \~~
    ~~--tag "foo/myapp:latest" \~~
    ~~--file "./main.Dockerfile" \~~
    ~~.~~
```

Bake definition

- Can be a **HCL** 💜, JSON or Compose file
- Multiple files can be specified with `--file` flag
    - Configuration are **merged**
    - Defaults to:
        - **HCL**: `docker-bake.override.hcl > docker-bake.hcl`
        - **JSON**: `docker-bake.override.json > docker-bake.json`
        - **Compose**: `compose.yaml > compose.yml > docker-compose.yaml > docker-compose.yml`
- `variable` block definition
    - use as **build arguments** in your Dockerfile
    - interpolate them in attribute values in your Bake file
- `function` block definition
    - A set of general-purpose functions provided by go-cty are available
    - User defined functions are also supported

More info: https://docs.docker.com/build/bake/reference/

HCL format

docker buildx bake image-all

```
group "default" {
  targets = ["image"]
}

variable "TAG" {
  default = "v1.0.0"
}

target "image" {
  context = "."
  dockerfile = "./main.Dockerfile"
  cache-from = ["type=registry,ref=foo/myapp"]
  cache-to = ["type=inline"]
  labels = {
    "org.opencontainers.image.title" = "myapp"
    "org.opencontainers.image.ref" = "https://github.com/foo/myapp"
  }
  tags = ["foo/myapp:latest", "foo/myapp:${TAG}"]
}

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

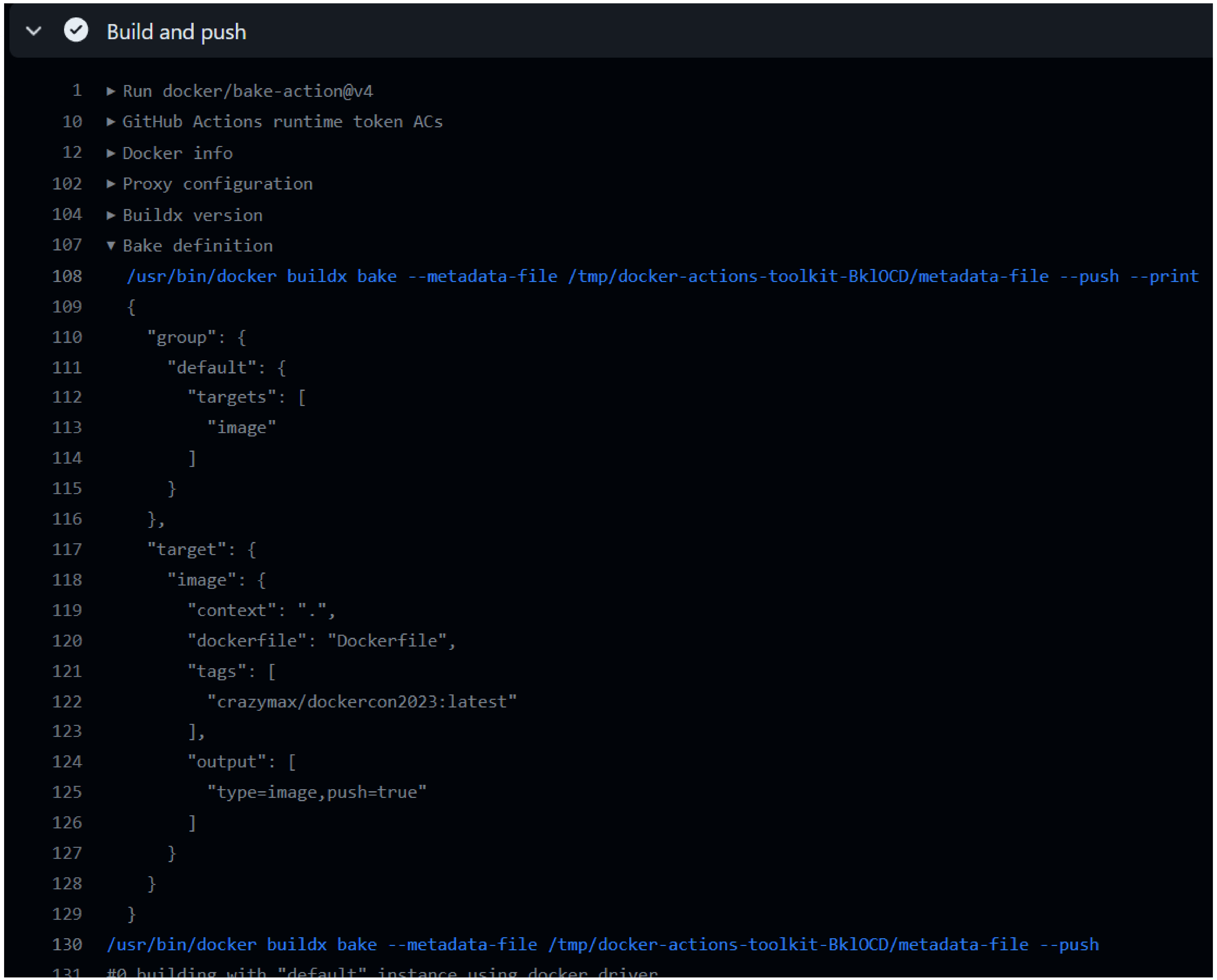
Bake action

```
# docker-bake.hcl

group "default" {
  targets = ["image"]
}

variable "IMAGE_NAME" {
  default = "foo/bar"
}

target "image" {
  tags = ["${IMAGE_NAME}:latest"]
}
```

```
name: ci

on:
  workflow_dispatch:

env:
  IMAGE_NAME: "crazymax/dockercon2023"

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      -
        name: Checkout
        uses: actions/checkout@v4
      -
        name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      -
        name: Build and push
        uses: docker/bake-action@v4
        with:
          push: true
```

Bake action

```hcl
# docker-bake.hcl

group "default" {
  targets = ["image"]
}

variable "IMAGE_NAME" {
  default = "foo/bar"
}

target "image" {
  tags = ["${IMAGE_NAME}:latest"]
}
```



Build and push

```
▶ Run docker/bake-action@v4
▶ GitHub Actions runtime token ACs
▶ Docker info
▶ Proxy configuration
▶ Buildx version
▼ Bake definition
  /usr/bin/docker buildx bake --metadata-file /tmp/docker-actions-toolkit-Bkl0CD/metadata-file --push --print
  {
    "group": {
      "default": {
        "targets": [
          "image"
        ]
      }
    },
    "target": {
      "image": {
        "context": ".",
        "dockerfile": "Dockerfile",
        "tags": [
          "crazymax/dockercon2023:latest"
        ],
        "output": [
          "type=image,push=true"
        ]
      }
    }
  }
/usr/bin/docker buildx bake --metadata-file /tmp/docker-actions-toolkit-Bkl0CD/metadata-file --push
#0 building with "default" instance using docker driver
```

Secrets

Testing our code

```dockerfile
# syntax=docker/dockerfile:1

FROM --platform=$BUILDPLATFORM golang:1.20-alpine AS base
WORKDIR /src
ENV CGO_ENABLED=0

FROM base AS test
RUN --mount=type=bind,target=. \
    go test ./...

FROM base AS build
ARG TARGETOS TARGETARCH
RUN --mount=type=bind,target=. \
    GOOS=$TARGETOS GOARCH=$TARGETARCH go build -ldflags "-s -w" -o /usr/bin/app .

FROM alpine
COPY --from=build /usr/bin/app /usr/bin/app
CMD ["app"]
```

```hcl
group "default" {
  targets = ["image"]
}

variable "IMAGE_NAME" {
  default = "foo/bar"
}

target "image" {
  tags = ["${IMAGE_NAME}:latest"]
}

target "test" {
  target = "test"
}
```

```yaml
-
  name: Test
  uses: docker/bake-action@v4
  with:
    targets: test
-
  name: Build and push
  uses: docker/bake-action@v4
  with:
    push: true
```

Using a build-time secret

```
# syntax=docker/dockerfile:1

FROM --platform=$BUILDPLATFORM golang:1.20-alpine AS base
WORKDIR /src
ENV CGO_ENABLED=0

FROM base AS test
RUN --mount=type=bind,target=. \
    --mount=type=secret,id=ghtoken \
    GITHUB_TOKEN=$(cat /run/secrets/ghtoken) go test ./...

FROM base AS build
ARG TARGETOS TARGETARCH
RUN --mount=type=bind,target=. \
    GOOS=$TARGETOS GOARCH=$TARGETARCH go build -ldflags "-s -w" -o /usr/bin/app .

FROM alpine
COPY --from=build /usr/bin/app /usr/bin/app
CMD ["app"]
```

```
group "default" {
  targets = ["image"]
}

variable "IMAGE_NAME" {
  default = "foo/bar"
}

target "image" {
  tags = ["${IMAGE_NAME}:latest"]
}

target "test" {
  target = "test"
  secret = ["id=ghtoken,env=GITHUB_TOKEN"]
}
```

```
-
  name: Test
  uses: docker/bake-action@v4
  with:
    targets: test
  env:
    GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
-
  name: Build and push
  uses: docker/bake-action@v4
  with:
    push: true
```

Container-based workflow

Validate our code

```hcl
# docker-bake.hcl

group "default" {
  targets = ["image"]
}

variable "IMAGE_NAME" {
  default = "foo/bar"
}

target "image" {
  tags = ["${IMAGE_NAME}:latest"]
}

target "test" {
  target = "test"
  secret = ["id=ghtoken,env=GITHUB_TOKEN"]
}

target "lint" {
  target = "lint"
}
```

```dockerfile
# syntax=docker/dockerfile:1

FROM --platform=$BUILDPLATFORM golang:1.20-alpine AS base
WORKDIR /src
ENV CGO_ENABLED=0

FROM base AS test
RUN --mount=type=bind,target=. \
    --mount=type=secret,id=ghtoken \
    GITHUB_TOKEN=$(cat /run/secrets/ghtoken) go test ./...

FROM base AS lint
RUN wget -O- -nv \
    https://raw.githubusercontent.com/golangci/golangci-lint/master/install.sh | sh -s v1.51.1
RUN --mount=type=bind,target=. \
    golangci-lint run

FROM base AS build
ARG TARGETOS TARGETARCH
RUN --mount=type=bind,target=. \
    GOOS=$TARGETOS GOARCH=$TARGETARCH go build -ldflags "-s -w" -o /usr/bin/app .

FROM alpine
COPY --from=build /usr/bin/app /usr/bin/app
CMD ["app"]
```

Updating our workflow

```
# docker-bake.hcl

group "default" {
  targets = ["image"]
}

variable "IMAGE_NAME" {
  default = "foo/bar"
}

target "image" {
  tags = ["${IMAGE_NAME}:latest"]
}

target "test" {
  target = "test"
  secret = ["id=ghtoken,env=GITHUB_TOKEN"]
}

target "lint" {
  target = "lint"
}
```

```
-
  name: Lint
  uses: docker/bake-action@v4
  with:
    targets: lint
-
  name: Test
  uses: docker/bake-action@v4
  with:
    targets: test
  env:
    GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
-
  name: Build and push
  uses: docker/bake-action@v4
  with:
    push: true
```

Lint step logs

```
#10 [lint 1/3] RUN wget -O- -nv https://raw.githubusercontent.com/golangci/golangci-lint/master/install.sh | sh -s
v1.51.1
#10 0.584 Connecting to raw.githubusercontent.com (185.199.110.133:443)
#10 0.909 writing to stdout
#10 0.909 -                    100% |********************************| 11114  0:00:00 ETA
#10 0.909 written to stdout
#10 0.930 golangci/golangci-lint info checking GitHub for tag 'v1.51.1'
#10 1.090 golangci/golangci-lint info found version: 1.51.1 for v1.51.1/linux/amd64
#10 1.821 golangci/golangci-lint info installed ./bin/golangci-lint
#10 DONE 1.9s

#11 [lint 2/3] WORKDIR /src
#11 DONE 0.0s

#12 [lint 3/3] RUN --mount=type=bind,target=.     golangci-lint run
#12 DONE 16.7s

#13 exporting to image
#13 exporting layers
#13 exporting layers 1.3s done
#13 writing image sha256:0a3fd3fc35b273b8111d2c5d674fa1736d348b808a22ce3aa33117b2d6b177a8 done
#13 DONE 1.3s
```

Don’t export build output

```
# docker-bake.hcl

group "default" {
  targets = ["image"]
}

variable "IMAGE_NAME" {
  default = "foo/bar"
}

target "image" {
  tags = ["${IMAGE_NAME}:latest"]
}

target "test" {
  target = "test"
  secret = ["id=ghtoken,env=GITHUB_TOKEN"]
  output = ["type=cacheonly"]
}

target "lint" {
  target = "lint"
  output = ["type=cacheonly"]
}
```

```
#10 [lint 1/3] RUN wget -O- -nv https://raw.githubusercontent.com/golangci/golangci-
lint/master/install.sh | sh -s v1.51.1
#10 0.209 Connecting to raw.githubusercontent.com (185.199.110.133:443)
#10 0.282 writing to stdout
#10 0.283 -                    100% |********************************| 11114  0:00:00
ETA
#10 0.283 written to stdout
#10 0.290 golangci/golangci-lint info checking GitHub for tag 'v1.51.1'
#10 0.499 golangci/golangci-lint info found version: 1.51.1 for v1.51.1/linux/amd64
#10 1.549 golangci/golangci-lint info installed ./bin/golangci-lint
#10 DONE 1.6s

#11 [lint 2/3] WORKDIR /src
#11 DONE 0.0s

#12 [lint 3/3] RUN --mount=type=bind,target=.     golangci-lint run
#12 DONE 15.9s
```

More info: https://docs.docker.com/build/exporters/

Simplify our workflow

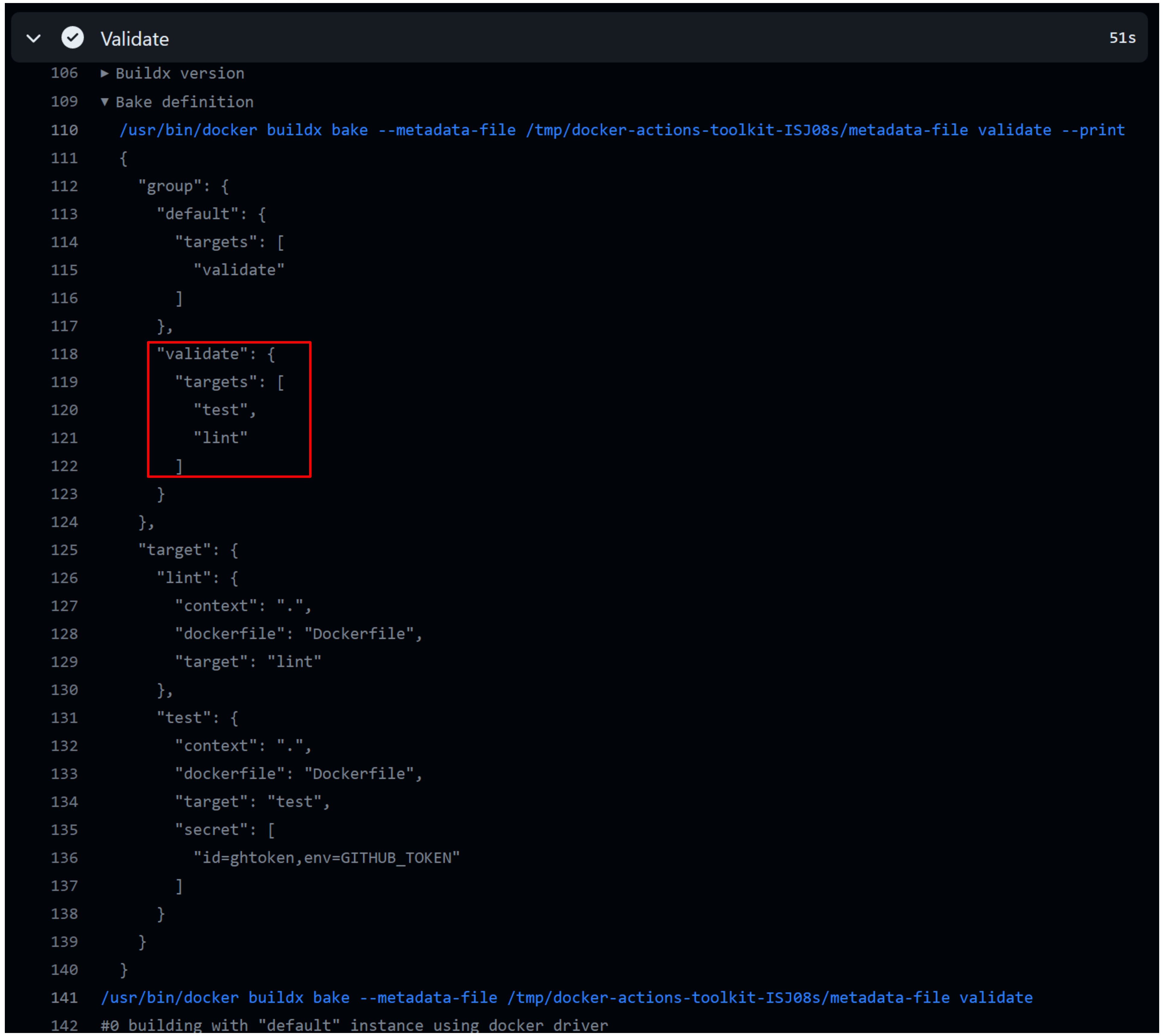
```hcl
# docker-bake.hcl

group "default" {
  targets = ["image"]
}

variable "IMAGE_NAME" {
  default = "foo/bar"
}

target "image" {
  tags = ["${IMAGE_NAME}:latest"]
}

group "validate" {
  targets = ["test", "lint"]
}

target "test" {
  target = "test"
  secret = ["id=ghtoken,env=GITHUB_TOKEN"]
}

target "lint" {
  target = "lint"
}
```

```yaml
-
  name: Lint
  uses: docker/bake-action@v4
  with:
    target: lint
-
  name: Test
  uses: docker/bake-action@v4
  with:
    target: test
  env:
    GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
-
  name: Build and push
  uses: docker/bake-action@v4
  with:
    push: true
```

```yaml
-
  name: Validate
  uses: docker/bake-action@v4
  with:
    targets: validate
  env:
    GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
-
  name: Build and push
  uses: docker/bake-action@v4
  with:
    push: true
```

Simplify our workflow

```
# docker-bake.hcl

group "default" {
  targets = ["image"]
}

variable "IMAGE_NAME" {
  default = "foo/bar"
}

target "image" {
  tags = ["${IMAGE_NAME}:latest"]
}

group "validate" {
  targets = ["test", "lint"]
}

target "test" {
  target = "test"
  secret = ["id=ghtoken,env=GITHUB_TOKEN"]
}

target "lint" {
  target = "lint"
}
```



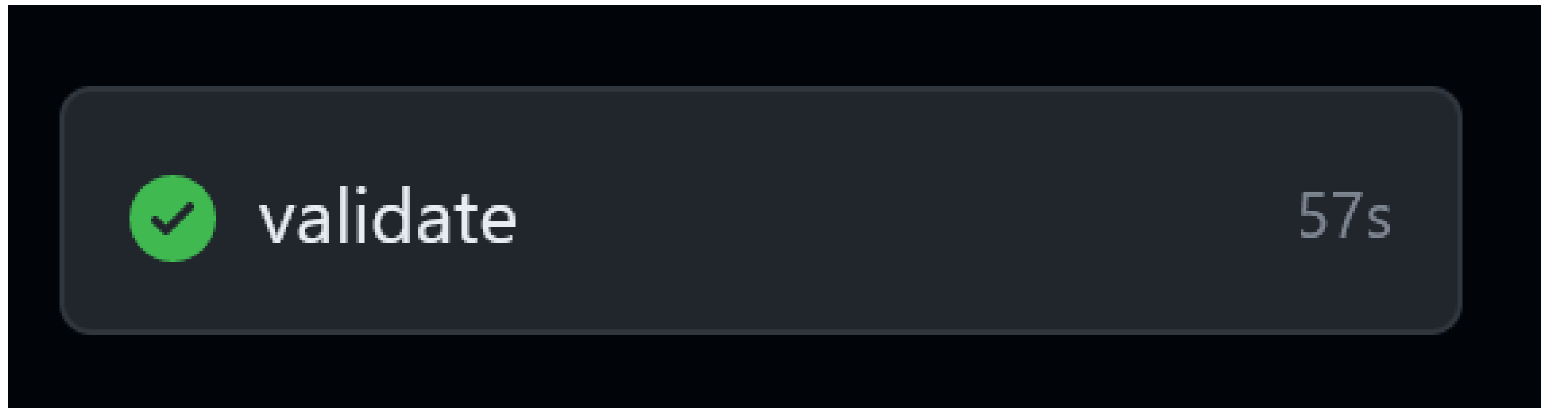
```
Validate                                                                51s
▶ Buildx version
▼ Bake definition
  /usr/bin/docker buildx bake --metadata-file /tmp/docker-actions-toolkit-ISJ08s/metadata-file validate --print
  {
    "group": {
      "default": {
        "targets": [
          "validate"
        ]
      },
      "validate": {
        "targets": [
          "test",
          "lint"
        ]
      }
    },
    "target": {
      "lint": {
        "context": ".",
        "dockerfile": "Dockerfile",
        "target": "lint"
      },
      "test": {
        "context": ".",
        "dockerfile": "Dockerfile",
        "target": "test",
        "secret": [
          "id=ghtoken,env=GITHUB_TOKEN"
        ]
      }
    }
  }
/usr/bin/docker buildx bake --metadata-file /tmp/docker-actions-toolkit-ISJ08s/metadata-file validate
#0 building with "default" instance using docker driver
```

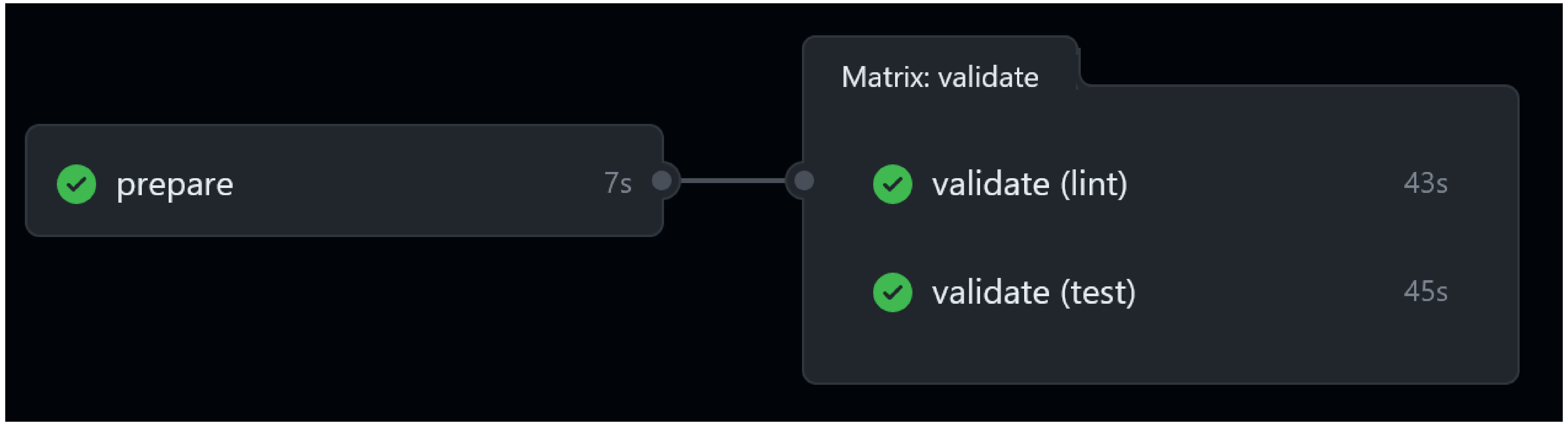
Distributed builds

```yaml
jobs:
  validate:
    runs-on: ubuntu-latest
    steps:
      -
        name: Checkout
        uses: actions/checkout@v4
      -
        name: Validate
        uses: docker/bake-action@v4
        with:
          targets: validate
```

```yaml
jobs:
  prepare:
    runs-on: ubuntu-latest
    outputs:
      targets: ${{ steps.targets.outputs.matrix }}
    steps:
      -
        name: Checkout
        uses: actions/checkout@v4
      -
        name: Matrix
        id: targets
        run: echo "matrix=$(docker buildx bake validate --print | jq -cr '.group.validate.targets')" >> $GITHUB_OUTPUT

  validate:
    runs-on: ubuntu-latest
    needs: prepare
    strategy:
      matrix:
        target: ${{ fromJson(needs.prepare.outputs.targets) }}
    steps:
      -
        name: Checkout
        uses: actions/checkout@v4
      -
        name: Validate
        uses: docker/bake-action@v4
        with:
          targets: ${{ matrix.target }}
```

Distributed builds





Output binary

```
group "default" {
  targets = ["image"]
}

variable "IMAGE_NAME" {
  default = "foo/bar"
}

target "image" {
  tags = ["${IMAGE_NAME}:latest"]
}

group "validate" {
  targets = ["test", "lint"]
}

target "test" {
  target = "test"
  secret = ["id=ghtoken,env=GITHUB_TOKEN"]
}

target "lint" {
  target = "lint"
}

target "binary" {
  target = "binary"
  output = ["./bin"]
}
```

```
# syntax=docker/dockerfile:1

FROM --platform=$BUILDPLATFORM golang:1.20-alpine AS base
WORKDIR /src
ENV CGO_ENABLED=0

FROM base AS test
RUN --mount=type=bind,target=. \
    --mount=type=secret,id=ghtoken \
    GITHUB_TOKEN=(cat /run/secrets/ghtoken) go test ./...

FROM base AS lint
RUN wget -O- -nv \
    https://raw.githubusercontent.com/golangci/golangci-lint/master/install.sh | sh -s v1.51.1
RUN --mount=type=bind,target=. \
    golangci-lint run

FROM base AS build
ARG TARGETOS TARGETARCH
RUN --mount=type=bind,target=. \
    GOOS=$TARGETOS GOARCH=$TARGETARCH go build -ldflags "-s -w" -o /usr/bin/app .

FROM scratch AS binary
COPY --from=build /usr/bin/app /

FROM alpine
COPY --from=build /usr/bin/app /usr/bin/app
CMD ["app"]
```

Updating our workflow

```hcl
group "default" {
  targets = ["image"]
}

variable "IMAGE_NAME" {
  default = "foo/bar"
}

target "image" {
  tags = ["${IMAGE_NAME}:latest"]
}

group "validate" {
  targets = ["test", "lint"]
}

target "test" {
  target = "test"
  secret = ["id=ghtoken,env=GITHUB_TOKEN"]
}

target "lint" {
  target = "lint"
}

target "binary" {
  target = "binary"
  output = ["./bin"]
}
```

```yaml
-
  name: Validate
  uses: docker/bake-action@v4
  with:
    target: validate
-
  name: Build binary
  uses: docker/bake-action@v4
  with:
    targets: binary
-
  name: Build and push
  uses: docker/bake-action@v4
  with:
    push: true
```

Publish binary when a tag is pushed

```yaml
on:
  push:
    tags:
      - "v*"

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      -
        name: Checkout
        uses: actions/checkout@v4
      -
        name: Build binary
        uses: docker/bake-action@v4
        with:
          targets: binary
      -
        name: GitHub Release
        uses: softprops/action-gh-release@v1
        with:
          files: ./bin/*
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Conclusion

- Benefits of **cache exporters**, such as GitHub Actions Cache backend to improve build time and overall pipeline performance.
- Create optimized container images for **different architectures**.
- Effective strategies to manage and **automate tags and labels** based on CI events as well as versioning consistency.
- **Bake** to simplify your workflows and reduce the overhead in your CI pipeline.
- Securely integrate **build-time secrets** in your workflow.
- Vast potential of building using **containers in CI** to achieve **greater portability** between the inner loop and CI environment.

Container-based projects

- https://github.com/docker/buildx - Docker CLI plugin for extended build capabilities with BuildKit
  - Build binaries (cross)
  - Update vendor, docs, authors file, generated files (proto)
  - Tests, lint (golangci-lint)
  - Check for outdated dependencies (mod-outdated)

- https://github.com/tonistiigi/xx - Dockerfile cross-compilation helpers
  - Tests, lint (shfmt, shellcheck)
  - Use bake matrix feature to build linkers

- https://github.com/crazy-max/diun - Notifications when an image is updated on a registry
  - Build binaries (cross) and multi-platform image
  - Update vendor, docs, generated files (proto)
  - Tests, lint (golangci-lint)
  - Check for outdated dependencies (mod-outdated)

dockercon 23

THANK YOU