Publishing Docker images

You can publish Docker images to a registry, such as Docker Hub or GitHub Packages, as part of your continuous integration (CI) workflow.

In this article

Note: GitHub-hosted runners are not currently supported on GitHub Enterprise Server. You can see more information about planned future support on the GitHub public roadmap.

Introduction

This guide shows you how to create a workflow that performs a Docker build, and then publishes Docker images to Docker Hub or GitHub Packages. With a single workflow, you can publish images to a single registry or to multiple registries.

Note: If you want to push to another third-party Docker registry, the example in the "Publishing images to GitHub Packages" section can serve as a good template.

Prerequisites

We recommend that you have a basic understanding of workflow configuration options and how to create a workflow file. For more information, see "Learn GitHub Actions."

You might also find it helpful to have a basic understanding of the following:

- "Using secrets in GitHub Actions"
- "Automatic token authentication"
- "Working with the Docker registry"

About image configuration

This guide assumes that you have a complete definition for a Docker image stored in a GitHub repository. For example, your repository must contain a *Dockerfile*, and any other files needed to perform a Docker build to create an image.

You can use pre-defined annotation keys to add metadata including a description, a license, and a source repository to your container image. For more information, see "Working with the Container registry."

In this guide, we will use the Docker `build-push-action` action to build the Docker image and push it to one or more Docker registries. For more information, see `build-push-action`.

Note: GitHub Actions on your GitHub Enterprise Server instance may have limited access to actions on GitHub.com or GitHub Marketplace. For more information, see "Managing access to actions from GitHub.com" and contact your GitHub Enterprise site administrator.

Publishing images to Docker Hub

Each time you create a new release on GitHub Enterprise Server, you can trigger a workflow to publish your image. The workflow in the example below runs when the `release` event triggers with the `created` activity type. For more information on the `release` event, see "Events that trigger workflows."

In the example workflow below, we use the Docker `login-action` and `build-push-action` actions to build the Docker image and, if the build succeeds, push the built image to Docker Hub.

To push to Docker Hub, you will need to have a Docker Hub account, and have a Docker Hub repository created. For more information, see "Pushing a Docker container image to Docker Hub" in the Docker documentation.

The `login-action` options required for Docker Hub are:

- `username` and `password`: This is your Docker Hub username and password. We recommend storing your Docker Hub username and password as secrets so they aren't exposed in your workflow file. For more information, see "Using secrets in GitHub Actions."

The `metadata-action` option required for Docker Hub is:

- `images`: The namespace and name for the Docker image you are building/pushing to Docker Hub.

The `build-push-action` options required for Docker Hub are:

- `tags`: The tag of your new image in the format `DOCKER-HUB-NAMESPACE/DOCKER-HUB-REPOSITORY:VERSION`. You can set a single tag as shown below, or specify multiple tags in a list.
- `push`: If set to `true`, the image will be pushed to the registry if it is built successfully.

YAML

```yaml
# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
# separate terms of service, privacy policy, and support
# documentation.

# GitHub recommends pinning actions to a commit SHA.
# To get a newer version, you will need to update the SHA.
# You can also reference a tag or branch, but the action may change without
warning.

name: Publish Docker image

on:
  release:
    types: [published]

jobs:
  push_to_registry:
    name: Push Docker image to Docker Hub
    runs-on: [self-hosted]
    steps:
      - name: Check out the repo
        uses: actions/checkout@v4

      - name: Log in to Docker Hub
        uses: docker/login-action@f4ef78c080cd8ba55a85445d5b36e214a81df20a
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_PASSWORD }}

      - name: Extract metadata (tags, labels) for Docker
        id: meta
        uses: docker/metadata-action@9ec57ed1fcdbf14dcef7dfbe97b2010124a938b7
        with:
          images: my-docker-hub-namespace/my-docker-hub-repository

      - name: Build and push Docker image
        uses: docker/build-push-
action@3b5e8027fcad23fda98b2e3ac259d8d67585f671
        with:
          context: .
          file: ./Dockerfile
          push: true
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}
```

The above workflow checks out the GitHub repository, uses the `login-action` to log in to the registry, and then uses the `build-push-action` action to: build a Docker image based on your repository's `Dockerfile`; push the image to Docker Hub, and apply a tag to the image.

Publishing images to GitHub Packages

Note: Container registry is currently in beta for GitHub Enterprise Server and subject to change.

Both GitHub Packages and subdomain isolation must be enabled to use Container registry. For more information, see "Working with the Container registry."

Each time you create a new release on GitHub Enterprise Server, you can trigger a workflow to publish your image. The workflow in the example below runs when the `release` event triggers with the `created` activity type. For more information on the `release` event, see "Events that trigger workflows."

In the example workflow below, we use the Docker `login-action` and `build-push-action` actions to build the Docker image, and if the build succeeds, push the built image to GitHub Packages.

The `login-action` options required for GitHub Packages are:

- `registry`: Must be set to `containers.HOSTNAME`.
- `username`: You can use the `${{ github.actor }}` context to automatically use the username of the user that triggered the workflow run. For more information, see "Contexts."
- `password`: You can use the automatically-generated `GITHUB_TOKEN` secret for the password. For more information, see "Automatic token authentication."

The `build-push-action` options required for GitHub Packages are:

- `push`: If set to `true`, the image will be pushed to the registry if it is built successfully.
- `tags`: Must be set in the format `containers.HOSTNAME/OWNER/REPOSITORY/IMAGE_NAME:VERSION`.

  For example, for an image named `octo-image` stored on GitHub Enterprise Server at `https://HOSTNAME/octo-org/octo-repo`, the `tags` option should be set to `containers.HOSTNAME/octo-org/octo-repo/octo-image:latest`. You can set a single tag as shown below, or specify multiple tags in a list.

Notes:

- This workflow uses actions that are not certified by GitHub. They are provided by a third-party and are governed by separate terms of service, privacy policy, and support documentation.
- GitHub recommends pinning actions to a commit SHA. To get a newer version, you will need to update the SHA. You can also reference a tag or branch, but the action may change without warning.

YAML

Beside Inline

```yaml
name: Create and publish a Docker image
```

```yaml
on:
  push:
    branches: ['release']
```

Configures this workflow to run every time a change is pushed to the branch called `release`.

```yaml
env:
  REGISTRY: containers.HOSTNAME
  IMAGE_NAME: ${{ github.repository }}
```

Defines two custom environment variables for the workflow. These are used for the Container registry domain, and a name for the Docker image that this workflow builds.

```yaml
jobs:
  build-and-push-image:
    runs-on: [self-hosted]
```

There is a single job in this workflow. It's configured to run on the latest available version of Ubuntu.

```yaml
    permissions:
      contents: read
      packages: write
```

Sets the permissions granted to the `GITHUB_TOKEN` for the actions in this job.

```yaml
    steps:
      - name: Checkout repository
```

```
        uses: actions/checkout@v4
```

```
      - name: Log in to the Container registry
        uses: docker/login-action@65b78e6e13532edd9afa3aa52ac7964289d1a9c1
        with:
          registry: ${{ env.REGISTRY }}
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}
```

Uses the `docker/login-action` action to log in to the Container registry registry using the account and password that will publish the packages. Once published, the packages are scoped to the account defined here.

```
      - name: Extract metadata (tags, labels) for Docker
        id: meta
        uses: docker/metadata-action@9ec57ed1fcdbf14dcef7dfbe97b2010124a938b7
        with:
          images: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}
```

This step uses docker/metadata-action to extract tags and labels that will be applied to the specified image. The `id` "meta" allows the output of this step to be referenced in a subsequent step. The `images` value provides the base name for the tags and labels.

```
      - name: Build and push Docker image
        uses: docker/build-push-
action@f2a1d5e99d037542a71f64918e516c093c6f3fc4
        with:
          context: .
          push: true
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}
```

This step uses the `docker/build-push-action` action to build the image, based on your repository's `Dockerfile`. If the build succeeds, it pushes the image to GitHub Packages. It uses the `context` parameter to define the build's context as the set of files located in the specified path. For more information, see "Usage" in the README of the `docker/build-push-action` repository. It uses the `tags` and `labels` parameters to tag and label the image with the output from the "meta" step.

The above workflow is triggered by a push to the "release" branch. It checks out the GitHub repository, and uses the `login-action` to log in to the Container registry. It then extracts labels and tags for the Docker image. Finally, it uses the `build-push-action` action to build the image and publish it on the Container registry.

Publishing images to Docker Hub and GitHub Packages

Note: Container registry is currently in beta for GitHub Enterprise Server and subject to change.

Both GitHub Packages and subdomain isolation must be enabled to use Container registry. For more information, see "Working with the Container registry."

In a single workflow, you can publish your Docker image to multiple registries by using the `login-action` and `build-push-action` actions for each registry.

The following example workflow uses the steps from the previous sections ("Publishing images to Docker Hub" and "Publishing images to GitHub Packages") to create a single workflow that pushes to both registries.

YAML

```yaml
# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
# separate terms of service, privacy policy, and support
# documentation.

# GitHub recommends pinning actions to a commit SHA.
# To get a newer version, you will need to update the SHA.
# You can also reference a tag or branch, but the action may change without
warning.

name: Publish Docker image

on:
  release:
    types: [published]

jobs:
  push_to_registries:
    name: Push Docker image to multiple registries
    runs-on: [self-hosted]
    permissions:
      packages: write
      contents: read
    steps:
      - name: Check out the repo
        uses: actions/checkout@v4

      - name: Log in to Docker Hub
        uses: docker/login-action@f4ef78c080cd8ba55a85445d5b36e214a81df20a
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_PASSWORD }}

      - name: Log in to the Container registry
        uses: docker/login-action@65b78e6e13532edd9afa3aa52ac7964289d1a9c1
        with:
          registry: containers.HOSTNAME
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}

      - name: Extract metadata (tags, labels) for Docker
        id: meta
        uses: docker/metadata-action@9ec57ed1fcdbf14dcef7dfbe97b2010124a938b7
        with:
          images: |
            my-docker-hub-namespace/my-docker-hub-repository
            containers.HOSTNAME/${{ github.repository }}

      - name: Build and push Docker images
        uses: docker/build-push-
action@3b5e8027fcad23fda98b2e3ac259d8d67585f671
        with:
          context: .
          push: true
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}
```

The above workflow checks out the GitHub Enterprise Server repository, uses the `login-action` twice to log in to both registries and generates tags and labels with the `metadata-action` action. Then the `build-push-action` action builds and pushes the Docker image to Docker Hub and the Container registry.