

Deploy an App with Docker

In this topic

Requirements

Requirement for cf ssh Support

Benefits of Specifying Tags

Port Configuration

Start Command

Push a Docker Image From Docker Hub


Push a Docker Image from a Private Registry

Push a Docker Image From a Registry with Authentication

Push a Docker Image From Google Container Registry (GCR)

Docker Volume Support

Page last updated: October 9, 2019

 **Warning:** Pivotal Cloud Foundry (PCF) v2.4 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

Page last updated: October 9, 2019

This topic describes how to use the [Cloud Foundry Command Line Interface \(cf CLI\)](#) to push an app with a new or updated Docker image. Cloud Foundry then uses the Docker image to create containers for the app.

See [Using Docker in Cloud Foundry](#) in the Cloud Foundry documentation for an explanation of how Docker works in Cloud Foundry.

Requirements

To push apps with Docker, you need the following:

- A Cloud Foundry deployment that has Docker support enabled. To enable Docker support, see the [Enable Docker](#) section of *Using Docker in Cloud Foundry* in the Cloud Foundry documentation.
- A Docker image that meets the following requirements:
 - The Docker image must contain an `/etc/passwd` file with an entry for the `root` user. In addition, the home directory and the shell for that `root` user must be present in the image file system.
 - The total size of the Docker image file system layers must not exceed the disk quota for the app. The maximum disk allocation for apps is set by the Cloud Controller. The default maximum disk quota is 2048 MB per app.

 **Note:** If the total size of the Docker image file system layers exceeds the disk quota, the app instances do not start.

- The location of the Docker image on [Docker Hub](#) or another Docker registry.
- A registry that supports the [Docker Registry HTTP API V2](#) and presents a valid certificate for HTTPS traffic.

Requirement for cf ssh Support

If you want to log in to your app container using the `cf ssh` command, you must make a shell such as `sh` or `bash` available in the container.

The SSH server in the container looks for the following executables in absolute locations or the `PATH` environment variable:

- `/bin/bash`
- `/usr/local/bin/bash`
- `/bin/sh`
- `bash`
- `sh`

Benefits of Specifying Tags

If you want your app container to be consistent after platform updates and code changes, specify a tag when you push your Docker image. Otherwise, the platform applies the `latest` tag without respecting changes to `PORT` or `ENTRYPOINT`.

If you push your Docker image without specifying a tag, you must run `cf restage` for the changes to take effect.

Port Configuration

By default, apps listen for connections on the port specified in the `PORT` environment variable for the app. Cloud Foundry allocates this value dynamically.

When configuring a Docker image for Cloud Foundry, you can control the exposed port and the corresponding value of `PORT` by specifying the `EXPOSE` directive in the image Dockerfile. If you specify the `EXPOSE` directive, then the corresponding app pushed to Cloud Foundry listens on that exposed port. For example, if you set `EXPOSE` to `7070`, then the app listens for connections on port 7070.

If you do not specify a port in the `EXPOSE` directive, then the app listens on the value of the `PORT` environment variable as determined by Cloud Foundry.

If you set the `PORT` environment variable via an `ENV` directive in a Dockerfile, Cloud Foundry overrides the value with the system-determined value.

Cloud Foundry supports only one exposed port on the image.

Start Command

By default, Docker uses the start command specified in the Docker image. You can override the start command either by using a command-line parameter or by specifying it in a manifest file.

For more information about command-line parameters for `docker start`, see [docker start](#) in the Docker Documentation.

Push a Docker Image From Docker Hub

To deploy a Docker image from a Docker Hub repository, run `cf push APP-NAME --docker-image REPO/IMAGE:TAG`. Replace the placeholder values in the command as follows:

- `APP-NAME`: The name of the app being pushed
- `REPO`: The name of the repository where the image is stored
- `IMAGE`: The name of an image from Docker Hub
- `TAG`: (Optional, recommended) The tag or version for the image

For example, the following command pushes the `my-image` image from Docker Hub to a Cloud Foundry app:

```
$ cf push my-app --docker-image cloudfoundry/my-image
```

Push a Docker Image from a Private Registry

As an alternative to Docker Hub, you can use any Docker image registry that presents a valid certificate for HTTPS traffic, such as a company-internal Docker registry.

To deploy a Docker image using a specified Docker registry, run `cf push APP-NAME --docker-image MY-PRIVATE-REGISTRY.DOMAIN:PORT/REPO/IMAGE:TAG`. Replace the placeholder values in the command as follows:


- `APP-NAME`: The name of the app being pushed
- `MY-PRIVATE-REGISTRY.DOMAIN`: The path to the Docker registry
- `PORT`: The port where the registry serves traffic
- `REPO`: The name of the repository where the image is stored
- `IMAGE`: The name of the image being pushed
- `TAG`: (Optional, but recommended) The tag or version for the image

For example, the following command pushes the `v2` version of the `my-image` image from the `my-repo` repository of the `internal-registry.example.com` registry on port `5000`:

```
$ cf push my-app --docker-image internal-registry.example.com:5000/my-repo/my-image:v2
```

Push a Docker Image From a Registry with Authentication

Many Docker registries control access to Docker images by authenticating with a username and password. Follow the steps below to deploy a Docker image with registry authentication:

 **Note:** This procedure does not work with the Amazon Web Services Elastic Container Registry (AWS ECR) registry, which authenticates with temporary password tokens rather than username and password.

1. Make sure the `CF_DOCKER_PASSWORD` environment variable is set to the Docker registry user password.

2. Run the following command: `$ CF_DOCKER_PASSWORD=YOUR-PASSWORD cf push APP-NAME --docker-image REPO/IMAGE:TAG --docker-username USER`

Replace the placeholder values in the command as follows:

- `YOUR-PASSWORD`: The password to use for authentication with the registry
- `APP-NAME`: The name of the app being pushed
- `REPO`: The repository where the image is stored:
 - For Docker Hub, this is just the repository name
 - For a private registry, this includes the registry address and port, as described in [Push a Docker Image from a Private Registry](#), in the format: `MY-PRIVATE-REGISTRY.DOMAIN:PORT/REPO`
- `IMAGE`: The name of the image being pushed
- `TAG`: (Optional, but recommended) The tag or version for the image
- `USER`: The username to use for authentication with the registry

Push a Docker Image From Google Container Registry (GCR)

PCF supports pushing apps from images hosted on Google Container Registry (GCR) service. This feature requires that you use [json_key based authentication](#).

Step 1: Authenticate with GCR

To authenticate with GCR using PCF, you must create a JSON key file and associate it with your project:

1. Create a GCP service account. See [Creating and Enabling Service Accounts for Instances](#).

```
$ gcloud iam service-accounts create MY-ACCOUNT --display-name "MY DISPLAY NAME"
```

2. Create a JSON key file and associate it with the service account:

```
$ gcloud iam service-accounts keys create key.json --iam-account=MY-ACCOUNT@MY-PROJECT-ID.iam.gserviceaccount.com
```

3. Set your project ID:

```
$ gcloud config set project MY-PROJECT-ID
```

4. Add the IAM policy binding for your project and service account:

```
gcloud projects add-iam-policy-binding MY-PROJECT --member serviceAccount:MY-ACCOUNT@MY-PROJECT-ID.iam.gserviceaccount.com --role roles/storage.objectViewer
```


Step 2: Deploy the GCP Image

Run the following command to deploy your GCR image using the cf CLI:

```
CF_DOCKER_PASSWORD="$(cat key.json)" cf push APP-NAME --docker-image docker://MY-REGISTRY-URL/MY-PROJECT/MY-IMAGE-NAME --docker-username _json_key
```

Replace the placeholder values in the command as follows:

- `APP-NAME` : The name of the app being pushed
- `MY-REGISTRY-URL` : The URL of your registry
- `MY-PROJECT` : The name of your project
- `MY-IMAGE-NAME` : The name of your image

 **Note:** The `key.json` file must point to the file you created in the previous step.

 **Note:** For information about specifying `MY-REGISTRY-URL`, see [Pushing and Pulling Images](#)  on the Google Cloud documentation.

Docker Volume Support

You can use volume services with Docker apps. For more information about enabling volume support, see the [Using an External File System \(Volume Services\)](#) topic.