Deploy with Docker

This guide shows you how to build an evaluation-only Docker image by using the Dockerfile provided inside `PingGateway-2024.9.0.zip`.

WARNING

ForgeRock provides no commercial support for production deployments that use ForgeRock's evaluation-only Docker images. When deploying the ForgeRock Identity Platform using Docker images, you must build and use your own images for production deployments.

This guide is for Ping Identity Platform developers who want an easy-to-use example of containerized deployment, and for PingGateway developers who want to configure a production environment for containerized deployment.

For information about deploying Ping Identity Platform by using DevOps techniques, refer to ForgeOps' Start here.

This guide assumes that you are familiar with the following topics:

- PingGateway, to edit the basic configuration and test the changes.
- Docker, to build and run and Docker images.

The examples in this guide use some of the following third-party tools:

- **`curl`** : https://curl.haxx.se
- **`HTTPie`** : https://httpie.org
- **`jq`** : https://stedolan.github.io/jq/
- **`keytool`** : https://docs.oracle.com/en/java/javase/17/docs/specs/man/keytool.html

Build and run a Docker image

A Dockerfile is delivered inside `PingGateway-2024.9.0.zip` to help you build an evaluation-only, base Docker image for PingGateway. After building and running the Docker image, add a configuration as described in Add configuration to a Docker image.

WARNING

WARNING

ForgeRock provides no commercial support for production deployments that use ForgeRock's evaluation-only Docker images. When deploying the ForgeRock Identity Platform using Docker images, you must build and use your own images for production deployments.

The Docker image has the following characteristics:

- The Docker image runs on Linux and Mac operating systems.
- PingGateway binaries are delivered in `/opt/ig`.
- The environment variable `$IG_INSTANCE_DIR` has the value `/var/ig`.
- A ForgeRock user with username: `forgerock` and uid: `11111`, runs the PingGateway process and owns the configuration files.

Build the base image for PingGateway

1. Download `PingGateway-2024.9.0.zip` from the BackStage download site, and unzip. The directory `/path/to/identity-gateway-2024.9.0` is created.
2. Go to `/path/to/identity-gateway-2024.9.0`.
3. With a Docker daemon running, build a base Docker image:

```
$ docker build . -f docker/Dockerfile -t ig-image

Sending build context to Docker daemon
Step 1/7 : FROM gcr.io/forgerock-io/...:latest
latest: Pulling from forgerock-io/...
...
Successfully tagged ig-image:latest
```

4. Make sure the Docker image is available:

```
$ docker image list

REPOSITORY                     TAG        IMAGE ID
ig-image                       latest
gcr.io/forgerock-io/...  latest
```

Run the Docker image

The following steps run the Docker image on port `8080`. Make sure the port is not being used, or use a different port as described in the procedure.

1. With a Docker daemon running, run the Docker image:

```
$ docker run -p 8080:8080 ig-image
```

PingGateway starts up, and the console displays the message log.

2. Go to http://localhost:8080 to view the PingGateway welcome page.

Stop the Docker image

1. List the Docker containers that are running:

```
$ docker container ls
```

2. For a container with the status `Up`, use the container ID to stop the container:

```
$ docker container stop CONTAINER_ID
```

Run options

Consider using the following options when you run the Docker image:

`-e IG_OPTS=-Dig.pid.file.mode=value ig-image`

Allow startup if there is an existing PID file. PingGateway removes the existing PID file and creates a new one during startup. The following example passes an environment variable with the value `override` as a Java runtime option:

```
$ docker run -e "IG_OPTS=-Dig.pid.file.mode=override" ig-image
```

To prevent restart if there is an existing PID file, set the value to the default `fail`.

`-p port:port`

The default ports `8080:8080` equate to `local-machine-port:internal-container-port`. PingGateway can run on a different port, but the container must always run on `8080`. The following example runs PingGateway on port `8090`:

```
$ docker run -p 8090:8080 ig-image
```

`-v configuration directory`

The default configuration directory is `/var/ig/`. The following example sets the configuration directory to `$HOME/.openig`:

```
$ docker run -v $HOME/.openig:/var/ig/ ig-image
```

-user ***user***

Run the image as the provided user. The following example uses the ID `11111`:

```
$ docker run --user 11111 ig-image
```

it

Run the image in interactive mode:

```
$ docker run -it ig-image
```

sh

Run the image in sh shell:

```
$ docker run ig-image sh
```

Add configuration to a Docker image

The following sections describe how to add configuration to your Docker image. Before working through this section, complete the procedures in Build and run a Docker image.

Run an image with a mutable configuration

This section describes how to add a basic route to your local PingGateway configuration folder, and mount the configuration to the Docker container.

If you change your configuration in a way that doesn't require PingGateway to restart, you see the change in your running Docker image without restarting it or rebuilding it. For information changes that require restart, refer to When to restart PingGateway after changing the configuration.

Use this procedure to manage configuration externally to the Docker image. For example, use it when developing routes.

1. Add the following route to your local PingGateway configuration as `$HOME/.openig/config/routes/hello.json`:

```
{
  "name": "hello",
  "handler": {
    "type": "StaticResponseHandler",
    "config": {
      "status": 200,
      "headers": {
        "Content-Type": [ "text/plain; charset=UTF-8" ]
      },
      "entity": "Hello world!"
    }
  },
  "condition": "${find(request.uri.path, '^/hello')}"
}
```

The configuration contains a static response handler to return a "Hello world!" statement when the URI of a request finishes with `/hello`.

2. Run the Docker image, using the option to mount the local PingGateway configuration directory:

```
$ docker run -p 8080:8080 -v $HOME/.openig:/var/ig/ ig-image
```

3. Go to http://localhost:8080/hello to access the route in the mounted configuration.

   The "Hello world!" statement is displayed.

4. Edit `hello.json` to change the "Hello world!" statement, and save the file.

   Go again to http://localhost:8080/hello to see that the message changed without changing your Docker image.

Run an image with an immutable configuration

This section describes how to add a basic route to your local PingGateway configuration folder, copy it into a new Docker image, and run that Docker image.

Unlike the previous example, the Docker image is immutable. If you change your configuration locally, the Docker image is not changed.

Use this procedure to manage configuration within the Docker image. For example, use it when you want to deploy the same configuration multiple times.

1. Add the following route to your local PingGateway configuration as `$HOME/.openig/config/routes/hello.json`:

```
{
  "name": "hello",
  "handler": {
    "type": "StaticResponseHandler",
    "config": {
      "status": 200,
      "headers": {
        "Content-Type": [ "text/plain; charset=UTF-8" ]
      },
      "entity": "Hello world!"
    }
  },
  "condition": "${find(request.uri.path, '^/hello')}"
}
```

The configuration contains a static response handler to return a "Hello world!" statement when the URI of a request finishes with `/hello`.

2. Add the following file to your local PingGateway configuration as `$HOME/.openig/Dockerfile`, where `$HOME/.openig` is the instance directory:

```
FROM ig-image
COPY config/routes/hello.json
"$IG_INSTANCE_DIR"/config/routes/hello.json
```

The Dockerfile copies `hello.json` into the Docker image. The `$IG_INSTANCE_DIR` environment variable is defined in the PingGateway base image.

3. Build the Docker image:

```
$ docker build . -t ig-custom

Sending build context to Docker daemon
 Step 1/2 : FROM ig-image
 Step 2/2 : COPY config/routes/hello.json
"$IG_INSTANCE_DIR"/config/routes/hello.json
 Successfully tagged ig-custom:latest
```

4. Make sure the Docker image is available:

```
$ docker image list

REPOSITORY                TAG                 IMAGE ID
```

```
ig-custom                  image_tag         51b...3b7
gcr.io/forgerock-io/ig     image_tag         404...a2b
```

5. Run the Docker image on port `8080`:

```
$ docker run -p 8080:8080 ig-custom
```

6. Go to http://localhost:8080/hello. The "Hello world!" statement is displayed.

Was this helpful?