



Deployment Guide

/ ForgeRock Identity Gateway 7

Latest update: 7.0.2

ForgeRock AS.
201 Mission St., Suite 2900
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Copyright © 2017-2021 ForgeRock AS.

Abstract

Tutorials for deploying ForgeRock® Identity Gateway with Docker, with best practices for containerized deployment in production environments.



This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-nd/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

ForgeRock® and ForgeRock Identity Platform™ are trademarks of ForgeRock Inc. or its subsidiaries in the U.S. and in other countries. Trademarks are the property of their respective owners.

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

DejaVu Fonts

Bitstream Vera Fonts Copyright

Copyright (c) 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Bitstream" or the word "Vera".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Bitstream Vera" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the names of Gnome, the Gnome Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the Gnome Foundation or Bitstream Inc., respectively. For further information, contact: fonts at gnome dot org.

Arev Fonts Copyright

Copyright (c) 2006 by Tavmjong Bah. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the modifications to the Bitstream Vera Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Tavmjong Bah" or the word "Arev".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Tavmjong Bah Arev" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL TAVMJONG BAH BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the name of Tavmjong Bah shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from Tavmjong Bah. For further information, contact: tavmjong @ free . fr.

FontAwesome Copyright

Copyright (c) 2017 by Dave Gandy, <https://fontawesome.com/>.

This Font Software is licensed under the SIL Open Font License, Version 1.1. See <https://opensource.org/licenses/OFL-1.1>.

Table of Contents

Preface	iv
About This Guide	iv
1. Build and Run a Docker Image	1
Build the Base Image for IG	1
Run the Docker Image	2
Stop the Docker Image	2
2. Add Configuration to a Docker Image	3
Run an Image With a Mutable Configuration	3
Run an Image With an Immutable Configuration	4

Preface

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see <https://www.forgerock.com>.

About This Guide

This document describes how to build a Docker image by using the Dockerfile provided inside [IG-7.0.2.zip](#).

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

For information about deploying ForgeRock Identity Platform by using DevOps techniques, see ForgeOps' *Start Here*.

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

- IG, 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/11/tools/keytool.html>

Chapter 1

Build and Run a Docker Image

ForgeRock delivers a Dockerfile inside `IG-7.0.2.zip`, to help you build a base Docker image for IG. After building and running the base image, add a configuration as described in *"Add Configuration to a Docker Image"*.

The Dockerfile builds a base Docker image with the following characteristics:

- The Docker image runs on Linux and Mac operating systems.
- IG 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 IG process and owns the configuration files.

The following sections describe how use the Dockerfile to build a base image for IG:

- "Build the Base Image for IG"
- "Run the Docker Image"
- "Stop the Docker Image"

Build the Base Image for IG

1. Download `IG-7.0.2.zip` from the ForgeRock BackStage download site, and unzip.

The directory `/path/to/identity-gateway` is created.

2. With a Docker daemon running, build a base Docker image, using `/path/to/identity-gateway`:

```
/path/to/identity-gateway$ docker build . -f docker/Dockerfile -t ig-image

Sending build context to Docker daemon
Step 1/7 : FROM gcr.io/forgerock-io/java-11:latest
latest: Pulling from forgerock-io/java-11
...
Successfully tagged ig-image:latest
```

3. Make sure that the Docker image is available:

```
$ docker image list
REPOSITORY          TAG          IMAGE ID
ig-image            latest      gcr.io/forgerock-io/java-11
gcr.io/forgerock-io/java-11  latest
```

Run the Docker Image

The following steps run the Docker image on port **8080**. Make sure that 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
```

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

2. Go to <http://localhost:8080> to view the IG welcome page.

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

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

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

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

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

- Run the image in sh shell, in interactive mode, using the provided Forgerock user:

```
$ docker run -p 8080:8080 -it --user 11111 ig-image sh
```

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
```

Chapter 2

Add Configuration to a Docker Image

The following sections describe how to add configuration to your Docker image:

- "Run an Image With a Mutable Configuration"
- "Run an Image With an Immutable Configuration"

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 IG configuration folder, and mount the configuration to the Docker container.

If you change your configuration in a way that doesn't require IG to restart, you see the change in your running Docker image without restarting it or rebuilding it. For information about configuration changes that require restart, see "Changing the Configuration and Restarting IG" in the *Gateway Guide*.

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

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

```
{
  "name": "hello",
  "handler": {
    "type": "StaticResponseHandler",
    "config": {
      "status": 200,
      "reason": "OK",
      "headers": {
        "Content-Type": [ "text/plain" ]
      },
      "entity": "Hello world!"
    },
  },
  "condition": "${matches(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 IG 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 IG 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 IG configuration as `$HOME/.openig/config/routes/hello.json`:

```
{
  "name": "hello",
  "handler": {
    "type": "StaticResponseHandler",
    "config": {
      "status": 200,
      "reason": "OK",
      "headers": {
        "Content-Type": [ "text/plain" ]
      },
      "entity": "Hello world!"
    }
  },
  "condition": "${matches(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 IG 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 IG 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 that the Docker image is available:

```
$ docker image list
REPOSITORY          TAG          IMAGE ID
ig-custom           image_tag   51b...3b7
gcr.io/forgerock-io/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.