ENFOCUS

PITSTOP LIBRARY CONTAINER

Installation Guide

05 - 2023

Contents

1. What is the PitStop Library Container?

Put 25 years of PDF preflight and correction knowhow in your cloud. Validate PDF files, using PitStop Library, as they are submitted.

OEM vendors, print service providers, ad agencies, marketing content creators, and any other manufacturer that wants to put PDF preflight in their web-to-print, management information system, online proofing solutions, or any other customer portal solution, being it for B2B or B2C purposes, can access the power of PitStop via REST API. Provided as a Docker image, PitStop Library Container makes integration easier and future-forward by providing a cloud-ready preflight solution that works on any platform of your choice, being it private cloud or using any available cloud service. It's also possible to add a preflight service to your internal processes and by deploying the service on premise.

Control over job queuing, scalability, and distribution is in the hands of the developer.

Preflight Profiles and Action Lists created with PitStop Pro and PitStop Server can be used with the PitStop Library Container, keeping uniformity across print production.

PitStop Library Container is a perfect tool for:

- OEMs with the need to put PDF preflight in their cloud solutions
- W2P and MIS vendors wanting to incorporate preflight
- Print businesses with home-brewed or off the shelf W2P systems
- Publications that receive print ads from multiple sources

Benefits:

- Preflight PDFs during the job onboarding process
- Capture errors before PDFs hits production, saving time and money
- Fix errors to avoid delays and provide an even higher level of service
- Provide instant feedback on preflight results
- Preflight reports can be generated in PDF, XML, and JSON
- Unrestricted scaling
- Spread load by running multiple preflight processing tasks in parallel on a single server or load-balance between multiple servers
- Run on any platform and OS of your choice

The PitStop Library Container supports the following languages: English, German, Spanish, French, Italian, Dutch, Japanese, Portuguese, Chinese, and Polish.

For more information about the Docker technology, refer to *https://www.docker.com/resources/what-container/*

2. Getting the Docker image

2.1. Docker Desktop

To install the Docker container on your host machine, you need Docker Desktop, which can be downloaded from *https://www.docker.com/get-started/*

Docker Desktop is available for Mac, Windows, and Linux.

If you want to run PitStop Library Container in a cloud environment, see *further*.

2.2. AWS CLI

The PitStop Library Container is available through an AWS ECR repository. More information on AWS ECR is available at *https://aws.amazon.com/ecr/*

To get the image, you need to install the AWS CLI environment, which is available at *https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html*

If the AWS CLI environment is installed, open the console and make sure both Docker Desktop and AWS CLI are installed correctly by checking the version:

```
>> docker --version
```

```
>> aws --version
```

If everything is installed correctly, it shows the corresponding version.

2.3. Adding credentials

Then add the credentials to AWS CLI by entering the following command:

```
>> aws configure
```

It will ask for the **AWS Access Key ID** and the **AWS Secret Access Key**, which are provided by Enfocus.

It will also ask for the **Default region name**, where you can enter *eu-west-1* and the **Default output format**, which can be left empty (just press enter).

Then log in to AWS by entering this command:

```
>> aws ecr get-login-password --region eu-west-1 | docker login --username AWS --
password-stdin 118129972018.dkr.ecr.eu-west-1.amazonaws.com
```

If the credentials were entered correctly, then this command will show: *Login Succeeded*.

2.4. Downloading the image

To download the latest version of the image from the AWS ECR repository, use the following command:

```
>> docker pull
118129972018.dkr.ecr.eu-west-1.amazonaws.com/enfocus-plc-preflight-worker:latest
```

Now it starts downloading, extracting, and installing the Docker image. You can also see in the Docker Desktop application that the image is now available.

2.5. Downloading a specific version

The current latest version is 2319.

If, for some reason, you want to use a previous version of the PLC, this can be done with the following command:

```
>> docker pull
118129972018.dkr.ecr.eu-west-1.amazonaws.com/enfocus-plc-preflightworker:2303
```

So you can just replace "latest" with the specific version you want to download. The versions that are supported are: 22.0, 2303 and 2319.

Note: As of 2023, we will format the version number as follows: YYWW where YY is the number of the year and WW is the week number. So 2319 means the version that is released in week 19 of 2023.

3. Configuring the PitStop Library Container

After downloading the Docker image, and before you can run it, you need to configure it correctly. For the configuration, we make use of the docker-compose.yml file. This file contains some parameters and environment variables that need to be adjusted according to your specific situation. The *docker-compose.yml file* is provided by Enfocus and can be edited.

More generic information on how this file is built, can be found at *https://docs.docker.com/compose/compose-file/*

An example docker-compose.yml file is shipped along with the documentation (see *the appendix of this document*).

The docker-compose.yml file looks like this:

```
version: '3.2'
services:
  worker-service:
    image: <image name>
    container_name: node-worker
    ports:
      - 3002:3002
    volumes:
      - ~/.aws/credentials:/root/.aws/credentials:ro
 - ~/local_dir:/root/dir_within_docker:ro
 - ~/output_dir_local:/root/dir_output_within_docker:
    restart: always
    environment:
        - <some environment variables>
```

The different properties are:

- The version property contains the Docker compose version.
- The image property should contain the full image name as you see it in your Docker Desktop.
- The container_name property can be whatever value. It just gives a name to the container.
- The ports property defines on what port the Docker image is listening for incoming API calls.
- The restart property defines what needs to happen when the Docker container closes unexpectedly. The always value means that it will automatically restart when it was closed unexpectedly. You can leave out this property, but it is recommended to keep it as such.
- The volumes property contains mounted drives that will be accessible from within the Docker container. The example above is an example of the AWS credentials files that can be used to verify access to a AWS SQS queue.
- The environment property contains all environment variables explained below. Note that the signature is as follows: ENVIRONMENT_VARIABLE=VALUE

Environment variable	Default value	Description
COM_ENFOCUS_CLOUD_PREFLIGHT_JOB_MANAGERS_MAX_COUNT		
	3	The number of slave workers that will be started with the Docker container. This should not be more than the

Environment variable	Default value	Description
		number of cores. The master worker is not counted here, so with 3, we start 4 workers (1 master and 3 slaves).
COM_ENFOCUS_CLOUD_PREFLIGHT_PROCESSOR_URL_PORTS		
	<empty>	Comma separated field that contains port numbers that will be assigned to the workers. Typically, this can be left empty. Only in exceptional cases where there are specific port requirements, this should be added.
COM_ENFOCUS_CLOUD_PREFLIGHT_SQS_NAME		
	<empty>	The name of the AWS SQS queue to pull jobs from (see Queuing for PLC).
COM_ENFOCUS_CLOUD_AWS_REGION		
	eu-west-1	This is the region where the SQS queue is configured.
COM_ENFOCUS_CLOUD_PLC_REGION		
	eu	This is the region of the Enfocus License server. Can be either eu or us, depending on your location.
COM_ENFOCUS_CLOUD_PREFLIGHT_AUTHENTICATION_TOKEN		
	<empty>	This is the authentication token that is used to authenticate. This token is provided by Enfocus and is uniquely linked to your account.
COM_ENFOCUS_CLOUD_EQUIPMENT_TOKEN		
	<empty>	This is the so-called equipment token that is used for accessing your license on our licensing server. This token is provided by Enfocus and is uniquely linked to your account. This token is a secret, which should never be shared externally.
COM_ENFOCUS_CLOUD_ACCOUNT_NUMBER		
	<empty>	This is your account number. Every customer has a unique account number, which is provided by Enfocus.
COM_ENFOCUS_CLOUD_SUBSCRIPTION_TYPE_ID		
	ENF-PLC-PLCTR	This is the subscription type identifier that is provided by Enfocus. This contains a fixed value for the PitStop Library Container.

Important: To run the PitStop Library Container, there should be a connection to the Enfocus license server. At start-up, the PitStop Library Container checks connectivity and will only continue if a connection can be made.

4. Running the PitStop Library Container

After the image has been downloaded to your local environment, and the necessary configuration options have been set up and saved in your docker-compose.yml file, you can run the Docker software by changing the working directory to the one where your docker-compose.yml file is located and run the following command:

```
>> docker-compose up
```

5. To queue or not to queue

There are 2 ways to start a job, either by direct access to the API or by queuing jobs and let them be pulled by the Docker image. Both have their advantages, and you should choose the best approach for your application.

Direct access is used when jobs need immediate processing. This is the fastest way, but if all workers are busy processing other jobs, new jobs will fail to start, and you will get an error message in the response when doing an API call. If you need more workers, you will have to scale the Docker image to be able to process enough in parallel. Disadvantage is that you need to set up scaling and this comes with a hardware cost. Second disadvantage is that you will have to deal with unprocessed jobs and build a mechanism to start them again.

Queuing is typically used when you either have high volume or unpredictable peak moments, and don't necessarily need to have immediate response. With this approach, the hardware is used in the most efficient way, and it can cope perfectly with peak moments. All jobs are just added to the queue, and they are pulled from the queue and processed one after the other. If you see that your queue keeps growing, then it's a good practice to start scaling. Even with this approach, you can track progress, but as long as the job is in the queue, you don't see any progress.

6. API doc

Once the Docker image runs on your local system, you can access the API Swagger documentation in your browser on the following location:

http://localhost:3002/api-docs

Note that 3002 is the default port. If you have chosen a different port, then change the URL accordingly.

6.1. Start a job

The parameters are passed in the body of the POST call.

Request URL (POST):

```
/job
```

Example parameters in the body:

```
{
  "jobStatusURL": "http://localhost:8080/result",
  "reportProgress":true,
  "abortingTypeValue": "5",
  "progressTimeoutValue: "100",
  "inputFileURL": "url",
  "profileURL": "url",
  "actionListURLs": "string",
  "variableSetURL": "url",
  "jobTicketURL": "url",
  "extraFontsFolderURL": "url",
  "reportTemplate": {
      "configFileURL": "string",
      "templateFileURL": "string"
     },
  "outputFixedFileURL": "url",
  "reportLanguage": "en-US",
  "maxItemsPerCategory": "100",
  "maxNumOccurrencesPerItem": "100",
  "reportURLs": {
       "JSON": "string",
       "XML": "string",
       "PDF": "string"
     }
}
```

Input files (inputFileURL, profileURL, actionListURLs, variableSetURL, jobticketURL, extraFontsFolderURL, reportTemplate) should be available through URLs that are accessible for the Docker container. Be careful with spaces. Depending on the webserver, you should replace spaces with the URL encoded value of %20. It's also possible to work with local files if you mount a folder in the volumes section of the docker-compose.yml file.

Output files (reportURLs, outputFixedFileURL) will be stored on the given location with a HTTP PUT command. Typically, pre-signed URLs are used for that, but any URL supporting PUT operations will work fine. You can also write files to a local folder (see *Using local files instead of URLs* on page 11).

The **jobStatusURL** will be used to post the result of the job. This is a webhook that can be used as trigger for further processing.

actionListURLs is a JSON array of URLs (using [] and a comma (,) as separator).

More information about the parameters can be found in the Swagger documentation when the image is running locally: *http://localhost:3002/api-docs*

When adding the job, you will get a response as follows:

The id can be used in the */job/progress* API call and the */job/abort* API call.

6.1.1. Using local files instead of URLs

To prevent Preflight Profiles or Action Lists to be downloaded every time, it is possible to work with local files. First you need to mount them by adding the following in the volumes section of your docker-compose.yml file:

```
[code]
    volumes:
    - ~/.aws/credentials:/root/.aws/credentials:ro
    - ~/myuser/plc/my_input_folder:/root/input/
    - ~/myuser/plc/my_output_folder:/root/output/
[/code]
```

where *~/myuser/plc/my_input_folder* and *~/myuser/plc/my_output_folder* are the folders in your host system, and */root/input* and */root/output* are the folders inside the Docker container.

Once you have added this and restarted the PLC, you can start using files through this mechanism.

Your parameters could look like this:

```
profileURL:/root/input/my_preflight_profile.ppp
actionListURLs:["/root/input/my_action_list.eal"]
outputFixedFileURL: "/root/output/my_pdf.pdf"
```

Note that you also can combine this with URLs. Typically your input PDF file can be served through URL, while your Preflight Profile (.ppp) and Action Lists (.eal) can be served through local file.

It's also possible to use the <JOBID> parameter in your local file path, for example:

```
outputFixedFileURL: "/root/output/<JOBID>/my_pdf.pdf"
```

This will create a folder in */root/output/* with the job ID and the *my_pdf.pdf* file will be created inside that folder.

Note that output files are being overwritten, so don't use the same filenames for your *outputFixedFileURL* or your *reportURLs*.

6.2. Progress of a job

With this API call, you can track the progress of a job.

Request URL (GET):

```
/job/progress/<job_id>
```

The job_id is the ID you get as response from the job API call.

There are different statuses:

1. If a job is waiting to get processed, then the response is:


```
{
    "id": "a9d0a863-3223-4cfe-b78d-a07f519a0fff",
    "status": "Initializing"
}
```


2. If a job is in progress, then the response is:


```
{
    "currentPage": 1,
    "fraction": 0.6953266562500001,
    "totalPages": 1,
    "type": "processingPages",
    "id": "f6999547-86f3-41db-8ce8-b3b550ddf75a",
    "progress": 70
}
```


   Different values for the type are:

   - idle if the job is pending processing
   - processingPages if the job is processing
   - savingJsonReport if the job is creating the JSON report (if applicable)
   - savingXmlReport if the job is creating the XML report (if applicable)
   - savingPdfReport if the job is creating the PDF report (if applicable)

3. If a job is completed, then the response is:


```
{
    "id": "a9d0a863-3223-4cfe-b78d-a07f519a0fff",
    "metrics": {
        "pagesNumber": 1,
        "sizeInBytes": 2371419
    },
    "status": "Completed"
}
```


Note: The completed status is temporary. When a new job has been started, the previous job is deleted, and you will get the same response as if a job is not found.

4. If a job is not found, then you get an error response:

```
{
    "error": "No job found with id cce2ef34-e77a-40c0-af06-823a43022a53"
}
```

6.3. Abort a job

If a job takes too long or seems to be stuck, you can always abort the job with the abort API call to free up resources. This call needs the job ID that you got from the /job API call.

Request URL (GET):

```
/job/abort/<job_id>
```

If the job is found, you get the following response:

```
{
    "message": "Currently processing job will be aborted."
}
```

If a job is not found, then you get an error response:

```
{
    "error": "No job found with id cce2ef34-e77a-40c0-af06-823a43022a53"
}
```

6.4. Health check

You can check if the Docker instance is still actively running by doing a health check. There are 2 methods that you can use to check if the instance is OK.

The first method is a version check. This method always gives you the same response as shown below.

Request URL (GET):

```
/version.json
```

Example response:

```
{
    "name": "enfocus-preflight-worker",
    "org": "com.enfocus.cloud",
    "rev": "local"
    "version": "23.03"
}
```

The second method is a check to see how many workers there are, and what their status is, as shown below.

Request URL (GET):

```
/isAlive
```

Example response:


```
{
    "alive": true,
    "processors": {
        "36491": {
            "isStarted": true,
            "processing": false,
            "type": "Push mechanism"
        },
        "36492": {
            "isStarted": true,
            "processing": false,
            "type": "Push mechanism"
        },
        "36493": {
            "isStarted": true,
            "processing": false,
            "type": "Push mechanism"
        },
        "36494": {
            "isStarted": true,
            "processing": false,
            "type": "Push mechanism"
        }
    }
}
```

7. Debugging

When running the PitStop Library Container via command line, the logs are shown in the command line interface. However, you can also check the logs with the following command:

```
>> docker logs --details <container-name>
```

The container name is defined in the docker-compose.yml file.

8. Deployment, queuing and scaling

Control over job queuing, scalability and distribution of Docker instances on one or multiple servers is in the hands of the developer. Various readily available tools can be utilized to help you with your implementation.

8.1. Running on AWS

See: *https://docker-curriculum.com/#docker-on-aws*

8.2. Running on Azure

See: *https://learn.microsoft.com/en-us/training/modules/run-docker-with-azure-container-instances/*

8.3. Running on Google Cloud

See: *https://cloud.google.com/run/docs/deploying*

8.4. AWS SQS

SQS is a queueing mechanism from AWS that allows jobs to be placed in a queue at any volume to prevent jobs from being lost. PitStop Library Container has native support for SQS and can be configured easily.

First, you need to create a credentials file. On Windows, this can be done by creating a file named: c:\users\<username>\.aws\credentials

Edit this file with Notepad and put the following in the file:

```
[default]
aws_access_key_id=<your_aws_access_key_id>
aws_secret_access_key=<your_aws_secret_access_key>
region=<your_region>
```

The region can be *eu-west-1* for instance.

Second, you need to configure the PitStop Library Container so that it reads jobs from the AWS SQS queue. You can do this by editing your docker-compose.yml file and adding the following environment variables:

```
- COM_ENFOCUS_CLOUD_PREFLIGHT_SQS_NAME=<name_of_your_queue>
```

```
- COM_ENFOCUS_CLOUD_AWS_REGION=<region_of_your_queue>
```

You need to add these variables to the environment section.

Also, you need to link your AWS credentials file in the docker-compose.yml file by adding this line to the volumes section:

```
- ~/.aws/credentials:/root/.aws/credentials:ro
```

This makes sure that the PitStop Library Container can access the AWS SQS queue.

Third, you need to send jobs to the SQS queue. For this, you need to make use of the AWS SDK. Documentation can be found here: *https://docs.aws.amazon.com/index.html* (search for AWS SDK, and you will find the documentation for multiple programming languages).

Below is an example for PHP, but you can use other languages to do this as well.

```
// This loads the AWS SDK. Make sure to download it and put it in this dir.
require 'aws/aws-autoloader.php';

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;

$sharedConfig = array(
    'profile' => 'default',
    'region' => 'eu-west-1',  // don't forget to change this to your region!
    'version' => 'latest',
     'aws_access_key_id' => '...',    // fill in your credentials!
     'aws_secret_access_key' => '...'  // fill in your credentials!
);

$client = new SqsClient($sharedConfig);

try {
     $params = array(
          'DelaySeconds' => 10,
          'MessageBody' => '
                {
                "reference": "...",// fill in the correct variables
                "inputFileURL": "...",
                "profileURL": "...",
                "outputFixedFileURL": "...",
                "reportURLs": {
                           "JSON":"..."
                     }
                }
                ',
          'QueueUrl' => '<your queue url>'// the url of your sqs queue
     );
     $result = $client->sendMessage($params);
     var_dump($result);

} catch (AwsException $e) {
    // output error message if fails
     echo $e->getMessage();
}
```

More information on how to set up an SQS queue and send jobs can be found at *https://aws.amazon.com/sqs/*

8.5. Docker Swarm

With Docker Swarm, you can set up scaling.

More information can be found at *https://docs.docker.com/engine/swarm/*

Note: You need the correct license to allow for scaling.

9. Resource constraints

By default, a container has no resource constraints and can use as much of a given resource as the host's kernel scheduler allows.

However, you can configure the resources as you wish. More information can be found at *https://docs.docker.com/config/containers/resource_constraints/*

10. System requirements

The PitStop Library Container is designed to run on any OS that supports Docker images, see *https://docs.docker.com/engine/faq/#does-docker-run-on-linux-macos-and-windows*.

"*The Docker platform runs natively on Linux (on x86-64, ARM and many other CPU architectures) and on Windows (x86-64). Docker Inc. builds products that let you build and run containers on Linux, Windows and macOS.*"

PDF Preflighting is a CPU intensive and memory consuming operation. The minimum RAM for running the PitStop Library Container is 2GB. However, we recommend at least 2GB RAM **per worker**.

Enfocus specifically tested the PitStop Library Container on the following OS'es:

Windows

- OS: **Windows 10 64-bit**: Home or Pro (build 19041 or later), Enterprise or Education (build 18363 or later).
- Processor: 64-bit processor with Second Level Address Translation (SLAT)
- RAM (minimum): 2GB
- RAM (recommended): 2GB per worker
- Recommended hard drive type: SSD

Mac

- OS: MacOS 10.15 or newer (Monterey, Big Sur, Catalina)
- RAM (minimum): 2GB
- RAM (recommended): 2GB per worker
- Mac hardware must be a 2010 model or newer, with Intel's hardware support for Memory Management Unit (MMU) virtualization, including Extended Page Tables (EPT) and Unrestricted Mode

Note: These are the minimum system requirements for running the PitStop Library Container. We are assuming that the Docker engine is already installed on your system, which will require some additional resources from the system.

11. Dependencies

The current version of PitStop Library that is used within PitStop Library Container is 2021 update 1.

12. Appendix

Available for download:

- An example *docker-compose.yml file* that you can use. You still need to fill in your credentials and region.
- A *postman collection file* that you can use to test.

13. Copyrights

© 2023 Enfocus BV all rights reserved. Enfocus is an Esko company.

Certified PDF is a registered trademark of Enfocus BV.

Enfocus PitStop Pro, Enfocus PitStop Workgroup Manager, Enfocus PitStop Server, Enfocus BoardingPass, Enfocus Connect YOU, Enfocus Connect ALL, Enfocus Connect SEND, Enfocus StatusCheck, Enfocus CertifiedPDF.net, Enfocus PDF Workflow Suite, Enfocus Switch, Enfocus SwitchClient, Enfocus SwitchScripter, Enfocus TestDrive, Enfocus SwitchScriptTool, Enfocus Browser, PitStop Library Container, Enfocus Review, Enfocus FastLane and Enfocus Appstore are product names of Enfocus BV.

Adobe, Acrobat, Distiller, InDesign, Illustrator, Photoshop, FrameMaker, PDFWriter, PageMaker, Adobe PDF Library™, the Adobe logo, the Acrobat logo and PostScript are trademarks of Adobe Systems Incorporated.

Datalogics, the Datalogics logo, PDF2IMG™ and DLE™ are trademarks of Datalogics, Inc.

Apple, Mac, macOS, Macintosh, iPad and ColorSync are trademarks of Apple Computer, Inc. registered in the U.S. and other countries. Windows and Windows Server are registered trademarks of Microsoft Corporation.

PANTONE® Colors displayed here may not match PANTONE-identified standards. Consult current PANTONE Color Publications for accurate color. PANTONE® and other Pantone, Inc. trademarks are the property of Pantone, Inc. ©Pantone, Inc., 2006.

OPI is a trademark of Aldus Corporation.

Quark, QuarkXPress, QuarkXTensions, XTensions and the XTensions logo among others, are trademarks of Quark, Inc. and all applicable affiliated companies, Reg. U.S. Pat. & Tm. Off. and in many other countries.

Docker and the Docker logo are trademarks or registered trademarks of Docker, Inc. in the United States and/or other countries. Docker, Inc. and other parties may also have trademark rights in other terms used herein.

This product and use of this product is under license from Markzware under U.S.Patent No. 5,963,641.

Other brand and product names may be trademarks or registered trademarks of their respective holders. All specifications, terms and descriptions of products and services are subject to change without notice or recourse.