

**Qwitter**

**Qwitter**

**DevOps Phase3**

**Prepared By**

**Fares Atef**

**Qwitter DevOps Engineer**

**5/12/2023**

Contents

[Introduction 2](#_Toc152710259)

[1) Backend And Frontend Jenkins Jobs 3](#_Toc152710260)

[2) Automating Image Pulling on the Server 9](#_Toc152710261)

# Introduction

In this phase, I used Jenkins to automate the process of pushing changes to GitHub. Instead of manually pulling these changes, building the image, and executing the steps from the previous phases, Jenkins will handle the entire process automatically.

# 1) Backend And Frontend Jenkins Jobs

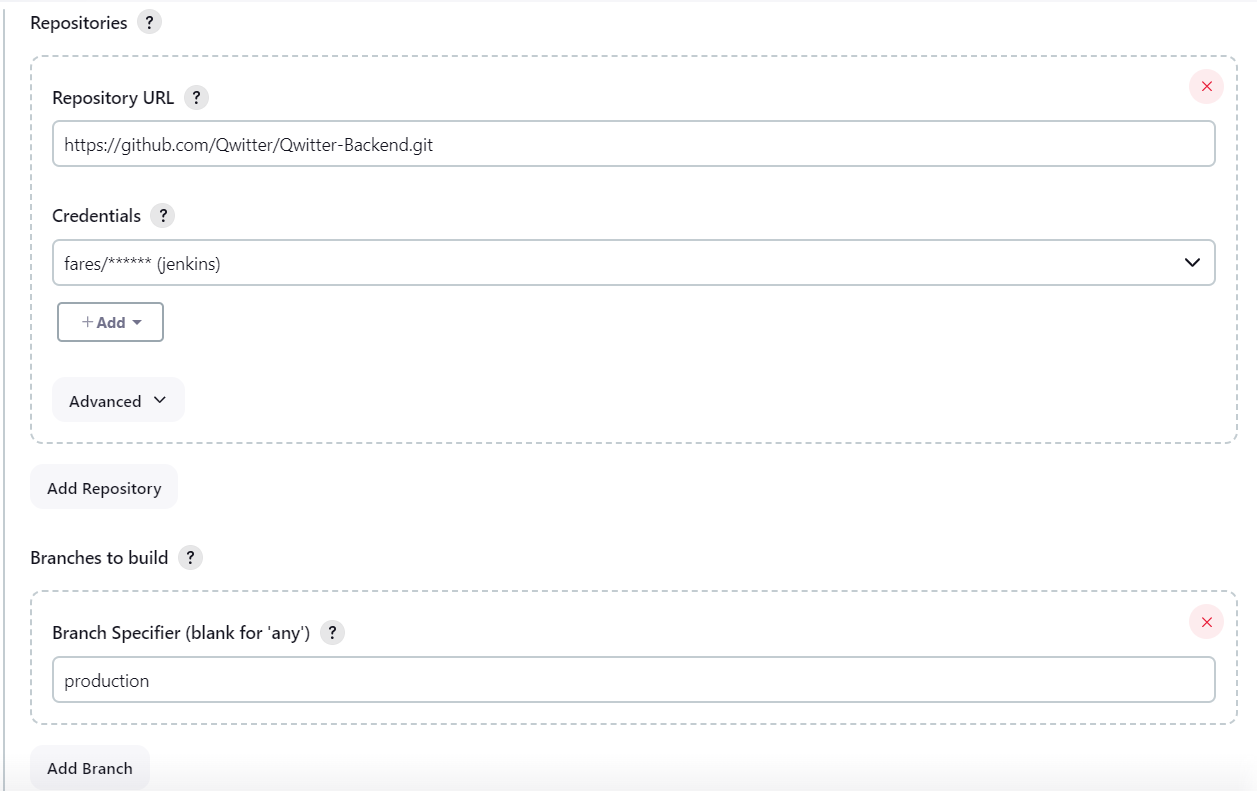
I created Jenkins jobs to automate the process of pulling changes from the GitHub repository, building a new image, pushing it to Docker Hub, pulling it on the server, and restarting the container.

For creating these jobs, I followed these steps:

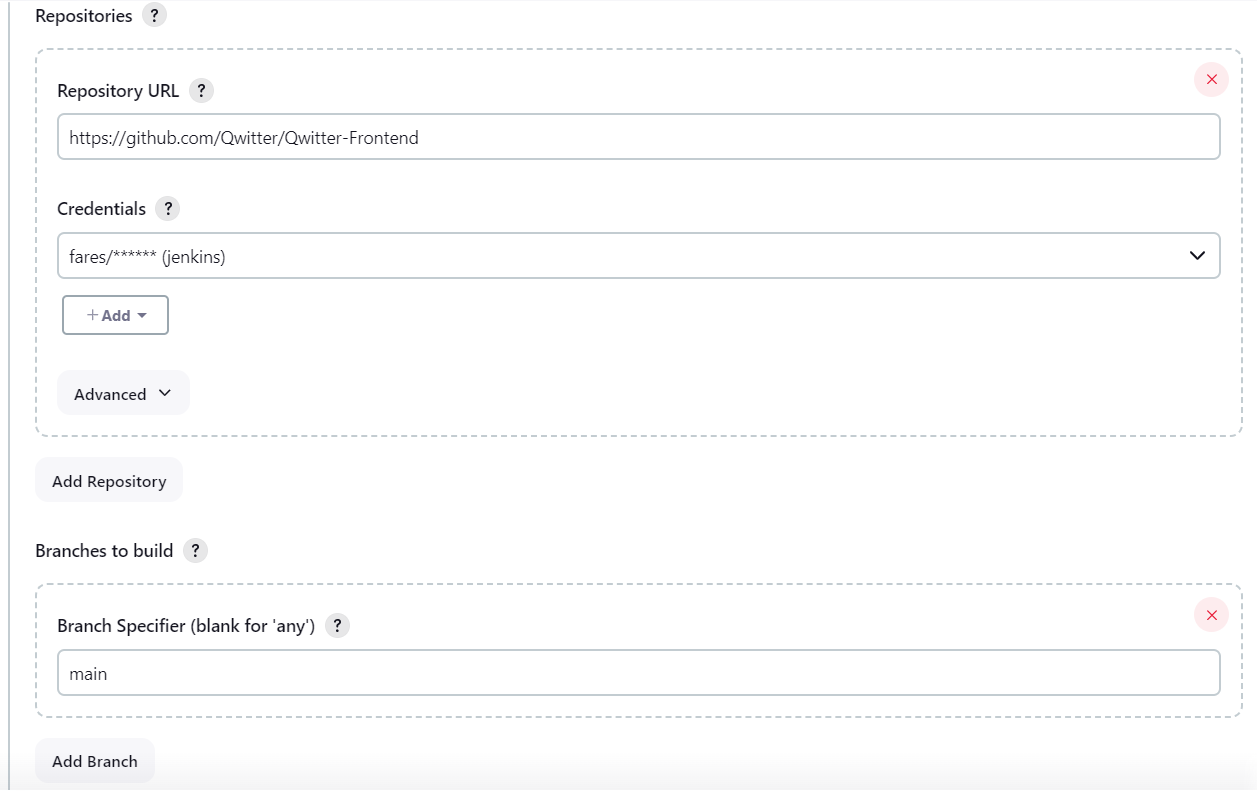
* Added the Repo URL and the corresponding credentials.
* Included a GitHub hook trigger.
* Integrated a shell script to build the Docker image and push it to Docker Hub.
* Implemented a post-build action for Email Notification.

**1) Add the Repo URL, and the Credentials**

In this step, I added the GitHub repository URL and the necessary credentials for access, given that all Qwitter repositories are private. I specified the production branch name.



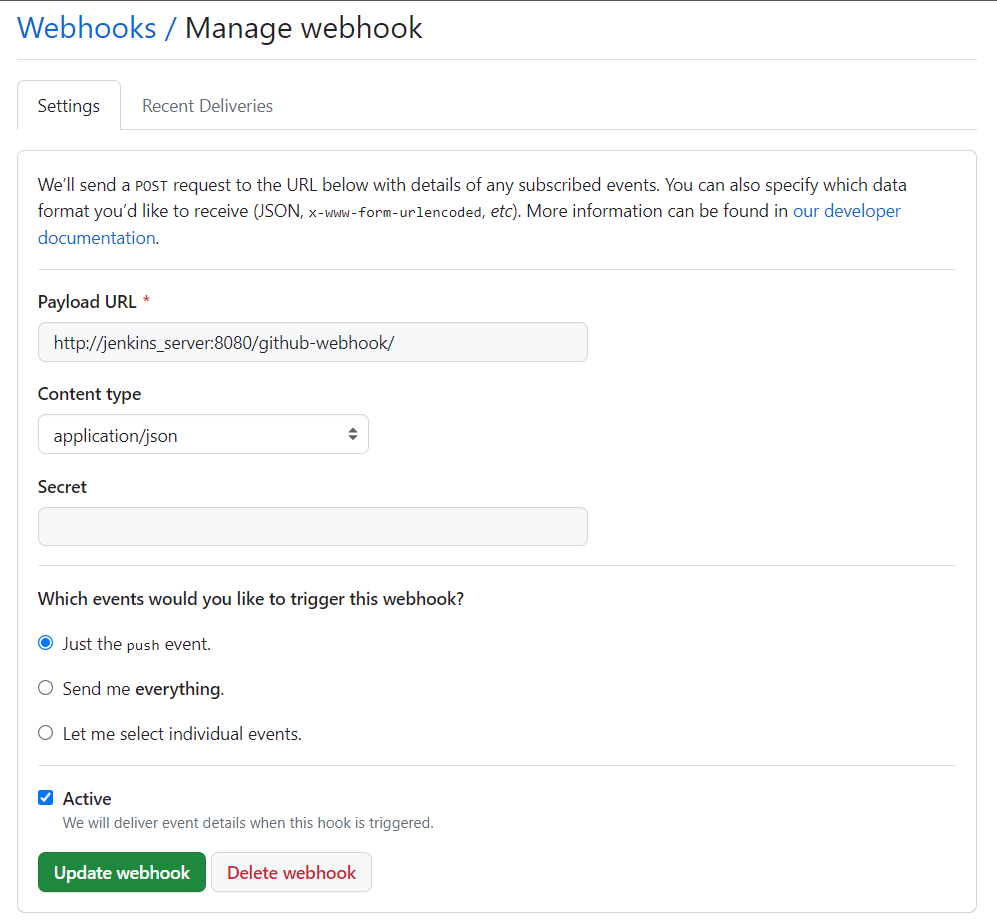
*Fig1: Git conf. for backend job*



*Fig2: Git conf. for frontend job*

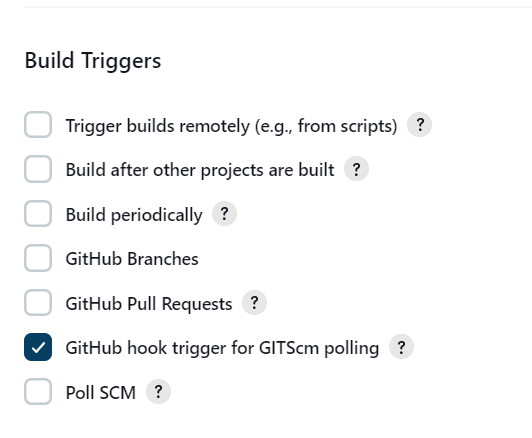
**2) Add build trigger**

In this step, I added a build trigger—a GitHub hook. This hook initiates the jobs whenever someone pushes a commit to the specified repository and branch in the Git configuration.



*Fig3: GitHub Hook conf.*

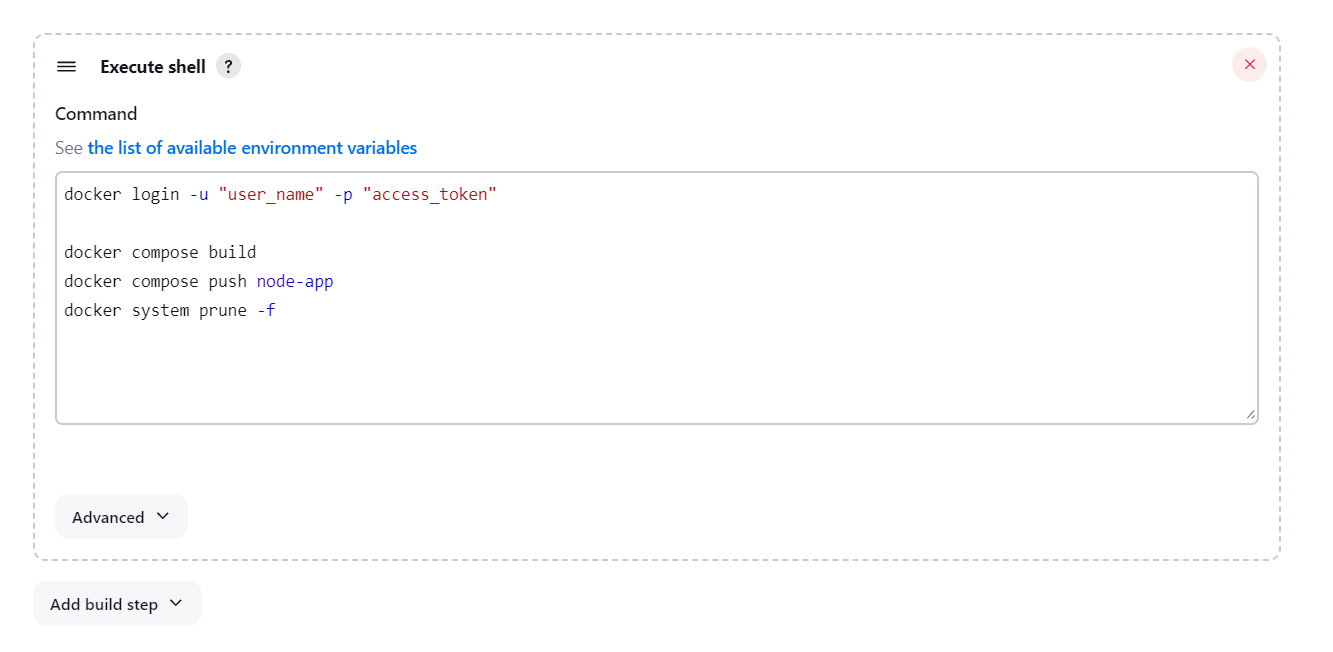
After configuring this hook, I activated this option in the Jenkins job.



*Fig4: Jenkins GitHub hook option*

**3) Shell script**

In this step, I added a shell script to build the Docker image and then push it to Docker Hub.



*Fig5: Shell Script for the backend job*

**

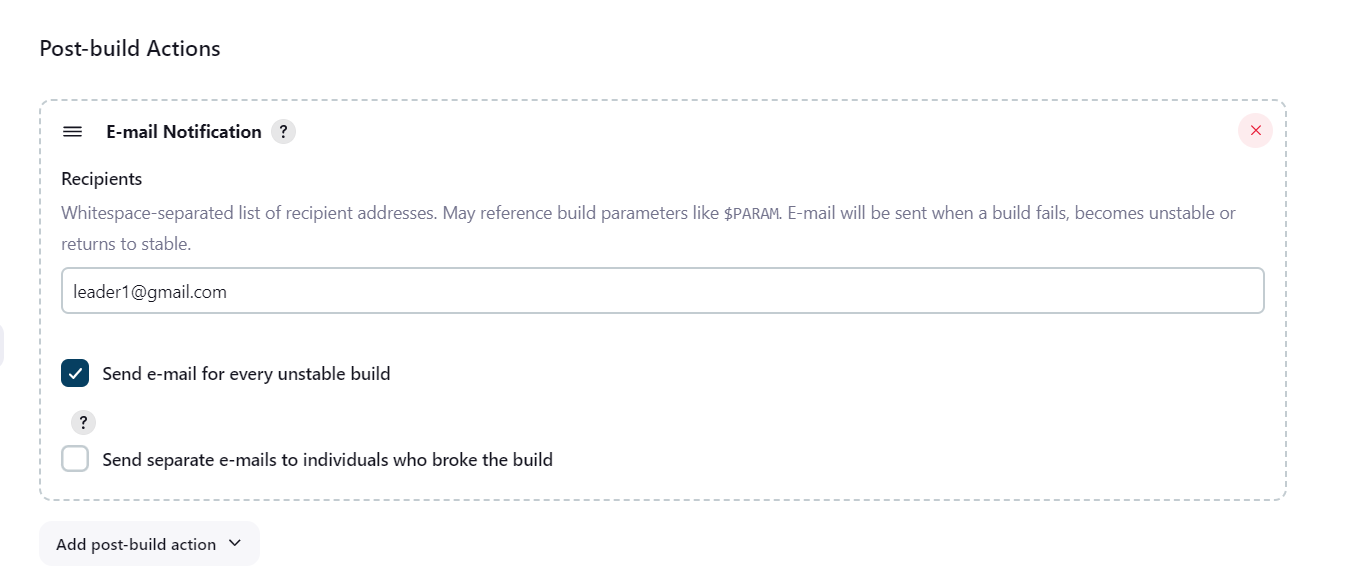
*Fig6: Shell Script for the frontend job*

**Note**: The docker system prune -f command is used to remove unused images and containers.

**4) Email Notification**

In this step, I specified the sub-team leader's email address, ensuring that they receive an email notification in case of a build failure.

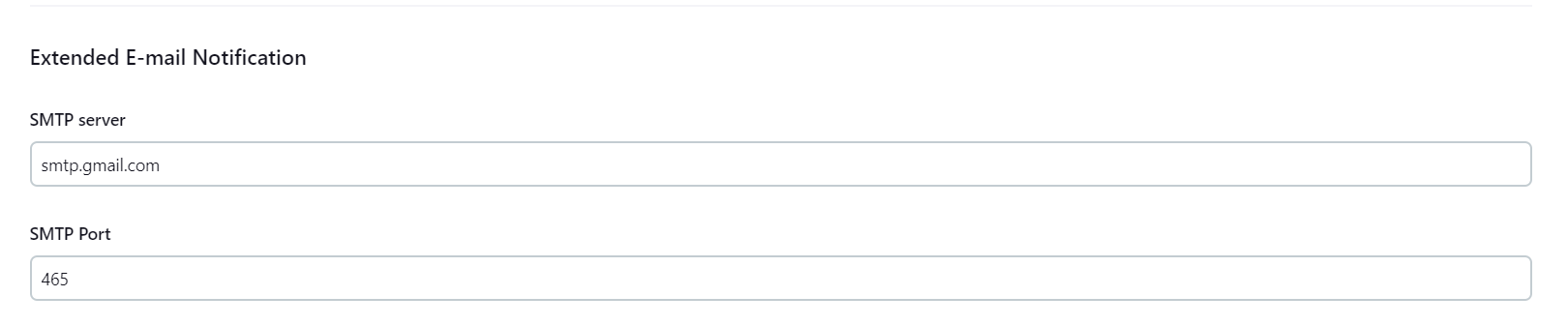
**Note**: The impact of this step will be evident in the upcoming phase when I implement the full pipeline, encompassing building and testing stages.

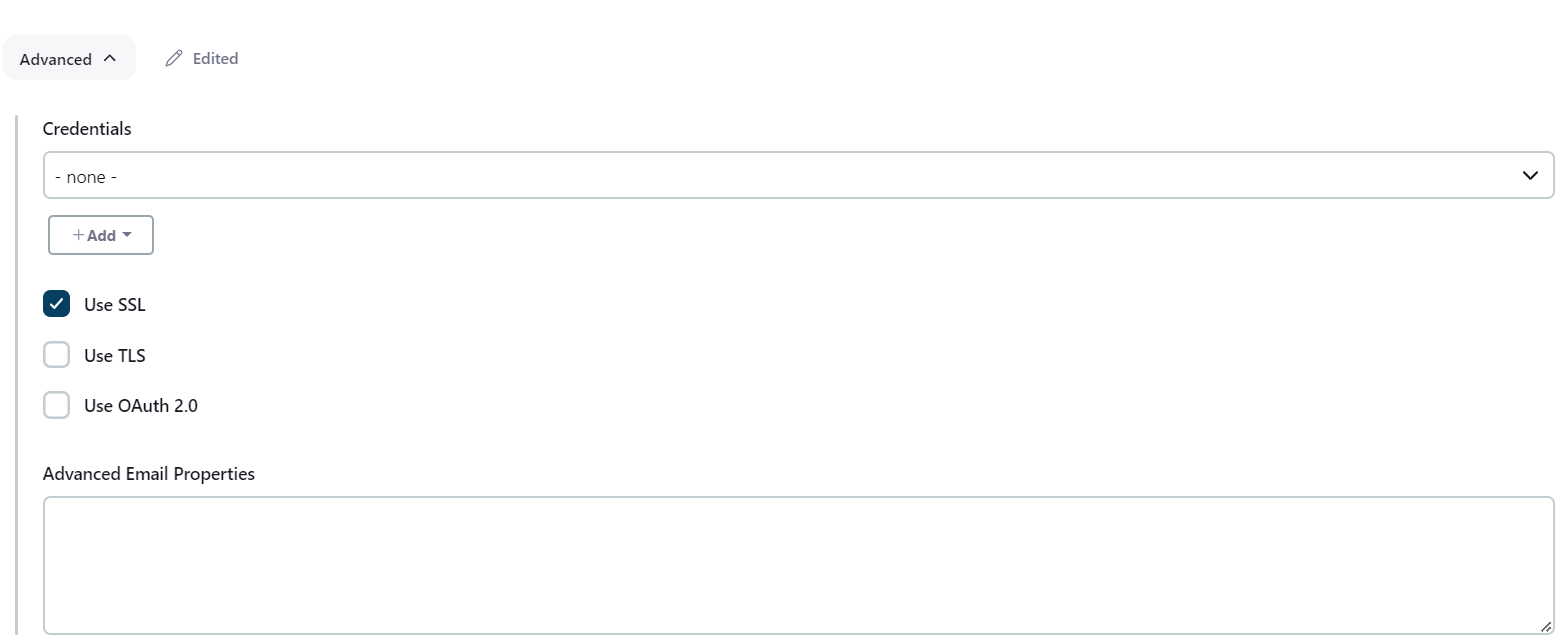


*Fig7: E-mail Notification conf.*

This step requires some initial configuration. Here are the instructions:

1. Navigate to manage Jenkins > System, Extended E-mail Notification:

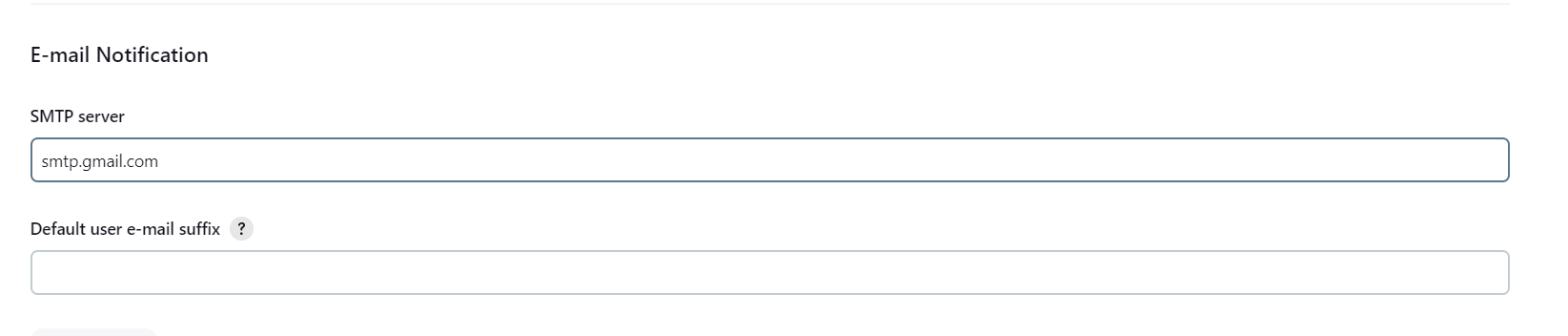


**

*Fig8: Extended E-mail Notification*

This step requires enabling the inbound rules for port 465 on the Jenkins server.

1. manage Jenkins > System, E-mail Notification:





*Fig9: E-mail Notification*

The password above is an app password generated from the Qwitter email.

# 2) Automating Image Pulling on the Server

After the job is triggered, I used a tool called Watchtower to automatically pull the new image once it pushed from Jenkins. This tool is a container that needs to run on both the backend and the frontend servers.

I used the following command to run it:

docker run -d --name watchtower -e WATCHTOWER\_TRACE=true -e WATCHTOWER\_POLL\_INTERVAL=30 -v /var/run/docker.sock:/var/run/docker.sock containrrr/watchtower service\_name

I specified a 30-seconds interval, so that it checks every 30 seconds for any new images pushed to Docker Hub.