Prerequisites

1. Docker Desktop is installed
2. PowerShell accessible machine
3. YOURLS has a successful installation with passwords and usernames set
4. NOTE: for non-Windows PCs and for your own personal issues, copy-paste this guide into ChatGPT for help navigating the setup on Linux or for troubleshooting purposes.

Open Docker Desktop and PowerShell

1. Download the performance and locust test scripts
2. Navigate to the directory storing the locust and the performance test scripts
3. Open Docker Desktop and run it as a background process

Run YOURLS + MySQL

1. In PowerShell run

```
docker network create yourls-net
```

This creates a networking space for connecting the containers.

```
docker pull yourls
```

This pulls the default image from Docker Hub

```
docker run -d `
  --name yourls-mysql `
  --network yourls-net `
  -e MYSQL_ROOT_PASSWORD=rootpassword `
  -e MYSQL_DATABASE=yourls `
  -e MYSQL_USER=yourlsuser `
  -e MYSQL_PASSWORD=yourlspass `
  Mysql:5.7
```

This establishes a running instance of the YOURLs database from the YOURLs default Docker image and Docker Compose.

```
docker run -d `
  --name yourls-app `
  --network yourls-net `
  -e YOURLS_DB_HOST=yourls-mysql:3306 `
  -e YOURLS_DB_USER=yourlsuser `
  -e YOURLS_DB_PASS=yourlspass `
  -e YOURLS_DB_NAME=yourls `
```

```
-e YOURLS_SITE="http://localhost:8080" `
-e YOURLS_USER=username `
-e YOURLS_PASS=password `
-p 8080:80 `
yourls
```

Fill in the red fields with the configurations from your installation process. This command establishes a connection. The username and password you enter will become your username and password for the site.

Open: http://localhost:8080/admin

Follow Admin set up

Configure the Performance Test:

1. **Log in with username and password**
2. **Click "Tools" in the drop-down menu**
3. **Scroll down to the section called "Secure passwordless API call signature"**
4. **Copy paste your API signature into the designated placement in the performance test and save it.**

```
python yourls_performance_test.py
```

Run the performance test.

Adding Redis Cache

```
docker pull redis:latest
```

Pull the image down from the cloud (Docker Hub).

```
docker run -d `
  --name yourls-redis `
  --network yourls-net `
  Redis:latest
```

Check that it's working:

```
docker run -it --rm `
  --network yourls-net `
  redis:latest `
  redis-cli -h yourls-redis ping
```

If you see "pong" enter "exit": you have confirmed communication and working cache.

Refresh: http://localhost:8080/admin

```
python yourls_performance_test.py
```

Add Bloom Filter:

```
docker rm -f yourls-redis
```

Strip the current yourls instance and reinstall the image using the latest version that includes a bloom filter.

```
docker run -d `
  --name yourls-redis `
  --network yourls-net `
  redislabs/rebloom:latest
```

Test the Connection and Working Bloom Filter:

```
docker run -it --rm `
  --network yourls-net `
  redislabs/rebloom:latest `
  redis-cli -h yourls-redis
```

Check the Bloom Filter:

```
BF.RESERVE myfilter 0.01 1000
BF.ADD myfilter hello
BF.EXISTS myfilter hello
BF.EXISTS myfilter world
```

Seeing output “0”, “1”, “0”, representing True/False bits, then you are good to go.

Open: http://localhost:8080/admin

Run Python tests:

```
python yourls_performance_test.py
```