# {root-project-name} (4.3.1)

Maksim Kostromin

Version 4.3.1, 2018-06-10 01:40:54 EEST

# Table of Contents

Travis CI status:

PDF:

# Chapter 1. Introduction

Streaming file server — java based project on top of spring-boot. This is a simple file-server which is allowed upload and download files with no memory limitation. It uses file multipart protocol

# Chapter 2. Installation

## 2.1. download files

*if you have docker installed and wanna use postgres, then download docker-compose.yml file*

```
wget https://github.com/daggerok/streaming-file-server/releases/download/4.3.1/docker-compose.yml
```

*file items service*

```
wget https://github.com/daggerok/streaming-file-server/releases/download/4.3.1/file-items-service-4.3.1.jar
```

*file server*

```
wget https://github.com/daggerok/streaming-file-server/releases/download/4.3.1/file-server-4.3.1.jar
```

# Chapter 3. Run

## 3.1. postgres database

*install using postgres in docker*

```
# docker compose file for postgres database
docker-compose up -d

# file-items data service
java -jar file-items-service-4.3.1.jar --spring.profiles.active=db-pg

# file server
java -jar file-server-4.3.1.jar --app.upload.path=./path/to/file-storage

# cleanup
docker-compose down -v
```

## 3.2. in-memory h2 database

*if you do not have docker — feel free to use h2 in memory database for file items service:*

```
java file-items-service-4.3.1.jar
# or
java file-items-service.jar --spring.profiles.active=db-h2
```

1. and then run file server:

```
java file-items-service-{project-version}.jar --spring.profiles.active=db-h2
```

## 3.3. take advantages of spring-boot executable jar

*if you are using bash — run even simply*

```
wget https://github.com/daggerok/streaming-file-server/releases/download/4.3.1/file-
items-service-4.3.1.jar
bash file-items-service-4.3.1.jar

wget https://github.com/daggerok/streaming-file-server/releases/download/4.3.1/file-
server-4.3.1.jar
bash file-server-4.3.1.jar --app.upload.path=./path/to/file-storage
```

# Chapter 4. Usage scripts

## 4.1. simplicity bootstrap with automation shell-script

### 4.1.1. unix (bash)

*postgres in docker*

```
# get
wget https://github.com/daggerok/streaming-file-
server/releases/download/4.3.1/application.bash

# start
bash application.bash start ./path/to/file-storage

# stop
bash application.bash stop

# cleanup
bash application.bash clean ./path/to/file-storage
```

download: application.bash

*h2 in-memory database*

```
# fetch
wget https://github.com/daggerok/streaming-file-
server/releases/download/4.3.1/application-h2.bash

# start
bash application-h2.bash start ./path/to/file-storage

# stop
bash application-h2.bash stop

# cleanup
bash application-h2.bash clean ./path/to/file-storage
```

download: application-h2.bash

*note: binaries* `wget`, `docker-compose` *and of course jre (binaries:* `java` *and* `jps`*) are required* ==== windows (batch cmd) .postgres in docker

```
@rem start
application.cmd start path\to\file-storage

@rem stop
application.cmd stop

@rem cleanup
application.cmd clean path\to\file-storage
```

download: application.cmd

*h2 in-memory database*

```
@rem start
application-h2.cmd start path\to\file-storage

@rem stop
application-h2.cmd stop

@rem cleanup
application-h2.cmd clean path\to\file-storage
```

download: application-h2.cmd

*note: binaries* `wget`, `docker-compose` *and of course jre (binaries:* `java` *and* `jps`*) are required*

# Chapter 5. Create new release

to create new release do next

1. bump version in:
    a. build.gradle
    b. README.md
    c. scripts/application.cmd
    d. scripts/application.bash
    e. scripts/application-h2.cmd
    f. scripts/application-h2.bash
2. comment scipts tests in .travis.yml
3. commit, push and check CI if builds was successfully passed
4. create release on github, put:
    a. modules/apps/*/build/libs/*
    b. scripts/*
    c. modules/docker/postgres/docker-compose.yml
5. uncomment .travis.yml
6. commit, push and check CI again to verify if scripts tests was successfully passed

download all files here

---

Enjoy :)