**Deployment of NPM-repositories with Artipie.**

In this article I am going to demonstrate how to deploy own NPM-repository server with help of Artipie.

I will show how to configure and start new NPM-repository, how use standard npm-tool to publish NPM-package to repository and how to install NPM-package receiving it from repository.

**Introduction of Artipie**

[**Artipie**](https://github.com/artipie) is a free open source binary artifact management tool project under a MIT license.

Artipie is quickly growing project that was born in 2020 and currently supports plenty repository types:

* NPM - for storing and sharing of Node JS packages.
* Docker - Docker registry for images.
* RPM - repository of .rpm-files for RHEL, CentOs, Fedora, PCLinuxOS, AlmaLinux, openSUSE, OpenMandriva, Oracle Linux, etc.
* Debian - repository packages for Debian-based Linux distros(Debian, Mint, Ubuntu, MX Linux, Raspberry Pi OS, Parrot OS, etc).
* Go - storage of Go packages.
* Maven - Java, Kotlin, Groovy, Scala, Clojure artifacts of such types as .jar, .war, .klib, etc.
* PyPI - Python packages index.
* Anaconda - packages for data science for Python, R, Lua, C, C++ and etc. languages.
* HexPM - for storing and sharing packages for Elixir and Erlang languages.
* Gem - hosting service of RubyGem for Ruby language.
* Helm - Helm charts repository.
* NuGet - .NET package hosting service.
* Composer - PHP source packages.
* Binary(files) storage - for hosting any types of files.

Artipie is flexible enough to create custom configurations so you can use it like “Lego constructor” by organizing own repository or multiple repositories for storing artifacts.

Artipie consists of following main parts:

* Storage is used for saving and receiving artifacts on server.
* Repository maintains all logic to work with concreate type of artifact for example NPM, RPM and etc. Different repository types are implemented as Artipie-adaptors.
* Artipe-engine is server of Artipie.
* Artipie-fronend provides web-dashboard to create and configure repositories.

Storages of Artipie are used to keep/receive artifacts in/from some source of data. Atripie supports following range of well-known storage types:

* File system storage
* Amazon S3 storage
* Redis storage

Additionally, Artipe provides custom storage type to create a new custom storage type.

Artipie-engine is designed as binary artifact’s storage management system for high loads. Artipie-engine design follows principals of reactive approaches that suppose working asynchronous with files and network operations.

Artipie is a secure server that control access by using users and groups, granting permissions on resources and operations.

Artipie allows to store all artifacts either in one “flat”-repository or separate set of repositories.

If you are interested in getting more information about Artipie please visit [github](https://github.com/artipie/artipie) and [wiki](https://github.com/artipie/artipie/wiki) pages.

**Starting of NPM-repository.**

Artipie is java application and there are two ways to launch it as:

* Java jar-file
* Docker container

In this article I use Docker-engine and Docker compose as deployment environment for Artipie NPM-repository.

So first of all you should be sure that [Docker-engine](https://docs.docker.com/get-docker/) and [Docker compose](https://docs.docker.com/compose/install/) are already installed on your work station.

I use Windows 10 operation system that already installed on my work station, but you can use either kind of Unix/Linux/MacOs operation system that has support of Docker-engine and Docker compose.

All configuration files and deployed on Artipie npm-packages I am going to keep inside one folder “C:\\artipie\”.

First of all we need create a configuration of Artipie for Docker compose, for it I will get ‘docker-compose.yaml’ file that you can obtain by:

* Downloading file from [Artipie’s github page](https://github.com/artipie/artipie/blob/master/docker-compose.yaml)
* Cloning [Artipie git repository](https://github.com/artipie/artipie) by using [git](https://git-scm.com/) and [clone](https://git-scm.com/docs/git-clone) commands

I put downloaded ‘docker-compose.yaml’ file into “C:\artipie\” folder:

|  |
| --- |
| c:\artipie>  docker-compose.yaml |

The content of ‘docker-compose.yaml’ file is simple enough and defines two Docker-applications of Artipie:

* First one has name ‘artipie’. This application is Artipie-engine.
* Second one has name ‘front’. This applicatin provides we-dashboard to control repositories.

I use last versions of Artipie’s Docker-images ‘artipie/artipie:latest’ and ‘artipie/front:latest’. List of Artipie’s images can be found on docker hub [page](https://hub.docker.com/u/artipie).

‘docker-compose.yaml’ contains port forwarding instructions to allow get access to ‘artipe’-application on local port ‘8081’ and ‘dashboard’ on local port 8080.

Listing of ‘docker-compose.yaml’:

|  |
| --- |
| version: "3.3"  services:  artipie:  image: artipie/artipie:latest  container\_name: artipie  restart: unless-stopped  environment:  - ARTIPIE\_USER\_NAME=artipie  - ARTIPIE\_USER\_PASS=artipie  networks:  - artipie-net  ports:  - "8081:8080"  volumes:  # change /usr/local/artipie to any convenient location, this is the config folder  - /usr/local/artipie:/var/artipie/repo  front:  image: artipie/front:latest  container\_name: front  restart: unless-stopped  networks:  - artipie-net  environment:  - ARTIPIE\_USER\_NAME=artipie  - ARTIPIE\_USER\_PASS=artipie  - TKN\_KEY=abc123  volumes:  # change /usr/local/artipie to any convenient location, this is the config folder  - /usr/local/artipie:/var/artipie/repo  ports:  - "8080:8080"  networks:  artipie-net:  driver: bridge |

I modify a little this base configuration to map local folder to container directories of Artipie:

1. Because Artipie-engine (‘artipe’-application) supposes to find its configuration in ‘/etc/artipie/’ directory so I am going to add mapping of local folder ‘C:\artipie\config’ to ‘/etc/artipie/’ container’s directory.
2. I am going to store configuration of all Artipie-repositories in container’s directory ‘/var/artipie/repo’ so I map a local folder ‘C:\artipie\repo’ to it.
3. I am going to store npm-packages of Artipie-repository in container directory ‘/var/artipie/packages’ so I map a local folder ‘C:\artipie\packages’ to it.

Listing of modified ‘docker-compose.yaml’:

(See highlighted text with added changes)

|  |
| --- |
| version: "3.3"  services:  artipie:  image: artipie/artipie:latest  container\_name: artipie  restart: unless-stopped  environment:  - ARTIPIE\_USER\_NAME=artipie  - ARTIPIE\_USER\_PASS=artipie  networks:  - artipie-net  ports:  - "8081:8080"  volumes:  # change /usr/local/artipie to any convenient location, this is the config folder  - C:\artipie\config:/etc/artipie/ # mount of Artipie configuration  - C:\artipie\repo:/var/artipie/repo # mount of repository configurations  - C:\artipie\packages:/var/artipie/packages # mount of npm-packages directory  front:  image: artipie/front:latest  container\_name: front  restart: unless-stopped  networks:  - artipie-net  environment:  - ARTIPIE\_USER\_NAME=artipie  - ARTIPIE\_USER\_PASS=artipie  - TKN\_KEY=abc123  volumes:  # change /usr/local/artipie to any convenient location, this is the config folder  - C:\artipie\repo:/var/artipie/repo  ports:  - "8080:8080"  networks:  artipie-net:  driver: bridge |

On next step I need to put configurations of Artipie-engine and NPM-repository.

Artipie-engine’s main configuration defines:

* Layout as ‘flat’ that means to store all artifacts in on place: **layout: flat**
* Type of storage as ‘fs’ that means to use file-storage: **type: fs**
* Path to directory where repository configurations are stored: **path: /var/artipie/repo**

Listing of ‘C:\artipie\config\artipie.yml’:

|  |
| --- |
| meta:  storage:  type: fs  path: /var/artipie/repo #path to repository configurations  layout: flat |

All repositories configured to be stored in directory ‘: /var/artipie/repo’ that mapped on local folder ‘C:\artipie\repo\’. So now I define and put NPM-repository configuration into file ‘C:\artipie\repo\my-npm.yaml’. This NPM-repository configuration contains:

* Type of repository as ‘npm’: **type: npm**
* Url of repository: **url:** [**http://localhost:8080/my-npm**](http://localhost:8080/my-npm)
* Place where npm-packages are stored: **path: /var/artipie/packages.** This path is mapped on local folder ‘C:\artipie\packages’.
* Permissions to access to npm-repository: any one can download and publish npm-packages.

Listing of ‘C:\artipie\repo\my-npm.yaml:

|  |
| --- |
| repo:  type: npm  url: http://localhost:8080/my-npm  storage:  type: fs  path: /var/artipie/packages  permissions:  "\*":  - download  - publish |

Resulting view of all files and folders on my local work station:

|  |
| --- |
| C:\artipie  docker-compose.yaml  config/  artipie.yml  repo/  my-npm.yaml  packages/ |

Now we are ready to launch Artipie in Docker compose.

Run Windows command line terminal (‘cmd’) and inside folder “C:\artipie” invoke following command:

|  |
| --- |
| docker-compose up |

This command starts Artipie-server and Artipie-dashboard.

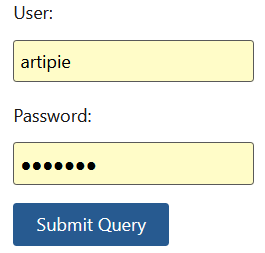
Listing of startup log:

|  |
| --- |
| docker compose up  Network artipie\_artipie-net Creating  Network artipie\_artipie-net Created  Container front Creating  Container artipie Creating  Container front Created  Container artipie Created  Attaching to artipie, front  front | SLF4J: Class path contains multiple SLF4J providers.  front | SLF4J: Found provider [org.slf4j.log4j12.Log4j12ServiceProvider@2f7a2457]  front | SLF4J: Found provider [org.slf4j.simple.SimpleServiceProvider@566776ad]  front | SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.  front | SLF4J: Actual provider is of type [org.slf4j.log4j12.Log4j12ServiceProvider@2f7a2457]  front | [INFO] 2022-09-22 12:50:15,976 main com.artipie.front.Service - starting service on port: 8080  front | [INFO] 2022-09-22 12:50:16,010 Thread-0 org.eclipse.jetty.util.log - Logging initialized @522ms to org.eclipse.jetty.util.log.Slf4jLog  front | [INFO] 2022-09-22 12:50:16,066 Thread-0 spark.embeddedserver.jetty.EmbeddedJettyServer - == Spark has ignited ...  front | [INFO] 2022-09-22 12:50:16,066 Thread-0 spark.embeddedserver.jetty.EmbeddedJettyServer - >> Listening on 0.0.0.0:8080  front | [INFO] 2022-09-22 12:50:16,068 Thread-0 org.eclipse.jetty.server.Server - jetty-9.4.31.v20200723; built: 2020-07-23T17:57:36.812Z; git: 450ba27947e13e66baa8cd1ce7e85a4461cacc1d; jvm 17.0.2+8-86  front | [INFO] 2022-09-22 12:50:16,088 Thread-0 org.eclipse.jetty.server.session - DefaultSessionIdManager workerName=node0  front | [INFO] 2022-09-22 12:50:16,088 Thread-0 org.eclipse.jetty.server.session - No SessionScavenger set, using defaults  front | [INFO] 2022-09-22 12:50:16,092 Thread-0 org.eclipse.jetty.server.session - node0 Scavenging every 600000ms  front | [INFO] 2022-09-22 12:50:16,116 Thread-0 org.eclipse.jetty.server.AbstractConnector - Started ServerConnector@4eb377f2{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}  front | [INFO] 2022-09-22 12:50:16,116 Thread-0 org.eclipse.jetty.server.Server - Started @630ms  front | [INFO] 2022-09-22 12:50:16,150 main com.artipie.front.Service - service started on port: 8080  artipie | [INFO] 2022-09-22 12:50:16,399 main com.artipie.VertxMain - Used version of Artipie: latest  artipie | [INFO] 2022-09-22 12:50:16,412 main org.eclipse.jetty.util.log - Logging initialized @590ms to org.eclipse.jetty.util.log.Slf4jLog  artipie | [INFO] 2022-09-22 12:50:16,776 main org.reflections.Reflections - Reflections took 57 ms to scan 2 urls, producing 39 keys and 89 values  artipie | [INFO] 2022-09-22 12:50:16,796 main com.artipie.asto.factory.Storages - Initiated storage factory [type=fs, class=FileStorageFactory]  artipie | [INFO] 2022-09-22 12:50:16,797 main com.artipie.asto.factory.Storages - Initiated storage factory [type=s3, class=S3StorageFactory]  artipie | [INFO] 2022-09-22 12:50:16,797 main com.artipie.asto.factory.Storages - Initiated storage factory [type=etcd, class=EtcdStorageFactory]  artipie | [INFO] 2022-09-22 12:50:16,907 main com.artipie.VertxMain - Artipie was started on port 8080  artipie | [INFO] 2022-09-22 12:50:16,922 ForkJoinPool.commonPool-worker-1 com.artipie.asto.fs.FileStorage - Found 1 objects by the prefix "" in /var/artipie/repo by /var/artipie/repo: [my-npm.yaml]  artipie | [INFO] 2022-09-22 12:50:16,962 main com.artipie.VertxMain - Artipie repo 'my-npm' was started on port 8080  artipie | [INFO] 2022-09-22 12:50:16,982 main org.quartz.impl.StdSchedulerFactory - Using default implementation for ThreadExecutor  artipie | [INFO] 2022-09-22 12:50:16,983 main org.quartz.simpl.SimpleThreadPool - Job execution threads will use class loader of thread: main  artipie | [INFO] 2022-09-22 12:50:16,991 main org.quartz.core.SchedulerSignalerImpl - Initialized Scheduler Signaller of type: class org.quartz.core.SchedulerSignalerImpl  artipie | [INFO] 2022-09-22 12:50:16,991 main org.quartz.core.QuartzScheduler - Quartz Scheduler v.2.3.2 created.  artipie | [INFO] 2022-09-22 12:50:16,991 main org.quartz.simpl.RAMJobStore - RAMJobStore initialized.  artipie | [INFO] 2022-09-22 12:50:16,991 main org.quartz.core.QuartzScheduler - Scheduler meta-data: Quartz Scheduler (v2.3.2) 'DefaultQuartzScheduler' with instanceId 'NON\_CLUSTERED'  artipie | Scheduler class: 'org.quartz.core.QuartzScheduler' - running locally.  artipie | NOT STARTED.  artipie | Currently in standby mode.  artipie | Number of jobs executed: 0  artipie | Using thread pool 'org.quartz.simpl.SimpleThreadPool' - with 10 threads.  artipie | Using job-store 'org.quartz.simpl.RAMJobStore' - which does not support persistence. and is not clustered.  artipie |  artipie | [INFO] 2022-09-22 12:50:16,992 main org.quartz.impl.StdSchedulerFactory - Quartz scheduler 'DefaultQuartzScheduler' initialized from default resource file in Quartz package: 'quartz.properties'  artipie | [INFO] 2022-09-22 12:50:16,992 main org.quartz.impl.StdSchedulerFactory - Quartz scheduler version: 2.3.2  artipie | [INFO] 2022-09-22 12:50:16,992 main org.quartz.core.QuartzScheduler - Scheduler DefaultQuartzScheduler\_$\_NON\_CLUSTERED started.  artipie | [WARN] 2022-09-22 12:50:17,408 vert.x-eventloop-thread-1 com.artipie.api.RestApi - File credentials are not set, users API is not available  artipie | [INFO] 2022-09-22 12:50:17,452 vert.x-eventloop-thread-1 com.artipie.api.RestApi - Rest API started |

My congratulations!

You have just launched own NPM-repository!

Note:

Additionally, you can open dashboard in browser to control repository configurations: <http://localhost:8080/dashboard/artipie>

In prompted login form please use the built-in credentials: ‘artipie’ as user and ‘artipie’ as password.

To control all aspects of Artipie the REST API services can be used also.

But in this tutorial, I suppose to modify configuration of Artipie manually by editing files in folder ‘c:\artipie’.

**Publishing of package**

Now we have running NPM-repository and this is time to create npm-package for publishing.

Let’s create simple node js file in separate folder, I create folder ‘C:\package\’ and put there two files: index.js and package.json.

The content of index.js:

|  |
| --- |
| exports.greeting = function () {  console.log("Hello world!");  }; |

The content of package.json:

|  |
| --- |
| {  "name": "greeting",  "version": "1.0.0",  "description": "Greeting",  "main": "index.js",  "author": "",  "license": "ISC"  } |