**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 your artifacts.

Artipie consists of following main parts:

* Storage - keeps artifacts in some source of data for example it can store artifacts in file system.
* Repository - understands format of some type of artifacts and organizes work with this type of artifacts. For example, there is NPM-repository type that can work with NPM-packages and npm-package manager tool.
* Artipe-engine - is a server of Artipie.
* Artipie-fronend - provides web-based dashboard to control repository configurations.

Atripie supports following types of storages:

* File system storage
* Amazon S3 storage
* Redis storage
* Custom storage type

Artipie-engine is designed as binary artifact’s storage management system for high loads. Artipie-engine design follows principals of reactive approach that supposes usage asynchronous file and network operations.

Artipie provides server access control by using users and groups, granting permissions on resources and operations.

Artipie provides two kinds of repository layouts:

* “flat” when all artifacts lay in one repository
* “org” when artifacts are organized in separate user’s repositories.

To get more information about Artipie please visit [github](https://github.com/artipie/artipie) and [wiki](https://github.com/artipie/artipie/wiki) pages.

**Preparation of NPM-repository.**

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

* As Java application from jar-file
* As container in Docker-engine

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

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

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

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

On next step I need to put two configuration files to “C:\artipie” folder: one for Artipie-engine and second for NPM-repository.

Artipie-engine configuration file is located by path: ‘C:\artipie\config\artipie.yml’ and it defines following parameters:

* **type: fs**

The storage type, where ‘fs’ means to use file-storage type for storing artifacts in file system.

* **path: /var/artipie/repo**

Specifies the path to directory where stored configurations of all repositories, including NPM-repository configuration used in this article.

* **layout: flat**

The artifact’s layout definition, ‘flat’ means to store all artifacts in one directory of storage.

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

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

NPM-repository configuration file is located by path: ‘C:\artipie\repo\my-npm.yaml’ and it defines following paramters:

* **type: npm**

The type of repository, here is ‘npm’ for deployment of NPM-repository.

* **url:** [**http://localhost:8080/my-npm**](http://localhost:8080/my-npm)

The url of repository. It means http-enpoint to access to NPM-repository by ‘npm’ command tool.

* **path: /var/artipie/packages**

Specifies the path where published npm-packages should be stored**.**

* **permissions**:

Configure access permissions on npm-repository’s artifacts: here is everyone has permissions to 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 files on my local work station:

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

Now we are ready to launch Artipie as container of Docker-engine.

Run following command in terminal:

|  |
| --- |
| docker run \  -v C:\artipie\config:/etc/artipie/ \  -v C:\artipie\repo:/var/artipie/repo \  -v C:\artipie\packages:/var/artipie/packages \  -p 8081:8080  artipie/artipie:latest |

The command starts Artipie-engine inside Docker-engine. Command mounts 3 local directories to Docker-container as volumes and forwards local port 8081 to container’s port 8080:

1. Artipie-engine expects to find its main configuration inside ‘/etc/artipie/’ directory so we mount local directory ‘C:\artipie\config’ to ‘/etc/artipie/’ container’s directory.
2. Artipie-engine looks up repository configurations inside directory ‘/var/artipie/repo’ so we mount local directory ‘C:\artipie\repo’ to it.
3. Published npm-packages are in directory ‘/var/artipie/packages’ so we mount local directory ‘C:\artipie\packages’ to it.

**Publishing of package**

Now we have running Artipie-engine and deployed NPM-repository. Let’s create new npm-package and publish it to NPM-repository.

I create node JS package folder: ‘C:\ workdir\@hello\simple-npm-project\’ and put to it two files of package: index.js and package.json.

Listing of index.js:

|  |
| --- |
| console.log("Hello world"); |

Listing of package.json:

|  |
| --- |
| {  "name": "@hello/simple-npm-project",  "version": "1.0.1",  "description": "",  "main": "index.js",  "scripts": {  "test": "echo \"Error: no test specified\" && exit 1"  },  "author": "",  "license": "ISC"  } |

Now go to ‘C:\ workdir’ folder and run following command in terminal:

|  |
| --- |
| npm publish @hello\simple-npm-project --registry <http://localhost:8081/my-npm> |

The parameter ‘--registry’ is required to specify Artipie’s NPM-repository.

Command above published ‘@hello/simple-npm-project’ NPM-package to Artipie’s NPM-repository.

Published package can be found inside folder ‘C:\artipie\packages’.

**Installing of package**

Let’s install package ‘@hello\simple-npm-project’ by using standard command ‘npm install’:

|  |
| --- |
| npm install @hello/simple-npm-project --registry http://localhost:8081/my-npm |

The parameter ‘--registry’ is required to specify Artipie’s NPM-repository.

Command above downloads NPM-package from Artipie NPM-repository.

**Conclusion**

The article demonstrates the simplicity and flexibility usage of Artipie to deploy custom configuration of NPM-repository and interaction with deployed NPM-repository by standard ‘npm’ tool.