# Hosting a Maven repository on github

Asked 12 years ago Modified 3 years, 1 month ago Viewed 137k times

337

I have a fork of a small open sourced library that I'm working on github. I'd like to make it available to other developers via maven, but I don't want to run my own Nexus server, and because it's a fork I can't easily deploy it to oss.sonatype.org.

What I'd like to do is to deploy it to github so that others can access it using maven. What's the best way to do this?

maven github github-pages mvn-repo

Share
Improve this question
Follow

edited Jul 10, 2021 at 0:06
George Z.
6,798 ● 4 ● 32 ● 52

asked Dec 23, 2012 at 18:43
emmby
100k ● 66 ● 194 ● 251

5 what licensing issues are you facing in OSS Sonatype? Just curious since I use it myself. – Archimedes Trajano Feb 3, 2014 at 6:10

5 There's a tool that allows you to expose your GitHub repo via maven directly. jitpack.io stackoverflow.com/a/28483461/3975649 – metrimer Apr 24, 2015 at 9:02

1 Github also announced a package registry that supports maven. Currently in public-beta: github.com/features/package-registry – Kaan Jul 14, 2019 at 10:23

## 9 Answers

Sorted by: Highest score (default)

511

The best solution I've been able to find consists of these steps:

1. Create a branch called `mvn-repo` to host your maven artifacts.
2. Use the github site-maven-plugin to push your artifacts to github.
3. Configure maven to use your remote `mvn-repo` as a maven repository.

There are several benefits to using this approach:

- Maven artifacts are kept separate from your source in a separate branch called `mvn-repo`, much like github pages are kept in a separate branch called `gh-pages` (if you use github pages)
- Unlike some other proposed solutions, it doesn't conflict with your `gh-pages` if you're using them.

- Ties in naturally with the deploy target so there are no new maven commands to learn. Just use `mvn deploy` as you normally would

The typical way you deploy artifacts to a remote maven repo is to use `mvn deploy`, so let's patch into that mechanism for this solution.

First, tell maven to deploy artifacts to a temporary staging location inside your target directory. Add this to your `pom.xml`:

```
<distributionManagement>
    <repository>
        <id>internal.repo</id>
        <name>Temporary Staging Repository</name>
        <url>file://${project.build.directory}/mvn-rep
    </repository>
</distributionManagement>

<plugins>
    <plugin>
        <artifactId>maven-deploy-plugin</artifactId>
        <version>2.8.1</version>
        <configuration>

<altDeploymentRepository>internal.repo::default::file:
repo</altDeploymentRepository>
        </configuration>
    </plugin>
</plugins>
```

Now try running `mvn clean deploy`. You'll see that it deployed your maven repository to `target/mvn-repo`. The next step is to get it to upload that directory to GitHub.

Add your authentication information to `~/.m2/settings.xml` so that the github `site-maven-plugin` can push to GitHub:

```
<!-- NOTE: MAKE SURE THAT settings.xml IS NOT WORLD RE
<settings>
  <servers>
    <server>
      <id>github</id>
      <username>YOUR-USERNAME</username>
      <password>YOUR-PASSWORD</password>
    </server>
  </servers>
</settings>
```

(As noted, please make sure to `chmod 700 settings.xml` to ensure no one can read your password in the file. If someone knows how to make site-maven-plugin prompt for a password instead of requiring it in a config file, let me know.)

Then tell the GitHub `site-maven-plugin` about the new server you just configured by adding the following to your pom:

```
<properties>
    <!-- github server corresponds to entry in ~/.m2/s
    <github.global.server>github</github.global.server
</properties>
```

Finally, configure the `site-maven-plugin` to upload from your temporary staging repo to your `mvn-repo` branch on Github:

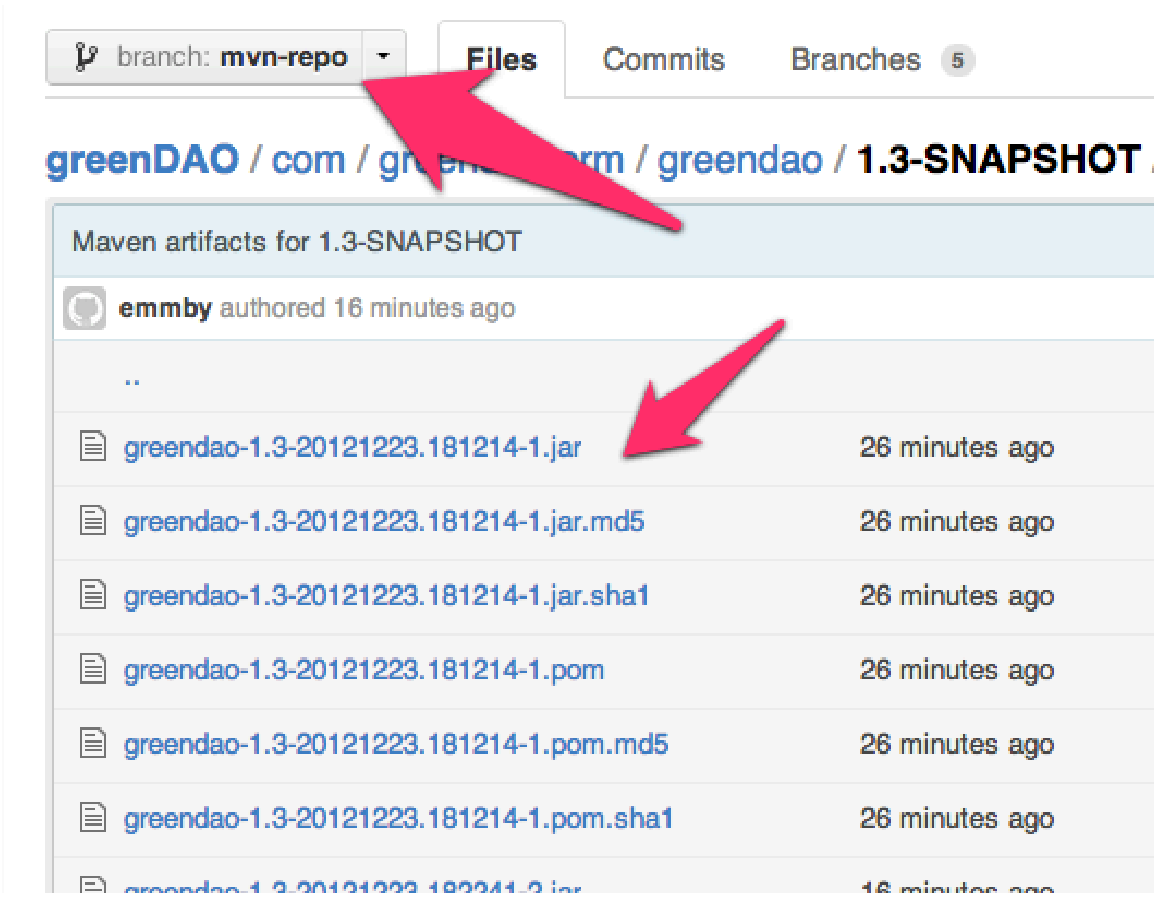
```
<build>
    <plugins>
        <plugin>
            <groupId>com.github.github</groupId>
            <artifactId>site-maven-plugin</artifactId>
            <version>0.11</version>
            <configuration>
                <message>Maven artifacts for ${project
git commit message -->
                <noJekyll>true</noJekyll>
disable webpage processing -->
                <outputDirectory>${project.build.direc
repo</outputDirectory> <!-- matches distribution manag
above -->
                <branch>refs/heads/mvn-repo</branch>
remote branch name -->
                <includes><include>**/*</include></inc
                <repositoryName>YOUR-REPOSITORY-NAME</
github repo name -->
                <repositoryOwner>YOUR-GITHUB-USERNAME<
github username  -->
            </configuration>
            <executions>
              <!-- run site-maven-plugin's 'site' targ
normal 'deploy' phase -->
              <execution>
                <goals>
                  <goal>site</goal>
                </goals>
                <phase>deploy</phase>
              </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

The `mvn-repo` branch does not need to exist, it will be created for you.

Now run `mvn clean deploy` again. You should see maven-deploy-plugin "upload" the files to your local staging repository in the target directory, then site-maven-plugin committing those files and pushing them to the server.

```
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------------------------------------
[INFO] Building DaoCore 1.3-SNAPSHOT
[INFO] ----------------------------------------------------
...
[INFO] --- maven-deploy-plugin:2.5:deploy (default-dep
Uploaded: file:///Users/mike/Projects/greendao-emmby/D
repo/com/greendao-orm/greendao/1.3-SNAPSHOT/greendao-1
(77 KB at 2936.9 KB/sec)
Uploaded: file:///Users/mike/Projects/greendao-emmby/D
repo/com/greendao-orm/greendao/1.3-SNAPSHOT/greendao-1
(3 KB at 1402.3 KB/sec)
Uploaded: file:///Users/mike/Projects/greendao-emmby/D
repo/com/greendao-orm/greendao/1.3-SNAPSHOT/maven-meta
KB/sec)
Uploaded: file:///Users/mike/Projects/greendao-emmby/D
repo/com/greendao-orm/greendao/maven-metadata.xml (282
[INFO]
[INFO] --- site-maven-plugin:0.7:site (default) @ gree
[INFO] Creating 24 blobs
[INFO] Creating tree with 25 blob entries
[INFO] Creating commit with SHA-1: 0b8444e487a8acf9caa
[INFO] Updating reference refs/heads/mvn-repo from
ab7afb9a228bf33d9e04db39d178f96a7a225593 to
0b8444e487a8acf9caabe7ec18a4e9cff4964809
[INFO] ----------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ----------------------------------------------------
[INFO] Total time: 8.595s
[INFO] Finished at: Sun Dec 23 11:23:03 MST 2012
[INFO] Final Memory: 9M/81M
[INFO] ----------------------------------------------------
```

Visit github.com in your browser, select the `mvn-repo` branch, and verify that all your binaries are now there.



Maven artifacts for 1.3-SNAPSHOT

emmby authored 16 minutes ago

..

| | |
|---|---|
| greendao-1.3-20121223.181214-1.jar | 26 minutes ago |
| greendao-1.3-20121223.181214-1.jar.md5 | 26 minutes ago |
| greendao-1.3-20121223.181214-1.jar.sha1 | 26 minutes ago |
| greendao-1.3-20121223.181214-1.pom | 26 minutes ago |
| greendao-1.3-20121223.181214-1.pom.md5 | 26 minutes ago |
| greendao-1.3-20121223.181214-1.pom.sha1 | 26 minutes ago |

**Congratulations!**

You can now deploy your maven artifacts to a poor man's public repo simply by running `mvn clean deploy`.

There's one more step you'll want to take, which is to configure any poms that depend on your pom to know where your repository is. Add the following snippet to any project's pom that depends on your project:

```
<repositories>
    <repository>
        <id>YOUR-PROJECT-NAME-mvn-repo</id>
        <url>https://github.com/YOUR-USERNAME/YOUR-PRO
repo/</url>
        <snapshots>
            <enabled>true</enabled>
            <updatePolicy>always</updatePolicy>
        </snapshots>
    </repository>
</repositories>
```

Now any project that requires your jar files will automatically download them from your github maven repository.

Edit: to avoid the problem mentioned in the comments ('Error creating commit: Invalid request. For 'properties/name', nil is not a string.'), make sure you state a name in your profile on github.

Share Improve this answer

Follow

edited Mar 4, 2020 at 21:20

Ginkobonsai
187 ● 3 ● 13

answered Dec 23, 2012 at 18:43

emmby
100k ● 66 ● 194 ● 251

28 Note also that this solution will overwrite your previous artifacts every time you deploy. This is appropriate for snapshot repositories, but not for released artifacts. To disable that behavior, set `<merge>true</merge>` in your site-maven-plugin configuration. If you do that, though, I think you'll have to manually create the mvn-repo branch in github and delete all its files the first time around. – emmby Dec 23, 2012 at 21:47

13 +1 clever and well presented. My only criticism is that you did not include a link to the Maven plugins site: github.com/github/maven-plugins. Thx I was looking for a way to publish my Maven site to github! – Mark O'Connor Dec 23, 2012 at 23:46

8 This approach does not work when Two-Factor authentication is used on github. See my note in the issue here: github.com/github/maven-plugins/issues/36#issuecomment-31005606 – Dag Dec 20, 2013 at 12:09

20 In order to make this work for **multi-module projects**, you can also simply use `<altDeploymentRepository>internal.repo::default::file://${user.dir}/target/mvn-repo</altDeploymentRepository>` with the *maven-deploy-plugin*, and `<outputDirectory>${user.dir}/target/mvn-repo</outputDirectory>` with *site-maven-plugin*. This will deploy all artifacts into the root ("parent") project, and push them to the respective parent direcory on github. Otherwise, the build of each sub-module will overwrite that of the sub-module built before... – s.d Feb 25, 2014 at 16:58

7 Two suggestions that make it work (at least for me): Set current version of Github plugin (right now it would be 0.11). Also I would suggest everybody to use a OAUTH token instead of the password. You can generate it in 'Settings->Applications->Personal Access Tokens'. Than you also can inline it into the POM via and store the token as environment variable. `<github.global.userName>YourUserName</github.global.userName> <github.global.password>${GITHUB_OAUTH_TOKEN</github.global.password>` – Florian Loch Feb 12, 2015 at 18:55

130

**Don't use GitHub as a Maven Repository.**

*Edit: This option gets a lot of down votes, but no comments as to why. This is the correct option regardless* *of the technical capabilities to actually host on GitHub. Hosting on GitHub is wrong for all the reasons outlined below and without comments I can't improve the answer to clarify your issues.*

**Best Option - Collaborate with the Original Project**

The best option is to convince the original project to include your changes and stick with the original.

**Alternative - Maintain your own Fork**

Since you have forked an open source library, and your fork is also open source, you can upload your fork to Maven Central (read [Guide to uploading artifacts to the Central Repository]()) by giving it a new `groupId` and maybe a new `artifactId`.

Only consider this option if you are willing to maintain this fork until the changes are incorporated into the original project and then you should abandon this one.

Really consider hard whether a fork is the right option. Read the myriad Google results for ['why not to fork']()

**Reasoning**

**Bloating your repository with jars increases download size for no benefit**

A jar is an `output` of your project, it can be regenerated at any time from its `inputs`, and your GitHub repo should contain only `inputs`.

Don't believe me? Then check Google results for 'dont store binaries in git'.

GitHub's help Working with large files will tell you the same thing. Admittedly jar's aren't large but they are larger than the source code and once a jar has been created by a release they have no reason to be versioned - that is what a new release is for.

**Defining multiple repos in your pom.xml slows your build down by Number of Repositories times Number of Artifacts**

Stephen Connolly says:

> If anyone adds your repo they impact their build performance as they now have another repo to check artifacts against... It's not a big problem if you only have to add one repo... But the problem grows and the next thing you know your maven build is checking 50 repos for every artifact and build time is a dog.

That's right! **Maven needs to check every artifact (and its dependencies) defined in your pom.xml against every Repository you have defined**, as a newer version might be available in any of those repositories.

Try it out for yourself and you will feel the pain of a slow build.

The best place for artifacts is in Maven Central, as its the central place for jars, and this means your build will only ever check **one** place.

You can read some more about repositories at Maven's documentation on Introduction to Repositories

Share Improve this answer Follow

edited Jan 28, 2018 at 1:59

answered Mar 5, 2014 at 22:30

Bae
7,624 ● 5 ● 38 ● 42

3 Totally agree, and makes sense for forks you want to keep around for awhile. But this can be a lot of overhead for just a small patch to an existing project. – emmby Mar 6, 2014 at 19:02

6 I doubt Github has a problem with it, as they wrote the plugin that enables this capability. I agree it's less than idea, but c'est la vie. – PHY6 Jun 25, 2014 at 17:05

4 It's not always possible to deploy an open source project on Sonatype. For instance when your project depends on another open source project that it isn't already deployed (and it can't be deployed because it doesn't meet the sonatype requirements). – Gab Aug 28, 2014 at 12:27

1 @Gab then your dependency is not really open source. You should contact the other project and explain this and get them to fix their licensing. (Sun was a culprit of this behaviour in the past) – Bae Oct 8, 2014 at 5:59

1 @Bae It is not a question of licensing. Some project owner decide not to publish on central simply because it's not their

priority. Your way is not possible in the real world. If you want to test: convince this to publish on Central code.google.com/p/sd-dss . It's a big Open Source project funded by EU community :) – Gab Oct 12, 2014 at 11:16

52

You can use JitPack (free for public Git repositories) to expose your GitHub repository as a Maven artifact. Its very easy. Your users would need to add this to their pom.xml:

1. Add repository:

```
<repository>
    <id>jitpack.io</id>
    <url>https://jitpack.io</url>
</repository>
```

2. Add dependency:

```
<dependency>
    <groupId>com.github.User</groupId>
    <artifactId>Repo name</artifactId>
    <version>Release tag</version>
</dependency>
```

As answered elsewhere the idea is that JitPack will build your GitHub repo and will serve the jars. The requirement is that you have a build file and a GitHub release.

The nice thing is that you don't have to handle deployment and uploads. Since you didn't want to maintain your own artifact repository its a good match for your needs.

Share Improve this answer

Follow

edited Jan 25, 2019 at 9:55

Anton Krosnev

**4,122** ● 1 ● 22 ● 38

answered May 4, 2015 at 16:39

Andrejs

**27.7k** ● 13 ● 108 ● 97

1 JitPack is pretty good, but forces you to change every groupId that you have around. They say that this can be avoided, but it requires that you add an entry to the DNS of your company, which is totally impractical in most cases. I've been trying with JP once, then I decided that this is too stupid to go ahead. – zakmck Sep 22, 2015 at 15:31

1 Changing the groupId of your projects is not necessary. You can still install those projects using 'com.github.User' groupId. But perhaps your use case is different. – Andrejs Sep 22, 2015 at 16:41

Yes, it is very much. Because I already have tens of them around my organisation and external users, and because I want my own brand on them. How one can be so fool to try to force me into his own groupId is one of the things why I'm thinking of making a career change. – zakmck Sep 22, 2015 at 16:45

Moreover, I don't see any real need for JP guys to throw such requirement at me (they could just intercept Maven requests from the repository spec). – zakmck Sep 22, 2015 at 16:51

1 Good idea, I've done it: github.com/jitpack/jitpack.io/issues/209, thanks :-) – zakmck Sep 23, 2015 at 12:44

25

Since 2019 you can now use the new functionality called Github package registry.

Basically the process is:

- generate a new personal access token from the github settings
- add repository and token info in your `settings.xml`

- deploy using

```
mvn deploy -Dregistry=https://maven.pkg.github.com
Dtoken=yor_token
```

Share Improve this answer Follow

edited Dec 7, 2019 at 5:55
HRJ
17.7k ● 11 ● 59 ● 80

answered Sep 29, 2019 at 1:46
Ruslan López
4,467 ● 2 ● 29 ● 40

1 As of 2019, this is the best option. – HRJ Dec 7, 2019 at 5:56

2 But for using it by someone else, it looks like he/she needs to configure settings.xml with respective URL and auth info – hemu May 9, 2020 at 5:45

3 Very strange... You create your public package, but another people need authentication before get it – Amerousful May 12, 2020 at 9:32

However, for private repos, after certaing usages / month, the pricing comes into picture – Lokeshwar Tailor Jun 12, 2020 at 15:37

5 `Github Package Registry` is useless for open source projects because clients cannot download artifacts without authorization. – expert Feb 3, 2021 at 19:04

9

Another alternative is to use any web hosting with webdav support. You will need some space for this somewhere of course but it is straightforward to set up and a good alternative to running a full blown nexus server.

add this to your build section

```
      <extensions>
         <extension>
         <artifactId>wagon-webdav-jackrabbit</artifactI
         <groupId>org.apache.maven.wagon</groupId>
         <version>2.2</version>
         </extension>
     </extensions>
```

Add something like this to your distributionManagement section

```
<repository>
    <id>release.repo</id>
    <url>dav:http://repo.jillesvangurp.com/releases/</
</repository>
```

Finally make sure to setup the repository access in your settings.xml

add this to your servers section

```
    <server>
        <id>release.repo</id>
        <username>xxxx</username>
        <password>xxxx</password>
    </server>
```

and a definition to your repositories section

```
            <repository>
                <id>release.repo</id>
                <url>http://repo.jillesvangurp.com/rel
                <releases>
                    <enabled>true</enabled>
                </releases>
                <snapshots>
                    <enabled>false</enabled>
                </snapshots>
            </repository>
```

Finally, if you have any standard php hosting, you can use something like sabredav to add webdav capabilities.

Advantages: you have your own maven repository
Downsides: you don't have any of the management capabilities in nexus; you need some webdav setup somewhere

Share Improve this answer Follow

answered Sep 27, 2013 at 13:49

Jilles van Gurp
**8,276** ● 4 ● 41 ● 47

---

7

As an alternative, [Bintray]() provides free hosting of maven repositories. That's probably a good alternative to [Sonatype OSS]() and Maven Central if you absolutely don't want to rename the groupId. But please, at least make an effort to get your changes integrated upstream or rename and publish to Central. It makes it much easier for others to use your fork.

Share Improve this answer

answered Aug 24, 2014 at 13:00

Follow

Guillaume
18.9k ● 8 ● 55 ● 76

3 I couldn't believe it when I tried, but Bintray doesn't support snapshots. Useless. – zakmck Sep 22, 2015 at 15:33

6 It's not free anymore. $150 a month. – Johann Oct 6, 2016 at 0:46

I think it is fee for open source software projects: jfrog.com/open-source – iBiber Sep 9, 2017 at 8:26

1 JFrog is shutting down Bintray and JCenter. jfrog.com/blog/… – Joe Bowbeer Apr 5, 2021 at 3:03

0

If you have only `aar` or `jar` file itself, or just don't want to use plugins - I've created a simple shell script. You can achieve the same with it - publishing your artifacts to Github and use it as public Maven repo.

Share Improve this answer

Follow

answered Oct 9, 2018 at 20:58

Orest Savchak
4,559 ● 1 ● 20 ● 27

0

I'd like to add another alternative, a Gradle plugin I've been working on lately: magik.

Basically it allows to publish directly on a github repository acting as a maven repository.

answered May 26, 2021 at 6:48

Share Improve this answer

Follow

elect
7,150 ● 10 ● 61 ● 126

0

I came here looking to do the same thing, unlitmately host my Maven repository for free, but after more research I ended up here: https://jfrog.com/start-free/

The setup was quite strightforward, has a good free tier which will serve me for the forseeable future, and has additional (paid for) upgrades which may well come in handy in the future.

So far I am very pleased indeed !

Share Improve this answer

Follow

answered Nov 21, 2021 at 20:13

MarkA
1,226 ● 1 ● 12 ● 21

**Highly active question**. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.