Working with the Apache Maven registry

In this article

You can configure Apache Maven to publish packages to GitHub Packages and to use packages stored on GitHub Packages as dependencies in a Java project.

Note: This package type may not be available for your instance, because site administrators can enable or disable each supported package type. For more information, see "Configuring package ecosystem support for your enterprise."

Authenticating to GitHub Packages

GitHub Packages only supports authentication using a personal access token (classic). For more information, see "Managing your personal access tokens."

You need an access token to publish, install, and delete private, internal, and public packages.

You can use a personal access token (classic) to authenticate to GitHub Packages or the GitHub Enterprise Server API. When you create a personal access token (classic), you can assign the token different scopes depending on your needs. For more information about packages-related scopes for a personal access token (classic), see "About permissions for GitHub Packages."

To authenticate to a GitHub Packages registry within a GitHub Actions workflow, you can use:

- `GITHUB_TOKEN` to publish packages associated with the workflow repository.
- a personal access token (classic) with at least `read:packages` scope to install packages associated with other private repositories (which `GITHUB_TOKEN` can't access).

For more information about `GITHUB_TOKEN` used in GitHub Actions workflows, see "Automatic token authentication."

Authenticating with a personal access token

You must use a personal access token (classic) with the appropriate scopes to publish and install packages in GitHub Packages. For more information, see "Introduction to GitHub Packages."

You can authenticate to GitHub Packages with Apache Maven by editing your *~/.m2/settings.xml* file to include your personal access token (classic). Create a new *~/.m2/settings.xml* file if one doesn't exist.

In the `servers` tag, add a child `server` tag with an `id`, replacing USERNAME with your GitHub username, and TOKEN with your personal access token.

In the `repositories` tag, configure a repository by mapping the `id` of the repository to the `id` you added in the `server` tag containing your credentials. Replace HOSTNAME with the host name of your GitHub Enterprise Server instance, and OWNER with the name of the personal account or organization that owns the repository. Because uppercase letters aren't supported, you must use lowercase letters for the repository owner even if the GitHub user or organization name contains uppercase letters.

If you want to interact with multiple repositories, you can add each repository to separate `repository` children in the `repositories` tag, mapping the `id` of each to the credentials in the `servers` tag.

GitHub Packages supports `SNAPSHOT` versions of Apache Maven. To use the GitHub Packages repository for downloading `SNAPSHOT` artifacts, enable SNAPSHOTS in the POM of the consuming project or your *~/.m2/settings.xml* file.

If your instance has subdomain isolation enabled:

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
                      http://maven.apache.org/xsd/settings-1.0.0.xsd">

  <activeProfiles>
    <activeProfile>github</activeProfile>
  </activeProfiles>

  <profiles>
    <profile>
      <id>github</id>
      <repositories>
        <repository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
        </repository>
        <repository>
          <id>github</id>
          <url>https://maven.HOSTNAME/OWNER/REPOSITORY</url>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </repository>
      </repositories>
    </profile>
  </profiles>

  <servers>
    <server>
      <id>github</id>
      <username>USERNAME</username>
      <password>TOKEN</password>
    </server>
  </servers>
</settings>
```

If your instance has subdomain isolation disabled:

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
```

```
                      http://maven.apache.org/xsd/settings-1.0.0.xsd">

  <activeProfiles>
    <activeProfile>github</activeProfile>
  </activeProfiles>

  <profiles>
    <profile>
      <id>github</id>
      <repositories>
        <repository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
        </repository>
        <repository>
          <id>github</id>
          <url>HOSTNAME/_registry/maven/OWNER/REPOSITORY</url>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </repository>
      </repositories>
    </profile>
  </profiles>

  <servers>
    <server>
      <id>github</id>
      <username>USERNAME</username>
      <password>TOKEN</password>
    </server>
  </servers>
</settings>
```

Publishing a package

By default, GitHub publishes the package to an existing repository with the same name as the package. For example, GitHub will publish a package named `com.example:test` in a repository called `OWNER/test`.

If you would like to publish multiple packages to the same repository, you can include the URL of the repository in the `<distributionManagement>` element of the *pom.xml* file. GitHub will match the repository based on that field. Since the repository name is also part of the `distributionManagement` element, there are no additional steps to publish multiple packages to the same repository.

For more information on creating a package, see the maven.apache.org documentation.

1. Edit the `distributionManagement` element of the *pom.xml* file located in your package directory, replacing HOSTNAME with the host name of your GitHub Enterprise Server instance, `OWNER` with the name of the personal account or organization that owns the repository and `REPOSITORY` with the name of the repository containing your project.

   If your instance has subdomain isolation enabled:

   ```
   <distributionManagement>
      <repository>
        <id>github</id>
        <name>GitHub OWNER Apache Maven Packages</name>
        <url>https://maven.HOSTNAME/OWNER/REPOSITORY</url>
      </repository>
   </distributionManagement>
   ```

   If your instance has subdomain isolation disabled:

```
<distributionManagement>
   <repository>
     <id>github</id>
     <name>GitHub OWNER Apache Maven Packages</name>
     <url>https://HOSTNAME/_registry/maven/OWNER/REPOSITORY</url>
   </repository>
</distributionManagement>
```

2. Publish the package.

```
mvn deploy
```

After you publish a package, you can view the package on GitHub. For more information, see "Viewing packages."

Installing a package

To install an Apache Maven package from GitHub Packages, edit the *pom.xml* file to include the package as a dependency. If you want to install packages from any repository for a specified repository owner, use a repository URL like `https://maven.HOSTNAME/OWNER/*`. For more information on using a *pom.xml* file in your project, see "Introduction to the POM" in the Apache Maven documentation.

1. Authenticate to GitHub Packages. For more information, see "Authenticating to GitHub Packages."
2. Add the package dependencies to the `dependencies` element of your project *pom.xml* file, replacing `com.example:test` with your package.

```
<dependencies>
 <dependency>
    <groupId>com.example</groupId>
    <artifactId>test</artifactId>
    <version>1.0.0-SNAPSHOT</version>
  </dependency>
</dependencies>
```

3. Install the package.

```
mvn install
```

Further reading

- "Working with the Gradle registry"
- "Deleting and restoring a package"