Working with the NuGet registry

In this article

You can configure the `dotnet` command-line interface (CLI) to publish NuGet packages to GitHub Packages and to use packages stored on GitHub Packages as dependencies in a .NET 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).

Authenticating in a GitHub Actions workflow

Use the following command to authenticate to GitHub Packages in a GitHub Actions workflow using the `GITHUB_TOKEN` instead of hardcoding a personal access token in a nuget.config file in the repository:

```
dotnet nuget add source --username USERNAME --password ${{ secrets.GITHUB_TOKEN
}} --store-password-in-clear-text --name github
"https://nuget.HOSTNAME/NAMESPACE/index.json"
```

Replace `NAMESPACE` with the name of the personal account or organization that owns the repository where your packages are hosted.

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

Authenticating with a personal access token

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).

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."

To authenticate to GitHub Packages with the `dotnet` command-line interface (CLI), create a *nuget.config* file in your project directory specifying GitHub Packages as a source under `packageSources` for the `dotnet` CLI client.

You must replace:

- `USERNAME` with the name of your personal account on GitHub.
- `TOKEN` with your personal access token (classic).
- `NAMESPACE` with the name of the personal account or organization that owns the repository where your packages are hosted.
- `HOSTNAME` with the host name for your GitHub Enterprise Server instance.

If your instance has subdomain isolation enabled:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <packageSources>
        <clear />
        <add key="github" value="https://nuget.HOSTNAME/NAMESPACE/index.json" />
    </packageSources>
    <packageSourceCredentials>
        <github>
            <add key="Username" value="USERNAME" />
            <add key="ClearTextPassword" value="TOKEN" />
        </github>
```

```
    </packageSourceCredentials>
</configuration>
```

If your instance has subdomain isolation disabled:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <packageSources>
        <clear />
        <add key="github"
value="https://HOSTNAME/_registry/nuget/NAMESPACE/index.json" />
    </packageSources>
    <packageSourceCredentials>
        <github>
            <add key="Username" value="USERNAME" />
            <add key="ClearTextPassword" value="TOKEN" />
        </github>
    </packageSourceCredentials>
</configuration>
```

Publishing a package

You can publish a package to GitHub Packages by authenticating with a *nuget.config* file, or by using the `--api-key` command line option with your GitHub personal access token (classic).

Publishing a package using a GitHub personal access token as your API key

If you don't already have a personal access token to use for your account on your GitHub Enterprise Server instance, see "Managing your personal access tokens."

1. Create a new project. Replace `PROJECT_NAME` with the name you'd like to give the project.

   ```
   dotnet new console --name PROJECT_NAME
   ```

2. Package the project.

   ```
   dotnet pack --configuration Release
   ```

3. Publish the package using your personal access token as the API key. Replace `PROJECT_NAME` with the name of the project, `1.0.0` with the version number of the package, and `YOUR_GITHUB_PAT` with your personal access token.

   ```
   dotnet nuget push "bin/Release/PROJECT_NAME.1.0.0.nupkg"  --api-key
   YOUR_GITHUB_PAT --source "github"
   ```

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

Publishing a package using a *nuget.config* file

When publishing, the `OWNER` of the repository specified in your *.csproj* file must match the `NAMESPACE` that you use in your *nuget.config* authentication file. Specify or increment

the version number in your *.csproj* file, then use the `dotnet pack` command to create a *.nuspec* file for that version. For more information on creating your package, see "Create and publish a package" in the Microsoft documentation.

1. Authenticate to GitHub Packages. For more information, see "Authenticating to GitHub Packages."

2. Create a new project. Replace `PROJECT_NAME` with the name you'd like to give the project.

   ```
   dotnet new console --name PROJECT_NAME
   ```

3. Add your project's specific information to your project's file, which ends in *.csproj*. Make sure to replace:

   - `1.0.0` with the version number of the package.
   - `OWNER` with the name of the personal account or organization that owns the repository to which you want to publish your package.
   - `REPOSITORY` with the name of the repository to which you want to connect your package.
   - `HOSTNAME` with the host name for your GitHub Enterprise Server instance.

   ```
   <Project Sdk="Microsoft.NET.Sdk">

     <PropertyGroup>
       <OutputType>Exe</OutputType>
       <TargetFramework>netcoreapp3.0</TargetFramework>
       <PackageId>PROJECT_NAME</PackageId>
       <Version>1.0.0</Version>
       <Authors>AUTHORS</Authors>
       <Company>COMPANY_NAME</Company>
       <PackageDescription>PACKAGE_DESCRIPTION</PackageDescription>
       <RepositoryUrl>https://HOSTNAME/OWNER/REPOSITORY</RepositoryUrl>
     </PropertyGroup>

   </Project>
   ```

4. Package the project.

   ```
   dotnet pack --configuration Release
   ```

5. Publish the package using the `key` you specified in the *nuget.config* file. Replace `PROJECT_NAME` with the name of the project, and replace `1.0.0` with the version number of the package.

   ```
   dotnet nuget push "bin/Release/PROJECT_NAME.1.0.0.nupkg" --source "github"
   ```

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

Publishing multiple packages to the same repository

To connect multiple packages to the same repository, use the same GitHub repository URL in the `RepositoryURL` fields in all *.csproj* project files. GitHub matches the repository

based on that field.

The following example publishes the projects MY_APP and MY_OTHER_APP to the same repository:

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp3.0</TargetFramework>
    <PackageId>MY_APP</PackageId>
    <Version>1.0.0</Version>
    <Authors>Octocat</Authors>
    <Company>GitHub</Company>
    <PackageDescription>This package adds a singing Octocat!</PackageDescription>
    <RepositoryUrl>https://HOSTNAME/my-org/my-repo</RepositoryUrl>
  </PropertyGroup>

</Project>
```

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp3.0</TargetFramework>
    <PackageId>MY_OTHER_APP</PackageId>
    <Version>1.0.0</Version>
    <Authors>Octocat</Authors>
    <Company>GitHub</Company>
    <PackageDescription>This package adds a dancing Octocat!</PackageDescription>
    <RepositoryUrl>https://HOSTNAME/my-org/my-repo</RepositoryUrl>
  </PropertyGroup>

</Project>
```

Installing a package

Using packages from GitHub in your project is similar to using packages from *nuget.org*. Add your package dependencies to your *.csproj* file, specifying the package name and version. For more information on using a *.csproj* file in your project, see "Working with NuGet packages" in the Microsoft documentation.

1. Authenticate to GitHub Packages. For more information, see "Authenticating to GitHub Packages."
2. To use a package, add `ItemGroup` and configure the `PackageReference` field in the *.csproj* project file. Replace the `PACKAGE_NAME` value in `Include="PACKAGE_NAME"` with your package dependency, and replace the `X.X.X` value in `Version="X.X.X"` with the version of the package you want to use:

   ```
   <Project Sdk="Microsoft.NET.Sdk">

     <PropertyGroup>
       <OutputType>Exe</OutputType>
       <TargetFramework>netcoreapp3.0</TargetFramework>
       <PackageId>My-app</PackageId>
       <Version>1.0.0</Version>
      <Authors>Octocat</Authors>
       <Company>GitHub</Company>
      <PackageDescription>This package adds an Octocat!</PackageDescription>
       <RepositoryUrl>https://HOSTNAME/OWNER/REPOSITORY</RepositoryUrl>
     </PropertyGroup>
   ```

```
  <ItemGroup>
    <PackageReference Include="PACKAGE_NAME" Version="X.X.X" />
  </ItemGroup>

</Project>
```

3. Install the packages with the `restore` command.

```
dotnet restore
```

Troubleshooting

Your NuGet package may fail to push if the `RepositoryUrl` in *.csproj* is not set to the expected repository.

If you're using a nuspec file, ensure that it has a `repository` element with the required `type` and `url` attributes.

If you're using a `GITHUB_TOKEN` to authenticate to a GitHub Packages registry within a GitHub Actions workflow, the token cannot access private repository-based packages in a different repository other than where the workflow is running in. To access packages associated with other repositories, instead generate a personal access token (classic) with the `read:packages` scope and pass this token in as a secret.

Further reading

- "Deleting and restoring a package"