Create your own Chocolatey Caching Host for $7 a month

So, you want to operate a Chocolatey proxy and caching server. Well, you're in luck as we happen to know of a great product that is free to use for this type of need. ProGet installs in minutes and has a powerful free version with a *lot of great features* that you can upgrade when ready.

ProGet is a Self-hosted, Cross-Platform Package & Container Repository that keeps all your packages in one place, allowing you to scan for vulnerabilities and control who can access different feeds. It also will provide a proxy and caching agent for the public Chocolatey repositories.

Avoid Rate Limiting from Chocolatey.org

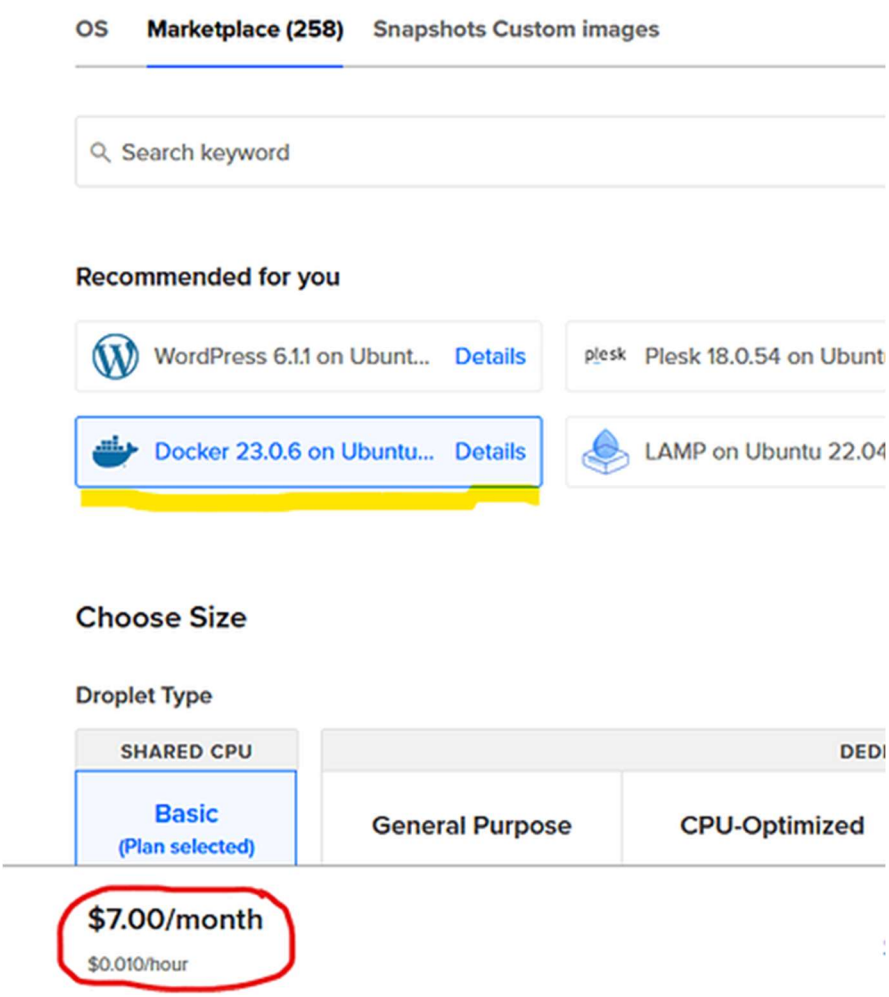
Chocolatey.org has strict rate-limiting policies, which means you may start receiving that dreaded "429 error" in the middle of deploying or upgrade packages.

You can prevent these errors by caching Chocolatey packages in ProGet so that your package deployment will not be interrupted.

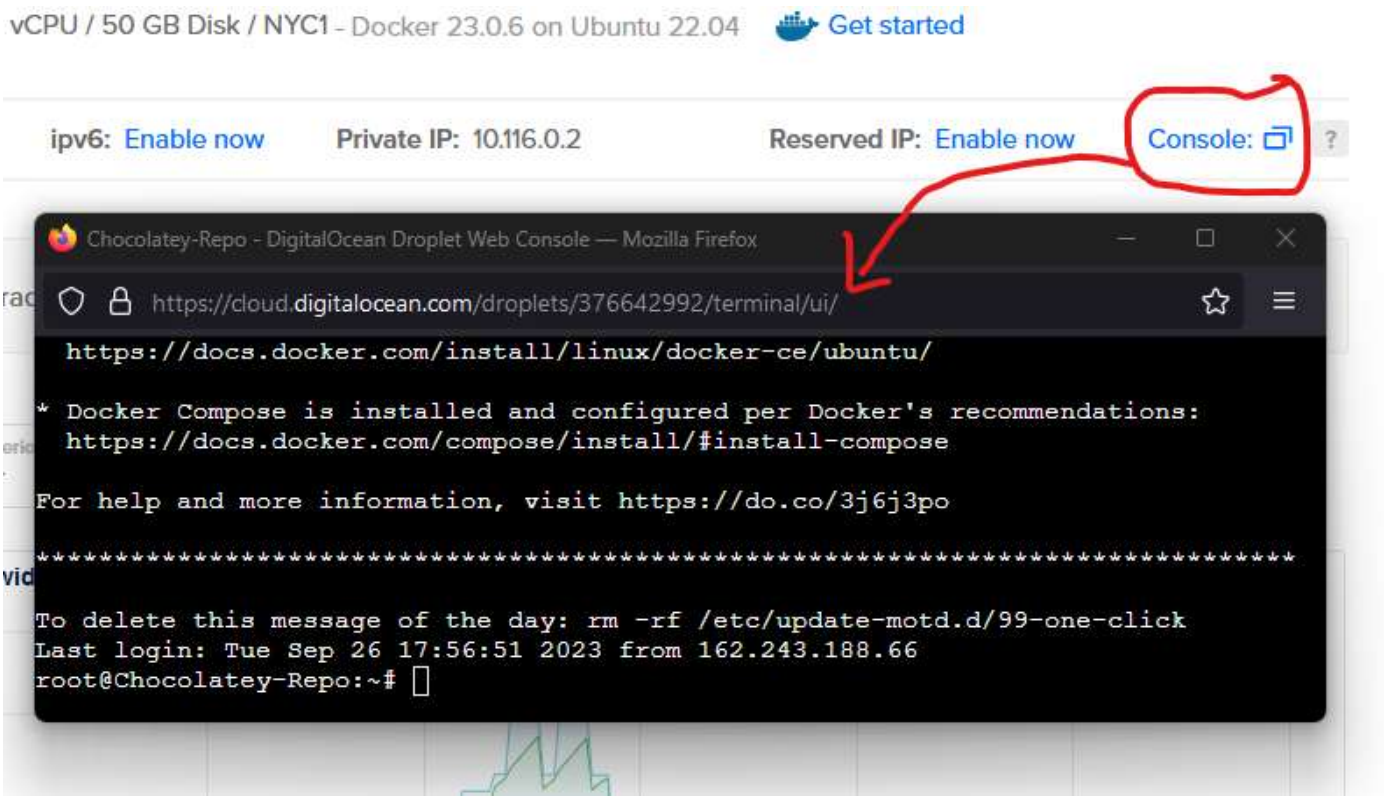
Its takes about 10 minutes of your time to get a fully deployed system going and ready to cache Chocolatey Packages.

The first thing you want to do is get a Linux box with Docker deployed in the cloud, you can do it locally if you have the space and bandwidth. I personally like Digital Ocean, its super quick and easy to get "droplets" up and running and its cheap.

In Digital Ocean create a new droplet, give it the bare minimums to start, you can always resize it as needed. Select the marketplace and then the Docker-Ubuntu image and then select the basic shared CPU and the minimal CPU size which at current prices is running 1 cent per hour or about $7 a month.



Now once you have it running, use the console feature to access shell of the Ubuntu system as the root user.



Create a Network

First, you'll need to create a network for the SQL Server and Inedo Product containers to communicate.

```
docker network create inedo
```

Create the Database

Inedo products requires an SQL Server database. You can either host this database externally or simply use an SQL Server Docker image; it doesn't matter how it's hosted, as long as your instance can access it. Make sure to change the password in the following commands («YourStrong!Passw0rd»') to something else.

To start an SQL Server container on the `inedo` network you created, use this command:

```
docker run --name inedo-sql \
  -e 'ACCEPT_EULA=Y' -e 'MSSQL_SA_PASSWORD=*«YourStrong!Passw0rd»'* \
  -e 'MSSQL_PID=Express' --net=inedo --restart=unless-stopped \
  -d mcr.microsoft.com/mssql/server:2019-latest
```

Once you have an SQL Server instance running, you'll need to create an empty database.

Example: ProGet SQL Server Database

To create a database called `ProGet` on the SQL Server instance running in the **inedo-sql** container:

```
docker exec -it inedo-sql /opt/mssql-tools/bin/sqlcmd \
  -S localhost -U SA -P '*«YourStrong!Passw0rd»'* \
  -Q 'CREATE DATABASE [ProGet] COLLATE SQL_Latin1_General_CP1_CI_AS'
```

Start: ProGet Server

```
docker run -d --name=proget --restart=unless-stopped \
  -v proget-packages:/var/proget/packages -p 80:80 --net=inedo \
  -e PROGET_SQL_CONNECTION_STRING='Data Source=inedo-sql; Initial
Catalog=ProGet; User ID=sa; Password=*«YourStrong!Passw0rd»'* \
  proget.inedo.com/productimages/inedo/proget:latest
```

Afterwards you should have a generic ProGet web proxy running and it will need configuration to secure it and turn it into a proxy and cache server.

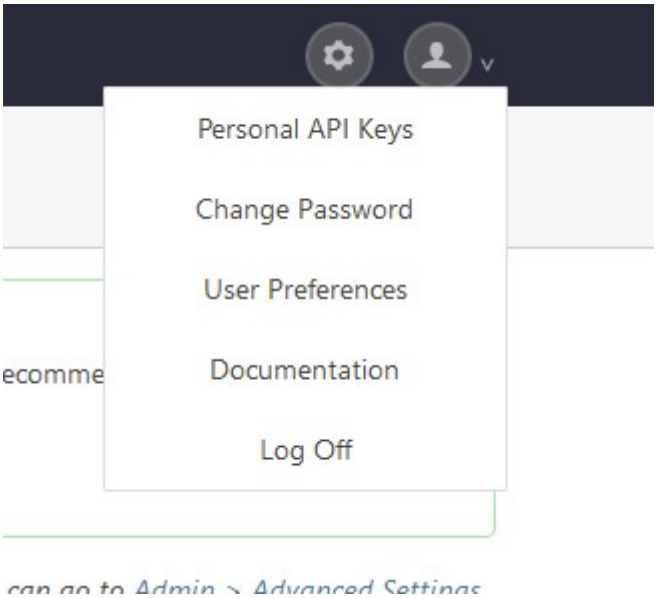
Open a web browser and go to the external IP address of the new droplet.(http://123.123.123.123)

Acquire a License Key

You'll need to have a valid license key once you get your Inedo product running. You can request a license key within the software itself, or use a license key that you already have, or request one from my.inedo.com.

The standard free version requires a KEY, it will acquire the key directly when you register it.

There will be some online instructions that you can follow but the big take away is first thing to do is secure the web portal. By default, it has permissions set to allow anonymous users the ability to manage the web, add users and all that good stuff. It allows you to quickly get into system and start pushing buttons but it's really poor security wise so we need to fix that first.

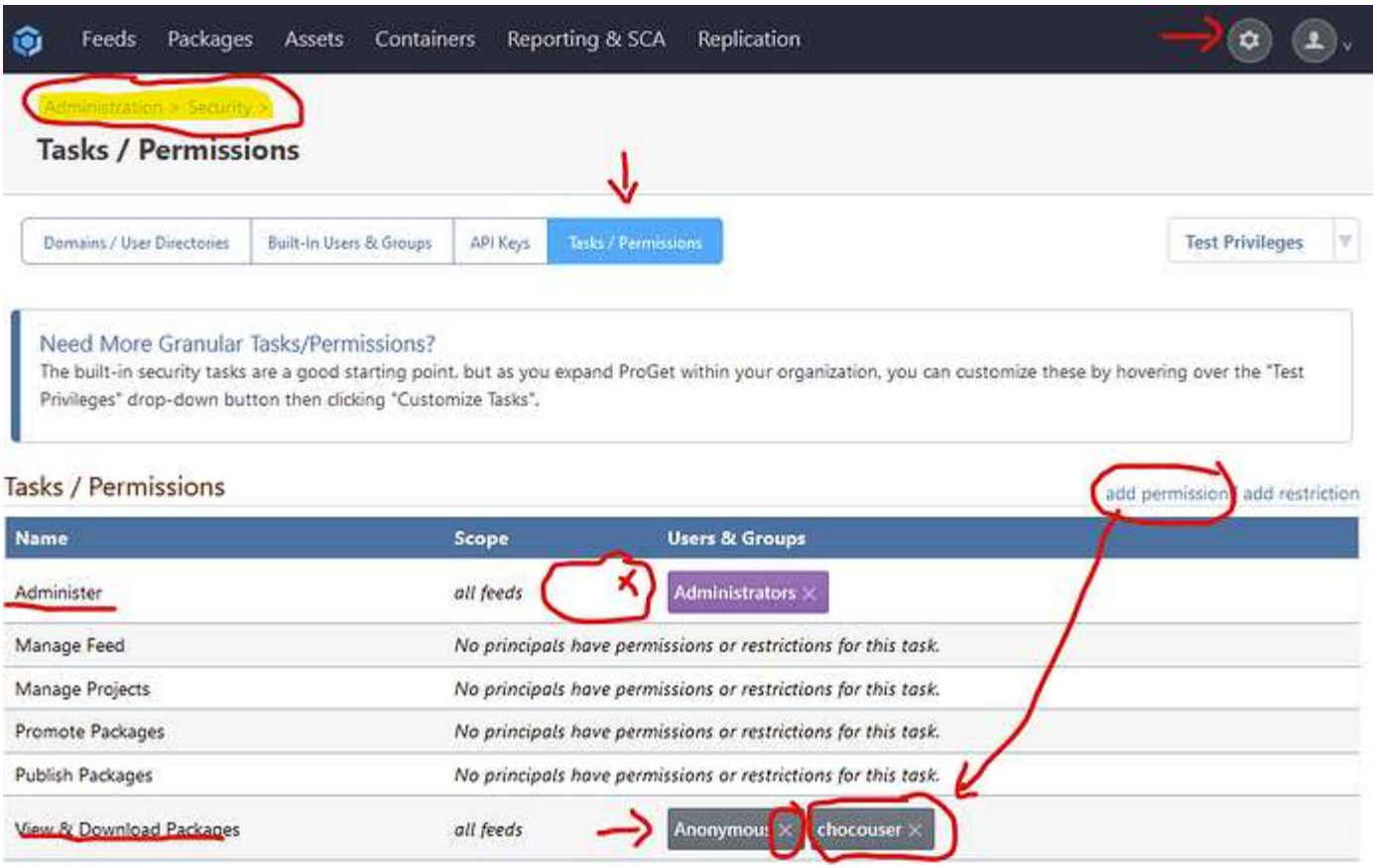


At the top find the account menu drop down, select it and select to change password.

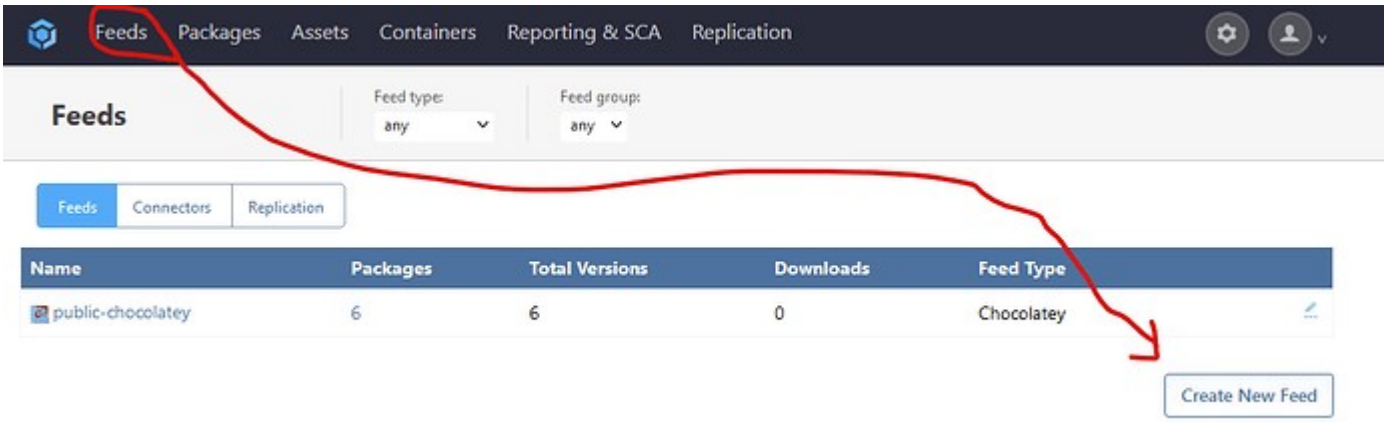
The default password is **Admin** and is case sensitive.

Next remove the Anonymous User access to admin side, you can choose to run a public proxy (no user required) or add one or more users to the web site and then remove Anonymous from public access (read only)

Select the (Gear) then select **Manage Security** menu item. Once that opens select the tasks and permissions tab to get to the permissions rules.

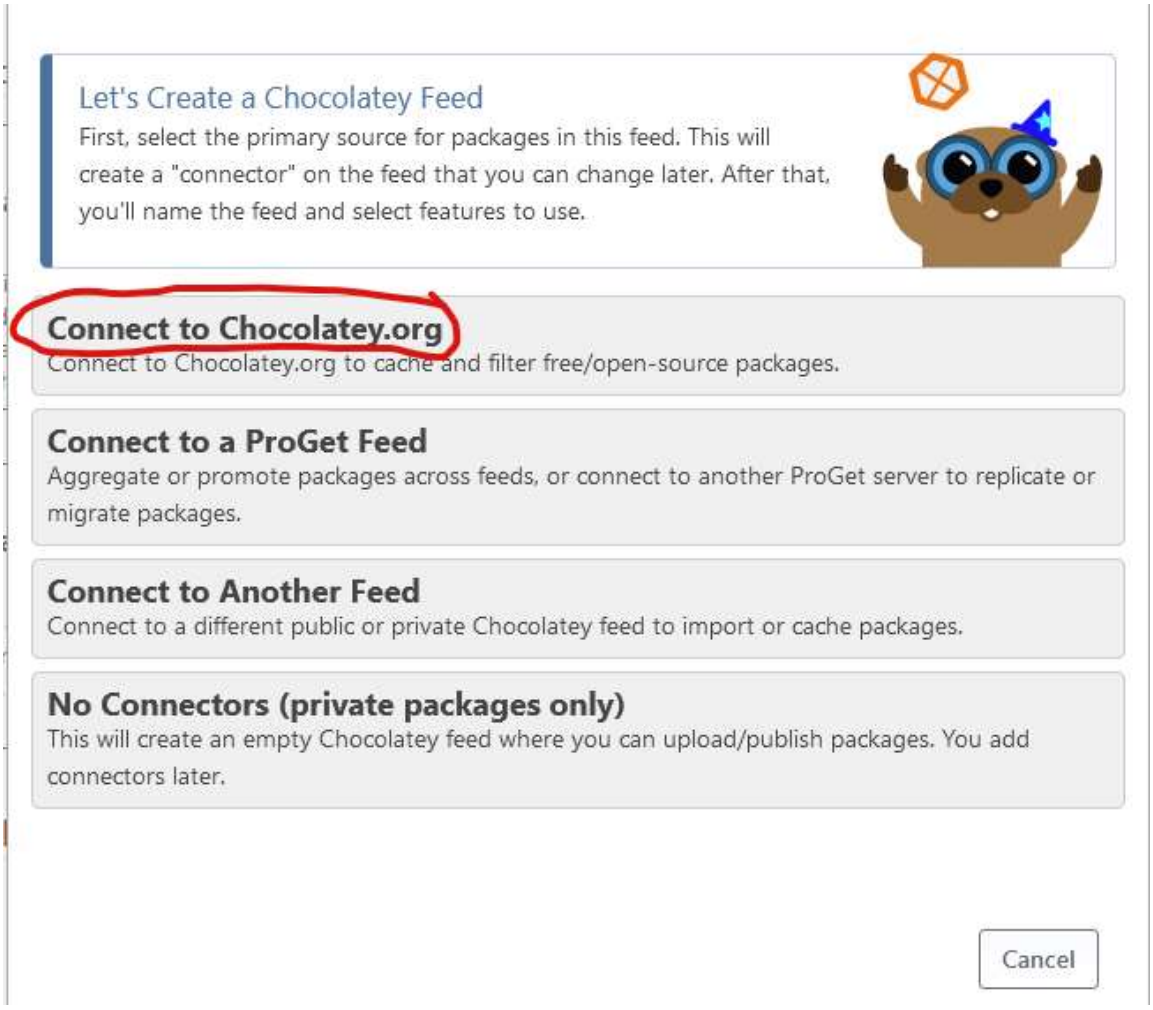


Lastly, we need to create a feed. This feed is simple to create, we select from the feed menu to create a new feed.

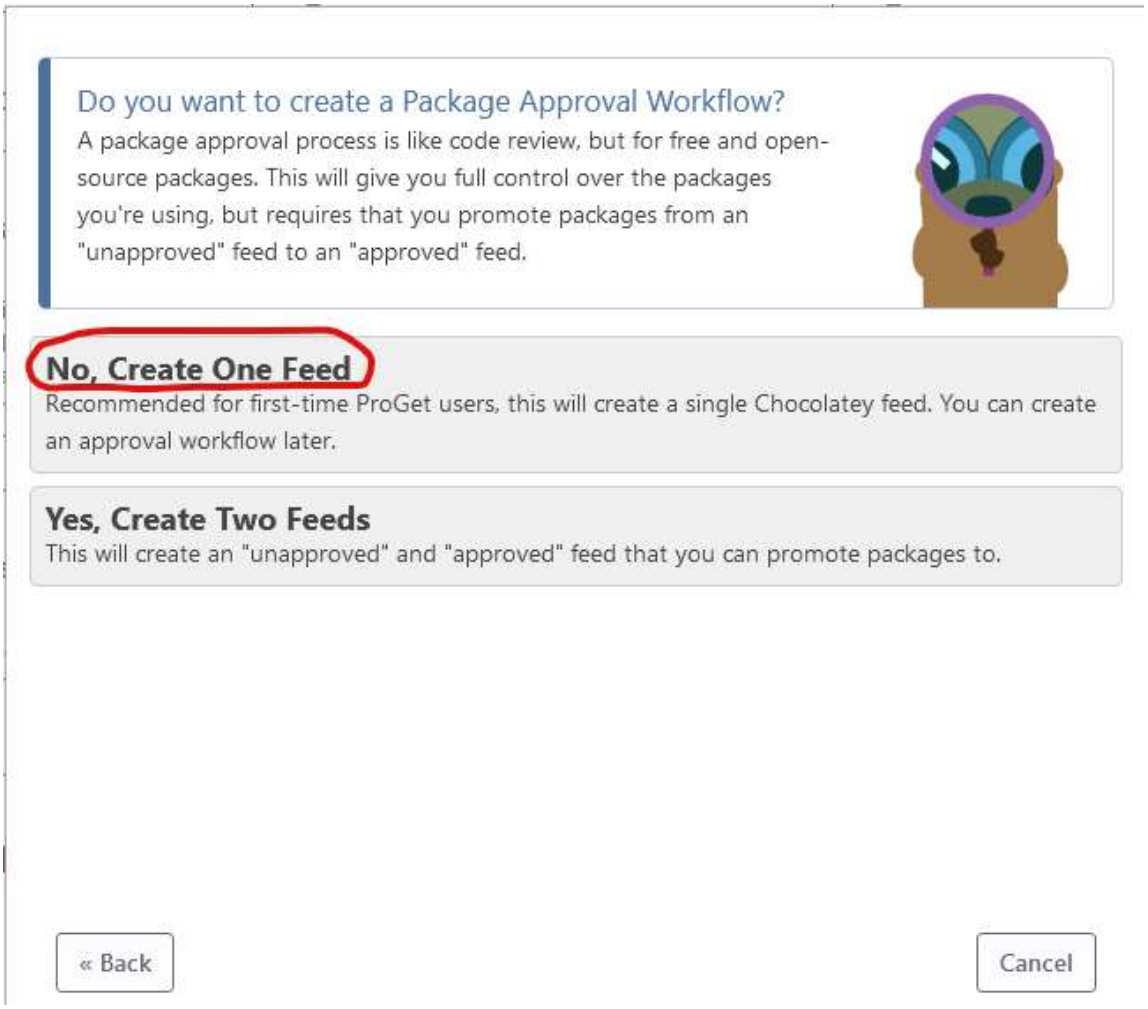


It will be populated with several feed types, select the Chocolatey Feed type.

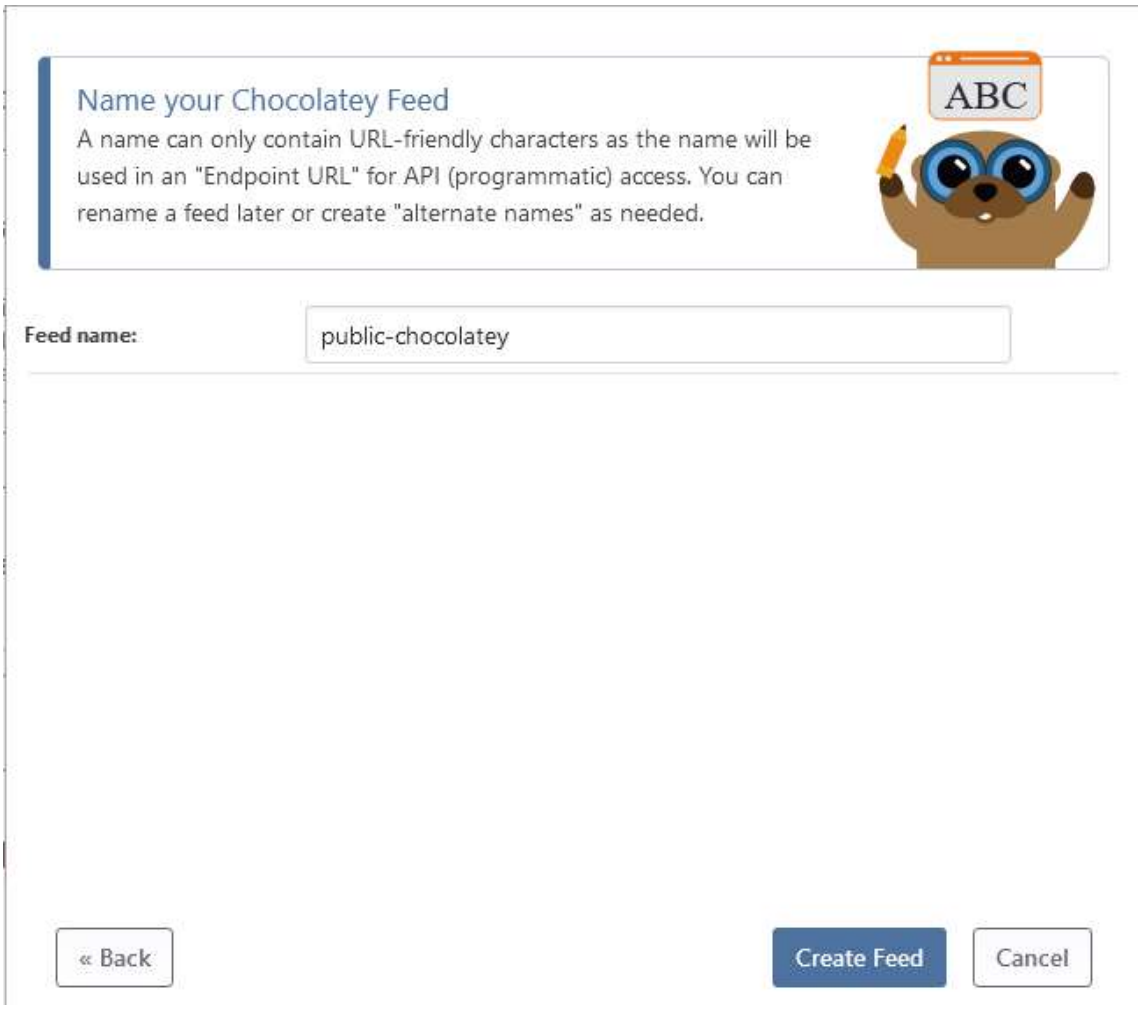
In the prompt select **Connect to Chocolatey.org**



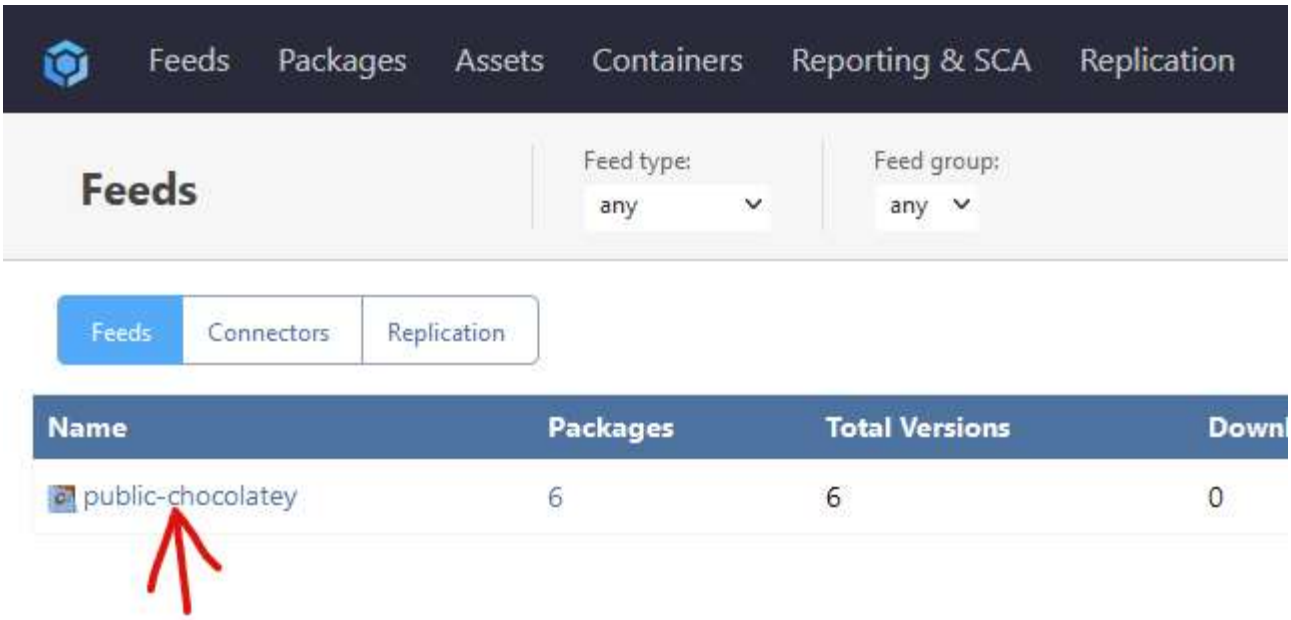
At the next prompt select “No- just one feed” this is for ease of use as it will only hold what you requested to install as cache.



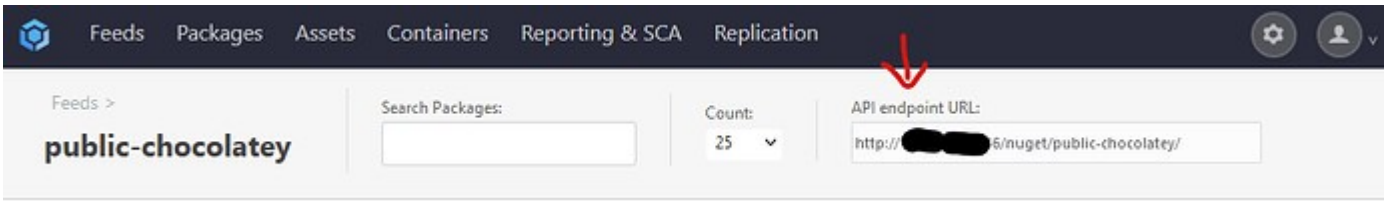
Lastly give your feed a name.



To find your feed URL (address) go to feeds view and click on your new feed.



Your feed URL will be displayed at the top right side of the feed view.



This is the URL you will use as a source for all Chocolatey packages. When a request comes into this feed the system will look in cache and see if the package is there. If not, it will proxy the request on to Chocolatey to capture package manifests and such. You can look at feed to determine how much it's being used and how many packages it has cached, all from inside the web console of ProGet.

Read more on how to create and host your own Line of Business apps.