
packmaker Documentation

Mark Crewson

May 28, 2022

Contents

1	Contents	3
1.1	Guides	3
1.2	Reference	6
1.3	FAQ	27

Packmaker is a command line tool used to build Modded Minecraft modpacks on Linux systems. It can take a yaml-based description of a pack and generate a package suitable for use in the following formats:

- **Curseforge** - A curseforge package suitable for uploading directly to the curseforge.com website, or importing into a curseforge compatible minecraft launcher like PolyMC or the Curseforge Client.
- **Routhio** - A package format used by the minecraft.routh.io community and the routh.io minecraft launcher.
- **Local client** - A local minecraft client installation, suitable for standalone games. Packmaker can even launch your modded minecraft installation in offline mode itself. Great for testing your modpack during development.
- **Server** - A minecraft server installation, suitable for use by clients to connect with and play together.

If you're new to packmaker, begin with the [Getting Started](#) guide. That guide walks you through installing packmaker, setting it up how you like it, and starting to build your first modpack.

Then you can get a more detailed look at packmaker's features in the [Command-Line Interface](#) and [Configuration](#) references.

If you still need help, you can drop by the #packmaker channel on our [Discord server](#), or [file a bug](#) in the issue tracker. Please let us know where you think this documentation can be improved.

And if you're really ambitious, [checkout the code](#) and help us improve packmaker for you and everyone.

1.1 Guides

This section contains a couple of walkthroughs that will help you get familiar with packmaker. If you're new to packmaker, you'll want to begin with the *Getting Started* guide.

1.1.1 Getting Started

Welcome to **packmaker**! This guide will help you begin using it to make Modded Minecraft modpacks.

Installing

You will need Python. Packmaker works on **Python 3.4** and later.

If you have **pip**, just say `pip3 install packmaker` (or `pip3 install --user packmaker` if you run into permissions problems).

To install without pip, download packmaker from [its PyPI page](#) and run `python setup.py install` in the directory therein.

The best way to upgrade packmaker to a new version is by running `pip3 install -U packmaker`.

Configuring

Packmaker has a config file that you will need to provide. Packmaker will look in the following locations on the filesystem for this config file, in this order. The first file found is the one it uses:

- `./packmaker.conf`, in the current working directory.
- `${HOME}/packmaker.conf`, in your home directory.
- `${HOME}/.config/packmaker.conf`, the `.config` subdirectory of your home directory.

- /usr/local/etc/packmaker.conf
- /etc/packmaker.conf

Currently, this configuration file only contains some required parameters for dealing with the Curseforge API and some optional locations for where to store downloaded and built artifacts.

```
[curseforge]
authentication_token = <curseforge authentication token>
moddb_filename = curseforge.db

[locations]
build = ./build
cache = ./build/cache
release = ./build/release
```

Packmaker needs to call the Curseforge API to retrieve mod information, and this api requires authentication. A [curseforge.com](https://www.curseforge.com) account is required and your authentication token can be generated using the [Curseforge Console](#). Copy the generated token into your config file.

Packmaker can download the entire list of minecraft mods and generate a local database of information to be used later when building modpacks. The file this database is contained in also needs to be specified.

The `[locations]` section is optional but allows you to specify the locations where the files packmaker downloads and the artifacts it generates will be kept on your filesystem.

The `build` location is where packmaker puts its generated artifacts when building a pack. If not defined, the default build location is a `build` subdirectory of the current working directory.

The `cache` location is where packmaker places the files its downloads during the build process. Minecraft files, forge libraries, mods, etc are all downloaded and kept in this cache location. Packmaker tries to optimize its downloading and checks for files in this cache location before attempting to download a file so that it does not have to download the same file twice. If not defined, the default cache location is a `cache` subdirectory in the build location.

Finally, the `release` location is where packmaker will put the final artifacts generated by the build process. For example, the zip file you will want to upload to curseforge is placed here. If not defined, the default release location is a `release` subdirectory in the build location.

CurseforgeDB

With a configuration file in place, you can now generate a local database of information on minecraft mods. Run the following command to generate it:

```
packmaker updatedb
```

This command will use the curseforge api to retrieve the list of all available mods. As there are literally thousands of available mods, this command may take a few minutes to complete.

The Example Modpack

Packmaker modpacks are defined in a YAML file. By default, packmaker will look `packmaker.yml` in the current directory, although you can use any file name you like when you specify it on the packmaker command line.

Here is a short, quick example packmaker definition file:

```

---
name:  examplepack
title: Example Pack
version: 1.0.0

authors:
  - mcrewson

minecraft: 1.12.2
forge: 14.23.5.2838

mods:
  - jei:
  - journeymap:
  - tinkers-construct:
  - the-one-probe:

files:
  - location: src/

```

Fields should be fairly obvious. The [Packmaker Definition Files](#) explains all of these options (and more) in details. But in summary, this example modpack is defined as follows:

- The modpack filename(s) will be a combination of the *name* and *version* field. For example, *examplepack-1.0.0.zip*
- The modpack metadata will include *title* and *authors* fields.
- This pack is for version 1.12.2 of Minecraft and version 14.23.5.2838 of Forge
- Packmaker will include the following mods in the pack:
 - Jei, Just Enough Items
 - Journeymap
 - Mantle (a dependent library mod)
 - Tinker’s Construct
 - The One Probe
- Packmaker will include all of the content of the *src* subdirectory in the modpack

With the modpack defined in the yaml file, next it needs to be locked, and then built.

Locking a pack, with the *packmaker lock* command, will resolve all of the information in the yaml file into actionable items. Primarily, mods will be resolved to be downloadable urls. A *packmaker.lock* file will be generated with all of the resolved information in it, and subsequent build steps will reference this lock file.

During the lock process, packmaker will also attempt to resolve any mod dependencies. If a mod defined in the pack has a dependency on another mod that is not also listed in the pack definition, packmaker will automatically resolve and add the dependent mod to the lock file for you. This way you do not need to worry about defining all of the dependent library mods often required in a pack. Packmaker can figure this out for you.

After the pack is locked, you can build it into an installable modpack, suitable for running in your launcher, or uploading to the Curseforge website. Packmaker supports and is capable of building the pack into a number of different formats.

To build a curseforge compatible pack, run *packmaker build-curseforge*. You can upload the generate zip file (*build/release/examplepack-1.0.0.zip*) to curseforge, or import it into the Cureforge client or PolyMC client directly.

To build a server, suitable for multiplayer play, run *packmaker build-server*. You'll find the server files in *build/server/*. There is even a simple *start.sh* script you can use to run the server.

A local single player version of the pack can also be generated with *packmaker build-local*, found in *build/local/*. For pack testing purposes, there is even a convenient way to launch the pack in offline mode, *packmaker launch*.

1.2 Reference

This section contains reference materials for various parts of packmaker. To get started with packmaker as a new user, though, you may want to read the *Getting Started* guide first.

1.2.1 Packmaker Definition Files

Minecraft modpacks built using Packmaker are defined in a YAML file, typically called `packmaker.yml`¹.

The `packmaker.yml` file defines all of the metadata about a modpack, title, version, authors, etc, and all of the details about what is needed to be installed in a Minecraft instance to create the modpack. This include things like what version of Minecraft, what version of Forge, and any mods and/or resourcepacks to add to the instance.

There is an simple example of a `packmaker.yml` for a simple modpack in the packmaker [git repository](#).

Introduction

Packmaker definition files are written in YAML syntax.

YAML is used because it is easier for humans to read and write than other common data formats like XML or JSON.

The definition file can be broken down into a number of different sections:

metadata All the pack metadata, including name, version, author, etc. Most of the metadata is required to properly define a modpack.

mods The list of mods to be installed in the modpack. Mods are optional, and if this section is omitted, no mods will be installed. (But what would a modpack be without mods?)

resourcepacks The list of resourcepacks to be installed in the modpack. Like mods, resourcepacks are optional, and if not are needed, this section can be omitted.

files The list of additional files to be installed in the modpack. Again, files are optional and can this section can be omitted if there are no files to install.

The following example packmaker definition file defines every single element, and every parameter for mods, resourcepacks, and files at least once.

```
---
name: examplepack
title: Example ModPack
version: 1.0.0
author:
  - mcrewson
  - crouth
minecraft: 1.12.2
```

(continues on next page)

¹ `packmaker.yml` is the default name used for the this yaml file, and this reference document assumes the default filename is used, but you can use any filename you like, as long as you provide it on the command line.

(continued from previous page)

```

forge: 14.23.5.2854

releasetypes:
  mods:
    - beta
    - release
  resourcepacks:
    - alpha
    - beta
    - release

mods:
  - jei:
  - journeymap:
    curseforgeid: 32274
    release: latest
  - the-one-probe
    ingoreddependencies:
      - mantle
  - tinkers-construct:
    release: TConstruct-1.12.2-2.13.0.183.jar
    clientonly: false
    serveronly: false
    optional: false
    recommendation: starred
    selected: false

resourcepacks:
  - faithful-x32:
    release: Faithful 1.15.2-r1
    optional: false
    recommendation: starred
    selected: false

files:
  - location: src
    clientonly: false
    serveronly: false

routhio:
  html:
    - location: html
  launch:
    flags:
      - "-Dfml.ignoreInvalidMinecraftCertificates=true -XX:+UseG1GC -XX:UseSSE=3"
  ...

```

Metadata

Modpack metadata is needed to properly identify the modpack

name The name of the modpack. Currently used as part of the final filename for the released modpack file. Typically this should be short and simple, with no spaces or punctuation in the name, for better filenames.

Required.

title The title of the modpack. This should be the full title of the pack. For example, the title is what is used in a curseforge manifest file, so when you import your into the curseforge client or polymc, this is the default name it will appear as.

Required.

version The version of the modpack. Can be any string, but would recommend your follow a consistent versioning pattern ([semantic versioning](#) is a good example). Incrementing the version of a pack is how launchers are able to detect and upgrade a modpack.

Version is also used in building the final filename for the released modpack file.

Required.

authors A list of names your want to credit with the creation of the modpack. If only one name is needed, you can specify this as a string, rather than a list.

Some modpack formats, the curseforge format in particular, do not support more than one author in their metadata. When building modpacks in those formats, packmaker only uses the first name in the list.

Required.

minecraft This is the version of minecraft this pack is being built for. Any valid version string that Mojang has defined for Minecraft should be valid here, although it is recommended you use some of the more common versions:

- 1.7.10
- 1.10.2
- 1.11.2
- 1.12.2
- 1.13.2
- 1.14.4
- 1.15.2

This field is required.

forge The version of the Forge modloader that the pack is being built for. Any valid version string that Forge has defined should be valid here, although it is recommended you use the latest for recommended version as defined by Forge. In fact, if you use the value 'recommended' or 'latest' for this version, packmaker will calculate the actual version based on what minecraftforge.net has defined for the version of minecraft you are using.

This is an optional field. If not defined, your modpack will not include Forge, most likely resulting in a vanilla minecraft pack, one with no mods loaded in it.

The forge and fabric modloaders should not both be defined in the pack definition. That would be crazy.

fabric The version of the Fabric modloader that the pack is being built for. Any valid version string the Fabric has defined should be valid here, although it is recommended you use the latest version as defined by Fabric. In fact, if you use the value 'latest' for this version, packmaker will calculate the actual version based on what fabricmc.net has defined for the version of minecraft you are using.

This is an optional field. If not defined, the Fabric modloader will not be included, most likely resulting in a vanilla minecraft pack, or at least one that cannot load Fabric mods.

The forge and fabric modloaders should not both be defined in the pack definition. That leads to madness.

Environment Variables in Metadata

Packdef metadata can be specified using environment variable syntax, and packmaker will use the values of the environment variable when building the pack. For example:

```
----
name: examplepack
title: Example ModPack
version: ${VERSION:-development}
```

The version of this pack will be set at build time to the value of the `VERSION` environment variable. If the `VERSION` environment variable is not defined, the value `development` will be used as a fallback.

Environment variables are only valid for the metadata items in the packdef files. They will not be interpolated in other parts of the packdef such as the `mods` or `resourcepacks` lists.

Release Types

Release types are a way to control the versions or releases of the mods and resourcepacks installed in the pack. Every version of a mod or resourcepack in Curseforge is classified by a release type. While the meaning of these release types can be a little vague, and ultimately is up to the mod authors, the three available release types typically mean the following:

- `alpha` : An alpha release is considered the least stable version of a mod. Testing may have not been completed and there may be game breaking bugs.
- `beta` : A beta release is the middle ground, more stable than alpha, less than release. More testing has been done, but there may be some smaller issues still to be worked out.
- `release` : The most stable of the release types, a “released” mod or resourcepack is has been deemed by the mod author as ready for everyone to use. Testing and most bugs should be worked out.

Typically, although not universally, and not guaranteed, a mod author will make many alpha releases of his project while developing his code. When he decides that it is ready for use by a larger audience, he may make a beta release or two, which fixes the bugs found by his alpha testers. Eventually, once the mod author is happy with the testing and is certain the mod or resourcepack is ready, he releases a final “release” version of his project.

As the modpack author, you can specify which types of releases you want to include in your pack. If you specify nothing, packmaker will find and include the most recent (the “latest”) version of the mods and resourcepacks specified in your pack definition file, regardless of their release type. If you do not want to include “alpha” and/or “beta” versions in your modpack, you can control that by specifying which release types are acceptable.

The `releasetypes` element in the pack definition can control this for all mods and resourcepacks in your pack:

```
releasetypes:
  mods:
    - alpha
    - beta
    - release
  resourcepacks:
    - alpha
    - beta
    - release
```

The allowed release types for mods and resourcepacks are specified separately (so you could allow “alpha” resourcepacks but not “alpha” mods). For each, you can specify a list of allowed release types. This list will be used for *ALL* of the mods or resourcepacks. You can override this default `releasetype` value for individual mod and resourcepacks by specifying a `release` parameter in the mod or resourcepack definition (see below).

It is allowed, although maybe a little odd, to specify unusual combinations of releasetypes, and packmaker will respect it. For example, you can specify that you want “alpha” and “release”, but not “beta” types, and packmaker will install the latest version of a mod that is either an alpha or a release, but not a beta (even if there is a newer version of the mod that was released as a beta).

Mod Definitions

`mods` is a list of minecraft mod definitions. Each definition requires the *slug name* of the mod, as found on the [curseforge.com](https://www.curseforge.com) site.

You can find the *slug name* of a mod using packmaker’s search function. Search results will contain both the full name of a mod and the *slug name*.

The *slug name* of a mod on curseforge can also be found in the mod’s address on the site. For example, the curseforge page for the [Tinker’s Construct](https://www.curseforge.com/minecraft/mods/tinkers-construct) mod is:

<https://www.curseforge.com/minecraft/mc-mods/tinkers-construct>

The last part of the address, “**tinkers-construct**”, is the *slug name* for this mod.

Each mod listed in this `packmaker.yml` file can be specified as either a string value, if no additional parameters are required for the mod, or as a dictionary, where you can supply additional parameters to control the specific version of the mod and other aspects of it within the modpack.

All of these parameters are optional.

curseforgeid An integer value, specifying the exact mod to be installed, based on its Curseforge ID in the curseforge api. When this parameter is used, packmaker will not search for the mod in curseforge using the slug name specified, but instead assume the id specified here is exactly the one wanted, and no further checks will be done to verify it.

This can be useful in some situations, where the curseforge search api is not returning the correct mod or cannot find it at all, but should be used as a last resort it will circumvent some of packmaker’s safety checks. For example, it will not ensure the mod is compatible with the chosen modloader or minecraft version.

Optional, and undefined by default. Meaning, the slug name of the mod specified will be used to find the requested mod in curseforge.

release The specific release of the mod to be installed into the modpack. If not defined, packmaker will default to using the latest version of the mod available for the version of minecraft that the pack is built for.

The `release` parameter must contain either the entire filename of the version of the mod desired, usually as specified on the curseforge site for the mod, or the curseforge id of the release. The curseforge id of the release can be useful if the curseforge search api is not find the exact mod you want.

Alternatively, one of the following “magic” values can be used to have packmaker figure out which version be installed, based on the release types specified by the curseforge api:

- `latest-alpha` : install the latest version of the mod with a release type of “alpha”, “beta”, or “released”
- `latest-beta` : install the latest version of the mod with a release type “beta” or “released”. “Alpha” releases will be ignored.
- `latest-release` : install the latest version of the mod with a release type of “released”. “Alpha” and “releases” will be ignored.

- `latest` : install the latest version of the mod with release type as specified in `releasetypes` packdef element.

Using these “magic” release values overrides the `releasetypes` specified globally (see above) for this mod.

Optional, and defaults to `latest`, meaning the version of the mod installed will be the latest available, as allowed by the `packmaker.conf` configuration file.

url To include a mod that cannot be found on Curseforge in a modpack, there are two options. You can include it directly in the pack as a file (see below), which works great but will greatly increase the size of your modpack and potentially create a licensing problem from mod owners that prefer their mods only be installed from their site. The alternative is to specify a `url` for the mod.

If the `url` parameter is used, it must be a complete and accurate download url for the mod jar file. Packmaker will then include only this url in your modpack (except for curseforge modpacks, see the next paragraph for that caveat), and a client launcher will be able to download the mod for the player directly from this url. This means the modpack doesn’t need to contain the large mod jar file, and the mod owner gets the download directly from their site.

Caveat, Curseforge modpacks do not have a concept of downloading a mod from just a url, so packmaker will download and include the actual mod jar file into your curseforge modpack when building it. So the mod can still be defined in the packdef file with this url parameter, and packmaker will build it properly for all pack types.

It should be noted that when the `url` parameter is used, all release and version checking that packmaker normally does to ensure a mod is compatible with your modloader and minecraft is ignored.

clientonly A boolean value, indicating whether this mod is specific to the minecraft client. If true, packmaker will not install this mod into any server builds.

Optional, and defaults to false if not specified, meaning the mod will be installed in both client and server builds.

serveronly A boolean value, indicating whether this mod is specific to the minecraft server. If true, packmaker will only install this mod in server builds.

Optional, and default to false if not specified, meaning the mod will be installed in both client and server builds.

ignoreddependencies Either the string “all”, or a list of mods. This parameter specifies mods to be ignored when trying to resolve the dependencies of a mod. If the value of this parameter is `all`, no mod dependency resolution will be done for this mod. Otherwise if a mod’s dependency is part of this list, it will not be added to the set of mods in the pack as part of the mod dependency resolution process.

Optional, and defaults to an empty list, meaning all dependent mods will be added to the pack automatically.

optional An optional parameter, only used for routhio builds. This will mark the mod as optional, and the routhio launcher will present it as an optional mod that can be installed but is not required to launch the modpack.

When building servers and other types of modpacks, for example curseforge modpacks, packmaker will ignore this parameter.

recommendation An optional parameter, only used for routhio builds. This will mark an optional mod as recommended, marking the mod in the launcher as a recommended optional mod or an optional mod to avoid. Should have the value *starred* or *avoid*. Any other value will not make sense to the routhio launcher and will cause the mod to be ignored. Not including this parameter on an optional mod will make the launcher neither recommend to install nor avoid it.

When building servers and other types of modpacks, for example curseforge modpacks, packmaker will ignore this parameter.

selected An optional parameter, only used for routhio builds. This will mark an optional mod as selected, making the optional mod installed by default, unless the user explicitly chooses not to. Should have the value a boolean value, true or false.

When building servers and other types of modpacks, for example curseforge modpacks, packmaker will ignore this parameter.

Resourcepack Definitions

Resourcepacks are always minecraft client only elements. They will be ignored when building servers for the pack. So there is no need for `clientonly` or `serveronly` parameters for a resourcepack.

curseforgeid An integer value, specifying the exact resourcepack to be installed, based on its Curseforge ID in the curseforge api. When this parameter is used, packmaker will not search for the resourcepack in curseforge using the slug name specified, but instead assume the id specified here is exactly the one wanted, and no further checks will be done to verify it.

This can be useful in some situations, where the curseforge search api is not returning the correct resourcepack or cannot find it at all, but should be used as a last resort it will circumvent some of packmaker's safety checks. For example, it will not ensure the resourcepack is compatible with the chosen modloader or minecraft version.

Optional, and undefined by default. Meaning, the slug name of the resourcepack specified will be used to find the requested resourcepack in curseforge.

release The specific release of the resourcepack to be installed into the modpack. If not defined, packmaker will default to using the latest version of the resourcepack available for the version of minecraft that the pack is built for.

The `release` parameter must contain either the entire filename of the version of the resourcepack desired, usually as specified on the curseforge site for the resourcepack, or the curseforge id of the release. The curseforge id of the release can be useful if the curseforge search api is not find the exact resourcepack you want.

Alternatively, one of the following “magic” values can be used to have packmaker figure out which version be installed, based on the release types specified by the curseforge api:

- `latest-alpha` : install the latest version of the resourcepack with a release type of “alpha”, “beta”, or “released”
- `latest-beta` : install the latest version of the resourcepack with a release type “beta” or “released”. “Alpha” releases will be ignored.
- `latest-release` : install the latest version of the resourcepack with a release type of “released”. “Alpha” and “releases” will be ignored.
- `latest` : install the latest version of the resourcepack with release type as specified in `releasetypes` packdef element.

Using these “magic” release values overrides the `releasetypes` specified globally (see above) for this resourcepack.

Optional, and defaults to `latest`, meaning the version of the resourcepack installed will be the latest available, as allowed by the `packmaker.conf` configuration file.

The `release` parameter must contain the entire filename of the version of the resourcepack desired, usually as specified on the curseforge site for the resourcepack.

The value `latest` can also be used to explicitly specify that the latest version be installed.

url To include a resourcepack that cannot be found on Curseforge in a modpack, there are two options. You can include it directly in the pack as a file (see below), which works great but will greatly increase the size of your modpack and potentially create a licensing problem from resourcepack owners that prefer their resourcepack only be installed from their own site. The alternative is to specify a `url` for the resourcepack.

If the `url` parameter is used, it must be a complete and accurate download url for the resourcepack zip file. Packmaker will then include only this url in your modpack (except for curseforge modpacks, see the next paragraph for that caveat), and a client launcher will be able to download the resourcepack for the player directly from this url. This means the modpack doesn't need to contain the large resourcepack zip file, and the resourcepack owner gets the download directly from their site.

Caveat, Curseforge modpacks do not have a concept of downloading a resourcepack from just a url, so packmaker will download and include the actual resourcepack zip file into your curseforge modpack when building it. So the resourcepack can still be defined in the packdef file with this url parameter, and packmaker will build it properly for all pack types.

It should be noted that when the `url` parameter is used, all release and version checking that packmaker normally does to ensure a resourcepack is compatible with your modloader and minecraft is ignored.

optional An optional parameter, only used for routhio builds. This will mark the resourcepack as optional, and the routhio launcher will present it as an optional resourcepack that can be installed but is not required to launch the modpack.

When building other types of modpacks, for example curseforge modpacks, packmaker will ignore this parameter.

recommendation An optional parameter, only used for routhio builds. This will mark an optional resourcepack as recommended, marking the resourcepack in the launcher as a recommended optional resourcepack or an optional resourcepack to avoid. Should have the value *starred* or *avoid*. Any other value will not make sense to the routhio launcher and will cause the resourcepack to be ignored. Not including this parameter on an optional resourcepack will make the launcher neither recommend to install nor avoid it.

When building other types of modpacks, for example curseforge modpacks, packmaker will ignore this parameter.

selected An optional parameter, only used for routhio builds. This will mark an optional resourcepack as selected, making the optional resourcepack installed by default, unless the user explicitly chooses not to. Should have the value a boolean value, true or false.

When building other types of modpacks, for example curseforge modpacks, packmaker will ignore this parameter.

Files

The `files` section of the `packmaker.yml` file specifies the local files to be included into the modpack. Typically these would be an additional configurations, scripts, resources, etc that you want to add to the minecraft install when it is installed.

This section is a list of file locations, typically subfolders or directories within your packmaker project. Each location in the list should be unique and not overlap another location, as all locations will be combined in the final product.

The contents of each location folder will be copied into the minecraft folder of the built instance, with no alterations or attempt to preserve what is already stored in the destination, so be care not to accidentally overwrite any files that may

have already been installed by minecraft itself or mod installations.

A typical use for multiple file locations is to specify some files be installed only on client builds, and other only installed on server builds. For example:

```
files:
- location: src-all
- location: src-client
  clientonly: true
- location: src-server
  serveronly: true
- location: templates
  template: true
```

Each location in the files list can have the following parameters.

location The location, relative to the `packmaker.yml` file itself, of the local files to be included in the modpack.

If this location does not physically exist when building the modpack, a warning is printing, by this location will be ignored.

clientonly An optional boolean flag (true or false), indicating that this location of files should only be included with client builds.

Defaults to false, meaning the location will be included in both client and server build, unless the corresponding `serveronly` parameter is specified.

serveronly An optional boolean flag (true or false), indicating that this location of files should only be included with server builds.

Defaults to false, meaning the location will be included in both client and server build, unless the corresponding `clientonly` parameter is specified.

template An optional boolean flag (true or false), indicating that this location of files are Jinja2 templates. Rather than simply copying files from this location to the instance destination, Packmaker will load and render these files using the Jinja2 templating system. The packdef metadata (name, title, version, minecraft_version, etc) can be used as template variables inside each template file however you want.

This can be useful for rendering pack data, for example the pack title and version, into mod configuration files, such as custom main menu text.

Defaults to false, meaning the files at this location are not templates, but plain files that will be copied to an instance with no modifications.

Routhio

The `routhio` section of the `packmaker.yml` file is specific to only one type of modpack build, a modpack for the routhio launcher. All other build types will ignore this section.

This section includes the following elements:

html A location of additional html files, typically a local folder with the project, that will be displayed within the launcher when a user selected this specific modpack.

launch Additional options the launcher will use when launching a minecraft instance. Currently, the only launcher option that can be specified here is `flags`, which is used to provide additional java command line arguments on the minecraft process.

1.2.2 Command-Line Interface

Commands

build-curseforge

```
packmaker build-curseforge [-h] [--build-dir BUILD_DIR]
                             [--release-dir RELEASE_DIR]
                             [--cache-dir CACHE_DIR]
                             [--release-format {zip,tgz}]
                             [lockfile [lockfile ...]]
```

Build a curseforge compatible modpack.

positional arguments:

lockfile modpack lock file

optional arguments:

-h, --help show this help message **and** exit
 --build-dir BUILD_DIR, -b BUILD_DIR base directory **for** build artifacts
 --release-dir RELEASE_DIR, -r RELEASE_DIR base directory **for** release artifacts
 --cache-dir CACHE_DIR base directory **for** cached artifacts
 --release-format {zip,tgz} archive **format for** release package

Describe cmd here

Optional command flags:

build-local

```
packmaker build-local [-h] [--build-dir BUILD_DIR]
                        [--release-dir RELEASE_DIR]
                        [--cache-dir CACHE_DIR]
                        [--release-format {zip,tgz}]
                        [lockfile [lockfile ...]]
```

Build a local installation

positional arguments:

lockfile modpack lock file

optional arguments:

-h, --help show this help message **and** exit
 --build-dir BUILD_DIR, -b BUILD_DIR base directory **for** build artifacts
 --release-dir RELEASE_DIR, -r RELEASE_DIR base directory **for** release artifacts
 --cache-dir CACHE_DIR base directory **for** cached artifacts
 --release-format {zip,tgz} archive **format for** release package

Describe cmd here

Optional command flags:

build-routhio

```
packmaker build-routhio [-h] [--build-dir BUILD_DIR]
                        [--release-dir RELEASE_DIR]
                        [--cache-dir CACHE_DIR]
                        [--release-format {zip,tgz}]
                        [lockfile [lockfile ...]]
```

Build a Routh.io compatible modpack.

positional arguments:

lockfile	modpack lock file
----------	-------------------

optional arguments:

-h, --help	show this help message and exit
--build-dir BUILD_DIR, -b BUILD_DIR	base directory for build artifacts
--release-dir RELEASE_DIR, -r RELEASE_DIR	base directory for release artifacts
--cache-dir CACHE_DIR	base directory for cached artifacts
--release-format {zip,tgz}	archive format for release package

Describe cmd here

Optional command flags:

build-server

```
packmaker build-server [-h] [--build-dir BUILD_DIR]
                        [--release-dir RELEASE_DIR]
                        [--cache-dir CACHE_DIR]
                        [--release-format {zip,tgz}]
                        [lockfile [lockfile ...]]
```

Build a server modpack.

positional arguments:

lockfile	modpack lock file
----------	-------------------

optional arguments:

-h, --help	show this help message and exit
--build-dir BUILD_DIR, -b BUILD_DIR	base directory for build artifacts
--release-dir RELEASE_DIR, -r RELEASE_DIR	base directory for release artifacts
--cache-dir CACHE_DIR	base directory for cached artifacts
--release-format {zip,tgz}	archive format for release package

Describe cmd here

Optional command flags:

convert

```
packmaker convert [-h] manifest [packdef]
```

Convert a curseforge modpack manifest json file to a packmaker yaml packdef file

positional arguments:

```
manifest    manifest json file
packdef      output packdef file
```

optional arguments:

```
-h, --help  show this help message and exit
```

Describe cmd here

Optional command flags:

findupdates

```
packmaker findupdates [-h] [-f {csv,json,table,value,yaml}] [-c COLUMNS]
                        [--max-width <integer>] [--fit-width]
                        [--print-empty]
                        [--quote {all,minimal,none,nonnumeric}]
                        [--noindent] [--sort-column SORT_COLUMN]
                        [lockfile [lockfile ...]]
```

Search curseforge **for** newer/updated versions of addons.

positional arguments:

```
lockfile          modpack lock file
```

optional arguments:

```
-h, --help          show this help message and exit
```

output formatter:

```
output formatter options
```

```
-f {csv,json,table,value,yaml}, --format {csv,json,table,value,yaml}
                                the output format, defaults to table
```

```
-c COLUMNS, --column COLUMNS
                                specify the column(s) to include, can be repeated
```

```
--sort-column SORT_COLUMN
                                specify the column(s) to sort the data (columns
                                specified first have a priority, non-existing columns
                                are ignored), can be repeated
```

Table formatter:

```
--max-width <integer>
                                maximum display width, <1 to disable. You cal also use
                                the MAX_DISPLAY_WIDTH environment variable, but the
                                parameter takes precedence.
```

(continues on next page)

(continued from previous page)

<code>--fit-width</code>	Fit the table to the display width. Implied if <code>--max-width</code> greater than 0. Set the environment variable <code>FIT_WIDTH=1</code> to always enable
<code>--print-empty</code>	Print empty table iof there is no data to show.
CSV formatter:	
<code>--quote {all,minimal,none,nonnumeric}</code>	when to include quotes, default to nonnumeric
json formatter:	
<code>--noindent</code>	whether to disable indenting the JSON

Describe cmd here

Optional command flags:

info

```
packmaker info [-h] [-f {json,shell,table,value,yaml}] [-c COLUMNS]
               [--max-width <integer>] [--fit-width] [--print-empty]
               [--noindent] [--prefix PREFIX]
               [lockfile [lockfile ...]]
```

Display information about the modpack.

positional arguments:

- lockfile modpack lock file

optional arguments:

- h, --help show this help message **and** exit

output formatter:

- output formatter options

- f {json,shell,table,value,yaml}, --format {json,shell,table,value,yaml} the output **format**, defaults to table
- c COLUMNS, --column COLUMNS specify the column(s) to include, can be repeated

Table formatter:

- `--max-width <integer>` maximum display width, <1 to disable. You cal also use the `MAX_DISPLAY_WIDTH` environment variable, but the parameter takes precedence.
- `--fit-width` Fit the table to the display width. Implied **if** `--max-width` greater than 0. Set the environment variable `FIT_WIDTH=1` to always enable
- `--print-empty` Print empty table iof there **is** no data to show.

json formatter:

- `--noindent` whether to disable indenting the JSON

shell formatter:

- a **format** a UNIX shell can parse (variable="**value**")
- `--prefix PREFIX` add a prefix to **all** variable names

Describe cmd here

Optional command flags:

launch

```
packmaker launch [-h] [--build-dir BUILD_DIR]
                  [--release-dir RELEASE_DIR] [--cache-dir CACHE_DIR]
                  [--release-format {zip,tgz}]
                  [lockfile [lockfile ...]]
```

Launch a local installation

positional arguments:

lockfile modpack lock file

optional arguments:

-h, --help show this help message **and** exit
 --build-dir BUILD_DIR, -b BUILD_DIR
 base directory **for** build artifacts
 --release-dir RELEASE_DIR, -r RELEASE_DIR
 base directory **for** release artifacts
 --cache-dir CACHE_DIR
 base directory **for** cached artifacts
 --release-format {zip,tgz}
 archive **format for** release package

Describe cmd here

Optional command flags:

lock

```
packmaker lock [-h] [--strict] [--no-strict] [packdef [packdef ...]]
```

Lock the modpack. Find mod download urls, generate a packmaker.lock file.

positional arguments:

packdef modpack definition file

optional arguments:

-h, --help show this help message **and** exit
 --strict enforce strict minecraft version matching
 --no-strict do **not** enforce strict minecraft version matching [default]

Describe cmd here

Optional command flags:

modsinfo

```
packmaker modsinfo [-h] [-f {csv,json,table,value,yaml}] [-c COLUMNS]
                   [--max-width <integer>] [--fit-width]
                   [--print-empty]
```

(continues on next page)

(continued from previous page)

```

        [--quote {all,minimal,none,nonnumeric}] [--noindent]
        [--sort-column SORT_COLUMN]
        [lockfile [lockfile ...]]

Display information about the mods in the modpack.

positional arguments:
  lockfile                modpack lock file

optional arguments:
  -h, --help              show this help message and exit

output formatter:
  output formatter options

  -f {csv,json,table,value,yaml}, --format {csv,json,table,value,yaml}
                                the output format, defaults to table
  -c COLUMNS, --column COLUMNS
                                specify the column(s) to include, can be repeated
  --sort-column SORT_COLUMN
                                specify the column(s) to sort the data (columns
                                specified first have a priority, non-existing columns
                                are ignored), can be repeated

Table formatter:
  --max-width <integer>
                                maximum display width, <1 to disable. You cal also use
                                the MAX_DISPLAY_WIDTH environment variable, but the
                                parameter takes precedence.
  --fit-width              Fit the table to the display width. Implied if --max-
                                width greater than 0. Set the environment variable
                                FIT_WIDTH=1 to always enable
  --print-empty            Print empty table iof there is no data to show.

CSV formatter:
  --quote {all,minimal,none,nonnumeric}
                                when to include quotes, default to nonnumeric

json formatter:
  --noindent               whether to disable indenting the JSON

```

resourcepacksinfo

```

packmaker resourcepacksinfo [-h] [-f {csv,json,table,value,yaml}]
                             [-c COLUMNS]
                             [--max-width <integer>]
                             [--fit-width]
                             [--print-empty]
                             [--quote {all,minimal,none,nonnumeric}]
                             [--noindent]
                             [--sort-column SORT_COLUMN]
                             [lockfile [lockfile ...]]

Display information about the resourcepacks in the modpack.

```

(continues on next page)

(continued from previous page)

```

positional arguments:
  lockfile          modpack lock file

optional arguments:
  -h, --help        show this help message and exit

output formatter:
  output formatter options

  -f {csv,json,table,value,yaml}, --format {csv,json,table,value,yaml}
                                the output format, defaults to table
  -c COLUMNS, --column COLUMNS
                                specify the column(s) to include, can be repeated
  --sort-column SORT_COLUMN
                                specify the column(s) to sort the data (columns specified
→first have a priority, non-existing columns are ignored), can be repeated

Table formatter:
  --max-width <integer>
                                maximum display width, <1 to disable. You cal also use the
→MAX_DISPLAY_WIDTH environment variable, but the parameter takes precedence.
  --fit-width
                                Fit the table to the display width. Implied if --max-width
→greater than 0. Set the environment variable FIT_WIDTH=1 to always enable
  --print-empty
                                Print empty table iof there is no data to show.

CSV formatter:
  --quote {all,minimal,none,nonnumeric}
                                when to include quotes, default to nonnumeric

json formatter:
  --noindent
                                whether to disable indenting the JSON

```

Describe cmd here

Optional command flags:

search

```

packmaker search [-h] [-f {csv,json,table,value,yaml}] [-c COLUMNS]
                  [--max-width <integer>] [--fit-width] [--print-empty]
                  [--quote {all,minimal,none,nonnumeric}] [--noindent]
                  [--sort-column SORT_COLUMN]
                  searchstring [searchstring ...]

```

Search curseforge **for** mods.

```

positional arguments:
  searchstring      search string

optional arguments:
  -h, --help        show this help message and exit

output formatter:
  output formatter options

```

(continues on next page)

(continued from previous page)

```

-f {csv,json,table,value,yaml}, --format {csv,json,table,value,yaml}
    the output format, defaults to table
-c COLUMNS, --column COLUMNS
    specify the column(s) to include, can be repeated
--sort-column SORT_COLUMN
    specify the column(s) to sort the data (columns
    specified first have a priority, non-existing columns
    are ignored), can be repeated

Table formatter:
  --max-width <integer>
    maximum display width, <1 to disable. You can also use
    the MAX_DISPLAY_WIDTH environment variable, but the
    parameter takes precedence.
  --fit-width
    Fit the table to the display width. Implied if --max-
    width greater than 0. Set the environment variable
    FIT_WIDTH=1 to always enable
  --print-empty
    Print empty table if there is no data to show.

CSV formatter:
  --quote {all,minimal,none,nonnumeric}
    when to include quotes, default to nonnumeric

json formatter:
  --noindent
    whether to disable indenting the JSON

```

Describe cmd here

Optional command flags:

updatedb

```

packmaker updatedb [-h] [--ignore-mods] [--ignore-resourcepacks]

Download and compile a new mods database from curseforge.

optional arguments:
  -h, --help            show this help message and exit
  --ignore-mods         Do not scan for mods when updating the curseforge db
  --ignore-resourcepacks
                        Do not scan for resourcepacks when updating the curseforge db

```

Describe cmd here

Optional command flags:

help

```

packmaker help [-h] cmd

Print detailed help for another command

positional arguments:

```

(continues on next page)

(continued from previous page)

```
cmd          name of the command

optional arguments:
  -h, --help  show this help message and exit
```

Describe cmd here

Optional command flags:

Global Flags

Packmaker has a few “global” flags that affect all commands. These must appear between the executable name (packmaker) and the command—for example, `packmaker -v search ...`.

1.2.3 Configuration

Packmaker has a simple configuration file that lets you control and customize its operation. To configure packmaker, you create a file called `packmaker.conf`. The location of this file depends on your platform:

- On Linux and other Unix-like Oses, packmaker looks for `~/.config/packmaker.conf`.
- On Windows, packmaker uses `%APPDATA%\packmaker.conf`. This is usually a folder like `C:\Users\You\AppData\Roaming`.
- On Mac OS X, you can use either the unix location or `~/Library/Application Support/config.yml`.

It possible to customize the location of the configuration file. See the *Configuration Location*, below.

The config file uses a common INI syntax, with sections of the configuration delimited by `[section]` headers, and simple `key = value` parameters in each section.

You can include comments in the configuration file. Comments start with a `#` (hash), and packmaker will ignore the comment until the end of the line.

You can also include environment variables in the values for most options, using the `${VARIABLE}` syntax. Packmaker will attempt to resolve these definitions for the environment.

The rest of this page enumerates all of the configuration options available.

Curseforge options

This section of the configuration controls how the Curseforge api is accessed and cached.

authentication_token

The curseforge api authentication token. This token is required to allow packmaker access to the api. Most api calls will fail without it, so packmaker will simply refuse to run most commands if it is not defined.

You can obtain an authentication token with a curseforge.com account, using their [Curseforge Console](#) site.

ignore_mods

A boolean value, when true tells packmaker to not include any mods when it is generating the local moddb file, using the `updatedb` command. This is an optimization if you know you will not be dealing with mods in your modpack, and will speed up the `updatedb` operation.

This is optional, and defaults to `false` when not defined. Mods will be included.

ignore_resourcepacks

A boolean value, when true tells packmaker to not include any resourcepacks when it is generating the local moddb file, using the `updatedb` command. This is an optimization if you know you will not be dealing with resourcepacks in your modpack, and will speed up the `updatedb` operation.

This is optional, and defaults to `false` when not defined. Resourcepacks will be included.

moddb_filename

The location and filename of the local cache of api information about mods and resourcepacks from curseforge. Packmaker uses this 'moddb' as a way to speed up many api calls, looking up addon information from this file rather than calling curseforge. The `updatedb` command is used to generate and maintain this file.

This file is optional. If you do not define it packmaker will still work, but it will be forced to make many more curseforge api calls which can significantly slow down its operation.

HINT: If you specify a common location for this file, it can be reused for multiple packs. For example:
`moddb_filename = ${HOME}/packmaker/addons.db.`

Launch options

This section specifies parameters related to Java. You can control specifically how java is called when launching minecraft using the `launch` command, and when building minecraft servers.

java_arguments

The java command line arguments to use when launching Minecraft. If you have unique java requirements, you specify them here.

This is optional, and defaults to the following set of java arguments when not defined:

```
-XX:+UseG1GC -XX:+UnlockExperimentalVMOptions +XX:G1NewSizePercent=20  
+XX:G1ReservePercent=20 -XX:MaxGCPauseMillis=50 -XX:G1HeapRegionSize=32M -Dsun.  
rmi.dgc.server.gcInterval=2147483646 -Dfml.readTimeout=180
```

java_binary

The location of the actual Java executable to use when launching Minecraft. If java is not on your `PATH`, or you need to use the non-default version of java, you can specify it here.

This is optional, and defaults to just `java` when not defined.

java_memory

How much memory to assign to the java heap space when launching Minecraft. This game, especially when modded, needs more memory to run than allocated by default by java itself.

This is optional, and default to 4096 megabytes (4 gigabytes) when not defined.

Locations options

This section defines the locations, folders or directories, that packmaker will use when building your modpack.

build

This is the root location of all build artifacts, although some can be changed using the location configurations below.

This is optional, and defaults to a directory named `build` in the current working directory.

Note that if not otherwise defined, both the cache and release locations are configured as subdirectories of the build location.

cache

This location of files downloaded during the build command. Minecraft files, Java libraries, mods and resourcepacks are all downloaded to this cache location first, before being copied into their final location for the modpack.

This is for optimization, to reduce multiple downloads of the same file for subsequent builds. When packmaker needs to download a file, it will first check this cache location, and use the cached file rather than downloading another copy. This greatly speeds up modpack building.

This is optional, and default to a directory named `cache` under the build location.

HINT: you can share this cache location between multiple modpacks if you defined it to be in a common location. Then you will only ever download a version of minecraft once, and reuse it for every modpack. For example: `cache = ${HOME}/packmaker/cache`.

release

The location of the final release artifacts built by packmaker. After a build command is completed you will be able to find the zip file for your modpack in this location.

This is optional, and defaults to a directory named `release` under the build location.

Configuration Locations

Packmaker uses a specific search order to find its configuration file on the filesystem. This search order is dependent on the platform it is running on.

Linux or Unix-like systems

On Linux and Unix, packmaker will look in the following locations for its configuration file, in this order, using the first one it finds.

- `./packmaker.conf`
- `${XDG_CONFIG_HOME}/packmaker.conf`, if the `XDG_CONFIG_HOME` environment variable is defined
- `~/.config/packmaker.conf`
- `/usr/local/etc/packmaker.conf`
- `/etc/packmaker.conf`

Windows

On Windows, packmaker will look in the following locations for its configuration file, in this order, using the first one it finds.

- `.\packmaker.conf`
- `~\AppData\Roaming\packmaker.conf`
- `%APPDATA%\packmaker.conf`, if the `APPDATA` environment variable is defined

Mac OS X

On Mac OS X, packmaker will look in the following locations for its configuration file, in this order, using the first one it finds.

- `./packmaker.conf`
- `${XDG_CONFIG_HOME}/packmaker.conf`, if the `XDG_CONFIG_HOME` environment variable is defined
- `~/.config/packmaker.conf`
- `~/Library/Application Support/packmaker.conf`
- `/usr/local/etc/packmaker.conf`
- `/etc/packmaker.conf`

Example

Here is an example configuration file, with all options defined:

```
# packmaker.conf

[curseforge]
authentication_token = xxxxxxxxxxxxxxxxxxxxxxxx
ignore_mods = false
ignore_resourcepacks = false
moddb_filename = ${HOME}/.local/share/packmaker/curseforge.db

[launch]
java_arguments = -XX:+UseG1GC -XX:UseSSE=3
java_binary = /usr/lib/jvm/java-8-openjdk-amd64/bin/java
java_memory = 8192m

[locations]
```

(continues on next page)

(continued from previous page)

```
build = ./build
cache = ${HOME}/.local/share/packmaker/cache
release = ./release
```

1.3 FAQ

Here are some answers to frequently-asked questions from IRC and elsewhere. Got a question that isn't answered here? Try the [discord server](#), or filing an issue in the bug tracker.

- *How do I...*
- *Why does packmaker...*

1.3.1 How do I...

tbd...

1.3.2 Why does packmaker...

tbd...