google-cloud Documentation

Release 2.4.4

Google LLC

Jul 25, 2020

Contents

Google API Artifact Manager (`artman`) is a program used to automate the generation and publishing of API client libraries.

In order to be consumed by artman, APIs require:

- A Protocol Buffers description of the API, specified using proto3 syntax.
- A service configuration stub. This is YAML configuration which designates how the API is housed within Google's infrastructure.
- A GAPIC configuration. This provides extra information specific to generating a client library.
- An artman configuration. This is the file artman uses as the entry point, and it points to the previous items in this list.

The artman tool is a wrapper around toolkit; it takes the configuration enumerated above, normalizes it, and sends it to toolkit, which generates a client library on disk, and then artman performs some concluding cleanup.

Client libraries produced in this way are executable "out of the box", and include basic reference documentation, and appropriate packaging and metadata files.

CHAPTER 1

Installation

1.1 Prerequisites

1.1.1 Docker

Because of the many, many things that must be on your system for artman to do its job, artman utilizes Docker and ships a container which has appropriate dependencies available. You will, therefore, need Docker to be on your system.

Note: It is also possible to run artman locally (use `--local`) when invoking it. This is generally only recommended for development of artman itself, but is an option. If you do this, use our Dockerfile as a reference to what dependencies you are likely to need.

1.1.2 googleapis

Currently, artman is primarily used for building Google Cloud client libraries, and depends on a large, interdependent configuration structure. This is housed in the *googleapis* repository.

In order to run artman, you will need to clone this repository, and generally you should be in this directory when invoking artman.

```
$ git clone git@github.com:googleapis/googleapis.git googleapis/
```

1.2 Installation

1. If you have not already, install pip and virtualenv. (Use `which pip` or `which virtualenv` to see if you already have them.)

1. Install Armin Ronacher's pipsi. This is a tool for installing scripts on your machine without touching system Python and without having to worry about the script's virutal environment. In other words, it is the correct tool for exactly this problem.

```
# This instruction is from the pipsi README; if you have trouble,
# double-check there.
$ curl https://raw.githubusercontent.com/mitsuhiko/pipsi/master/get-pipsi.py |␣
↪python
```

1. Install artman itself:

```
$ pipsi install googleapis-artman
```

This will place an executable spelled `artman` on your path. If you need to upgrade artman in the future, you can use `pipsi upgrade googleapis-artman` to do so.

1.3 Configuration

The artman tool requires some configuration in order to run (some of this is legacy from before artman was primarily run in Docker).

When you try to run `artman` the first time, it will complain and ask for a configuration file. Run `configure-artman` which will interactively walk you through the steps to create one, and then place it in the right spot.

The configuration tool will ask you for a "repository root". You should specify the directory *above* wherever you cloned googleapis to, and then you can use the defaults for everything else it asks.

This tool will be improved or removed in a future version.

CHAPTER 2

Getting Started

It is time to build a client library!

Using `artman` requires two concepts: (1) pointing to a artman configuration file, which tells artman what source material it is working off of, and (2) telling it what you want it to do.

Here is an example command:

```
# Right now, artman is location-aware; you need to be in the
# googleapis directory for everything to work correctly.
$ cd /path/to/googleapis/

# Build the language.v1 API library, and produce a Python client.
$ artman --config google/cloud/language/artman_language_v1.yaml \
    generate python_gapic
```

Unpacking that command:

- The `--config` switch (which is actually required) tells artman what inputs to use. In this case, the configuration is for the Natural Language API.

  Note: As of this writing, some artman YAML configs have a version at the end, and some do not. (We are moving in the direction of consistently having them.)

- `generate` is the command / verb. (artman also supports `publish`, which is able to, for example, automatically create a pull request on GitHub.)
- `python_gapic` is the objective – the thing you want to be produced. You essentially always want a GAPIC (an old code name for the complete, auto-generated API client), but seven languages are supported at this time; `ruby_gapic`, `java_gapic`, etc. will give you what you expect.

The package will be dropped in a newly created `artman-genfiles/` directory within your current working directory; the output of the command will tell you precisely where it put the library.

CHAPTER 3

Support

3.1 Python Versions

artman is currently tested with Python 2.7, Python 3.4, Python 3.5, and Python 3.6.

Note: The authors of this documentation are humans, and this is an exceptionally easy spot to end up out of date.

Our `nox.py` and `.circleci/config.yml` files are the real source of truth for what versions of Python we test against.

3.2 Versioning

This library follows Semantic Versioning

It is currently in major version zero (`0.y.z`), which means that anything may change at any time and the public API should not be considered stable.

3.3 Contributing

Contributions to this library are always welcome and highly encouraged.

See the CONTRIBUTING documentation for more information on how to get started.